SMART TRANSFORMER COMMUNICATION AND APPLICATION IN RURAL MICROGRID SETTINGS

Cornel Verster

Thesis presented in partial fulfilment of the requirements for the degree
Master of Engineering (Research) in the Faculty of Engineering
at Stellenbosch University

Supervisor: Dr. Johan Beukes
Department of Electrical & Electronic Engineering

March 2015

Stellenbosch University https://scholar.sun.ac.za

Declaration

By submitting this thesis electronically I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Copyright © 2015 Stellenbosch Univeristy
All rights reserved

Stellenbosch University https://scholar.sun.ac.za

Acknowledgements

I would like to express my utmost gratitude to my Lord and Saviour Jesus Christ for the motivation, inspiration, strength and ability to do the work contained in this thesis. All of the glory goes to Him, and I myself am deeply thankful for His love and His grace in my life. He is everything to me.

I would like to thank my study leader, Dr. Johan Beukes for his continual support and guidance during this project. Also for the expertise in various fields that he shared with and taught me.

I would like to thank my mother, Hanlie Verster, for her support, love and continual sacrifice.

I would like to thank my father, who passed away this year. It would have been great had you been able to be here when I graduate dad, but I am thankful to the Lord for what He allowed us to have while you were here. I love you, and you are an inspiration to me.

I would also like to thank all of my friends and family who have loved and supported me through my student years, including but not restricted to: Bernard and Danielle van der Veen, Sias le Roux, Julian Bunge, Amo O'Kennedy, Johan Kotze, Marinus van den Berg and Marinus Bosch.

Stellenbosch University https://scholar.sun.ac.za

Abstract

The Smart Grid is an initiative to make the existing utility grid more effective and efficient by making utility infrastructure smarter. The initiative affects all areas of the utility grid and all utility hardware.

Communication to utility hardware for monitoring and remote configuration is central to the smart grid vision. The focus of this project is the Smart Transformer, a distribution transformer with built-in intelligence and communication capabilities. Data acquisition and remote configuration hardware and software was developed and installed on a distribution transformer for application in deep rural areas. The solution included communication capabilities and adheres to industry standards.

The solution was tested and data acquisition and management were done using the OSIsoft PI System software. Field tests were performed to evaluate the effectiveness of the solution in a deep rural setting. It was found that the smart transformer can be effectively monitored, configured and controlled in a deep rural setting.

The smart transformer concept was investigated in a microgrid context. The potential of a smart transformer within a microgrid was explored and the smart transformer as a microgrid market-enabler was focussed on. A simulation was performed to evaluate the role of a smart transformer as a microgrid market-enabling device. It was found that the smart transformer has the potential to serve as a market-enabling device.

Key words

Smart Grid

Smart Transformer

DNP3

Markets

Microgrids

Stellenbosch University https://scholar.sun.ac.za

Abstrak

Die slim kragnetwerk is ‘n initiatief om die bestaande kragnetwerk meer effektief en doeltreffend te maak deur kragnetwerk infrastruktuur se intelligensie te vermeerder. Die initiatief beïnvloed alle aspekte van die kragnetwerk en kragnetwerk hardeware.

Kommunikasie met kragnetwerk hardeware vir moniteering en instelling oor ‘n afstand is sentraal aan die slim kragnetwerk visie. Die fokus van hierdie projek is die slim transformator, ‘n distribusie transformator met ingeboude intelligensie en kommunikasie vermoëns. Data verkryging en afstandelike instelling hardeware en sagteware was ontwikkel en installeer op ‘n distribusie transformator vir toepasing in diep-landelike gebiede. Die oplossing sluit kommunikasie vermoëns in en voldoen aan industrie standaarde.

Die oplossing was getoets en data verkryging en bestuur was geïmplementeer met gebruik van OSIsoft se PI Stelsel sagteware. Veldtoetse was gedoen om die effektiwiteit van die oplossing in diep-landelike gebiede te evalueer. Dit was gevind dat die slim transformator effektief gemoniteer, ingestel en beheer kan word in ‘n diep-landelike omgewing.

Die slim transformator konsep was ondersoek in ‘n mikro-kragnetwerk konteks. Die potensiaal van ‘n slim transformator binne ‘n mikro-kragnetwerk was verken en die vermoë van ‘n slim transformator om ‘n mark binne ‘n mikro-kragnetwerk in staat te stel was op gefokus.’n Simulasie was uitgevoer om die vermoë wat ‘n slim transformator het om ‘n mark binne ‘n mikro-kragnetwerk in staat te stel te evalueer. Dit was gevind dat ‘n slim transformator die vermoë het om ‘n mark binne ‘n mirko-kragnetwerk in staat the stel.

Sleutel woorde

Slim Kragnetwerk

Slim Transformator

DNP3

Markte

Mikro-kragnetwerk

Stellenbosch University https://scholar.sun.ac.za

Contents

Stellenbosch University https://scholar.sun.ac.za

Stellenbosch University https://scholar.sun.ac.za

Stellenbosch University https://scholar.sun.ac.za

Stellenbosch University https://scholar.sun.ac.za

List of tables

Stellenbosch University https://scholar.sun.ac.za

List of figures

Stellenbosch University https://scholar.sun.ac.za

Stellenbosch University https://scholar.sun.ac.za

Stellenbosch University https://scholar.sun.ac.za

List of acronyms and abbreviations

AC	Alternating Current
APN	Access Point Name
BB	Beaglebone Black
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
DC	Direct Current
DG	Distributed Generation
DNP3	Distributed Network Procotol v.3
FTP	File Transfer Protocol
GB	Gigabytes
GHz	Gigahertz
GPRS	General Packet Radio Service
GSM	Global Systems for Mobile communications
HTTP	Hypertext Transfer Protocol
IMEI	International Mobile Equipment Identity
IP	Internet Protocol
kbits	kilobits
kW	kilowatts
LCS	Level Crossing Sampling
MIB	Management Information Base
MP	Main Program
NTP	Network Time Protocol
PIN	Personal Identification Number
POC	Point of Connection
s	seconds
SDRAM	Synchronous Dynamic Random Access Memory
SG	Smart Grid
SNMP	Simple Network Monitoring Protocol
SRAPET	Self Regulated and Protected Electrification Transformer
SSH	Secure Shell
ST	Smart Transformer
STCD	Smart Transformer Communication Device
USB	Universal Serial Bus
VPN	Virtual Private Network
XML	eXtensible Markup Language

Stellenbosch University https://scholar.sun.ac.za

CHAPTER 1
INTRODUCTION

1.1 MOTIVATION

The main motivation of this study is to contribute toward effective data management in the Smart Grid (SG).

The area of focus of this study is the enabling of microgrids in the SG. The area of application for this study is low voltage networks, as they are currently the real-life network that most closely resembles the structure of a microgrid. The specific application focussed on is that of a smart transformer (ST) applied as a data aggregator and microgrid agent within a microgrid.

One of the main components of the SG is an integrated sensor network. The network consists of distributed sensor and actuator nodes that enable remote monitoring and control of the grid [1]. This study included the development of a device that improves the data management and communication capabilities of an existing electronic solid-state tap-changing transformer. The creation of a ST applicable in microgrid networks is the end-goal of this study.

The transformer chosen as a hardware foundation for this study is for application in deep-rural areas known as the Self-Regulated and Protected Electrification Transformer (SRAPET). For this reason, the device developed in this study to improve the transformer must be a robust, maintenance-free device. Data management techniques minimize bandwidth usage while maintaining good visibility of device and transformer operations. The device is developed to improve transformer intelligence, thus lessening the computational load on central data management systems.

1.2 BACKGROUND

A project was initiated by Eskom at Stellenbosch University to develop a distribution transformer for application in deep-rural environments. The project resulted in the development of the SRAPET, a distribution transformer with built-in intelligence for condition monitoring and asset control. In essence, the SRAPET is a solid-state tap-changing voltage regulator. Its primary function is to perform voltage regulation. Secondary functions include:

- Transformer fault detection.
- Transformer condition monitoring.
- Local logging of transformer readings and significant events.
- Remote communication capabilities via a GPRS modem integrated with the SRAPET controller board. This allows the device to be monitored and configured remotely. Logged measurement and event data can be downloaded from the device.

Stellenbosch University https://scholar.sun.ac.za

The SRAPET was successfully developed and tested in the field, with a number of units deployed for field tests [2].

The Smart Grid (SG) is an initiative to make the existing power grid more cooperative, responsive, and economic and improve the penetration of renewable energy sources [3]. An important aspect in the process of achieving this is obtaining greater visibility and situational awareness of the power grid as well as automating control of the grid as far as possible. This requires a sophisticated data retrieval and actuator network where data is gathered from a mass of internal and external sources, and effectively managed and analysed. Decisions are made from the data analyses and applied via the actuators.

Realization of the data retrieval and actuator network takes place by installing various sensing and actuating devices on power grid assets. The constant inflow of data allows operators to gain greater awareness of current grid operations. Data sensors and actuators are arranged in a hierarchical structure where data aggregators serve multiple sensing and actuating devices as data management systems and decision makers. Data aggregators in turn communicate with data servers, that perform further analyses on the data and visualize it for operators [1].

1.3 ESKOM STANDARD

In 2013 / 2014, Eskom released the Remote Device Communication Standard for Data Retrieval and Remote Access that all remote data reporting devices must adhere to [4]. This aimed to standardize the capabilities possessed by, and protocols used on data retrieval devices across the Eskom network. According to the standard, a key criterion for operational success is greater visibility of the power grid. The standard aims to ensure that data retrieval devices on the Eskom network provide sufficient visibility of the power grid to enable operational and non-operational outcomes that add value.

The main requirements of the standard applicable on the SRAPET are listed below:

- Distributed Network Protocol version 3 (DNP3) for operational and engineering data communication.
- Simple Network Management Protocol (SNMP) version 1 or higher for management of the communication link to the device.
- HyperText Transfer Protocol (HTTP) used to host a web interface for monitoring and changing configuration settings.
- File Transfer Protocol (FTP) for file transfer to and from the device.
- Network Time Protocol (NTP) version 3 for time synchronization.
- Internal clock with a resolution of 10 ms to time-stamp events.
- Implementation of authentication and 128-bit encryption algorithms.

Stellenbosch University https://scholar.sun.ac.za

- Physical Ethernet Port (100Base-T).

For the remainder of this thesis, the Eskom Remote Device Communication Standard for Data Retrieval and Remote Access standard will simply be referred to as the Eskom standard.

The SRAPET with functionality that allows it to meet the requirements of the Eskom standard is considered a ST. Thus, a ST consists of the SRAPET distribution transformer with added hardware and software that allows it to meet the Eskom standard.

1.4 AIMS AND OUTLINE OF THESIS

The main objectives are listed below.

- *Extend SRAPET functionality to meet the Eskom Remote Device Communication Standard for Data Retrieval and Remote Access*. This requires both hardware and software development to take place. A device is thus to be developed that interfaces with the SRAPET transformer to enable the desired functionality.
- *Effectively retrieve and manage data from a deep-rural distribution transformer*. The developed unit monitors the distribution transformer. The unit performs some data processing and communicates data to a data server. This creates a standalone system for data retrieval, management, analyses and visualization. The data server performs data storage, analyses and visualization.
- *Field-test the system*. Results from field-testing of the system in deep-rural areas will provide valuable insights into the effectiveness of the system.
- *Investigate the role of an ST in a microgrid.* Investigation includes the study of the microgrid concept and how the ST fits into the microgrid. Roles and objectives of an ST in a microgrid are discussed.
- *Investigate the ST as a microgrid market-enabling device.* The potential of the ST as a device that enables the effective operation of an electricity market within the microgrid is explored and simulated.

Stellenbosch University https://scholar.sun.ac.za

CHAPTER 2
SMART GRID

2.1 INTRODUCTION

The SG is an initiative to make the existing electricity grid more cooperative, responsive and economic and to improve the penetration of renewable energy sources [3]. The initiative aims to create a two-way flow of both energy and information from utility to customer and customer to utility. This vision is to be realised by the implementation of a sophisticated communication network on top of the electricity grid that enables sensing, processing and control functionality.

The monitoring of SG assets is vital to increase situational awareness of the power grid. This will require the implementation of data retrieval capabilities in SG assets. Valuable data is sent from SG assets to operators. An example of such an asset is the distribution transformer. Monitoring of distribution transformers provide benefits including life loss prediction, fault detection and analyses, over- and under-loading detection and transformer life time prediction [5].

New technologies such as distributed computing will play a major role in the realization of the SG.

This section gives an overview of the SG concept, investigates the importance of asset condition monitoring, explores the concept of a ST and investigates the relevance of an ST within a microgrid.

2.2 SMART GRID OVERVIEW

With the growing demand on the power grid and the introduction of new role-players such as distributed generation (DG) and electric vehicles, new challenges arise in managing the power grid. These challenges are of direct implication on the stability and manageability of the power grid. As new technologies emerge in the fields of data management and communication, opportunities arise to meet these challenges. These opportunities are redefining the way that we think about the power grid.

Progress towards a SG is a process of incremental steps rather than a once-off change. The process is dependent on factors such as the development of various communication technologies and the SG must be able to accommodate new technologies as they emerge. Incremental upgrading of the current power grid infrastructure takes place as new technologies are developed and found to be useful. With the process of moving towards a SG, new challenges arise that require innovate solutions. An example of one such challenge is that of cyber-security for SG communication networks. If a utility is to control assets in the field using a communication network, it must be secure.

Stellenbosch University https://scholar.sun.ac.za

SG objectives are applicable to all realms of the utility grid. The National Institute of Standards and Technology (NIST) in the USA have divided the SG into seven domains to create a conceptual overview of the SG. Table 2-1 gives a brief description of the seven domains.

Table 2-1 - Smart Grid Domains [3]

Domain	Actors	Applications
Bulk Generation	Generators of bulk quantities of electricity	Power generation, asset management.
Transmission	Actors that carry electricity over long distances	Monitoring and control systems to optimize transmission and keep it stable
Distribution	Actors distribute electricity to customers and gather electricity from customers (if customer generation is present)	Asset management, recording of power and asset measurements, asset management, substation automation
Customer	Consumers (and possible generators) of electricity	Home automation, solar and wind generation
Service Provider	Organizations that provide services to utility customers	Installation and maintenance of assets, billing, customer management
Operations	Managers of the movement of electricity on the grid	Network operations, grid monitoring, grid control, analysis of data, customer support
Markets	Participants and operators in electricity market	Retailing and trading, market management

The various domains of the SG each have a set of *actors,* which are usually systems, programs or devices that make decisions affecting that domain. Actors exchange information with one another. This happens both inside each individual domain and across different domains. *Applications* are tasks performed by actors [3].

It is predicted that making use of advanced communication techniques will greatly increase the reliability, security, interoperability and efficiency of the electrical grid [6]. A major goal of SG that aims to achieve this is to gain greater situational awareness of the electricity grid by increasing the visibility of grid operations. Creating communication infrastructure on various levels of the grid will enable this. These levels represent communication between various SG assets and to operational systems mainly located in central control centres. Both the physical and functional position of a SG asset determines to what level it belongs. An example of such a level, and the lowest hierarchical level, is the Home Area Network (HAN).

The HAN is a network of communicating devices contained within a customer house or building. These devices include various appliances that use communication technologies to achieve SG

Stellenbosch University https://scholar.sun.ac.za

goals. Devices communicate among one another in the HAN, and a Smart Meter (SM) communicates the electricity usage of the household to the utility and controls certain devices within the HAN. Usage information is first communicated to the next level in the hierarchical chain: the Neighbourhood Area Network (NAN). In the NAN, consumption data from a variety of households are collected, processed and communicated upwards in the hierarchical structure. This process continues until data reaches the operational centre.

Various levels of the SG communication hierarchy work together to create an effective neural network of sensors and actuators that can monitor and control the electricity grid. Data processing takes place at various stages in the SG communication hierarchy. The neural network created by sensors and actuators provide utilities with greater situational awareness of the power grid. The applications of various communication technologies are vast, and specific technologies are applicable to specific areas of the SG. Table 2-2 contains a brief overview of the various SG communication levels and lists some example members and communication technologies applicable on those levels.

Table 2-2 - Smart Grid Communication Levels

Level	Example Members	Example Technologies
Home Area Network (HAN)	PCs, fridges, pool pumps, thermostats	ZigBee, WiFi, OpenHAN
Neighborhood Area Network (NAN)	Modems, relays, access points, birdges	WiMAX, ADSL, Cellular, ANSI C12 protocols
Wide Area Network (WAN)	Routers, repeaters, ground stations, towers	Satellite / microwave, frame relay, IEC 61850, DNP3

The area of focus for this study is that of the microgrid. The microgrid is a component of the SG. The larger SG consists of multiple smaller microgrids. The communication layer applied to microgrids form a NAN.

2.3 SENSOR AND ACTUATOR NETWORKS

A Sensor and Actuator Network (SANET) is a network consisting of a large number of sensor and actuator nodes. A SANET monitors the operational characteristics of the grid, allowing for more effective prevention of outages and disturbances. Sensors measure various system parameters (the ST measures voltage levels, current levels, transformer temperatures etc.). As opposed to sensors, actuators are devices that take the information read by sensors and control sent by management systems as input and turns this information into actions, such as displaying the information read by sensors [3].

Stellenbosch University https://scholar.sun.ac.za

Because of the vast geographical spread and complexity of the electricity grid, sensing its operational characteristics using SANETs meets functionality and scalability issues. The SANET must be a well-organized entity, ensuring that the right data gets to the right people at the right time. This is done by organizing the sensors in the SANET effectively and creating cooperation between them. A proposed model is given in [7], organizing sensors in a cluster tree topology [3].

Sensor units are ordered in a hierarchical manner forming domains. Most domains contain a data aggregator: a device that collects data from sensors. The data aggregator processes the data and can make decisions based on the processing. In the case of a microgrid, the ST acts as a data aggregator for various microgrid agents. The data is processed on the ST and then communicated to a remote data server if found to be significant. Data servers located above the ST in the hierarchical tree can also be structured hierarchically, with data servers acting as data aggregators for other data servers.

Event-based reporting is the philosophy that data is only to be reported when significant events occur or when required by operators. The concept of event-based reporting is implemented in SANETs for two reasons:

- To minimize network traffic as only important data is reported.
- To distribute the computational load of the grid by applying intelligence in the form of significant data identification at various points in the utility grid.

Following this model, data is only reported when it is considered to be of significant priority. All other data is stored locally on data servers (possibly rolled out according to region, see Figure 2-1) but is not brought to the attention of management unless requested.

Stellenbosch University https://scholar.sun.ac.za

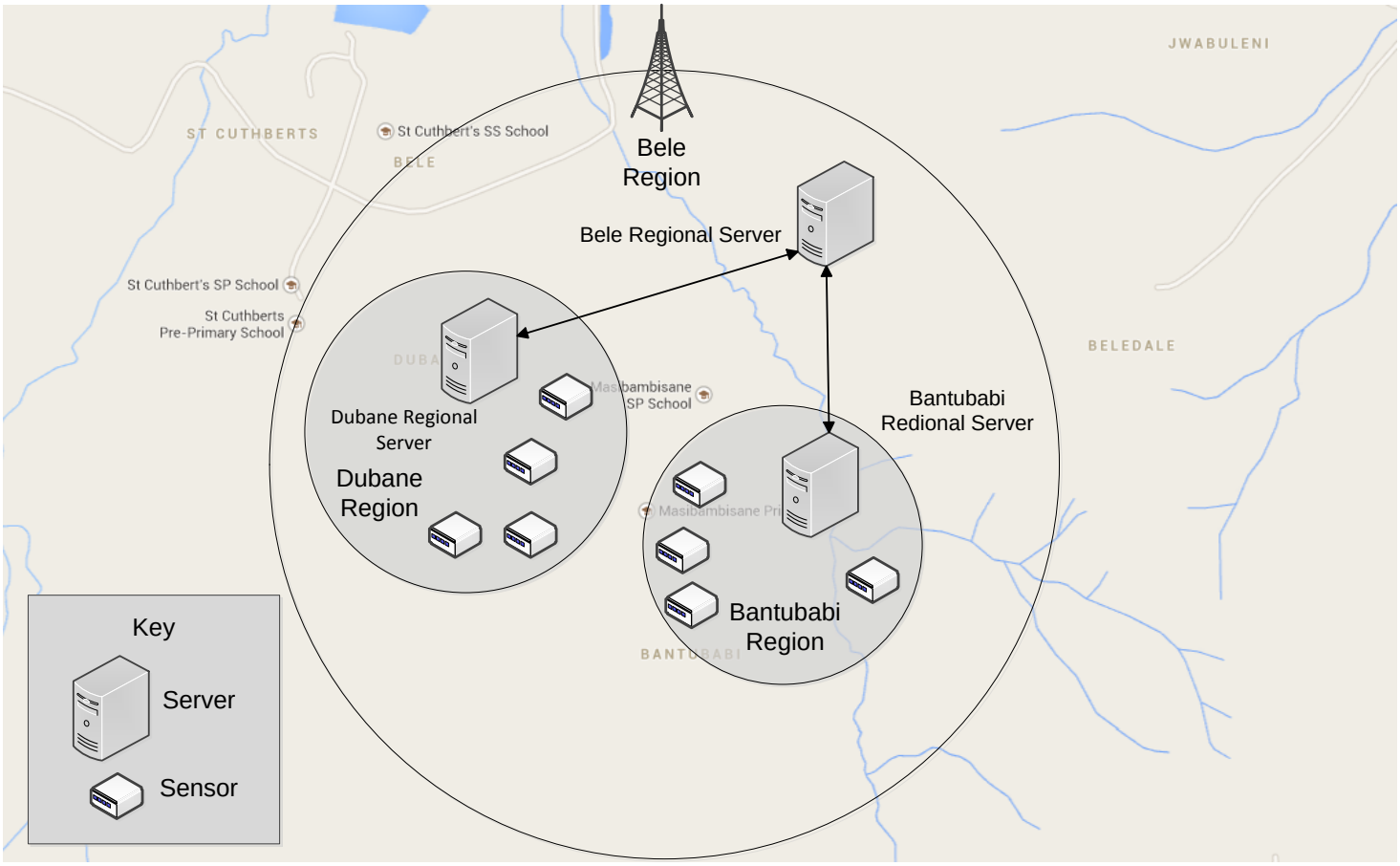


Figure 2-1 - Regional Data Servers

2.4 MICROGRIDS

The microgrid is considered the building block of the Smart Grid (SG). A microgrid is a localized grouping of connected electricity generation, loads and storage that form an entity. It can function both when connected and disconnected from the traditional utility grid (macrogrid). Table 2-3 contains a number of generation sources, loads and storage options commonly found in a microgrid [8].

Table 2-3 - Microgrid Role Players

Generation	Loads	Storage
Solar	Household appliances	Batteries
Wind	Machinery	Flywheels
Microhydropower	Electric vehicles	Supercapacitors
Diesel		
Induction generators		
Synchronous generators		
Fuel cells		

Stellenbosch University https://scholar.sun.ac.za

The microgrid is seen from the macrogrid as a single, controllable entity [9]. It has a Point of Connection (POC) at which it is connected to the macrogrid. The power utility company can disconnect this point from the macrogrid. When the POC is disconnected, it effectively *islands* the microgrid. Microgrids are designed to be able to continue functioning when it is islanded from the macrogrid [8].

The microgrid consists of all infrastructure necessary to facilitate its effective functioning as a mini-grid when connected to, or disconnected from the macrogrid. This includes electricity generation sources, loads, electricity storage, power lines, microgrid agents and a communication network that allows various agents within the microgrid to communicate.

Stability within the microgrid is essential to effective operation. Stability must be maintained when connected to, or disconnected, from the macrogrid.

Microgrid agents are devices placed at various power infrastructures throughout the microgrid. The role of a microgrid agent is three-fold: creating greater visibility of microgrid operations, enabling predictions of future operation and performing both manual and automated control in the microgrid [9].

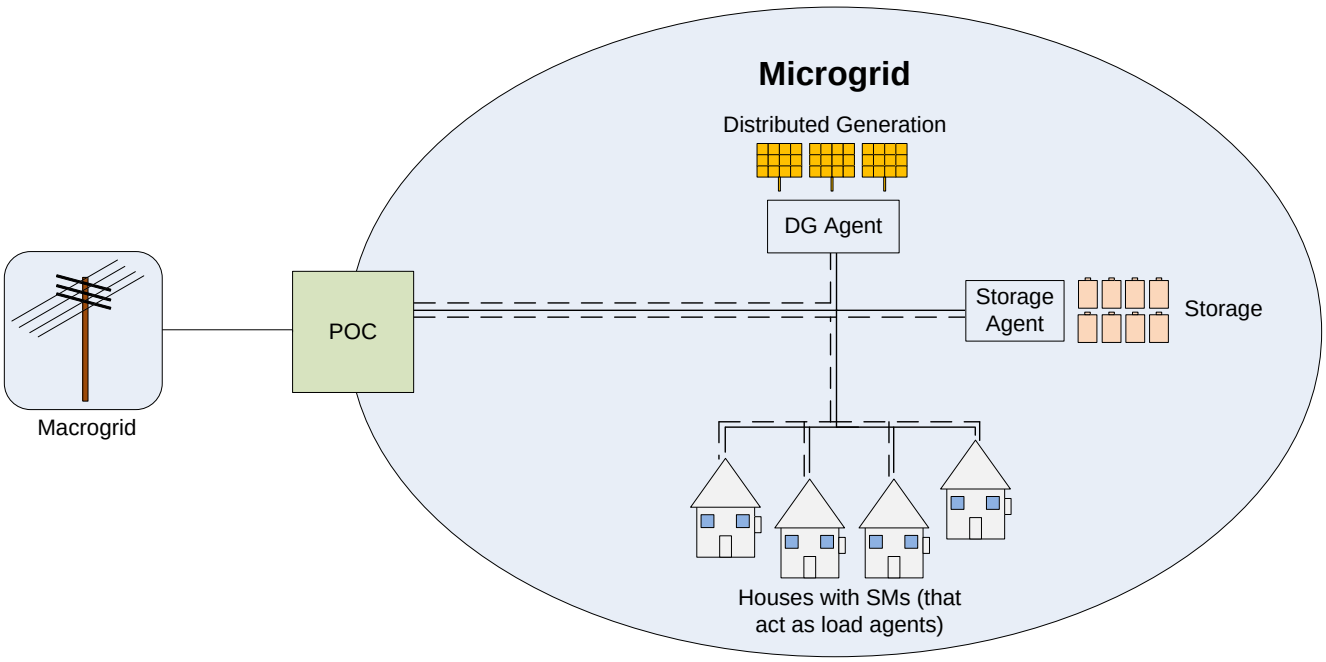


Figure 2-2 - Microgrid Structure

Microgrids can be either AC or DC, referring to the form of electricity transportation along the lines. The appropriate converters implemented at various positions in the microgrid accommodate the microgrid type. DC microgrids are currently less popular than AC microgrids as more AC loads are present on the network. As more DC loads emerge, DC microgrids will grow in popularity and research on such microgrids will enable their implementation [10].

Stellenbosch University https://scholar.sun.ac.za

The microgrid contains the necessary intelligence to facilitate power flow within itself to maintain stable operating conditions. This means balancing out power generation, usage and storage.

2.4.1 Microgrid Agents

With the introduction of unpredictable generation (renewable energy such as wind power) and dynamic loads (such as electric vehicles and other dynamic loads), a greater need arises to keep the grid stable. This means controlling line voltage of the grid by controlling electricity generation, loads and storage. Control of the line voltage happens at various points in the system.

A microgrid agent is a device that monitors and controls a microgrid component. An agent can sense operational changes of the equipment it is monitoring, effectively make decisions by taking the changes and user-defined parameters into account, and act autonomously based on the decisions made [8].

Various microgrid agents and a communication network that allows them to communicate enable local control of the microgrid voltage. Microgrids contain DG agents, load agents, storage agents and an agent located at the POC. These agents gather data from their respective microgrid components, and exercise control over these components. In total, all of the agents within the microgrid create a type of data nervous system [11].

In this project, the ST is suggested as a microgrid agent at the POC. The ST is a solid-state tap-changing transformer with built-in intelligence for monitoring and control.

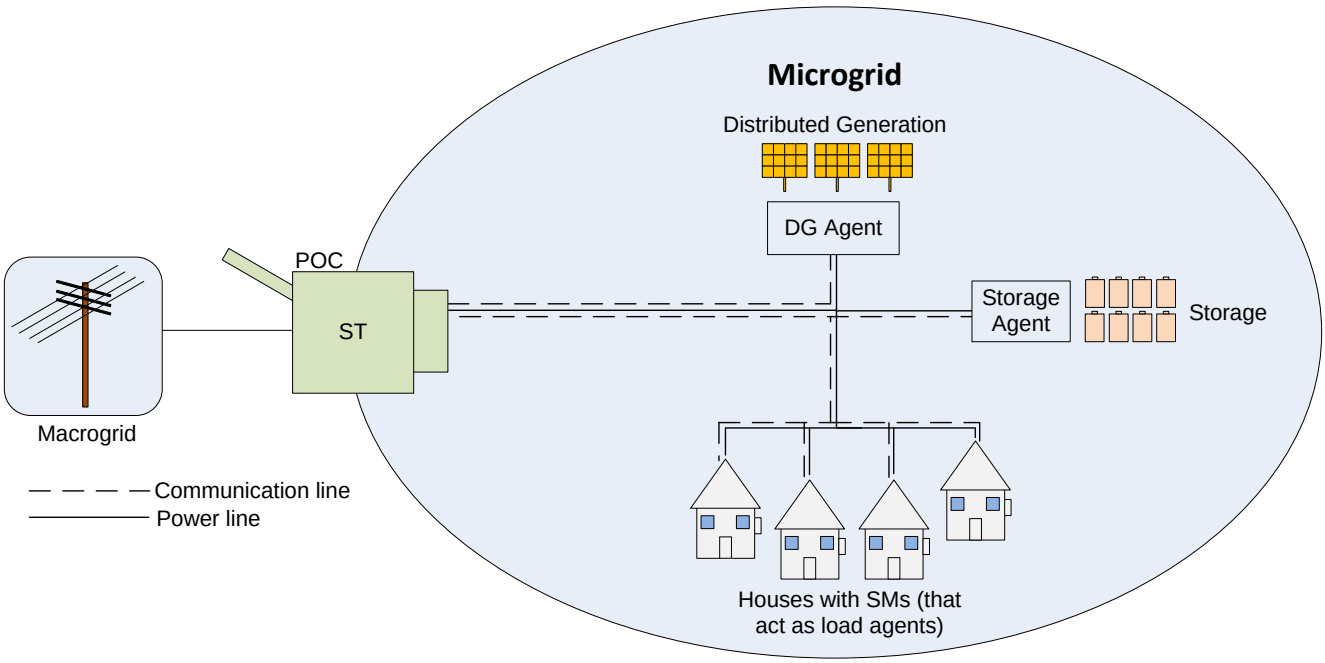


Figure 2-3 - Microgrid with ST

Each agent performs unique tasks to contribute toward the stability of the microgrid. Each agent also performs tasks that are generic to all agents. A description of both unique and generic tasks follows.

Stellenbosch University https://scholar.sun.ac.za

2.4.1.1 Generic Tasks

A microgrid agent acts as a data gatherer, decision maker and actuator. Each agent performs some tasks that are generic to all microgrid agents. This includes gathering data from the microgrid component it is servicing, and performing analysis on that data to transform it into usable formats. The analysis performed on data is dependent on the level of computational ability possessed by the microgrid agent.

Raw data can be seen as the lowest form of data, usually gathered directly from underlying hardware. This mainly consists of readings of different data points present on equipment. Microgrid agents perform data analysis locally on this data to transform it into information. Information is data transformed to express something about the component's operation and contribute towards the decision making process. An example would be monitoring the load voltage level of a distribution transformer. The immediate load voltage comes in as a raw data reading. When the voltage surpasses a defined threshold, it is a significant event. Knowing that a meaningful event has occurred is information [12].

Once converted to information, data is sent to the ST located at the POC, which acts as a data aggregator and contains greater processing power than the other microgrid agents contain. At the data aggregator, knowledge is produced from information. Knowledge is the usage of information to gain a greater understanding of the operations of the microgrid in real time. Machine learning and other techniques are used to accomplish this. Knowledge allows microgrid intelligence to recognize normal operating conditions and deviances from those conditions.

Continuing with the example of measuring distribution-transformer voltage levels, the information produced by this monitoring is used to produce knowledge. When the source voltage of a distribution transformer drops below 115 V, it is known that the transformer is experiencing a state of *undervoltage*. Conversely, when the source voltage rises above 265 V, it is known that the transformer is experiencing a state of *overvoltage*. Thus, by looking at the measurements of source voltage taken from all transformers, their under / overvoltage status is obtained.

At the data aggregator, knowledge is used to produce wisdom. Wisdom is knowledge applied to enable decision-making and perform predictions. Wisdom is used in the microgrid to issue control commands to various microgrid agents and role-players. Wisdom also encompasses predicting future microgrid operations and allows components to prepare for predicted conditions [12].

Again continuing with the previous example, it is known that if a transformer functions in state of overvoltage for a period of time, the life expectancy of the transformer decreases [13]. Using this knowledge, the life expectancy of the distribution transformer can be predicted according to how much time it spends in the state of overvoltage [13]. From this prediction, planning of transformer replacement can be done. This is knowledge applied to produce wisdom.

Stellenbosch University https://scholar.sun.ac.za

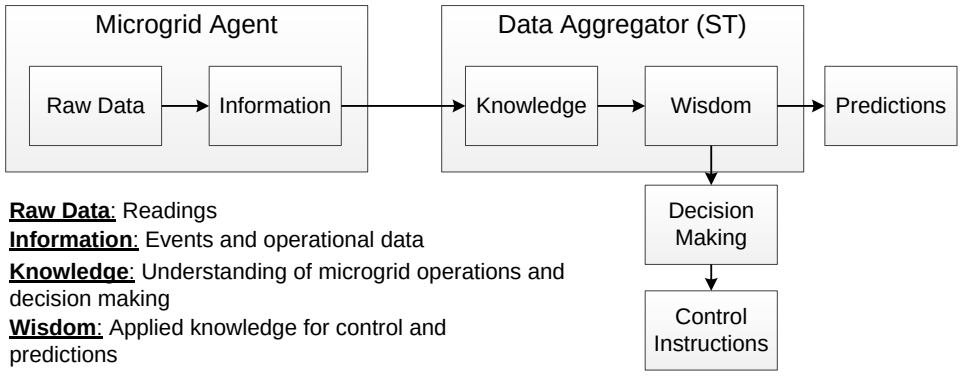


Figure 2-4 - Microgrid Data Hierarchy

In this project, it is suggested that the ST act as both a microgrid agent, and a data aggregator within the microgrid structure. It fulfils these two roles separately from one another. This is discussed in the following sections. At this point it should be noted that this project only considers the model of microgrid where a POC agent is used.

2.4.1.2 Unique Tasks

As previously mentioned, apart from the generic tasks, each microgrid agent also performs unique tasks to contribute towards the stability of the microgrid.

POC Agent

The POC agent is located at the point of coupling between the microgrid and the traditional utility grid. The POC agent allows control of the connection or disconnection of the microgrid from the macrogrid.

The POC agent facilitates two-way electricity flow between the microgrid and the macrogrid. Thus, electricity can either enter or leave the microgrid as is necessary to meet demand and maintain grid stability.

In this project, the ST is used as the POC agent. It monitors the microgrid line voltage and uses transformer tap changes to contribute towards stabilizing the microgrid line voltage. As a secondary function, the ST agent minimizes the amount of tap changes performed to elongate the lifetime of the tap-changing transformer [11].

In this project, the POC agent also acts as a data aggregator for the microgrid. A description of the functionality of the data aggregator follows later in this section.

DG Agent

The DG agent is located at DG sources. The DG agent monitors the line voltage of the microgrid. It helps to control the line voltage by determining the amount of current that the DG component delivers into the microgrid. This is done by increasing or decreasing the amount of energy generated by the DG source. Thus, DG sources are dynamically controlled to ensure microgrid stability. In this project, it is assumed that the DG agent is controllable.

Stellenbosch University https://scholar.sun.ac.za

Load Agent

Load agents are located at concentrations of loads within the microgrid such as houses or buildings. Load agents service multiple loads. They monitor the line voltage of the microgrid and turn controllable loads, within their respective sphere of influence, on and off. This is done to contribute towards controlling the microgrid line voltage level. Controllable loads are non-critical and can thus be turned on and off according to demand and supply.

An example of a controllable load is a geyser. Geysers can be turned on when microgrids have an excess of generation and turned off when there is an excess of demand. As the level of implementation of the SG increases, more loads that are controllable will arise in both households and businesses. The major foreseen controllable loads include geysers and electric vehicles.

Storage Agent

Storage agents monitor the line voltage of the microgrid and control electricity storage to contribute towards control of the line voltage. Storage agents make storage available when there is an excess of generation and pushes electricity back into the grid when there is an excess in demand.

The storage components are the main stabilizers in the microgrid. Storage agents measure the line voltage of the microgrid and ensure that it remains at a pre-set reference voltage by either storing excess generation or by delivering electricity when there is an excess in demand.

Optionally, electric vehicles can also be used as a form of electricity storage in a microgrid. Electric vehicles are charged using excess generation at night time. In peak times, electricity is extracted from electric vehicle batteries to meet demand. It is uncertain at what times electric vehicles will be connected to the grid and thus they cannot be seen as a reliable source of storage and energy. With good control, they can positively contribute towards stabilizing the line voltage of the microgrid.

Data Aggregator

The data aggregator functions as the data manager of the microgrid. The main task of the data aggregator is to perform intra-microgrid data collection and management. Other microgrid agents communicate with the data aggregator and send data to it. The data aggregator then analyses this data, obtains results from the analyses, reports the data to central servers, makes decisions based on the analyses results and performs control throughout the microgrid.

The data aggregator can be seen as a local data manager within the microgrid. Where the POC is a singular point of connection between the microgrid and the macrogrid, the data aggregator is a single point of communication between the microgrid and central data servers.

2.4.2 Microgrid Balance

Overall, the microgrid has three main variables: generation, demand and storage. These three need to be controlled in order to maintain the stability of the line voltage in the microgrid.

Stellenbosch University https://scholar.sun.ac.za

Generation needs to meet demand, and storage is utilized either to store excess generation or as an energy source to meet excess demand.

To contribute towards the stability of the line voltage in the microgrid, DG agents control the amount of generation DG sources produce, load agents turn controllable loads on and off thus managing the total load of households and businesses, storage agents manage storage in the microgrid and in the context of the ST, the POC agent uses tap-changing mechanisms.

The data aggregator keeps track of the total amount of generation, the total demand and the condition of all storage components in the microgrid (capacity of storage occupied and capacity free). The data aggregator and the microgrid agents work together to keep the microgrid stable. A communication network allows these components to communicate with one another.

When the microgrid is islanded from the macrogrid, agents make the necessary changes to allow for normal functioning to continue. Agents respond in accordance to the current and future predicted line voltage and make adjustments to ensure that it remains within a specified range [11], [9].

2.4.3 Communication Network

The microgrid communication network connects all agents in the microgrid to allow effective monitoring and control. Macrogrid operators need only communicate with the ST agent (the data aggregator) as it contains sufficient information about microgrid operations and can issue control commands to other agents.

Communication network technologies vary according to the specific application. Wired connections between microgrid agents are always preferred as they are the fastest connections, but there is a cost penalty. Power line carrier (PLC) and wireless technologies such as WiMAX, radio frequency and WiFi can also be used for intra-microgrid communications depending on the application [3]. In this project, mobile communication was used to communicate with ST units in rural areas. A wired communication network was also utilized to simulate microgrid operation between various microgrid agents. These applications will be discussed in later sections.

It is advised to have a fast, reliable communication link between the ST / POC agent and the macrogrid operators. This enables real-time visibility, decision making and control.

2.5 DISTRIBUTED GENERATION

The current electricity grid relies heavily on centralized generation. This generation often consists of large plants that produce large amounts of electricity. Transportation of electricity throughout the grid happens over transmission lines. This approach has a number of disadvantages. Firstly, generation of electricity happens far from the loads that it supplies resulting in transportation losses. Secondly, the grid is heavily reliant on only a few large sources of electricity. From a reliability perspective, it is a better option to have many smaller sources of generation than a few

Stellenbosch University https://scholar.sun.ac.za

large sources. When a smaller electricity source fails, it has less impact on the stability of the grid than when a large electricity source fails. A possible solution to these issues is DG.

The DG principle entails generating electricity in smaller amounts at plants located closer to the loads that they serve. These generating plants may be either fossil fuel or renewable-based sources of electricity. The concept that the electricity grid can be divided into smaller microgrids consisting of DG and loads was formulated around the introduction of DG [3]. Microgrids offer advantages from both the grid-side and customer-side points of view. From the grid-side, a microgrid appears as a single controllable unit. From the customer-side, the microgrid offers increased reliability. This is because connecting more DG results in more unpredictable generation and demand patterns. The microgrid facilitates the integration of DG into the utility grid by dealing with adverse effects of connecting DG electricity sources to the grid [14]. An example of one of these adverse effects is voltage instability due to unpredictable generation patterns.

Communication to and between distributed elements will play a major role in the development of the microgrid concept. Effective monitoring and control of distributed elements is crucial to ensure the reliable and efficient functioning of the grid [3].

It is shown that the power exchange between a microgrid and the utility grid can be controlled by inserting a ST at the connection point between a microgrid and the utility grid. An advantage of this is that the utility grid only needs to communicate with the ST and not every DG element individually. Indirect control of DG elements by the droop control method then takes place on the microgrid-side of the ST [15].

2.6 DEMAND DRIVEN ENERGY MARKET

As more and more distributed energy sources are connected to the grid, it becomes harder to manage them all. With the realization of microgrids, this problem is already addressed, but microgrids and the technologies that they introduce open other doors to managing the grid more effectively. One such door leads to a demand driven energy market (DDEM).

A DDEM is an energy market that allows electricity to be bought and sold within both the microgrid and the larger macrogrid. The price of electricity is determined by the energy demand and supply levels both locally within the microgrid, and within the macrogrid. A local electricity market where electricity is bought and sold is created in each microgrid. These markets interface with the larger electricity market of the macrogrid [9]. Electricity generators generate electricity and pay a fee to the power utility to use their lines for electricity transport. Effective use of microgrid agents enables the DDEM.

Grid visibility greatly increases with the realization of the microgrid concept in the SG. This goes down to the level of knowing the total generation and demand within a microgrid, and knowing how much each energy source is generating and each load is demanding. This knowledge allows the generation of electricity tariffs in real time, a practice known as real-time pricing (RTP) [9].

Stellenbosch University https://scholar.sun.ac.za

Microgrid agents are used to manage the market within a microgrid. The ST acts as the data aggregator within the microgrid and the main point of DDEM management. It determines the total generation and demand within the microgrid by communicating with the various microgrid agents and determines electricity tariffs by taking microgrid agent information, as well as information acquired from the power utility (macrogrid side), into account [9].

The ST agent then informs other microgrid agents of current electricity tariffs rates. DG sources can decide how much they want to generate depending on the current tariffs rate. The utility sets a minimum generation level that DG sources must adhere to. Load owners can also determine when they want to switch on their non-vital loads by monitoring the current tariffs with their smart meters (SMs). Research on home energy management based on RTP is discussed in [16]. Microgrid storage is used to store excess generation and act as an energy source during excess demand.

To ensure that the market functions properly, all data sources including the tariffs are time-stamped accurately. A tariff that was active in a specific time range is applicable on electricity generation and usage readings from DG and load agents within the same time range. The ST determines the monetary compensation that DG sources should be awarded and loads should be charged and sends this information to the central macrogrid operators where the transactions can be issued.

2.7 SMART TRANSFORMER CONTEXT

The ST discussed in this project is for application in deep rural, hard-to-reach areas. Dispersed housing and farms characterize these areas. Research has shown that approximately 38% of the South African population live in rural areas [17]. Deep rural areas offer unique challenges to STs in terms of communication and maintenance as electrical and communication infrastructure in these areas are often elementary.

In the early 1990's the South African government in conjunction with the country's power utility, Eskom, began an initiative to provide basic electricity to all South Africans, including those living in rural areas [18]. In 2006 Eskom initiated a project to develop a distribution transformer for specific application in rural areas. The Self-Regulated and Protected Electrification Transformer (SRAPET) was developed as a robust solution capable of decreasing the cost of rural electrification. The SRAPET is available in both 16 kVA single and 32 kVA dual phase models and contains intelligence that allows for remote monitoring and control of the transformer. The transformer contains a solid-state electronic tap-changer for regulating the voltage on the low voltage side. This allows the extension of the low voltage feeder. It also contains protection mechanisms that protect the transformer against overloading. Four single-phase SRAPET units were deployed for field tests by Eskom and showed successful performance results [2].

In 2013, Eskom published a new communication standard for all remote data retrieval devices [4]. The standard sets a list of requirements for all substation-based and non-substation-based data

Stellenbosch University https://scholar.sun.ac.za

retrieval devices to ensure that they are reliable, robust, and effective. Among these requirements, a set of protocols are required that enable the remote monitoring and control of any such device. The main data retrieval protocol may be either IEC 61850 for substation-based devices, or DNP3 for non-substation-based devices.

For non-substation-based devices, the requirements contained in the standard are tabulated in Table 2-4.

Stellenbosch University https://scholar.sun.ac.za

Table 2-4 - Eskom Standard Specification Profile

Table Number	Requirement	Description	Required Protocol
1	Operational monitoring	Non-substation based devices must possess the capability to be monitored remotely by making use of industry standard communication protocols.	Distributed Network Protocol 3
2	Configuration interfaces	Remotely monitored devices must also be remotely configurable. Remote configuration of the device includes the capability to: • Apply settings remotely • Update firmware remotely • Change passwords remotely • View access and statistical logs remotely • Extract the sequence of events remotely • View all operational data remotely Remote configuration interfaces must have username and password authentication.	Hypertext Transfer Protocol (HTTP), File Transfer Protocol (FTP) and Secure Shell (SSH)
3	Network monitoring	The networking functionality of remotely monitored devices must be remotely monitored. Certain prerequisites were set as to what information must be made available for monitoring.	Simple Network Monitoring Protocol (SNMP) version 1 or higher
4	Time synchronization	Non-substation based devices must support a method by which their clocks can be synchronized. Internal clock accuracy must be at least 10ms.	Network Time Protocol (NTP) v3
5	Communication interfaces	Remotely monitored devices must support effective communication interfaces to communicate data.	Cellular modem and physical Ethernet port (100Base-T)
6	Authentication	All remotely monitored devices are to support authentication by means of a username and associated password. Preferably, three levels of authentication should be supported with varying levels of access to the device.	Not specified

Stellenbosch University https://scholar.sun.ac.za

7	Security	All non-substation based devices must support encrypted links to the devices. Such encryption must be scalable and support at least two outgoing and incoming connections. Encryption algorithms must support a key of 128 bits or larger. Certificates may be used as the encryption techniques.	Not specified

A separate device that interfaces with the SRAPET transformer was developed to enable the transformer to meet the new Eskom standard. The device uses the DNP3 protocol for data communication. The device was prototyped and tested and the results and applications are recorded in this project thesis.

2.8 TRANSFORMER CONDITION MONITORING

Condition Monitoring (CM) of assets forms a crucial part of SG monitoring. Fault detection and maintenance prediction are two of the benefits derived from CM of electrical grid assets.

Transformers fulfil a crucial role in the electricity grid. Transformer faults affect a large number of customers and thus they carry a high priority. Condition monitoring of transformers add value in the sense that transformer faults can be detected and identified and the number and duration of outages can be reduced [19].

The concept of a ST requires a two-way communication system with the utility that enables functionality such as monitoring transformer variables, evaluation of transformer loading, planning of grid expansion and replacement of overloaded transformers [20].

Some parameters that are of interest in transformer CM include: currents, voltages, temperatures, phase and frequency, peak values, true power, apparent power, power factor, harmonics, sub-harmonics, RMS values and accumulated energy (watt-hours) [19]. Some of these parameters are measured while others are derived. Measuring these parameters enable analyses techniques such as thermal analysis [21].

As mentioned in section 2.7, the SRAPET is a transformer developed for application in deep rural areas. The SRAPET has built in intelligence to monitor the following parameters:

- Source and load voltage levels.
- Load and earth current levels.
- Transformer internal, external and heatsink temperatures.
- Estimation of transformer hotspot and top oil temperatures.
- Transformer electronic tap position.
- Transformer operational status.

Stellenbosch University https://scholar.sun.ac.za

These parameters are logged locally and can be downloaded from the SRAPET using Short Message Service (SMS) and GPRS based communication. They can also be acquired by streaming them via an RS232 serial port [2].

2.9 DISTRIBUTED NETWORK PROTOCOL VERSION 3

2.9.1 Introduction

Many protocols have been suggested for use in retrieving data from SG assets. The use of certain protocols will be more effective in certain situations due to unique protocol characteristics and specific situational needs.

In the field of Supervisory Control and Data Acquisition (SCADA), the two main protocols in use are IEC61850 and DNP3. DNP3 was the protocol specified for use in non-substation devices in the Eskom Remote Device Communication Standard for Data Retrieval and Remote Access [4]. This was concluded on the basis that DNP3 sends less data per message over the network than IEC61850, thus using less bandwidth.

2.9.2 DNP3 Overview

DNP3 is an open, efficient, robust and intelligent utility protocol designed to optimize the transmission of information pertaining to data acquisition and control commands across a utility network. The protocol was originally created by Westronic Inc. between 1992 and 1994 to address various issues that they identified in protocols in the utility industry. One of the main objectives in the DNP3 project is to reduce the amount of bandwidth used to communicate data while not compromising on data integrity [22], [23].

2.9.3 Basic Topology

The typical DNP3 topology consists of a *Master* station communicating with one or more *Outstations*. These outstations have interfaces to physical devices to monitor and control them. The outstations collect data from the underlying devices and send commands to them, exercising control. A variety of communication media are used to connect outstations to master stations. Master stations act as data servers, collecting data from outstations, managing that data and performing control on outstations. An illustration of the basic DNP3 topology can be seen in Figure 2-5.

Stellenbosch University https://scholar.sun.ac.za

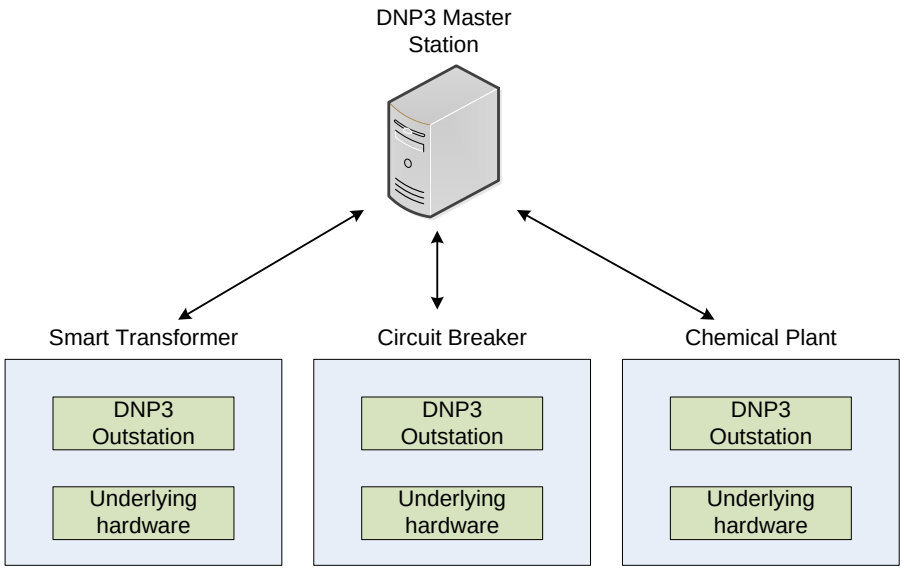


Figure 2-5 - DNP3 Basic Topology

2.9.4 DNP3 Features

The designers took various functionalities that they had deemed useful in other utility protocols and incorporated them in the DNP3 protocol. Some of the more important features of the DNP3 protocol are listed below [22], [23]:

- **Multiple Device Addressing –** A single message can be sent to over 65 000 other devices on a single link.
- **Time-stamped Data –** DNP3 offers time-stamping on almost all data types. This functionality is useful for gathering historical data.
- **Time Synchronization –** DNP3 provides a means of time-synchronization between master and outstation. This solution is derived from various other protocols and takes software time-delays into consideration.
- **Diagnostic Information –** DNP3 uses quality flags to see if data is valid and what the reason for the validity or invalidity of the data is. DNP3 also uses diagnostic data points known as *Internal Indications* to monitor the health of outstation devices.
- **Various Data Formats –** DNP3 possesses functionality to report data in a variety of formats. The user can define the data format to be used.
- **Data Grouping –** A user can use DNP3 *Scan Groups* to group otherwise unrelated data for mass reporting.
- **Report by Exception –** Perhaps the feature that has contributed most to DNP3's success is its ability to report data only when changes occur. This greatly helps to reduce sent bytes in an environment where cost-per-byte optimization is required.
- **File Transfer –** DNP3 offers the ability for file transfer in both directions: master to outstation and outstation to master. This allows outstations to send batch reading information to masters and to download new configuration files from masters.

Stellenbosch University https://scholar.sun.ac.za

- **Polled or Spontaneous Reporting –** Data may be requested from an outstation by a master, or an outstation may send data to a master spontaneously when a relevant change occurs.

2.9.5 DNP3 Reporting

DNP3 reports data using both the techniques of data polling and spontaneous data reporting. In data polling, a master station requests a specific set of data from a single or multiple outstations and the outstation/s respond with a *solicited response* containing the requested data. In spontaneous reporting, an outstation sends a spontaneous or *unsolicited* message to a master station when a significant event occurs.

When a master station polls an outstation for data, it may specify what *scan class* of DNP3 data must be returned by the outstation. DNP3 scan classes are user-defined data classifications. Data can either belong to scan class 0, 1, 2 or 3. Scan class 0 automatically contains all static data while data in the other three scan classes are prioritized according to the scan class containing it. Scan class one contains the highest priority data while scan class three contains the lowest. A scan class can be seen as an empty box that a user can fill with whatever data he or she wishes.

DNP3 stores data in *data points*. A data point is a logical entity that can either be an input or an output. Every data point possesses a type and a value. Point types include *binary input, binary output, analog input, analog output, counters and frozen counters.* A data point has a static value at a specific time. The point may generate events when this value changes. Points are stored on a DNP3-capable device in a database-like structure. An outstation contains a database of points containing a table for each point type with each individual point entry possessing a unique numeric identifier. A master station stores an exact replica of each outstation database that it serves. When data points in the outstation are updated, the data points in the master's database are also updated. This concept is illustrated in Figure 2-6.

Stellenbosch University https://scholar.sun.ac.za

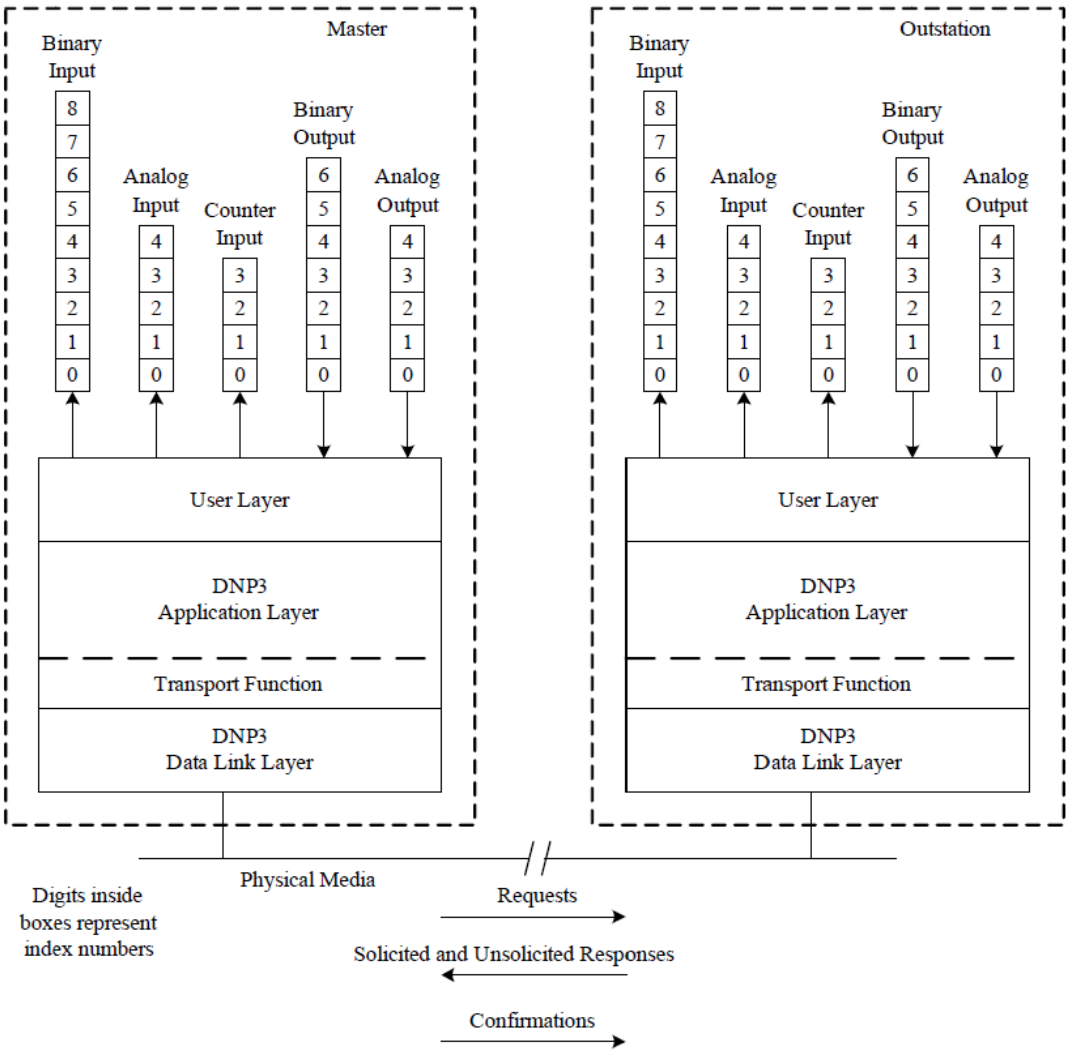


Figure 2-6 - DNP3 Communication Concept [23]

As can been seen in Figure 2-6, DNP3 possesses a number of protocol layers. These layers are used to build and decode DNP3 messages depending on whether they are being sent or received respectively. For more on the DNP3 protocol, please see APPENDIX A.

SUMMARY

The SG is an initiative to make the existing electricity grid more cooperative, responsive and economic and to improve the penetration of renewable energy sources.

Microgrids can be considered the building blocks of the SG and are clusters of electricity loads, generators and storage. Microgrids have a point of connection to the larger utility grid (macrogrid) and can be either connected to or disconnected from the utility grid at this point. Various important microgrid hardware components are fitted with intelligent agents that can interact with one another creating a communication network within the microgrid.

The ST is the agent situated at the point of coupling between the microgrid and the macrogrid. The ST must effectively monitor and communicate its own operations and interact with other microgrid agents. The ST implements the DNP3 and SNMP protocols for operational and network

Stellenbosch University https://scholar.sun.ac.za

monitoring. The ST must be remotely accessible and configurable using the HyperText Transfer Protocol (HTTP) interface and a stateless ASCII interface over a Telnet session (SSH).

DNP3 is an open, efficient, robust and intelligent utility protocol designed to optimize the transmission of information pertaining to data acquisition and control commands across a utility network. DNP3 was selected by Eskom as the remote data retrieval protocol to be used by all non-substation based remote data retrieval devices.

This thesis investigates the role of the ST in the microgrid as a distributed computing device. The specific area of context is deep rural areas.

Stellenbosch University https://scholar.sun.ac.za

CHAPTER 3
SMART TRANSFORMER COMMUNICATION DEVICE

3.1 INTRODUCTION

This chapter documents the development of the Smart Transformer Communication Device (STCD).

The STCD is a device developed to extend the functionality of the SRAPET transformer. The device interfaces with the SRAPET transformer and enables it to meet the Eskom Remote Device Communication Standard for Data Retrieval and Remote Access [4]. The development of the STCD aims to move the SRAPET towards a fully-fledged ST. The STCD uses the DNP3 protocol to communicate measurement and operational data to a remotely located data server.

3.1.1 Reasoning

The STCD is a stand-alone device developed to interface with the SRAPET transformer. This was done for prototyping purposes to keep the existing functionality and the added functionality separate. The aim of following this process is to allow the separately developed device to reach maturity before integrating it into the SRAPET system.

The separation of the STCD and the SRAPET controller board meant that communication had to take place between the two devices. This would incur a time delay between when measurements are taken by the SRAPET and when they are reported by the STCD. This issue was not considered to be of significant importance that it should change the aim to first prototype the desired functionality before redesigning the SRAPET controller. Serial communication was chosen as the transmission medium between the two devices as the SRAPET controller board already had an available RS-232 serial port that could be utilized.

It was later recognized that because of the rapid pace at which communication technologies advance, the transformer operational and communication devices should be kept separate with an interface between them. This is done to allow easy upgrading of the communication capabilities of the transformer without having to redesign the operational hardware.

The STCD would need to report data to a remotely located data server. The data server fulfils the role of a DNP3 master station that services multiple DNP3 outstations in the form of STCD units. This communication could take place over a selection of channels including fibre optic cables, Broadband Power Line Carrier (BPLC), WiMAX and public cellular networks [24]. For the STCD, it was decided that a public cellular network would be used to communicate to the data server. A public cellular network was chosen because infrastructure for cellular communication is already widely available in rural areas. The data server is located at Stellenbosch University, and the following requirements were identified for communication between the STCD and data server:

Stellenbosch University https://scholar.sun.ac.za

- A secure connection must be established between the STCD and data server.
- A static IP address must be assigned to each STCD unit to allow the data server to poll them for data.

These requirements could be optimally met by the use of a private APN to facilitate communication between devices. Because of the cost of hiring a private APN, it was decided that a Virtual Private Network (VPN) would be implemented to fulfil the stated communication requirements.

Data communication to the data server would need to be efficient and effective, utilizing the processing capabilities of the microprocessor-based STCD to save bandwidth. Thus, a reporting philosophy was implemented that effectively communicates data to the data server. The reporting philosophy focusses on saving bandwidth and distributing the computational load of data analysis in the power grid.

3.1.2 Device Goals and Relevance to ST

Certain functional goals were set out for the STCD to achieve:

- Implement software to meet the Eskom standard as specified in Table 2-4.
- Implement an eXtensible Markup Language (XML) interface that a user may use to configure the STCD. This is not strictly required by the Eskom standard, but it was decided that such an interface would be useful.
- Implement database software to store data and configuration settings locally and to act as a communication channel between various programs on the STCD (for example, a central storage where settings are saved and can be accessed by all programs).
- Develop the STCD to be a robust system that is capable of automatically fixing faults that occur on it.
- Perform in-lab and field-testing on the STCD.

The functionality that the STCD provides allows the SRAPET transformer to meet the Eskom Remote Device Standard. For the remainder of this thesis, the STCD and SRAPET together will be referred to as the Smart Transformer (ST).

3.1.3 STCD System Topology

The STCD system topology illustrates how the ST interfaces with the outside world. The topology is shown in Figure 3-1.

Stellenbosch University https://scholar.sun.ac.za

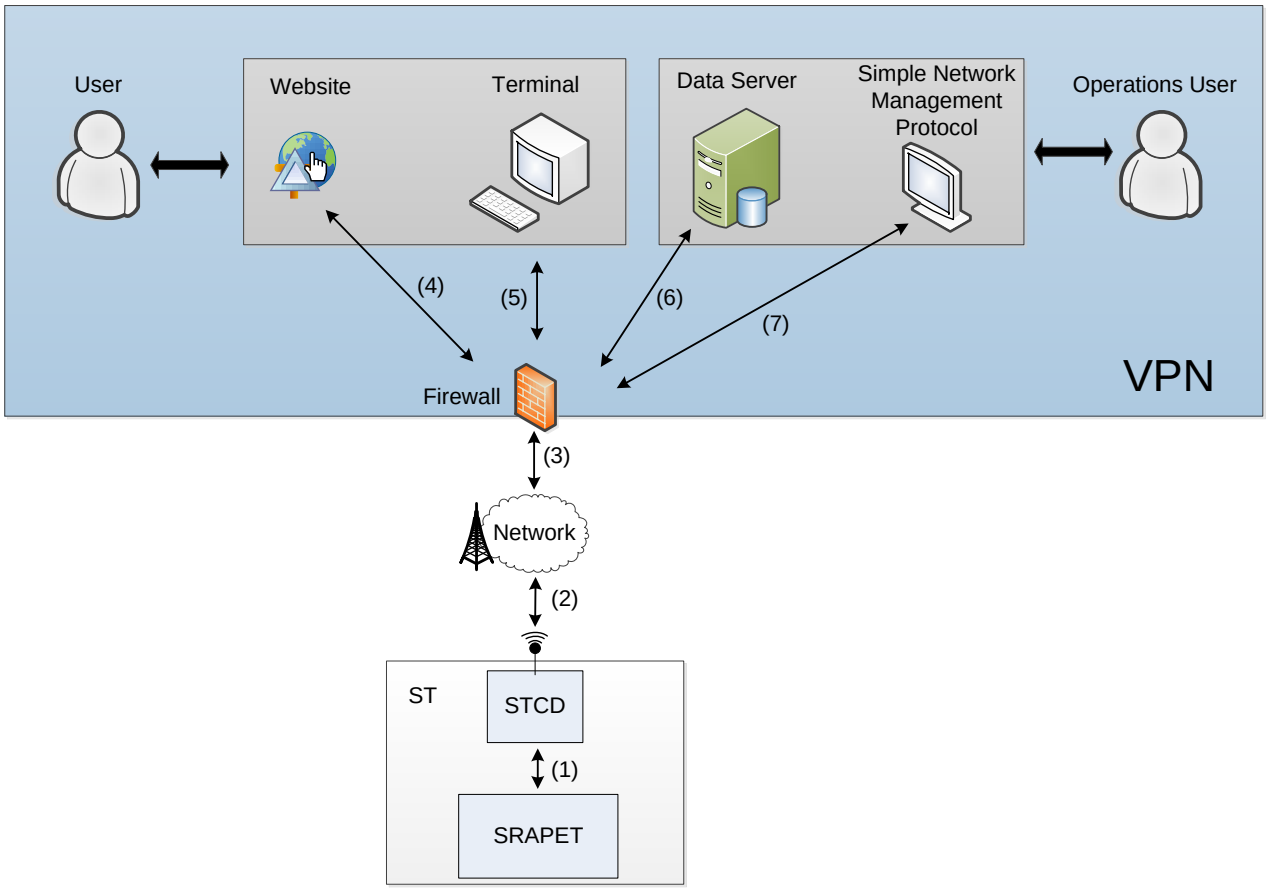


Figure 3-1 - Global Software Topology

The system topology is described in Table 3-1.

Table 3-1 - Global Software Topology Table

Communication Link	Description	User
RS-232 Connection **(1)**	The STCD and the SRAPET interface over a RS-232 serial connection.	None
Mobile Internet Connection **(2)**	The STCD connects to a cellular company's APN to gain access to the internet through a modem. The implementation of a connection via modem satisfies point 5 in Table 2-4.	None
Connection of STCD to a VPN **(3)**	The STCD is connected to a Virtual Private Network (VPN). This allows it to securely connect to the data server and host various interfaces on the private network through a firewall. This is described in greater detail in section 3.3.13 of this document. The use of a VPN satisfies point 6 in Table 2-4.	None

Stellenbosch University https://scholar.sun.ac.za

Web interface hosted on STCD **(4)**	The web interface hosted on the STCD enables a user to connect to a website to perform various monitoring and configuration tasks. The HTTP and FTP protocols are used to enable this connection. The web interface is described in greater detail in section 3.3.7 of this document. The implementation of a web interface contributes to the satisfaction of point 2 in Table 2-4.	Any VPN user with authentication
Secure Shell (SSH) connection to the STCD **(5)**	Users may connect to the STCD's command-line interface using the SSH protocol. This interface allows users to perform various tasks in the Linux command-line environment. The implementation of a SSH connection contributes to the satisfaction of point 2 in Table 2-4.	Any VPN user with authentication
STCD connection to the data server **(6)**	The data server and STCD act as a DNP3 master station and outstation respectively. The DNP3 protocol is used to connect the STCD to the data server. The implementation of the DNP3 protocol satisfies point 1 in Table 2-4.	Operational users may access data on the data server
Connection from an SNMP client program to the STCD **(7)**	The STCD hosts a SNMP agent capable of responding to requests from a SNMP client program. This enables SNMP monitoring of the STCD. The implantation of the SNMP protocol satisfies point 3 in Table 2-4.	Operational users with authentication may use SNMP client programs to gather SNMP data from the STCD

A complete wiring diagram of the hardware setup is given in Figure 3-2.

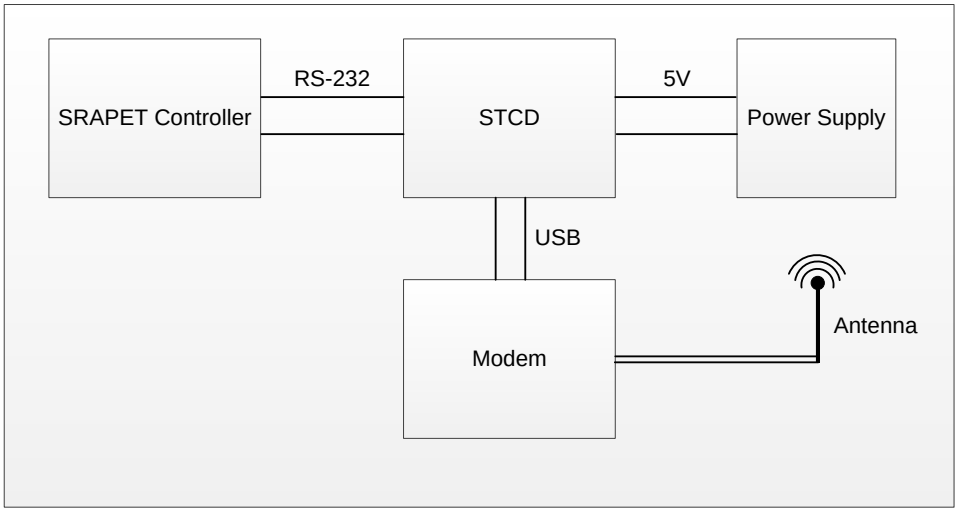


Figure 3-2 - Hardware Wiring Diagram

Stellenbosch University https://scholar.sun.ac.za

3.2 HARDWARE

The STCD uses a Beaglebone Black (BB) development board from Texas Instruments as a hardware foundation. The BB is a prototyping board that possesses adequate hardware to meet the hardware needs of the STCD [25]. It implements a 1GHz Arm processor and possesses 512MB DDR3 SDRAM memory and 2GB of Embedded MMC storage. Power is supplied to the board via a five-volt jack and a USB port is available on the BB to connect peripherals. The board also possesses a Ethernet port for network connectivity. The BB board is shown in Figure 3-3.

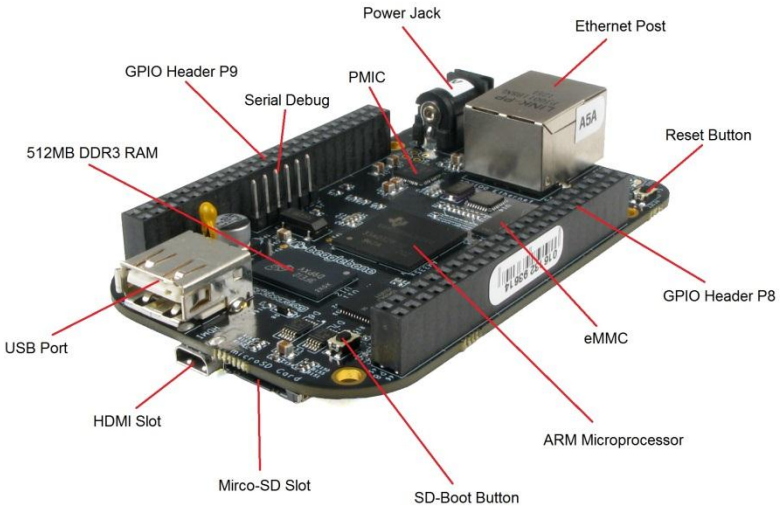


Figure 3-3 - Beaglebone Black

Serial communication between the STCD and the SRAPET would require the use of a Universal Asynchronous Receiver / Transmitter (UART) on the BB and a RS-232 converter module to convert serial signals to the appropriate levels as BB serial signals differ from standard RS-232 signals.

A 3G USB modem was utilized to enable communication between the STCD and the data server.

3.2.1 The SRAPET Controller Board

The SRAPET transformer possesses a controller board that the BB interfaces with to gather transformer measurement data and to send instructions to the transformer.

The controller board is responsible for:

- Processing the transformer measurements.
- Controlling voltage regulation through tap switching.
- Implementing protection measures.

A block diagram that illustrates the various components of the SRAPET controller board is given in

.

Stellenbosch University https://scholar.sun.ac.za

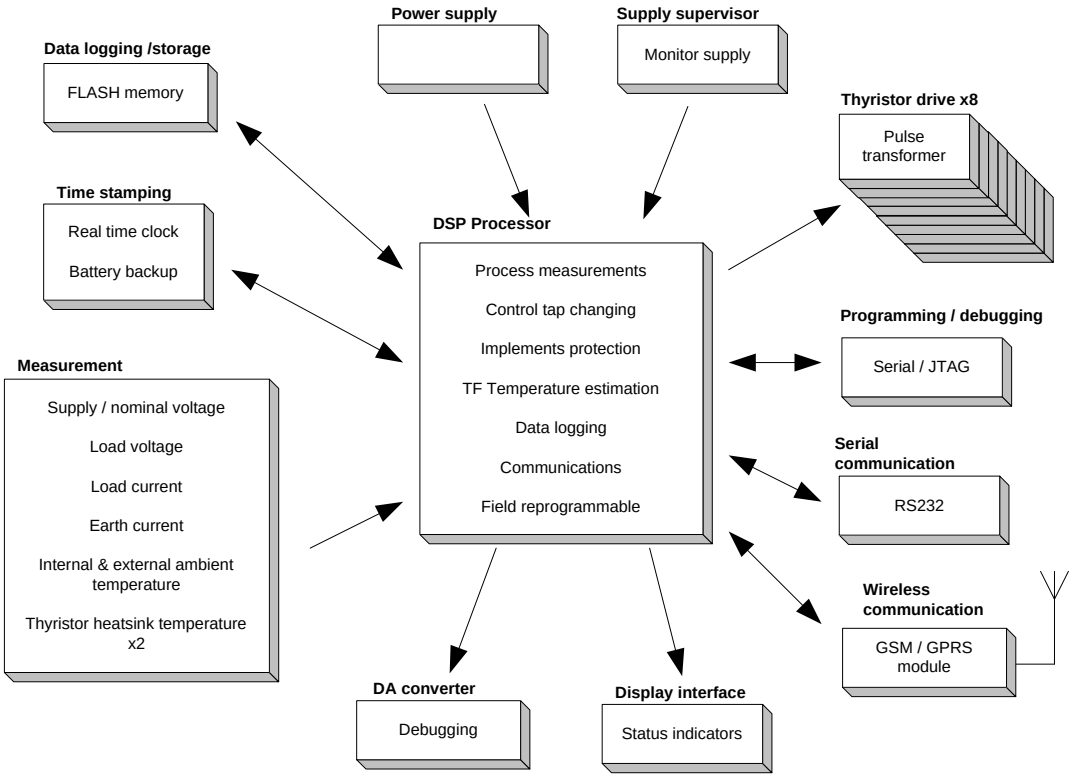


Figure 3-4 - SRAPET Controller Block Diagram [26]

The controller board also has local log retrieval and data streaming capabilities accessible via a RS-232 serial port located on the board, or via a wireless GSM connection [2].

The SRAPET controller board software was extended to allow the STCD to download measurement information from the controller and send instructions to the controller via the RS-232 port. The SRAPET controller board is shown in Figure 3-5.

Stellenbosch University https://scholar.sun.ac.za

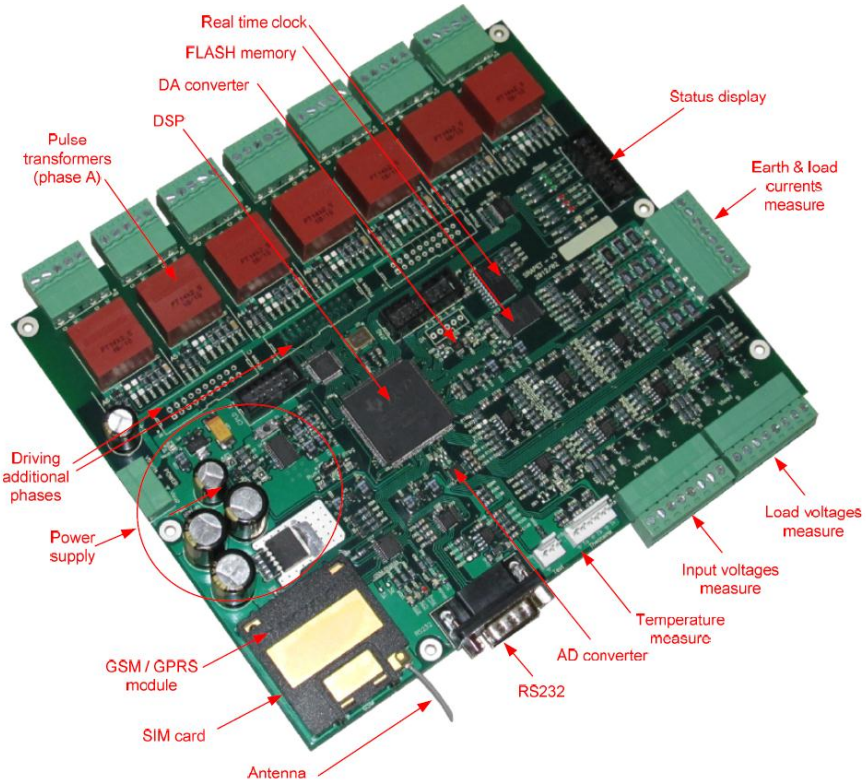


Figure 3-5 - SRAPET Controller Board [2]

3.2.2 Serial Communication between STCD and SRAPET

The STCD uses a serial connection to communicate with the SRAPET controller board.

The serial connection uses the RS-232 protocol, utilizing a single UART on the BB to connect to a RS-232 port available on the SRAPET controller board via a RS-232 converter. The RS-232 converter is used to convert BB serial messages to RS-232 standard messages.

3.2.2.1 RS-232 Protocol

RS-232 (Recommended Standard – 232) is a standard for serial communication developed by the Electronic Industry Association (EIA) and Telecommunications Industry Association (TIA) of America.

The standard fully specifies how communication is to take place between the two components in RS-232 communication: a DTE (Data Terminal Equipment, such as a computer terminal) and a DCE (Data Circuit-Terminating Equipment, such as a modem). RS-232 is a complete standard, specifying electrical, functional and mechanical characteristics for inter-device communication to ensure compatibility between systems. The standard uses wired media to connect devices fitted with RS-232 ports. A RS-232 port contains a number of pins, each used for sending or receiving a unique signal [27].

Stellenbosch University https://scholar.sun.ac.za

Electrical Characteristics

The RS-232 standard specifies that a voltage between -3 V and -15 V is seen as a logical one, while a voltage between 3 V and 15 V is seen as a logical zero. A maximum slew rate of 30 V/µs and a maximum data rate of 20 kbits/s are specified. This reduces the chance of cross-talk. A maximum capacitive load of 2500 pF that can be seen by the device is specified. This in turn sets a limit on the maximum cable length between devices, determined by the capacitive load of the cable [27].

Functional Characteristics

The RS-232 standard defines a total of 24 functional signals. Each signal performs a specific task. The signals are categorized as either *common*, *data*, *control* or *timing*. Few, if any, real-world applications use the full spectrum of RS-232 signals. Most applications use a subset of eight of these signals. This eight-signal application is used for the STCD. The eight signals are listed in Table 3-2. Note that the table context is from the viewpoint of the DTE.

Table 3-2 - RS-232 Signals [27]

Signal Name	Description	Signal Direction
Data Carrier Detect (DCD)	Indicates that the DCE is connected to a telephone line.	DCE to DTE
Receive Data (RD)	Data from the DCE to the DTE is carried on this pin.	DCE to DTE
Transmit Data (TD)	Data from the DTE to the DCE is carried on this pin.	DTE to DCE
Data Terminal Ready (DTR)	Indicates that a serial connection between a DTE and DCE is established.	DTE to DCE
Ground (GND)	Even though this is not a strictly RS-232 signal, it is included on RS-232 adapters.	Common
Data Set Ready (DSR)	The DCE uses this signal to indicate it is ready to receive commands or data.	DCE to DTE
Request to Send (RTS)	The DTE requests that the DCE prepare to receive data.	DTE to DCE
Clear to Send (CTS)	The DCE shows that it is ready to accept data using this signal.	DCE to DTE
Ring Indicator (RI)	This signal is used to indicate the DCE has detected an incoming ring signal on the telephone line.	DCE to DTE

Stellenbosch University https://scholar.sun.ac.za

Mechanical Characteristics

The RS-232 standard specifies the mechanical characteristics for RS-232 interfacing. This is realized in a 25-pin connector. The DCE connector has a male connector housing and female pins. The DTE connector has a female housing with male pins. As not all of the pins are used in all applications, smaller connectors have been developed [27]. The 9-pin DB9S DTE connector was used in this project and is shown in Figure 3-6.

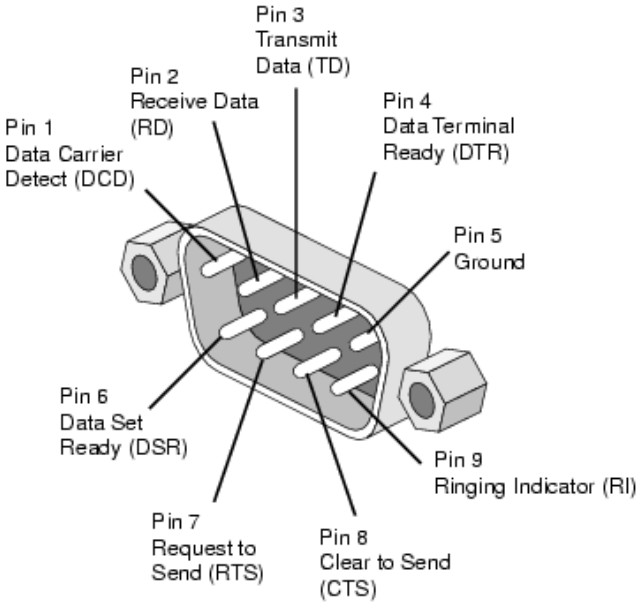


Figure 3-6 - DB9S DTE Connector [28]

The BB serial voltage levels are different to the RS-232 standard voltage levels and must be regulated to enable serial communication. To achieve this regulation, a serial converter, namely the Digilent PMOD RS-232 converter, is used. The converter is shown in Figure 3-7.

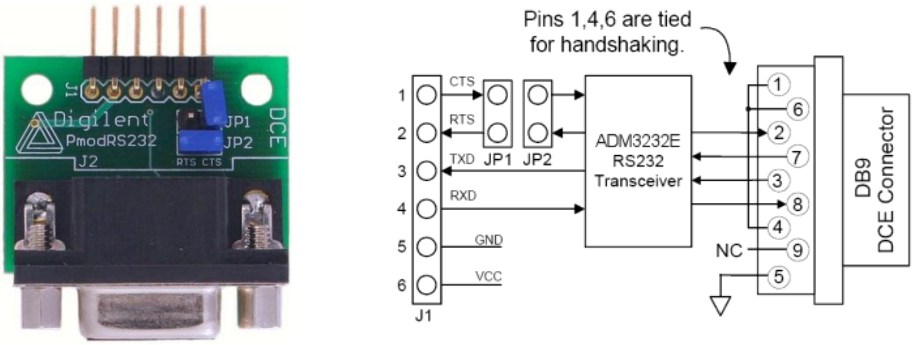


Figure 3-7 - Digilent PMOD RS-232 Converter

As can be seen from Figure 3-7, the converter has six pins.

- CTS (Clear to Send) and RTS (Request to Send) are used as a handshaking mechanism to ensure that a message may be sent serially. These pins are connected to one another to ensure that messages will always be sent.

Stellenbosch University https://scholar.sun.ac.za

- TXD is the Transmit Data pin and RXD is the Receive Data pin. Data sent by the converter leaves via the TXD pin and data is received by the converter via the RXD pin. The TXD pin is connected to the UART4_RXD pin (pin number eleven on the P9 header) of the BB. The RXD pin is connected to the UART4_TXD pin (pin number thirteen on the P9 header) of the BB. These connections ensure that data can be sent to, and received from the converter using UART4 on the BB.
- The GND and VCC pins on the converter represent Ground and Reference Voltage respectively. The GND pin is connected to the GND pin (pin number one or two on the P9 header) of the BB. The VCC pin is connected to the DC_3.3V pin (pin number three or four on the P9 header) of the BB.

The pin connections on the BB are shown in Figure 3-8.

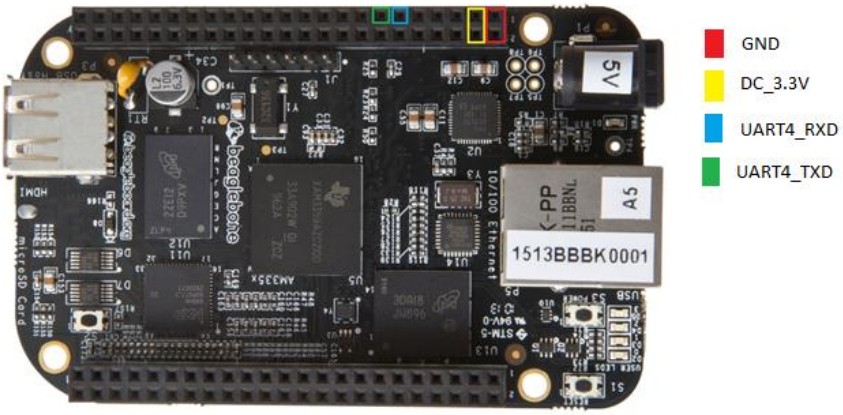


Figure 3-8 - BeagleBone Black Serial Connection Pins

The RS-232 converter is connected to the RS-232 port on the SRAPET controller board to allow serial communication between the BB and SRAPET controller to take place. The wiring diagram is given in Figure 3-9.

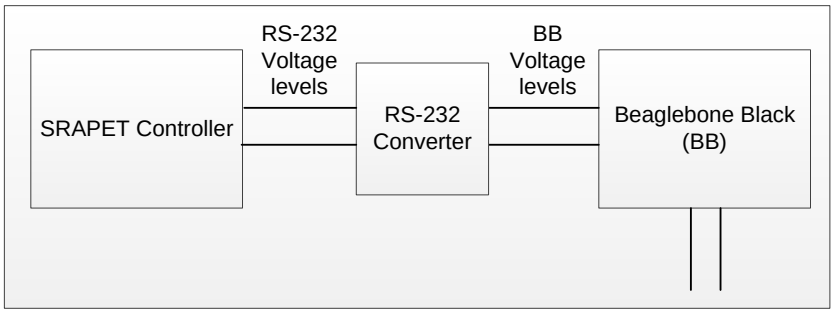


Figure 3-9 - RS-232 Wiring Diagram

Stellenbosch University https://scholar.sun.ac.za

3.2.3 3G Modem

3.2.3.1 3G Mobile Communication

3G (Third Generation) is a term that was coined to indicate the next generation of mobile technologies replacing 2.5G mobile technologies. The service offers a minimum transmission rate of 144 kbits/s in an outdoor mobile environment and a minimum transmission rate of 2 Mbps in a controlled indoor environment [29].

3.2.3.2 Huawei E3131B Modem

The Huawei E3131B modem was selected as the 3G USB modem to be used in the ST project. The E3131B is a modem with a USB 2.0 interface and a slot to support an external antenna. An external antenna was connected to the modem to allow better connectivity by placing the antenna outside of the transformer box. The modem can connect to the internet using either GPRS or 3G technology. The modem is shown in Figure 3-10.

Figure 3-10 - Huawei E3131B USB 3G Modem [30]

3.2.4 Power Supply

Using the BB USB interface as a power supply limits the current supplied to the board to 500 mA max as typical USB ports have this limitation [25]. This is not enough to effectively power the BB, RS-232 converter and a typical 3G Modem. An alternative power supply was thus incorporated into the system. The power supply is controlled by the SRAPET controller board and resets the STCD automatically if a fault occurs on it.

3.2.5 Data Server

The data server is a server computer that runs data storage and analysis software. The physical machine is located in the Power Electronics Group (PEG) laboratory at Stellenbosch University. The main objectives of the Data Server are:

- To act as a DNP3 master station for ST units.
- To effectively gather and store data from the ST.
- To analyse data gathered from the ST and perform calculations on the data enabling predictions to be made from the data.

- To effectively display data and analysis results to users.

The data server computer is a Dell Optiplex 9010 running the Windows Server 2008 R2 operating system. The OSIsoft PI Server data management system is run on the computer for data gathering, storage, analysis and visualization and will be discussed in greater detail in CHAPTER 4 of this document.

3.2.6 Hardware Overview

The STCD assembly is shown in Figure 3-11. It consists of a see-through base plate onto which all the BB components are mounted. (A see-through base plate is used to prevent the BB assembly from obscuring the existing SRAPET components underneath it.)

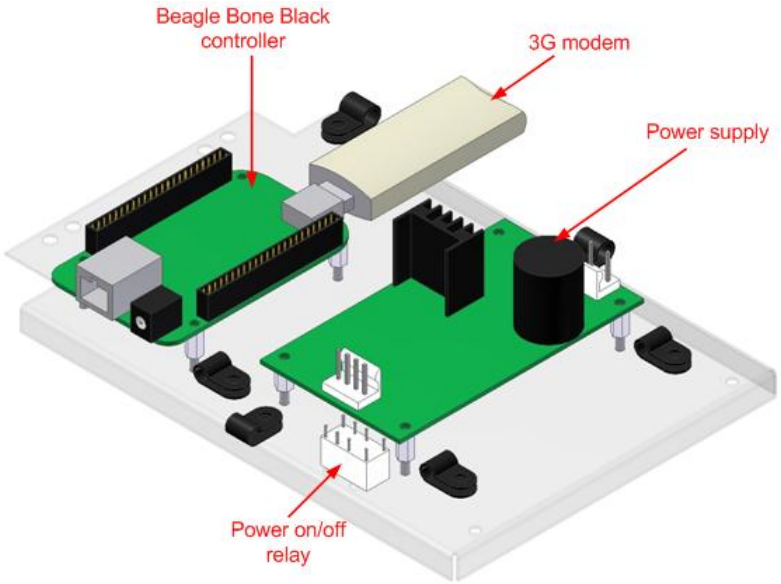


Figure 3-11 - BeagleBone Black with power supply assembly (STCD)

To add the BB assembly to the SRAPET, while still providing access to the SRAPET controller, required construction of a hinged assembly, with the two controllers mounted on top of each other. The assembly is shown in Figure 3-12.

Stellenbosch University https://scholar.sun.ac.za

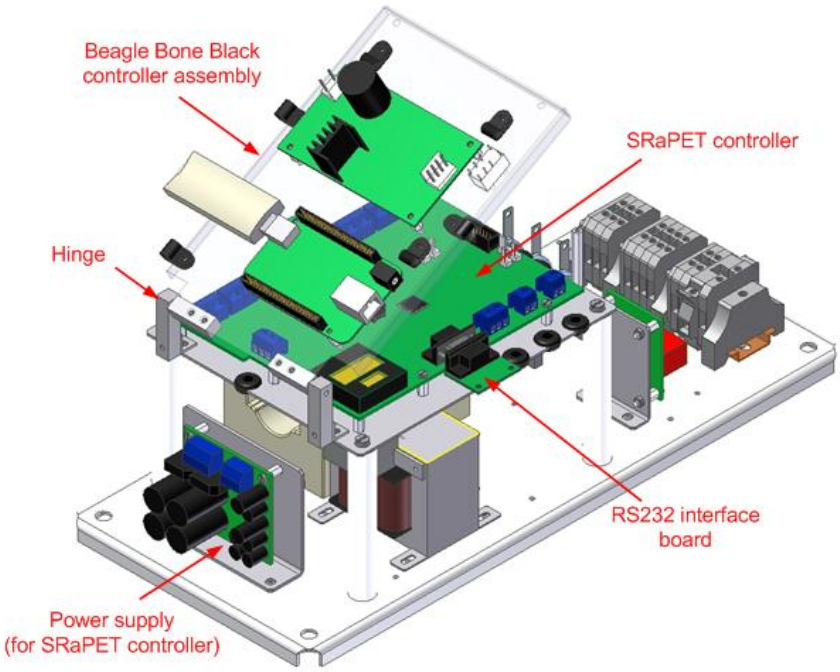


Figure 3-12 - SRAPET base plate assembly

For the first ST prototype, it was decided to retain the communication capabilities of the existing SRAPET for the time being, and adding the STCD in parallel. Therefore, an additional antenna was required on the outside of the ST to obtain cellular signal reception for the 3G modem connected to the BB controller.

The ST prototype is shown in Figure 3-13.

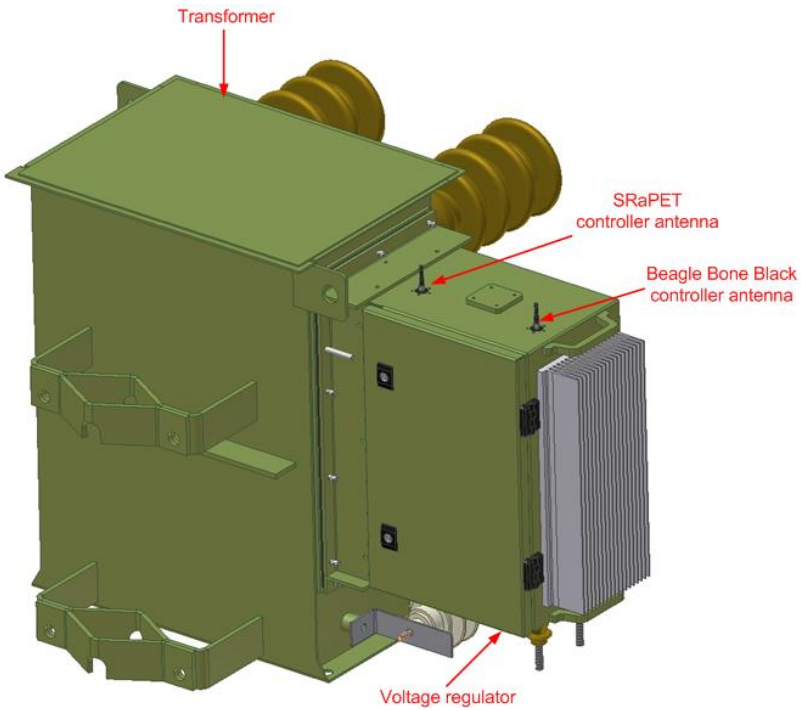


Figure 3-13 - Smart Transformer Prototype

Stellenbosch University https://scholar.sun.ac.za

3.3 SOFTWARE

This section documents the software development that took place for the STCD.

To reach the set goals as tabulated in Table 2-4, various software tools were implemented and custom developed on the STCD. The ARM processor found on the BB enables the use of the Linux operating system for development. Various third party Linux software packages were used to achieve functionality required to meet objectives set out for the STCD.

To satisfy the Eskom standard, the following objectives were met:

- The STCD runs an operating system capable of supporting the various required functionalities including:
    - Authenticated access to the device.
    - Remote access via the SSH and FTP protocols.
    - Handles hardware ports for easy access and use.
    - Implements a file management system.

  The implementation of the operating system contributes to the satisfaction of points 2 and 7 of Table 2-4.

- A custom program named Main Program (MP) enables and facilitates serial communication between the STCD and the SRAPET controller board as well as DNP3 communication between the STCD and the data server. The MP is implemented to satisfy point 1 in Table 2-4.
- An HTTP web interface is hosted to provide remote access for users to the STCD. The web interface is implemented to contribute to the satisfaction of point 2 in Table 2-4.
- A XML parser program monitors a folder and updates configuration settings when a valid file is placed inside of it. This effectively creates an XML interface that a user can utilize to configure the STCD. XML configuration files are copied to the relevant folder from an external computer using the FTP protocol. The XML interface is not strictly required by the Eskom standard, but the author believed it to be a useful extra interface that contributes the functionality required by point 2 in Table 2-4.
- SNMP version 2C implemented for communication and vital statistics monitoring. The implementation of the SNMP protocol satisfies point 3 in Table 2-4.
- A set of two programs collectively known as *ModemTalk* gathers valuable data from the 3G modem and allows the SNMP interface to access and publish that data. The ModemTalk programs contribute to meeting sub-requirements of point 3 in Table 2-4.
- Network Time Protocol (NTP) is implemented for the synchronization of internal clocks. Thie implementation of NTP satisfies point 4 in Table 2-4
- Watchdog mechanisms are implemented to ensure that the system continuously functions correctly.

Stellenbosch University https://scholar.sun.ac.za

- Mobile internet connectivity implemented with various software tools. The implementation of a mobile internet connection satisfies point 5 in Table 2-4
- Virtual Private Network (VPN) software implemented to ensure a reliable and secure connection to the data server. The implementation of a VPN satisfies point 5 in Table 2-4
- A MySQL database acts as a communication medium between the MP, the web interface and the XML interface. The database is also used for storing configuration settings, logging information and measurement data.

3.3.1 Software Configuration

The software configuration illustrates how software programs within the STCD interact with one another. Each software component is described in greater detail in various section of this document. The software configuration is shown in Figure 3-14.

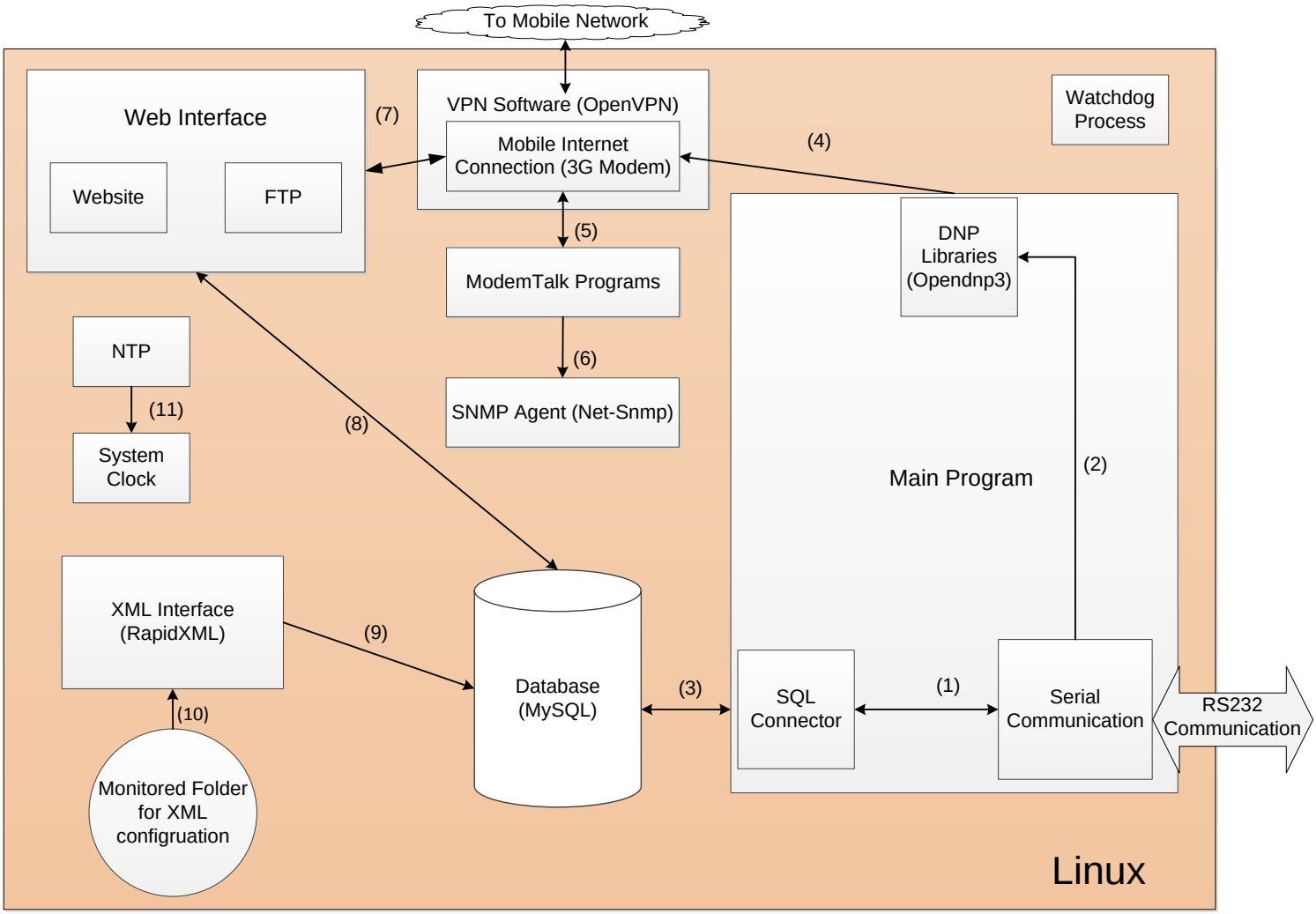


Figure 3-14 - Local Software Configuration

The software components are described in Table 3-3, with the communication links between the components described in Table 3-4.

Stellenbosch University https://scholar.sun.ac.za

Table 3-3 - Local Software Configuration Components

Software Component	Description
Linux	The STCD runs Linux as its operating system. Linux acts as a software foundation and provides a file system and a wide variety of functionalities and features to build the STCD system upon.
Main Program	The main program is written in C++ and facilitates serial communication with the SRAPET controller board and DNP3 communication with the data server. The program also interfaces with the MySQL database for logging and communication purposes.
Serial communication	The main program facilitates serial communication with the SRAPET controller via a serial UART using the RS-232 protocol. A RS-232 converter is used to convert STCD serial messages to RS-232 messages.
Web interface	The web interface is the main interface by which users remotely access the STCD. The web interface consists of a web server that hosts a website accessible via internet browser software. The Mongoose web server software was used to implement the web interface.
XML interface	The XML interface monitors a folder in the Linux file system. When a user uploads an XML configuration file to this folder, the XML interface updates the STCD configuration to the configuration specified in the uploaded file. Files are uploaded to the monitored folder using the FTP protocol. The XML interface consists of a XML parser program and a folder monitoring program. The RapidXML libraries are used in a custom-coded program to implement the XML interface.
SNMP Agent	The SNMP agent gathers data from STCD hardware, software and communication channels and publishes the data for monitoring purposes with the SNMP protocol. The Net-Snmp software agent was implemented as the SNMP agent on the STCD.
ModemTalk	The ModemTalk program interfaces with the mobile internet modem to gather important data and make it available on the SNMP agent.
Database	The database acts as a communication medium between the MP, the web interface and the XML interface. It is also used for data logging and storing of configuration information. The MySQL database software was implemented as the database on the STCD.
Mobile internet connection	Software compatible with Linux is used to establish and maintain a connection via a 3G modem, connecting the STCD to a network. The modem can establish either a 3G or a GPRS connection to the internet.
VPN software	VPN software is used to securely connect the STCD to a private network for ease-of-access and security purposes.

Stellenbosch University https://scholar.sun.ac.za

Watchdog process	The watchdog processes monitor programs to ensure that all of them are functioning properly. If one of them is malfunctioning, relevant action is taken to correct the problem.
Network Time Protocol (NTP)	The network time protocol is used to synchronize the STCD system clock to an external time server.

Table 3-4 - Local Software Configuration Communication Links

Communication Link	Description
Serial data to the database **(1)**	Measurements and other data read serially from the SRAPET controller are stored in the database for logging purposes. Instructions may also be written serially to the SRAPET controller if desired by the user. The user uses the web interface to change values in the database, which are then written serially to the SRAPET controller.
Serially received measurements are packaged for DNP3 **(2)**	DNP3 libraries are used to communicate measurement data received serially from the SRAPET controller board to the data server as specified by the user.
Communication between the MP and the database **(3)**	Measurement and operational data from the MP are stored in the database. The database is also used as a channel for the user to communicate with the MP via either the web interface or the XML interface.
DNP3 communication over the network **(4)**	The mobile internet connection is used to send DNP3 data over the network to the data server.
Mobile internet modem communication with ModemTalk programs **(5)**	The ModemTalk program gathers valuable data from the mobile internet modem and makes it available to the SNMP agent.
ModemTalk interface to SNMP agent **(6)**	The ModemTalk programs make data concerning the mobile internet modem available to the SNMP agent by writing the data to text files. The SNMP agent monitors the text files and publishes the information in them over the network using the SNMP protocol. The SNMP agent also publishes hardware, software and networking data via the SNMP protocol.
Web interface communication over network **(7)**	The web interface consists of a webserver that hosts a website. The website can be accessed by a user over the network. The user can monitor STCD data, change STCD configuration settings and send instructions to the STCD via the website. The website is accessed using a web browser.
Communication between the web interface and the database **(8)**	The database is used as a communication channel from the web interface to the MP and vice versa.

Stellenbosch University https://scholar.sun.ac.za

XML interface folder monitoring **(10)** and communication with the database **(9)**.	The XML interface monitors a folder. When an XML configuration file (with correct syntax) is placed inside the folder being monitored, an XML parser interprets the configuration file and updates configuration settings in the database according to what the user has specified. It also informs the MP that a change in configuration information has taken place.
Network Time Protocol (NTP) synchronization **(11)**	The network time protocol synchronizes the STCD system clock using software methods.

3.3.2 Linux

Linux is an open-source operating system developed by Linus Torvalds in the early 1990s. Linux offers a wide range of functionalities for many different applications in almost all spheres of computer usage, from engineering to office use. It is an operating system characterized by its no-cost availability, speed, efficiency, scalability and flexibility and is widely implemented in network servers, workstations and embedded applications.

Linux is built on the Linux kernel; the core program behind the operating system. Many different distributions of Linux have been built upon this foundational kernel, each characterized by a unique selection of programs, functionalities and implementation purposes. Some distributions specialize in providing a stable desktop environment for a PC, while other distributions are developed for use in embedded systems where resources are scarce and must be optimally used, etc.

All Linux distributions offer a *shell* and a *file system.* The *shell* is an interactive and non-interactive command-line (text based) interpreter that a user can use to perform various tasks on the system. Interactive because a user can enter instructions via the command line to perform tasks and view information, non-interactive because operations can be run in the background whilst the user attends to other tasks. Common shell programs include the Bourne Again Shell (BASH), the Z shell and the TCSH shell, with the BASH shell being the default installed on most Linux distributions. The BASH shell was used for the STCD as it is the most widely used shell program incorporates the BASH scripting language that was used to perform various tasks.

The Linux *file system* is a hierarchically connected structure of files and directories. A user may access, edit, rearrange and create files and directories from the shell. The shell provides effective tools for a user to easily perform these operations. The hierarchical structure of the Linux file system begins with the root directory simply known as “/”. Many other directories branch from the root directory. Root may be seen as the base directory of the file system, within which all other files and directories are contained [31].

The Archlinux Linux distribution was selected for use in the STCD. Archlinux is a Linux distribution that focusses on simplicity, minimalism and code elegance. The simplicity and light-weight design

Stellenbosch University https://scholar.sun.ac.za

of Archlinux make it suitable for embedded projects as resources are used as sparingly as possible.

Other Linux distributions including Ubuntu, Debian and Ångström were considered and tested. These options were all rejected. Ubuntu was found to not be light-weight enough in default software package sizes. Ångström was the first operating to be considered for use in the STCD and was developed on for more than a year during the project. Ångström was found to be a good solution in the areas of performance and being light-weight. Ångström does not, however, support all necessary software that the STCD requires. Debian and Archlinux are both valid options for the STCD operating system. The decision to use Archlinux was based on it being more light-weight and configurable than Debian.

For more on the Linux operating system please see APPENDIX B.

3.3.2.1 Bash Scripting

The BASH (Bourne Again Shell) scripting language is a programming language that can interact with the Linux operating system. BASH may be used to automate the execution of tasks on the Linux operating system.

BASH offers various functionalities that can be utilized by developers including: input, output, variables, logic operations, loops, logical comparisons etc. [31].

Various tasks are performed on the STCD using BASH. BASH scripts are used in partnership with a program called *systemd* to create Linux *services* for various processes running on the STCD.

3.3.2.2 Systemd

Systemd is a system management tool designed for Linux. Systemd allows users to create *services* that execute and monitor tasks. A service executable can be any executable action that a user wishes to perform in Linux. Systemd starts the service by executing the action and monitors the services by continuously checking the state of the action [32].

Various services were created for the STCD. Each service performs a specific task on the STCD. Services are used in partnership with BASH scripts to perform tasks. Systemd also keeps services alive upon user request, restarting processes when they exit. The various services and their functionality are discussed in the relevant sections of this document.

3.3.3 Startup Service

The startup service is a systemd service that runs upon STCD startup. It performs two tasks:

- Prepares the BB UART for serial communication.
- Runs the Network Time Protocol (NTP) program to synchronize the STCD system clock with a national time server.

Stellenbosch University https://scholar.sun.ac.za

3.3.4 The Database Manager Service

The database manager is a service that manages the STCD MySQL database. The database manager calls a BASH script that performs the following tasks:

- Logs STCD reboots in the MySQL database.
- Checks MySQL database integrity.
- Repairs the MySQL database if there is a fault.
- Writes ST measurement entries to time-stamped files and truncates (clears) MySQL tables to prevent them from becoming saturated.

The MySQL database software is installed on the STCD to create a local MySQL database and allow the usage thereof. When installed, the MySQL software creates a service known as *mysqld*. Mysqld starts the MySQL database upon system startup and then maintains it. The mysqld and database manager services work together to ensure that the MySQL database starts up and continues to function properly. It should be noted that the database manager service only checks the database once every hour.

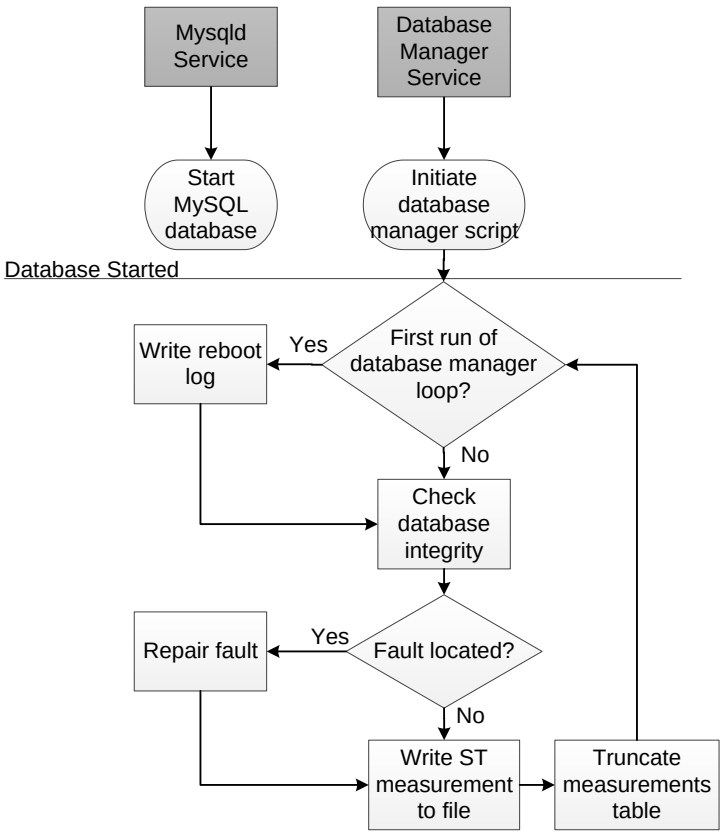


Figure 3-15 - Database Manager Service

3.3.5 MySQL

The MySQL relational database was developed by Michael Widenius and David Axmark and was first released to the general public in 1995. It has since become an extremely popular database solution with over four million users worldwide, including companies such as NASA, Cisco and

Stellenbosch University https://scholar.sun.ac.za

Google. The database software proved to be both reliable and scalable, showing good and lasting performance [33].

A database consists of data entries contained in tables. A table may have various fields that characterize an entry contained inside of it. Individual entries are usually identified by a key or primary key entry that is unique to each entry. A database may contain many tables and a table many unique entries [34].

Structured Query Language (SQL) is a programming language built for extracting information from a database or for modifying the database. It provides syntax similar to English that allows users to easily perform desired tasks on databases such as extracting entries from a table, updating entries, creating a new table etc.

Some important MySQL commands that are used in the STCD software are:

- *SELECT* – Selects and returns data defined by user-entered criteria from a table.
- *UPDATE* – Update values of entries contained in a table.
- *INSERT INTO* – Used to insert new entries into a table.
- *TRUNCATE TABLE* – Removes all entries from a specified table.
- *REPAIR TABLE* – Repairs a possibly corrupted table.
- *CREATE TABLE* – Creates a new table.
- *CREATE DATABASE* – Creates a new database.

3.3.5.1 The STCD Database

As mentioned in previous sections, the database acts as a communication channel, a logging tool and a store for STCD configuration and operational information. The database contains a number of tables, each fulfilling a specific function. The tables contained in the database and a brief explanation of their functions is given here.

- **counters –** This *counters* table is used to store counters for tracking how many times errors have occurred in the MP. Once a predefined number of reoccurring errors is reached, the MP is restarted.
- **deltas –** The *deltas* table stores Level Crossing Sampling (LCS) information for the MP. For more on LCS, see section 3.3.9.2.
- **deviceinfo –** The *deviceinfo* table stores device information of the connected SRAPET transformer.
- **login –** The *login* table stores user authentication information for the web interface.
- **loginstats –** The *loginstats* table stores login statistics for the web interface.
- **prestart –** The *prestart* table is used as a flag for restarting the MP. When a value in this table is set high and detected by the MP, the MP is restarted and the value is set low.
- **averageBuffer –** The averageBuffer table stores measurement readings taken from the SRAPET controller. Once a user-defined time has elapsed, the entries stored in this table

Stellenbosch University https://scholar.sun.ac.za

are averaged and the averages are returned to the MP. The entries stored in averageBuffer are then copied into the *readings* table and the averageBuffer table is truncated.

- **readings –** The *readings* table logs measurements received from the SRAPET device. The first entry in the table is overwritten at every write to the table and is always the latest measurement received. Data is copied from the averageBuffer table to the readings table to log previous readings.
- **reboots –** The *reboots* table logs the amount of system reboots that have taken place.
- **sync –** The *sync* table is used as a flag to indicate when time synchronization with the SRAPET controller board is requested by the user via the web interface. This initiates an action that synchronizes the STCD time to that of the SRAPET controller.
- **xmlconfigstats –** The *xmlconfigstats* table logs the number of events where a user has updated configuration settings using the XML interface.

To enable communication between various programs on the STCD to take place, a *change* field is included in relevant tables. The *change* field is of integer type and is used to indicate that data contained in the table has changed. Programs monitor relevant MySQL tables and check when the value of *change* field is set to one. When this takes place, the program knows that the values contained in the table have changed and relevant action is taken. After all desired actions have been performed, the monitoring program resets the change field's value to zero.

An example is taken from the *deltas* table. This table contains entries specifying the LCS information for each data point on the STCD. The table also contains a change field. When a user updates the *deltas* table via the web interface by configuring the LCS information for each data point, the *change* field in the *deltas* table is set to 1. The MP monitors the *change* field, and when it changes to 1, LCS values in the MP are updated from the *deltas* table and the *change* field is reset to 0.

3.3.6 Simple Network Management Protocol

The Simple Network Management Protocol (SNMP) was developed to manage Internet Protocol (IP) devices. This is done by fetching relevant data from underlying hardware on a device, packaging this data, and sending it over a network to a client device. The protocol allows for remote viewing and editing of various networking and resource related data found on a device [35].

As in DNP3, SNMP also enforces a master-outstation reporting philosophy. The master requests data from an outstation and the outstation responds with the data requested if it is available. In cases where significant events occur on an outstation, the outstation may also send relevant data to a master spontaneously indicating that an event has taken place without a master requesting it. These spontaneous messages are known as unsolicited messages.

The master uses a text-based file called a MIB (Management Information Base) located on both itself and the outstation to gain knowledge about what data it can access on an outstation. A MIB

Stellenbosch University https://scholar.sun.ac.za

defines what variables are available on an outstation, what data types those variables are, and where the master can find them (at what addresses). It may be seen as a database of managed objects that the outstation is monitoring. SNMP supports a variety of data types that data can be stored in. These include integers, strings, counters, IP address variables etc. [35]. A basic diagram of SNMP communication between an outstation and a master is shown in Figure 3-16.

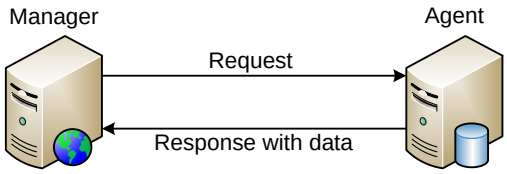


Figure 3-16 - Communication between a SNMP Master and Outstation

A master accesses data on an outstation by requesting data from a specific address on that outstation. This address is known as an OID (Object Identifier). The OIDs on the outstation are structured in a tree format. The master must “follow a certain path” down the tree structure via “branches” to reach its desired destination node. This concept of OIDs is illustrated in Figure 3-17.

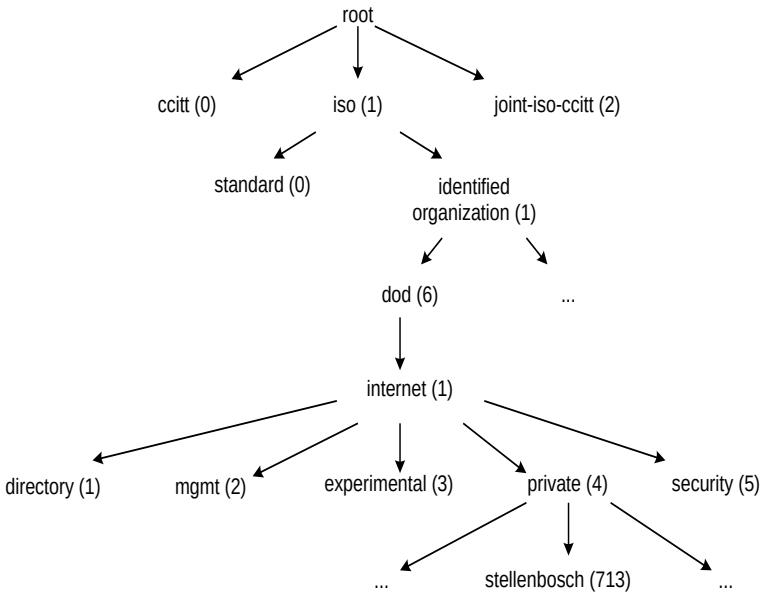


Figure 3-17 - OID Tree Structure

An OID is a numeric representation of the path followed to get to a certain data node (e.g. .1.1.6.1.4.713.1). When a master requests a variable at such an address from an outstation, the outstation interprets the address and checks to see if there is any data available at that location. If there is data, it sends the data to the master. If there is no data, it sends an error message.

When a private organization creates a custom MIB for one of their products containing information on the data available via SNMP for that product, they can register the OID addresses that their MIB references on a public OID tree maintained by IANA (Internet Assigned Numbers Authority) to avoid duplication. Thus, a company may register a MIB at certain addresses on an international

Stellenbosch University https://scholar.sun.ac.za

tree structure, but the contents of the MIB need not be made public. The MIB is registered to ensure that no other developers use the allocated addresses to store information at. If a device is developed with various software components developed by various companies, the public OID tree acts as a method to make sure that multiple sources of information is not published to the same OID address on an SNMP-enabled device.

Not all devices will contain functionality to return the data specified in MIBs created by companies, but only those for which their SNMP agents have been developed with functionality supporting those MIBs. Additional C code must be written to support all custom-developed variables [35].

A string name may also be given as an alias to represent a certain OID address. For example, the name “enterprises” is commonly used as an alias for the OID 1.3.6.1.1.4.1. Stellenbosch University is in possession of the OID branch { enterprises 713 }.

Managed objects may be stored in tables. Tables contain fields that describe entries in that table. Each field in a table contains data that describes the relevant entry.

Data is extracted from an SNMP outstation by using SNMP operation messages. One such message is the *get* message. The *get* operation is initiated by a master and is sent to an outstation. The function of this message is to gather data at a specific OID and return it to the master. The outstation interprets the *get* message and responds with a *get-response* message containing relevant data if it is present. The *get-next* operation allows a user to retrieve a group of values from an outstation. When a master issues a *get-next* command, the outstation traverses its OID tree in lexicographic order and returns all data that branches from a root node specified in the command [35].

The SNMP *set* operation is used to change the value of an existing managed object or to create a new row in a SNMP table. A managed object must be made accessible for writing in its MIB file to allow usage of the *set* operation [35].

Various SNMP agent software packages exist that can be utilized to enable SNMP communication to and from a device [35]. The Net-Snmp software package was selected for application on the STCD.

3.3.6.1 SNMP Versions

Various versions of the SNMP protocol have been developed, and these versions are described in documents known as Requests for Comments (RFCs). Responsibility for defining the standard protocols that govern internet traffic falls to the Internet Engineering Task Force (IETF). The IETF defines specifications for many protocols that are related to IP networks. These specifications are the RFCs. A RFC moves through three stages before being recognized as a standard: proposed status, draft status and finally standard status. There are not many RFCs that have reached standard status.

Stellenbosch University https://scholar.sun.ac.za

SNMP has three versions, of which only one has been given IETF standard status. A short description of each version is given [35]:

- SNMP Version 1 (SNMPv1): The first and only version of SNMP to reach IETF standard status thus far. It supports a basic security structure that works with plain-text string passwords. The version is defined in RFC 1157 [35], [36].
- SNMP Version 2 (SNMPv2): Follows the same framework as SNMPv1, but added improvements in security, confidentiality and supports 64 bit counter variable types. Some vendors offer support for SNMPv2 even though it has not reached IETF standard status. SNMPv2 is defined in RFC 1905, RFC 1906 and RFC 1907. [35].
- SNMP Version 3 (SNMPv3): Version 3 adds support for private communication and stronger authentication between managed devices. SNMPv3 will be the next version to reach full IETF standard status. SNMPv3 is defined in RFC 1905, RFC 1906, RFC 1907, RFC 2571, RFC 2572, RFC 2573, RFC 2574 and RFC 2575 [35].

SNMPv2 was required by the Eskom standard and implemented on the STCD [4].

3.3.6.2 More on MIBs

As mentioned, a MIB may be seen as a database of managed objects that an agent is monitoring [35]. MIB information is stored in a text file that uses Structure of Management Information (SMI) syntax to define the managed objects stored in the MIB. There are various versions of SMI, and SMI version 2 was used to develop the MIB for the STCD. For more on SMI syntax, see APPENDIX C.2.

The SNMP agent software used for the STCD supports a default set of MIB modules. The data variables supported by default may be viewed on the Net-Snmp website [37]. By default, the Net-Snmp agent supports the IF-MIB module that allows interfaces connected to the STCD to be monitored. Some of the data contained in the IF-MIB module met requirements of the Eskom Remote Device standard, and were consequently used [4].

The Eskom standard for remote devices requires that certain GSM-related MIB data be supported by remote devices on the Eskom network [4]. Developers are required to design custom MIBs and agent extensions to support the required functionality. A custom MIB and agent extension code was written to equip the STCD with the required SNMP functionality. The custom MIB file contains variables to support the following network-related data:

The notation used to describe the variables, looks as follow:

- *descriptive name (variable name): Description*

- *GSM Network Operator* (mobGsmNetOp): The GSM Network operator name.

- *Network Attachedment* (mobNetAttach): The type of network attachment (e.g. EDGE, HSDPA)
- *Signal Strength* (mobSigStr): The current signal strength experienced by the GSM modem.
- *IMEI* (mobModImei): The International Mobile Equipment Identification (IMEI) number.
- *Sim Serial* (mobSimSerial): The SIM serial number.
- *Modem Manufacturer* (mobModManufac): The name of the modem manufacturer.
- Modem *Model* (mobModModel): The name of the modem model.
- *Firmware Version* (mobFirmVer): The firmware version installed on the modem.

Other variables were also required by the Eskom Remote Device standard. These variables are contained in MIBs supported by Net-Snmp by default and did not require custom development. The required variables, and the respective MIBs that they are contained in, are listed below:

- *Sent and Received bytes and Packets* (ifInOctets, ifOutOctets, ifInUcastPkts, ifOutUcastPkts, ifInNUcastPkts and ifOutNUcastPkts): Sent and received bytes and packets for the GSM connection, contained in the ifTable of the IF-MIB module.
  (Note: UcastPkts are packets sent to and received from a subnetwork-unicast address, meaning that it is sent to or received from a single network device. NUcastPkts are packets sent to or received from a non-unicast address, meaning they are broadcast or multicast messages [38], [39].*)*
- *Operational state of interfaces* (ifOperState): The status of all interfaces connected to the STCD (Up or down). Contained in the ifTable of the IF-MIB module.
- *Device Uptime* (sysUpTime): The time since the network management portion of the system was last re-initialized. Contained in the system table of the MIB-II module.
- Interface Uptimes (ifLastChange): The value of sysUpTime at the time when the interface entered its current state. Contained in the ifTable of the IF-MIB module.
  (Note: This value has a direct relation to the sysUpTime variable. If an interface has an operational status value of up, this variable will record the uptime of the interface. If an interface has an operational status value of down, this variable will record the downtime of the interface.) [40], [4].

3.3.6.3 Net-Snmp

Net-Snmp is a free SNMP agent created for Linux and Windows. Net-Snmp enables a device to publish SNMP data that can be read by a SNMP reader client program. Net-Snmp contains a default data-set that has been set-up for basic SNMP use and is compatible with most devices running Linux or Windows as an operating system. Most of the default modules are generic and will automatically detect and publish data unique to the device that the agent is installed on [35].

Stellenbosch University https://scholar.sun.ac.za

3.3.6.4 Installing Net-Snmp on the STCD

The Net-Snmp source code can be downloaded from the Net-Snmp website [41]. The Net-Snmp source code comes with a configure script which may be run to create a Makefile. The Makefile may in turn be run to compile and install the program. Not all of the default MIB modules supported by Net-Snmp were required. Thus the agent was installed without the unnecessary modules. These modules are excluded by using the "--without-mib-module" option when running Net-Snmp's configure script.

3.3.6.5 Extending Net-Snmp

The Net-Snmp program contains a module for extending the agent to support custom MIB modules. This allows developers to gather desired data from underlying hardware or software and publish this data so that it may be read using a SNMP reader program. The extension module only has functionality to publish gathered data on the Net-Snmp agent, the gathering of this data is left to the developer [35].

The developer is required to construct a custom MIB file. This file contains the structure of the data that the developer wants to publish. There is no supporting software in the Net-Snmp agent to help with constructing the MIB file, and this must be done in a text-editor by the developer using SMI syntax. The MIB file constructed to support the custom functionality required by the Eskom standard was named *PEG-MIB* and may be viewed in APPENDIX C.4. The PEG-MIB file structures data in a non-table format, publishing variables at OID addresses under the Stellenbosch-owned branch of the OID tree (enterprises 713).

3.3.6.6 Loading a custom MIB into Net-Snmp

Custom MIB modules must be added to the Net-Snmp agent for the agent to recognize the OIDs contained in them. MIB modules can be added to the agent by placing the custom MIB file in the default MIB directory of the agent device (/usr/share/snmp/mibs on the STCD), and using the *"+m "[MIB name]""* option when executing a Net-Snmp command. For example:

snmpwalk –v1 –c public +m "PEG-MIB" localhost [35]

Note: For more on Net-Snmp commands, see APPENDIX C.3.

The name given as the +m argument must match the name in the first line of the MIB file:

e.g. *PEG-MIB DEFINITIONS ::= BEGIN (see APPENDIX C.2 for SMI syntax explanation)*

If the MIB syntax is correct, the Net-Snmp agent will then recognize the OIDs contained in the MIB as the data variables specified in the MIB.

3.3.6.7 Mib2c

The Net-Snmp software includes a program called *mib2c*. Mib2c requires a complete MIB text file as input, and produces framework C code for the extension of the Net-Snmp agent as output.

Stellenbosch University https://scholar.sun.ac.za

Various configuration files developed by the creators of Net-Snmp may be used by the program to act as a reference for the structure of the C code to be generated. The program is run in the Linux shell using the following command syntax:

mib2c –c [configuration file] [custom MIB]

The configuration files are contained in the /usr/share/snmp directory on the STCD. The *mib2c.scalar.conf* configuration file was used to create a C code framework for variables that are not contained in a SNMP table. The generated C code contains:

- OID assignments for the custom variables.
- Functions that register the custom variables as scalars on the Net-Snmp agent.
- Functions that handle requests for the custom variables received from SNMP clients.

The C files (pegmib.c and pegmib.h) used to extend the Net-Snmp agent on the STCD may be viewed in APPENDIX C.5 and C.6. Code to gather desired data from the underlying hardware is included in the custom variable handler functions generated by the *mib2c* program and modified for the STCD. Simple text files were used to store data gathered from the STCD. Desired data from the 3G modem is gathered using self-written C++ programs that communicate with the modem using AT Commands.

3.3.6.8 Gathering Data from the Modem

Serial communication is used to exchange data with the mobile internet modem. ATtention (AT) commands are used to communicate with modems. Two C++ programs (ModemTalk.cpp and ModemTalk_D.cpp) were written to write AT commands to the mobile internet modem and to record the responses. The responses from the modem are then analysed and desired data are extracted from them. Data is extracted from the responses by saving the responses in a string variable and using the *sscanf() C*++ function to extract a portion of the string and save it in a different variable.

Data received from the modem is divided into two categories: static and dynamic. Static data is data that does not change with time, while dynamic data does change with time. The only dynamic data gathered from the modem with custom code is the current signal strength experienced by the modem. Static data is gathered by both requesting it from the modem before a connection is established to the internet, and by allowing the user to manually enter data via the STCD web interface. This was done as the modem used is not capable of returning all of the required data.

All data gathered from the modem is stored in text files: one for static data, one for dynamic data and one for manually entered data. These files are contained in the */var/snmphook* directory and are read by the pegmib.c Net-Snmp extension code and published on the Net-Snmp agent.

Stellenbosch University https://scholar.sun.ac.za

3.3.6.9 Including the custom C-files into the Net-Snmp agent

The custom C-code files generated using the mib2c program must be included in the Net-Snmp agent to extend it. There are two methods of including custom C files for agent extension. The first is to compile the C files with the Net-Snmp code when installing the Net-Snmp agent. This is done by placing the custom MIB file in the "netsnmp_soure_directory/mibs" directory, placing the custom C-files in the "netsnmp_source_directory/agent/mibgroup" directory and configuring the package with the option:

./configure --with-mib-modules="CUSTOM_MIB_NAME"

This method is tedious as installing the Net-Snmp agent takes a long time and it must be recompiled if any changes are made to the self-written C-files.

A more effective method is used whereby extension is done dynamically after agent installation. This is done by dynamically loading the custom C-files into the Net-Snmp agent. A shared object (.so) file is created from the custom C-files using a GCC compiler *Makefile*. The shared object file is in essence a shared library of functions. This file makes functions available for easy inclusion in other programs [42]. Once the shared library has been created, it is included into the Net-Snmp agent by specifying the location of the library in the Net-Snmp configuration file. For a more in-depth look at this process, see APPENDIX C.1.

3.3.6.10 SNMP Systemd Service

A SNMP systemd service that starts and monitors the *snmpd* executable is created. Snmpd is the executable used to start the Net-Snmp agent.

3.3.7 The Web Interface

The web interface is the primary means by which a user configures the STCD. It possesses adequate functionality to allow for effective management of STCD operations. A web interface was chosen as it does not require any special software to be installed on a user computer other than a web browser. The user can thus easily access an STCD unit, provided the user can supply the necessary authentication details for the unit.

A web interface is a Hypertext Transfer Protocol (HTTP) server hosted locally on a device. A user can access web pages, data or files hosted by the device by navigating to the device's Internet Protocol (IP) address using a web browser such as Internet Explorer or Mozilla Firefox.

A web server is used to host a website. The website consists of multiple web pages that a user may access. These pages contain data and buttons or links that allow a user to view data, perform configuration tasks on the device and navigate the website. The Mongoose web server software package was used to host the web interface on the STCD [43].

The STCD web interface was developed using a mixture of Hypertext Markup Language (HTML) and PHP code that interfaces with the STCD MySQL database. The mixture was necessary as

Stellenbosch University https://scholar.sun.ac.za

HTML code provides the ability to create a web interface layout while PHP code allows the web interface to connect to the MySQL database. The web interface runs as an independent process to the MySQL database and MP.

3.3.7.1 Layout

A website layout refers to the manner in which the website is arranged to achieve desired functionality. The layout chosen for the STCD website focuses on functionality and simplicity. The layout consists of an authentication pane, a navigation pane and a main window. The authentication and navigation panes remain constant, always allowing the user to easily access all areas of the website. The content of the main window changes as the user navigates the website. The authentication pane is used to direct the user to pages related to web interface authentication. The navigation pane is used to direct the user to various configuration and monitoring tools on the web interface.

The web interface is shown in Figure 3-18.

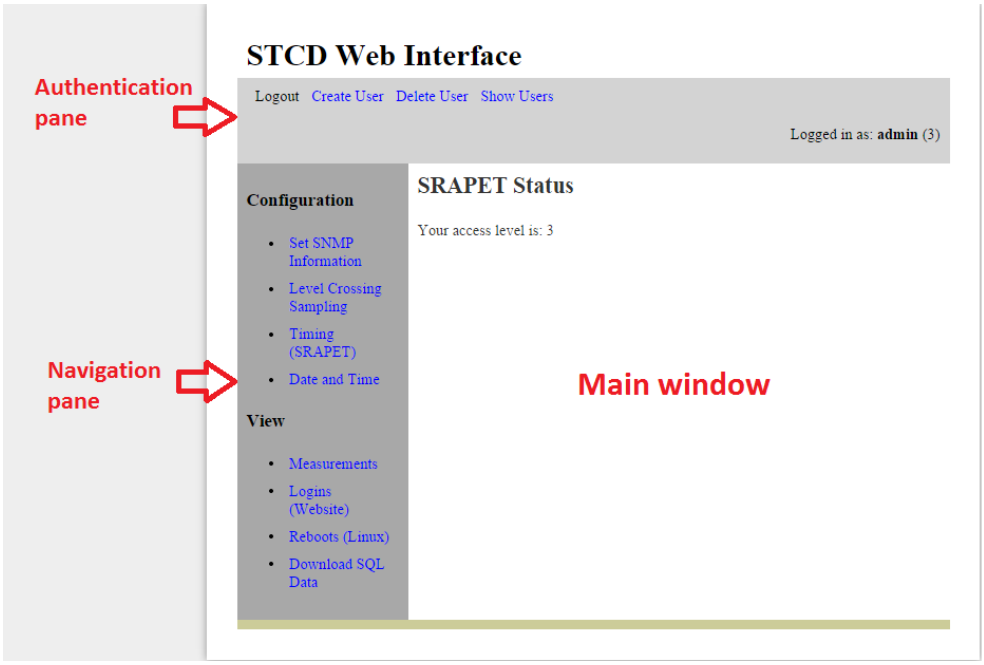


Figure 3-18 - Web Interface Layout

3.3.7.2 HTML

HTML is the base language that web pages are produced with. It consists of a variety of tags that can be used to define page structure and layout for web pages [44]. The layout for the STCD web interface was created using the HTML language.

3.3.7.3 PHP

PHP is a scripting language created by Rasmus Lerdorf in 1995 and developed in the years that followed. The language became extremely popular after the release of PHP 4.0 in 2000, reaching

Stellenbosch University https://scholar.sun.ac.za

around 3.6 million users only a few months after release [33]. PHP can be embedded in HTML websites.

Some of PHP's useful features are listed below [33], [34]:

- **Object-Oriented Capabilities:** PHP offers object oriented programming that includes sub-features such as explicit constructors and destructors, object cloning, class abstraction, variable scoping, interfaces etc.
- **Try / Catch Exception Handling:** PHP includes exception handling techniques found in languages such as C++, C# and Java. This route was followed because creating custom error-handling techniques are error-prone.
- **Encryption and Authentication:** PHP offers native encryption support using algorithms such as Blowfish, MD5 and TripleDES. PHP also offers features that easily allow developers to implement username and password or IP-based authentication systems.
- **Session-handling Support:** PHP offers developers a means for tracking interactions between the user and the website and functionality to create and store variables unique to a session where a user accesses the website.
- **Support for Various Languages:** PHP provides support such as incorporating string parsing functionality from Perl into PHP and offering binding to Java objects.

3.3.7.4 Authentication

The STCD web interface uses role-based authentication. The login screen for the web interface is shown in Figure 3-19.

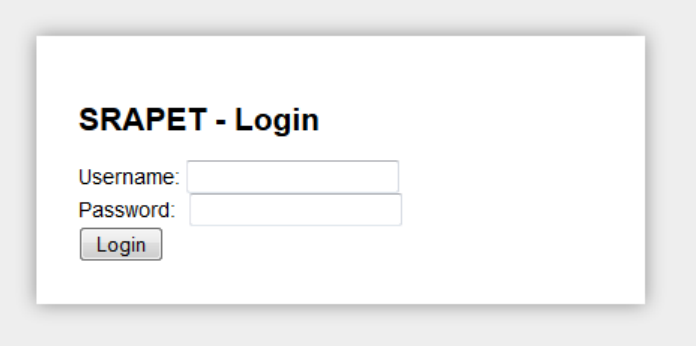


Figure 3-19 - Website Login

Each user account is assigned a level of access upon creation that determines what functionality the user will be able to access on the web interface. This form of authentication is implemented to satisfy point 6 in Table 2-4. There are three levels of access:

- **Administrator –** Has access to all functionality and data on the web interface.
- **Superuser –** Has access to all data and functionality on the web interface, but may not create / remove users or truncate MySQL tables.

Stellenbosch University https://scholar.sun.ac.za

- **Read-only User –** May view data on the web interface, but may not make any changes to STCD settings or configuration.

Every user is assigned a username, password and access level. When a user navigates to a STCD IP address, the user will be directed to the ST login page and will be required to enter valid user credentials (username and password). Once logged into the web interface, the user is redirected to the index page shown in Figure 3-20.

The access level of the user is displayed in the top right of the page. The username can be seen in bold text, with the access level in brackets next to the username (3: Administrator; 2: Superuser; 1: Read-only user). If a user attempts to access a page on the web interface without correct authentication, the user is redirected to the login page. This is achieved using a PHP *session*. A PHP session stores information about the user currently accessing the web interface. This information is used to track used behaviour. Authentication is assigned to a user when the user logs in, and remains constant for the remainder of the user's web interface access session [45].

Figure 3-20 - Website Index Page

3.3.7.5 Web Interface Functionality

The website possesses adequate functionality to effectively manage the STCD. The functionality will be described in this section.

- **Level Crossing Sampling:**
  Current level crossing sampling values may be viewed and configured in the *Level Crossing Sampling* tab shown in Figure 3-21.

Stellenbosch University https://scholar.sun.ac.za

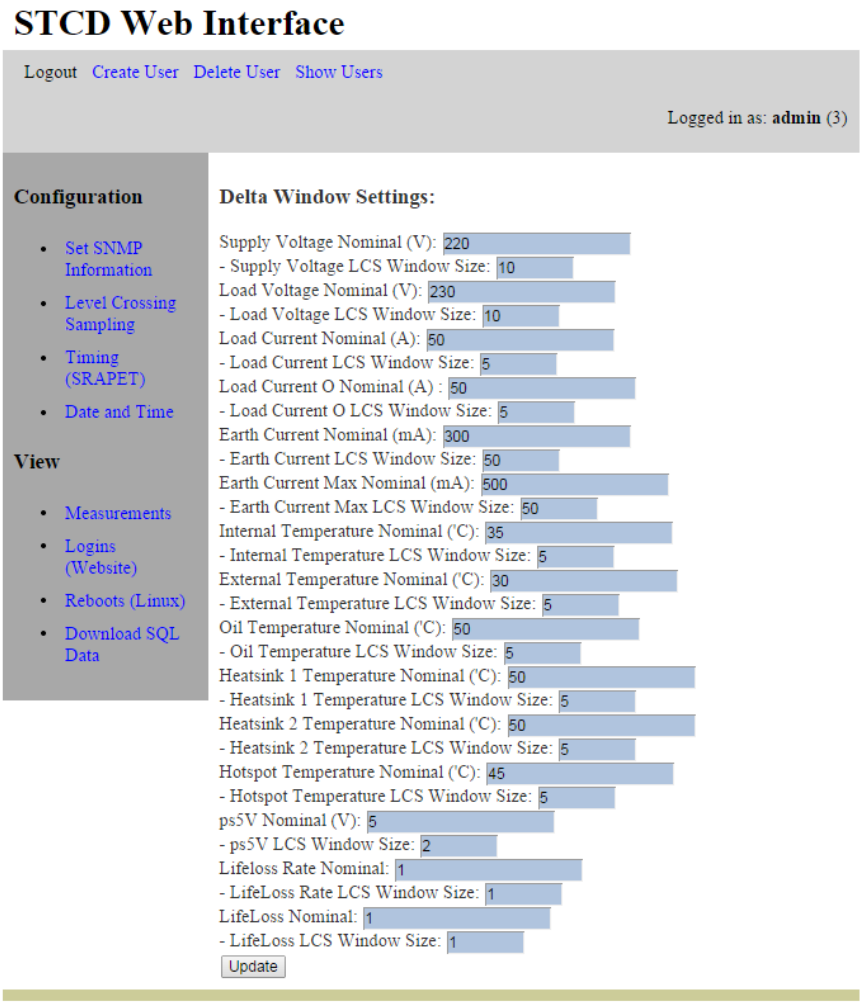


Figure 3-21 - Website Level Crossing Sampling Tab

Level crossing sampling is configured by setting a nominal value and a window size value for each data point. For more on level crossing sampling, see section 3.3.9.2. This functionality is achieved by writing values entered by a user to a MySQL table that is monitored by the MP. Values entered by a user are recorded and written into the MySQL *deltas* table using PHP.

- **Timing:**

  The time interval between consecutive measurements sent from the SRAPET controller board to the STCD may be set in the *Timing* tab. This functionality is achieved by reading from and writing to the MySQL *timing* table.

- **Download XML Reading Files:**

  The web interface possesses functionality to create and download SRAPET readings stored by the STCD in XML format. This functionality is utilized using the *Download SQL Data* tab. The web interface takes readings from the MySQL database and stores them in

Stellenbosch University https://scholar.sun.ac.za

XML files. The XML files can then be downloaded by a user. The user can choose to either generate XML files containing reading data for the past day, week or month.

XML files are used as a generic type for data transfer. XML is a language used to describe and exchange database-type data between various information sources. XML allows users to define data elements with sub-elements and attributes. The XML format is an effective way of communicating small or large amounts of data [46].

- **Date and Time:**

  A user may see the local time on the STCD in the *Date and Time* tab and update the time using one of two methods. The first is to update the time by manually entering it via the web interface. The second method utilizes the clock on the SRAPET controller by synchronizing the local time on the STCD with the local time on SRAPET controller itself by clicking the *Sync Time with SRAPET* button. This will set the time on the STCD to the local time on the SRAPET controller board.

  This functionality is achieved by executing Linux BASH scripts with PHP and using the MySQL *sync* table. Two scripts exist on the STCD that are used for reading and returning the current time and updating the current time respectively. These are executed when a user indicates that a time-read or update is desired. When the user indicates that time synchronization with the SRAPET controller is desired, the MySQL *sync* table is written to. This table is monitored by the MP, and the MP detects changes to the table. The table acts as a switch. When the value of an entry in the table is changed, the time is synchronized with the SRAPET controller and the value in the table is reset.

- **View Readings:**

  The most recent measurement sent from the SRAPET controller board to the STCD is recorded using the MySQL *readings* table. The most recent measurement can be viewed in the *Measurements* tab on the web interface. Readings older than the most recent reading are logged in the MySQL *readings* table, but are not displayed on the web interface. The *Measurements* tab is displayed in Figure 3-22.

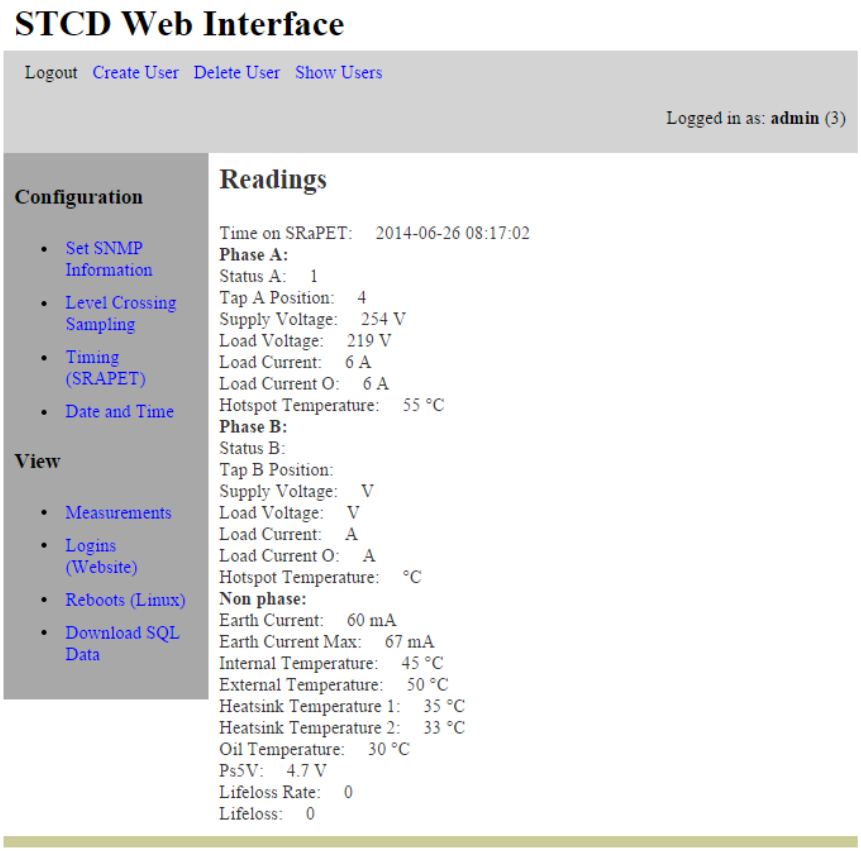


Figure 3-22 - Website Measurements Tab

- **User Management:**

  The web interface allows an administrator to create new user accounts, delete existing user accounts and view existing user accounts. Each ST unit has a unique set of user accounts. The *Create User, Delete User* and *Show Users* links are used to achieve this. The user creation page is shown in Figure 3-23. An administrator can specify a username, password and access level when creating a user account.

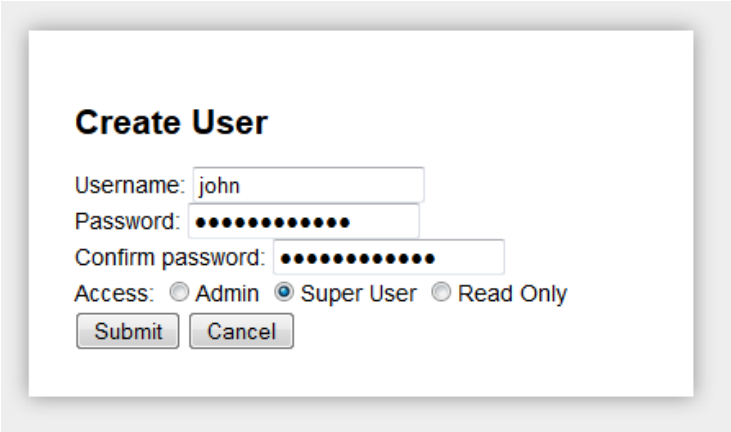


Figure 3-23 - Website User Creation

The *Show Users* and *Delete User* pages are shown in Figure 3-24 and Figure 3-25 respectively.

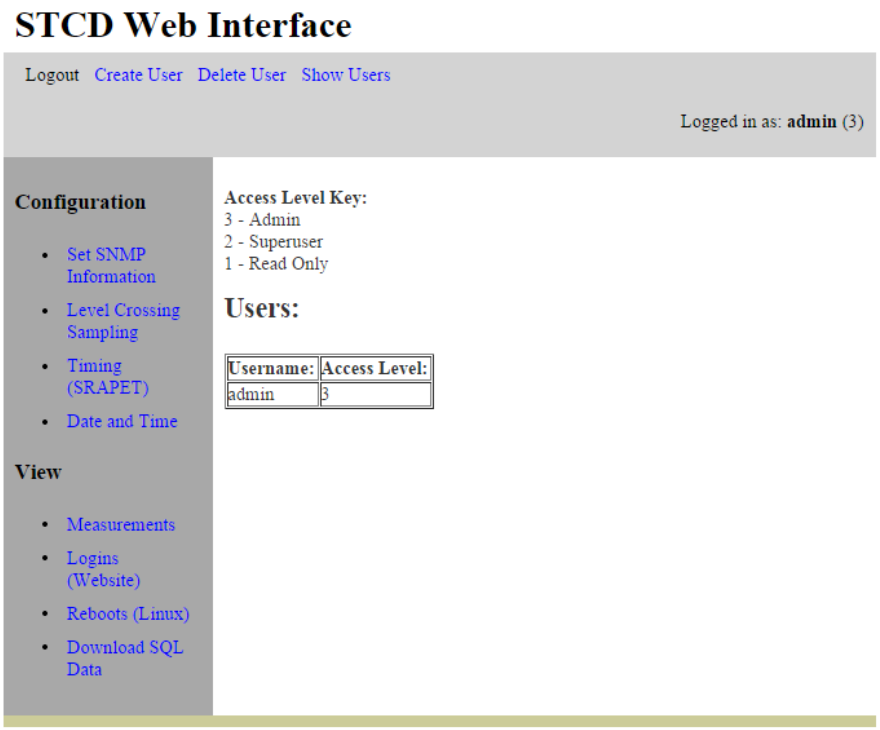


Figure 3-24 - Website Show Users Page

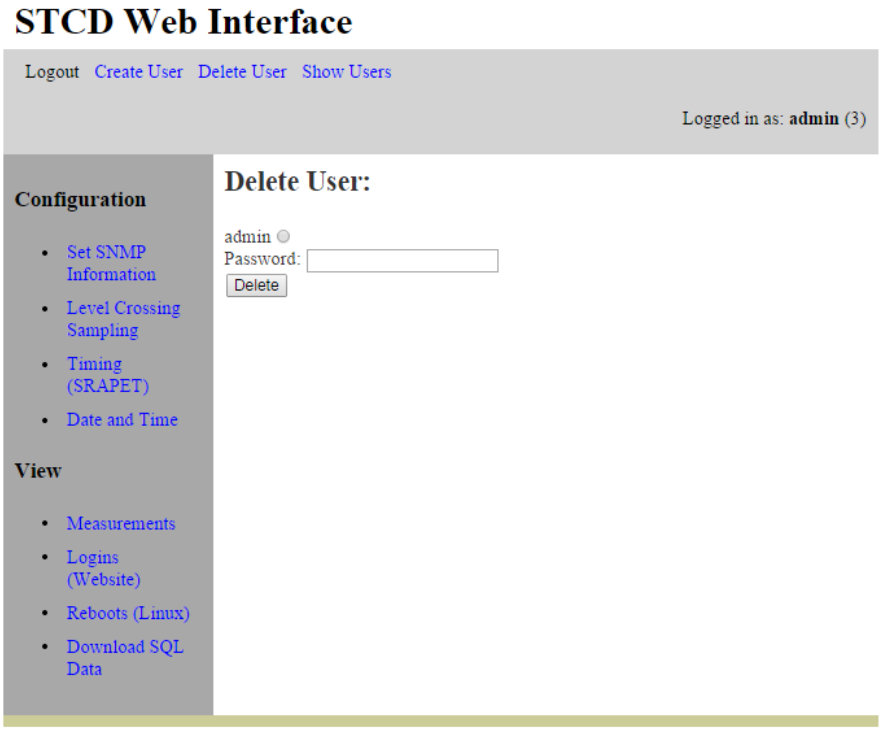


Figure 3-25 - Website Delete User Page

As can be seen in Figure 3-25, an administrator must supply the correct password for a user account to be able to delete it.

Stellenbosch University https://scholar.sun.ac.za

3.3.7.6 The Mongoose Systemd Service

A systemd service is created to start and maintain the mongoose webserver. When, for whatever reason, the Mongoose webserver exits, it is restarted. This ensures that the web interface remains functioning permanently.

3.3.8 The C++ Programming Language

The C++ programming language is a language designed to be an improved version of the C programming language. It supports object-oriented programming and data abstraction and inherits much of its functionality from the C programming language [47].

A C++ program consists of multiple source and header files. This form of organization is followed to reduce compilation time and to keep code organized. Header files contain declarations of variables and functions while source files contain their implementation. Header files are used to ensure that all declarations pertaining to a specific object remain consistent. Source files contain function code that enables various functionalities in a program. All C++ programs have a *main* function. The main function is the very first function that the C++ program executes. The program may branch into various other functions from the main function [47].

The C++ programming language was used to create the MP and ModemTalk programs on the STCD.

3.3.9 The Main Program

The main program (MP) refers to the software program on the STCD that enables and facilitates DNP3 communication, interfaces with the MySQL database and facilitates serial communication with the SRAPET controller board.

The MP is written completely in the C++ and C programming languages. The main objectives of the program are to achieve the following:

- **Two-way serial communication with the SRAPET controller board** – The MP enables the STCD to gather measurement data from the SRAPET controller and send instructions to the SRAPET controller.
- **DNP3 communication with the data server** – The MP facilitates DNP3 communication over a network to the data server. It determines what measurement data to send to the data server and sends the identified data.
- **Interfacing with the MySQL database –** The MySQL database is used as a communication medium between the web interface, the XML interface and the MP. The database is also used to store configuration data and operational logs.
- **Local logging –** The MP offers a log interface (logs to files and standard outputs) that enables more effective debugging and event tracking for the program.

Stellenbosch University https://scholar.sun.ac.za

The following sections give an overview of the MP and explore its main components.

3.3.9.1 MP Overview

The MP functions as a three-state state-machine. The first state is the *SETUP* state, the second is the *RESTART* state and the third is the *ONLINE* state. The MP enters the SETUP state upon startup. Configuration and system setup takes place in the SETUP state. The MP enters the RESTART state once all setup configurations have taken place. The RESTART state establishes communication with the SRAPET. Once the communication channel is open and functional, the MP enters the ONLINE state. In the ONLINE state, the SRAPET controller streams measurement data to the STCD. The ONLINE state is a continuous state that the MP remains in until a system restart or fault occurs.

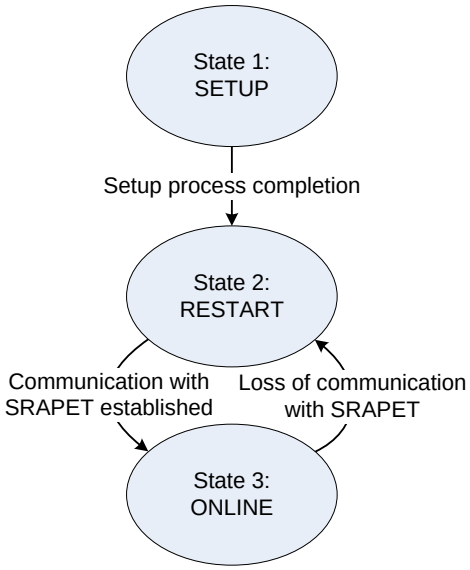


Figure 3-26 - Main Program State Machine

Configuring the STCD as a DNP3 outstation, fetching of configuration settings from the MySQL database and setup of timers takes place in the SETUP state. The MP sets up the STCD to function as a DNP3 outstation that can accept connections from, and establish connections to, a DNP3 master station. The MP is setup to operate using sixty-six DNP3 analog output points and three DNP3 binary output points. These points represent all data points measured by the STCD and are listed in APPENDIX A.11.

Stellenbosch University https://scholar.sun.ac.za

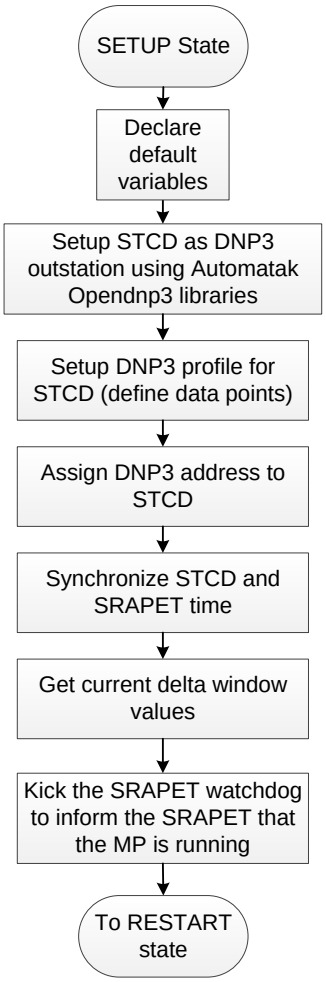


Figure 3-27 - SETUP State

Once the MP has finished all actions in the SETUP state, it enters the RESTART state. The purpose of the RESTART state is to establish functional communication with the SRAPET controller board. The MP identifies key SRAPET controller characteristics and determines how communication between the STCD and the SRAPET controller should take place. This identification takes place by serially sending a command to the SRAPET controller requesting configuration settings from it. The configuration settings are stored in a text file on the STCD and the MP identifies whether the attached SRAPET controller is single or dual phase. The phase of the SRAPET controller determines how many measurable data points it will possess. The MP adapts to accommodate the SRAPET phase by creating variables for all available data points. The MP then sends a command that informs the SRAPET controller to begin streaming measurement data to the STCD over the serial link. The RESTART state is shown in Figure 3-28.

Stellenbosch University https://scholar.sun.ac.za

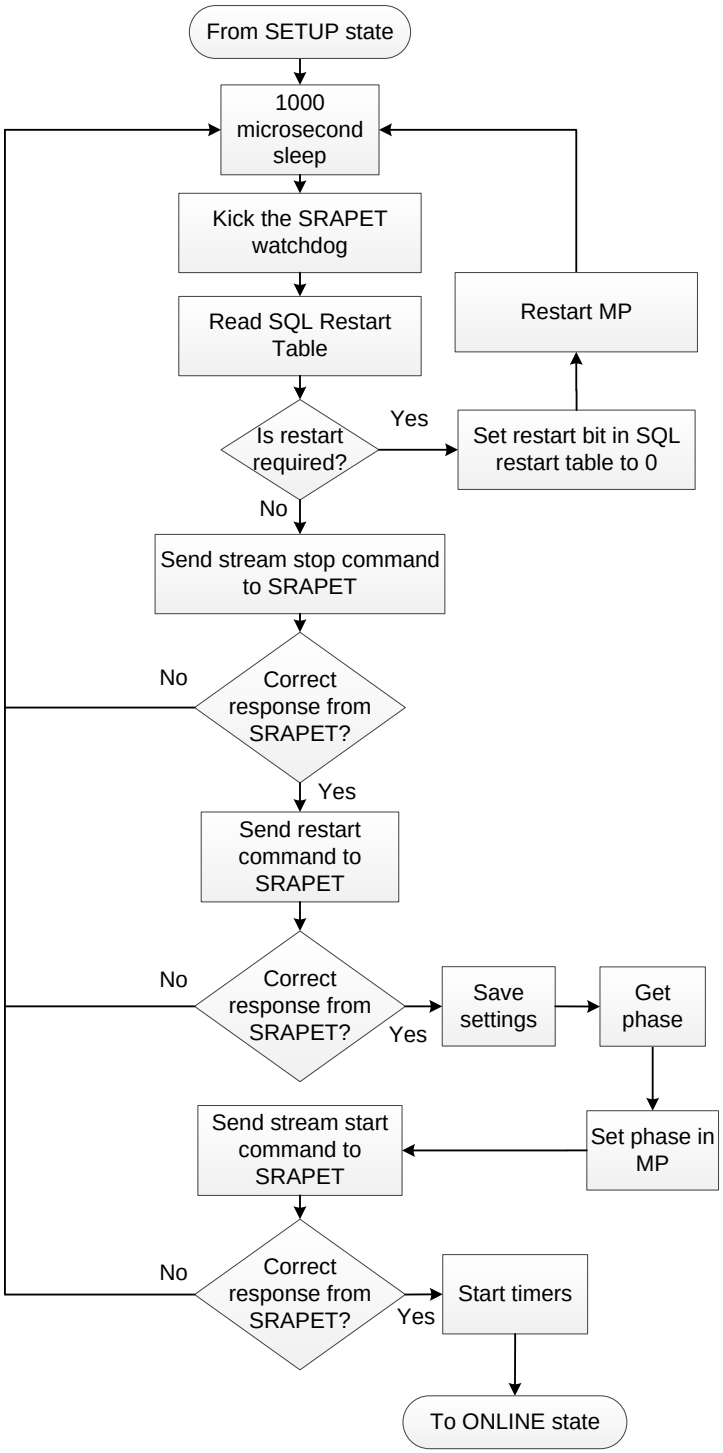


Figure 3-28 - RESTART State

An acknowledge message is received if the command to begin the serial streaming of measurement data has been successfully received by the SRAPET controller. Once the acknowledge message has been received, the MP enters the ONLINE state. The MP remains in the ONLINE state until a system restart or fault occurs. In the ONLINE state, the MP:

- Continuously monitors the MySQL database for configuration setting changes made by a user via either the web or XML interfaces.
- Continuously receives measurement data from the SRAPET controller serially.

Stellenbosch University https://scholar.sun.ac.za

- Reports relevant measurement data to the data server via DNP3.
- Kicks the STCD hardware watchdog implemented on the SRAPET controller board.

The ONLINE state is shown in Figure 3-29.

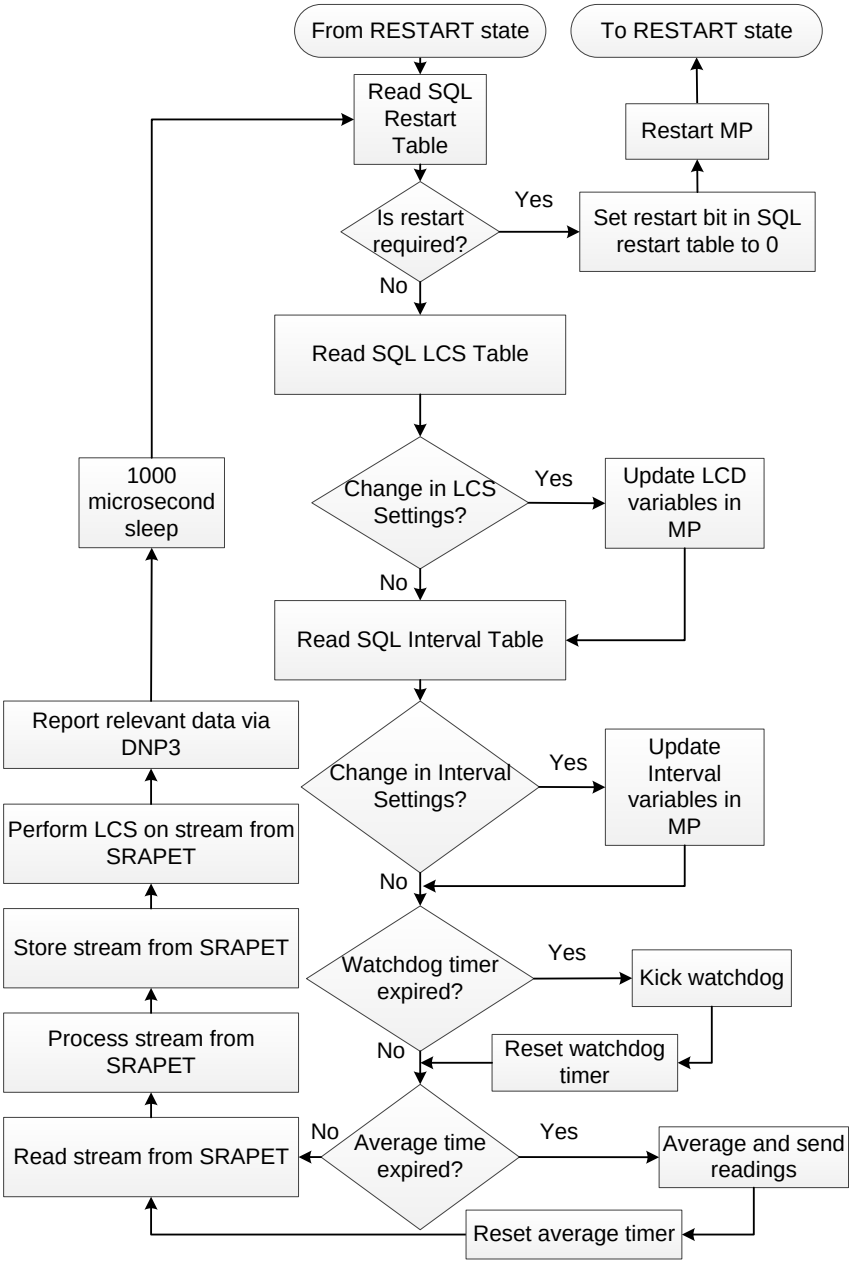


Figure 3-29 - ONLINE State

3.3.9.2 DNP3 Reporting Philosophy

The philosophy behind the management of data in a microgrid is that of distributed computing. Many intelligent nodes work together to perform a global task. At each node, device intelligence is used to process data and to convert it into information and knowledge.

Once data is converted to knowledge it is communicated to other nodes. Because knowledge is data that has been processed, it contains more information than raw data. Thus, communicating

Stellenbosch University https://scholar.sun.ac.za

knowledge over transmission media will use less bandwidth than communicating raw data, and the same amount of information will be captured [48].

This philosophy utilizes the intelligence possessed by various devices to distribute data processing and minimize the bandwidth used to gain visibility of the power grid. The ST acts as a data aggregator for the microgrid. Thus, it fulfils two functions:

- It gathers data from its own operations and processes the data. It then communicates the processed data and includes it in calculations.
- It receives information from other microgrid agents and converts the information into knowledge. This knowledge is then used to produce wisdom, make decisions and communicate the state of the microgrid to operators.

The MP implements a reporting philosophy for its own operational data that relies on both periodic and event-based data reporting.

Readings are taken from the SRAPET controller in quick succession, up to a reading every second. After a user-defined period, averages of these values are taken and the averages are sent to the data server (e.g. every 5 minutes). All individual readings are stored locally in the MySQL database. These readings can be downloaded with time-stamps from the STCD using the HTTP protocol.

Events are changes in data that are considered significant. An example of an event is a reading for load voltage that is above a user-defined threshold. This would indicate that the transformer is in a condition that is of interest to those monitoring its operation. In this case, it would be beneficial for operators to know the state of the transformer and therefor data is reported. A user defines what events the MP reports, using a technique known as Level-Crossing Sampling (LCS) [13].

LCS defines a number of thresholds that divide the data variable range into a discrete number of windows. As long as the current value of the data variable remains within a window (does not cross a threshold), no event reporting takes place. However, when the current value crosses a threshold, from one window to another, a data event is reported. Using this technique, operators can specify what the STCD should see as significant events. Operators specify a nominal value considered the *central* or *average* value for the data point. They then specify a window size and windows of the given size are constructed in both positive and negative directions of the variable range with the nominal value as the central value. The LCS reporting technique is illustrated in Figure 3-30. The equation used to determine the current window for LCS is given below.

$$New\ Window = \ \lfloor |CV - NV| \ / \ WS \rfloor$$

Where:

CV = Current value of variable

Stellenbosch University https://scholar.sun.ac.za

NV = LCS Nominal Value

WS = LCS Window Size

It is important to note that all data points are reported when an event occurs. This gives operators a more complete picture of the status of the transformer. Also, because of protocol overheads, it makes more sense to report all data points in a single message than to send a separate message for every data point.

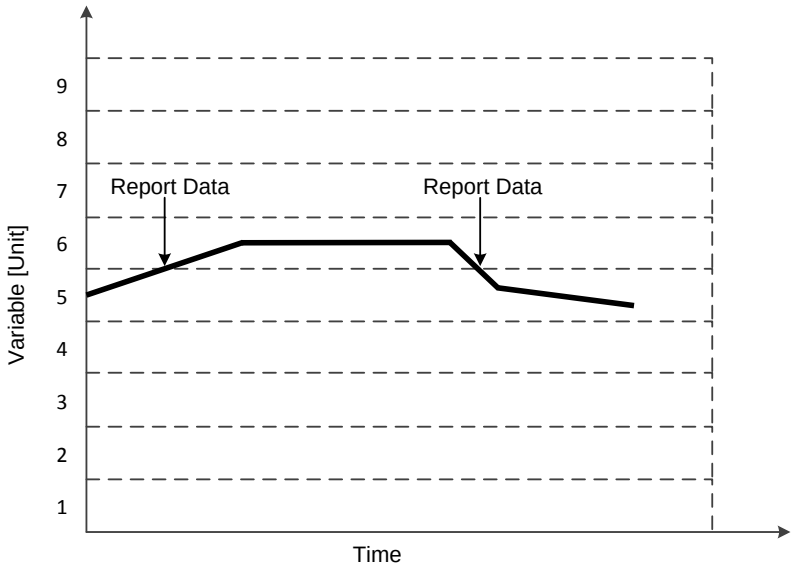


Figure 3-30 – Level Crossing Sampling Example

The user can also specify the time interval at which to periodically report averages. Using this reporting philosophy reduces bandwidth while maintaining visibility as only important events and periodic averages are reported. The reporting philosophy is illustrated in Figure 3-31.

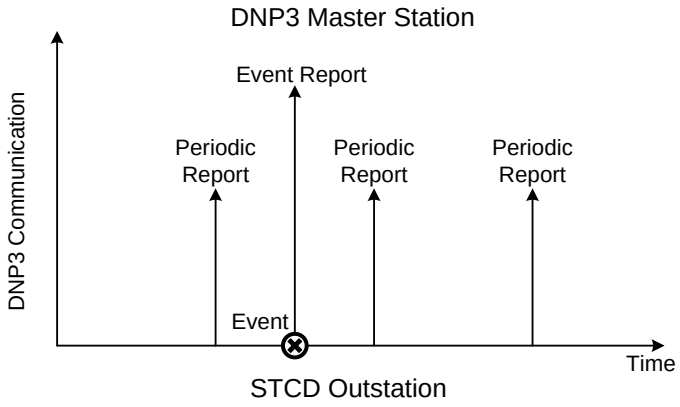


Figure 3-31 - Reporting Philosophy

Originally, functionality was included in the STCD software to allow a user to specify which data points are to be reported and which data points are not to be reported. It was decided to remove this functionality and to report all data points when an event occurs or averages are sent. This decision was made on basis of the bandwidth usage to send one data variable versus all data

Stellenbosch University https://scholar.sun.ac.za

variables in a message, and because a report of all data variables gives more insight into transformer operations.

3.3.9.3 MP Serial Communication

Serial communication with the SRAPET controller is enabled using code taken from the website found at [49]. The technique used in the code makes use of the *Boost Asio* library. The Asio library is a multi-platform I/O library that offers an efficient, scalable, portable and easy to use solution for connecting to a variety of devices via sockets or ports [49], [50]. The two files in the MP that perform the fundamental serial communication tasks are *TimeoutSerial.h* and *Timeoutserial.cpp.* The TimeoutSerial.cpp file contains implementation functions for serial communication while the TimeoutSerial.h file describes the interface and defines default values.

The *TimeoutSerial* class is used to both read serially from the SRAPET and write serially to the SRAPET from the STCD. When the STCD reads from the SRAPET, the program halts for a period in anticipation of an incoming serial message. If no serial data is read on the serial port before a pre-set timeout expires, an error is returned and the program continues.

The *SerialPort* class uses the functions contained in the *TimeoutSerial* class to write specific commands to the SRAPET and read from it. The *SerialPort* class also handles logging of events pertaining to serial communication (such as a timeout occurring on a serial read) and interfaces with the MySQL database to update counters when these timeouts occur. If more than three timeouts occur, the *SerialPort* class will write to the *prestart* table in the MySQL database which in turn will cause the MP to be put into the RESTART state.

Stellenbosch University https://scholar.sun.ac.za

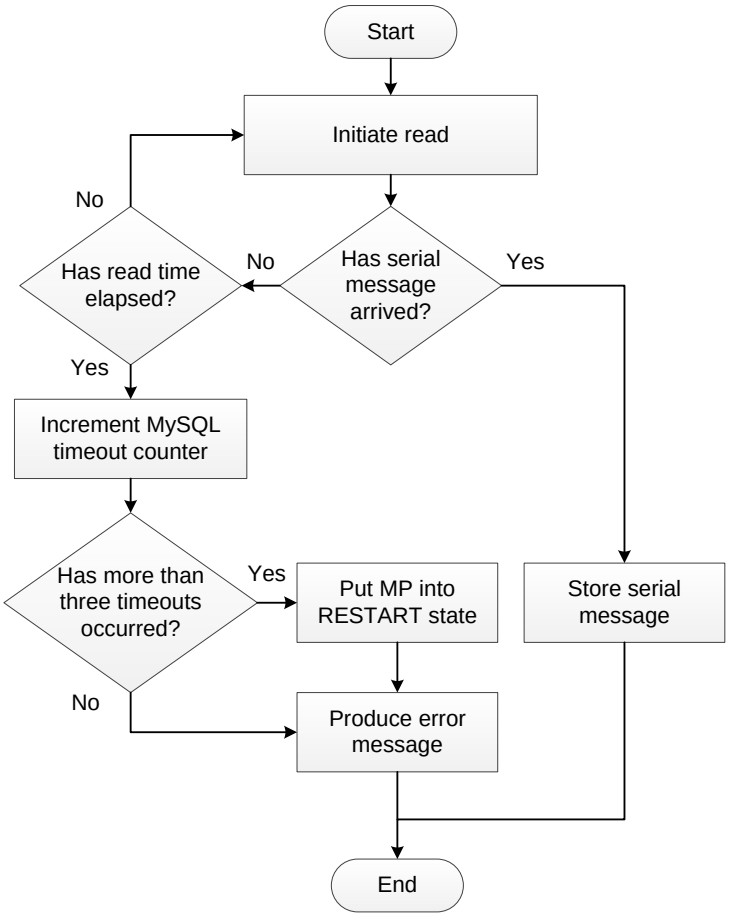


Figure 3-32 - MP Serial Read

Measurement data is received in a string format from the SRAPET controller. Upon reception of a reading from the SRAPET, the data is stored locally in the MP using a data structure. The data structure contains various variables representing the data points monitored by the SRAPET controller. Once stored in a variable, the readings are processed to see if an event has occurred. If an event has occurred, the measurement is reported via DNP3 to the data server. The data structures used to store single and dual phase transformer measurements are shown in Figure 3-33 and Figure 3-34 respectfully.

```
struct SStream
{
uint32_t time;
uint16_t statusA;
uint16_t tapA;

float vsA;
float vlA;
float ilA;
float iloA;

float iEarth_mA;
float iEarthMax_mA;
float tInt;
float tExt;
float tHeatsink1;
float tHeatsink2;
float tOil;
```

Stellenbosch University https://scholar.sun.ac.za


```
float tHotspotA;

float ps5V;
float lifeLossRate;
float lifeLoss;

uint16_t crc;
};
```

Figure 3-33 - Single Phase Measurement Data Structure

```
struct DStream
{
    uint32_t time;
    uint16_t statusA;
    uint16_t tapA;
    uint16_t statusB;
    uint16_t tapB;

    float vsA;
    float vlA;
    float ilA;
    float iloA;

    float vsB;
    float vlB;
    float ilB;
    float iloB;

    float iEarth_mA;
    float iEarthMax_mA;
    float tInt;
    float tExt;
    float tHeatsink1;
    float tHeatsink2;
    float tOil;
    float tHotspotA;
    float tHotspotB;
    float ps5V;
    float lifeLossRate;
    float lifeLoss;

    uint16_t crc;
};
```

Figure 3-34 - Dual Phase Measurement Data Structure

Incoming readings are stored in the *averageBuffer* table of the MySQL database. This table is used to calculate reading averages over a user-defined period. When the defined period expires, readings in the averageBuffer table are averaged, the averages are reported via DNP3, the readings are copied into the *readings* MySQL table and the averageBuffer table is cleared. This process is illustrated in Figure 3-35.

Stellenbosch University https://scholar.sun.ac.za

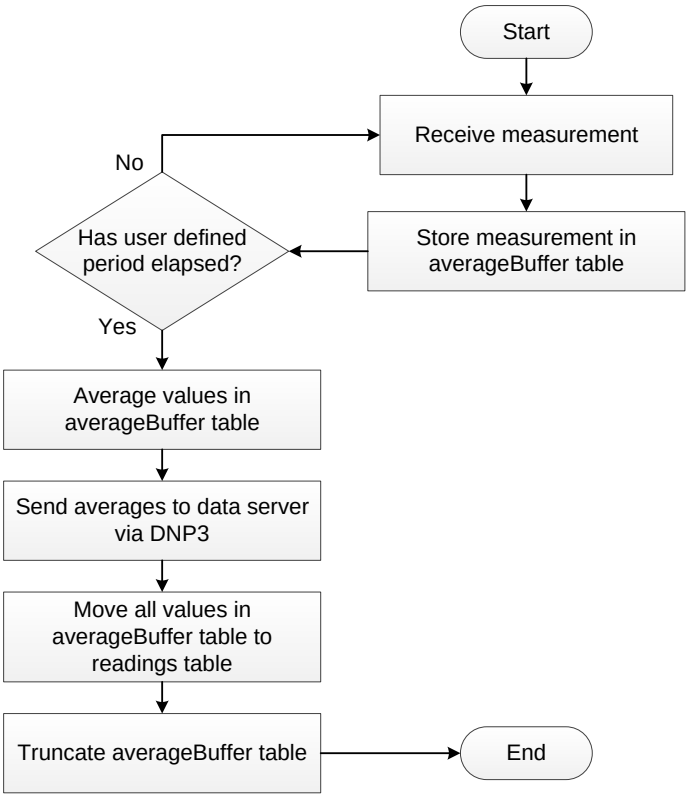


Figure 3-35 - MP Serial Measurement Read

A custom communicated protocol was used in the serial communication between the STCD and the SRAPET. Cyclic Redundancy Check (CRC) checks are performed to ensure the data integrity of messages across the serial interface. Information on the protocol is found in APPENDIX D and [51].

3.3.9.4 SQL Connector

The SQL Connector is the interface between the MP and the MySQL database. A public C++ SQL Connector library available from [52] was implemented to enable communication with the MySQL database.

The *MySQL C++ Connector* is a database connector that provides C++ libraries for connecting to a MySQL database [53]. The library offers a mechanism for connecting to a MySQL server using authentication and for performing various database and table related tasks on the server. Tables can both be read from and written to.

The *sqlConnector* class in the MP contains functions to read from, write to and update MySQL tables. When run, the functions also indicate whether they were successfully executed or not. Figure 3-36 and Figure 3-37 illustrate the processes of the MP reading from and writing to the MySQL database respectively.

Stellenbosch University https://scholar.sun.ac.za

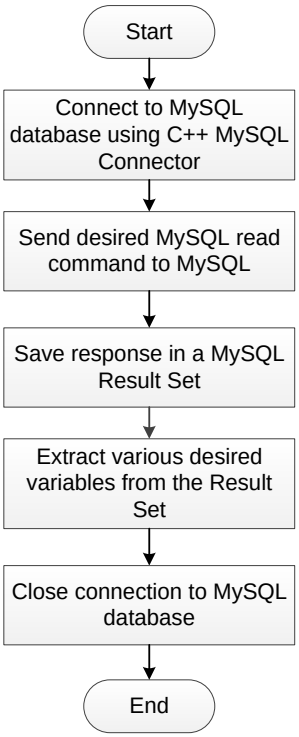


Figure 3-36 - MP MySQL Read

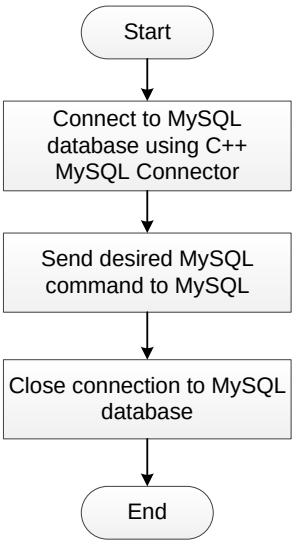


Figure 3-37 - MP MySQL Write

The functions used to implement MySQL reading and writing are contained in the sqlConnector.cpp and sqlConnector.h classes. These functions allow communication with the MySQL database for reading, writing and logging purposes.

3.3.9.5 Automatak Opendnp3 Libraries

The Automatak Opendnp3 libraries from [54] is a fork (a project that was built on another project) from the original Opendnp3 project. It was created and is maintained by Adam J. Crain. The Automatak Opendnp3 libraries are C++ libraries that can be installed on Windows or Linux. The libraries provide functionality that implements DNP3 up to level 2+.

Stellenbosch University https://scholar.sun.ac.za

The Opendnp3 libraries are used to setup the STCD to function as a DNP3 outstation. The STCD is initialized to have 4 Binary and 64 Analog data points. These data points include all data points received from the SRAPET controller and derived data points such as maximum and minimum values. To view the full list of data points, please see APPENDIX A.11.

The Opendnp3 libraries are used to only allow specific master stations access to the STCD outstations. This is done by specifying what IP addresses to accept DNP3 connections from. As both the STCD and the data server are on a single VPN, both of their IP addresses are known and secure connections can be created.

3.3.9.6 Data Handler

The dataHandler class handles the flow (gathering and sending) of data in the MP. It contains functions that are called from the program's main loop to perform various actions on or with data. The following function types are contained within the dataHandler class:

- **SRAPET Communication Functions** – These are functions that send commands to the SRAPET controller and analyse the responses received using the SerialPort class. Operations performed by these functions include:
    - Getting the transformer device information. This determines whether the transformer is a single- or dual-phase transformer, what the ID number of the transformer is, how long the device has been on or off for and the reason why the device last restarted. These settings are logged in a text file and are written to the MySQL database using the sqlConnector class. The obtaining of transformer device information is illustrated in Figure 3-38.

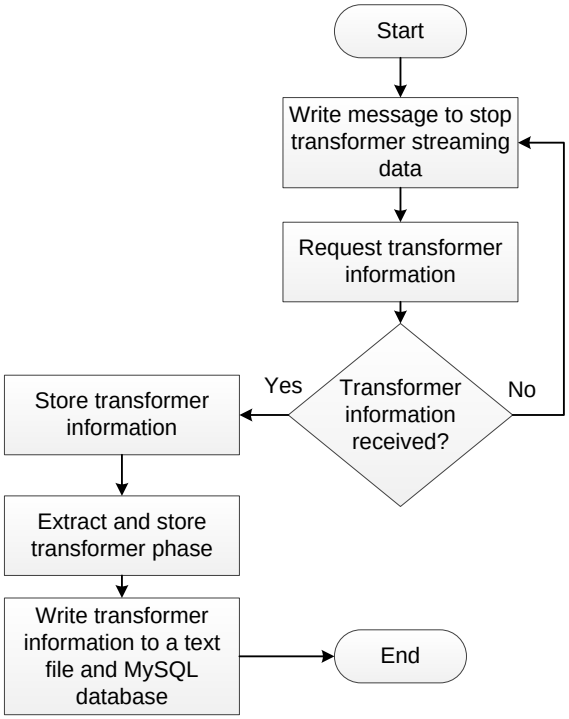


Figure 3-38 - Obtaining Transformer Device Information

Stellenbosch University https://scholar.sun.ac.za

  - Sending of commands to the SRAPET controller to either start or stop streaming measurement data to the STCD. When the streaming of measurement data is in progress the STCD reads the data, stores the data in variables, saves the data in the MySQL database and sends the data to the data server via DNP3 periodically or if an event occurs (see Figure 3-35).
  - Setting the time interval at which consecutive SRAPET readings must be streamed from the controller to the STCD.
  - Performing CRC checks to ensure that the integrity of the data received from the SRAPET controller is not compromised.
  - Asking the SRAPET controller for the current time of its real-time clock and updating the STCD clock in order to synchronize the two clocks.
- **LCS Functions** – These functions perform Level-Crossing Sampling (LCS) tests on the incoming data measurements, streamed from the SRAPET controller. When an LCS test is passed, the reading is reported to the data server. If an LCS test is not passed, the reading is stored in the MySQL database to be used in calculating the averages of data points over a user-defined period.
- **Report Functions** – The dataHandler class uses the Automatak Opendnp3 libraries to send data to the data server via DNP3.
- **Min and Max Functions** – These functions record the minimum and maximum values of data points, stores them and reports them when they change.
- **Generic Opendnp3 Functions** – The dataHandler class contains functions that the Opendnp3 libraries require to function.

3.3.9.7 Logging

A basic logging library is used to enable the MP to write logs to a text file. The log library allows a developer to log events with various priority levels: ERROR, WARNING, INFO and DEBUG. In the MP the following is logged:

- Errors that occur during serial communication.
- Writes to the *prestart* MySQL table, indicating the MP should be restarted.
- The MP being put into RESTART mode.

The current time is included as a timestamp when a log is written. Logging is a valuable tool for debugging and gaining a better perspective of what events took place on the STCD.

3.3.10 The XML Interface

The XML Interface allows users to configure the STCD using an XML configuration file.

The XML interface consists of two sub-programs: *Notify* and *Parser*. Notify is a program that continuously monitors a specific directory on the STCD. The Notify program uses the iNotify C libraries to detect changes to a specified directory and execute actions upon occurrence of those

Stellenbosch University https://scholar.sun.ac.za

changes. When a file is uploaded into the monitored directory, Notify moves the uploaded file to another specified directory and initializes the Parser program.

The Parser program takes the file that Notify moved and examines it. The file must be of a pre-set format to be acceptable to the Parser program. The file is an XML configuration file and contains data that configures LCS for the STCD. If the file is of correct format, the Parser program interprets the data contained in the file and updates the STCD MySQL database with the new configuration information.

Once the configuration information contained in the file has been saved, the file is deleted. The structure of the XML is given below:

```
<?xml version="1.0" encoding="utf-8"?>
        <configuration version="1.0" type="config">
        <deltas svn="220" svd="10" lvn="230" lvd="10" lcn="50" lcd="5" lcon="50" lcod="5"
            ecn="300" ecd="50" ecmn="500" ecmd="50" itn="35" itd="5" etn="30" etd="5" otn="50"
            otd="5" h1tn="50" h1td="5" h2tn="50" h2td="5" hstn="45" hstd="5" ps5vn="5" ps5vd="2"
            llrn="1" llrd="1" lln="1" lld="1">
        </deltas>
        </configuration>
```

In the structure above, LCS nominal values and window sizes are specified in the *deltas* object of the XML file.

The *RapidXml* libraries are utilized to parse incoming XML files. RapidXml is an XML parser programmed in the C++ programming language that focuses on speed, useability and portability [55]. The RapidXml libraries are used to parse the XML configuration file and identify various XML nodes containing configuration data. The data is then stored in the STCD MySQL database using the C++ MySQL connector libraries (see section 3.3.9.4).

Flow diagrams for the Notify and Parser programs are shown in Figure 3-39 and Figure 3-40 respectively.

Stellenbosch University https://scholar.sun.ac.za

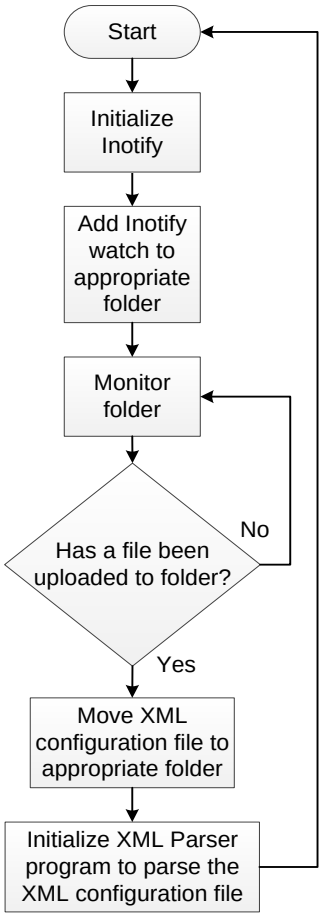


Figure 3-39 – XML Interface Folder Watcher (Notify)

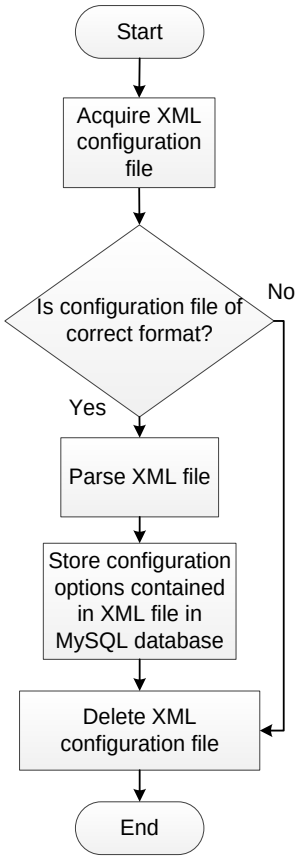


Figure 3-40 – XML Interface Functionality (Parser)

3.3.11 The ModemTalk Programs

ModemTalk consists of two programs (ModemTalk and ModemTalk_D) written to extract valuable data from the 3G modem and make it available for the Net-Snmp agent.

A program called *modeswitch* is used to switch a connected modem to a mode where it is usable by the operating system. When a USB modem is connected, it is often detected as USB storage devices to enable access to software stored on the modem. Modeswitch switches the modem to a mode where virtual ports for communication to and from the modem are created. These ports can be found in the */dev* directory of the Linux file system and use the naming convention *ttyUSBX*, where X represents a number. These ports are used to communicate serially with the modem from Linux.

The ModemTalk program utilizes these virtual ports using the C++ programming language. A USB modem responds with data to AT commands. When a modem is not connected to the internet, the *ttyUSB0* port, created when it is switched into the correct mode, may be used to extract data from the modem. The ModemTalk program uses this interface to extract the following data from the USB modem:

Stellenbosch University https://scholar.sun.ac.za

- GSM network operator name.
- Location area code (LAC).
- Network attachment. For example, GSM, GPRS, EDGE, HSDPA or HSUPA.
- International Mobile Station Equipment Identity (IMEI) code.
- Modem manufacturer.
- Modem model.
- Modem firmware version.

When a USB modem is connected to the internet, the ttyUSB0 virtual port is made unavailable for communication. In this case, the ttyUSB2 virtual port is used by the ModemTalk_D program to extract the current signal strength experienced by the modem.

All data extracted by the ModemTalk program is static and does not change with time. The ModemTalk program is executed once before the modem is connected. All data extracted by ModemTalk is saved in a file, and does not need to be modified again. The ModemTalk_D program runs continuously after the USB modem has been connected as data gathered by ModemTalk_D is dynamic and changes with time.

3.3.12 Persistent Modem Connection Service

The persistentModemConnection systemd service establishes and maintains the 3G USB modem connection. The service calls a BASH script called modemDetect.sh. The script utilized the BASH programming language to:

- Ensure that the modem is detected by the STCD and can be used.
- Establish the modem connection by executing the *wvdial* program.
- Establish the VPN connection by calling the *OpenVPN* program.
- Maintain both the 3G and the VPN connections.

The modemDetect.sh script also takes relevant action to correct any faults that it detects in modem operations. These include:

- Ensuring that only one instance of itself is running by using a lockfile.
- Resetting the BB USB port if modem is not detected.
- Checking if internet and VPN connections exist and establishing them if they do not.
- Checking that internet and VPN connection are active using ping. The connections are re-established if they are inactive.

The workings of this script is illustrated in Figure 3-41.

Stellenbosch University https://scholar.sun.ac.za

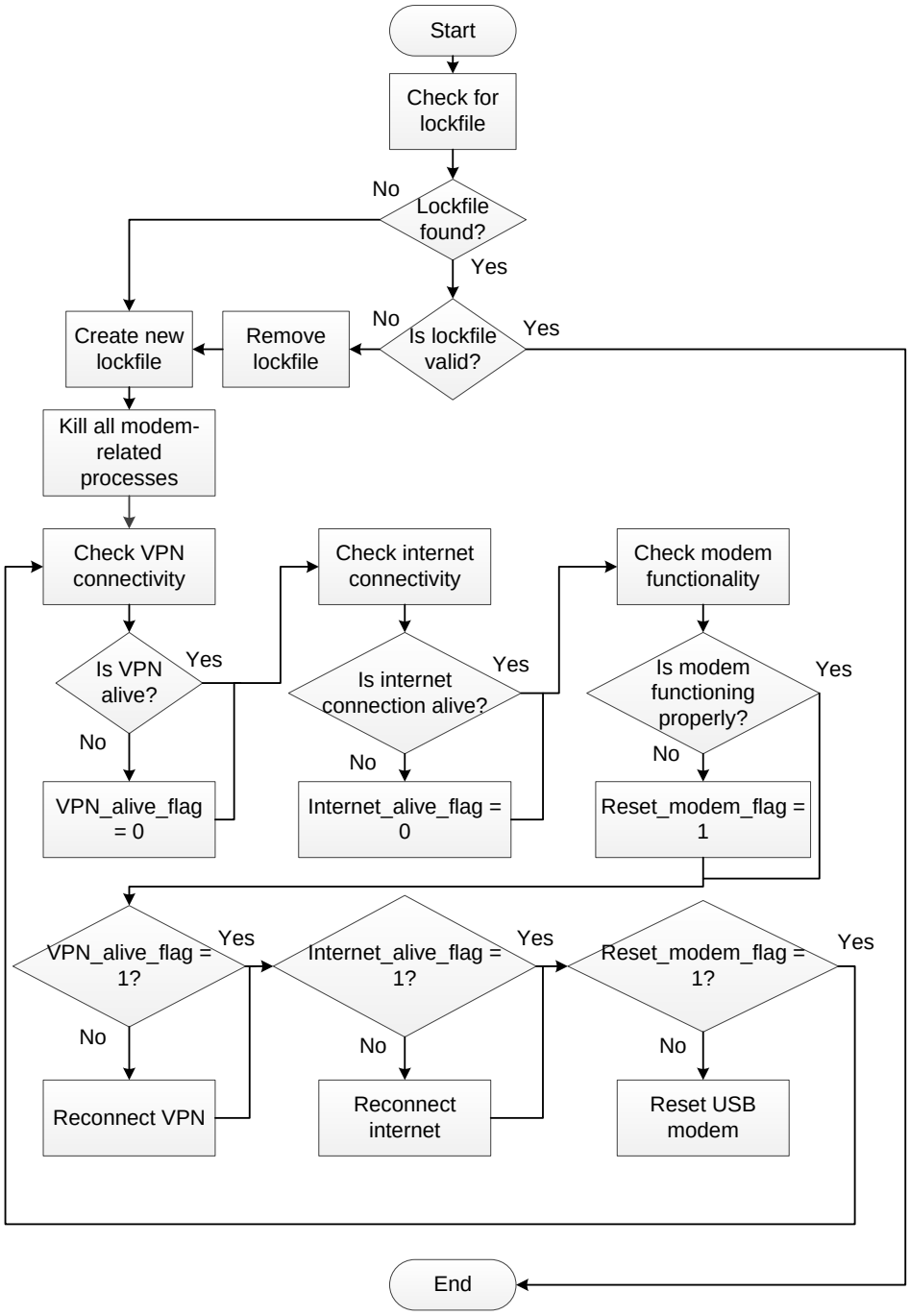


Figure 3-41 - ModemDetect Script

Preventing Duplication of Script Runs

modemDetect.sh uses a lockfile to ensure that only one instance of a script is running at a time. When Linux runs a process, a Process Identifier (PID) is assigned to that instance of the process. The PID remains constant for as long as the instance of the process is running. When a new instance of the process is started, a new PID is assigned to the new instance.

A lockfile is a file created by a program that contains the PID of the program. This file is used to ensure that two instances of a process do not run at the same time. This is done by a program checking whether the PID contained in the lockfile is still active. If the PID is active, an instance of the process is already running.

Stellenbosch University https://scholar.sun.ac.za

USB Modem Detection

As mentioned in section 3.3.11, the USB modem creates virtual ports when connected to the BB. These ports can be used to serially communicate with the modem. By experimentation it was found that when the modem experiences weak signal strength, the virtual ports created by the modem are removed, and the modem cannot be communicated with.

The modemDetect.sh script checks whether the USB modem has been connected to the BB using the *lsusb* command. This command lists all USB devices that are connected to the USB port of the BB [56]. If the modem is not found under connected USB devices, the BB USB host is reset. This is done using a program called *devmem2* to toggle the power of the USB hub on and off. A script (toggleUsbPower.sh) is called from modemDetect.sh to perform this operation.

Once the modem is detected using lsusb, a check is done to see if the virtual ports have been created that the modem is accessed by. If these ports have not been created, the modem is reset. If they are detected, the modemDetect.sh script calls a C++ program called *CheckModemOk*. CheckModemOk checks to see if the modem communicates properly by sending the “AT” AT command and recording the response. If the modem is functioning properly, it should respond with “OK”. If it is not functioning properly, it is reset.

Connecting to the Internet

The Wvdial program is used to connect the STCD to the internet via the 3G USB modem. Wvdial is a point-to-point dialer program that dials a modem and starts a point-to-point connection [57]. Wvdial is easy-to-use and automatically detects the settings required to connect the modem to a 3G network. The *wvdialconf* terminal command is used to automatically detect settings necessary to connect a modem to a network and create a configuration file for *wvdial* with these settings. A user need only insert into the configuration file the desired APN to connect to, the number the modem must dial to connect and authentication if required.

The modemDetect.sh script checks to see if there is an active internet connection by attempting to ping a website. If an active internet connection is not found, any instances of wvdial that are running are killed and a new instance of wvdial is started. This creates a new connection to the internet.

The same process is followed to maintain the VPN connection. Instead of pinging a website, an attempt is made to ping the VPN server. The next section describes the VPN in greater detail.

3.3.13 Virtual Private Network

A Virtual Private Network (VPN) acts as a private communication channel between devices. It is a replacement technique for historical leased lines that offer a private connection between devices or networks. A VPN uses authentication to ensure that only valid users may access the virtual network it creates and also encrypts all data sent across it, adding security. VPNs do not need

Stellenbosch University https://scholar.sun.ac.za

private lines but run on existing networks and offer secure point-to-point connections over a public network [58].

VPN networks have a server-client topology similar to a TCP/IP network. Clients connect to a server using VPN software and are assigned a unique IP address only applicable on the VPN network, creating a new virtual network for the server and its clients. A client can now be accessed through the new, custom IP address assigned by the server. The server's role is to manage security certificates that authenticate connections on the VPN and to route data travelling between clients.

When a VPN client connects to a VPN server, a virtual tunnel is created between the two devices and the server assigns a custom address to the client. The server itself also has a custom address on the virtual network. The VPN topology is given in Figure 3-42.

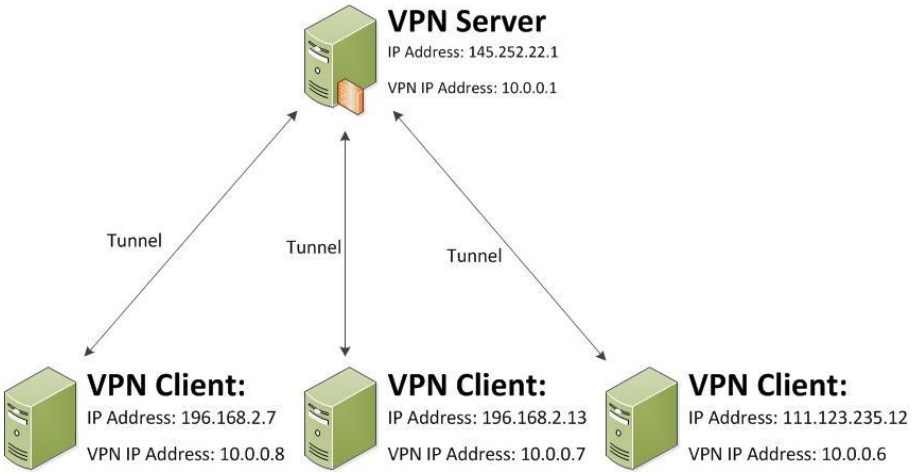


Figure 3-42 - VPN Topology

A DNP3 master station must know the IP addresses of outstation devices to communicate with them. When a device connects to a public internet network (APN) via 3G or another technology, it is assigned an IP address that cannot be predetermined by the DNP3 master station. Thus, a need for static IP addresses on DNP3 outstations arises. This need is met by connecting both the outstations and master station to a VPN server that assigns virtual static IP addresses to the clients. In this scenario, only the VPN server needs a real static IP address so that clients will know where to establish the VPN connection to.

Thus, the clients are all instructed to connect to the static IP of the server address and then become nodes on the VPN network. The VPN server takes care of routing and encrypting data to ensure that secure, reliable connections are established. Pre-created certificate files on the VPN clients determine the IP address that is assigned to them by the server.

The STCD and data server both connect to a VPN server as clients. When connections with the VPN server are established, each client is appointed a new IP address through which other clients can now access it.

Stellenbosch University https://scholar.sun.ac.za

3.3.13.1 OpenVPN

OpenVPN is an open source VPN software solution package maintained by a private company based in California, USA. OpenVPN is a full-featured SSL VPN that supports client certificate authentication [59], [60]. OpenVPN source code can be obtained from [61].

OpenVPN software is run on both the server and client devices. When running OpenVPN, a configuration file must be supplied that specifies how OpenVPN will run. This file can either run OpenVPN in a server or a client mode. Certificate files are generated by the OpenVPN server and copied to client devices enabling them to pass authentication to connect to the server. The server also defines what IP addresses clients will be assigned on the VPN.

The OpenVPN software creates a new interface on the STCD that is used to transfer data to the data server. This interface is known as a virtual tunnel and is usually listed as *tun0* in the ifconfig program. The ifconfig Linux program is used to list all active network interfaces. The tun0 interface is treated as the active network interface and connection to the data server.

3.3.13.2 OpenVPN Configuration

For this project, a OpenVPN server was setup at Stellenbosch University to service the PI server and various STCD units. The server is located behind the Stellenbosch University firewall and issues authentication certificates to all devices that wish to connect to it. Once a device has connected to the server, the server adds the device to the VPN. The device is given a custom IP address on the VPN and a secure link is established between the device and other devices on the VPN.

3.3.14 Network Time Protocol

The Network Time Protocol (NTP) is used to synchronize the STCD clock to a time server source.

For this to take place, the NTP Linux program was installed on the STCD and the correct timezone was selected and set as the default Linux timezone for the STCD. Linux has a set of possible timezones that the default timezone can be set to. These files are found in the */share/zoneinfo* directory. To set a default timezone, a symbolic link is made to the desired timezone file and stored in the */etc* directory under the name *localtime*. For example, to set Johannesburg as the default timezone, the following command is executed:

ln -s /usr/share/zoneinfo/Africa/Johannesburg /etc/localtime

This creates the */etc/localtime* file as a symbolic link to the */usr/share/zoneinfo/Africa/Johannesburg* timezone file.

3.4 SUMMARY

In this chapter, a full description of both the hardware and software elements of the STCD was given.

Stellenbosch University https://scholar.sun.ac.za

The STCD was developed as an open source solution that interfaces with the SRAPET transformer to allow it to meet the Eskom standard [4]. The STCD was developed as a prototype device to be deployed and tested on SRAPET units. The main objective of the STCD is to move the SRAPET towards a fully functional smart transformer. Table 3-5 tabulates the functional objectives set out for the STCD and how they were met. The *Relevant Sections* column contains the section numbers that describe how the specific objective was met in the STCD.

Stellenbosch University https://scholar.sun.ac.za

Table 3-5 - STCD Objective Satisfaction

Table Nr.	Requirement	Description	Required Protocol	Implementation	Relevant Sections
1	Operational monitoring	Non-substation based devices must possess the capability to be monitored remotely by making use of industry standard communication protocols.	Distributed Network Protocol 3	Open-source Distributed Network Protocol 3	3.3.9
2	Configuration interfaces	Remotely monitored devices must also be remotely configurable. Remote configuration of the device includes the capability to: • Apply settings remotely • Update firmware remotely • Change passwords remotely • View access and statistical logs remotely • Extract the sequence of events remotely • View all operational data remotely Remote configuration interfaces must have username and password authentication.	Hypertext Transfer Protocol (HTTP), File Transfer Protocol (FTP) and Secure Shell (SSH)	HTTP web interface, SSH interface via Linux, FTP capability and an XML interface	3.3.2, 3.3.7 and 3.3.10

Stellenbosch University https://scholar.sun.ac.za

3	Network monitoring	The networking functionality of remotely monitored devices must be remotely monitored.	Simple Network Monitoring Protocol (SNMP) version 1 or higher	SNMP functionality via Net-Snmp agent	3.3.6 and 3.3.11
4	Time synchronizatio n	Non-substation based devices must support a method by which their clocks can be synchronized. Internal clock accuracy must be at least 10ms.	Network Time Protocol (NTP) v3	NTP v3	3.3.14
5	Communicatio n interfaces	Remotely monitored devices must support effective communication interfaces to communicate data.	Cellular modem and physical Ethernet port (100Base-T)	A 3G mobile internet modem and an available Ethernet port	3.2, 3.2.3 and 3.3.12
6	Authentication	All remotely monitored devices are to support authentication by means of a username and associated password. Preferably, three levels of authentication should be supported with varying levels of access to the device.	Not specified	Linux authenticati on for SSH access and custom authenticati on for web interface access.	3.3.2 and 3.3.7.4

Stellenbosch University https://scholar.sun.ac.za

7	Security	All non-substation based devices must support encrypted links to the devices. Such encryption must be scalable and support at least two outgoing and incoming connections. Encryption algorithms must support a key of 128 bits or larger. Certificates may be used as the encryption techniques.	Not specified	A Virtual Private Network using OpenVPN software	3.3.13

The STCD was developed to use DNP3 as its main data reporting protocol and SNMP as its main network monitoring protocol. The STCD was connected to the public internet using a commercial 3G modem. The STCD used the OpenVPN software to connect to a private VPN through which it can communicate with a data server.

Various interfaces were developed through which operators can access the STCD and were described in this chapter. These included an HTTP-based web interface, a XML configuration interface and a SSH remote access interface.

Existing software was used in combination with custom-written software to achieve the objectives set out for the STCD in section 3.1.2.

The STCD is a distributed computing device that adds intelligence to the SRAPET distribution transformer. The STCD makes use of a microprocessor to exercise distributed intelligence and contribute towards the effective working of the smart grid.

Stellenbosch University https://scholar.sun.ac.za

CHAPTER 4
DATA SERVER – THE PI SYSTEM

4.1 INTRODUCTION

A data server is the main entity to which remote units communicate data. The data server acts as a data aggregator that collects data from remote sources, stores the data and performs actions on the data. It is as an intelligent entity that can manage data effectively to produce valuable outcomes.

The OSIsoft PI System software was used to realize the data server functionality for this project. The PI System is a real-time data collection, archiving, management, analyses and visualization system that consists of a number of products. Each product adds a unique functionality to the system. The PI System can run on a single, or on various computers. When running on various computers, each computer performs a dedicated task of data collection, storage, analysis or visualization. Dedicated computers will improve system performance, but are not necessary in small systems. A PI System may consist of various servers connected in a hierarchical architecture. Servers on lowest level of the hierarchy receive data from devices in the field. The received data is processed and the processed data is communicated upwards in the hierarchy to another data server. This process continues until data reaches the control center [62], [63].

This hierarchical architecture is used to more effectively distribute the load of data processing. Each data server unit receives data from lower levels in the hierarchy, analyses the data and only reports valuable data upwards in the hierarchy. This process greatly lessens the load on central data servers as data processing takes place in a number of decentralized locations. Raw data is effectively turned into knowledge, and only the knowledge is communicated to the central servers. The concept is illustrated in Figure 4-1. Note that is the context of this project, the remote communication device is the STCD.

Stellenbosch University https://scholar.sun.ac.za

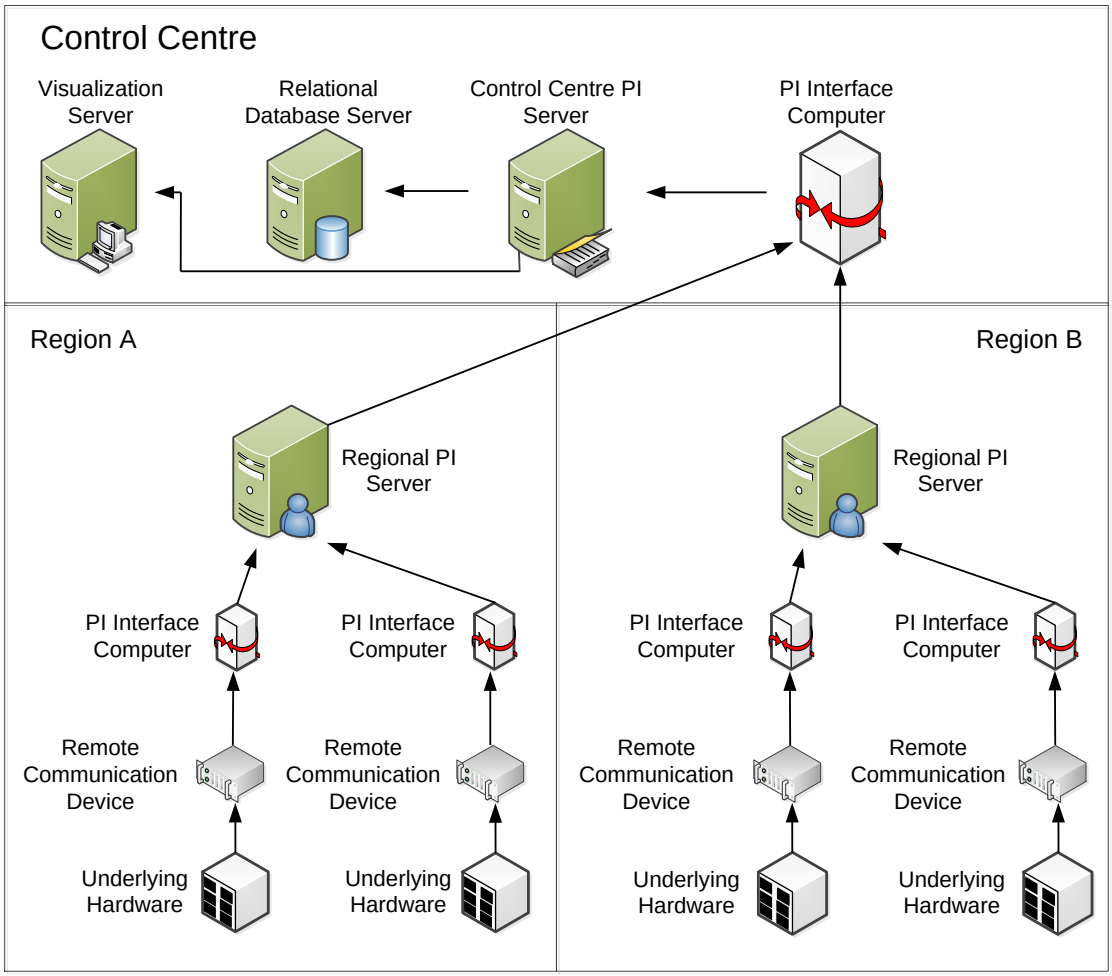


Figure 4-1 - Data Server Hierarchy

Data aggregators transform data into knowledge by identifying data considered important, performing actions on that data and discarding (or locally storing) all non-important data. In Figure 4-1, regional PI Servers identify valuable data received from remote devices in the field and communicate it upwards in the hierarchy to control center data servers. At control center level, servers are used to perform specialized functions such as data storage or visualization. A singular server computer could perform all of these functions, but penalties on performance would be incurred.

For development purposes, a single server was set up for the STCD. The server performs the functions of data gathering, storage, management, analyses and visualization. The STCD data server topology is shown in Figure 4-2.

Stellenbosch University https://scholar.sun.ac.za

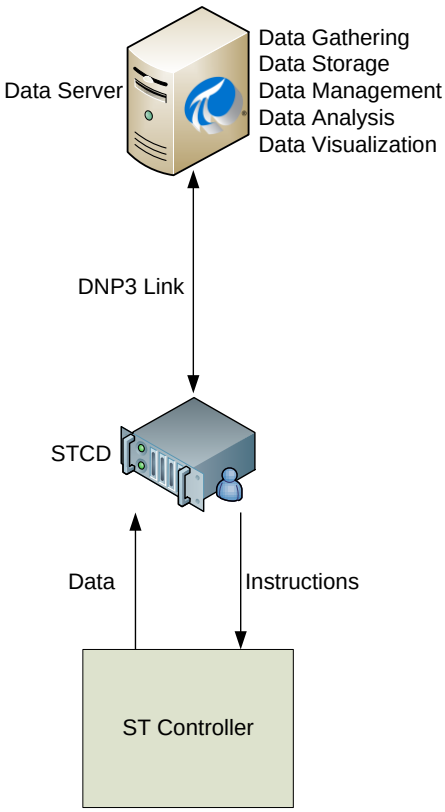


Figure 4-2 - STCD Data Server Topology

4.2 PI SYSTEM PRODUCT OVERVIEW

The main components that make up the PI System are:

- PI Server core software
- PI System Management Tools (SMT)
- PI Asset Framework (AF)
- PI Advanced Calculations Engine (ACE)
- PI ProcessBook (PB)
- PI Interfaces

A brief overview of these products is given in the following sections.

4.2.1 PI Server Core

The PI Server forms the core of the PI System. It stores all incoming data and makes it available to other PI applications.

The PI Server is built on a Microsoft MySQL database foundation. The server stores incoming data from a particular source in data points also known as PI tags. A data point has both a current value and previous values that are identified by unique timestamps. It also has a type and various configuration options that allow a user to define the data point. A single remote device may provide the PI Server with multiple data points. For example, a single device may report a temperature, voltage reading, current reading and pressure reading to the PI Server. The amount of data points

Stellenbosch University https://scholar.sun.ac.za

a single remote device can supply is only limited by the interface being used to report the data to the PI Server if such limits exist. For example, if a particular communication protocol limits the amount of data points a device may contain.

The PI Server offers various functionalities including:

- Data point creation.
- Data processing functionalities including data classification, data compression (a process that maximizes storage efficiency that will be discussed later in this section), basic data manipulation and various tools that vary depending on the interface being used to gather the data.
- Data management functionalities including archiving, backups and batch process monitoring and management.
- Database security.
- PI System performance monitoring. This includes monitoring of the PI System itself, all networking interfaces being used by the PI Server, all data contained in the PI Server and the underlying database powering the PI Server.

The functionalities offered by the PI Server are utilized using the PI System Management Tools (SMT). PI SMT offers a GUI for users to easily manage PI Server operations.

Data Point Creation

Users create data points using the PI SMT and link these points to external or internal data sources. A data point is identified by a unique name and a point source. Point sources are used to group data that have some common identifier, the most obvious being the source that generates them.

A data point can also be given one of the following variable types:

Table 4-1 - PI System Data Types [64]

Type Name	Description	Size	Range
Digital	Used for points with discrete values	-	ON / OFF, RED / BLUE / GREEN etc.
Int16	Integer value	16 bits	0 to 32767
Int32	Integer value	32 bits	-2147450880 to 2147483647
Float16	Scaled floating point number	16 bits	-

Stellenbosch University https://scholar.sun.ac.za

Float32	Single-precision floating point number, not scaled	32 bits	-
Float64	Double-precision floating point number, not scaled	64 bits	-
String	Used to store a string of data	976 bytes	976 characters
Blob	Binary large object	976 bytes	976 bytes
Timestamp	Any Time / Date value	-	1-January-1970 to 1-January-2038

The unit of measurement that the data should be measured in is also defined by the user.

Archiving of data contained in data points can be defined by a user. How much incoming data for a data point is stored or how often it is stored is defined by the user. A user has the ability to define whether all, or only some values recorded by a data point should be stored in the PI database. The criteria determining whether data should be stored or not is based on the deviation of a new reading from readings taken previous to it. If a value deviates significantly from previous values, it is stored, if not, it is discarded. The amount of deviation that is considered significant is defined by the user. Data can also be stored according to set time intervals.

A data point may also have various configuration parameters that vary according to which interface was used to acquire it. For example, using the PI DNP3 interface, a user can specify the DNP3 index number, DNP3 address and DNP3 object type of the data point. These configuration points may also be used to specify DNP3 internal indications points [65].

Data Compression

Stellenbosch University https://scholar.sun.ac.za

The PI Server optionally performs data compression on incoming data. This technique processes data in a data point and stores incoming data only if there is a significant change from previous data readings. An alghorithm is applied in which a user specifies a range in which data can deviate from previous readings without being considered significant. If a data reading is taken that deviates more than the specified range, the data reading is stored and all readings before it that did not deviate significantly are discarded. The technique is illustrated in

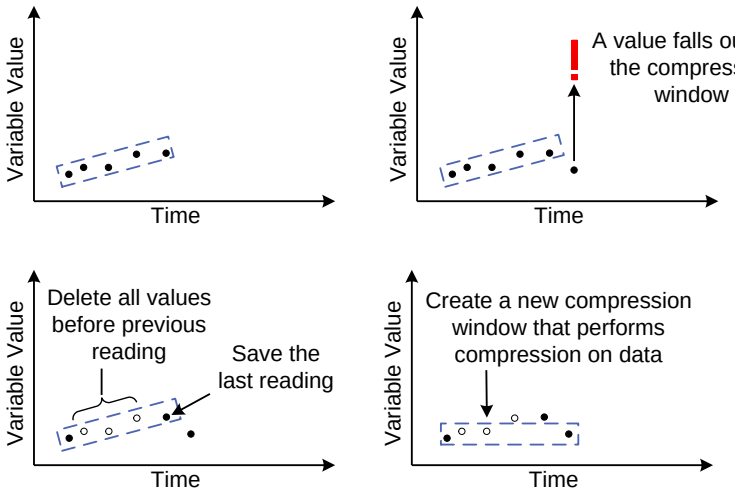


Figure 4-3.

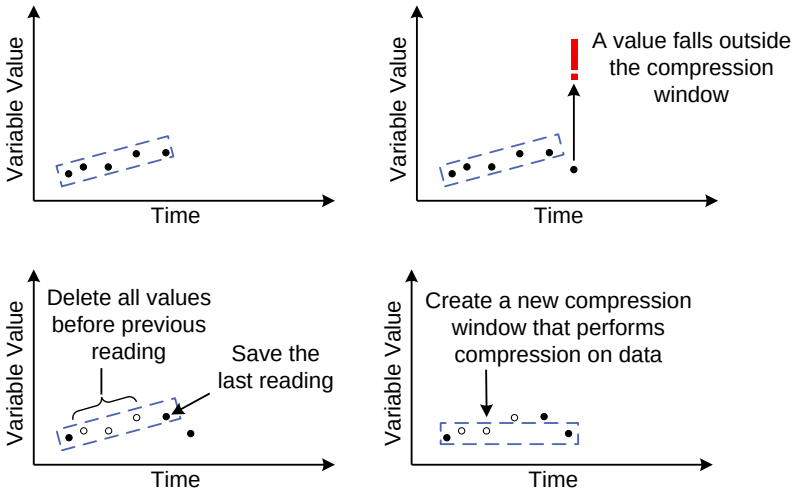


Figure 4-3 - PI Data Compression

PI System Security

The PI System uses a system known as PI Trusts to allow certain users, applications or devices access to the PI System.

A PI Trust is a security system that resembles a certificate security system. PI Trusts associate users, hostnames or applications with PI System users. PI System users can be created with a certain level of access to the PI System. Thus, PI Trusts allow external users a certain amount of access to the PI System without requiring a username and password.

Stellenbosch University https://scholar.sun.ac.za

PI Trusts are defined to allow access to the PI System based on a number of connection criteria. These criterea can include:

- Application name (of the application attempting to access the PI System)
- Domain name
- IP address
- Host
- Operation system user name

Criterea can either be implemented individually or in combination for a PI trust. The more criterea are implemented, the more access to the PI Server is restricted for that trust.

PI Trusts are also used to allow interfaces from various data sources access to the PI Server. It creates a virtual tunnel through which data can pass into and out of the PI System .

4.2.2 PI Asset Framework

The PI Asset Framework (AF) software allows users to effectively organize and manage data. The software allows the creation of object-oriented data stores. A PI AF object may represent anything that a user feels significantly links various data points. An example would be a transformer, with all of the readings taken on it as data points.

PI AF allows a user to create data objects for monitored assets and organize these assets in a hierarchical manner. Organization may take place according to asset type, geographical position or any other format that a user desires.

For this project, the ST unit was seen as a single asset with various data points representing parameters being monitored.

Event Frames

PI AF allows users to create event-driven objects known as *Event Frames.* Event Frames allow users to search for data in terms of significant events. Data can thus be organized according to events that take place on assets such as downtimes, overvoltages etc. This helps users to easily locate significant events that occured to assist in processes, such as of fault analysis and investigation.

4.2.3 PI Advanced Calculation Engine

The PI Advanced Calculation Engine (ACE) is a software tool that allows users to perform calculations on data stored in the PI System.

PI ACE consists of the PI ACE Wizards, the PI ACE Manager and the PI ACE Scheduler. Users use the wizard to create the framework code required to perform calculations on data. The framework code generated using the wizard imports data points into an Integrated Development

Stellenbosch University https://scholar.sun.ac.za

Environment (IDE) (either Microsoft Visual Basic or Microsoft Visual Studio with .NET) and allows the user to manipulate data points using programming languages.

Through ACE, a user has access to a PI data point in the form of a programable variable. The variable may be manipulated either individually or in context with other variables, forming equations. This is done using the PI ACE Manager: a program that allows users to manage ACE equations and schedule ACE tasks.

The PI ACE Scheduler allows a user to run created equations at set times. ACE equations may be performed periodically and the results stored in PI data points. The resulting data points may then be stored, visualized or manipulated in the same way as any other data point [66], [67].

4.2.4 PI Processbook

PI Processbook (PB) is the main visualization software tool used in the PI System.

PI PB allows users to visualize data points stored in the PI System in a variety of ways including:

- Plotting data points on graphs.
- Using widgets such as bars and smart objects* to display data points.
- Dsiplaying data points as raw text and values.

*Smart objects are objects that change in appearance when the data that they represent changes. An example is a square that changes from green to red when the data value it represents exceeds a certain threshold.

Using these techniques, PI PB displays are constructed. A PB display consists of the visualization of various data points on a single screen.

Various PB displays may be linked to one another, allowing a user to navigate between displays. This is useful when displays are set up in a hierarchical manner. For example, one display may contain a map showing the position of all distribution transformers in an area. When a user selects one of the transformer units, another display containing information on the specific unit is opened.

4.2.5 PI Interfaces

The PI System uses various interfaces to collect data. These interfaces can each be added to the PI System as individual members.

An interface is a mechanism by which the PI System reads data in from a particular source using a specific protocol. This allows the PI System to access data from a variety of data sources. The goal of every PI interface is to allow the PI System to communicate with a desired data source.

PI interfaces can be run as Windows services. This means that interfaces run continuosly in the background, monitoring incoming data from data sources

The PI DNP3.0 interface collects data from STCD units. The PI DNP3.0 interface enables the PI System to function as a DNP3 master station, collecting data from various DNP3 outstations. Data read in via the PI DNP3.0 interface is stored in data points. The PI DNP3.0 interface can monitor

Stellenbosch University https://scholar.sun.ac.za

multiple DNP3 outstation devices simultaneously. IP and DNP3 addresses are entered into the interface and a DNP3 connection is established between the PI System and the outstation.

The PI DNP3.0 interface can either poll DNP3 outstation devices for data, or can receive unsollicated DNP3 messages from outstations [65]. Other functionalities such as integrity scans and DNP3 internal indications are also available through the PI DNP3.0 interface.

4.3 FIELD TESTING OF THE STCD

This section contains results of field-testing performed on the STCD.

To test the STCD, two units were deployed on distribution transformers in rural areas. The areas of deployment are located in the Eastern Province and can be seen in Figure 4-4.

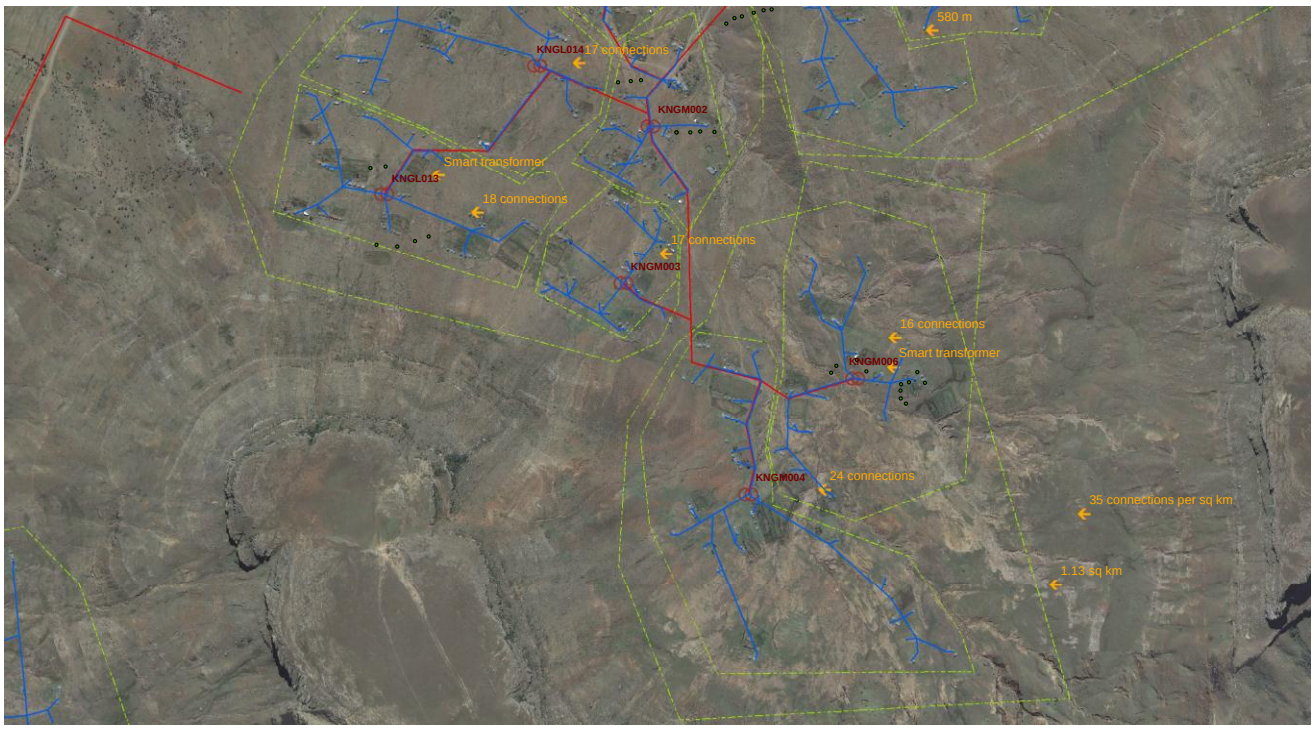


Figure 4-4 - Map of Deployed Units

The STCD units were deployed on two 16 kVA SRAPET distriubtion transformers. A distribution transformer in the field is shown in Figure 4-5.

Stellenbosch University https://scholar.sun.ac.za

Figure 4-5 - Pole-Top Distribution Transformer

The remote monitoring and configurational capabilities of the STCD were tested. The main protocol for monitoring the STCD is DNP3. The SNMP protocol is used to monitor network statistics and performance. Configuration of the STCD takes place over the web and XML interfaces.

4.3.1 DNP3 Testing

The DNP3 interface was set up on a STCD unit with the following Level-Crossing Sampling (LCS) parameters. The parameters were configured using the web interface.

Stellenbosch University https://scholar.sun.ac.za

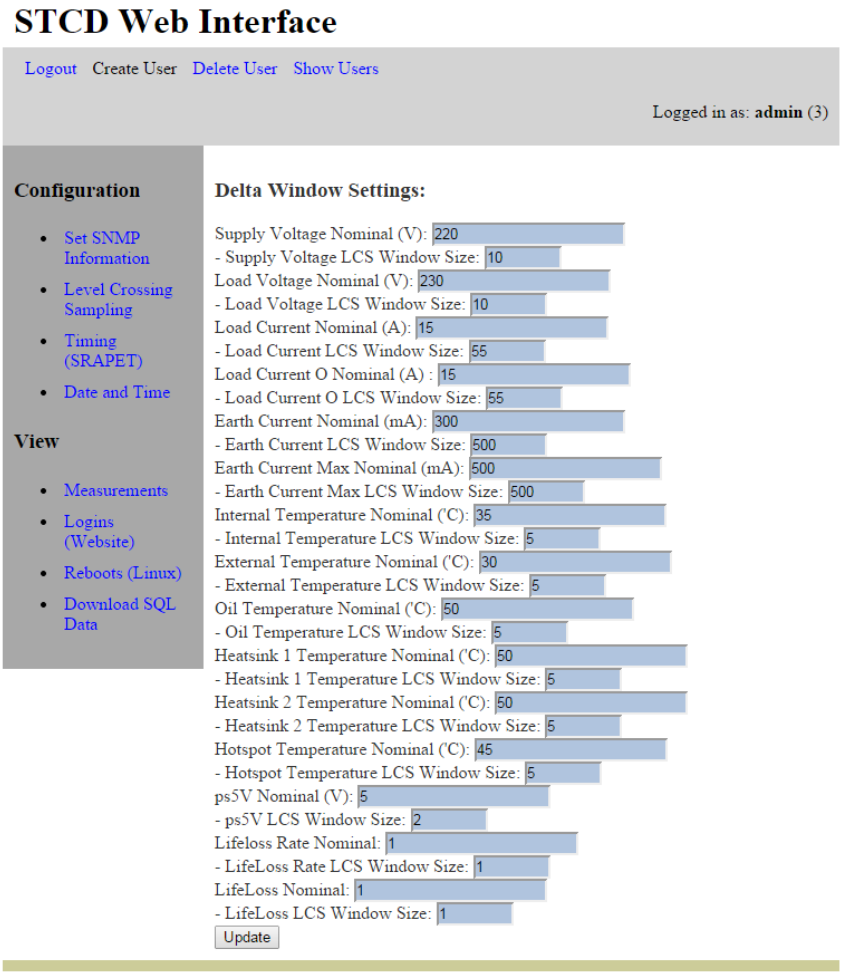


Figure 4-6 - LCS Values for Field Testing

The STCD generates data according to the configuration that a user issues using the web or XML interface. Data is communicated by the STCD unit to the PI System. The data is stored on the PI System as a data point. A data point is an array of entries, each with a timestamp and a value that pertains to data gathered from a certain variable on a source. An example is load voltage on the STCD. The PI System Management Tool (SMT) can be used to view archived data values for a data point. Figure 4-7 shows values for the STCD source voltage reading averages taken every 5 minutes over a period of one hour.

Stellenbosch University https://scholar.sun.ac.za

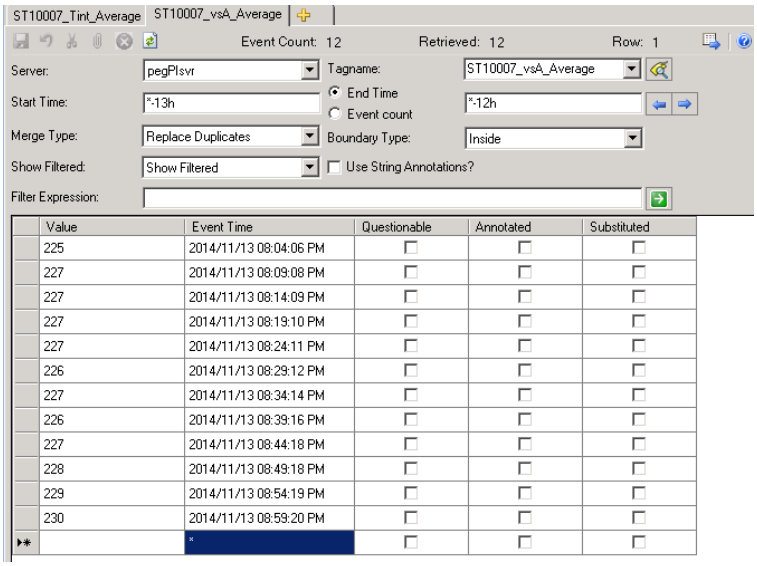


Figure 4-7 - PI System Management Tools Average Data

The PI System has comprehensive search tools that allow users to easily find data stored within its database. Data from various sources are uniquely identified and can be retrieved for user-defined time intervals. In Figure 4-7, it can be seen that the search values for *Start Time* and *End Time* are "*-13h" and "*-12h" respectively. The * symbol represents the current time. Thus, this timespan requests data gathered between thirteen and twelve hours ago (a one hour timespan).

The PI ProcessBook software is used to display data in a variety of visual formats. Data variables can be displayed individually or in comparison to other variables. Figure 4-8 shows the average load current of the STCD displayed on a graph over a period of 24 hours.

Stellenbosch University https://scholar.sun.ac.za

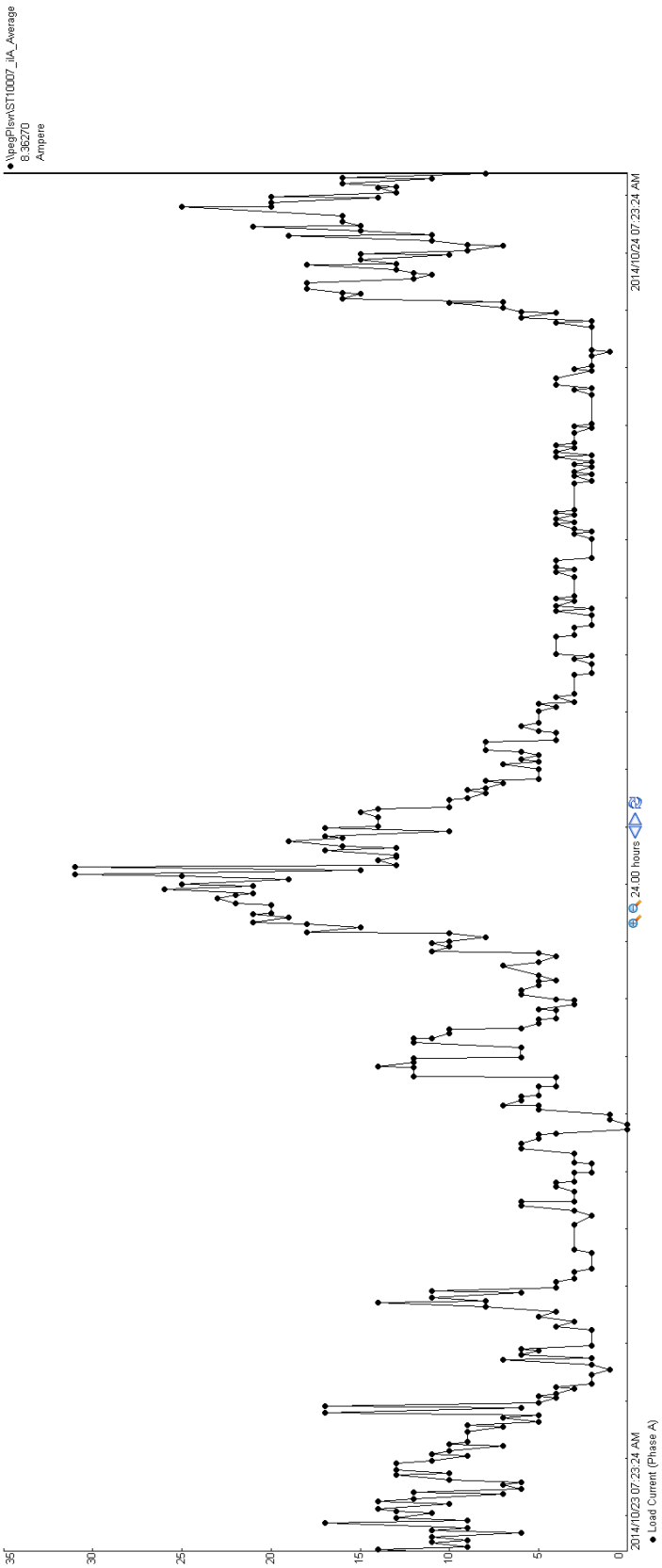


Figure 4-8 - ST Load Current Averages (24 hours)

Stellenbosch University https://scholar.sun.ac.za

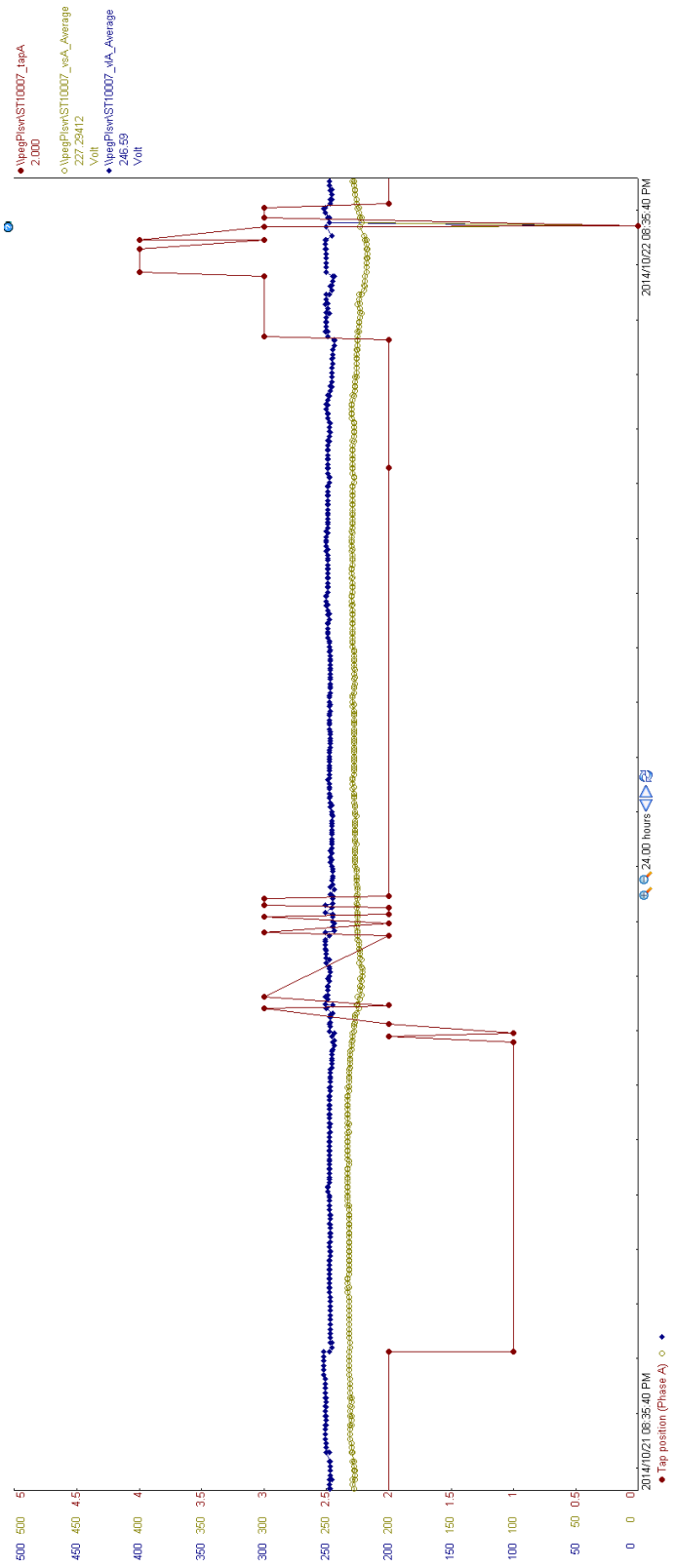


Figure 4-9 - ST Source Voltage Averages, Load Voltage Averages and Tap Position (24 hours)

Stellenbosch University https://scholar.sun.ac.za

Figure 4-9 shows the ST average load voltage, average source voltage and transformer tap position all displayed on the same graph. Graphs such as this can easily and quickly be created using PI ProcessBook and offer valuable insight into the operations of the transformer being monitored. For example, Figure 4-9 shows how the tap position of the ST changes as the voltage varies. Note that there are multiple value scales on the left-hand side of the graph. Each data point has its own value scale according to the range which that data point spans. Various data point plots are distinguished according to colour.

A PI ProcessBook display was created to monitor the various data points available on the STCD. The display aims to give an overview of transformer operations and can be used to gain a more in-depth understanding of operations. Figure 4-10 shows the screen for data gathered over a period of six days. As can be seen from the figure, voltages, currents and temperatures are plotted against time with current (in context of time) values listed on the left-hand side of the screen.

The box on the left-hand side of the screen contains the latest measurements taken for all data points on the ST. The left-hand side graph plots the ST source voltage average, load voltage average and tap position against time. The centre graph plots the ST load current and earth current against time. The right-hand side graph plots the ST interior temperature, exterior temperature, top oil temperature, heatsink temperature and hotspot temperature against time. In the top right of the display, a connection status of all connections on the DNP3 interface is given. Note that one connection to a STCD unit is offline due to a faulty antenna connector.

Stellenbosch University https://scholar.sun.ac.za

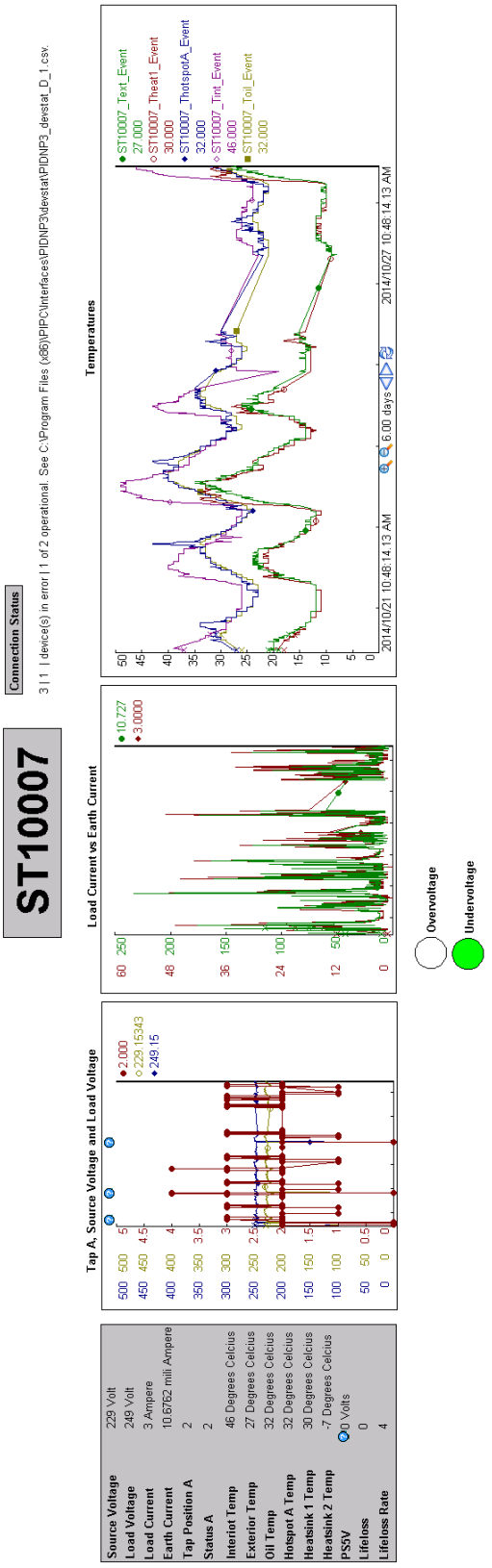


Figure 4-10 - DNP3 Data Display (6 days)

Stellenbosch University https://scholar.sun.ac.za

The time-scale for the graphs can easily be adjusted using the PI ProcessBook software. In Figure 4-11 and Figure 4-12 load and earth current average measurements (Figure 4-11) are compared to load and earth current event measurements (Figure 4-12) for a period of one day. A dot on the graph represents a reading taken. It can be noticed that averages are taken at constant intervals while event measurements are received sporadically as events occur.

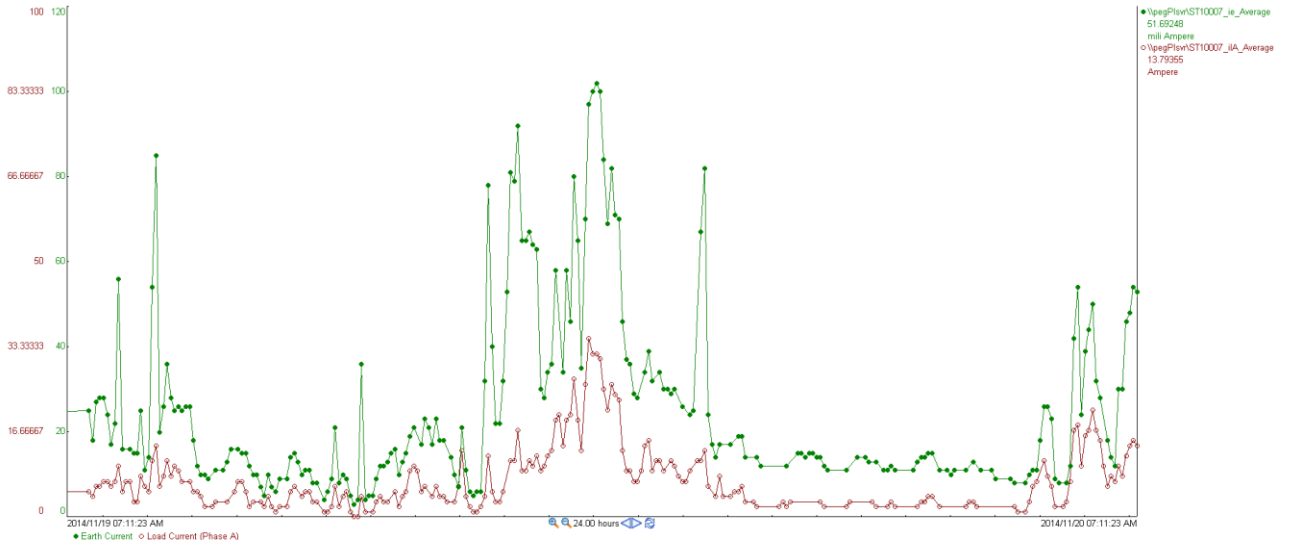


Figure 4-11 - Load and Earth Current Averages (1 day)

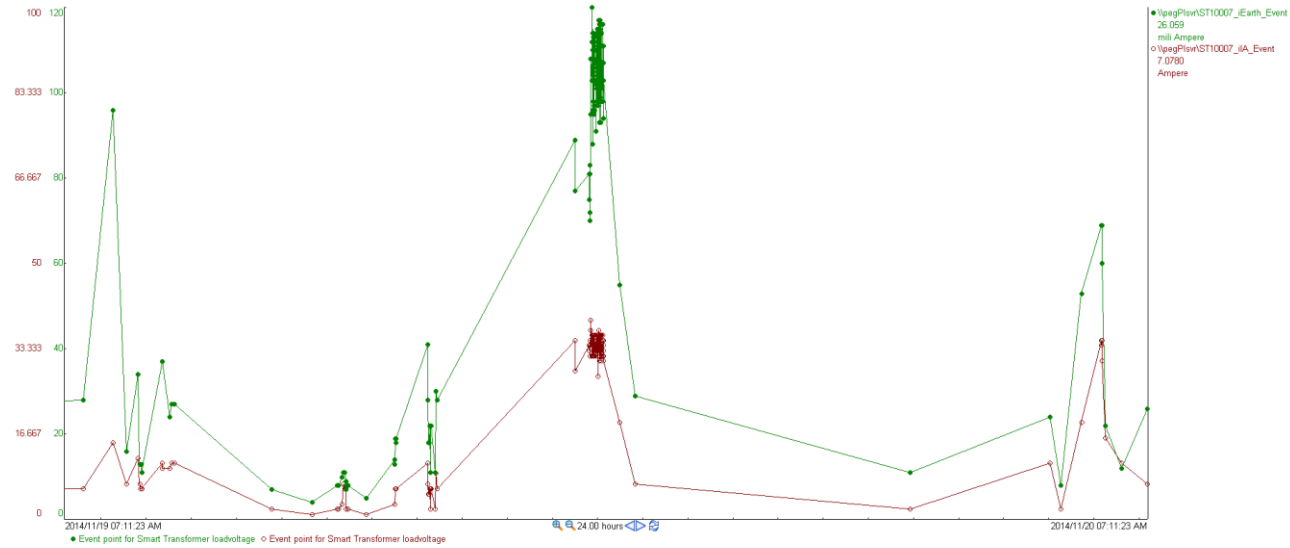


Figure 4-12 - Load and Earth Current Event Readings (1 day)

Data point average readings are reported at fixed intervals. Data readings taken from the SRAPET over a user-defined time are averaged and the averages are reported via DNP3. This can be seen in Figure 4-11 where data point averages for load and earth current were reported every five minutes.

Data point event reports are reported when any data point crosses an LCS window. This is shown in Figure 4-12 where load and earth current events were reported sporadically. All data points are reported when any one data point crosses to a new LCS window. This is done as DNP3 overheads

Stellenbosch University https://scholar.sun.ac.za

constitute a large part of the DNP3 message frame and thus it makes sense to include more measurement data in event reports for bandwidth considerations. From the comparison between average reports and event reports in Figure 4-11 and Figure 4-12, it can be seen that average reports give an overview of ST operations while event reports can be used to show the extremes of operation and to record significant events.

More events are reported in times of greater variation of data points. If one data point crosses many LCS windows in a short period of time, all data points are reported with the data point that is varying significantly. Thus, many event reports can be generated in quick succession for points that do not vary greatly within that time.

Rural Communication Issues

During field testing, it was found that communication with one STCD unit was lost. It was noted during installation that the antenna connection to the 3G modem for this STCD was not reliable. It is believed that the antenna connector disconnected from the 3G modem and thus a connection to the STCD was not possible.

It was also found that only a GPRS connection is possible in the deep rural areas of application for STCD field tests. Thus, a GPRS modem can be used instead of a 3G modem.

As a possible solution to the problem of the unreliable antenna connector, another modem is suggested for installation with the STCD. The modem suggested is the U-Blox GT100 modem. The GT100 is a GPRS modem that supports external antennas and has its own power supply (driven from 230V and stepped-down to 12V DC). The modem is accessed either by a RS-232 or a USB interface. The modem has a screw-in antenna slot that is more sturdy and reliable than the slot of a 3G dongle.

4.3.2 SNMP Monitoring

The ManageEngine MibBrowser software [68] was used to monitor SNMP data on the STCD. This tool allows basic SNMP monitoring functionality such as gathering various SNMP data variables from an SNMP agent.

Stellenbosch University https://scholar.sun.ac.za

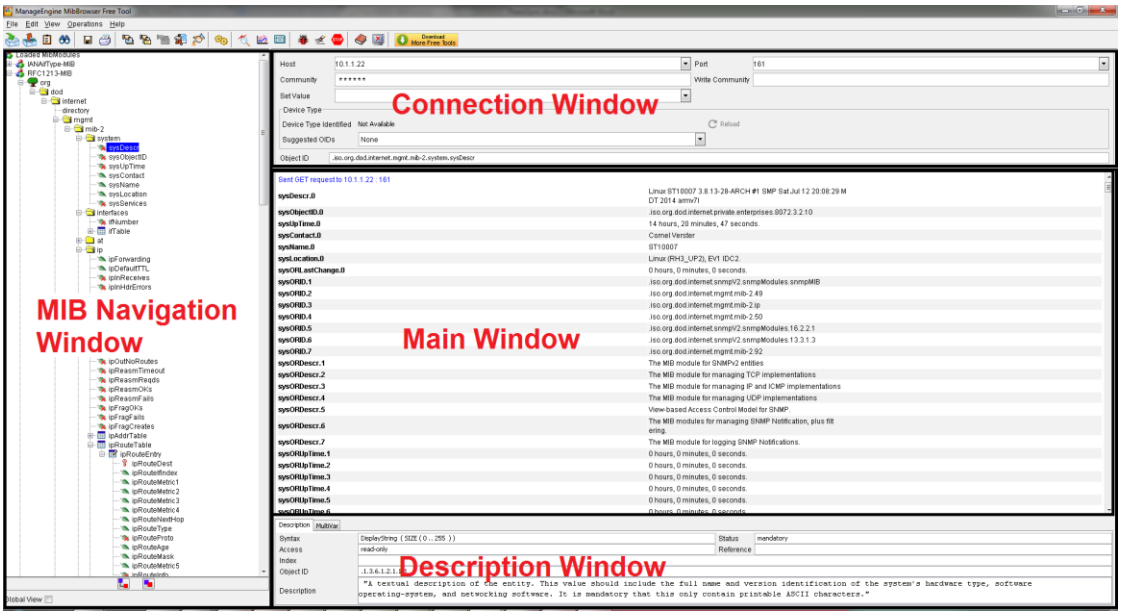


Figure 4-13 - MibBrowser Main Screen

Figure 4-13 shows the MibBrowser main screen layout. The connection window is used to establish a connection to the desired SNMP agent. The agent is specified by inserting a host address and required authentication if necessary. The MIB navigation window allows users to navigate through the structure of MIBs and select the SNMP data variables they would like to request from the SNMP agent. The description window contains a description of the selected SNMP data variable. The main window contains the results of requests for data sent to the SNMP agent.

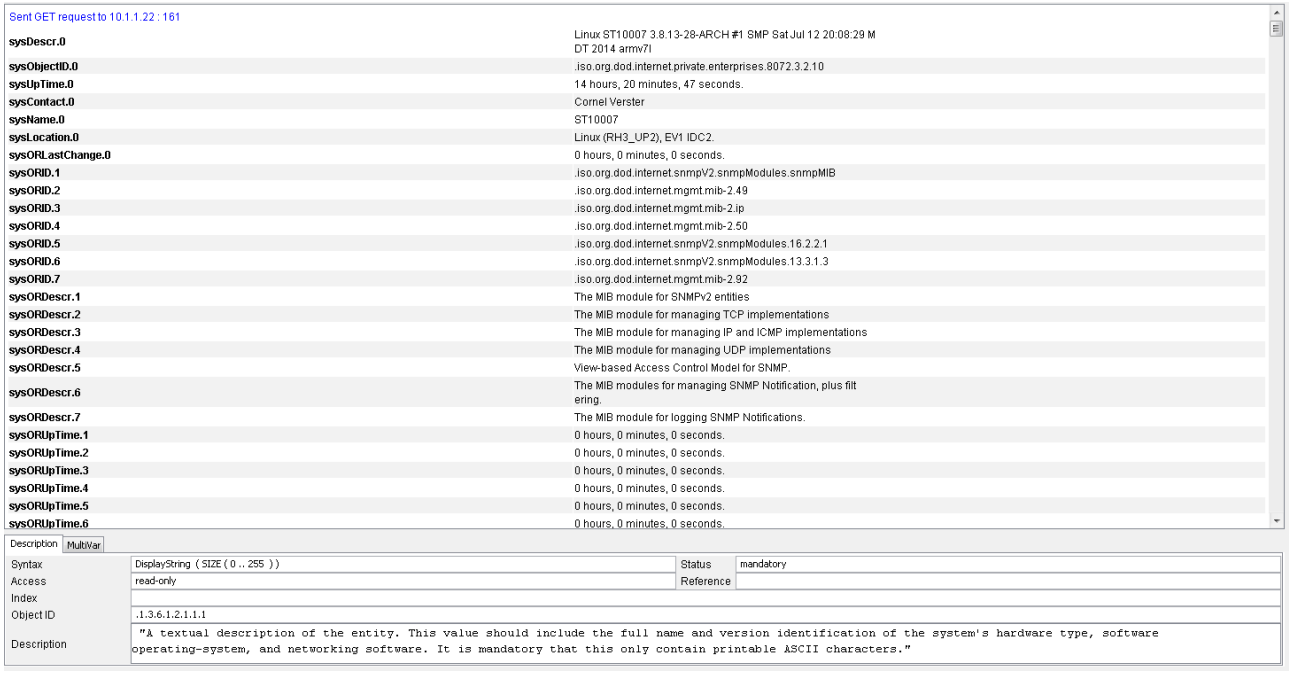


Figure 4-14 - System Description Information

Stellenbosch University https://scholar.sun.ac.za

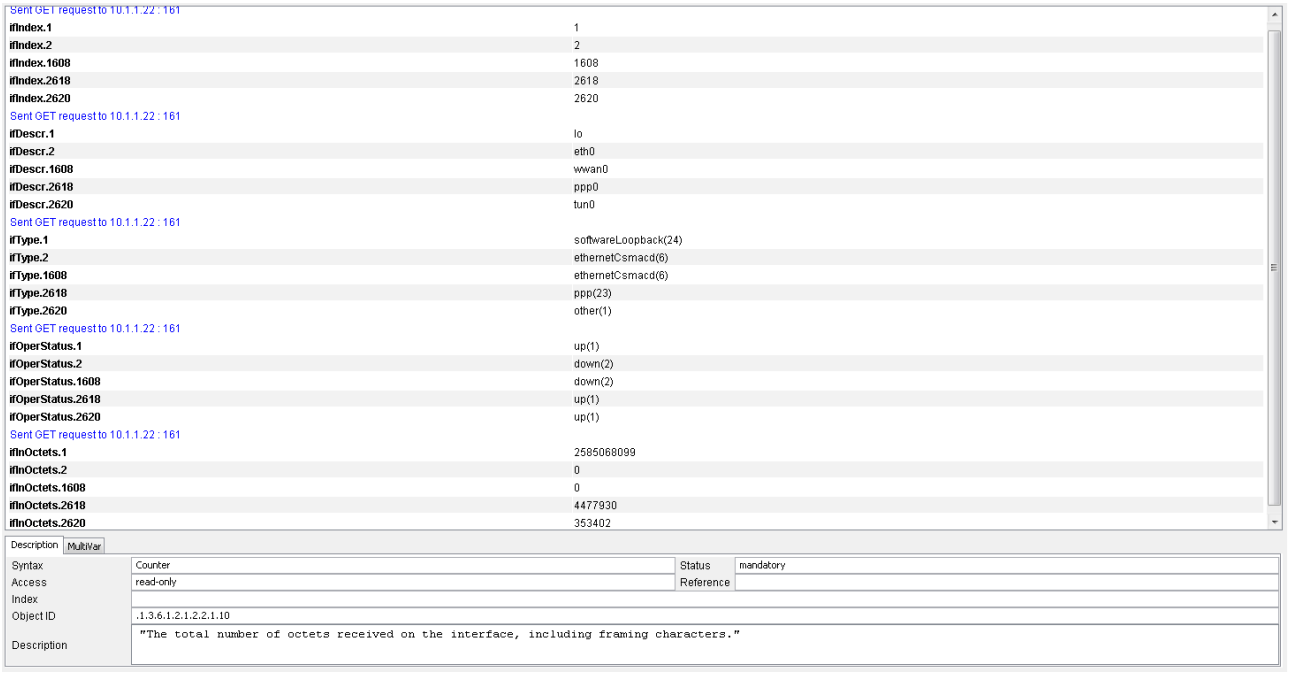


Figure 4-15 - Interface Table Information

Figure 4-14 and Figure 4-15 give STCD SNMP device system description and interface information respectively. The system description provided by SNMP variables include system name, contact information, system location etc.

The interface information supplies information about all network interfaces on a given device. Each interface is allocated an interface number (ifIndex). The ifDescr and ifType variables give a description of the interface and the type of interface respectively. The variable ifDescr.1 is a description for interface 1 with index number ifIndex.1.

From Figure 4-15 it can be seen that the STCD ppp0 interface was assigned index number 2618 while the tun0 interface (the VPN tunnel) was assigned index number 2620. The ppp0 interface is synonymous for the mobile internet connection interface. The ppp0 interface is identified as a PPP (Point-to-Point Protocol) interface under the ifType variable. The ifOperStatus variable indicates the operational status of the interface. The ifInOctets variable shows the total number of data octets received by the interface.

Figure 4-16 shows results for data retrievable for the custom developed PEG-MIB. The results show data that was available for retrieval from the 3G modem. The following three variables required by the Eskom standard are not retrievable from the 3G modem and are thus recognized as *not available*:

- The mobile network code
- The mobile location area code
- The connected mobile cell tower ID

Stellenbosch University https://scholar.sun.ac.za

Sent GET request to 10.1.1.22 : 161	
mobGsmNetOp.0	MTN-
Sent GET request to 10.1.1.22 : 161	
mobNetAttach.0	WCDMA
Sent GET request to 10.1.1.22 : 161	
mobSigStr.0	30
Sent GET request to 10.1.1.22 : 161	
mobSIMSecStat.0	0
Sent GET request to 10.1.1.22 : 161	
mobModImei.0	867547010000406
Sent GET request to 10.1.1.22 : 161	
mobModManufac.0	huawei
Sent GET request to 10.1.1.22 : 161	
mobModModel.0	E3131B
Sent GET request to 10.1.1.22 : 161	
mobFirmVer.0	21.157.48.00.864

Figure 4-16 - PEG-MIB Results

The sent and received bytes and packets for each interface representing a GSM connection or attached device ports (serial or Ethernet) are also required by the Eskom standard [4]. These values are contained within the ifTable SNMP table in the IF-MIB module and readings from the STCD are displayed in Figure 4-17. The ifDescr variable gives a description of each interface assigned to a certain index number.

Stellenbosch University https://scholar.sun.ac.za

ifDescr.1	lo
ifDescr.2	eth0
ifDescr.5570	wwan0
ifDescr.5742	ppp0
ifDescr.5743	tun0
Sent GET request to 10.1.1.22 : 161	
ifInOctets.1	3813352619
ifInOctets.2	0
ifInOctets.5570	0
ifInOctets.5742	601297
ifInOctets.5743	298062
Sent GET request to 10.1.1.22 : 161	
ifInUcastPkts.1	81062894
ifInUcastPkts.2	0
ifInUcastPkts.5570	0
ifInUcastPkts.5742	4301
ifInUcastPkts.5743	3930
Sent GET request to 10.1.1.22 : 161	
ifOutOctets.1	3813645300
ifOutOctets.2	0
ifOutOctets.5570	0
ifOutOctets.5742	655325
ifOutOctets.5743	361268
Sent GET request to 10.1.1.22 : 161	
ifOutUcastPkts.1	81065950
ifOutUcastPkts.2	0
ifOutUcastPkts.5570	0
ifOutUcastPkts.5742	4370
ifOutUcastPkts.5743	3994

Figure 4-17 - PEG-MIB In and Out Bytes and Packets

4.3.3 Remote Access and Operation

During the field-testing period for the STCD, remote access to the device was used to perform various updates, debugging and maintenance operations. The File Transfer Protocol (FTP) and Secure Shell (SSH) protocols were used to perform these operations.

During the field testing period, some bugs were noted in the software installed on the STCD. One such bug was that of incorrect reporting of LCS events due to faulty code written for the ST undervolage indicator. The code detected permanent undervoltage.

After identifying the bug, the relevant code was fixed by editing the program code remotely at Stellenbosch University. After the code had been edited, it was uploaded to the STCD in the field using the FTP protocol and installed on the device using the SSH protocol.

This shows that effective updating and debugging is possible remotely using the FTP and SSH protocols available on the STCD.

Stellenbosch University https://scholar.sun.ac.za

XML Interface Testing

The XML interface was tested by sending an XML configuration file containing desired configuration settings to the STCD using the FTP protocol. The XML file is copied to the */srv/xml/watch* directory, monitored by the XML notifier program. The following XML configuration file was copied to this folder while the STCD was in the field.

```
<?xml version="1.0" encoding="utf-8"?>
<configuration version="1.0" type="config">
        <deltas svn="220" svd="10" lvn="230" lvd="10" lcn="15" lcd="10" lcon="15" lcod="10"
            ecn="300" ecd="100" ecmn="500" ecmd="150" itn="35" itd="10" etn="30" etd="10"
            otn="50" otd="10" h1tn="50" h1td="10" h2tn="50" h2td="10" hstn="45" hstd="10"
            ps5vn="5" ps5vd="10" llrn="1" llrd="50" lln="1" lld="50">
        </deltas>
</configuration>
```

The resulting update to the LCS variables can be seen using the web interface. The LCS values gathered by the web interface from the MySQL database after the XML configuration file was sent is displayed in Figure 4-18.

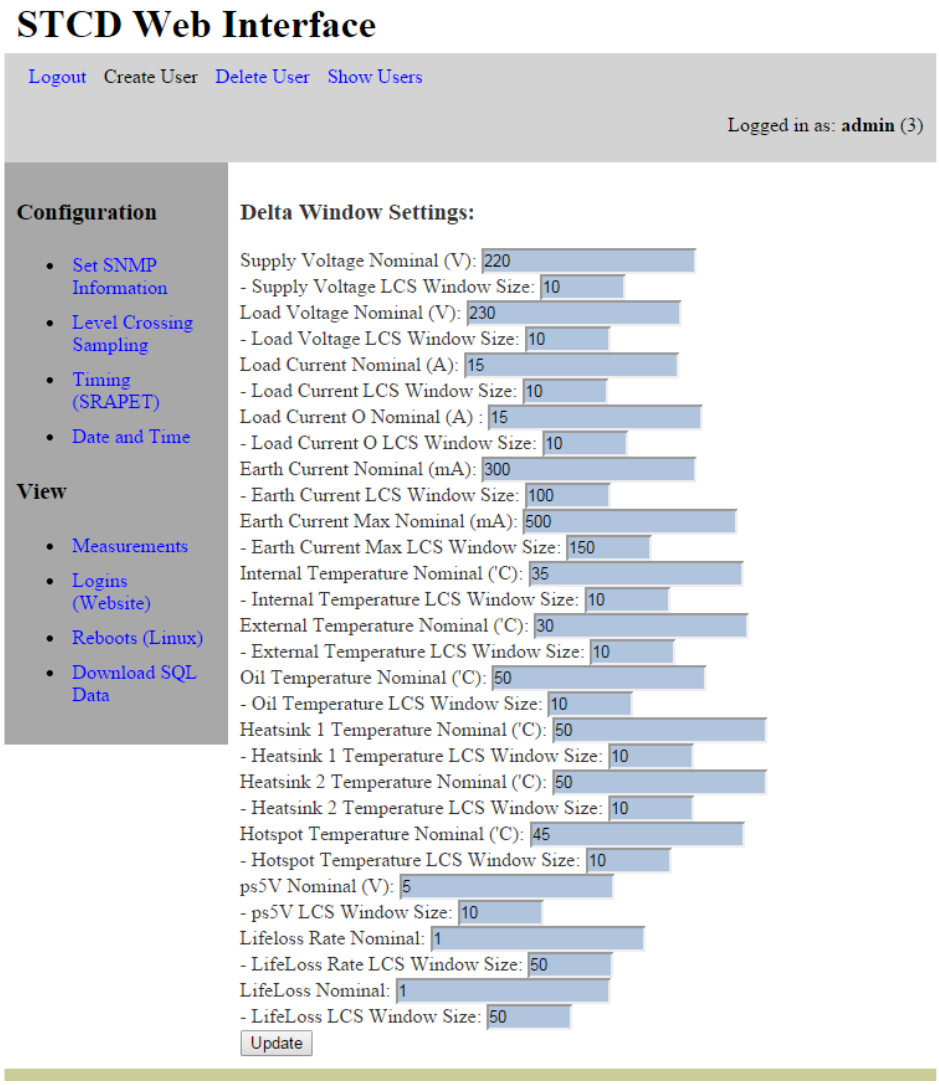


Figure 4-18 - LCS Values after XML Update

Stellenbosch University https://scholar.sun.ac.za

A record is kept in the MySQL database of all XML configuration updates. Update records are stored in the *xmlconfigstats* table and are shown in Figure 4-19.

```
MariaDB [db]> select * from xmlconfigstats;
+----+---------------------+---------+
| id | time                | result  |
+----+---------------------+---------+
|  1 | 2014-07-26 03:32:39 | success |
|  2 | 2014-11-21 11:12:08 | success |
+----+---------------------+---------+
2 rows in set (0.00 sec)
```

Figure 4-19 - The xmlconfigstats MySQL Table

In-Field Operational Logs

Logs of web interface logins and STCD reboots are kept in MySQL tables. The web interface login table is showed in Figure 4-20.

```
MariaDB [db]> select * from loginstats;
+----+---------------------+----------+----------+-----------+-----------------+
| id | time                | username | password | operation | ip              |
+----+---------------------+----------+----------+-----------+-----------------+
|  1 | 2014-07-26 03:32:58 | admin    | admin    | success   | 146.232.222.231 |
|  2 | 2014-07-29 10:01:35 | admin    | admin    | success   | 10.1.1.34       |
|  3 | 2014-07-29 10:01:36 | admin    | admin    | success   | 10.1.1.34       |
|  4 | 2014-07-29 20:52:14 | admin    | admin    | success   | 10.1.1.50       |
|  5 | 2014-07-31 08:50:02 | admin    | admin    | success   | 10.1.1.34       |
|  6 | 2014-10-21 14:21:38 | admin    | admin    | success   | 146.232.222.231 |
|  7 | 2014-10-21 15:22:45 | admin    | admin    | success   | 146.232.222.231 |
|  8 | 2014-10-23 09:29:37 | admin    | admin    | success   | 146.232.222.231 |
|  9 | 2014-11-06 10:27:00 | admin    | admin    | success   | 146.232.222.231 |
| 10 | 2014-11-06 10:31:32 | admin    | admin    | success   | 146.232.222.231 |
| 11 | 2014-11-12 17:16:32 | admin    | admin    | success   | 146.232.222.231 |
| 12 | 2014-11-13 12:51:55 | admin    | admin    | success   | 146.232.222.231 |
| 13 | 2014-11-20 15:22:09 | admin    | admin    | success   | 146.232.222.216 |
| 14 | 2014-11-20 15:23:21 | admin    | admin    | success   | 146.232.222.231 |
| 15 | 2014-11-21 10:58:07 | admin    | admin    | success   | 146.232.222.231 |
+----+---------------------+----------+----------+-----------+-----------------+
15 rows in set (0.01 sec)
```

Figure 4-20 - STCD Web Interface Login Statistics

The most recent reading received by the STCD from the SRAPET can be viewed using the web interface and is shown in Figure 4-21.

Stellenbosch University https://scholar.sun.ac.za

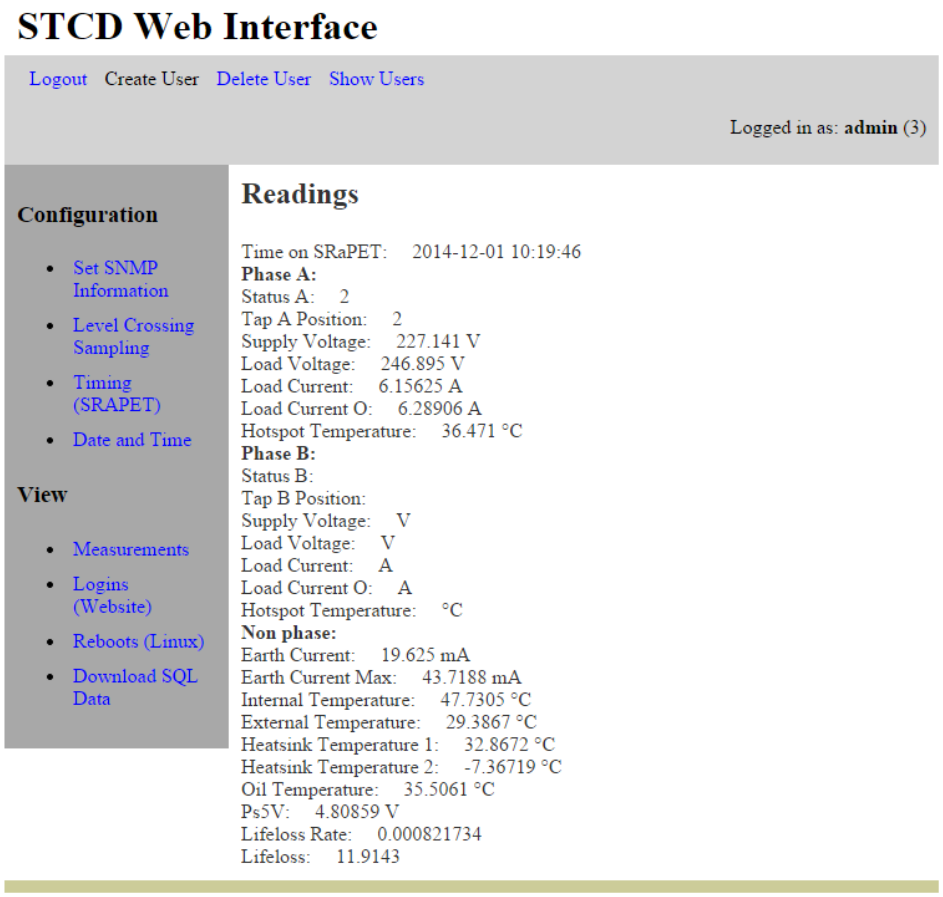


Figure 4-21 - Web Interface ST Readings

Network Time Protocol Testing

NTP was tested and found to work correctly. When the STCD creates an internet connection via the mobile internet modem, NTP automatically updates the current data and time of the STCD by synchronizing its system clock with a remote time server. Figure 4-22 shows the *date* Linux command used to show that the time on the STCD has been updated.

```
Last login: Mon Nov 24 12:36:27 2014 from 146.232.222.231
[root@ST10007 ~]# date
Sat Nov 29 14:29:51 SAST 2014
```

Figure 4-22 - Network Time Protocol Results

VPN Field Testing

The STCD units deployed in the field were set up to make a connection to a VPN server located at Stellenbosch University via a mobile internet connection. The connection to the VPN was successfully established. The data server also connects to the VPN server and thus communication between the data server and the STCD units in the field are possible via the encrypted VPN links. Connection information obtained using the Linux *ifconfig* program is displayed in Figure 4-23. Information for both the mobile internet connection (ppp0) and the VPN connection (tun0) is shown. It is evident that the connections are active and sending and receiving

Stellenbosch University https://scholar.sun.ac.za

traffic (this is seen from the TX (sent) and RX (received) packets for each interface). The Ethernet interface (eth0) is inactive as no Ethernet cable is connected to the device.

```
[root@ST10007 ~]# ifconfig
eth0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether 90:59:af:5c:23:20  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 40

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 0  (Local Loopback)
        RX packets 3881212  bytes 387879152 (369.9 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 3881212  bytes 387879152 (369.9 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

ppp0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST>  mtu 1500
        inet 10.62.44.158  netmask 255.255.255.255  destination 10.64.64.64
        ppp  txqueuelen 3  (Point-to-Point Protocol)
        RX packets 1305  bytes 166845 (162.9 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1945  bytes 308252 (301.0 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

tun0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST>  mtu 1500
        inet 10.1.1.22  netmask 255.255.255.255  destination 10.1.1.21
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen 100
(UNSPEC)
        RX packets 1210  bytes 75098 (73.3 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1848  bytes 186415 (182.0 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figure 4-23 - Virtual Private Network Interface Information

The VPN encrypts links between devices using a certificate system. The certificates are stored on both the VPN server and VPN client devices and are used for authentication. The OpenVPN configuration file used on a STCD unit deployed in the field is shown in the block below. The configuration file contains the certificate information that the client requires to connect to the OpenVPN server.

```
client

dev tun
proto udp
remote 146.232.129.232 1194

resolv-retry infinite
nobind

persist-key
persist-tun
```

Stellenbosch University https://scholar.sun.ac.za


```
persist-remote-ip
verb 3

keepalive 10 900
comp-lzo
<ca>
-----BEGIN CERTIFICATE-----
MIIDRDCCAiygAwIBAgIJANrm2tQLKvNZMA0GCSqGSIb3DQEBCwUAMBsxGTAXBgNV
BAMTEHBlZ3Zwbi5zdW4uYWMuemEwHhcNMTQwMjEyMTIxNTE4WhcNMjQwMjEwMTIx
NTE4WjAbMRkwFwYDVQQDExBwZWd2cG4uc3VuLmFjLnphMIIBIjANBgkqhkiG9w0B
AQEFAAOCAQ8AMIIBCgKCAQEAvHhl6LkBdkGGpE+2PiFnmsAK1gxtVGXA3MRSyw/i
G31fbfl6mZ714mz53nON+y0UodawZKfyJ+RCHYpy8eEtsczFzhM35roWB81d7Qhk
e8/LOw1sukirIc/7Jub1rwdogaFdEEBi+p3RDLE6urDMByVuRSYJUVXwPa1ITRQQ
Ei3ANvwRkuNNJ/T+bOE+K+U/BBvaG0+/MT1YGJNQ6NXBMelB8j/sVDcDpt81qpFV
OWGsxeGN/JDSX+F9wy5mFK20zVM7iEIAPsJdKn9Houg+2SSOPwlEK+Yo7+cEf6bb
wQqp+YHEZ+3hf6X1uJmJhUKEuoNjjaQp9W6UW5/8NKUz7QIDAQABo4GKMIGHMB0G
A1UdDgQWBBSPSrf7Av+rEv2WAl36yymDjWEnFzBLBgNVHSMERDBCgBSPSrf7Av+r
Ev2WAl36yymDjWEnF6EfpB0wGzEZMBcGA1UEAxMQcGVndnBuLnN1bi5hYy56YYIJ
ANrm2tQLKvNZMAwGA1UdEwQFMAMBAf8wCwYDVR0PBAQDAgEGMA0GCSqGSIb3DQEB
CwUAA4IBAQCDWRvI18171kIag5JWvQhxhEYRVN6ZIlgHYIu97LT6o/qTcseSwdfS
PFIP9uxRJ6ZiKXdkgnZenyeW0+r8J6tRDf1/HLoBSF3fj8YE9npnbrWPmZGQOSAI
FveanVUG5RCN6aSn3RQO8EirDXj8zIoNeGpY24f7zDHOn2dug2saAP01GzYU1meD
S114umxI7KW1H1KIC0GaHnZpBnvYrF06ByzIInBttQ+Xow1wjPv6Ep6Ec9r/oKRG
cT3aAu23tbE1SDafZV//oHc7q4EVUMHed9QMZX0bvslDakoufyMFJAvFFxOheVkn
5MO1lZeerMT+13a4YOWAS9onMsgAdX4U
-----END CERTIFICATE-----
</ca>
<cert>
-----BEGIN CERTIFICATE-----
MIIDRTCCAi2gAwIBAgIBCDANBgkqhkiG9w0BAQsFADAbMRkwFwYDVQQDExBwZWd2
cG4uc3VuLmFjLnphMB4XDTE0MDIxNzE0MTkyMFoXDTI0MDIxNTE0MTkyMFowEjEQ
MA4GA1UEAxMHYmVhZ2xlNDCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEB
AKb+YRuXmayVduxOLhidoi3REqHN6hsh5TXGI4xnFn+WY32PCY3PXA8SVMOwJ51V
BuXQQCbfXUAJSejH3v+BZbZbbu/nOa16sWAPdW1TGbgmRHD638Gu1IUTZtQJInTf
v3zf0TPhdnW90OSGTJVzlZf/rLz5K80csOPu21D9DBvbupyvaBgn/fzq1T24ijmb
WEs9sErXwkpGF6QIj0xg6NzyQhHxMq8fq3IPwGGh00V1b8wkJTfp+8IzydCVE5Sj
LMIrfQZkr/4WZcQU08NpPK9U/sF2RA3wOlRbIF0P3xHU/wQBAjAxXI9X+KEDIMQy
q2y8DFGM4cD9/1hWIzqw25ECAwEAAaOBnDCBmTAJBgNVHRMEAjAAMB0GA1UdDgQW
BBQyRLN4UeZQF/l58w/WfrLqV/IlODBLBgNVHSMERDBCgBSPSrf7Av+rEv2WAl36
yymDjWEnF6EfpB0wGzEZMBcGA1UEAxMQcGVndnBuLnN1bi5hYy56YYIJANrm2tQL
KvNZMBMGA1UdJQQMMAoGCCsGAQUFBwMCMAsGA1UdDwQEAwIHgDANBgkqhkiG9w0B
AQsFAAOCAQEAAJ+7qzWflS7s8n+HqOxCFP0hJtPqVGGKfSMev1K/t/dBVWtXruTF
ch1osBBTnqMbOcz5lDRgHtYsbSYIUxTPxKMjMkTmqRSucWfV55q6t2mH8xcTLM7p
T/jdCHgeFnnzWRrGhBTwMwFqphgdKUNcSKxuJtGVU64hSQr1nSnrILtUVmi/7BL1
tVisZ/XqykDnK13UJtEwVeXiHjOMMeLS/yJcDauaRo5qPEgmeMJ/6oATt5ef3sbQ
Nk9cEbtbZcmLSRU2VPHNdKd3ROo/CeZA92k6kmBV8rbRRfhTgXbl8PhkhE2V+bIQ
CnfqUsbuABn+68MUGe4qzwrO0YIoJ0Pzbw==
-----END CERTIFICATE-----
</cert>
<key>
-----BEGIN RSA PRIVATE KEY-----
MIIEogIBAAKCAQEApv5hG5eZrJV27E4uGJ2iLdESoc3qGyHlNcYjjGcWf5ZjfY8J
jc9cDxJUw7AnnVUG5dBAJt9dQAlJ6Mfe/4Fltltu7+c5rXqxYA91bVMZuCZEcPrf
wa7UhRNm1AkidN+/fN/RM+F2db3Q5IZMlXOVl/+svPkrzRyw4+7bUP0MG9u6nK9o
GCf9/OrVPbiKOZtYSz2wStfCSkYXpAiPTGDo3PJCEfEyrx+rcg/AYaHTRXVvzCQl
N+n7wjPJ0JUTlKMswit9BmSv/hZlxBTTw2k8r1T+wXZEDfA6VFsgXQ/fEdT/BAEC
MDFcj1f4oQMgxDKrbLwMUYzhwP3/WFYjOrDbkQIDAQABAoIBACz73UDHh7f70zs4
nCY5YFSjOzem9mc0rD/eDflmU6Mu6cIK4/H1Z8EPmRHVw/YXz3HctUtDZCU521T/
cky1mKaU3r91NA5NPLa5s1ItzoHSnSEqE1hjNhBa95oqk8OT+kszJ3GoTHbPGRRW
MQ8ZQbTeRLPrEyd/kQ71atLYN4jRKleTCOxVuJ8SAnwFYrfrWhhpx1b8yBWRHyk4
hp4V9PSEKI5/YB4x+lUqVds42m/CqDIeDCj5O0/2x1LjHL4c2FbOwTA4o/JZhNMr
G3432rm3k4mAoXx9JpPiY25RWtJzzOI2Hl2VO+vwWtV/HsI7A2WqxEQO8kj7+8yM
ijf9tAECgYEA2i99iPdiXi+FCYVyWJ9+/6BYgveJMtDI4+l8CVr0mlaF188TPJ5U
```

Stellenbosch University https://scholar.sun.ac.za


```
+XQio5AbMh7HPkAI5xOpCrNahn40P8ueKqw/N+q2gdisdBwONmIsKU+WdD2UAiSb
4UwkMa5sU+UbAl3gKuCDEBoulYNeCLwGwao7qEEWYqra9pPk/SjZGXECgYEAw++Y
tIiIEXp409cEo4CszgNEEy7+pQSYBK6/6r6wvHbrmhrYabPW1uUP15F6tvOfBrNB
Z76cWf+mfa2y7+Y7ANCI0olPZkeOjm5Is1gVW71x9TKWx7XIqLbhEUzcqTHK9nE+
RQelWCDgyL2exIt3cIWYWPHRu0Q3jmaHYc5T1CECgYBesvAcq6sBUETA7i/EMW3w
y4q/RWtCKAaZjXStptG1z0oGcjVaQSqGnxaNwXVcWqmF5ks26HydTV1ENLC+48TB
psNFpNUUJJQtTSQKDLA4OfgGedes2nDh56dp/Ne9zhnb/BzY31tjeMmxUTRWPC3+
jsGX9LAfoSKqlWR1UO5xsQKBgEsgzrPCejIBfNz1YED1MM1ZcTvN1DQf+84n10zC
S3AoRRR04LA/FcyWTinBDYWjCkH+b8DA1KCppwQfmzHjXJJmY2Ae6EP2ghHY9aVW
y/G+RTjYafovmL4gSVJh32Y8wm8WYuMDGIIfMTGni34fX+/UOeM47ctZT9gZNiXp
gDkhAoGAcJnDg9pJdWUlWjt1o8n5omxmsuqK13hcti9hm7nSCX2QODvj+TgOkdBK
HUE1bPatjtun7iYFcpspY/eYPzvLAxKNdagRCTebWuhXIgUSRqFsbYZ+e8U24//5
tCP7fWUAo0poTyJuSLsEeSdREkpzWpn/Un2k8Zw+jNPBfi4n8zk=
-----END RSA PRIVATE KEY-----
</key>
key-direction 1
<tls-auth>
-----BEGIN OpenVPN Static key V1-----
7e2b91c0d899d0380d16e84ad6541114
0d2dda8c5e60542016905fbfa035fc2c
96008b2e518fa4d39c91ca92142e6654
3b7e4aacb65745b37f3fa72b16db5a1f
2165ba1124b5f97df25b062ad6348223
18cbdf162292e98ff3f3aeca2dcdad7e
34430f19e5192e99ecd85b1bd0a16e89
e492dd6bbdea6437bc4f7392ed6d75bf
3029d8ad3db63f39e6b2e5a33251e49a
9d4040e2cf7c178896e15de6fe718c33
48092494f2b2b61e18b9891a29512d66
a66737d39c1d00c4421df8523aa90938
a4066e1bf5fab8e92bb25554de5c6fe1
24764c350cdd6b694a0768a39204b705
5ae33c8fc50ca868ff30f0a966cc836c
2a74ceb0cedc6530e005cd81e418b000
-----END OpenVPN Static key V1-----
</tls-auth>
```

Stellenbosch University https://scholar.sun.ac.za

CHAPTER 5
THE ST AS A MARKET ENABLER

5.1 INTRODUCTION

This chapter contains the method and simulation results of implementing the Smart Transformer (ST) as a market-enabling device for a microgrid system.

The microgrid as discussed in section 2.4 contains generation sources, loads, and energy storage. The microgrid has a single point of connection to the traditional utility grid (macrogrid) and is seen from the macrogrid as a single, controllable entity.

The microgrid can function both when connected to the macrogrid, and when disconnected from the macrogrid. When disconnected from the macrogrid, the microgrid enters a state known as being *islanded* from the macrogrid.

When connected to the macrogrid, electricity can enter the microgrid from the macrogrid and exit the microgrid to the macrogrid. This happens at the single point of coupling between the microgrid and the macrogrid.

Distributed generation (DG) present within the microgrid generates electricity for use within the immediate microgrid and possible use outside the microgrid. These DG sources may be privately owned. A need arises for cooperation between such sources and the utility grid to effectively meet demand. A locally managed electricity market within each microgrid is suggested as an effective way of creating cooperation between utility companies and DG sources in selling electricity.

In this project, the ST is suggested as the devices placed at the point of coupling between the microgrid and the macrogrid. The ST manages connection and disconnection of the microgrid to the macrogrid, monitors the flow of electricity into and out of the microgrid, and acts as the market-enabler for the microgrid.

5.2 MUTLI-AGENT MICROGRIDS

The concept of the multi-agent microgrid was explored in section 2.4. The microgrid contains DG sources, loads and electricity storage equipment. In this project, a ST is suggested as the device to be deployed at the point of coupling between the microgrid and the macrogrid.

An effective communication layer is suggested where agents are deployed at the various hardware within the microgrid. The agents are intelligent devices that communicate and cooperate with one another to effectively manage operations within the microgrid.

5.2.1 Basic Operations of the Microgrid

From the macrogrid, the microgrid is seen as a single, controllable entity [9]. The microgrid can operate both when connected to the macrogrid, and when disconnected from it.

Stellenbosch University https://scholar.sun.ac.za

When connected to the macrogrid, electricity can enter or leave the microgrid through the point of connection (POC). This project suggests the ST to be located at this point. The ST records the flow of electricity into and out of the microgrid and controls connection and disconnection to / from the macrogrid.

When disconnected from the macrogrid, the microgrid operates in a state known as being *islanded*. When in islanded mode, stability and effective operation within the microgrid must be maintained without interference from the macrogrid, as it is disconnected from the macrogrid. Much research has been done on maintaining stability within the microgrid in an islanded state [69], [70], [71].

Within the microgrid, DG sources generate electricity, loads consume electricity and electricity storage can either store excess generation or deliver electricity to the microgrid when there is excess demand. Demand within the microgrid must be met with sufficient generation. When there is more demand in a microgrid than the generation in the microgrid can meet, electricity can either enter the microgrid from the macrogrid or storage within the microgrid can deliver the difference between generation and demand.

5.2.2 Interaction between Microgrid Agents

As described in section 2.4, various agents within the microgrid interact to allow the microgrid to manage operations. The main agents focussed on in this project are the generation agent, load agent, storage agent and point of coupling agent (POC agent). The roles of the various agents have been described in section 2.4.

The main interactions between generation and load agents within the microgrid are electricity transfers and the financial transactions that go with them. DG sources generate electricity and sell that electricity to loads in the microgrid. Excess generated electricity must go somewhere and can either be stored in storage units within the microgrid, or can be sold to the utility and leave the microgrid through the POC.

Stellenbosch University https://scholar.sun.ac.za

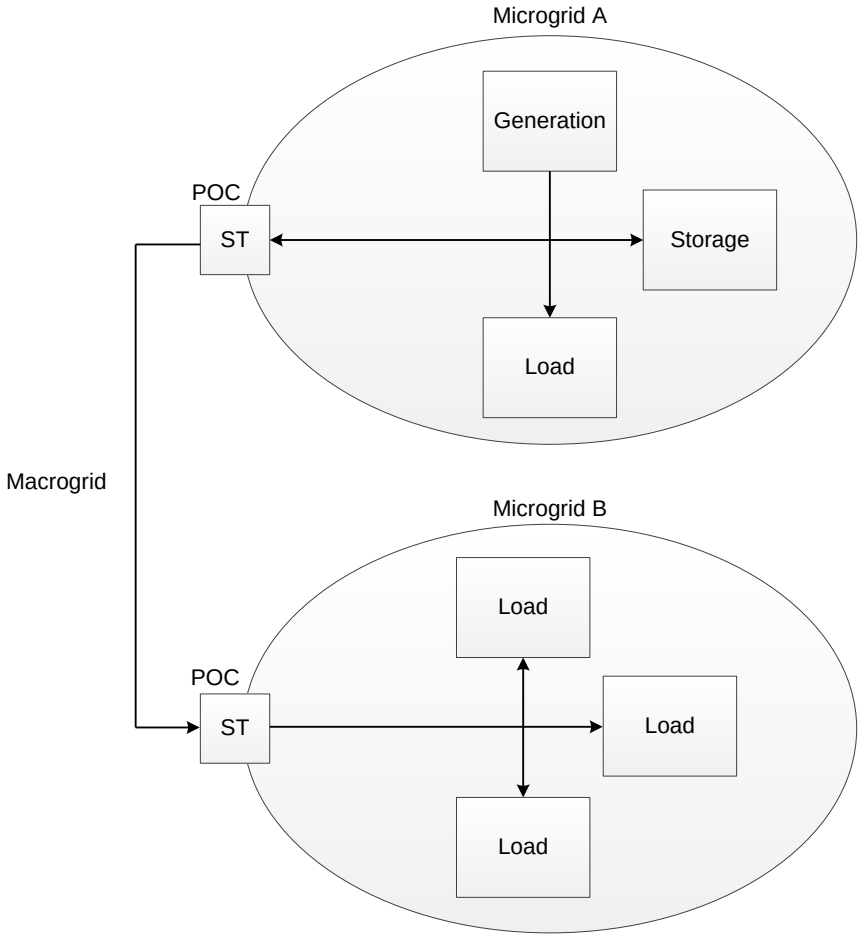


Figure 5-1 - Microgrid Agent Electricity Transfer Interaction

5.2.3 Microgrid Agent Communication and Web Services

Web services have been identified as a candidate tool for developing the information exchange and messaging protocol in a microgrid [72].

Web services are interfaces that allow the accessing of applications over a network. Applications can be hosted at any accessible location. Web services use standard internet technologies and protocols to access applications. A common example of a web service is an HTTP website. The website is hosted on a remote computer and can be accessed using a web browser.

Web services provide a platform-independent way of accessing applications. Thus, an application written entirely in Java may be accessed and utilized by a program written entirely in C++. It provides a common “language” by which applications on a network can access one another.

The goal of web services is the total modularization of the distributed computing environment. This means that essentially applications are built mainly from existing parts of other applications in the form of web services.

Web services separate functionality from platform. Thus, functionality can be developed and implemented on a variety of platforms. This is a benefit as platforms are rapidly advancing as new

Stellenbosch University https://scholar.sun.ac.za

technologies are developed, while desired functionalities remain constant for longer periods of time.

Web services were implemented in this project as the generic communication protocol between microgrid agents. Web services provide the ability to host various services on microgrid agents that make information available via a poll-response philosophy. In this project, web services were used to implement a microgrid market simulation. This is described in detail in section 5.5.

The gSOAP toolkit was used to implement web services in the microgrid market simulation.

For more on web services, please see APPENDIX E.

5.2.4 DG within the Microgrid

The main source of electricity provided to the grid is central, utility-owned generation. DG installed in microgrids will lessen the total demand that the utility needs to meet and will hold economic benefits for both buyers and sellers of electricity.

The introduction and increasing prevalence of DG in the grid has many advantages including [73]:

- DG is compact and flexible compared to central generators.
- Power loss reduction.
- Environmental protection.
- Reduces energy prices at retail level.

If DG is present in a microgrid, it can either supply electricity to the immediate microgrid that it is located in, or sell electricity back to the utility. Once electricity is sold to the utility and leaves the microgrid, the utility has freedom to transport the electricity to anywhere in the grid where there is demand (see Figure 5-1). DG contained in a microgrid may either be:

- Owned by small to medium-scale power generation companies.
- Privately owned.
- Owned by the utility.

Privately owned DG may be setup in a microgrid to meet demand for a few loads within the microgrid. Generated electricity can be used to meet owners' demand and the excess generated electricity can be sold to the utility. If the DG is not capable of meeting the owners' demand, then electricity is purchased from the utility to meet the demand.

Small to medium-scale generation companies may install larger DG sources to meet demand within the microgrid where they are located and to sell to the utility. These sources may need to install their own storage systems as required by the utility to maintain grid stability. The aim of such systems would be to ensure a reliable amount of electricity delivery by storing electricity in times of excess generation and releasing that electricity to meet delivery agreements made with the utility. This is beyond the scope of this project and will not be discussed.

Stellenbosch University https://scholar.sun.ac.za

DG sources contained within microgrids are expected to consist largely of renewable sources [74]. These sources can have unpredictable generation patterns and thus necessitates the effective management of generation and distribution within the microgrid. A management system is required to effective manage the buying and selling of electricity within the microgrid, and to facilitate interaction between generation sources and loads.

Such management systems are known as distributed energy resource managements systems (DERMs). DERMs aim to achieve a range of functionalities not limited to market implementation. Some of these functionalities include [75]:

- Fault location
- Data analytics
- Distributed intelligence
- Trading platform implementation
- DG management
- Distributed energy resource (DER) forecast
- Distributed network modelling
- Demand-side management (DSM)

The ST will function as a DERM, and has the potential to fulfil the requirements. The utility will need to specify exactly what requirements the ST will fulfil.

Storage should be utilized to minimize the effects of unpredictable generation by storing electricity in times of excess generation and delivering electricity in times of a lack thereof. This process must be managed effectively.

5.2.5 Voltage Stability within the Microgrid

When islanded from the macrogrid, a microgrid needs to maintain its own stability. Stability within the microgrid is maintained by ensuring that the microgrid line-voltage level remains between pre-set bounds. This is done by the effective collaboration of microgrid agents.

The microgrid line-voltage rises when there is an excess of generation within the microgrid [76]. When there is more generation than demand, the excess generation must be stored, and the generators should be controlled to generate less electricity. This continues until the microgrid line voltage once again becomes stable.

Conversely, when there is an excess of demand, electricity is delivered by storage components to meet demand and controllable loads are turned off to lessen demand. This continues until stability is restored. This process of maintaining voltage stability is continuous and happens automatically within the microgrid.

When connected to the macrogrid, the microgrid synchronizes with the macrogrid and electricity can either leave or enter the microgrid. The stability of the microgrid is then dependant on the

Stellenbosch University https://scholar.sun.ac.za

stability of the macrogrid. DG sources within the microgrid can be employed by the utility to generate electricity when there is a lack of generation in the macrogrid.

Microgrids can be disconnected from the macrogrid to restore stability of the macrogrid. In such a case, the microgrid will utilize sources, loads and storage within itself to maintain stability until reconnected to the macrogrid.

5.2.6 Smart Loads

Demand-response is an initiative aimed at increasing customer participation in decision making concerning electricity usage.

Demand-response aims to give customers the option of using electricity in non-peak times with an economic benefit to them. This is implemented by creating smart user-interface systems that interact with both the customer and the power utility. From the utility, these systems gather information on grid status and electricity pricing. From the customer, the systems gather user preferences and decisions concerning electricity usage [77]. The systems then execute the path of action that best suits customer and utility.

Some of the benefits of demand-response are [78]:

- Peak reduction
- Economic benefits for customers
- Greater grid stability

One example of such a system would be a smart washing machine. A user can specify a certain price at which he or she would like to purchase electricity for washing their clothes. The smart washing machine monitors the electricity prices in the microgrid and begins to wash the load when an acceptable price is available or after a user-defined time period.

It must be noted that not all loads can function in such a manner. Different loads carry varying levels of priority. For example, it is more important for a respirator to turn on immediately than it is for a washing machine. High-priority loads must receive immediate electricity and thus current electricity prices are irrelevant for the operation of such loads.

5.3 MICROGRID MARKETS

As privately owned DG becomes more prevalent within the grid, a need arises for a market structure to facilitate transactions between electricity generators and consumers. For the market to be effective there must be an economic benefit for both electricity sellers and buyers.

In the smart grid (SG) vision endorsed in this project, DG sources and customer loads are contained within microgrids. These microgrids make up the utility grid as a whole. DG sources can sell electricity to customers within the immediate microgrid that it is contained in, and to customers in other microgrids via the utility.

Stellenbosch University https://scholar.sun.ac.za

In this section, a market structure is introduced that effectively handles electricity transactions between the utility, DG sources and customers.

5.3.1 Assumptions

Some assumptions are made in the introduction of the market:

- **Local DG sources within microgrids can generate and sell electricity at a lower tariff than the utility and still make a profit**. This assumption is based on continuous advancement in DG technologies. As technology advances, the price of generating electricity using DG decreases. Thus, DG is continuously becoming a more economical option [79].
- **Customer loads contain smart devices that can interact with the grid**. This is based on the concept of demand-response. Demand-response allows the customer to participate in the decision making process of acquiring energy for usage. The customer decides whether he / she would incur a time penalty to gain an economic benefit when using appliances. In other words, he decides either to wait or not to wait until electricity prices have reached a certain low before a load will switch on [77].
- **The Smart Transformer (located at the point of common coupling (PCC) with the macrogrid) can effectively communicate with all microgrid agents**. This concept was introduced in section 2.4 and expanded on in section 5.2.
- **Microgrids can smoothly connect and disconnect from the macrogrid at the PCC**. This has been identified as a requirement for microgrids [80].

5.3.2 Free versus Centralised Markets

In a free electricity market, market participants (generators and consumers) are free to transact as they please. There is no centralised market authority and participants interact according to electricity demand and availability. This approach takes the form of a peer-to-peer network where each participant functions according to their own goals [81].

In a centralised market, there is a central authority that functions as a governing body to some extent. Producers and consumers of electricity submit their demand curves to this centralized authority. They then bid in pre-set time intervals (E.G. for every 15 minutes of the next day) for generation and consumption schedules [81].

In this project, a free market that is regulated by the utility is suggested. Free market principles such as asynchronous DG and customer interactions are suggested, with customer interactions being regulated by the utility. This ensures economic benefits to both customer and generator and contributes towards grid stability by regulation of generation and demand.

In the South African context, utility regulated markets are essential, as Eskom owns most power infrastructure. Eskom is the sole power utility in South Africa and works in cooperation with the

Stellenbosch University https://scholar.sun.ac.za

South African government. For a market implementation in South Africa to work, cooperation with Eskom is necessary. Third-party power generators would enter the market as not only generators, but also clients of Eskom, as Eskom regulates their operations.

The utility sets electricity tariffs at which it will sell electricity to customers and buy electricity from generators. These tariffs can be set in real-time, hourly, daily, or are fixed. DG sources must generate and sell electricity at a price that is better than that of the utility while still making a profit for customers and generators to gain an economic benefit. Tariffs are applied within each microgrid. DG sources within the microgrid can sell electricity to customers in the same microgrid, or can sell to the utility. Electricity sold to the utility can leave the microgrid and then be resold by the utility at the utility tariff in another microgrid.

A single market-enabling device within the microgrid manages the market. Every microgrid agent within the microgrid plays a unique role in contributing towards the effective functioning of the market [80]. This is described in detail in the next section.

5.4 THE ROLE OF THE ST IN THE MICROGRID

In a controlled microgrid, it has been suggested that a management device be implemented to oversee microgrid operations and perform central management [80], [82].

In this project, the ST is suggested as the management entity within the microgrid and the main market-enabling device. The ST is located at the PCC, giving it the ability to control connection and disconnection of the microgrid to the macrogrid and the ability to monitor all electricity exchange between the microgrid and the macrogrid and within the microgrid itself.

5.4.1 Distributed Intelligence

As the utility grid moves towards becoming a smart grid, both utility-side and customer-side devices are being released with greater computational capabilities. These capabilities have potential to distribute the computational load of utility network data handling across the grid.

The ST is no exception to this. In this project, an ARM-based micro-processing device was installed on the ST. This device contains processing capabilities that exceed the needs for basic monitoring and control of hardware. In this project, it was described how this intelligence was utilized to implement monitoring, control and communication functionality in the ST. In this section, a description is given to how this intelligence is utilized to make the ST a market-enabling device. In the next sections, the methodology to simulate the scenario is given.

5.4.2 ST Functionality

The ST functions as the manager and market-enabler in the microgrid.

Central management and control of a microgrid is not a new concept [80], [82], [83], [84]. The main objectives of a central controller in a microgrid include [83]:

Stellenbosch University https://scholar.sun.ac.za

- Providing power set points for DG units
- Economic scheduling
- Overseeing demand-side bidding for electricity
- Controlling of peak load to reduce peak load
- Control of non-critical loads during microgrid islanding
- Detecting islanding conditions determined by the measurements taken at the PCC
- To minimize system losses
- To initialize resynchronization between microgrid and macrogrid when the microgrid is reconnected to the macrogrid
- To monitor power flows from local DG sources and at the PCC

The microgrid has a good communication network that allows the ST to communicate with all microgrid agents including DG agents, load agents and storage agents. The ST can monitor all microgrid agents and send command messages to them.

DG agents report their generation statistics and predictions to the ST as well as tariffs at which they are selling electricity. Loads report their usage to the ST and can make bids for electricity to be used by non-critical loads. Storage units report their charge status and available capacity to the ST. The ST itself measures the total amount of electricity that enters or leaves the microgrid at the PCC.

The ST takes all of these sources of information into account and exercises control to maintain the market within the microgrid. When the microgrid is connected to the macrogrid, electricity can enter the microgrid when there is an excess in demand and leave the microgrid, or enter storage within the microgrid, when there is an excess in generation. The amount of electricity entering the microgrid indicates the amount purchased from the utility, while the amount of electricity leaving the microgrid indicates the amount that the utility purchases from DG sources to be used elsewhere in the grid.

The ST compares information from generating sources and loads within the microgrid to see how much each load is consuming and each source is generating. If there is more demand than generation, electricity must be purchased from the utility, while electricity can be sold to the utility when there is more generation than demand.

The ST takes bids from loads and communicates them to sources. Sources then decide whether they are willing to meet the bids and either accept or reject them. When a bid is accepted, the source generates the required electricity and the load consumes it. The ST facilitates these transactions and keeps track of generation of sources and usage of loads. The ST also keeps track of the monetary amounts that:

- Loads owe to the DG sources
- Loads owe to the utility

- The utility owes to DG sources for electricity purchased
- DG sources owe the utility for use of its transmission lines (postage fees)
- DG sources owe to the utility for instances where they cannot meet the demand that they indicated they would (with an incurred penalty). The penalty is present because the unmet demand can be seen as a breach of contract between the utility and the DG source.

The ST draws up a profile for each load and each source in the microgrid. These profiles are stored locally on the ST. The information profiles for generators include the following information:

- Production transaction records
- Reputation score
- Current generation tariffs
- Production predictions
- Monetary amount owed by customers
- Monetary amount owed by utility
- Monetary amount owed to utility

The information profile for loads include the following information:

- Usage transaction records
- Critical load usage
- Non-critical load usage
- Non-critical load electricity bids
- User preferences

5.4.3 Reputation Score

A generator reputation management system (RMS) is suggested in [85]. Such a system is intended to minimize misbehaviour of market participants.

A RMS is applicable on generation sources within a microgrid. The overall performance of the generation sources are considered to allocate a reputation score to each generator. The reputation score of the generator influences the perceived reliability of the generator. The perceived reliability in turn influences the generation tariffs that customers are willing to pay the generator for electricity and the generator's priority in receiving electricity usage bids from loads.

Various factors influence the reputation score of a generator. These include [85]:

- Outage times and duration
- Generation schedule violations
- Customer ratings
- Generated power quality

Stellenbosch University https://scholar.sun.ac.za

The ST keeps track of the reputation scores of all generators within a microgrid. If a generator is present with a low reputation score, a penalty is incurred upon the tariffs that the generator may set for electricity sales.

When a generator enters the microgrid market, it is assigned a starting reputation score which is high enough to be favourable for the generator. This ensures that generators can still compete within the microgrid market. When a generator misbehaves and does not act in a way that is expected of it, it loses reputation score.

A RMS must be monitored by a central governing authority. The ST acts as the implementer of RMS governing within a microgrid. In this project, a RMS was not included in the simulation due to time constraints.

5.4.4 Determining Monetary Compensation

The ST is responsible to determine and issue monetary compensation and billing for DG sources and customers within the microgrid.

The ST uses generation profiles, load profiles, POC measurements and electricity tariffs applicable in the microgrid to determine these amounts. The following scenario illustrates how these amounts are determined.

We consider a microgrid with one generator, one load, one storage unit and a ST at the POC as shown in Figure 5-2.

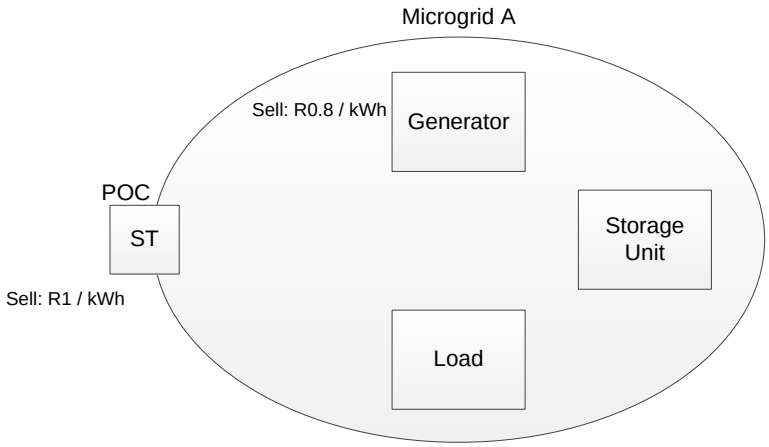


Figure 5-2 - Microgrid Market Setup

The utility is selling electricity at a rate of R1 per kilowatt-hour. The generator within the microgrid is selling electricity at a rate of 80 cents per kilowatt-hour. The load in the microgrid turns on a non-critical load that will consume 20 kW for a period of two hours and sends a bid to the ST to purchase electricity for this load.

Stellenbosch University https://scholar.sun.ac.za

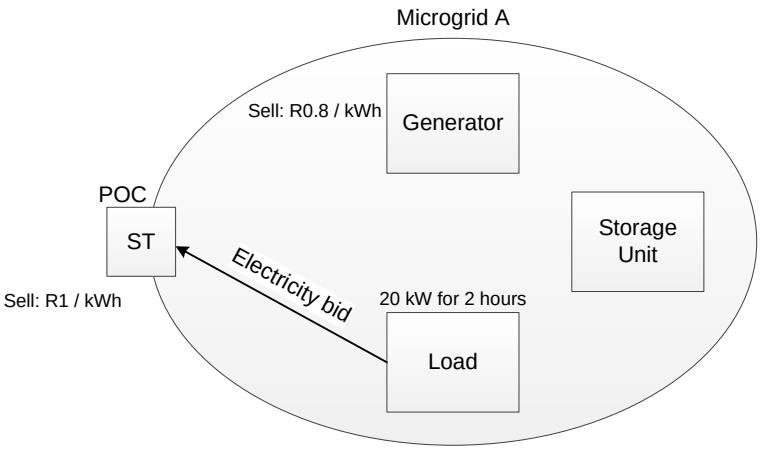


Figure 5-3 - Microgrid Load Bid

The ST receives the bid from the load and communicates the bid to the generator. The generator then considers the load and decides whether it will meet the bid or not. If it decides to meet the bid, the generator generates the required electricity and the load consumes. If it decides to reject the bid, the utility supplies the electricity.

In the scenario, the generator accepts the bid and generates a total of 25 kW for the first hour and 13 kW for the second hour. The variation in generation occurs as the generator is assumed to be a wind generator with unpredictable generation patterns. For the first hour, there is enough to meet the demand from the load and an additional 5 kW. One of three things can happen to this 5 kW. It can either be put into storage, be sold to the utility, or be used elsewhere in the microgrid. If put into storage the electricity can be utilized at a later time when there is a lack in generation to meet demand. If sold to the utility, the electricity leaves the microgrid and can be used as the utility pleases. If there is another active load within the microgrid, the 5 kW can be sold to that load.

The utility purchases electricity at the same price as customers within the microgrid. The electricity then leaves the microgrid via the POC and can enter another microgrid to meet demand. The utility sells the electricity in the second microgrid at utility tariffs. The profit the utility makes on the electricity should cover transportation costs.

The ST measures the total electricity entering and leaving the microgrid at the POC. In this way, a record is maintained of the total electricity purchased from the utility or purchased by the utility from DG sources. By taking the measurements at the POC and the various microgrid agents into account, the ST can determine monetary values to credit or debit generators and loads by.

The total amount of electricity purchased from the utility or by the utility is the difference between the total amount of electricity entering or leaving the microgrid. If a load is demanding 20 kW and generators within the microgrid can only deliver 15 kW, 5 kW will enter the microgrid to maintain stability. This 5 kW is purchased from the utility. Conversely, if a load is demanding 15 kW and generators within the microgrid generate 20 kW and sell it to the utility, the extra 5 kW will leave the microgrid or will have to enter storage.

Stellenbosch University https://scholar.sun.ac.za

In the scenario, the generator stores the excess 5 kW that it generated for the first hour.

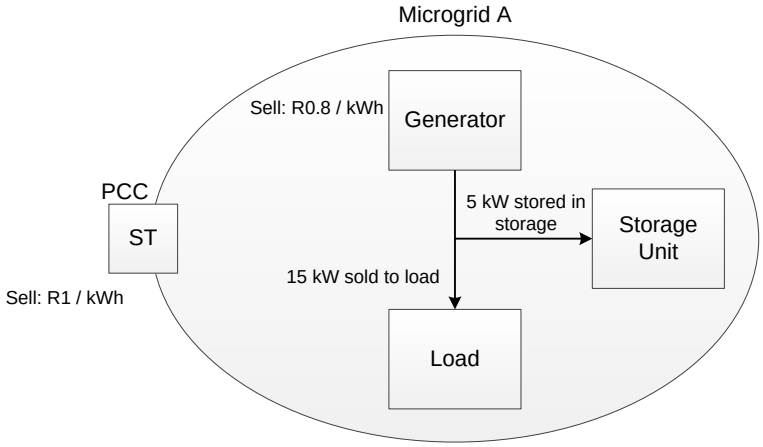


Figure 5-4 - Microgrid Operation for First Hour

For the second hour, the generator only generates 13 kW. This amount does not meet the demand of the load within the microgrid. The generator must either utilize electricity stored in the storage unit or risk not meeting the agreement with the load and receive a reputation score penalty. In the first hour, the storage system was charged by the generator. That charge can now be extracted from the storage unit to meet the excess demand.

5.4.5 Generation Timing

Customers may bid for electricity at any time. If local generators cannot immediately meet the demand, the utility will have to supply the difference in demand and generation within the microgrid to ensure stability. The generator must then still produce the agreed amount of electricity.

When a generator does not generate electricity on time, the utility may charge generators a cost as the electricity sold by the utility may have travelled a large distance to reach the microgrid. The travel cost for this electricity must be covered.

5.5 MARKET SIMULATION

A basic market simulation was performed to showcase the ability of the ST to fulfil the role of market-enabler.

For the market simulation, three agents were simulated: the ST, a generator agent and a load agent. The ST acts as the market manager within the microgrid. The load bids for electricity supplied to non-critical loads and the generator supplies these bids if possible.

The PI System was used to visualize interactions between the various components. The PI Rational Database Interface was used to gather data from the various simulated agents and store the data in the PI System.

Stellenbosch University https://scholar.sun.ac.za

This simulation is elementary and aims to showcase the ability of the ST to act as an intelligent market-enabling device and of web services as a communication protocol between microgrid agents.

5.5.1 Agent Configuration

Each agent was simulated using a separate microprocessor development board connected in an Ethernet mesh network.

Table 5-1 contains the hardware specification of each agent simulated.

Table 5-1 - Microgrid Agent Simulation Components [86] [87]

Agent/s	ST Agent	Generator and Load Agents
Development Board	Generators of bulk quantities of electricity	Power generation, asset management.
Development Board	BeagleBone Black	Raspberry PI Model B+
Microprocessor	1 GHz Texas Instruments Sitara AM3359 ARM Cortex A8	700 MHz ARM1176JZFS
Random Access Memory	512 MB DDR3L @ 400 MHz	512 MB SDRAM @ 400 MHz
Storage	2 GB on-board eMMC	Micro SD Card (8 GB)
Power Draw	210-460 mA @ 5V under varying conditions	650 mA @ 5V under varying conditions
GPIO Capability	65 pins	40 pins
Peripherals	1 USB host, 1 Mini-USB client, 1 10/100 Mbps Ethernet port, micro-HDMI port	4 USB hosts, 1 Micro-USB power connector, 1 10/100 Mbps Ethernet port, HDMI port

A Local Area Network (LAN) was used to establish a fast connection between all of the agents. Agents communicate by using web services over a TCP connection established via the LAN network. The agents were all connected to the larger Stellenbosch University network through which they communicate with one another. The network assigns a static IP address to each agent, acting as a domain name for web services.

Stellenbosch University https://scholar.sun.ac.za

The agents use web services to send and receive data to and from one another. An agent can contain both server and client web services. Server web services listen on certain network ports for incoming messages from other agents. Client web services construct and send messages to other agents. Web services were programmed in C++ using the gSOAP toolkit.

The MySQL database was used to store the configuration and operational data of the agents. The MySQL database was used as a communication medium between various C++ services running on the agents. Services can monitor the MySQL database for relevant changes and act accordingly when they occur. The functionality of the various services are described in detail in the following sections.

5.5.2 Software Configuration and Functionality

The simulation software configuration is illustrated in Figure 5-5.

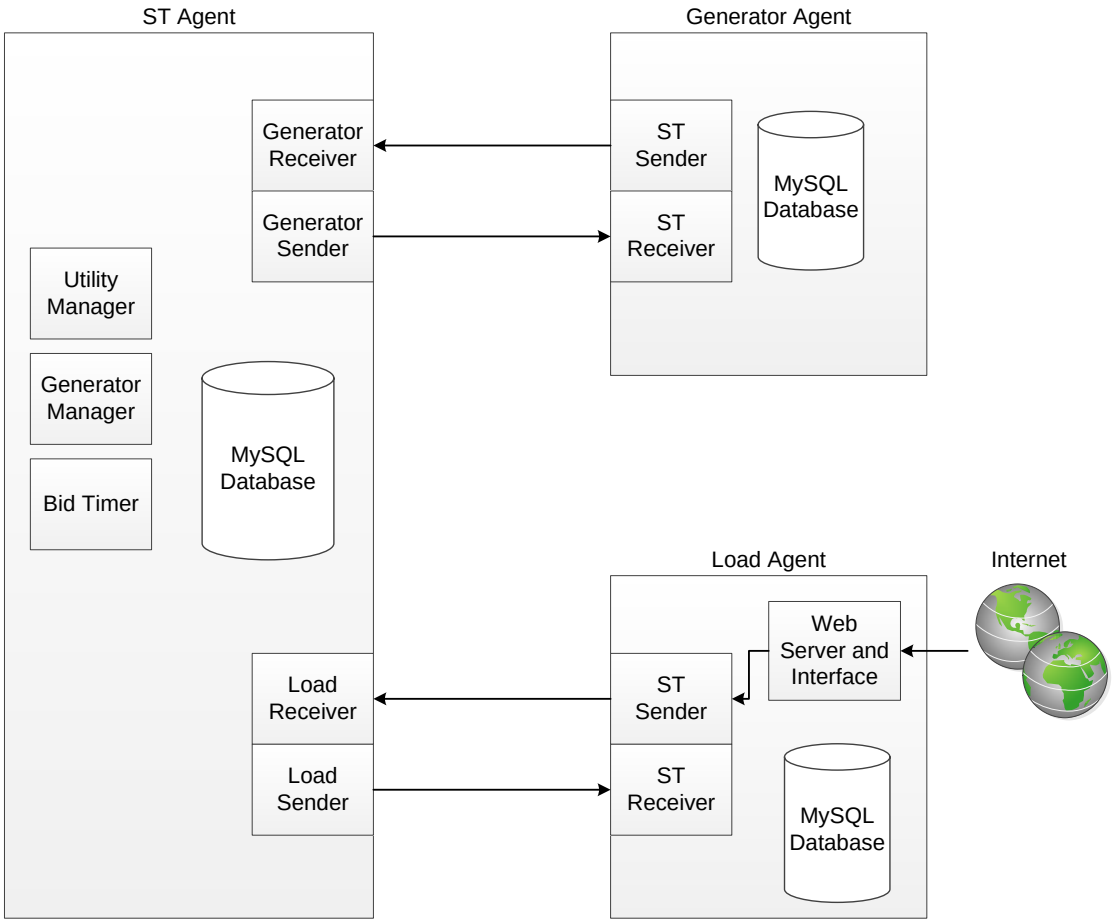


Figure 5-5 - Market Simulation Software Configuration

Various software components contained in the agents work together to create effective interactions between them. A service-based philosophy was adhered to in the implementation of the market simulations. This means that instead of writing single, large programs for each agent, various programs were written for each agent as services. Each service performs a specific task to facilitate effective interaction between agents. The functionality of each agent and a description of the services present on them will now be given.

Stellenbosch University https://scholar.sun.ac.za

Load Agent

The load agent represents any smart load in a microgrid.

The load agent manages loads found in a load collective such as a house or a building. It is assumed that in a Smart Grid with smart loads, loads will possess the ability to communicate bids to the load agent [88]. For example, when a customer switches on a washing machine, the washing machine communicates its energy requirements via the load agent to a central system. The energy requirements are then met by some source.

In the simulation, the load agent communicates bid information to the ST, which in turn communicates it to loads to find a suitable electricity source. Energy bids in the simulation have the following variables:

- A transaction / bid number to identify the bid.
- A timestamp assigned when the bid is issued.
- The rated watts required by the load.
- The timespan for which the supply of electricity is to continue.
- The maximum tariff the customer is willing to pay to a third party generator.
- A maximum time (timeout time) that the customer is willing to wait before the load switches on. This timeout is only applicable on local third-party generators. If the timeout expires, the utility will meet the bid.
- A flag to indicate whether the customer is looking for a local source to meet the bid, or if the utility should meet the bid.

Load bids are entered manually in the simulation via a web interface hosted on the load agent. The user enters the various bid parameters and submits the bid. The bid is then written to the MySQL database on the load agent as an entry in a table that contains all bids. Once written, a web service client program detects the new entry and sends the entry to the ST in a web service message. The process is shown in Figure 5-6.

Stellenbosch University https://scholar.sun.ac.za

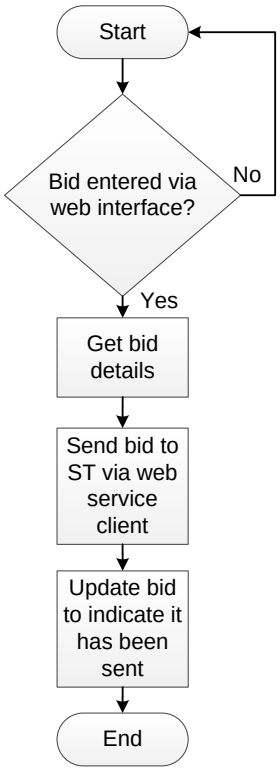


Figure 5-6 - Load Agent Bid Management

Another service on the load agent acts as a web service server or message receiver. Once the bid has been processed and a suitable source has been found, the ST sends a message to the load indicating what source will meet the bid and what tariff is electricity is to be purchased at. The ST will either indicate that the bid has been met by a private generation source or by the utility.

ST Agent

The ST agent is the market manager within the microgrid.

The ST agent interfaces with both loads and generators to manage the bidding process in the microgrid. A service on the ST agent listens for incoming messages from microgrid loads containing electricity bids. Once a message is received, the bid information is stored in the ST MySQL database.

Three programs monitor the ST MySQL database for incoming bids. One program named *Utility Manager* monitors for incoming bids with the utility flag (indicating the bid is to be met by the utility) set high. Another program named *Generator Manager* monitors the database for incoming bids with the utility flag set low. Finally, a program named *Bid Timer* monitors incoming bids to ensure that their timeout time does not expire.

The Generator Manager program takes incoming bids with the utility flag set low and sends them to possible generators. Generators respond with either a tariff price at which it can meet the bid, or an indication that it cannot meet the bid. If a generator can meet the bid, the ST stores the transaction information and communicates to the relevant load that it may proceed with electricity

Stellenbosch University https://scholar.sun.ac.za

usage and includes the tariff that the generator will charge. If no generators can meet the bid, the bid's utility flag is set high and the Utility Manager program detects the bid.

The Utility Manager program monitors the ST MySQL database for bids with the utility flag set high. When a bid is detected, the program fetches the correct utility tariff from another MySQL table and sends a web service message to the relevant load with the applicable tariff and an indication that it may proceed with electricity usage for the bid. The utility tariffs contained within the utility tariff table on the ST database are time-of-day based and can be updated by the utility remotely. This model was chosen for this simulation, but any tariff system may be applied. The tariffs issued for the simulation are found in Table 5-2.

Table 5-2 - Simulation Utility Tariffs

Time of Day	Tariff	Block Number
00:00:00 - 07:30:00	R1.23 / kWh	1
07:30:00 – 09:30:00	R1.45 / kWh	2
09:30:00 – 17:30:00	R1.25 / kWh	3
17:30:00 – 20:00:00	R1.51 / kWh	4
20:00:00 – 00:00:00	R1.23 / kWh	5

The Utility Manager program determines the current time of day and fetches the applicable utility tariff using a MySQL instruction.

The Bid Timer program monitors currently active bids to see whether their timeout time has expired. It takes the current time and compares it to the timestamp on the bid to determine the time elapsed since the bid has been issued. If a timeout has occurred, the utility flag for the bid is set high and the bid is met by the utility.

Stellenbosch University https://scholar.sun.ac.za

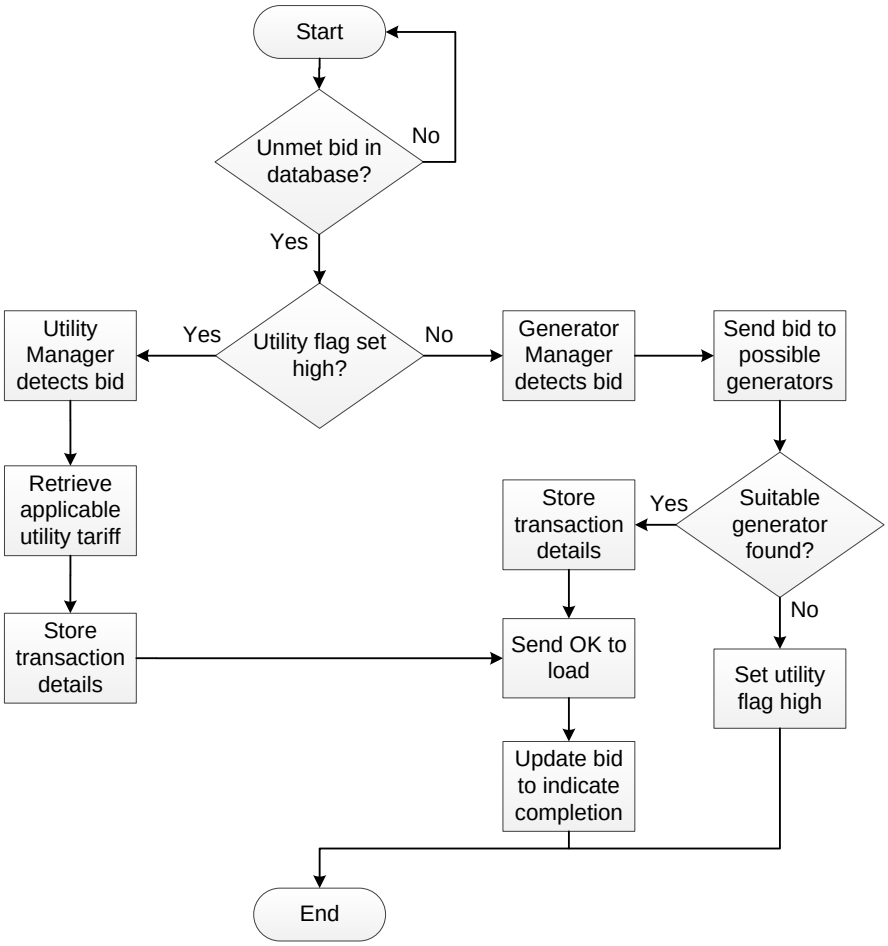


Figure 5-7 - ST Bid Management

The ST agent has a separate MySQL table for each load and each generator. Thus, full records of all transactions between loads and generators are maintained. The ST also has a table for transactions concerning the utility, so that utility transactions are also recorded.

Generator Agent

The generator agent has a web service server that listens for incoming bids from the ST. There is a MySQL table on the generator agent that contains the variables determining whether a generator can meet a specific bid including:

- The amount of watts the generator can supply.
- The amount of time the generator can supply the watts for.
- The current asking tariff of the generator.
- The amount of leeway that the generator is willing to give on the tariff.

When a bid is received from the ST agent, the generator agent analyses the bid and compares it to the values stored in the table described above to see if it can meet the bid. If it can meet the bid, it returns the tariff that it will generate electricity at to the ST agent. If it cannot meet the bid, it returns a zero to the ST agent indicating that it cannot meet the bid. The generator agent has a MySQL table for bids that it has accepted, and a MySQL table for bids that it has rejected.

Stellenbosch University https://scholar.sun.ac.za

A check is also done to ensure that the bid timeout has not yet been reached. If the timeout is reached, the generator does not attempt to meet the bid. The generator then discards the bid.

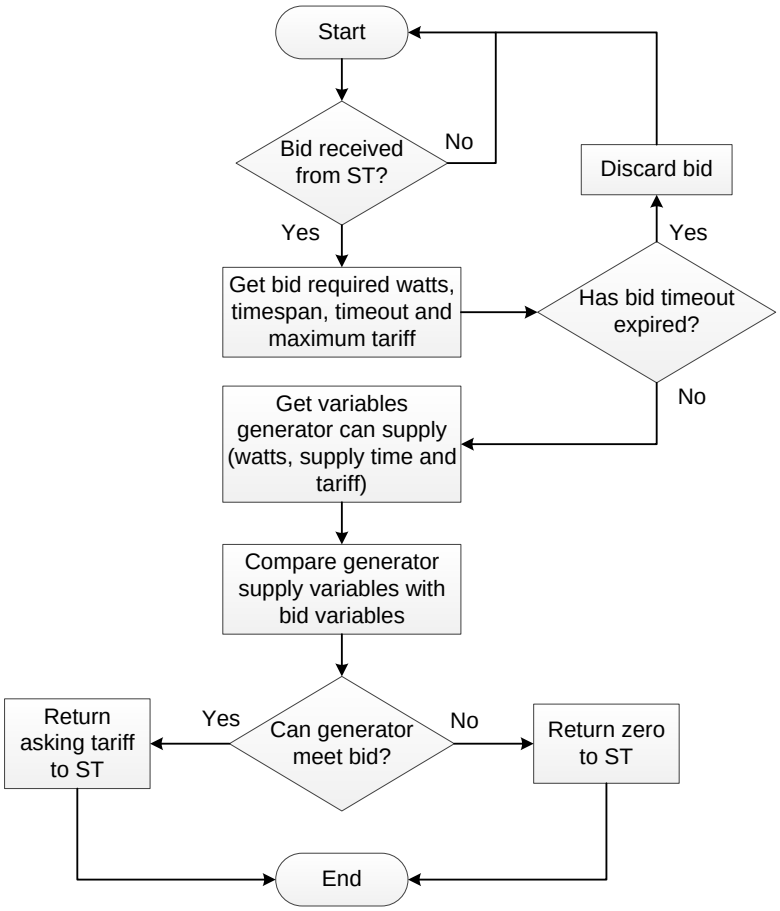


Figure 5-8 - Generator Bid Management

5.5.3 Reading Simulation Data into the PI Server

The PI System was used to visualize the microgrid simulation. To read data from the various agents into the PI System, the PI Relational Database Management System (RDBMS) interface was used.

Each microgrid agent simulated stores data in a MySQL database running on that agent. This database is the point of connection between the various agents and the PI System. The Windows Data Source (ODBC) tool, together with the MySQL ODBC connector is used to add external MySQL databases as data sources to the PI Server computer. This gives the server access to data within the MySQL database on remote devices.

The PI RDBMS interface utilizes these data source connections to extract data from MySQL databases on remote devices and store it within the PI System. For a device to allow an external server to access its MySQL database as a data source, the following steps must be followed (an overview is given here, see APPENDIX F for more details):

- The MySQL configuration on the device must be altered to bind the MySQL server to the device IP address.

- Network port 3306 must be opened to the connecting server as MySQL listens on this port.
- A user on the connecting server must be granted access to data within the MySQL database.

PI tags in the form of SQL queries can then be created for the RDBMS interface. For example, a tag can be specified to execute the following SQL query:

SELECT watts FROM loadTable WHERE id = 1;

This will give the PI data point the value of the *watts* column in the *loadTable* table where the *id* column is equal to one.

Three RDBMS interfaces were created for the ST agent, load agent and generator agent respectively. Various tags were created to get data from the MySQL databases on the various agents. SQL queries were also used to manipulate data. For example, the SQL COUNT function was used to calculate the total number of entries in a table on the ST, indicating the total number of bids that the ST has handled.

5.5.4 PI ProcessBook Simulation Screen

The screen created in PI ProcessBook to visualize the interfacing between the various microgrid agents is shown in Figure 5-9.

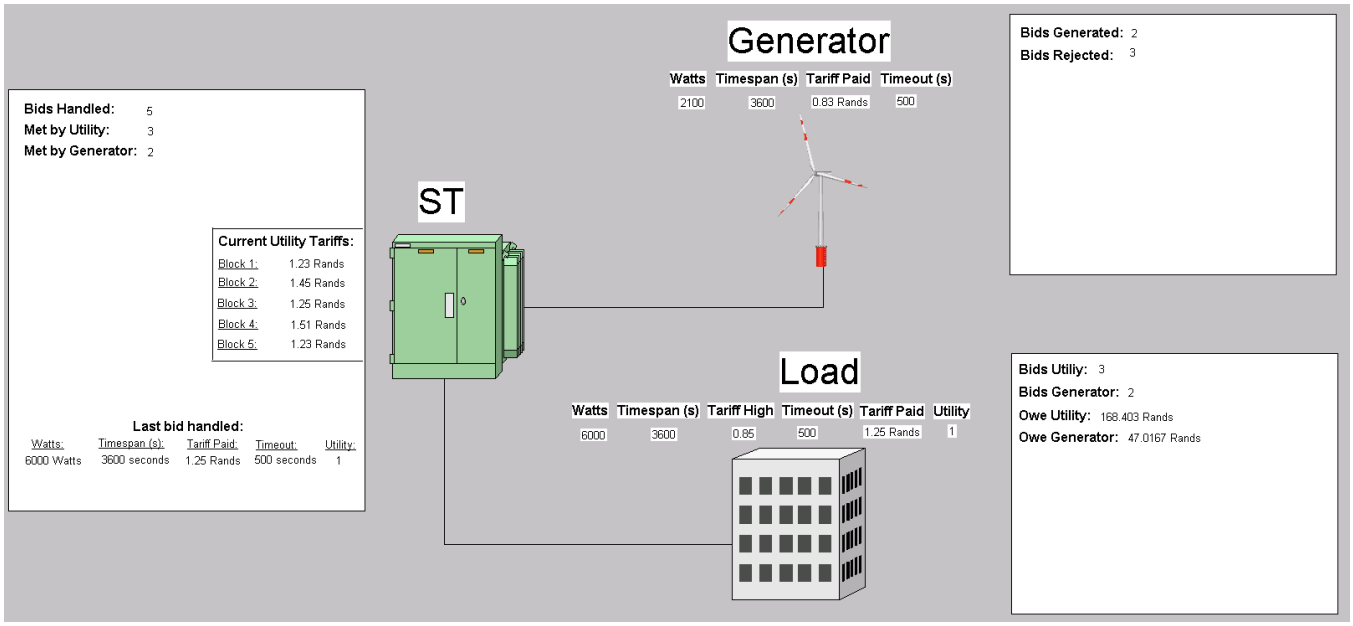


Figure 5-9 - Microgrid Market Simulation Screen

The screen contains information on the three agents simulated. Information on the last bid handled by each agent is displayed. Some statistics for each separate agent are also displayed on the screen. The screen updates in real-time as the values in the MySQL databases of the various agents changes.

The load agent displays information on the last bid it has issued. The information includes:

Stellenbosch University https://scholar.sun.ac.za

- Generation required by the bid in watts (*Watts*).
- Timespan for which the bid must continue in seconds (*Timespan*).
- The maximum tariff the load wishes to pay local generators in Rands / kWh (*Tariff High*).
- The time the load is willing to wait before the load switches on in seconds (*Timeout*).
- The tariff paid by the load to complete the bid (*Tariff Paid*). Note, this is issued only once a suitable generator has been found and the relevant tariff has been communicated to the load.
- Whether the bid was met by the local generator, or by the utility (*Utility*) (1 indicates the bid was met by the utility while 0 indicates it was met by a local generator).

Statistics for the load agent is displayed including:

- Total bids met by the utility
- Total bids met by the local generator
- Total amount owed to the utility
- Total amount owed to the local generator

The total amounts owed to the generator and the utility are calculated on the load agent using MySQL code.

The last bid handled by the ST is displayed with fields similar to those displayed at the load. Utility tariff information read from MySQL is displayed at the ST. Finally, statistics on bids handled by the ST are displayed.

At the generator, the last bid generated is displayed as well as statistics on how many bids the generator has either accepted or rejected.

5.5.5 Demonstration

A bid is entered manually into the load agent using the web interface displayed in Figure 5-10.

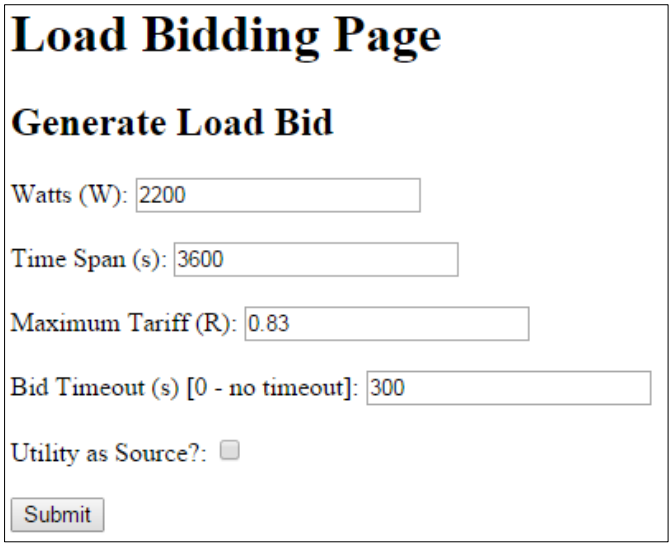


Figure 5-10 - Load Web Interface Bid Loader

Stellenbosch University https://scholar.sun.ac.za

Once the *Submit* button is pressed, the bid is stored in the load MySQL database using PHP. The PI Server detects the changes made to the MySQL database and displays information concerning the bid in ProcessBook.

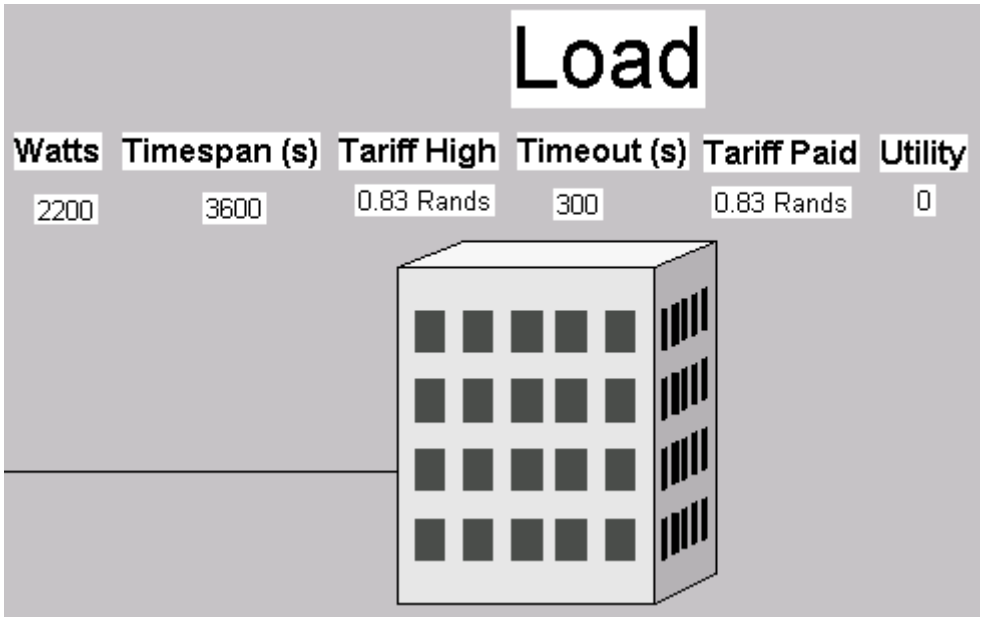


Figure 5-11 - Demonstration Load Update

A service on the load detects the new bid and sends the bid to the ST via web services. The ST receives the bid and stores bid information in its MySQL table for the sending load.

The bid is then sent by the ST to the generator via web services. The generator stores the bid in its MySQL database and decides whether it will meet the bid or not according to pre-set parameters for incoming bids. If it is capable of meeting the bid, its return value for the ST web service is the tariff at which it will generate the bid. If it cannot generate the bid, its return value is zero for the ST web service. It stores the bid in either a *bidsGenerated* or *bidsRejected* table depending on the result.

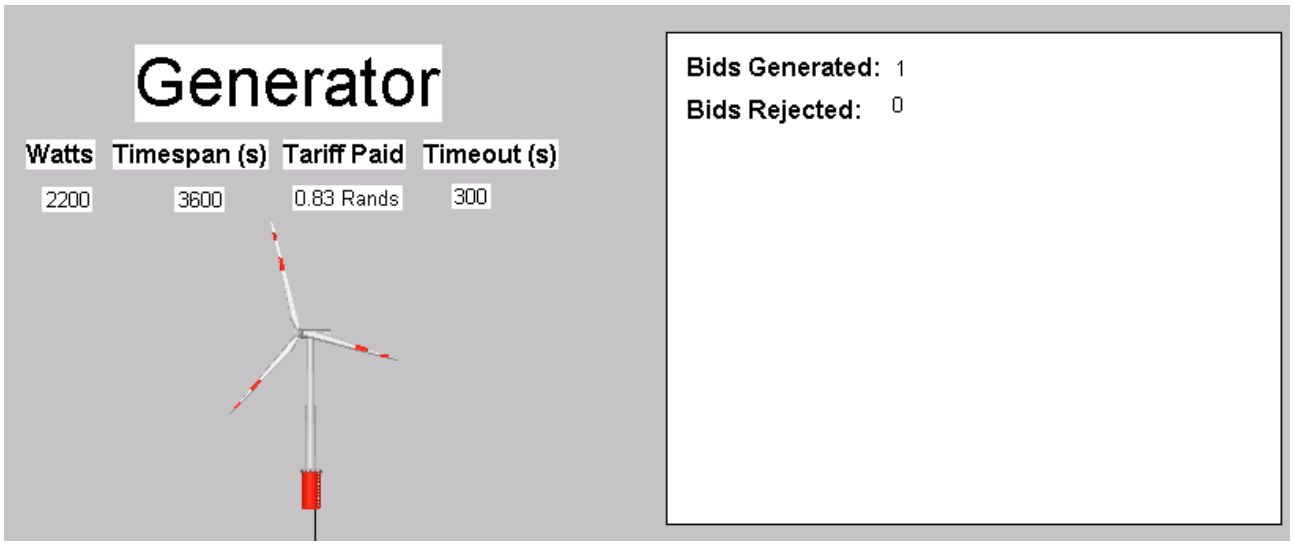


Figure 5-12 - Demonstration Generator Bid Information

Stellenbosch University https://scholar.sun.ac.za

The ST sends an indication that the bid has been met to the load and includes the source and tariff to be paid in the web service message sent. The ST also keeps track of transactions handled on its own MySQL database. In this way, a record of bids and generation for those bids are maintained on the ST. The ST has separate MySQL tables for each load and each generator within the microgrid. It also has a table for the electricity purchases and sales of the utility.

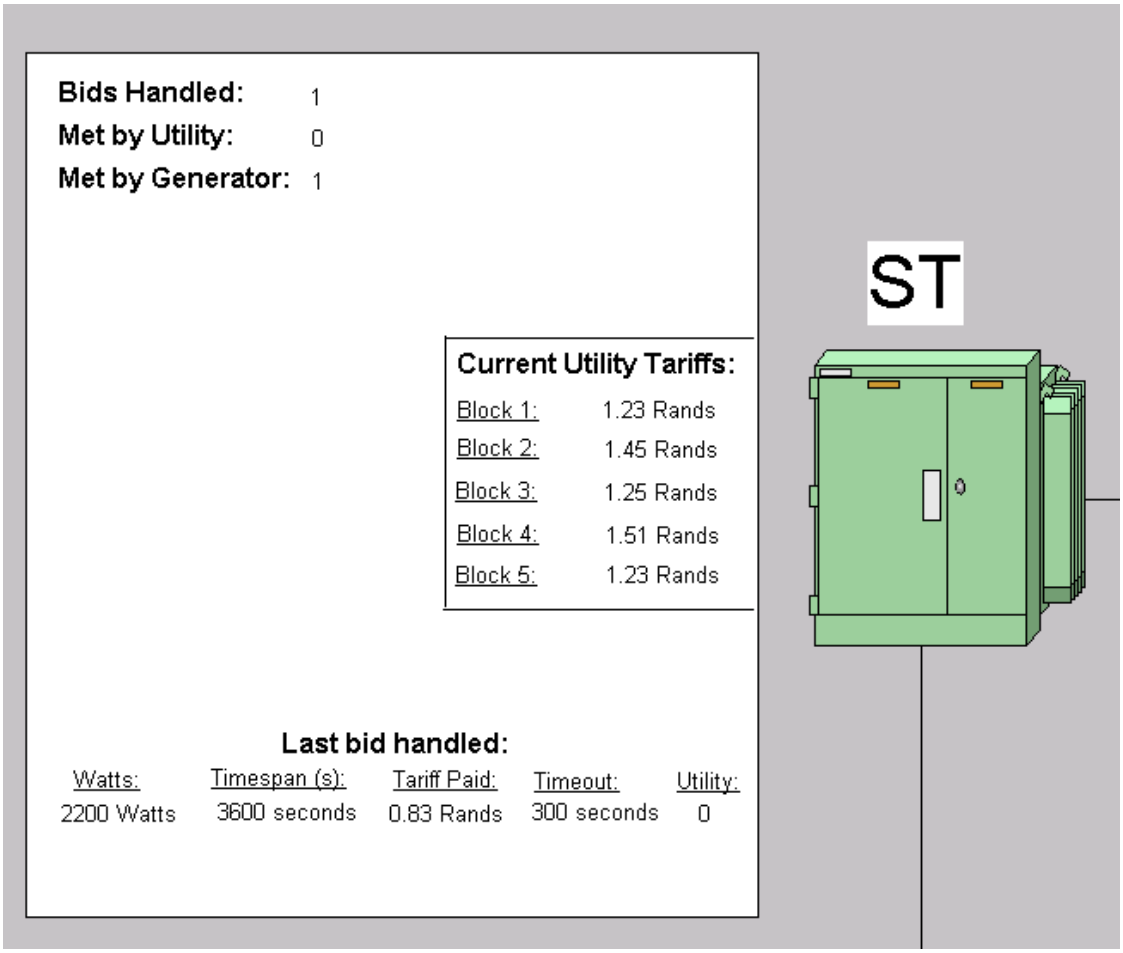


Figure 5-13 - Demonstration ST Information

Stellenbosch University https://scholar.sun.ac.za

CHAPTER 6
CONCLUSION

6.1 CONCLUSION

In this thesis, the concept and role of a smart transformer was investigated and an existing transformer was improved to move towards a fully functional smart transformer.

A device that interfaces with a rural electrification distribution transformer was developed. The device allows the transformer to meet the Eskom standard for all remote data retrieval devices on the Eskom network. The device employs the DNP3 and SNMP protocols for transformer condition and network monitoring respectively. The device is remotely configurable and can be accessed through either a web interface, an XML interface or an SSH interface. The device is developed to be robust, to automatically correct problems that may occur and to continue functioning without scheduled maintenance.

The device was incorporated into the transformer system and deployed in a deep rural setting for field testing. Field test results showed that the device functions reliably using the DNP3 and SNMP protocols to report data to a remote data server. The device was shown to be both remotely configurable and upgradable. Configuration settings on the device can be set and device software can be upgraded remotely.

The OSIsoft PI System was used to retrieve, store, manage and visualize data gathered from the transformers. The devices interfaces with the PI System on a VPN over the public internet. The connection was established using a commercial 3G mobile internet modem. The PI System proved to be an effective data management tool.

The concept of the smart transformer as a microgrid market-enabling device was also explored in this thesis. Microgrid markets and the role of the smart transformer within such a microgrid setting were researched. A simulation was set up to show the capability of the smart transformer to act as a microgrid market-enabling device.

6.2 FUTURE WORK

Future work should focus on continued development of the smart transformer concept. This includes the following aspects:

- Role of the smart transformer as microgrid market enabler.
- The smart transformer interfacing with other microgrid agents.
- The role of the smart transformer as the agent at the point of connection between the microgrid and the macrogrid. This includes the upgrading of the SRAPET transformer to support bi-direction electricity flow.

Stellenbosch University https://scholar.sun.ac.za

- The role of the smart transformer in maintaining microgrid stability. This includes control of electricity entering and leaving the microgrid, tap changing techniques and the interaction of the smart transformer with other microgrid players to ensure microgrid stability.
- The ability of the smart transformer to turn raw data into other usable formats such as knowledge and wisdom through machine learning techniques.

Stellenbosch University https://scholar.sun.ac.za

Bibliography

[1] A. Zaballos, A. Vallejo and J. M. Selga, “Heterogeneous Communication Architecture for the Smart Grid,” *IEEE Network,* September / October 2011.

[2] J. Serdyn and J. Beukes, “Self-Regulated and Protected Electrificaiton Transformer,” Eskom, 2012.

[3] E. Hossain, Z. Han and H. V. Poor, Smart Grid Communications and Networking, Cambridge: Cambridge University Press, 2012.

[4] T. J. Hyman, “Remote Device Communication Standard for Data Retrieval and Remote Access,” Eskom, 2013.

[5] K. D. McBee and M. G. Simões, “Reducing Distribution Transformer Losses Through the use of Smart Grid Monitoring,” in *North American Power Symposium*, Starkville, 2009.

[6] NIST, “NIST framework and roadmap for smart grid interoperability standards,” 2011. [Online]. Available: http://www.nist.gov/smartgrid/. [Accessed 13 May 2014].

[7] Y. Yang, F. Lambert and D. Divan, “A survey on technologies for implementing sensor networks for power delivery systems,” in *Proceedings of IEEE Power Engineering Society General Meeting*, 2007.

[8] H.-M. Kim, Y. Lim and T. Kinoshita, “An Intelligent Multiagent System for Autonomous Microgrid Operation,” *Energies,* vol. 5, pp. 3347 - 3362, 2012.

[9] Siemens AG, “Advanced Architectures and Control Concepts for More Microgrids,” Siemens AG, Erlangen, Germany, 2009.

[10] X. Liu, P. Wang and P. C. Loh, “A Hybrid AC/DC Microgrid and Its Coordination Control,” *IEEE Transactions on Smart Grid,* vol. 2, no. 2, pp. 278 - 286, 2011.

[11] A. Sajadi, S. S. Sebtahmadi, M. Koniak, P. Biczel and S. Mekhilef, “Distributed Control Scheme for Voltage Regulation in Smart Grids,” *International Journal of Smart Grid and Clean Energy,* vol. 1, no. 1, pp. 53-59, 2012.

[12] J. Hey, “The Data, Information, Knowledge, Wisdom Chain: The Metaphorical link,” December 2004. [Online]. Available: http://www.dataschemata.com/uploads/7/4/8/7/7487334/dikwchain.pdf. [Accessed 6 August 2014].

[1 M. S. Haider, M. Vaziri and D. Phillips, “Investigation of Overvoltage Effects for Commonly

Stellenbosch University https://scholar.sun.ac.za

3] Used Distribution Transformer," in *Power Engineering Society General Meeting*, Montreal, 2006.

[14] A. Engler, O. Osika, M. Barnes, N. Jenkins and A. Arulampalam, "Local Micro Source controller strategies and alghorithms," 2 February 2004. [Online]. Available: www.microgrids.eu/micro2000. [Accessed 16 May 2014].

[15] T. L. Vandoorn, J. D. M. D. Kooning, B. Meersman, J. M. Guerrero and L. Vandevelde, "Voltage-Based Control of a Smart Transformer in a Microgrid," in *IEEE Transactions on Industrial Electronics*, 2013.

[16] Q. Hu and F. Li, "Hardware Design of Smart Home Energy Management System with Dynamic Price Response," *IEEE Transactions on Smart Grid,* vol. 4, no. 4, pp. 1878-1887, 2013.

[17] World Bank Data, "Rural Population (% of total population)," [Online]. Available: http://data.worldbank.org/indicator/SP.RUR.TOTL.ZS. [Accessed 16 May 2014].

[18] ESKOM, "Eksom annual report 2006," 2006. [Online]. Available: http://www.eskom.co.za/annreport06/pdf/eskomar.pdf.

[19] N. M. Rao, R. Narayanan, B. R. Vasudevamurthy and S. K. Das, "Performance Requirements of Present-Day Distribution Transformers for Smart Grid," in *IEEE ISGT Asia*, Bangalore, 2013.

[20] J. O. Quevedo, J. C. Giacomini, R. C. Beltrame, F. E. Cazakevicius, C. Rech, L. Schuch, T. B. Marchesan, M. d. Campos, P. S. Sausen and J. R. Kinas, "Smart Distribution Transformer Applied to Smart Grids," in *Power Electronics Conference (COBEP)*, Gramado, Brazil, 2013.

[21] D. Martin, J. Wijaya, N. Lelekakis, D. Susa and N. Heyward, "Thermal analysis of two transformers filled with different oils," *Electrical Insulation Magazine,* pp. 39-45, 2014.

[22] M. Crabtree, Industrial Communications Handbook, Bedfordview: Crown Publications, 2013.

[23] IEEE Power and Energy Society, "IEEE Standard for Electric Power Systems Communications— Distributed Network Protocol (DNP3)," New York, 2012.

[24] Siemens, "Communication network solutions for smart grids," Nuremberg, 2011.

[25] G. Coley, "BeagleBone Black System Reference Manual," Texas Instruments, Dallas, 2013.

[26] J. Serdyn and J. Beukes, "Self-Regulated and Protected Electrification Transformer," Eskom, Stellenbosch, 2011.

Stellenbosch University https://scholar.sun.ac.za

[27] Dallas Semiconductor, “Application Note 83: Fundamentals of RS-232 Serial Communications,” [Online]. Available: http://www.pacontrol.com/download/RS232.pdf. [Accessed 17 June 2014].

[28] The Chinese University of Hong Kong, “Serial Pinout,” [Online]. Available: http://docs.sgi.com/library/dynaweb_docs/hdwr/SGI_EndUser/books/IXbricAdd/sgi_html/figures/IXbrick.serialport.pinouts.gif. [Accessed 29 November 2014].

[29] S. Asif, Wireless Communication Evolution to 3G and Beyond, Arthech House Inc., 2007.

[30] 4G Solution, “Huawei E3131 3G Dongle,” 4G Solution, [Online]. Available: http://www.4glteway.com/huawei-e3131-3g-dongle.html. [Accessed 8 October 2014].

[31] R. Peterson, Linux: The Complete Reference, Sixth Edition, McGraw-Hill, 2008.

[32] Archlinux, “systemd,” Archlinux, 27 July 2014. [Online]. Available: https://wiki.archlinux.org/index.php/systemd. [Accessed 7 August 2014].

[33] W. J. Gilmore, Beginning PHP 5 and MySQL, Apress, 2004.

[34] S. A. Gabarro, Web Application Design and Implementation, John Wilet & Sons, Inc., 2007.

[35] D. Mauro and K. Schmidt, Essential SNMP, O'Reilly Media, 2005.

[36] J. Case, M. Fedor, M. Schoffstall and J. Davin, “A Simple Network Management Protocol (SNMP),” Network Working Group, 1990.

[37] Net-Snmp, “README.agent-mibs,” [Online]. Available: http://www.net-snmp.org/docs/README.agent-mibs.html. [Accessed 15 March 2014].

[38] G. Fairhurst, “Unicast, broadcast and multicast,” 10 03 2009. [Online]. Available: http://www.erg.abdn.ac.uk/~gorry/course/intro-pages/uni-b-mcast.htmlv. [Accessed 15 03 2014].

[39] Manage Engine, *Manage Engine Mib-Browser Notes,* 2014.

[40] K. McCloghrie and M. Rose, “RFC1213 MIB-II,” Network Working Group, 1991.

[4 Net-Snmp, [Online]. Available: www.net-snmp.org. [Accessed 14 March 2014].

Stellenbosch University https://scholar.sun.ac.za

1]

[4 “Share libraries with GCC on Linux,” [Online]. Available:
2] http://www.cprogramming.com/tutorial/shared-libraries-linux-gcc.html. [Accessed 15 March 2014].

[4 Google Code, “Mongoose,” GNU, [Online]. Available: https://code.google.com/p/mongoose/.
3] [Accessed 8 August 2014].

[4 w3 Schools, “HTML Intro,” [Online]. Available: http://www.w3schools.com/html/html_intro.asp.
4] [Accessed 11 March 2014].

[4 W3C, “w3schools.com - PHP Sessions,” W3C, [Online]. Available:
5] http://www.w3schools.com/php/php_sessions.asp. [Accessed 18 August 2014].

[4 J. Pokorny, “XML Functionality,” in *Database Engineering and Applications Symposium*,
6] Yokohama, 2000.

[4 B. Stroustrap, The C++ Programming Language, Addison-Wesley Publishing Company, 1994.
7]

[4 Y.-A. Pignolet, T. Locher and R. Wattenhofer, “Principles of Distributed Computing,” Swiss
8] Federal Institute of Technology, Zurich, 2013.

[4 TFT, “Serial Ports in C++,” [Online]. Available:
9] http://www.webalice.it/fede.tft/serial_port/serial_port.html. [Accessed 12 March 2014].

[5 C. M. Kohlhoff, “Asio Rationale,” [Online]. Available:
0] http://www.boost.org/doc/libs/1_55_0/doc/html/boost_asio/overview/rationale.html. [Accessed 12 March 2014].

[5 J. Serdyn and C. Verster, “Self Regulated and Protected Electrification Transformer 2014,”
1] Eskom, Stellenbosch, 2013.

[5 MySQL, “Download Connector / C++,” [Online]. Available:
2] http://dev.mysql.com/downloads/connector/cpp/. [Accessed 13 March 2014].

[5 MySQL, “Chapter 1 Introduction to MySQL Connector/C++,” MySQL, [Online]. Available:
3] http://dev.mysql.com/doc/connector-cpp/en/connector-cpp-introduction.html. [Accessed 13 March 2014].

[5 A. J. Crain, “Automatak Github Repository,” [Online]. Available:
4] https://github.com/automatak/dnp3. [Accessed 17 March 2014].

[5 M. Kalicinski, “RAPIDXML Manual,” 2009. [Online]. Available:

Stellenbosch University https://scholar.sun.ac.za

5] http://rapidxml.sourceforge.net/manual.html. [Accessed 18 June 2014].

[5 6] T. Sailer, “lsusb manual page,” Linux Commands, 25 January 2005. [Online]. Available: http://linuxcommand.org/man_pages/lsusb8.html. [Accessed 12 August 2014].

[5 7] Archlinux, March 2013. [Online]. Available: https://wiki.archlinux.org/index.php/Wvdial.

[5 8] G. Holden, Guide to Firewalls and Network Security: Intrusion Detection and VPNs, Thomson Learning, 2004.

[5 9] OpenVPN, 2014 March. [Online]. Available: http://openvpn.net/index.php/about-menu/about-us.html.

[6 0] OpenVPN, 2014 March. [Online]. Available: http://openvpn.net/index.php/open-source/documentation/howto.html.

[6 1] OpenVPN, March 2014. [Online]. Available: https://openvpn.net/index.php/open-source/downloads.html.

[6 2] OSISoft, “Introduction to System Management,” OSISoft, San Leandro, 2012.

[6 3] OSISoft, “PI Server 2010: PI MDB to AF Transition Guide,” OSISoft, San Leandro, 2010.

[6 4] OSIsoft, “PI Processbook User Guide,” OSIsoft, San Leandro, California, 2005.

[6 5] OSISoft, “PI Interface for DNP3.0,” OSISoft, San Leandro, 2012.

[6 6] OSIsoft, “PI Advanced Computing Engine - OSIsoft,” 2012. [Online]. Available: http://www.osisoft.com/software-support/products/product_info/DS_PI_ACE_LT_EN.aspx. [Accessed 16 June 2014].

[6 7] OSIsoft Learning, “OSIsoft: Introduction to PI ACE. v2.1.8,” 25 May 2011. [Online]. Available: https://www.youtube.com/watch?v=EEt5r7rIiRU. [Accessed 16 June 2014].

[6 8] ManageEngine, “MibBrowser Free Tool,” ManageEngine, 2014. [Online]. Available: http://www.manageengine.com/products/mibbrowser-free-tool/. [Accessed 6 11 2014].

[6 9] J. A. P. Lopez, C. L. Moreira and A. Madureira, “Defining Control Strategies for MicroGrids Islanded Operation,” *IEEE Transactions on Power Systems,* vol. 21, no. 2, pp. 916-924, May 2006.

Stellenbosch University https://scholar.sun.ac.za

[70] R. M. Kamel, A. Chaouachi and K. Nagasaka, “Detailed Analysis of Micro-Grid Stability during Islanding Mode under Different Load Conditions,” *SCIRP Engineering,* vol. 3, no. 5, pp. 508 - 516, 2011.

[71] D. P. Ariyasinghe and D. M. Vilanthgamuwa, “Stability Analysis of Microgrids with Constant Power Loads,” in *International Conference on Sustainable Energy Technologies (ICSET)*, Singapore, 2008.

[72] S. Rahman, M. Pipattanasomporn and Y. Teklu, “Intelligent Distributed Autonomous Power Systems (IDAPS),” in *IEEE PES*, Tampa, Florida, 2007.

[73] R. Palma-Behnke, J. L. Cerda, L. S. Vargas and A. Jofre, “A Distribution Company Energy Acquisition Market Model with Integration of Distributed Generation and Load Curtailment Options,” *IEEE Transactions on Power Systems,* vol. 20, no. 4, 2005.

[74] J. P. Lopes, N. Hatziargyriou, J. Mutale, P. Djapic and N. Jenkins, “Intergrating distributed generation into electric power systems: A review of drivers, challenges and oppertunities,” *Electric Power Systems Research,* vol. 77, no. 9, pp. 1189-1203, 2006.

[75] O. Saadeh, October 2014. [Online]. Available: http://www.greentechmedia.com/research/report/distributed-energy-resource-management-systems-2014.

[76] Y.-F. Huang, P. Murphy, L. Rulli, M. D. Lemmon and T. G. Pratt, “Coupling Low-Voltage Microgrids into Mid-Voltage Distribution Systems,” University of Notre Dame - General Electric, South Bend, 2010.

[77] O. A. Sianaki, O. Hussain, T. Dillon and A. R. Tabesh, “Intelligent Decision Support System for Including Consumers' Preferences in Residential Energy Consumption in Smart Grid,” in *Second International Conference on Computational Intelligence, Modelling and Simulation*, 2010.

[78] L. Jidong, Z. Zehui, Z. Li and H. Xueshan, “Evaluating Short Term Benefits of Demand Response,” in *4th International Conference on Electric Utility Deregulation and Restructuring and Power Technologies*, Weihai, Shandong, 2011.

[79] J. A. Momoh and R. A. Sowah, “Distribution System Reliability in a Deregulated Environment: A Case Study,” *IEEE Transmission and Distribution Conference and Exposition,* vol. 2, pp. 562-567, 2003.

[80] F. Katiraei, R. Iravani, N. Hatziargyriou and A. Dimeas, “Microgrids Management: Controls and Operation Aspects of Microgrids,” IEEE Power and Energy Magazine, 2008.

Stellenbosch University https://scholar.sun.ac.za

[81] K. Eger, C. Gerdes and S. Rusitschka, “A Catallaxy-based Market Mechanism for Power Balancing,” in *6th International Conference on the European Energy Market*, Leuven, 2009.

[82] A. G. Tsikalakis and N. D. Hatziargyriou, “Centralized Control for Optimizing Microgrids Operation,” *IEEE Transactions on Energy Conversion,* vol. 23, no. 1, 2008.

[83] M. Rasheduzzaman, S. N. Bhaskara and B. H. Chowdhury, “Implementation of a Microgrid Central Controller in a Laboratory Microgrid Network,” in *North American Power Symposium*, Champaign, Illinois, 2012.

[84] F. Pilo, G. Pisano and G. G. Soma, “Neural Implementation of MicroGrid Central Controllers,” in *5th IEEE International Conference on Industrial Informatics*, Vienna, 2007.

[85] S. Krovvidi, “Competitive Microgrid Electricity Market Design,” Virginia Polytechnic Institute and State University, Arlington, Virginia, 2010.

[86] M. Leonard, “How to Choose the Right Platform: Raspberry Pi or BeagleBone Black?,” Make, 25 February 2014. [Online]. Available: http://makezine.com/magazine/how-to-choose-the-right-platform-raspberry-pi-or-beaglebone-black/. [Accessed 14 October 2014].

[87] Embedded Linux, “RPi Hardware,” eLinux, 16 July 2014. [Online]. Available: http://elinux.org/RPi_Hardware. [Accessed 17 October 2014].

[88] H.-J. Cha, J.-Y. Choi and D.-J. Won, “Smart Load Management in Demand Response using Microgrid EMS,” in *ENERGYCON*, Dubrovnik, Croatia, 2014.

[89] Ubuntu, “Meet Ubuntu,” Ubuntu, [Online]. Available: http://www.ubuntu.com/desktop. [Accessed 14 March 2014].

[90] Archlinux, “Archlinux,” Archlinux, [Online]. Available: https://www.archlinux.org/. [Accessed 15 March 2014].

[91] Archlinux, “The Arch Way,” Archlinux, [Online]. Available: https://wiki.archlinux.org/index.php/The_Arch_Way. [Accessed 15 March 2014].

[92] J. Snell, D. Tidwell and P. Kulchenko, Programming Web Services with SOAP, 1st ed., Sebastopol: O'Reilly, 2001.

[93] R. A. v. Engelen and K. A. Gallivan, “The gSOAP Toolkit for Web Services and Peer-To-Peer Computing Networks,” in *2nd IEEE International Symposium on Cluster Computing and the Grid*, Berlin, 2002.

Stellenbosch University https://scholar.sun.ac.za

APPENDIX A
DISTRIBUTED NETWORK PROTOCOL 3

This chapter discusses the DNP3 (Distributed Network Protocol Version 3) protocol standard.

DNP3 (Distributed Network Protocol Version 3) is an open, efficient, robust and intelligent utility protocol designed to optimize the transmission of information pertaining to data acquisition and control commands across a utility network. The protocol was originally created by Westronic Inc. between 1992 and 1994 to address various issues that they identified in protocols in the utility industry. Two of the main objectives in the DNP3 project were to reduce the amount of bandwidth used to communicate data while not compromising on data integrity [22] [23].

A.1 LAYERING

DNP3 is a layered protocol based on a simplified model of the Open Systems Interconnection (OSI) layering model for communication systems. DNP3 consists of three layers each performing a specific function, but working together to accomplish a unified goal. Layers are seen in a hierarchical structure and higher layers rely on lower layers for the completion of more foundational tasks that are needed to complete their own tasks. The three layers in the DNP3 protocol, listed according to hierarchy from lowest to highest are: the Data Link Layer, the Application layer and the User Layer. The concept is illustrated in Figure A-1.

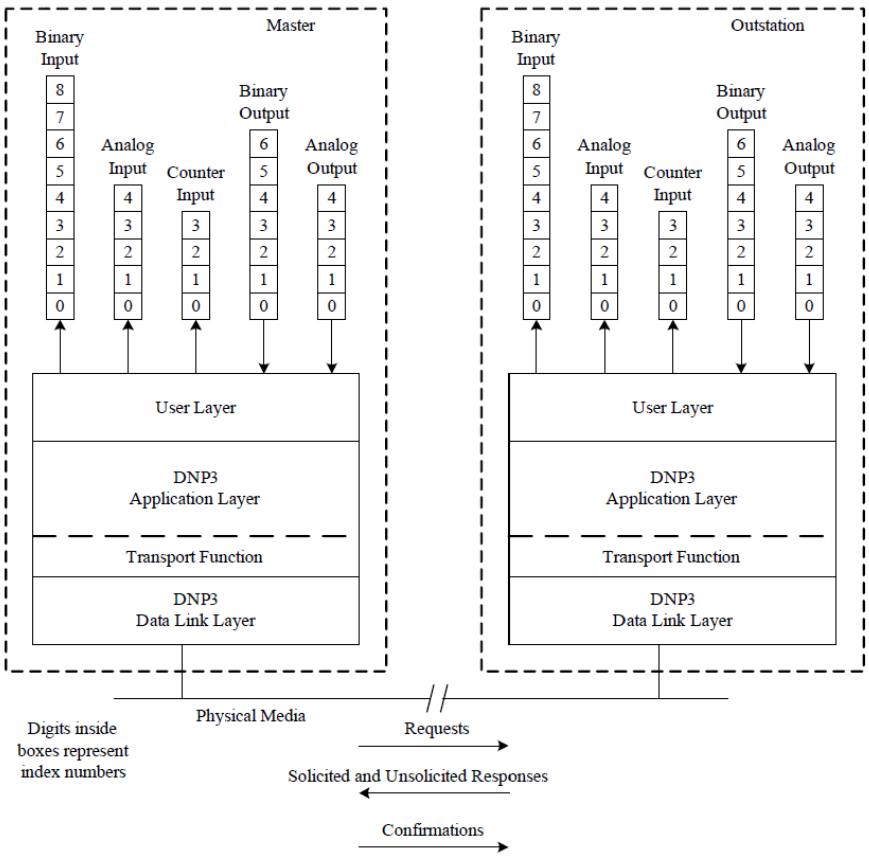


Figure A-1 - DNP3 Master-Outstation Model [23]

Stellenbosch University https://scholar.sun.ac.za

In Figure A-1 a Transport Function is included. This may be seen as a pseudo-layer contained within the Application layer. Layers act as information adders and removers that build packets of data to be transmitted across a transmission medium to another device. When a packet is initialized, every layer in the sending device, starting from the User layer and moving through to the Data Link layer, adds information to the packet. The packet is then sent over a transmission medium to a receiving device. When the receiving device receives the packet, the roles of the layers are reversed and each layer removes information from the packet, exposing the data that is desired to be read at the User layer.

A.2 DATA PACKET STRUCTURE

The DNP3 data packet is of a fixed structure. Each protocol layer modifies the packet and either adds information to it or removes information from it. Figure A-2 illustrates how the packet looks at each layer in the protocol. The purpose of the Transport Function contained in the Application layer is to partition the data packet received from the Application layer into smaller segments that can be handled by the Data Link layer. These segments are called Data Link layer frames and the Data Link layer adds information to the partitioned segments from the Transport Function, preparing them to be sent across transmission media.

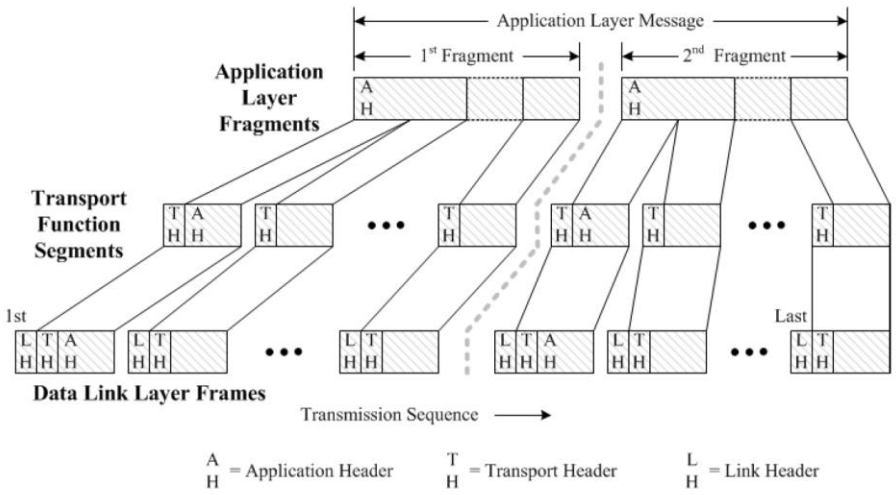


Figure A-2 - Fragmented Application Layer Message [23]

As can be seen in Figure A-2, the original message created is a data packet partitioned into two fragments. The number of fragments that a message is broken into is dependent on the size of the message. Smaller messages may fit into a single fragment while larger messages will need to be broken up into multiple fragments. The Application layer adds an Application Header to the message and sets the limit on memory requirements for message reception. The Transport Function then takes the message created by the Application layer and breaks each fragment into segments that fit into a Data Link frame. Each segment contains a Transport Header, but only the first segment of each fragment contains an Application Header. Each segment is limited to a maximum of 250 octets including the Transport Header. The Data Link layer then adds information

Stellenbosch University https://scholar.sun.ac.za

to each frame including a Link Header containing a Cyclic Redundancy Check (CRC) octet. At the Data Link layer, a frame may contain a maximum of 292 octets including the Link Header [23].

A.3 MESSAGE SEQUENCING

As previously mentioned, there are two possible causes for data transfer in DNP3: A master polls an outstation for data or an outstation sends an unsolicited (spontaneous) message to a master. The two scenarios are illustrated in Figure A-3 and Figure A-4.

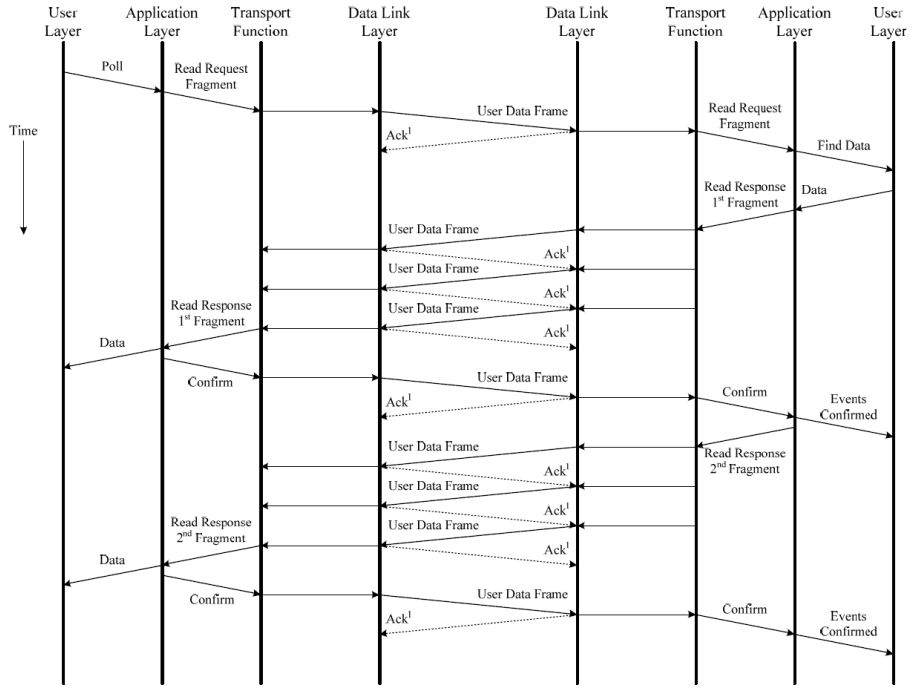


Figure A-3 - Polled Message Sequence [23]

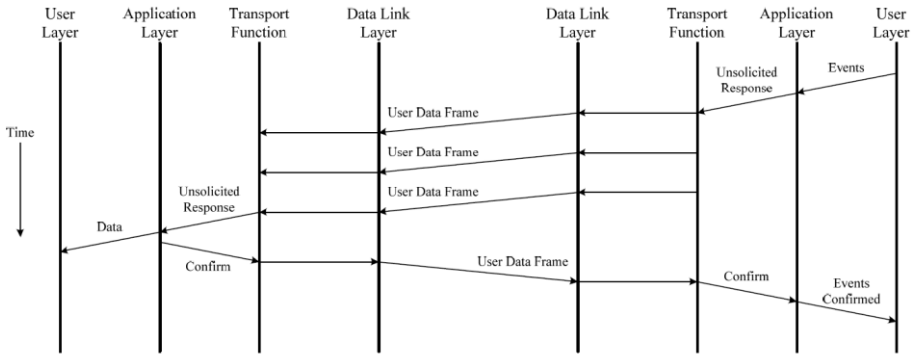


Figure A-4 - Unsolicited Response Sequence [23]

In the case of a master polling an outstation for information, the transaction is initiated in the User layer of the master station. The data packet is then constructed by the Application layer, Transport Function and Data Link layer and sent to the desired outstation. On reception of the message, the outstation responds with and acknowledge message and sends the received packet through its own layers to be broken up and interpreted. The outstation searches for the data requested by the master and once found sends the data through its layers to build a DNP3 data packet containing

Stellenbosch University https://scholar.sun.ac.za

the data. The outstation then sends each Data Link frame to the master in sequential order and the master responds with an acknowledge after reception of each frame. When all frames in the first fragment of the message is sent and received, the master sends a confirm message to the outstation confirming that it has received the first fragment. The outstation responds to the master with an acknowledge message and the second frame is then sent to the master in the same manner as the first. This process continues until all fragments have been sent [23].

An unsolicited message is sent in the same manner except that the message is initiated in the User layer of the outstation and the master does not send any acknowledge messages to confirm reception of Data Link frames. A single confirm message is sent from the master to the outstation at the end of the message to confirm that the message was received [23].

A.4 DNP3 POINTS

A DNP3 point is a single physical or logical entity defined as an input or an output. Points are classified by point types. Point types include binary input, analog inputs, counters, frozen counters, binary outputs and analog outputs. Points are identified using point indexes: a numeric identifier that defines a point from other points of the same point type. A point also has a static value at a certain time and may generate events when this value changes. Analog input points may have a name, a scale factor and a deadband in addition to a value. Point storage in an outstation may be viewed as an outstation containing a database of points, with all points of a specific type contained in a table. Each table contains a unique entry for each point, identified by its point index. This concept is illustrated in Figure A-1 where a “table” for each point type is shown above the User layer [23].

DNP3 Groups

DNP3 object groups classify points into categories that define point type and the method that was used to generate the data. As an example, object group 1 represents binary input data while object group 2 represents binary input event data (created by the occurrence of an event) [23].

DNP3 Variations

DNP3 variations offer various data formats for each data type. A DNP3 object group may have many possible variations. For example, object group 2 variation 0 represents any variation of binary input events while variation 1 represents only binary input events without an attached time and variation 2 represents binary input events with an absolute time attached [23].

DNP3 Events

DNP3 events occur when a significant happening takes place. This happening may be [23]:

- A state change.
- Analog values crossing pre-set deadband.

Stellenbosch University https://scholar.sun.ac.za

- A snapshot of data is taken.

DNP3 Class Scans

DNP3 classes are used to classify data into user-defined categories. Data can be assigned to either Scan-class 1, 2 or 3. Scan-class 0 automatically contains all static data. All data contained in a particular scan-class may be polled by a master station, giving users freedom to categorize data for mass retrieval [23].

A.5 THE APPLICATION LAYER

At the Application layer, the message to be sent is divided into fragments. A fragment contains an Application Header and zero or more DNP3 data objects, each with its own Data Object Header. The number of fragments a message is divided into is determined by the message size. Small messages can fit into a single fragment while larger messages may require multiple fragments to store the message data. A fragment may either contain a request or a response message. Request messages are sent from master stations to outstations requesting data while response messages are sent from outstations to master stations responding with data. The request and response fragments are shown in Figure A-6 and Figure A-7 respectively. The response fragment's Application Response Header contains one additional field to the Application Request Header: an internal indications field containing diagnostic data about the health of an outstation [23].

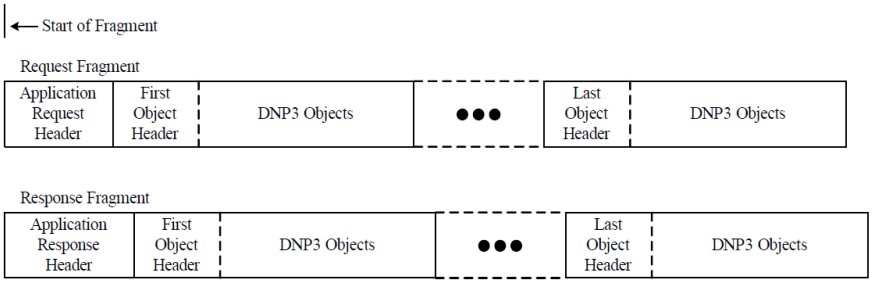
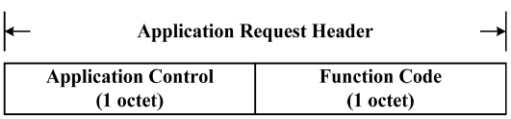


Figure A-5 - Application Layer Message Fragment Structure [23]

The Application Request Header is sent from master stations to outstations and is shown in Figure A-6. It contains two octets: an application control octet and a function code octet. The application control octet contains information that is used to reconstruct multiple fragment messages. The function code octet contains a function code that identifies the purpose of the message. A table containing all function codes can be seen in APPENDIX A.9. Request messages use function codes 0 through 128 and response messages use function codes 129 through 255. For example, a READ function code will tell an outstation to return the data requested in the message to the master.



Stellenbosch University https://scholar.sun.ac.za

Figure A-6 - Application Request Header [23]

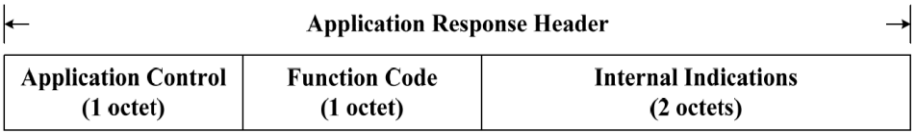


Figure A-7 - Application Response Header [23]

The application response header is shown in Figure A-7 and contains one additional field to the request header: the internal indications field. The internal indications field is two octets in length and contains information pertaining to certain states and error conditions in the outstation. A list of all internal indications bits and their purposes may be viewed in APPENDIX A.10. (Note that IIN1.0 refers to the first bit in the first octet and IIN2.1 refers to the second bit in the second octet).

As can be seen in Figure A-5, a fragment contains one or more DNP3 data objects each with an Object Header. The truth is, both request and response message fragments contain Data Object Headers, but only response message fragments contain actual data objects. This is because a request message sends no data to the outstation; it only requests data from it. In a request message, Data Object Headers specify which data values and formats are desired by the master. In the response message, similar Data Object Headers are found, but now each has an attached data object containing the data desired by the master. A Data Object Header is illustrated in Figure A-8.

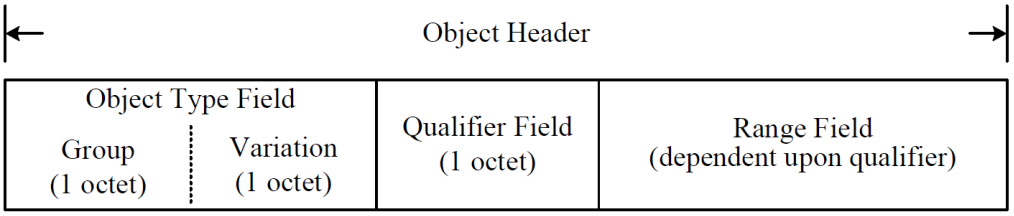


Figure A-8 – Object Header [23]

The group octet is important for specifying the data and value types contained in either a request or response data object. The same group octet will be contained in the request and response message as the request message specifies what data is requested by the master and the response message contains that data. The only exception to this is when object group 60 is used.

Object group 60 is a special group used only in request messages to gather data contained in a specific scan class. When object group 60 is specified in the request message, all data of either scan-class 0, 1, 2 or 3 will be returned depending on the variation contained in the variation octet. The response message (or messages) to this request will contain object headers that contain group numbers that specify what objects are being returned, not group 60 as in the request message [23].

Object variations determine what format the data in the data object is returned in. Variations can be used to return the data in a variety of pre-determined formats. For example, some data may be

Stellenbosch University https://scholar.sun.ac.za

returned as either 16-bit, 32-bit, floating point or long floating point values. Variations can also specify whether an additional flag octet must be included in the object.

Variation 0 is a special variation used in request messages that indicates that a master has no preference to the format in which data is returned from the outstation. When an outstation receives a request containing variation 0, it will determine the format in which to return the data itself. An outstation does not necessarily support all of the possible variations available in DNP3. If a variation is requested by a master and not supported by an outstation, the outstation must respond by setting internal indications bits indicating that the desired variation is not supported [23].

The Qualifier field contains information specifying the index number where data is to be found in the outstation and possibly also specifying the data object size. It also contains a range specifier code that describes the range field (see Figure A-8).

The Range field specifies the length of the data object and whether it possesses start and stop indices [23].

A.6 DEVICE STARTUPS

Outstation

When an outstation starts up, it sets the internal indications restart bit in all messages to the master to indicate a restart has occurred. If unsolicited reporting is enabled on an outstation it will, upon startup, send an unsolicited null response message (with no data objects) to the master station. The outstation will continue to send these unsolicited messages until it receives a confirmation message from the master indicating reception of the unsolicited message.

The master will reply with an unsolicited confirmation message whenever an unsolicited message is received from an outstation. If a master receives a message from an outstation where the restart internal indications bit is set, it will set up secure authentication between the master and the outstation and send a request to set the restart internal indications bit low. The master also performs an integrity poll when a restart is detected in an outstation. In an integrity poll, the master polls for all the data contained in an outstation and ensure that its point database and the outstation point database match [23].

Master

When starting up, the master performs a set of actions for each outstation connected to it. Firstly, the master sets up secure authentication with each outstation if it is supported by both the outstation and the master. The master then downloads a XML device profile from each outstation: a document stating the DNP3 capabilities of the outstation. The master then reads the device attributes of each outstation and performs a time-synchronization, synchronizing the outstation time to that of the master if so specified in the outstation configuration. Next the master station sets

Stellenbosch University https://scholar.sun.ac.za

up data attributes on the outstations and sends an optional integrity or event poll to the outstation, polling for data that changed during the time it was offline [23].

Secure Authentication

The DNP3 protocol supports secure authentication to protect against threats such as spoofing, modification, replay and eavesdropping. The secure authentication implementation occurs in the Application layer. DNP3 uses a authentication model whereby Message Authentication Codes (MAC) are calculated based on protocol messages received by both master and outstation devices. A MAC algorithm takes a protocol message as input, performs a calculation on it and produces a result to the calculation as output, the result being smaller than the original input.

DNP3 uses a challenge-response model for the authentication procedure. A challenge message is generated by either a master or an outstation and contains information (a MAC algorithm) specifying how a responding device should build a reply to the challenge message. The challenge message arrives at the device that is to respond and that device uses the challenge message to build a response message. This is done by calculations being performed on both devices using session keys that are known to both devices. If the responding device gets results to the calculations that match those calculated by the challenging device, authentication passes and communication continues. If there is a difference in calculation results, the challenging device responds to the responding device with an error message [23]. These concepts are illustrated in Figure A-9 and Figure A-10.

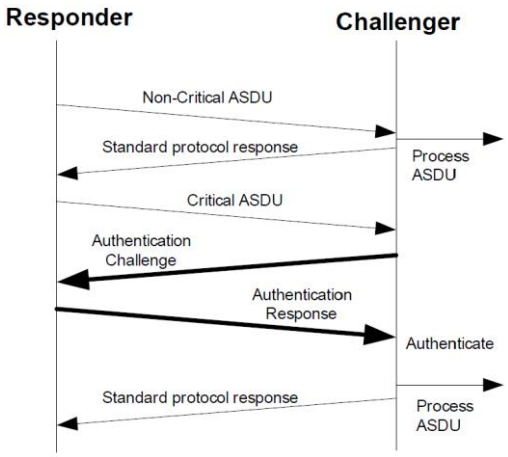


Figure A-9 - Successful Authentication [23]

Stellenbosch University https://scholar.sun.ac.za

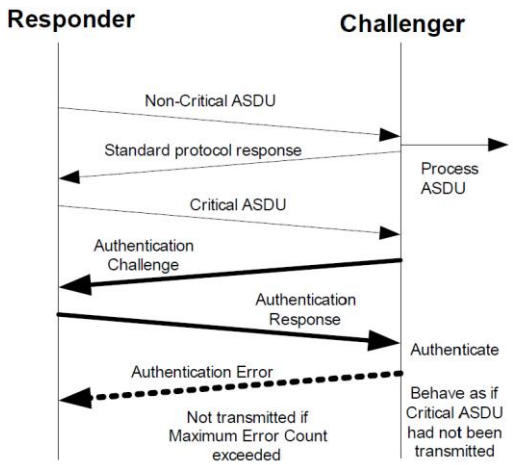


Figure A-10 – Failed Authentication [23]

A.7 TRANSPORT FUNCTION

The Transport Function can be seen as a pseudo layer that is contained inside the Application layer. All messages that pass into and out of the Application layer must pass through the Transport Function. DNP3 sends data in packets called Data Link Layer Frames, which have a maximum length that may be shorter than the complete message coming from the Application layer. For outgoing messages, the Transport Function breaks Application layer messages up into sizes that can fit into these Data Link frames and attaches a header to each frame that indicates its position in relation to the message as a whole, thus slotting frames into a sequential order.

For incoming messages, the Transport Function first checks that Data Link frames arrive in the correct order by checking the Transport Header that is attached to the frames in the Transport Function of the sending device, and then reassembles these frames to form a complete Application layer fragment. If any frame arrives out of order, the entire message fragment is discarded. The Transport Function will only notify the Application layer that a message has arrived when the message is received in its entirety [23].

A.8 THE DATA LINK LAYER

The Data Link layer is the lowest hierarchical layer in the DNP3 protocol. The layer has two main functions:

1) Facilitating the transportation of Data Link frames to another DNP3 device through an attached communication medium.
2) The Data Link layer manages error handling, flow control and data link frame synchronization.

The Data Link layer attaches the Data Link Header to Data Link frames. The header contains the addresses of the source and destination of the message, indicates whether it is being sent from master to outstation or outstation to master and includes a Cyclic Redundancy Check (CRC) field.

Stellenbosch University https://scholar.sun.ac.za

The CRC field contains results of a calculation performed using bits in the Data Link Header and its purpose is to ensure that data integrity is maintained. The CRC is calculated on both the sending and receiving device and the results are compared as a checking mechanism [23].

The Data Link layer frame is divided into multiple 16 octet data blocks. The amount of data blocks is determined by the amount of data to be contained in the Data Link layer frame. A CRC octet is attached to the end of each data block to ensure data integrity [23]. The Data Link Header and data blocks can be seen in Figure A-11.

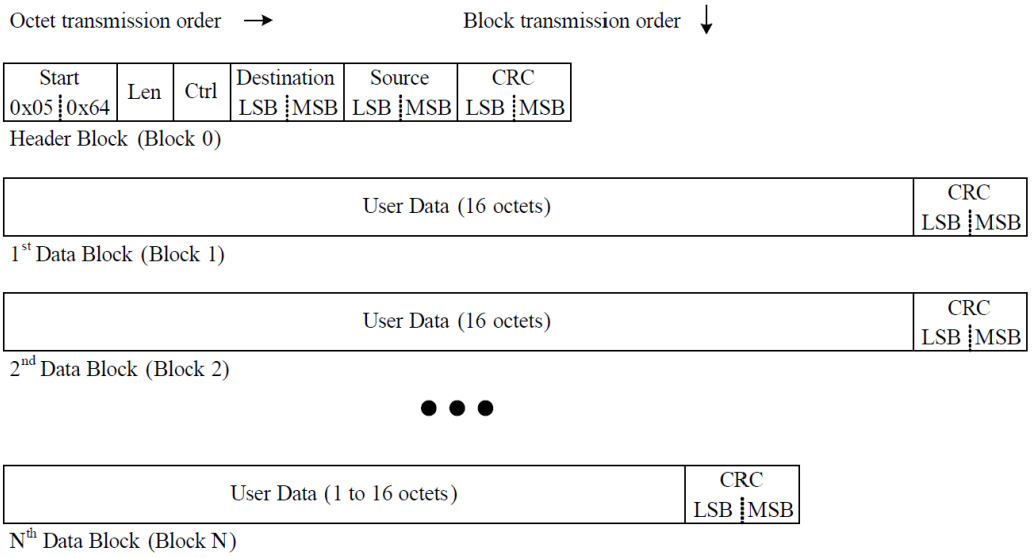


Figure A-11 – DNP3 Data Link Frame Format [23]

A.9 DNP3 FUNCTION CODES

Message Type	Code	Name	Brief Description
Confirmation	0 0x00	CONFIRM	Confirm Function Code: Master sends this to an outstation to confirm the receipt of an Application Layer fragment.
Request	1 0x01	READ	Read Function Code: Outstation shall return the data specified by the objects in the request.
Request	2 0x02	WRITE	Write Function Code: Outstation shall store the data specified by the objects in the request.
Request	3 0x03	SELECT	Select Function Code: Outstation shall select (or arm) the output points specified by the objects in the request in preparation for a subsequent operate command. The outstation shall not activate the outputs until a request with a matching Operate function code is received.

Stellenbosch University https://scholar.sun.ac.za

Request	4 0x04	OPERATE	Operate Function Code: Outstation shall activate the output points selected (or armed) by a previous select function code command.
Request	5 0x05	DIRECT_OPERATE	Direct Operate Function Code: Outstation shall immediately actuate the output points specified by the objects in the request. A prior matching select command is not required.
Request	6 0x06	DIRECT_OPERATE_NR	Direct Operate—No Response Function Code: Same as function code 5 but outstation shall not send a response.
Request	7 0x07	IMMED_FREEZR	Immediate Freeze Function Code: Outstation shall copy the point data values specified by the objects in the request to a separate freeze (or holding) buffer (or register).
Request	8 0x08	IMMED_FREEZE_NR	Immediate Freeze—No Response Function Code: Same as function code 7 but outstation shall not send a response.
Request	9 0x09	FREEZE_CLEAR	Freeze and Clear Function Code: Outstation shall copy the point data values specified by the objects in the request into a separate freeze (or holding) buffer (or register). After the copy operation, clear the point data values to zero.
Request	10 0x0A	FREEZE_CLEAR_NR	Freeze and Clear—No Response Function Code: Same as function code 9 but outstation shall not send a response.
Request	11 0x0B	FREEZE_AT_TIME	Freeze at Time Function Code: Outstation shall copy the point data values specified by the objects in the request to a separate freeze (or holding) buffer (or register) at the time and/or time intervals specified in a special time data information object.
Request	12 0x0C	FREEZE_AT_TIME_NR	Freeze at Time—No Response Function Code: Same as function code 11 but outstation shall not send a response.
Request	13 0x0D	COLD_RESTART	Cold Restart Function Code: Outstation shall perform a complete reset of all hardware and software in the device.
Request	14 0x0E	WARM_RESTART	Warm Restart Function Code: Outstation shall reset only portions of the device.
Request	15 0x0F	INITIALIZE_DATA	Initialize Data Function Code: Obsolete—Do not use for new designs.
Request	16 0x10	INITIALIZE_APPL	Initialize Application Function Code: Outstation shall place the applications specified by the objects in the request into the ready to run state.

Stellenbosch University https://scholar.sun.ac.za

Request	17 0x11	START_APPL	Start Application Function Code: Outstation shall start running the applications specified by the objects in the request.
Request	18 0x12	STOP_APPL	Stop Application Function Code: Outstation shall stop running the applications specified by the objects in the request.
Request	19 0x13	SAVE_CONFIG	Save Configuration Function Code: Outstation shall store into non-volatile memory the contents of a configuration file located in volatile memory. This code is deprecated—Do not use for new designs.
Request	20 0x14	ENABLE_UNSOLICITED	Enable Unsolicited Responses Function Code: Enables outstation to initiate unsolicited responses from points specified by the objects in the request.
Request	21 0x15	DISABLE_UNSOLICITED	Disable Unsolicited Responses Function Code: Prevents outstation from initiating unsolicited responses from points specified by the objects in the request.
Request	22 0x16	ASSIGN_CLASS	Assign Class Function Code: Outstation shall assign the events generated by the points specified by the objects in the request to one of the classes.
Request	23 0x17	DELAY_MEASURE	Delay Measurement Function Code: Outstation shall report the time it takes to process and initiate the transmission of its response. This allows the master to compute the propagation delay in the communications channel. Used for non-LAN time synchronization.
Request	24 0x18	RECORD_CURRENT_TIME	Record Current Time Function Code: Outstation shall save the time when the last octet of this message is received. Used for LAN time synchronization.
Request	25 0x19	OPEN_FILE	Open File Function Code: Outstation shall open a file.
Request	26 0x1A	CLOSE_FILE	Close File Function Code: Outstation shall close a file.
Request	27 0x1B	DELETE_FILE	Delete File Function Code: Outstation shall delete a file.
Request	28 0x1C	GET_FILE_INFO	Get File Information Function Code: Outstation shall retrieve information about a file.
Request	29 0x1D	AUTHENTICATE_FILE	Authenticate File Function Code: Outstation shall return a file authentication key.
Request	30 0x1E	ABORT_FILE	Abort File Function Code: Outstation shall abort a file transfer operation.

Stellenbosch University https://scholar.sun.ac.za

Request	31 0x1D	ACTIVATE_CONFIG	Activate Configuration Function Code: Outstation shall use the configuration specified by the objects in the request.
Request	32 0x20	AUTHENTICATE_REQ	Authentication Request: Outstation shall process a Secure Authentication object
Request	33 0x21	AUTH_REQ_NO_ACK	Authentication Request No Acknowledgment: Outstation shall process a Secure Authentication object, but outstation shall not send a response.
Request	34 to 128 0x22 to 0x80		Reserved.
Response	129 0x81	RESPONSE	Solicited Response: Master shall interpret this fragment as an Application Layer response to an Application Layer request sent by the master.
Response	130 0x82	UNSOLICITED_RESPONSE	Unsolicited Response: Master shall interpret this fragment as an unsolicited response that was not prompted by an explicit request.
Response	131 0x83	AUTHENTICATE_RESP	Authentication Response: Master shall interpret this fragment as an authentication response.
Response	132 to 255 0x84 to 0xFF		Reserved.

[23]

Stellenbosch University https://scholar.sun.ac.za

A.10 INTERNAL INDICATIONS BITS

Bit	Name	Brief description
IIN1.0	BROADCAST	A broadcast message was received. Reference 4.5.1
IIN1.1	CLASS_1_EVENTS	The outstation has unreported Class 1 events. Reference 4.5.2
IIN1.2	CLASS_2_EVENTS	The outstation has unreported Class 2 events. Reference 4.5.3
IIN1.3	CLASS_3_EVENTS	The outstation has unreported Class 3 events. Reference 4.5.4
IIN1.4	NEED_TIME	Time synchronization is required. Reference 4.5.5
IIN1.5	LOCAL_CONTROL	One or more of the outstation's points are in local control mode. Reference 4.5.6
IIN1.6	DEVICE_TROUBLE	An abnormal, device-specific condition exists in the outstation. Reference 4.5.7
IIN1.7	DEVICE_RESTART	The outstation restarted. Reference 4.5.8
IIN2.0	NO_FUNC_CODE_SUPPORT	The outstation does not support this function code. Reference 4.5.9
IIN2.1	OBJECT_UNKNOWN	Outstation does not support requested operation for objects in the request. Reference 4.5.10
IIN2.2	PARAMETER_ERROR	A parameter error was detected. Reference 4.5.11
IIN2.3	EVENT_BUFFER_OVERFLOW	An event buffer overflow condition exists in the outstation, and at least one unconfirmed event was lost. Reference 4.5.12
IIN2.4	ALREADY_EXECUTING	The operation requested is already executing. Support is optional. Reference 4.5.13
IIN2.5	CONFIG_CORRUPT	The outstation detected corrupt configuration. Support is optional. Reference 4.5.14
IIN2.6	RESERVED_2	Reserved for future use. Always set to 0. Reference 4.5.15
IIN2.7	RESERVED_1	Reserved for future use. Always set to 0. Reference 4.5.16

[23]

Stellenbosch University https://scholar.sun.ac.za

A.11 DNP3 DATA POINTS

DNP3 Index:	Point name:
ANALOG:	
1.	StatusA average
2.	TapA average
3.	StatusB average
4.	TapB average
5.	Source Voltage A average
6.	Load Voltage A average
7.	Load Current A average
8.	Load Current O A average
9.	Source Voltage B average
10.	Load Voltage B average
11.	Load Current B average
12.	Load Current O B average
13.	Earth Current (mA) average
14.	Earth Current Max (mA) average
15.	Interior Temperature average
16.	Exterior Temperature average
17.	Heat sink 1 Temperature average
18.	Heat sink 2 Temperature average
19.	Oil Temperature average
20.	Hotspot A Temperature average
21.	Hotspot B Temperature average
22.	ps5V average
23.	Life Loss Rate average
24.	Life Loss average
25.	StatusA event
26.	TapA event
27.	StatusB event
28.	TapB event
29.	Source Voltage A event
30.	Load Voltage A event
31.	Load Current A event
32.	Load Current O A event
33.	Source Voltage B event
34.	Load Voltage B event
35.	Load Current B event
36.	Load Current O B event
37.	Earth Current (mA) event
38.	Earth Current Max (mA) event
39.	Interior Temperature event
40.	Exterior Temperature event
41.	Heat sink 1 Temperature event
42.	Heat sink 2 Temperature event
43.	Oil Temperature event
44.	Hotspot A Temperature event
45.	Hotspot B Temperature event
46.	Ps5V event
47.	Life Loss Rate event
48.	Life Loss event
49.	Source Voltage A max

Stellenbosch University https://scholar.sun.ac.za

50. Source Voltage A min
51. Load Voltage A max
52. Load Voltage A min
53. Load Current A max
54. Load Current A min
55. Source Voltage B max
56. Source Voltage B min
57. Load Voltage B max
58. Load Voltage B min
59. Load Current B max
60. Load Current B min
61. Hotspot Temperature A max
62. Hotspot Temperature A min
63. Hotspot Temperature B max
64. Hotspot Temperature B min

BINARY:

---

1. Supply Voltage A Undervoltage -> Binary
2. Supply Voltage B Undervoltage -> Binary
3. Supply Voltage A Overvoltage
4. Supply Voltage B Overvoltage

Stellenbosch University https://scholar.sun.ac.za

APPENDIX B
LINUX

Linux is an open-source operating system developed by Linus Torvalds in the early 1990s. Linux offers a wide range of functionalities for many different applications in almost all spheres of computer usage, from engineering to office use. It is an operating system characterized by its no-cost availability, speed, efficiency, scalability and flexibility and is widely implemented in network servers, workstations and embedded applications.

Linux is built on the Linux kernel; the core program behind the operating system. Many different distributions of Linux have been built upon this foundational kernel, each characterized by a unique selection of programs, functionalities and implementation purposes. Some distributions specialize in providing a stable desktop environment for a PC, while other distributions are developed for use in embedded systems where resources are scarce and must be optimally used etc.

All Linux distributions offer a shell and a file system. The shell is an interactive and non-interactive command-line (text based) interpreter that a user can use to perform various tasks on the system. Interactive because a user can enter instructions via the command line to perform tasks and view information, non-interactive because operations can be run in the background while the user attends to other tasks. Common shells include the Bourne Again Shell (BASH), the Z shell and the TCSH shell, with the BASH shell being the default installed on most Linux distributions.

The Linux file system is a hierarchically connected structure of files and directories. A user may access, edit, rearrange and create files and directories from the shell. The shell provides effective tools for a user to easily perform these operations. The hierarchical structure of the Linux file system begin with the root directory simply known as “/”. Many other directories branch from the root directory and it may be seen as the base directory of the file system, within which all other files and directories are contained [31].

THE BASH SHELL

The BASH shell provides a set of default commands that allow a user to perform tasks. These tasks range from viewing all files and directories within a directory to accessing and controlling hardware and everything in between. The BASH shell also provides the user with the BASH scripting language; a language that can be used to perform a set of tasks in a predefined order, can take input from a user to produce output and can store data in variables and use them again [31].

FILE SYSTEM STRUCTURE

The default Linux file system has a defined structure containing various directories with specific functions. Linux contains two types of directories: home directories and system directories. Home

Stellenbosch University https://scholar.sun.ac.za

directories are assigned to users as their own directories where they can create, edit and remove files and directories. System directories are used to contain files and programs that run and maintain the system. Some of the more important default system directories are briefly described here.

- /bin – Holds standard program execution files. If an execution file is contained in the /bin directory, it may be run in the terminal as a command. For example, if there is an executable called “date” in the /bin directory that executes the date program, “date” may be entered as a command into the terminal and the date program will be executed.
- /usr – Holds files and commands used by the system. Contains sub-directories.
- /usr/bin – Contains execution files for user-oriented programs.
- /usr/sbin – Holds executable files for system administration-oriented programs.
- /usr/lib – Holds libraries for various programming languages.
- /usr/share/doc – Holds documentation for Linux and various installed programs.
- /var – Holds files that vary such as website files.
- /etc – Holds system configuration files.
- /dev – Holds file interfaces for various devices and sockets. For example, when a USB modem is connected via a USB port, interface files are created in the /dev directory that allow users to access the modem.

SOFTWARE PACKAGES

Linux programs are distributed in software packages that may be downloaded and installed as source code from either distributions websites or using a software package manager. Source code packages are usually compressed into an archive file known as a tar file. These files use a program called tar to compress and extract files and directories. Software package managers use online repositories that act as databases containing various software packages. The maintainers of the repository are responsible for keeping program versions up to date. Thus, a package manager may be used to download, install and update software packages [31].

Source code obtained from program websites must be extracted onto the file system and compiled to be installed. Some software packages are designed to run on various platforms and distributions of Linux. Compiling and installing the software on various platforms has been made considerably easier by the introduction of configure scripts and Makefiles. A configure script is run to detect all configuration settings of underlying hardware and software on a device. This enables a software package to customize installation for the specific device it is being installed on. Various options may also be specified when using a configure script to specify user preferences and configuration settings. Configure scripts are found in the source directories of extracted software packages and may be run using the following syntax:

./configure –[option identifier] [option arguments]

Stellenbosch University https://scholar.sun.ac.za

e.g. ./configure –prefix=/usr LDFLAGS='-L/usr –L/lib'

When a configure script has completed, a Makefile is created. The Makefile compiles the software, adding any necessary flags or commands that are specific to the local system. The Makefile is run using the "make" command. Once the software has been compiled, the Makefile is run again with the "install" option. This command then installs the compiled software on the local system. Once software is installed, the Makefile may be run a third time, this time with the "clean" option to remove any unwanted files created in compilation to free some memory [31]. Thus, the following set of commands will install a software package in most cases:

./configure

make

make install

DISTRIBUTION

Many Linux distributions are freely available for use. Three distributions were identified as possible operating systems for the STCD and were compared and tested to find the most suitable distribution.

Ubuntu Linux 12.04 LTS

Ubuntu Linux is a free Linux distribution developed to be used as a desktop distribution with thousands of free applications. Ubuntu is a trusted and well-documented Linux distribution with a large support community [89]. A pre-configured image is available to install Ubuntu Linux on the BeagleBone at [90].

Ubuntu was ruled out as an option for the STCD operating system because the default software packages for Ubuntu are too large. This is because Ubuntu is made for application on desktop computers where memory is abundant, and thus many extra features are included in software packages apart from the essentials. As memory on the STCD is limited, not all required programs can be installed on Ubuntu because of memory limitations.

Ångström Linux

Ångström Linux is a Linux distribution developed specifically for implementation on embedded devices where resources are limited. As Ångström is specifically developed for embedded systems, software packages for the distribution are smaller than those for Ubuntu and the operating system also runs more efficiently.

Ångström was originally chosen as the distribution of choice for the STCD. Development was done on Ångström and the STCD functioned using Ångström. In the late development stages, problems began occurring with the Ångström USB driver. The driver malfunctioned when attempting to connect a 3G USB modem to the STCD device and it was also found that the device often froze

Stellenbosch University https://scholar.sun.ac.za

when utilizing the USB port. The Ångström support community is small, and finding solutions to issues can be tedious. Because of these issues that were identified, it was decided that a more stable distribution should be used.

Arch Linux

The Arch Linux distribution is a distribution that focuses on being lightweight and flexible [14]. Arch Linux is a minimalist distribution with a small core system. The core principles that Arch Linux is built on is: Simplicity, Core-correctness over Convenience, User-Centeredness, Openness and Freedom [91].

The Arch Linux distribution was found to be both light weight and reliable enough to be used as the STCD distribution of choice. It supports all the required packages to achieve the desired functionality of the STCD. Software programs installed and run on the STCD will be described in detail in the following sections of this document.

Stellenbosch University https://scholar.sun.ac.za

APPENDIX C
SNMP-RELATED INFORMATION

C.1 CREATING SHARED LIBRARIES

The GNU compiler is used to create shared library files. This is done by creating a simple Makefile containing the following code:

```
CC = gcc
CFLAGS = -g -Wall -Wundef -O2
all:
        $(CC) $(CFLAGS) -o pegmib.so pegmib.c -fPIC -shared
clean:
        @rm -f *~ *.o *.so
```

This Makefile compiles the desired c files and creates the pegmib.so shared library file that can be included in the Net-Snmp agent. To include this file in the Net-Snmp agent, the following line of code is added to the snmpd.conf file. The snmpd.conf file is the Net-Snmp agent configuration file.

dlmod pegmib /some/path/pegmib.so

The snmpd service is then restarted and the extension code is loaded into the agent

NOTE: If there are errors in the extension code, it will not be loaded into the Net-Snmp agent and no error messages will be given.

C.2 SMI SYNTAX

The SMI syntax is used to define how managed objects are named and specifies their associated data types. SMIv2, defined in RFC 2578 was used to create the PEG-MIB module and will be discussed in this document. An example of a SMIv2 syntax MIB file is given and discussed. This is a stripped-down version of the IF-MIB module contained in RFC 1213 [40].

```
IF-MIB DEFINITIONS ::= BEGIN
IMPORTS
   MODULE-IDENTITRY, OBJECT-TYPE, Counter32, Gauge32, Counter64,
   Integer32, TimeTicks, mib-2,
   NOTIFICATION-TYPE                                FROM SNMPv2-SMI
   TEXTUAL-CONVENTIONm DisplayString,
   PhysAddress, TruthValue, RowStatus,
   TimeStamp                                        FROM SNMPv2-TC

ifMIB MODULE-IDENTITY
   LAST-UPDATED "2000061400002"
   ORGANIZATION "IETF Interfaces MIB Working Group"
   CONTACT-INFO
            "        Keith McCloghrie
                     Cisco Systems, Inc.
                     170 West Tasman Drive
                     San Jose, CA 95134-1706
                     US

                     408-526-5260
                     kzm@cisco.com"
   DESCRIPTION
            "The MIB module to describe generic objects for network
```

Stellenbosch University https://scholar.sun.ac.za


```
                    Interface sub-layers. This MIB is an updated version of MIB-II's
                    ifTable, and incorporates the extensions defined in RFC 1229."
      ::= { mib-2 31 }

   ifMIBObjects          OBJECT INDENTIFIER ::= { ifMIB 1 }

   interfaces   OBJECT IDENTIFIER ::= { mib-2 2 }

   --- scalar variables

   ifDescr      OBJECT-TYPE
      SYNTAX             DisplayString (SIZE  (0..255))
      MAX-ACCESS         read-only
      STATUS             current
      DESCRIPTION
   "A textual string containing information about the interface. This string should
   include the name of the manufacturer, the product name and the version of the
   Interface hardware/software."
      ::= { interfaces 1 }

   END
```

The above code is a MIB file containing a single variable. The first line of code defines the name of the file, in the case above, IF-MIB. The IMPORTS section allows a developer to import data types and OIDs from other MIB files. In the example above, the Counter32 data type and the mib-2 OID are imported from the SNMPv2-SMI file.

The MODULE-IDENTITY section offers a description of the MIB module. Basic information concerning the module as well as contact information of the developer is included here

The actual definition of a managed object uses the following syntax structure.

```
   <name> OBJECT-TYPE
      SYNTAX             <data type>
      UnitParts          <optional, see below>
      MAX-ACCESS         <see below>
      STATUS             <see below>
      DESCRIPTION
                "A description (textual) describing the managed object."
      ::= { <Unique OID of the object> }
```

UnitParts – The measurement unit of the managed object (e.g. seconds, litres etc.)

MAX-ACCESS – Access level that the object can be accessed with. Possibilities are: read-only, read-write, write-only, not-accessible and accessible-for-notify.

STATUS – Possibilities are: current, obsolete and deprecated. Describe how an agent should implement the managed object.

Managed objects may be structured in tables. This groups a set of managed objects together as characteristics of a single entry (for example, an interface with a description, an IP address and a max speed). In the PEG-MIB, tables were not used to maintain simplicity [35].

C.3 NET-SNMP COMMANDS

The Net-Snmp package offers a command line interface with various commands to perform a diversity of tasks. The outline of the interface commands is given:

Stellenbosch University https://scholar.sun.ac.za

<command> -v [SNMP version] –p [SNMP community] [host] [OID if applicable]

Snmpwalk

The snmpwalk command performs a SNMP get-next operation, gathering multiple managed objects and returning their names and values. An example of snmpwalk syntax is described below:

Snmpwalk –v2c –c public localhost

Here, the version is specified as version 2c and the SNMP community access is public. The host is localhost, meaning the local device is being scanned. This command will return all SNMP managed object on the local device.

Snmpget

The snmpget command issues a single get operation with syntax:

Snmpget –v[SNMP version] –c [SNMP community] [host] [object OID]

This command returns the value of the object contained at the specified OID.

Snmpconf

This command is an interactive script that allows a user to create Net-Snmp configuration files [35].

C.4 PEG-MIB SYNTAX

```
PEG-MIB DEFINITIONS ::= BEGIN

          IMPORTS
                              MODULE-IDENTITY, OBJECT-TYPE, Integer32, Unsigned32,
                              Gauge32, Counter32, Counter64, IpAddress, mib-2

                                                        FROM SNMPv2-SMI
                              MODULE-COMPLIANCE, OBJECT-GROUP

                                                        FROM SNMPv2-CONF
                              InetAddress, InetAddressType,
                              InetPortNumber
                    FROM INET-ADDRESS-MIB;

          internet            OBJECT IDENTIFIER ::= { iso(1) org(3) dod(6) 1 }
          enterprises         OBJECT IDENTIFIER ::= { internet(1) private (4) 1 }
          stellenbosch        OBJECT IDENTIFIER ::= { enterprises(1) 713 }

          pegmib MODULE-IDENTITY
                    LAST-UPDATED "200006140000Z"          -- 24-10-2013
                    ORGANIZATION
                                                        "Stellenbosch University Power
Electronics Group"
                    CONTACT-INFO
                                                        "Cornel Verster (editor)

                                                         Electronical Engineering
                                                         Stellenbosch University
                                                         Stellenbosch 7600

                                                         Phone: +27 84 951 2715
                                                         Email:   <verster.cornel@gmail.com>"
                    DESCRIPTION
                    "The MIB module to meet Eskom specifications for remote devices."
                    ::=       { stellenbosch 2 }

-- groups in PEG-MID
```

Stellenbosch University https://scholar.sun.ac.za


```
          mobile              OBJECT IDENTIFIER ::= { pegmib 1 }

          -- the Mobile variables

          mobGsmNetOp                  OBJECT-TYPE
                    SYNTAX             DisplayString
                    MAX-ACCESS         read-only
                    STATUS             current
                    DESCRIPTION
"The name of the GSM Network Operator that the remote unit is currently subscribed to."
                    ::=       { mobile 1 }

          mobNetAttach                 OBJECT-TYPE
                    SYNTAX             DisplayString
                    MAX-ACCESS         read-only
                    STATUS             current
                    DESCRIPTION
                    "The type of network attachment of the remote device (eg. EDGE, HSDPA)."
                    ::=       { mobile 2 }

          mobSigStr           OBJECT-TYPE
                    SYNTAX             Gauge32
                    MAX-ACCESS         read-only
                    STATUS             current
                    DESCRIPTION
                    "The current signal strength of the remote device."
                    ::=       { mobile 3 }

          mobSentBytes        OBJECT-TYPE
                    SYNTAX             Counter32
                    UNITS              "bytes"
                    MAX-ACCESS         read-only
                    STATUS             current
                    DESCRIPTION
                    "The number of sent bytes for the GSM connection."
                    ::=       { mobile 4 }

          mobRecBytes                  OBJECT-TYPE
                    SYNTAX             Counter32
                    UNITS              "bytes"
                    MAX-ACCESS         read-only
                    STATUS             current
                    DESCRIPTION
                    "The number of sent bytes for the GSM connection."
                    ::=       { mobile 5 }

          mobSentPackets      OBJECT-TYPE
                    SYNTAX             Counter32
                    UNITS              "packets"
                    MAX-ACCESS         read-only
                    STATUS             current
                    DESCRIPTION
                    "The number of packets sent for the GSM connection."
                    ::=       { mobile 6 }

          mobRecPackets       OBJECT-TYPE
                    SYNTAX             Counter32
                    UNITS              "packets"
                    MAX-ACCESS         read-only
                    STATUS             current
                    DESCRIPTION
                    "The number of packets received for the GSM connection."
                    ::=       { mobile 7 }

          mobEthPortStatus    OBJECT-TYPE
                    SYNTAX             INTEGER {
                                       up(1),             -- Ethernet port up
                                       down(2)   -- Ethernet port down
                                       }
                    MAX-ACCESS         read-only
                    STATUS             current
                    DESCRIPTION
                    "The status of all Ethernet ports. Either up or down."
                    ::=       { mobile 8 }

          mobNetCode                   OBJECT-TYPE
                    SYNTAX             DisplayString
```

Stellenbosch University https://scholar.sun.ac.za


```
                    MAX-ACCESS          read-only
                    STATUS              current
                    DESCRIPTION
                    "The mobile network code."
                    ::=       { mobile 9 }

          mobLocCode                    OBJECT-TYPE
                    SYNTAX              DisplayString
                    MAX-ACCESS          read-only
                    STATUS              current
                    DESCRIPTION
                    "The mobile location code."
                    ::=       { mobile 10 }

          mobCellTowId                  OBJECT-TYPE
                    SYNTAX                        DisplayString
                    MAX-ACCESS          read-only
                    STATUS                        current
                    DESCRIPTION
                    "The connected mobile cell tower ID."
                    ::=       { mobile 11 }

          mobSIMSecStat       OBJECT-TYPE
                    SYNTAX              INTEGER {
                                        unlocked(1),        -- device is unlocked
                                        pin(2),             -- personal identification number
                                        puk(3)              -- personal unlocking key
                                        }
                    MAX-ACCESS          read-only
                    STATUS                        current
                    DESCRIPTION
                    "The Subscriber Identity Module (SIM) security status."
                    ::=       { mobile 12 }

          mobModImei                    OBJECT-TYPE
                    SYNTAX                        DisplayString
                    MAX-ACCESS          read-only
                    STATUS                        current
                    DESCRIPTION
                    "The modem's International Mobile Equipment Identity (IMEI)."
                    ::=       { mobile 13 }

          mobSimSerial                  OBJECT-TYPE
                    SYNTAX                        DisplayString
                    MAX-ACCESS          read-only
                    STATUS                        current
                    DESCRIPTION
                    "The SIM serial number."
                    ::=       { mobile 14 }

          mobModManufac       OBJECT-TYPE
                    SYNTAX                        DisplayString
                    MAX-ACCESS          read-only
                    STATUS                        current
                    DESCRIPTION
                    "The modem manufacturer."
                    ::=       { mobile 15 }

          mobModModel         OBJECT-TYPE
                    SYNTAX                        DisplayString
                    MAX-ACCESS          read-only
                    STATUS                        current
                    DESCRIPTION
                    "The modem model."
                    ::=       { mobile 16 }

          mobFirmVer                    OBJECT-TYPE
                    SYNTAX                        DisplayString
                    MAX-ACCESS          read-only
                    STATUS                        current
                    DESCRIPTION
                    "The current firmware version."
                    ::=       { mobile 17 }

END
```

Stellenbosch University https://scholar.sun.ac.za

C.5 PEGMIB.H

```
/*
 * Note: this file originally auto-generated by mib2c using
 *        $
 */
#ifndef PEGMIB_H
#define PEGMIB_H

/* function declarations */
void init_pegmib(void);
Netsnmp_Node_Handler handle_mobGsmNetOp;
Netsnmp_Node_Handler handle_mobNetAttach;
Netsnmp_Node_Handler handle_mobSigStr;
Netsnmp_Node_Handler handle_mobNetCode;
Netsnmp_Node_Handler handle_mobLocCode;
Netsnmp_Node_Handler handle_mobCellTowId;
Netsnmp_Node_Handler handle_mobSIMSecStat;
Netsnmp_Node_Handler handle_mobModImei;
Netsnmp_Node_Handler handle_mobSimSerial;
Netsnmp_Node_Handler handle_mobModManufac;
Netsnmp_Node_Handler handle_mobModModel;
Netsnmp_Node_Handler handle_mobFirmVer;

#endif /* PEGMIB_H */
```

C.6 PEGMIB.C

```
/*
 * Note: this file originally auto-generated by mib2c using
 *        $
 */

#include <net-snmp/net-snmp-config.h>
#include <net-snmp/net-snmp-includes.h>
#include <net-snmp/agent/net-snmp-agent-includes.h>

#include <stdio.h>
#include <stdlib.h>

#include "pegmib.h"

FILE *fr;

/** Initializes the pegmib module */
void
init_pegmib(void)
{
          const oid mobGsmNetOp_oid[] = { 1,3,6,1,4,1,713,2,1,1 };
          const oid mobNetAttach_oid[] = { 1,3,6,1,4,1,713,2,1,2 };
          const oid mobSigStr_oid[] = { 1,3,6,1,4,1,713,2,1,3 };
          const oid mobNetCode_oid[] = { 1,3,6,1,4,1,713,2,1,9 };
          const oid mobLocCode_oid[] = { 1,3,6,1,4,1,713,2,1,10 };
          const oid mobCellTowId_oid[] = { 1,3,6,1,4,1,713,2,1,11 };
          const oid mobSIMSecStat_oid[] = { 1,3,6,1,4,1,713,2,1,12 };
          const oid mobModImei_oid[] = { 1,3,6,1,4,1,713,2,1,13 };
          const oid mobSimSerial_oid[] = { 1,3,6,1,4,1,713,2,1,14 };
          const oid mobModManufac_oid[] = { 1,3,6,1,4,1,713,2,1,15 };
          const oid mobModModel_oid[] = { 1,3,6,1,4,1,713,2,1,16 };
          const oid mobFirmVer_oid[] = { 1,3,6,1,4,1,713,2,1,17 };

  DEBUGMSGTL(("pegmib", "Initializing\n"));

          netsnmp_register_scalar(
        netsnmp_create_handler_registration("mobGsmNetOp", handle_mobGsmNetOp,
                               mobGsmNetOp_oid, OID_LENGTH(mobGsmNetOp_oid),
                               HANDLER_CAN_RONLY
        ));
          netsnmp_register_scalar(
        netsnmp_create_handler_registration("mobNetAttach", handle_mobNetAttach,
                               mobNetAttach_oid, OID_LENGTH(mobNetAttach_oid),
                               HANDLER_CAN_RONLY
```

Stellenbosch University https://scholar.sun.ac.za


```
        ));
    netsnmp_register_scalar(
        netsnmp_create_handler_registration("mobSigStr", handle_mobSigStr,
                                mobSigStr_oid, OID_LENGTH(mobSigStr_oid),
                                HANDLER_CAN_RONLY
        ));
          netsnmp_register_scalar(
        netsnmp_create_handler_registration("mobNetCode", handle_mobNetCode,
                                mobNetCode_oid, OID_LENGTH(mobNetCode_oid),
                                HANDLER_CAN_RONLY
        ));
          netsnmp_register_scalar(
        netsnmp_create_handler_registration("mobLocCode", handle_mobLocCode,
                                mobLocCode_oid, OID_LENGTH(mobLocCode_oid),
                                HANDLER_CAN_RONLY
        ));
          netsnmp_register_scalar(
        netsnmp_create_handler_registration("mobCellTowId", handle_mobCellTowId,
                                mobCellTowId_oid, OID_LENGTH(mobCellTowId_oid),
                                HANDLER_CAN_RONLY
        ));
    netsnmp_register_scalar(
        netsnmp_create_handler_registration("mobSIMSecStat", handle_mobSIMSecStat,
                                mobSIMSecStat_oid, OID_LENGTH(mobSIMSecStat_oid),
                                HANDLER_CAN_RONLY
        ));
          netsnmp_register_scalar(
        netsnmp_create_handler_registration("mobModImei", handle_mobModImei,
                                mobModImei_oid, OID_LENGTH(mobModImei_oid),
                                HANDLER_CAN_RONLY
        ));
          netsnmp_register_scalar(
        netsnmp_create_handler_registration("mobSimSerial", handle_mobSimSerial,
                                mobSimSerial_oid, OID_LENGTH(mobSimSerial_oid),
                                HANDLER_CAN_RONLY
        ));
          netsnmp_register_scalar(
        netsnmp_create_handler_registration("mobModManufac", handle_mobModManufac,
                                mobModManufac_oid, OID_LENGTH(mobModManufac_oid),
                                HANDLER_CAN_RONLY
        ));
          netsnmp_register_scalar(
        netsnmp_create_handler_registration("mobModModel", handle_mobModModel,
                                mobModModel_oid, OID_LENGTH(mobModModel_oid),
                                HANDLER_CAN_RONLY
        ));
          netsnmp_register_scalar(
        netsnmp_create_handler_registration("mobFirmVer", handle_mobFirmVer,
                                mobFirmVer_oid, OID_LENGTH(mobFirmVer_oid),
                                HANDLER_CAN_RONLY
        ));
}
int
handle_mobGsmNetOp(netsnmp_mib_handler *handler,
                           netsnmp_handler_registration *reginfo,
                           netsnmp_agent_request_info   *reqinfo,
                           netsnmp_request_info         *requests)
{
    /* We are never called for a GETNEXT if it's registered as a
       "instance", as it's "magically" handled for us.  */

    /* a instance handler also only hands us one request at a time, so
       we don't need to loop over a list of requests; we'll only get one. */
    char out[255];
          char str[255];
          fr = fopen("/etc/snmphook/static.txt", "r");

          while (!feof(fr))
          {
                    fgets(out, sizeof out, fr);
          }

          fclose(fr);

          sscanf(out, "%s %*s %*s %*s %*s %*s", str);

          strncpy(str, str, 5);
```

Stellenbosch University https://scholar.sun.ac.za


```
    switch(reqinfo->mode) {

        case MODE_GET:
            snmp_set_var_typed_value(requests->requestvb, ASN_OCTET_STR,
                                     /* XXX: a pointer to the scalar's data */&str,
                                     /* XXX: the length of the data in bytes */strlen(str));
            break;


        default:
            /* we should never get here, so this is a really bad error */
            snmp_log(LOG_ERR, "unknown mode (%d) in handle_mobSigStr\n", reqinfo->mode );
            return SNMP_ERR_GENERR;
    }

    return SNMP_ERR_NOERROR;
}
int
handle_mobNetAttach(netsnmp_mib_handler *handler,
                          netsnmp_handler_registration *reginfo,
                          netsnmp_agent_request_info   *reqinfo,
                          netsnmp_request_info         *requests)
{
    /* We are never called for a GETNEXT if it's registered as a
       "instance", as it's "magically" handled for us.  */

    /* a instance handler also only hands us one request at a time, so
       we don't need to loop over a list of requests; we'll only get one. */

          char out[255];
          char str[255];
          fr = fopen("/etc/snmphook/static.txt", "r");

          while (!feof(fr))
          {
                    fgets(out, sizeof out, fr);
          }

          sscanf(out, "%*s %s %*s %*s %*s %*s", str);

    switch(reqinfo->mode) {

        case MODE_GET:
            snmp_set_var_typed_value(requests->requestvb, ASN_OCTET_STR,
                                     /* XXX: a pointer to the scalar's data */&str,
                                     /* XXX: the length of the data in bytes */strlen(str));
            break;


        default:
            /* we should never get here, so this is a really bad error */
            snmp_log(LOG_ERR, "unknown mode (%d) in handle_mobSigStr\n", reqinfo->mode );
            return SNMP_ERR_GENERR;
    }

    return SNMP_ERR_NOERROR;
}
int
handle_mobSigStr(netsnmp_mib_handler *handler,
                          netsnmp_handler_registration *reginfo,
                          netsnmp_agent_request_info   *reqinfo,
                          netsnmp_request_info         *requests)
{
    /* We are never called for a GETNEXT if it's registered as a
       "instance", as it's "magically" handled for us.  */

    /* a instance handler also only hands us one request at a time, so
       we don't need to loop over a list of requests; we'll only get one. */

          char out[255];
          fr = fopen("/etc/snmphook/dynamic.txt", "r");

          while (!feof(fr))
          {
                    fgets(out, sizeof out, fr);
          }
```

Stellenbosch University https://scholar.sun.ac.za


```
          int gauge = atoi(out);

    switch(reqinfo->mode) {

        case MODE_GET:
            snmp_set_var_typed_value(requests->requestvb, ASN_GAUGE,
                                     /* XXX: a pointer to the scalar's data */&gauge,
                                     /* XXX: the length of the data in bytes */sizeof(gauge) );
            break;


        default:
            /* we should never get here, so this is a really bad error */
            snmp_log(LOG_ERR, "unknown mode (%d) in handle_mobSigStr\n", reqinfo->mode );
            return SNMP_ERR_GENERR;
    }

    return SNMP_ERR_NOERROR;
}
int
handle_mobNetCode(netsnmp_mib_handler *handler,
                          netsnmp_handler_registration *reginfo,
                          netsnmp_agent_request_info   *reqinfo,
                          netsnmp_request_info         *requests)
{
    /* We are never called for a GETNEXT if it's registered as a
       "instance", as it's "magically" handled for us.  */

    /* a instance handler also only hands us one request at a time, so
       we don't need to loop over a list of requests; we'll only get one. */

          char out[255];
          char str[255];
          fr = fopen("/etc/snmphook/snmpset.txt", "r");

          while (!feof(fr))
          {
                    fgets(out, sizeof out, fr);
          }

          sscanf(out, "%s %*s %*s", str);

    switch(reqinfo->mode) {

        case MODE_GET:
            snmp_set_var_typed_value(requests->requestvb, ASN_OCTET_STR,
                                     /* XXX: a pointer to the scalar's data */&str,
                                     /* XXX: the length of the data in bytes */strlen(str));
            break;


        default:
            /* we should never get here, so this is a really bad error */
            snmp_log(LOG_ERR, "unknown mode (%d) in handle_mobEthPortStatus\n", reqinfo->mode );
            return SNMP_ERR_GENERR;
    }

    return SNMP_ERR_NOERROR;
}
int
handle_mobLocCode(netsnmp_mib_handler *handler,
                          netsnmp_handler_registration *reginfo,
                          netsnmp_agent_request_info   *reqinfo,
                          netsnmp_request_info         *requests)
{
    /* We are never called for a GETNEXT if it's registered as a
       "instance", as it's "magically" handled for us.  */

    /* a instance handler also only hands us one request at a time, so
       we don't need to loop over a list of requests; we'll only get one. */

          char out[255];
          char str[255];
          fr = fopen("/etc/snmphook/snmpset.txt", "r");

          while (!feof(fr))
```

Stellenbosch University https://scholar.sun.ac.za


```
          {
                   fgets(out, sizeof out, fr);
          }

          sscanf(out, "%*s %s %*s", str);

    switch(reqinfo->mode) {

        case MODE_GET:
            snmp_set_var_typed_value(requests->requestvb, ASN_OCTET_STR,
                                     /* XXX: a pointer to the scalar's data */&str,
                                     /* XXX: the length of the data in bytes */strlen(str));
            break;


        default:
            /* we should never get here, so this is a really bad error */
            snmp_log(LOG_ERR, "unknown mode (%d) in handle_mobEthPortStatus\n", reqinfo->mode );
            return SNMP_ERR_GENERR;
    }

    return SNMP_ERR_NOERROR;
}
int
handle_mobCellTowId(netsnmp_mib_handler *handler,
                          netsnmp_handler_registration *reginfo,
                          netsnmp_agent_request_info   *reqinfo,
                          netsnmp_request_info         *requests)
{
    /* We are never called for a GETNEXT if it's registered as a
       "instance", as it's "magically" handled for us.  */

    /* a instance handler also only hands us one request at a time, so
       we don't need to loop over a list of requests; we'll only get one. */

          char out[255];
          char str[255];
          fr = fopen("/etc/snmphook/snmpset.txt", "r");

          while (!feof(fr))
          {
                   fgets(out, sizeof out, fr);
          }

          sscanf(out, "%*s %*s %s", str);

    switch(reqinfo->mode) {

        case MODE_GET:
            snmp_set_var_typed_value(requests->requestvb, ASN_OCTET_STR,
                                     /* XXX: a pointer to the scalar's data */&str,
                                     /* XXX: the length of the data in bytes */strlen(str));
            break;


        default:
            /* we should never get here, so this is a really bad error */
            snmp_log(LOG_ERR, "unknown mode (%d) in handle_mobEthPortStatus\n", reqinfo->mode );
            return SNMP_ERR_GENERR;
    }

    return SNMP_ERR_NOERROR;
}
int
handle_mobSIMSecStat(netsnmp_mib_handler *handler,
                          netsnmp_handler_registration *reginfo,
                          netsnmp_agent_request_info   *reqinfo,
                          netsnmp_request_info         *requests)
{
    /* We are never called for a GETNEXT if it's registered as a
       "instance", as it's "magically" handled for us.  */

    /* a instance handler also only hands us one request at a time, so
       we don't need to loop over a list of requests; we'll only get one. */

          int secStatus = 0;
```

Stellenbosch University https://scholar.sun.ac.za


```
    switch(reqinfo->mode) {

        case MODE_GET:
            snmp_set_var_typed_value(requests->requestvb, ASN_INTEGER,
                                     /* XXX: a pointer to the scalar's data */&secStatus,
                                     /* XXX: the length of the data in bytes */sizeof(secStatus));
            break;


        default:
            /* we should never get here, so this is a really bad error */
            snmp_log(LOG_ERR, "unknown mode (%d) in handle_mobSIMSecStat\n", reqinfo->mode );
            return SNMP_ERR_GENERR;
    }

    return SNMP_ERR_NOERROR;
}
int
handle_mobModImei(netsnmp_mib_handler *handler,
                          netsnmp_handler_registration *reginfo,
                          netsnmp_agent_request_info   *reqinfo,
                          netsnmp_request_info         *requests)
{
    /* We are never called for a GETNEXT if it's registered as a
       "instance", as it's "magically" handled for us.  */

    /* a instance handler also only hands us one request at a time, so
       we don't need to loop over a list of requests; we'll only get one. */

          char out[255];
          char str[255];
          fr = fopen("/etc/snmphook/static.txt", "r");

          while (!feof(fr))
          {
                    fgets(out, sizeof out, fr);
          }

          sscanf(out, "%*s %*s %s %*s %*s %*s", str);

    switch(reqinfo->mode) {

        case MODE_GET:
            snmp_set_var_typed_value(requests->requestvb, ASN_OCTET_STR,
                                     /* XXX: a pointer to the scalar's data */&str,
                                     /* XXX: the length of the data in bytes */strlen(str));
            break;


        default:
            /* we should never get here, so this is a really bad error */
            snmp_log(LOG_ERR, "unknown mode (%d) in handle_mobEthPortStatus\n", reqinfo->mode );
            return SNMP_ERR_GENERR;
    }

    return SNMP_ERR_NOERROR;
}
int
handle_mobSimSerial(netsnmp_mib_handler *handler,
                          netsnmp_handler_registration *reginfo,
                          netsnmp_agent_request_info   *reqinfo,
                          netsnmp_request_info         *requests)
{
    /* We are never called for a GETNEXT if it's registered as a
       "instance", as it's "magically" handled for us.  */

    /* a instance handler also only hands us one request at a time, so
       we don't need to loop over a list of requests; we'll only get one. */

          char str[80] = "124341";

    switch(reqinfo->mode) {

        case MODE_GET:
            snmp_set_var_typed_value(requests->requestvb, ASN_OCTET_STR,
                                     /* XXX: a pointer to the scalar's data */&str,
                                     /* XXX: the length of the data in bytes */strlen(str));
```

Stellenbosch University https://scholar.sun.ac.za


```
            break;


        default:
            /* we should never get here, so this is a really bad error */
            snmp_log(LOG_ERR, "unknown mode (%d) in handle_mobSimSerial\n", reqinfo->mode );
            return SNMP_ERR_GENERR;
    }

    return SNMP_ERR_NOERROR;
}
int
handle_mobModManufac(netsnmp_mib_handler *handler,
                          netsnmp_handler_registration *reginfo,
                          netsnmp_agent_request_info   *reqinfo,
                          netsnmp_request_info         *requests)
{
    /* We are never called for a GETNEXT if it's registered as a
       "instance", as it's "magically" handled for us.  */

    /* a instance handler also only hands us one request at a time, so
       we don't need to loop over a list of requests; we'll only get one. */

          char out[255];
          char str[255];
          fr = fopen("/etc/snmphook/static.txt", "r");

          while (!feof(fr))
          {
                    fgets(out, sizeof out, fr);
          }

          sscanf(out, "%*s %*s %*s %s %*s %*s", str);

    switch(reqinfo->mode) {

        case MODE_GET:
            snmp_set_var_typed_value(requests->requestvb, ASN_OCTET_STR,
                                     /* XXX: a pointer to the scalar's data */&str,
                                     /* XXX: the length of the data in bytes */strlen(str));
            break;


        default:
            /* we should never get here, so this is a really bad error */
            snmp_log(LOG_ERR, "unknown mode (%d) in handle_mobModManufac\n", reqinfo->mode );
            return SNMP_ERR_GENERR;
    }

    return SNMP_ERR_NOERROR;
}
int
handle_mobModModel(netsnmp_mib_handler *handler,
                          netsnmp_handler_registration *reginfo,
                          netsnmp_agent_request_info   *reqinfo,
                          netsnmp_request_info         *requests)
{
    /* We are never called for a GETNEXT if it's registered as a
       "instance", as it's "magically" handled for us.  */

    /* a instance handler also only hands us one request at a time, so
       we don't need to loop over a list of requests; we'll only get one. */

          char out[255];
          char str[255];
          fr = fopen("/etc/snmphook/static.txt", "r");

          while (!feof(fr))
          {
                    fgets(out, sizeof out, fr);
          }

          sscanf(out, "%*s %*s %*s %*s %s %*s", str);

    switch(reqinfo->mode) {

        case MODE_GET:
```

Stellenbosch University https://scholar.sun.ac.za


```
            snmp_set_var_typed_value(requests->requestvb, ASN_OCTET_STR,
                                     /* XXX: a pointer to the scalar's data */&str,
                                     /* XXX: the length of the data in bytes */strlen(str));
            break;


        default:
            /* we should never get here, so this is a really bad error */
            snmp_log(LOG_ERR, "unknown mode (%d) in handle_mobModModel\n", reqinfo->mode );
            return SNMP_ERR_GENERR;
    }

    return SNMP_ERR_NOERROR;
}
int
handle_mobFirmVer(netsnmp_mib_handler *handler,
                          netsnmp_handler_registration *reginfo,
                          netsnmp_agent_request_info   *reqinfo,
                          netsnmp_request_info         *requests)
{
    /* We are never called for a GETNEXT if it's registered as a
       "instance", as it's "magically" handled for us.  */

    /* a instance handler also only hands us one request at a time, so
       we don't need to loop over a list of requests; we'll only get one. */

          char out[255];
          char str[255];
          fr = fopen("/etc/snmphook/static.txt", "r");

          while (!feof(fr))
          {
                    fgets(out, sizeof out, fr);
          }

          sscanf(out, "%*s %*s %*s %*s %*s %s", str);

    switch(reqinfo->mode) {

        case MODE_GET:
            snmp_set_var_typed_value(requests->requestvb, ASN_OCTET_STR,
                                     /* XXX: a pointer to the scalar's data */&str,
                                     /* XXX: the length of the data in bytes */strlen(str));
            break;


        default:
            /* we should never get here, so this is a really bad error */
            snmp_log(LOG_ERR, "unknown mode (%d) in handle_mobFirmVer\n", reqinfo->mode );
            return SNMP_ERR_GENERR;
    }

    return SNMP_ERR_NOERROR;
}
```

Stellenbosch University https://scholar.sun.ac.za

APPENDIX D
SRAPET CONVERSATIONAL PROTOCOL

D.1 INTRODUCTION

The SRAPET Conversational Protocol (SCP) is a bi-directional communication protocol used for serial communication between the STCD and the SRAPET transformer.

SCP is a text based protocol with message formats that can easily be read and understood by humans, making it easier to develop and debug. It expects commands from sources external to the SRAPET. Once a command is received, the SRAPET replies with a status reply and possibly a data response containing desired data. The concept is illustrated in Figure D-1.

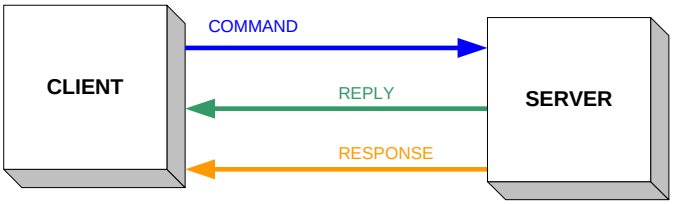


Figure D-1 - SCP Messaging

The status reply contains information on whether the incoming command was successfully executed while the data response contains desired data if it is available.

D.2 SYNTAX

To ensure interoperability between various operating systems and programming languages, plain text is kept to as far as possible for SCP command and response / reply message syntax. In the following description:

- < > indicates a field that is required and must be present.
- [ ] indicates a field that is optional and may be omitted.

D.2.1 Commands

A command is a case insensitive text string sent from a client to the SRAPET that either requests information from the SRAPET or that some task be performed by the SRAPET.

Commands are terminated by the carriage return <CR> and new line feed <LF> characters. If data is required to be sent along with a command, the data follows on a new line after the command. The end of the data is indicated with a “.” character. Command syntax is shown in .

Table D-1 - SCP Command Syntax

	Description
Syntax:	a) Command with required and optional parameters <cmd par> <par1> [par2] [par3] <CR><LF> b) Command with optional parameters <cmd par> <par1> [par2] [par3] <CR><LF> c) Command with data <cmd par> <par1> <CR><LF> <data> <CR><LF> . <CR><LF>
Example:	RD time -T<CR><LF> This command requests the current time from the SRaPET. The command parameter is ”RD”, while there is a required parameter / key, in this case “time”, which indicate the specific data to be requested. The optional “-T” option indicates that he response must be in plain text.

D.2.2 Reply

The reply is a text string that always contains status information about the success or failure of the requested command, and can contain small amounts of data.

The reply consists of a numeric status code and a description of the status code. Status codes ending in 00 (100, 200 etc.) are reserved for general replies without a special meaning such as “OK”. Table D-2 shows the reply codes that are linked to various functions.

Table D-2 - Reply Code Ranges

Reply Code Rage	Function (as used in standard text protocols)	Function (as used in SCP procotol)
1xx	Informative codes	Informative codes
2xx	Success	Success
3xx	Temporary error	Success
4xx	Permanent error	Error
5xx	Internal error	Error
6xx	-	Events

The reply message syntax is shown in Table D-3.

Table D-3 - Reply Syntax

	Description
Syntax:	Single line reply <reply status code> [description] <CR><LF> . <CR><LF> Multi line reply <reply status code>-[description] <CR><LF> "-" Indicate multi line [<reply status code>-[message]] <CR><LF> Optional line <reply status code> [message] <CR><LF> " " Indicate last line . <CR><LF>
Example (single line):	104 GSM will be reset <CR><LF> . <CR><LF> This is a single line reply, informing the user that the reset GSM module command will be executed. The reply status code "104" is associated with the "reset gsm" command, while the "GSM will be reset" message describes the status code in plain text to quickly grasp the meaning of the reply
Example (multi line):	207-OK [read time] <CR><LF> 207 8 bytes <CR><LF> This is a multi line reply. Note that each line starts with the reply status code followed by a "-", except for the last line where the reply status code is followed by a space. (There is no "." at the end, because a read time command also have a response, which is not shown here.)

D.2.3 Response

The response follows the reply and contains data that is required by a received command. The response contains either text or binary data and is terminated by a single period on a line by itself. Response message syntax is shown in Table D-4.

Table D-4 - Response Syntax

	Description
Syntax (text):	<text response: line#1><CR> <text response: line#2><CR> <text response: final line><CR><LF> . <CR><LF>
Syntax (binary):	<response - binary><CR><LF> . <CR><LF>

D.2.4 Applicable Commands for the STCD

Only some of the SCP messages were used by the STCD to communicate with the SRAPET controller. The applicable commands and their relevant responses are listed below.

Stellenbosch University https://scholar.sun.ac.za

Table D-5 - StartupInfo Command

Command / Response	Description
RD startupinfo	Command to retrieve SRAPET information required by STCD to interpret data sent from SRAPET to STCD.
295-OK [read startup info]	Relevant SRAPET information will be returned in binary format.

Table D-6 - Stream Start Command

Command / Response	Description
STREAM start	Command for SRAPET to start streaming measurement data to the STCD.
296 OK [streaming started]	Indicates streaming of data from SRAPET to STCD will commence.
455 STREAM ERR [retrieve startup info]	Indicates that information required by STCD has not been retrieved from SRAPET. This must be done before data streaming can commence.

Table D-7 - Stream Stop Command

Command / Response	Description
STREAM stop	Command for SRAPET to stop streaming measurement data to the STCD.
298 OK [streaming stopped]	Indicates streaming of data from SRAPET to STCD will cease.

Stellenbosch University https://scholar.sun.ac.za

Table D-8 - Stream Interval Command

Command / Response	Description
STREAM interval <number>	<number> is replaced with the number of desired milliseconds between successive measurement data messages. This command sets the interval time that the SRAPET waits between messages when streaming data.
298 OK [stream interval set]	Indicates the stream interval has been successfully changed.
457 STREAM ERR [invalid number]	Indicates an invalid number was entered as the desired streaming interval.
458 STREAM ERR [interval too short]	Indicates the desired stream interval is too short (less than 100 ms)
459 STREAM ERR [interval too long]	Indicates the desired stream interval is too long (more than 60 000 ms)

Data returned for the *RD startupinfo* and *STREAM start* commands are in binary format. The response message contains the relevant data in data structures. The data structures for SRAPET device information (retrieved with *RD startupinfo*) and SRAPET measurement streaming data are given below.

Table D-9 - SRAPET Device Information Structure

Structure	Description
struct {	
Uint16 deviceId;	- Device ID number
Uint16 numPhases;	- Number of phases
Uint32 timeOff;	- Time power was interrupted (Seconds passed since “1970/01/01 00:00:00)
Uint32 timeOn;	- Time power was turned on (Seconds passed since “1970/01/01 00:00:00)
Uint32 powerOffReason;	- Reason for power interruption.
} bbStartupInfoT;	

Stellenbosch University https://scholar.sun.ac.za

Table D-10 - SRAPET Streaming Data Structure

Structure (single phase)	Structue (dual phase)	Description
struct	*struct*	
{ float vsA;	{ float vsA;	- A: Supply / input voltage (V)
float vlA;	float vlA;	- A: Output voltage (V)
float ilA;	float ilA;	- A: Output current (A) - low scale
float iloA;	float iloA;	- A: Output current (A) - high scale
	float vsB;	- B: Supply / input voltage (V)
	float vlB;	- B: Output voltage (V)
	float ilB;	- B: Output current (A) - low scale
	float iloB;	- B: Output current (A) - high scale
float iEarth_mA;	float iEarth_mA;	- Earth current (mA)
float iEarthMax_mA;	float iEarthMax_mA;	- Max earth current over 10ms during current measurement interval (mA)
float tint;	float tint;	- Internal ambient temperature (°C)
float text;	float text;	- External ambient temperature (°C)
float ths1;	float ths1;	- Heat sink temperature #1 (°C)
float ths2;	float ths2;	- Heat sink temperature #2 (°C)
float toil;	float toil;	- Estimated top oil temperature (°C)
float thot;	float thot;	- Estimated hot spot temperature (°C)
float ps5V;	float ps5V;	- Power supply voltage (V)
float lifeLossRate;	float lifeLossRate;	- Life loss rate during current measurement interval
float lifeLoss;	float lifeLoss;	- Accumulated life loss since installation (h)
Uint32 tapHist[2];	Uint32 tapHist[2];	- Record of tap changes made during current measurement interval (can record up to 16 tap changes).
} measT;	} measT;	

Stellenbosch University https://scholar.sun.ac.za

APPENDIX E
WEB SERVICES

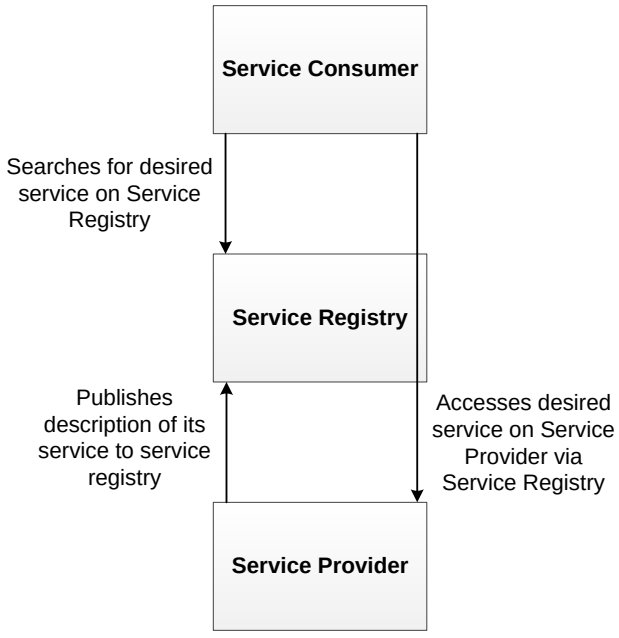
Web services are interfaces that allow the accessing of applications over a network. Applications can be hosted at any accessible location. Web services use standard internet technologies and protocols to access applications. A common example of a web service is an HTTP website. The website is hosted on a remote computer and can be accessed using a web browser.

Web services provide a platform-independent way of accessing applications. Thus, an application written entirely in Java may be accessed and utilized by a program written entirely in C++. It provides a common “language” that applications on a network can access one another by.

The goal of web services is the total modularization of the distributed computing environment. This means that essentially applications are built mainly from existing parts of other applications in the form of web services.

Web services separate functionality from platform. Thus, a functionality can be developed and implemented on a variety of platforms. This is a benefit as platforms are rapidly advancing as new technologies are developed, while desired functionalities remain constant for longer periods of time.

Web service architecture consists of three major role-players: the *service provider, service consumer* and *service registry.* The service provider publishes a description of the web service that it offers to the service registry. The service consumer then searches the service registry for a desired web services. Once found, the service consumer accesses the web service located at the service provider via the service registry [92]. The concept is illustrated in Figure E-1.



Stellenbosch University https://scholar.sun.ac.za

Figure E-1 - Web Services Architecture

5.2.1 Web Service Technology Stack

The web service architecture previously discussed is implemented through a five-tier technology stack. The layers of the stack resemble that of the well-known TCP/IP stack. This is to be expected as web services are built upon existing internet technologies. The stack with its five layers is illustrated in Figure E-2.

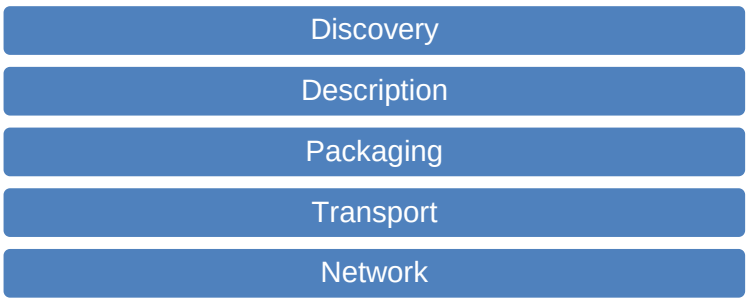


Figure E-2 - Web Services Technology Stack

Each layer of the stack is briefly discussed below.

Discovery

The discovery layer employs a mechanism for consumers to effectively obtain the correct services from providers. This is done by allowing the consumer to fetch descriptions of web services from providers. The discovery mechanism that is most used is known as Universal Description, Discovery and Integration (UDDI) [92].

Description

A description of a web service indicates what internet technologies and protocols that web service uses. Thiis describes how the service consumer is to contact and use the web service. The Web Service Description Language (WSDL) is the standard for providing web service descriptions. W3C's Resource Description Framework (RDF) and the DARPA Agent Markup Language (DAML) are alternatives to WSDL [92].

Packaging

Web service data must be packaged into a format understandable by all parties before being transported. XML forms the basis of most packaging done for web services. Simple Object Access Protocol (SOAP) is a packaging format built upon XML that is commonly used in web services [92].

Transport

The transport layer consists of the protocols used to transport web service data. These protocols are internet protocols such as HTTP, TCP and Jabber. The transport layer's main functionality is the transportation of data from one point to another on a network. Web services may be

Stellenbosch University https://scholar.sun.ac.za

implemented on almost any transport protocol. The transport protocol used may be application-specific as various transport protocols have different benefits [92].

Network

The network layer provides the basic communication, routing and addressing capabilities necessary for the transportation of web service data [92].

5.2.2 XML and SOAP

5.2.2.1 XML

XML is a document format used to represent data. XML messaging is a messaging technique where application communicate by using XML documents. An XML document allows a user to structure data in any desired way.

```
<person>
        <name>John</name>
        <surname>Samson</surname>
        <age>33</age>
        <birthdate>25/07/1981</birthdate>
</person>
```

Figure E-3 - XML Document Example

In Figure E-3, a person's details is recorded in an XML document. A *person* object with the attributes *name, surname, age* and *birthdate* is created. The user could create any object with any amount of attributes in this way.

When two applications communicate using XML documents, data can be structured in any desired way. Both applications would need to know what to expect to receive in an XML document from the other to communicate effectively. Thus, a need for standardization arises when defining the structure of XML documents used for messaging. The Simple Object Access Protocol (SOAP) does exactly this [92].

5.2.2.2 SOAP

The Simple Object Access Protocol (SOAP) is a technique of structuring XML documents for use in web service messaging. A SOAP message consists of a SOAP envelope, containing a header and a body. The header contains information pertaining to how the message is to be processed while the body contains the actual message to be sent.

SOAP is both language- and platform neutral and focuses on being an interoperable protocol. SOAP aims to make web services available on all applicable devices while remaining a lightweight protocol, ideal for embedded applications. A SOAP message can be sent over a variety of transport protocols, but HTTP is preferred. SOAP messages can also pass through a variety of nodes before reaching its final destination. Each node can reroute the message accordingly and the transport protocol used between various nodes need not be the same [93]

Stellenbosch University https://scholar.sun.ac.za

SOAP messages are used in a request-return philosophy. A client sends a message to a server requesting some data, the server then returns the requested data if it is available. SOAP messages are used in a variety of applications: purchase orders, acquiring the current price of a certain stock, acquiring the current temperature etc. [92]. In this project, SOAP messages were used to communicate between the various agents in a microgrid.

The Web Service Description Language (WSDL) is an XML format used to describe web services. WSDL describes the location of the web service and the operations that the web service makes available for use. WSDL can also be used to generate code for implementation in web service server and client programs.

5.2.2.3 Web Service Servers and Clients

In SOAP messaging, one node acts as a server that hosts a web service and allows connections from other nodes. The nodes that connect to the server are known as clients. A client sends a SOAP message to the server requesting data. The server interprets the request message to the best of its ability and sends a response message back to the client containing the requested data.

A web service runs as a service at a specific location on a server node or domain e.g. http://mydomain.com/webservices/someservice/. The client sends a message to this address to access the web service.

5.2.3 GSOAP

The gSOAP toolkit is a C / C++ toolkit for developing SOAP web services. The toolkit auto-generates C / C++ code for the creation of web service servers and clients. The toolkit offers technology that translates WSDL service descriptions to C / C++ definitions and vice versa. This means that WSDL XML documents can be used to create C / C++ code and conversely code can produce XML [93].

The gSOAP toolkit can create a C / C++ header file from a WSDL document using the *wsdl2h* WSDL parser program. It can conversely create a WSDL document from a C / C++ header file using the *soapcpp2* compiler program. The soapcpp2 program can then use the C / C++ header file to generate code for creating web service server and client programs.

To use the gSOAP toolkit, a C++ header file is created that contains a description of functions that the web service will implement and some information specific to gSOAP. An example (load.h) is given below:

```
#include <iostream>
#include <stdlib.h>

//gsoap ls service name:         load Load server service
//gsoap ls service style:        rpc
//gsoap ls service encoding:     encoded
//gsoap ls service namespace:    http://146.232.222.220/cgi-bin/load.wsdl
//gsoap ls service location:     http://146.232.222.220/cgi-bin/loadserver.cgi
```

Stellenbosch University https://scholar.sun.ac.za


```
//gsoap ls schema namespace:     urn:load

//gsoap ls service method-documentation: bid met by generator
int ls__bidMetGenerator(int transact, std::string datetime, double watts, double timespan,
double tariff, double *result);

//gsoap ls service method-documentation: bid met by utility
int ls__bidMetUtility(int transact, std::string datetime, double watts, double timespan,
double tariff, double *result);
```

The comments included in the above code are instruction for the gSOAP toolkit. "*ls*" specifies the desired namespace that the web service will use. The above web service is the load agent server web service in the microgrid market simulation. The web service server possesses two functions: bidMetGenerator and bidMetUtility. These functions belong to the ls namespace.

The above file (load.h) can be used by gSOAP to generate code necessary to implement the web service server and client programs that communicate with it. The following Linux command is issues to generate generic web service code:

soapcpp2 load.h

The gSOAP program then generates .wsdl, .nsmap, .cpp, .h and .xml files necessary to implement the web service. To generate code specifically for client programs, the following command is used.

soapcpp2 -C -i load.h

The command generates the soaploadProxy.cpp and soaploadProxy.h files necessary to implement web service clients. Similarly, the following command is used to generate code to implement the web service server program:

soapcpp2 -S -i load.h

The framework code can then be used to generate the web service server program and multiple web service client programs.

Stellenbosch University https://scholar.sun.ac.za

APPENDIX F
SETTING UP A REMOTE DATA SOURCE

CHANGING THE MYSQL CONFIGURATION ON THE REMOTE DEVICE

The MySQL *my.cnf* configuration file in both Debian and Archlinux can be found in the */etc/mysql/* directory. This file contains all of the MySQL configuration settings for the instance running on the device. The configuration file contains the following line:

bind-address = [ip_address];

This line defines what interface MySQL will listen on for incoming connections. Thus, the value of bind-address should be set to the IP address of the local device.

OPENING PORT 3306

The *iptables* program is used to open and close ports in Linux. Port 3306 is the default network port that MySQL utilizes. To open port 3306 on a remote device, the following command is used:

Iptables –A INPUT –i [interface] –s [host] –p tcp –destination-port 3306 –j ACCEPT

Where [host] is the hostname of the server to which the port should be opened (in our case the IP address of the PI Server) and [interface] is the desired interface over which to open the port (in our case eth0 for Ethernet)

To check the status of network port 3306, the following command can be used:

netstat –tulpn | grep :3306

GRANTING MYSQL PERMISSIONS TO USERS ON OTHER COMPUTERS

To grant privileges to users that log onto a MySQL database from another computer, the *mysql.user* table within the MySQL database must be edited. To grant all privileges to a user called *root* on a host with IP address *146.232.222.216* with password *password*, the following two commands are used in MySQL (note that this may only be done when using the root account).

*GRANT ALL PRIVILEGES ON *.* TO 'root'@'146.232.222.216' IDENTIFIED BY 'password';*

FLUSH PRIVILEGES;

To allow connections from any host, the IP address above can be replaced by a % symbol. This is the MySQL wildcard symbol.