Aukusti Ojala

TCP/IP-Based SMS Alarm Plug-in for SCADA System

Metropolia University of Applied Sciences

Bachelor of Engineering

Electrical and Automation Engineering

Bachelor’s Thesis

19 May 2024

Abstract

Author:	Aukusti Ojala
Title:	TCP/IP-Based SMS Alarm Plug-in for SCADA System
Number of Pages:	49 pages + 2 appendices
Date:	19 May 2024
Degree:	Bachelor of Engineering
Degree Programme:	Electrical and Automation Engineering
Professional Major:	Automation Engineering
Supervisors:	Matti Välikylä, Senior Lecturer
	Teemu Hirvonen, Project Manager

The goal of this thesis work was to develop a TCP/IP-based SMS alarm plug-in for Hitachi Energy's MicroSCADA control system. The currently used serial SMS plug-in puts restrictions as to where the SMS modem can be physically located and cannot be used in a virtualized MicroSCADA environment. TCP/IP-based SMS alarm plug-in would solve these issues.

The development of the new plug-in began by investigating the currently used plug-in's features. After this, a SMS device was chosen. The SMS device of choice was Teltonika Networks' TRB140 gateway device. Modbus TCP was chosen as the communication protocol as both TRB140 and MicroSCADA support the communication protocol.

Reading received SMS messages was not a supported Modbus parameter for TRB140. This was remedied by writing a script for TRB140 that allows received messages to be read via Modbus.

TRB140 was configured to communicate with a MicroSCADA demo system, where the rest of the plug-in was developed. The serial SMS plug-in's components were modified to be compatible with TRB140. An installation package and manual were created for deployment of the plug-in.

The result of this thesis work is a TCP/IP-based alarm plug-in for MicroSCADA that supports the most essential features of the previous plug-in. Some features of the previous plug-in were not possible implement. Some bugs were also encountered. These were documented and the possible solutions for these were presented. The plug-in's cybersecurity aspects should be investigated further before the plug-in can be offered to a customer.

Keywords: Alarm systems, MicroSCADA, Modbus TCP, SMS

Tiivistelmä

Tekijä:	Aukusti Ojala
Otsikko:	TCP/IP-pohjainen SMS-hälytyslisäosa SCADA-järjestelmälle
Sivumäärä:	49 sivua + 2 liitettä
Aika:	19.5.2024
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Sähkö- ja automaatiotekniikka
Ammatillinen pääaine:	Automaatiotekniikka
Ohjaajat:	Lehtori Matti Välikylä Projektipäällikkö Teemu Hirvonen

Opinnäytetyön tavoitteena oli kehittää TCP/IP-pohjainen SMS-hälytyslisäosa Hitachi Energyn MicroSCADA-valvomojärjestelmään. Tällä hetkellä käytössä oleva sarjaliikenteellä toimiva lisäosa asettaa rajoituksia sille, missä SMS-modeemi voi fyysisesti sijaita, eikä sitä voida käyttää virtualisoidussa MicroSCADA-ympäristössä. TCP/IP-pohjainen SMS-hälytyslisäosa ratkaisisi nämä ongelmat.

Uuden lisäosan kehittäminen aloitettiin tutkimalla nykyisin käytössä olevan lisäosan ominaisuuksia. Tämän jälkeen valittiin SMS-laite. SMS-laitteeksi valittiin Teltonika Networksin TRB140-yhdyskäytävälaite. Protokollaksi valittiin Modbus TCP, koska TRB140 sekä MicroSCADA tukevat kyseistä protokollaa.

Vastaanotettujen tekstiviestien lukeminen ei ollut tuettu Modbus-parametri TRB140:ssä. Tämä korjattiin luomalla skripti TRB140:lle, joka syöttää vastaanotetut viestit luettavaksi Modbusilla.

TRB140 konfiguroitiin kommunikoimaan MicroSCADA-demojärjestelmän kanssa, jossa kehitettiin loput lisäosasta. Sarjaliikenteellä toimivan lisäosan ohjelmakomponentteja muokattiin yhteensopiviksi TRB140:lle. Uudelle lisäosalle kehitettiin asennusmanuaali ja -paketti lisäosan käyttöönottoa varten.

Opinnäytetyön tuloksena on TCP/IP-pohjainen hälytyslisäosa MicroSCADA:lle, joka tukee edellisen lisäosan tärkeimpiä ominaisuuksia. Joitakin edellisen lisäosan ominaisuuksia ei ollut mahdollista toteuttaa. Lisäksi havaittiin joitakin bugeja. Nämä dokumentoitiin ja niihin esitettiin mahdolliset ratkaisut. Lisäosan kyberturvallisuutta olisi tutkittava laajemmin ennen kuin lisäosaa voidaan tarjota asiakkaalle.

Avainsanat: hälytysjärjestelmät, MicroSCADA, Modbus TCP, SMS

Contents

List of Abbreviations

AC:	Alarm class. An attribute is used to categorize an alarm to 7 categories in MicroSCADA.
ADU:	Application Data Unit. Modbus data packet.
AG:	Alarm generation. An attribute that defines which object values generates the alarm in MicroSCADA.
AL:	Alarm. An attribute that indicates whether alarm is active or not in MicroSCADA.
AR:	Alarm Receipt. An attribute that indicates whether or not the alarm has been acknowledged in MicroSCADA.
ASCII:	American Standard Code for Information Interchange. A character encoding standard.
COM:	Serial communication port interface
CPI:	Communication Programming Interface. Programmable interface used to connect external programs or communication protocols for MicroSCADA.
FTP:	File transfer protocol
HTML:	Hypertext Markup Language
HTTP:	Hypertext Transfer Protocol
I/O:	Input/Output

IA:	Internet address. An attribute used to define IP address or the host name in MicroSCADA.
IX:	Index. An attribute that seperates individual process group objects in MicroSCADA.
LAN:	Local area network
LIN:	Link. An object needed to establish process communication in MicroSCADA.
LN:	Logical name. An attribute defines which process group the object belongs to in MicroSCADA.
MTU:	Master Terminal Unit
NET:	Network communication unit
NOD:	Node. An object needed to establish process communication in MicroSCADA.
OA:	Object address. An attribute that defines the address of a signal within a station in MicroSCADA.
OI:	Object identifier. An attribute used as a descriptive and hierarchical text for an object in MicroSCADA.
OV:	Object value. An attribute that defines the value of an object in MicroSCADA.
OX:	Object Text. An attribute used as a descriptive text for an object in MicroSCADA.
PDU:	Protocol data unit. Part of Modbus message, which contains function code and data.

RS-232:	Recommended Standard 232. Standard for serial transmission.
RTU:	Remote terminal unit.
SCADA:	Supervisory Control and Data Acquisition
SCIL:	Supervisory Control Implementation Language. Programming language used in MicroSCADA.
SMS:	Short message service
SSH:	Secure Shell
STA:	Station. An object used to define a device in MicroSCADA.
TCP/IP:	Transmission Control Protocol / Internet Protocol
UN:	Unit number. The number of the station where an object is found in MicroSCADA.
USB:	Universal Serial Bus
UV:	User Variable. An attribute used as a global application variable in MicroSCADA.
WebUI:	Web user interface
XML:	Extensible Markup Language

1 Introduction

This thesis project was commissioned by Hitachi Energy Finland Oy. In industrial automation, SCADA systems are used to monitor and control processes and to notify the operators of alarms and other critical events. However, as operators are not always monitoring the process display, it is important to redirect SCADA system's alarms and events, for example via SMS or email. This is why a SMS plug-in has been developed for Hitachi Energy's MicroSCADA product family. This plug-in allows MicroSCADA system's alarms to be redirected via SMS to the recipients.

The current plug-in supports a serial SMS modem. The modem is connected to SCADA server's COM port, communicating over RS-232 which limits the transmission length. This puts restrictions as to where the SMS modem can physically be installed. As the server can be located underground where the modem's signal strength can be less than adequate, an alternative solution is needed. Also, as some MicroSCADA systems run in a virtualized environment, there are no physical COM ports available.

The main objective of this thesis project was to develop a TCP/IP-based SMS alarm plug-in for MicroSCADA. This gives the possibility to connect the SMS interface to a MicroSCADA system without having direct physical link, thus making it possible to locate the SMS interface in a more receptive area within the SCADA network. The main functionalities for the new SMS alarm plug-in are as follows:

- The communication between MicroSCADA and SMS interface needs to be a TCP/IP-based communication protocol.
- The alarms generated by the MicroSCADA system need to be sent to the recipients via SMS.
- The recipients need to be able to acknowledge MicroSCADA system alarms remotely via SMS.

Other goals for the thesis were performance evaluation of the new plug-in and making installation packages which are available for engineers. The new plug-in's cybersecurity aspects were also to be evaluated.

The new alarm plug-in was tested and developed in a demo MicroSCADA application. Ideally the new alarm plug-in will have the same functionalities as the serial SMS alarm plug-in and a complete or a near-complete product will be developed that can be offered to a customer. Missing features and bugs were documented.

2 Background

2.1 SMS

Short message service (SMS) is a protocol that is used to send short messages over wireless networks such as 3G, 4G and 5G. This protocol allows for text messages to be 160 characters or 70 characters long depending on the character encoding. [1.] SMS is used in everyday person-to-person messaging and in corporate applications such as business and SMS marketing.

SMS message's data format consists of components such as the actual message, the destination phone number, time stamp and data encoding scheme. Sent SMS messages are stored in a short message service center (SMSC) and are forwarded when the recipient is available in the network. [1.]

2.2 TCP/IP

Transmission Control Protocol/Internet Protocol (TCP/IP) is a protocol suite used to build interconnection of networks. These interconnections allow the communication between hosts on different networks over large geographical areas. TCP/IP provides a common platform of communication services for implementing communication interfaces independent of the underlying physical network. [2, 4-5.]

TCP/IP can be thought as a layer model in which layers communicate with those above and below as presented in figure 1. IP layer provides the ability to transfer data between hosts and TCP layer utilizes IP layer to provide applications with reliable data stream delivery. [2, 6.] The TCP/IP suite is defined with the following layers:

- Application layer. This layer is used to for the user process to access another process either on a different host or to communicate within a single host. Examples of applications include Telnet, File Transfer Protocol or Modbus.
- Transport layer. This layer is responsible for data delivery between applications. In addition to TCP, User Datagram Protocol (UDP) is a commonly used transport protocol.
- Network layer. The most important network protocol, Internet Protocol (IP), is responsible for delivering messages to their destination.
- Network interface layer. This layer is the interface to the actual network hardware. [2, 7-8.]

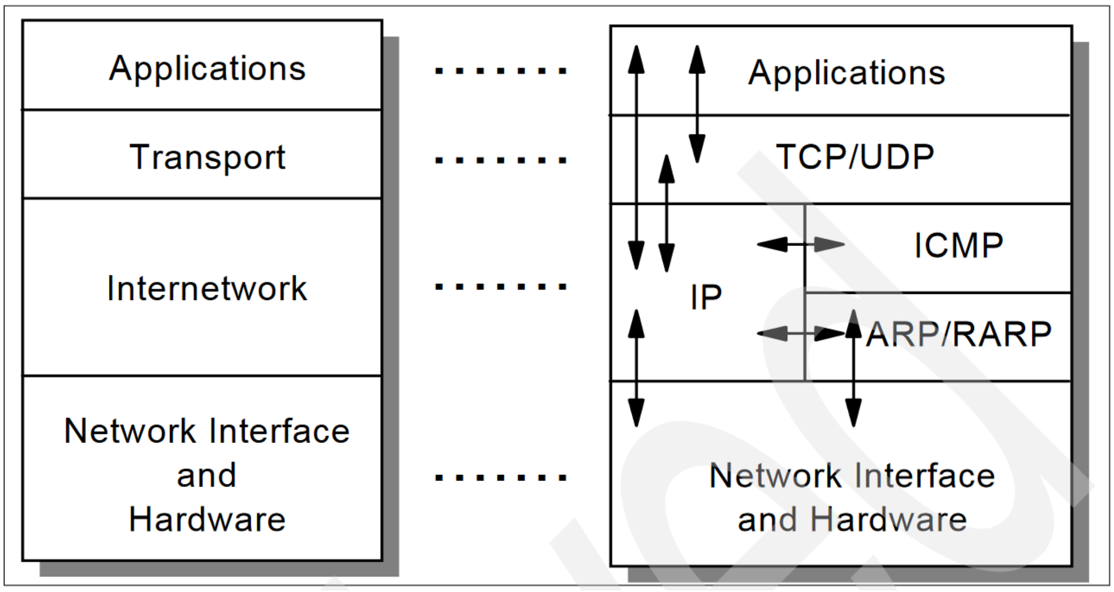


Figure 1. TCP/IP layer relations [2, 7].

Internet Protocol (IP) is a connectionless packet delivery protocol which introduces a virtual network view to hide the underlying physical network. IP addresses are used to identify a host in a network expressed in a dotted decimal form, which can be 127.0.0.1, for example. Another essential feature of IP layer is IP routing, which provides the mechanism to interconnect different physical networks. [2, 68.]

Transmission Control Protocol (TCP) layer provides a reliable connection service between pairs of processes. As lower-level layers, such as IP, cannot guarantee the reliability, TCP introduces concepts such as error recovery, flow control and reliability. [2, 149-150.]

TCP’s window principle, as presented in figure 2, ensures reliability by sending packet and then waiting for an acknowledgement before sending the next packet. If the acknowledgement has not been received within a certain amount of time, the packet is retransmitted. [2, 152.]

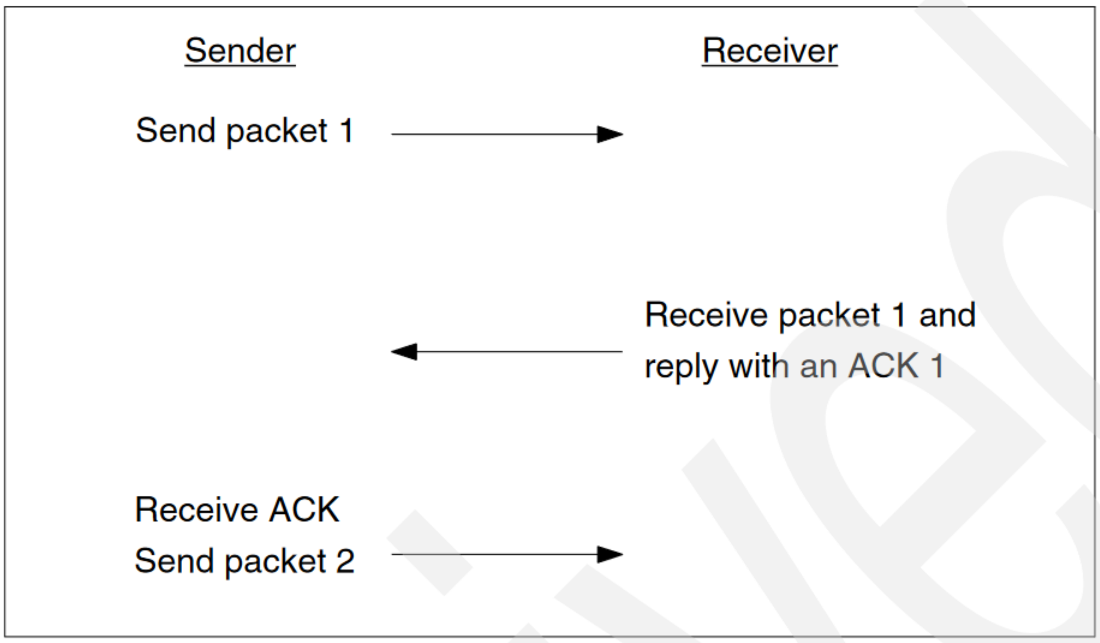


Figure 2. TCP window principle [2, 152].

2.3 Modbus TCP

Modbus TCP communication protocol was chosen as the communication protocol to be used in the new SMS alarm plug-in. This chapter provides an overview of the communication protocol.

Modbus is a client-server application-level communication protocol which can be implemented using TCP/IP over Ethernet or asynchronous serial transmission. Modbus message consists of a protocol data unit (PDU), which is defined by function code, data, and application data unit (ADU). ADU is dependent on which bus or network is used. Client builds the ADU which is then transmitted to server to initiate the wanted action. The client then receives a response from the server which can be the data requested or in case of an error, an exception code. [3, 2-4.]

In case of using Modbus over TCP/IP network, the Modbus request is encapsulated in a TCP packet. This request introduces Modbus application protocol (MBAP) header to the ADU (figure 3). The MBAP header consists of transaction identifier, protocol identifier, length and unit identifier. [4, 4-6.]

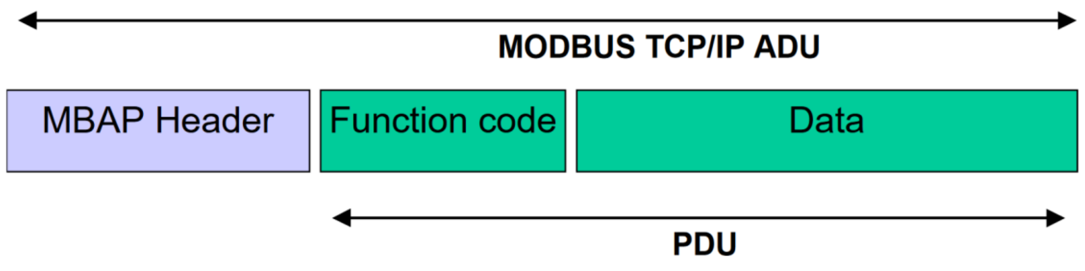
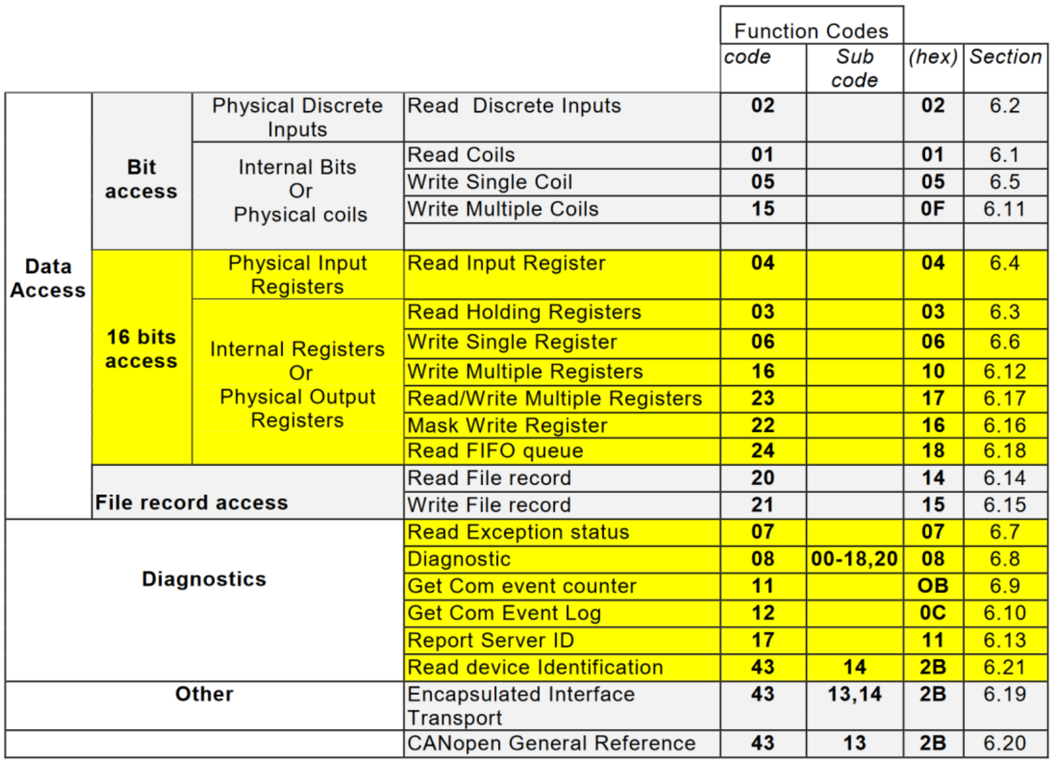


Figure 3. Modbus TCP message structure [4, 4].

Function codes define the type of action to be performed on a server, which can be used for reading and writing registers or coils, for example [4, 3]. Reading holding register is done by using function code 03 as seen in figure 4. This results in the contents of a holding register to be read from the server [3, 15].



				Function Codes			
				code	Sub code	(hex)	Section
Data Access	Bit access	Physical Discrete Inputs	Read Discrete Inputs	02		02	6.2
		Internal Bits Or Physical coils	Read Coils	01		01	6.1
			Write Single Coil	05		05	6.5
			Write Multiple Coils	15		0F	6.11
	16 bits access	Physical Input Registers	Read Input Register	04		04	6.4
		Internal Registers Or Physical Output Registers	Read Holding Registers	03		03	6.3
			Write Single Register	06		06	6.6
			Write Multiple Registers	16		10	6.12
			Read/Write Multiple Registers	23		17	6.17
			Mask Write Register	22		16	6.16
			Read FIFO queue	24		18	6.18
	File record access		Read File record	20		14	6.14
			Write File record	21		15	6.15
Diagnostics			Read Exception status	07		07	6.7
			Diagnostic	08	00-18,20	08	6.8
			Get Com event counter	11		OB	6.9
			Get Com Event Log	12		0C	6.10
			Report Server ID	17		11	6.13
			Read device Identification	43	14	2B	6.21
Other			Encapsulated Interface Transport	43	13,14	2B	6.19
			CANopen General Reference	43	13	2B	6.20

Figure 4. Modbus function codes [3, 11].

Modbus data model can be of four different types which are [3, 6.]:

- Discrete Inputs: read-only single bit data that can be provided by an I/O system.
- Coils: readable and writable single bit data which can be altered by an application prorgram
- Input Registers: read-only 16-bit word data which can be provided by an I/O system
- Holding Registers: readable and writable 16-bit word data which can be altered by an application program.

Modbus protocol specification uses big-Endian representation for addresses and data items which means that the most significant byte is sent first. If a register has a value of 0x1234, 0x12 is sent first, then 0x34 part. [3, 5.]

2.4 Overview of MicroSCADA

2.4.1 SCADA in General

SCADA (supervisory control and data acquisition) is a system used to monitor and control large scale distributed processes in industries such as oil or gas field, electrical transmission. Typical signals monitored from these remote locations include alarms, status indications and analog values. The signals sent from SCADA to the remote locations are usually binary bit changes or analog values. [5, 9-11.]

The major components of a SCADA system are composed of an operator interface, MTUs (master terminal units) and RTUs (remote terminal units) as seen in figure 5. The operator interface or operator console is a graphical interface to the process which the operator controls and monitors. The MTU is a server or computer which does the actual processing of the process control and monitoring. MTU can also have peripheral devices such as printers. MTU communicates with RTUs and perform request messages via a wired or wireless links, which are responsible to collect information from sensors and actuators from field equipment. [5, 11-12.]

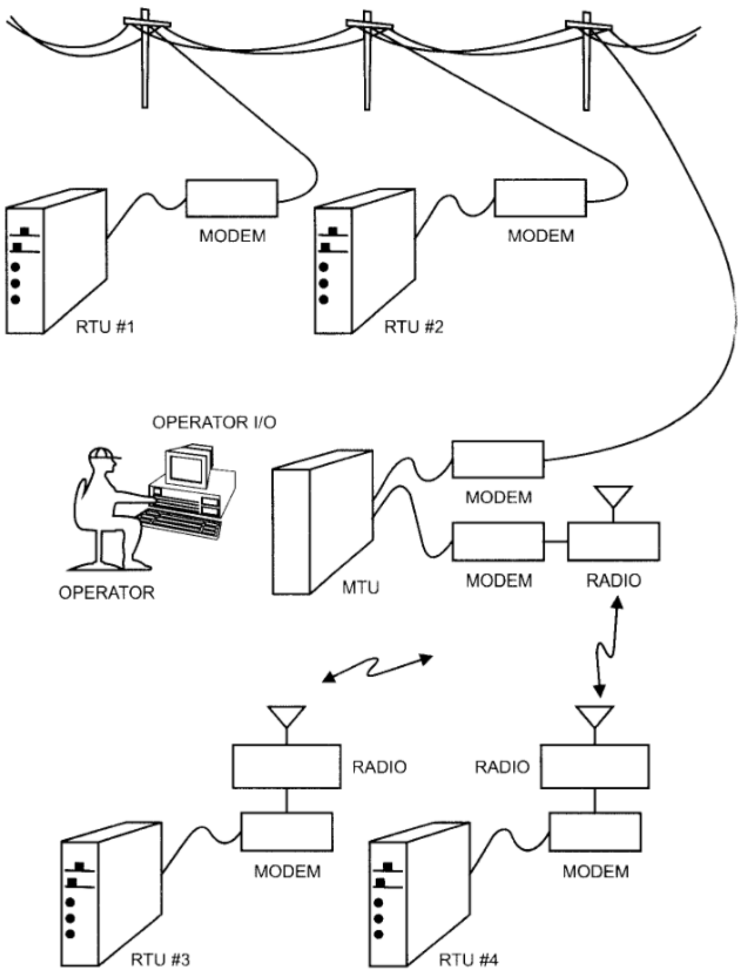


Figure 5. SCADA system's component overview [5, 13].

2.4.2 MicroSCADA in General

The current version of MicroSCADA, also known as MicroSCADA X SYS600, is a microcomputer-based, programmable and distributed SCADA system. It is mainly used for remote and local supervision and control of electrical distribution solutions. [6, 13.] The following chapters provide an overview regarding the MicroSCADA software components.

2.4.3 MicroSCADA System Architechture

A major part of MicroSCADA's functions is handled by a system server which contains base system, a component responsible for central data processing services. It collects all process related data via communication units, which is then distributed for further processing in the MicroSCADA system. Each base system can include one or several applications which are defined by the customer needs and operational functions. Application can consist of process pictures and dialogs, and process and history databases for example. Base system and application have an interconnection, meaning process data can not only be collected, but an operator can send control commands from application to the base system. [7, 10-11.] Figure 6 represents MicroSCADA system components.

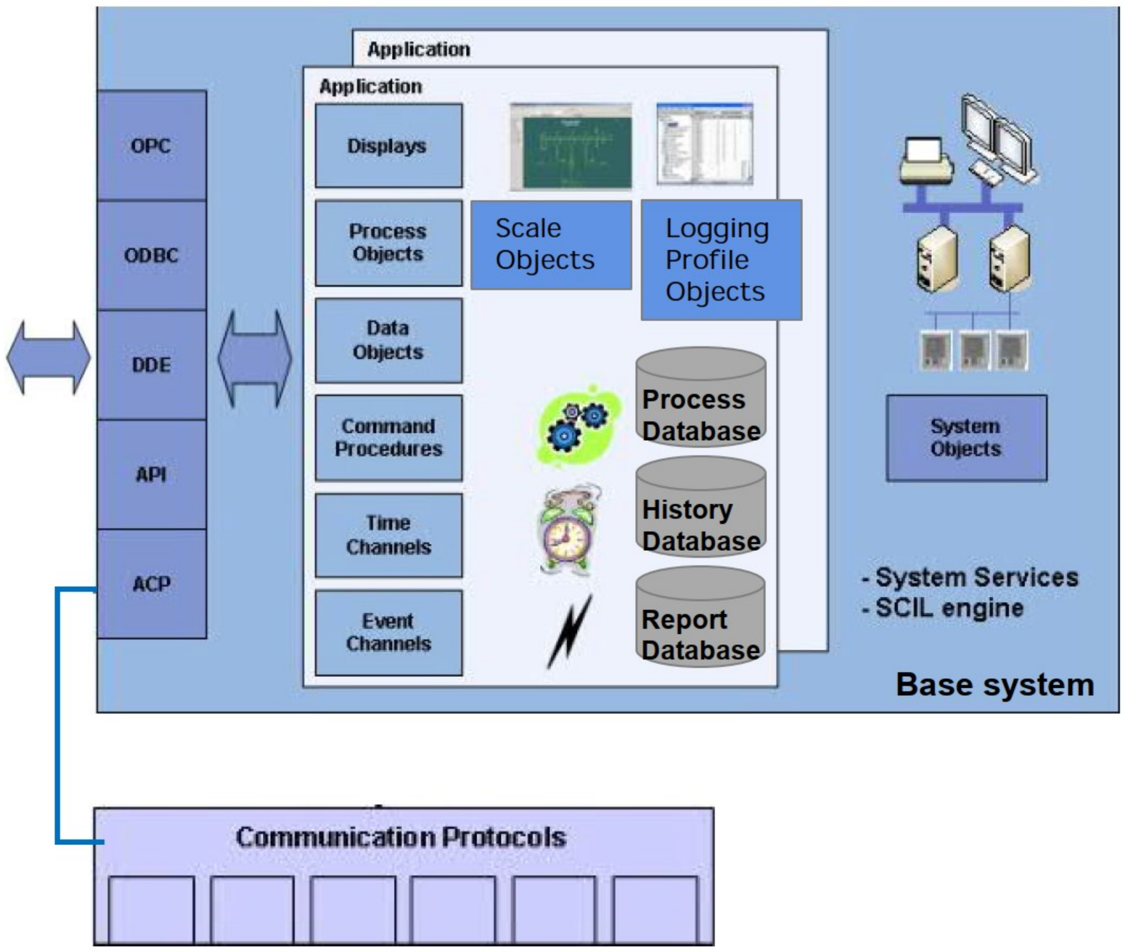


Figure 6. MicroSCADA system components [8].

2.4.4 Objects

MicroSCADA is an object-oriented environment, where objects represent process units, system functions or SCIL programs. Objects are defined by their attributes; properties and information associated with an object. [7, 15.] There are three categories of objects [7, 15-16]:

- Application objects
- System objects
- User interface objects.

Application Objects

Application objects perform tasks, such as process supervision and control, automatic time and event activations, data logging and much more. These objects have an interconnection; they communicate with each other. [9, 23.] There are eleven types of application objects followed by a SCIL object notation in brackets [9, 24.]:

- Process objects (P)
- Event handling objects (H)
- Scales (X)
- Data objects (D)
- Command procedures (C)
- Time channels (T)
- Event channels (A)
- Logging profiles (G)
- Event objects (E)
- Free Type objects (F)
- Variable objects (V).

Most of the process database functionality is based on process objects. Process objects supervise the input and output process signals and can be considered as data images of physical process devices such as disconnectors and relays. Sometimes process objects have no correspondence to a real process and are used for simulation purposes only. The physical process devices are connected to MicroSCADA through devices such as RTUs and relays and are referred as stations or units. [9, 37.]

Process objects have obligatory basic definition attributes which are Logical name (LN) and Index (IX). If the process object also has a correspondence to a real process, it also requires Unit (UN), Object Address (OA). [9, 122.] These four attributes for a real object are:

- LN: This attribute defines which process group the object belongs to.
- IX: This attribute seperates individual process group objects. Individual process objects are identified by indices.
- UN: The number of the station where the process object is found.
- OA: This attribute defines the address of a signal within a station. [9, 44-50.]

Most process objects have attributes which are used to identify the process object in an application. These are called identification attributes and the most important are listed below:

- Object Identifier (OI): This attribute defines a hierarchical information structure and a logical path. It is usually divided into three parts, such as “Substation”, “Bay” and “Device”.
- Object Text (OX): This attribute is a freely chosen text. It is used to identify the signal name, such as “Breaker position indication”. [9, 47-48.]

All Application Objects can be accessed, created and modified with Object Navigator. There are many different use cases for this tool from application engineering perspective, such as listing and searching objects, creating and modifying objects, and exporting and importing objects. [9, 225.]

System Objects

Typical system configuration task is to establish process communication for an application, which is done via System objects. These objects are responsible for handling system configuration and communication to MicroSCADA which are:

- Base system objects: Define the configuration for the base system. These objects consist of physical and logical connections and the software and hardware parameters of the base system and its applications.
- Communication system objects: Define the configuration and communication properties for the process communication system. They have the information which communication protocol is used on the communication line, for example. [10, 25.]

Process communication configuration requires the following steps:

- Configuring communication system objects in base system which are Link (LIN), Node (NOD) and Station (STA).
- Configuring process communication units, for example PC-NET, which handles communication most of MicroSCADA's communication protocols. [10, 58-59.]

System objects and process communication units can be created and configured using System Configuration Tool when establishing process communication. [6, 58-60.]

2.4.5 SCIL

SCIL (Supervisory Control Implementation Language) is an integral part of MicroSCADA as it can be used to control the entire system from application engineering to system configuration and can be used to create and manage all objects [6, 17]. SCIL is a programming language designed specifically for MicroSCADA. Most of the MicroSCADA's pre-existing tools are built on top of SCIL, for example the previously mentioned Object Navigator and System Configuration tool. [11, 25-26.] Figure 7 represents what can be done with SCIL.

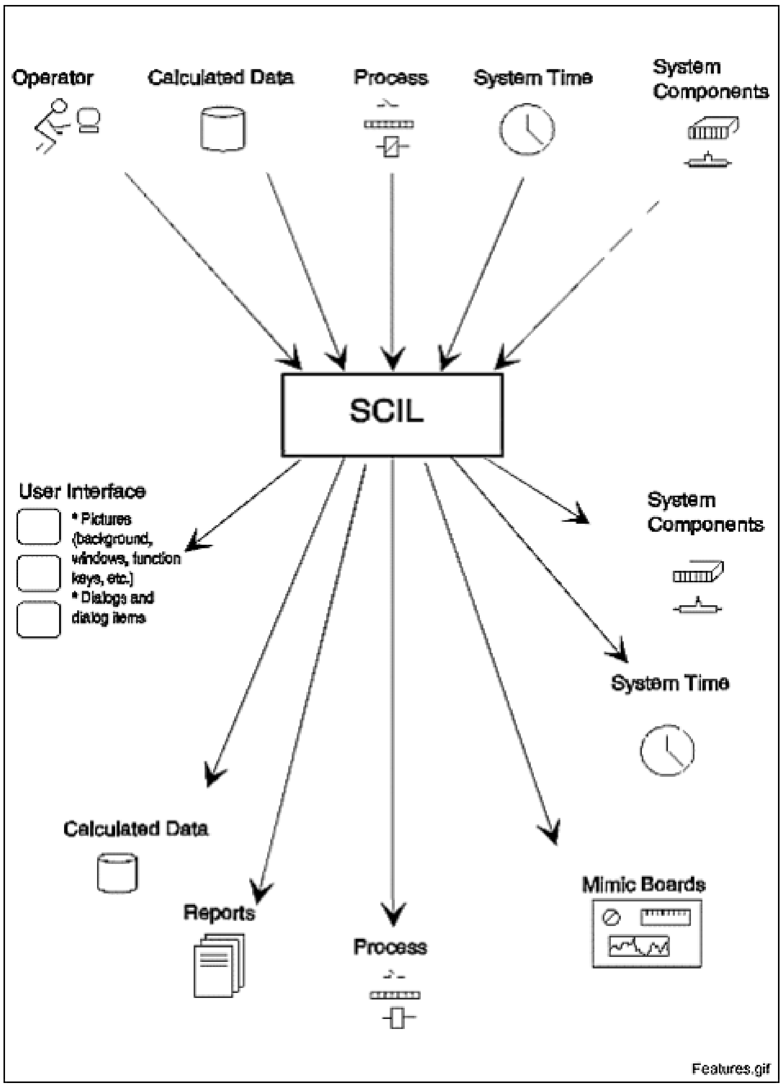
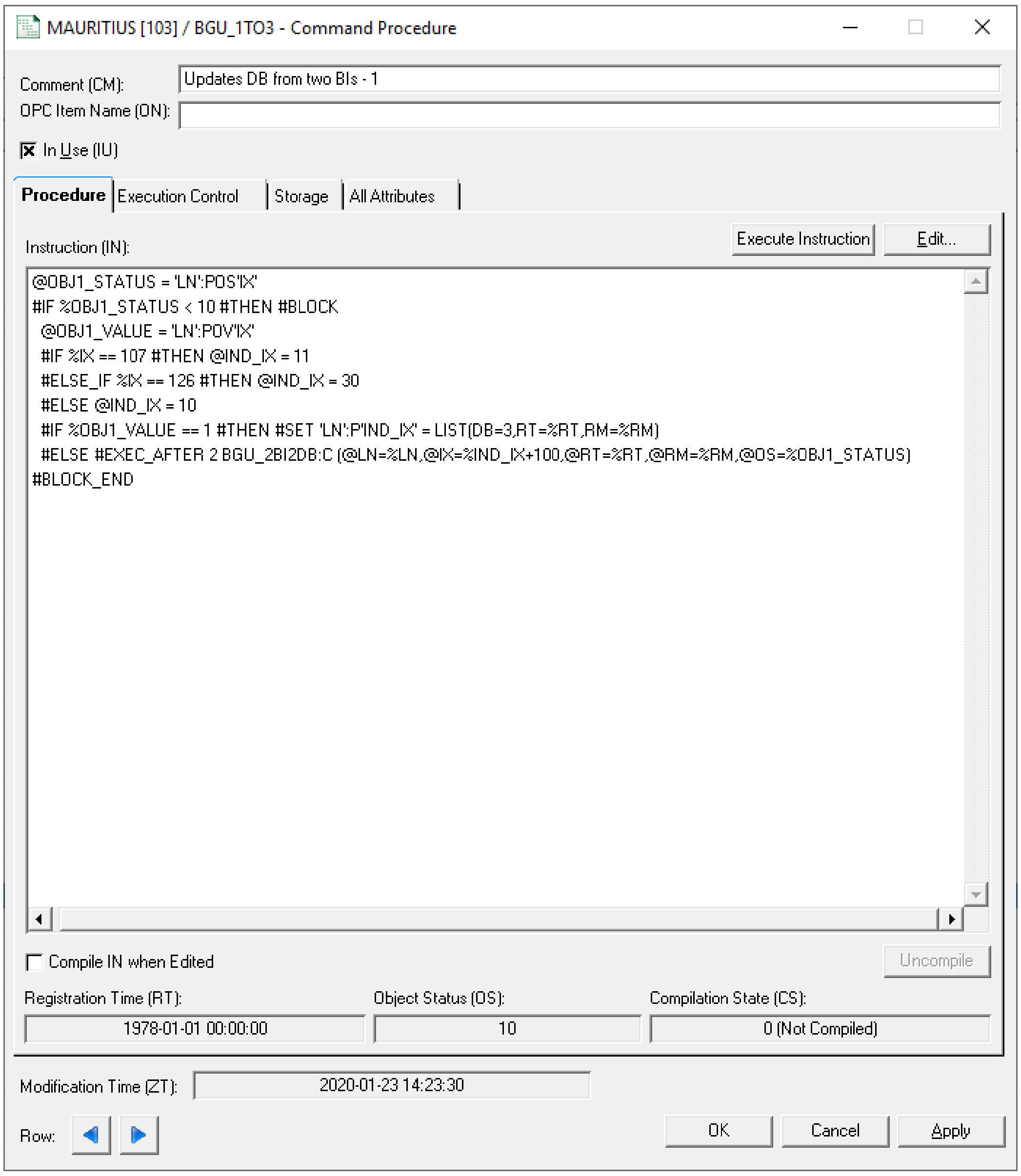


Figure 7. SCIL programming language use cases [11, 27].

SCIL program consists of statements which can be an instruction to a system task or a variable value assignment [11, 27]. SCIL statement may consist of the following components:

- Commands. Orders to the system about steps to be taken.
- Objects. All objects are accessible with SCIL.
- Variables. These are used for temporary storage of data.
- Function calls. SCIL has lots of built-in functions which can be used in string handling, database management and file handling, for example.
- Expressions: formulas which can contain constants, object notations, function calls and operators. [11, 29.]

The most prominent use case for SCIL is in Command Procedures objects. Command Procedure are SCIL programs that can contain instructions for controlling and providing information about an object and its attributes. Command procedures can be activated manually, or manually using Time Channel and Event Channel objects. [9, 163]. Command Procedures are used in establishing process control and logic, and user interfaces, for example. An example of a Command Procedure in presented in figure 8.



Comment (CM): Updates DB from two BIs - 1
OPC Item Name (ON):
[✓] In Use (IU)

Instruction (IN):

```
@OBJ1_STATUS = 'LN':POS'IX'
#IF %OBJ1_STATUS < 10 #THEN #BLOCK
 @OBJ1_VALUE = 'LN':POV'IX'
 #IF %IX == 107 #THEN @IND_IX = 11
 #ELSE_IF %IX == 126 #THEN @IND_IX = 30
 #ELSE @IND_IX = 10
 #IF %OBJ1_VALUE == 1 #THEN #SET 'LN':P'IND_IX' = LIST(DB=3,RT=%RT,RM=%RM)
 #ELSE #EXEC_AFTER 2 BGU_2BI2DB:C (@LN=%LN,@IX=%IND_IX+100,@RT=%RT,@RM=%RM,@OS=%OBJ1_STATUS)
#BLOCK_END
```

☐ Compile IN when Edited

Registration Time (RT): 1978-01-01 00:00:00
Object Status (OS): 10
Compilation State (CS): 0 (Not Compiled)
Modification Time (ZT): 2020-01-23 14:23:30

Figure 8. Example of a Command Procedure. This Command Procedure updates the value of a double binary object from two binary objects.

2.4.6 Alarm Management in MicroSCADA

Alarm is a critical type of event that is prioritized to notify the operator. In MicroSCADA, alarm can have four states:

- Active. The alarm is active but the alarm does not need to be acknowledged.
- Active unacknowledged. The alarm is active but not acknowledged.
- Active acknowledged. The alarm is active and acknowledged.
- Fleeting. The alarm has not been acknowledged after the alarm deactivated. [6, 21-22.]

Alarm is generated either through by process object's object value (OV) or through control supervision. In case the process object is binary data type, the alarm is generated with the following attributes:

- Alarm class (AC): This attribute is used to categorize an alarm to 7 categories, for example according to their severity. AC value 0 disables the alarm handling of the object and no alarm is generated.
- Alarm generation (AG): This attribute defines which object values generate the alarm. Process object can generate an alarm with object either with object value 0 or 1.

When the AC is greater than zero and AG condition is fulfilled, the process object's Alarm (AL) attribute is set to 1 and is shown in the alarm list. Receipt (RC) attribute defines whether an alarm needs an acknowledgement, and the alarm is not removed from the alarm list until it is acknowledged. Alarm is acknowledged with AR (alarm receipt) attribute. [9, 56-62.]

Alarm events can trigger Event Channel objects, which can have programs attached to them used for process control. Event Channel APL_ALARM activates whenever a process object's AL value changes and event channel APL_ALARM_ACK activates whenever object's AR value is set to 1. [9, 187-188.]

2.5 Overview of Serial SMS Alarm Plug-in

Before the new SMS alarm plug-in was developed, the serial SMS alarm plug-in's features were examined as these needed to be implemented in the new plug-in. The current plug-in consists of user interfaces, external programs and initialization files running outside of the MicroSCADA environment, and command procedures. The plug-in supports sending messages and receiving messages but also making outgoing calls and receiving incoming calls, so it is strictly not a SMS alarm plug-in per se.

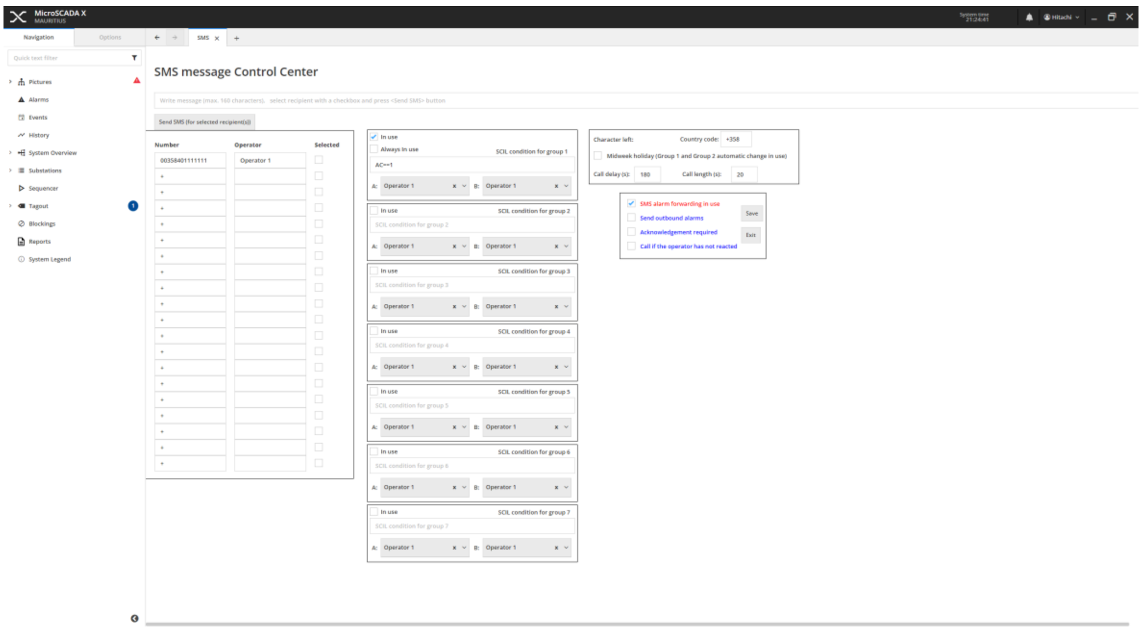
2.5.1 User Interface

There have been developed two dialogs via Visual SCIL (VSO) and the other as a View. The used dialog depends on the used user interface in the MicroSCADA system but functionally they operate in similar ways. The newer user interface designed for Workplace X will be examined.

The dialog reads out initialization data from SCADA server's operating system path. This file's content is defined by the SMS user interface user input. In figure 9, the user interface contains input fields for operator's number and name in the left. Text messages can be sent from the user interface by selecting a check box next to the operator's name, typing a message in the uppermost input field followed by a the press of a ”Send SMS (for selected recipients)” button.

The boxes in the middle contain input fields for an attribute condition that define the type of alarm to be sent and a check box whether an alarm group is in use. The most common SCIL condition is related to AC attribute but can be freely defined by user.

The box on the upper right contains a delay time for an outgoing call if an operator has not acknowledged the received alarm and a time how long the outgoing call will be. This can be activated by checking the ”Receipt required” and ”Call, if the operator has not reacted”, located in the box below. If “Receipt required" is checked, only reminder text messages are sent to the operator and no outbound calls are made. SMS alarm forwarding can also be set enabled or disabled from this box.



SMS message Control Center

Write message (max. 160 characters), select recipient with a checkbox and press <Send SMS> button

Number	Operator	Selected
00358401111111	Operator 1	☐

[✓] In use
☐ Always in use
SCIL condition for group 1: AC==1
A: Operator 1 B: Operator 1

☐ In use
SCIL condition for group 2
A: Operator 1 B: Operator 1

☐ In use
SCIL condition for group 3
A: Operator 1 B: Operator 1

☐ In use
SCIL condition for group 4
A: Operator 1 B: Operator 1

☐ In use
SCIL condition for group 5
A: Operator 1 B: Operator 1

☐ In use
SCIL condition for group 6
A: Operator 1 B: Operator 1

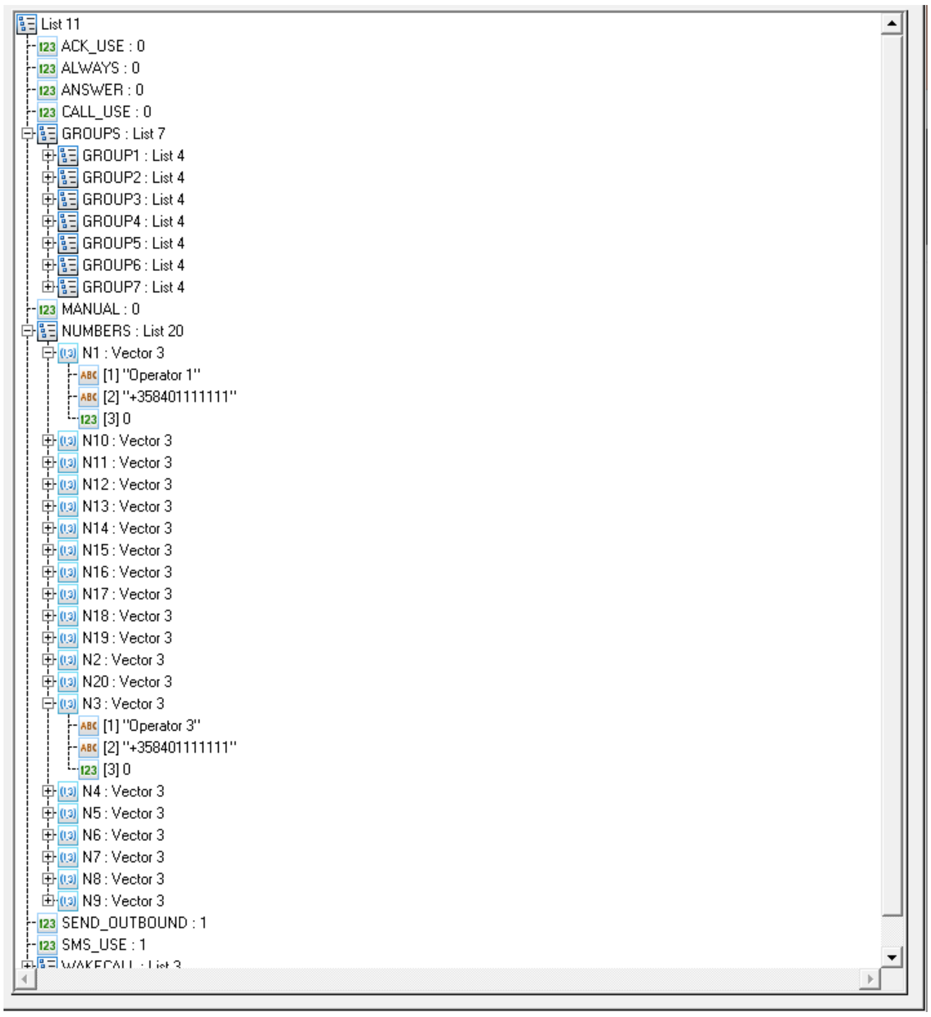
☐ In use
SCIL condition for group 7
A: Operator 1 B: Operator 1

Character left: Country code: +358
☐ Midweek holiday (Group 1 and Group 2 automatic change in use)
Call delay (s): 180 Call length (s): 20

[✓] SMS alarm forwarding in use
☐ Send outbound alarms
☐ Acknowledgement required
☐ Call if the operator has not reacted

Figure 9. SMS plug-in management page for Workplace X.

When save button is pressed, the inputted data is set to a global variable as a SCIL data type of list as shown in figure 10. The data is also saved as a file on the SCADA server's hard drive that serves as the initialization data for the user interface as mentioned in the beginning of the previous chapter.



```
List 11
├ ACK_USE : 0
├ ALWAYS : 0
├ ANSWER : 0
├ CALL_USE : 0
⊟ GROUPS : List 7
│ ⊞ GROUP1 : List 4
│ ⊞ GROUP2 : List 4
│ ⊞ GROUP3 : List 4
│ ⊞ GROUP4 : List 4
│ ⊞ GROUP5 : List 4
│ ⊞ GROUP6 : List 4
│ ⊞ GROUP7 : List 4
├ MANUAL : 0
⊟ NUMBERS : List 20
│ ⊟ N1 : Vector 3
│ │ ├ [1] "Operator 1"
│ │ ├ [2] "+358401111111"
│ │ └ [3] 0
│ ⊞ N10 : Vector 3
│ ⊞ N11 : Vector 3
│ ⊞ N12 : Vector 3
│ ⊞ N13 : Vector 3
│ ⊞ N14 : Vector 3
│ ⊞ N15 : Vector 3
│ ⊞ N16 : Vector 3
│ ⊞ N17 : Vector 3
│ ⊞ N18 : Vector 3
│ ⊞ N19 : Vector 3
│ ⊞ N2 : Vector 3
│ ⊞ N20 : Vector 3
│ ⊟ N3 : Vector 3
│ │ ├ [1] "Operator 3"
│ │ ├ [2] "+358401111111"
│ │ └ [3] 0
│ ⊞ N4 : Vector 3
│ ⊞ N5 : Vector 3
│ ⊞ N6 : Vector 3
│ ⊞ N7 : Vector 3
│ ⊞ N8 : Vector 3
│ ⊞ N9 : Vector 3
├ SEND_OUTBOUND : 1
├ SMS_USE : 1
⊞ WAKECALL : List 3
```

Figure 10. Global variable inspected with test dialog. This variable contains operator data and other options set in the SMS management page.

2.5.2 Technical Features

When a process object generates an alarm event, the process object passes its LN, IX an AL attributes to SMS_ALARM via APL_ALARM event channel. SMS_ALARM command procedure handles the sending of alarm text messages. The command procedure initializes all the data contained in the previously mentioned global variable. The initial alarm text contains OI and OX attributes of the process object. After this, SMS_ALARM adjusts the alarm text depending on the process object's alarm state which is then sent to a recipient via an external program called SMSSend.

If the alarm requires acknowledgement, SMS_ALARM activates a time channel SMS_ACKNOT'I', where 'I' is the alarm group's number, executing a command procedure of the same name. SMS_ACKNOT'I' sends a notification text message if the alarm has not been acknowledged and an outgoing call is made to the recipient. Text message notifications and calls to the recipient or recipients are made until the alarm is acknowledged via an acknowledgement text message or a call.

When a call or text message is sent to the modem, an external program called SMSserv captures the inbound phone number and the inbound text message. Acknowledgement is handled in SMS_USER command procedure. If the phone number exists in the operator list, the command procedure sets the sent process objects' AR attribute to 1. The received phone number can only acknowledge the alarms sent to the alarm group it belongs to.

Alarm acknowledgement triggers APL_ALARM_ACK event channel. This activates a connected SMS_INIT command procedure which resets SMS_ACKNOT'I' time channels and no more notifications messages or calls are made. SMS_INIT also clears an alarm buffer which limits the amount of text messages to be sent in a short period of time to prevent performance issues. Alarm buffer clearing sends the remaining text messages that could not be sent in the initial alarm event. The alarm buffer's size can be adjusted from an initialization file.

There are pre-programmed SMS control commands such as turning the SMS alarm forwarding on or off. Custom control commands are also supported for controlling a breaker from the process database, for example.

This SMS alarm plug-in monitors the state of the modem and MicroSCADA server. If a modem encounters an error, it is indicated in a process object for the user and in a notification window for administrator. The SMS alarm plug-in periodically checks the running status of the MicroSCADA server and informs the user in case of a server or MicroSCADA software failure. All events related to the serial SMS plug-in, such as modem errors, are also logged to an external file.

The SMS alarm plug-in supports redundancy and can be installed in two MicroSCADA servers in a hot standby system. As the other modem is installed to another server, it ensures that SMS alarm plug-in continues to work even though one of the servers or modems fail.

2.6 Teltonika TRB140

Teltonika Networks is an industrial network device manufacturer, focusing on IoT, M2M and enterprise networking solutions. Their product portfolio includes many modems and gateway devices that support sending text messages. [12.] The following device examined in this thesis is Teltonika Networks' TRB140 which was selected to be used as a SMS interface for the new plug-in due to its availability, price, the various options to access SMS interface via TCP/IP supported protocols and the extensive documentation related to the device.

2.6.1 Overview

TRB140 is a gateway device with Ethernet connectivity which runs on a Linux operating system called RutOS. It supports many networking protocols such as HTTP, HTTPS, SSH and FTP which can be used to manage the device. Physical interfaces include input and output sockets, Ethernet port and USB port, SIM card slot and connector for an external antenna. The device can be powered via a dedicated power socket or alternatively via Ethernet port using Power over Ethernet. [13.] The top view of TRB140 is shown in figure 11.

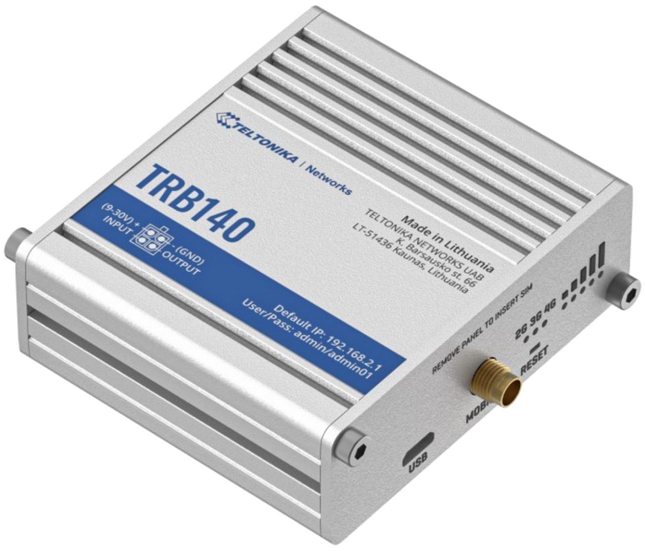
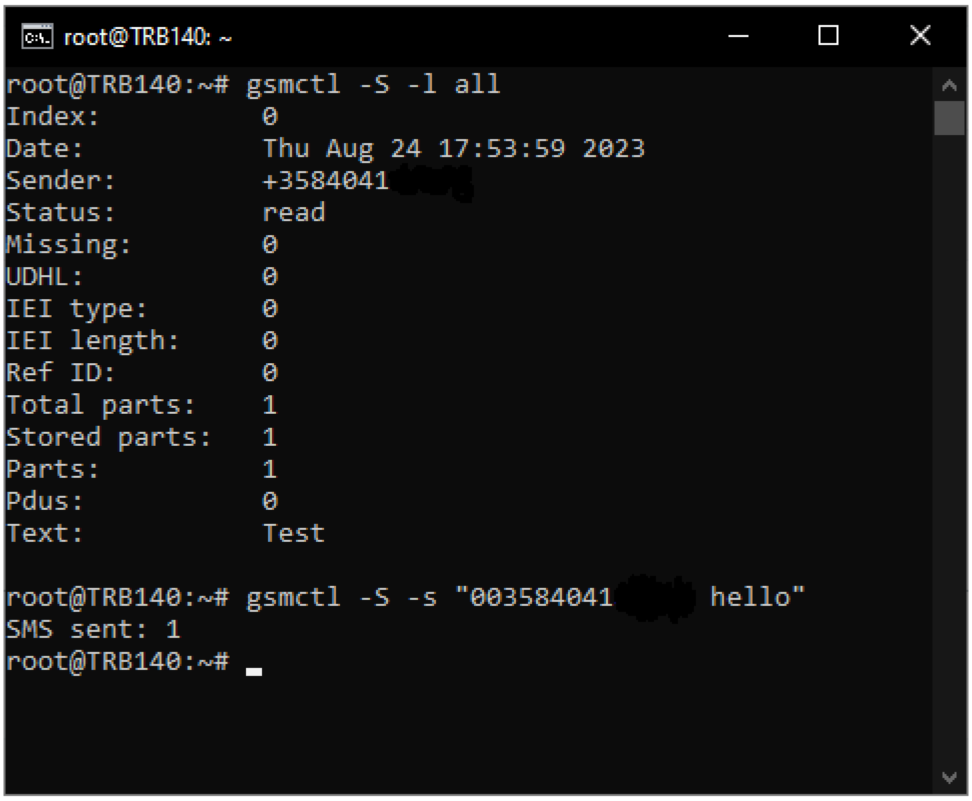


Figure 11. Top view of Teltonika Networks’ TRB140 [14].

The device offers a WebUI (Web user interface), which can be accessed with the device’s IP address. WebUI provides access to the device’s information, configurations and actions. [15.]

2.6.2 SMS Interfaces

TRB140’s SSH interface gives access to gsmctl command set, which is a utility that relay commands to the device’s modem to control and obtain information. The command ”gsmctl -S -l all” shows all received messages. To send text messages, the command ”gsmctl -S -s <NUMBER> <TEXT>" is used as shown in figure 12. [16.]



```
root@TRB140: ~
root@TRB140:~# gsmctl -S -l all
Index:          0
Date:           Thu Aug 24 17:53:59 2023
Sender:         +3584041[redacted]
Status:         read
Missing:        0
UDHL:           0
IEI type:       0
IEI length:     0
Ref ID:         0
Total parts:    1
Stored parts:   1
Parts:          1
Pdus:           0
Text:           Test

root@TRB140:~# gsmctl -S -s "003584041[redacted] hello"
SMS sent: 1
root@TRB140:~# _
```

Figure 12. Reading and sending SMS messages with gsmctl commands

The second option is using POST/GET which allows the user to read and send text messages by sending HTTP POST/GET strings to the device using compatible software, such as cURL [17]. Listing 1 shows GET method for reading text messages using cURL.

```
curl -X GET http://192.168.1.1/cgi-
bin/sms_list?username=user1&password=user_pass
```

Listing 1. GET method for reading text messages [17].

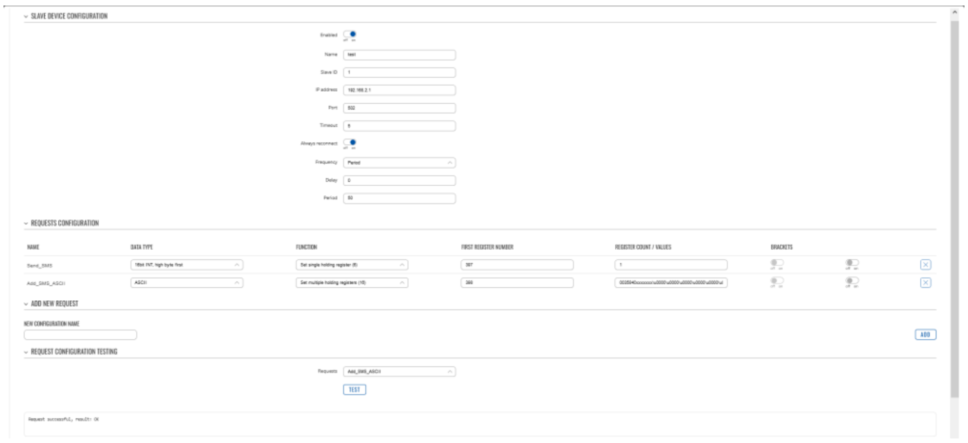
Sending text messages via the GET method can be done as presented in listing 2.

```
curl -X GET http://192.168.1.1/cgi-
bin/sms_send?username=user1&password=user_pass&number=0037000000000&te
xt=testmessage
```

Listing 2. GET method for sending text messages [17].

The third option for sending text messages is with Modbus TCP. TRB140 contains registers for text message content and a register to send the text message. Other Modbus parameters such as system uptime, temperature and mobile signal strength can also be read which can be useful for system diagnostics. [18.]

Modbus TCP slave option needs to be enabled for the device from the WebUI. The device has a master option where the slave device's configuration can be set (figure 13). The slave device's name and IP address are set from here. Requests configuration tab contain the Modbus requests where the register number and function are defined. [18.]


SLAVE DEVICE CONFIGURATION

Enabled: on
Name: test
Slave ID: 1
IP address: 192.168.2.1
Port: 502
Timeout: 5
Always reconnect: on
Frequency: Period
Delay: 0
Period: 60

REQUESTS CONFIGURATION

NAME	DATA TYPE	FUNCTION	FIRST REGISTER NUMBER	REGISTER COUNT / VALUES	BRACKETS
Send_SMS	16bit INT, high byte first	Set single holding register (6)	397	1	
Add_SMS_ASCII	ASCII	Set multiple holding registers (16)	399	003584xxxxxxx\u0000\u0000\u0000\u0000\u0000\u	

ADD NEW REQUEST

NEW CONFIGURATION NAME

REQUEST CONFIGURATION TESTING

Requests: Add_SMS_ASCII

Request successful, result: OK

Figure 13. TRB140's Modbus master view.

The "Add SMS" registers are written with the function "Set multiple holding registers (16)" as a data type of ASCII where the first 10 registers are reserved for the recipient's phone number and the remaining 80 registers are reserved for the message content. Each register is composed of two ASCII symbols. The message format is presented in figure 14. 003706xxx1594 is the phone number which is preceded by 00 which equals + symbol in this context. The phone number is followed by \u0000 which fills the remaining reserved registers for the phone number. The "test" is the message content. Value 1 is written to "Send SMS" register and the text message is sent to the recipient. [19.]

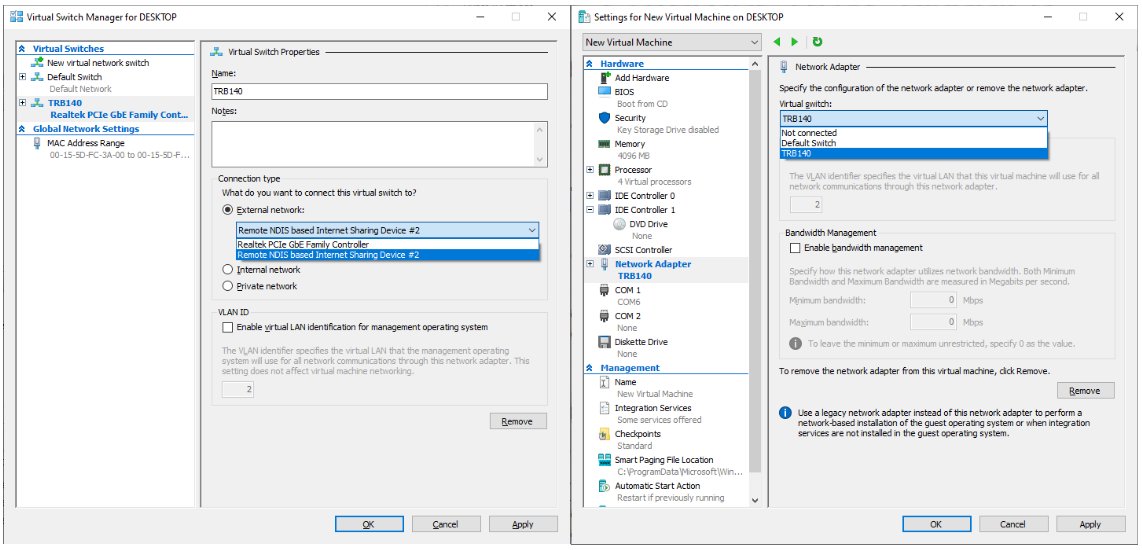
```
003706xxx1594\u0000\u0000\u0000\u0000\u0000\u0000\u0000test
```

Figure 14. Message format for "Add SMS" registers [19].

3 Development of TCP/IP-Based SMS alarm plug-in

3.1 Connecting TRB140 with MicroSCADA

Testing of TRB140 was done in a Hyper-V virtual computer with MicroSCADA software running a single system demo application with pre-installed serial SMS alarm plug-in. TRB140 network connection was configured to work as a virtual switch in Hyper-V as presented in figure 15. This allows a device to share its connection with a virtual computer.



Virtual Switch Manager for DESKTOP

Virtual Switches
New virtual network switch
Default Switch
Default Network
TRB140
Realtek PCIe GbE Family Cont...
Global Network Settings
MAC Address Range
00-15-5D-FC-3A-00 to 00-15-5D-F...

Virtual Switch Properties

Name: TRB140

Notes:

Connection type
What do you want to connect this virtual switch to?
[✓ External network:] Remote NDIS based Internet Sharing Device #2
Realtek PCIe GbE Family Controller
Remote NDIS based Internet Sharing Device #2
Internal network
Private network

VLAN ID
☐ Enable virtual LAN identification for management operating system
The VLAN identifier specifies the virtual LAN that the management operating system will use for all network communications through this network adapter. This setting does not affect virtual machine networking.
2

Remove

OK Cancel Apply

Settings for New Virtual Machine on DESKTOP

New Virtual Machine

Hardware
Add Hardware
BIOS
Boot from CD
Security
Key Storage Drive disabled
Memory
4096 MB
Processor
4 Virtual processors
IDE Controller 0
IDE Controller 1
DVD Drive
None
SCSI Controller
Network Adapter
TRB140
COM 1
COM6
COM 2
None
Diskette Drive
None
Management
Name
New Virtual Machine
Integration Services
Some services offered
Checkpoints
Standard
Smart Paging File Location
C:\ProgramData\Microsoft\Win...
Automatic Start Action
Restart if previously running

Network Adapter

Specify the configuration of the network adapter or remove the network adapter.

Virtual switch: TRB140
Not connected
Default Switch
TRB140

The VLAN identifier specifies the virtual LAN that this virtual machine will use for all network communications through this network adapter.
2

Bandwidth Management
☐ Enable bandwidth management
Specify how this network adapter utilizes network bandwidth. Both Minimum Bandwidth and Maximum Bandwidth are measured in Megabits per second.
Minimum bandwidth: 0 Mbps
Maximum bandwidth: 0 Mbps
To leave the minimum or maximum unrestricted, specify 0 as the value.

To remove the network adapter from this virtual machine, click Remove.

Remove

Use a legacy network adapter instead of this network adapter to perform a network-based installation of the guest operating system or when integration services are not installed in the guest operating system.

OK Cancel Apply

Figure 15. Hyper-V virtual machine network configuration for TRB140.

3.2 Selecting a Communication Protocol

Originally the chosen communication protocol for the new plug-in would have been HTTP POST/GET requests, as the examples provided by Teltonika looked simple, and HTTP is a widely used and well documented protocol. Although MicroSCADA does not support this protocol, external programs such as cURL, can be launched from MicroSCADA environment using ops_process SCIL function. ops_process function allows MicroSCADA to start an external program running outside the MicroSCADA environment [11, 208]. This method was tested by writing a simple SCIL program, that performs SMS message sending using the previously mentioned POST/GET example (listing 3).

```
#local sms_send = "http://192.168.2.1/cgi-
bin/sms_send?username=user1&password=user1&number=003584041111111&
text=Test"

@sms_process = ops_process("curl -X GET " + """'sms_send'""")
```

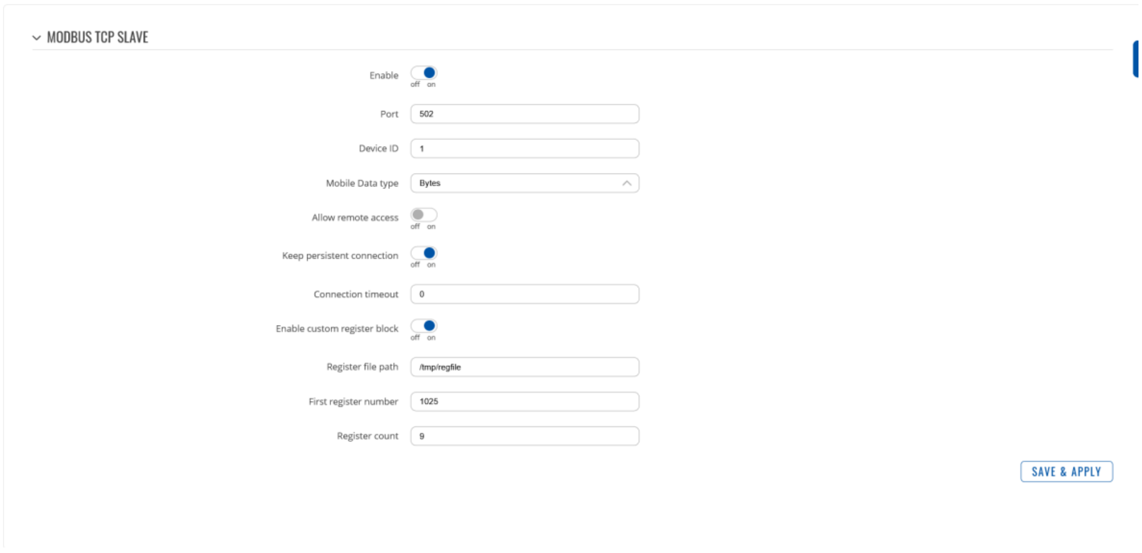
Listing 3. SCIL code for sending SMS messages via POST/GET method.

However, reading text messages could not be done from MicroSCADA as POST/GET method is not a supported communication protocol in MicroSCADA. If an external program would handle the POST/GET requests, there would need to be a MicroSCADA supported communication protocol in the program, and creating a communication protocol converter would have been challenging. POST/GET requests are technically possible from MicroSCADA if a CPI (Communication Programming Interface) program would be implemented. CPI is a protocol environment software, which enables MicroSCADA to communicate with external programs and protocols and includes functions for sending and receiving messages to or from MicroSCADA. The program would be emulated as a station in MicroSCADA. [20, 7.] To create a functional CPI program would have been challenging as this requires in-depth knowledge about CPI and the external communication protocol.

The remaining option, Modbus TCP, was selected as it is a well understood and well documented protocol. Most importantly, MicroSCADA has a native support for this protocol [21, 5]. Also, as other Modbus parameters for TRB140 were available to read and write such as system temperature and mobile signal strength, it seemed a compelling reason to select this protocol. It did not, however, support reading text messages but was solved by using a custom Modbus register block, which allowed user-defined data to be inputted to custom registers which is detailed more in next chapter.

3.3 TRB140 Configuration and Engineering

By default, TRB140 does not support reading received text messages via Modbus. This was remedied by using a custom Modbus register block which allows user defined data to be read from a file. The register block is enabled from the TRB140's WebUI (figure 16). [22.] The Modbus register block content is read from the default register file path, /tmp/regfile. The register number was also left as default except register count value was adjusted to 9, which left enough space for a 14-character-long phone number and a 4-character-long message.



Enable	on
Port	502
Device ID	1
Mobile Data type	Bytes
Allow remote access	off
Keep persistent connection	on
Connection timeout	0
Enable custom register block	on
Register file path	/tmp/regfile
First register number	1025
Register count	9

Figure 16. Enabling Modbus slave option and custom Modbus register block.

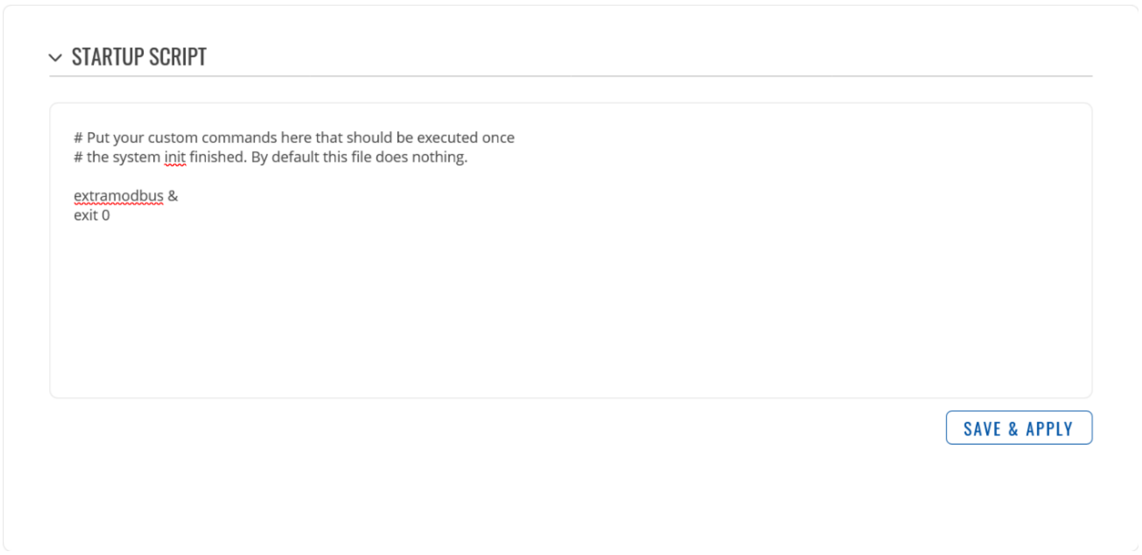
To write content to /tmp/regfile, a shell script called extramodbus was created using Almquist shell. The script runs in an infinite loop in 1 second intervals, constantly checking the output of ”gsmctl -S -l all” command. As the gsmctl command output is in the format as shown in previous figure 12, it was sensible to strip it down to a simpler form. When the gsmctl command outputs a received message, the sender's number and message are concatenated to a format of “00358401111111Test” which is outputted to regfile and waits five seconds. The message is then deleted from memory before showing the next message. In case gsmctl does not detect messages, space characters are outputted to regfile.

The script ensures that the register block is always filled as reading a partially filled Modbus register resulted in an error in the Modbus master device. In case the received text message is shorter than four characters, the script appends space characters until the required length is met. The message read from registers will always be four characters long assuming that the received number will be 14 characters long. Messages exceeding the register block length will be cut off and only the first four characters can be read from the registers.

The script was made executable with the following command from the command line interface:

```
chmod +x /bin/extramodbus
```

After this the script was set to launch during the boot of the device and runs in the background (figure 17).



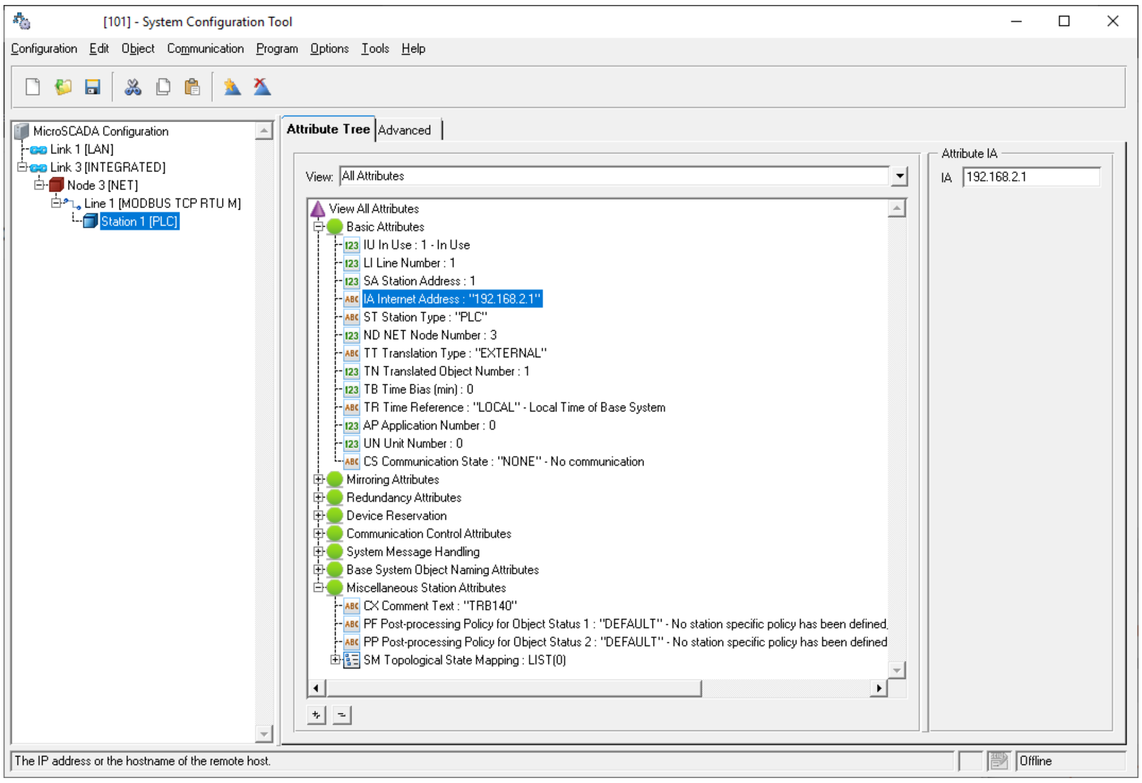
```
# Put your custom commands here that should be executed once
# the system init finished. By default this file does nothing.

extramodbus &
exit 0
```

Figure 17. Launching extramodbus during the boot of the device.

3.4 MicroSCADA System Configuration

TRB140 was configured as a process communication unit for MicroSCADA system with System Configuration Tool as presented in figure 18. The communication protocol line is defined within a Node that already exists in the demo application where Modbus TCP Master Line and a station object for the device were created. After this the device's IP address is set to Internet Address (IA) field. Other line and station attributes were initially left at a default value.


[101] - System Configuration Tool
Configuration Edit Object Communication Program Options Tools Help
Attribute Tree
Advanced
View:
Attribute IA
Offline

MicroSCADA Configuration
Link 1 [LAN]
Link 3 [INTEGRATED]
Node 3 [NET]
Line 1 [MODBUS TCP RTU M]
Station 1 [PLC]

View: All Attributes

```
View All Attributes
  Basic Attributes
    IU In Use : 1 - In Use
    LI Line Number : 1
    SA Station Address : 1
    IA Internet Address : "192.168.2.1"
    ST Station Type : "PLC"
    ND NET Node Number : 3
    TT Translation Type : "EXTERNAL"
    TN Translated Object Number : 1
    TB Time Bias (min) : 0
    TR Time Reference : "LOCAL" - Local Time of Base System
    AP Application Number : 0
    UN Unit Number : 0
    CS Communication State : "NONE" - No communication
  Mirroring Attributes
  Redundancy Attributes
  Device Reservation
  Communication Control Attributes
  System Message Handling
  Base System Object Naming Attributes
  Miscellaneous Station Attributes
    CX Comment Text : "TRB140"
    PF Post-processing Policy for Object Status 1 : "DEFAULT" - No station specific policy has been defined
    PP Post-processing Policy for Object Status 2 : "DEFAULT" - No station specific policy has been defined
    SM Topological State Mapping : LIST(0)
```

IA 192.168.2.1

The IP address or the hostname of the remote host.

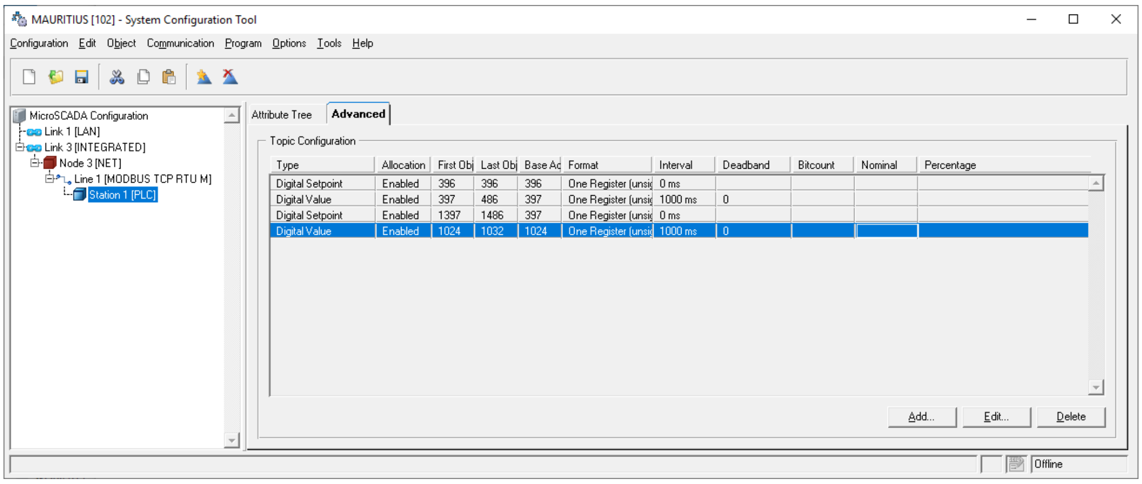
Figure 18. MicroSCADA process communication configuration for TRB140.

Scanning of Modbus device data is defined by the device topics which can be found in the Advanced tab of the System Configuration Tool [21, 32]. A topic type of digital value was set to read "Add SMS" registers in the format of unsigned value. When defining a digital value topic for a Modbus register, the following parameters need also to be set:

- First Object Address: the first MicroSCADA process object address.
- Last Object Address: the last MicroSCADA process object address.
- Base Address: the actual Modbus register address.
- Interval: the scanning frequency of the defined topic. [21, 32-36.]

The number of Modbus registers read and written by a topic is calculated as follows: Number of items = LastObjectAddress - FirstObjectAddress + 1 [21, 34].

Figure 19 presents the topic configuration for TRB140. When reading ”Add SMS” parameter, the amount of registers needed is 90. The first object address is set to 397 and the last object address is set to 486 according to the previous formula. Base address is the the Modbus address. In this case it is set to 397. Interval is the frequency the topic data is read from the device and is left at the default 1000 milliseconds. The same Modbus parameter was configured as a topic type of digital setpoint for writing the Modbus registers and also “Send SMS” parameter as a topic of same type. Custom register block was also added as a topic type of Digital Value.



Type	Allocation	First Obj	Last Obj	Base Ad	Format	Interval	Deadband	Bitcount	Nominal	Percentage
Digital Setpoint	Enabled	396	396	396	One Register (unsi	0 ms				
Digital Value	Enabled	397	486	397	One Register (unsi	1000 ms	0			
Digital Setpoint	Enabled	1397	1486	397	One Register (unsi	0 ms				
Digital Value	Enabled	1024	1032	1024	One Register (unsi	1000 ms	0			

Figure 19. MicroSCADA topic configuration for TRB140

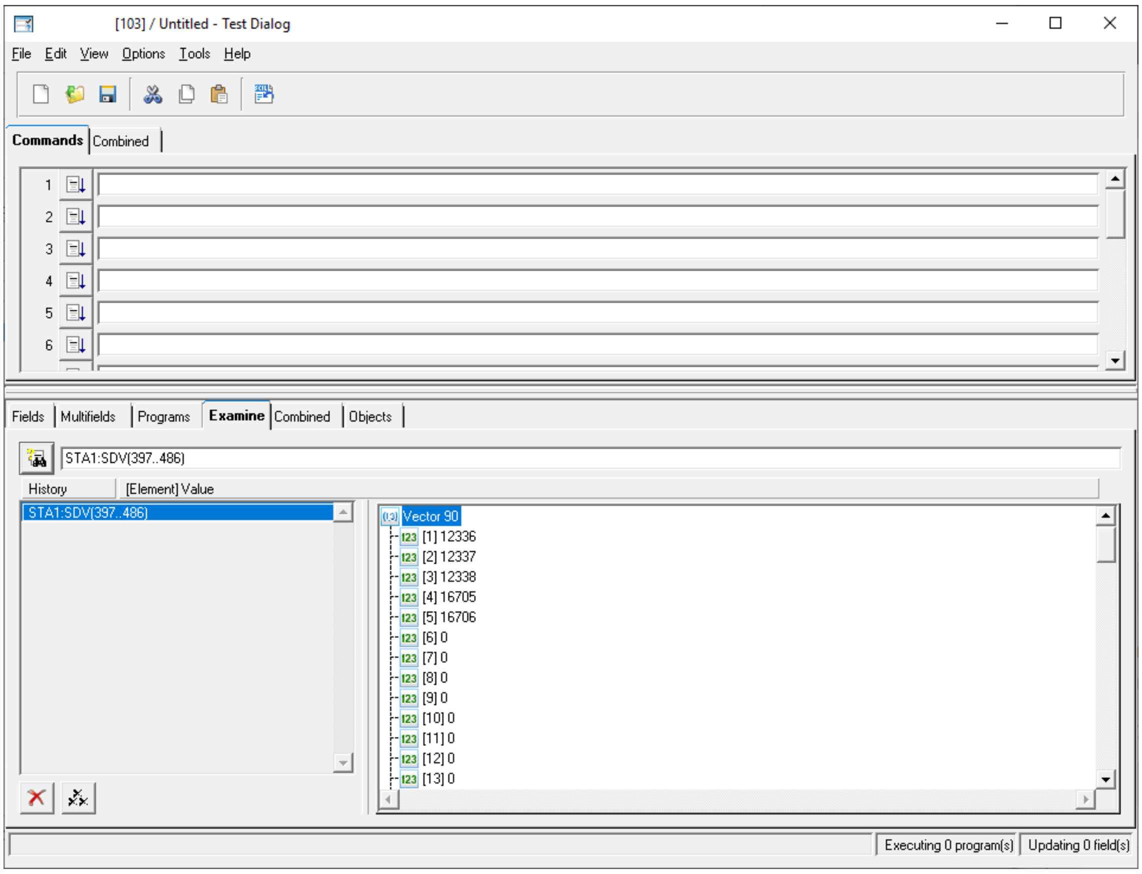
3.5 MicroSCADA Application Engineering

The initial step in testing text message sending was interpreting the ”Add SMS” registers’ format from MicroSCADA. This parameter’s first five registers were written in the TRB140’s Modbus master service with a string ”000102AAAB” as a data type of ASCII to inspect the register value changes in MicroSCADA using

Test dialog. Reading the Modbus registers from MicroSCADA is done using the following SCIL statement [21, 54]:

```
@VALUE = STA1:SDV(1..2)
```

The previous SCIL statement was adjusted accordingly for ”Add SMS” registers and the SCIL excerpt was inserted into the Examine field of Test Dialog (figure 20). This resulted in the Modbus registers being read from application.


[103] / Untitled - Test Dialog
File Edit View Options Tools Help
Commands Combined
Fields Multifields Programs Examine Combined Objects
History [Element] Value
Executing 0 program(s) Updating 0 field(s)

```
STA1:SDV(397..486)

Vector 90
[1] 12336
[2] 12337
[3] 12338
[4] 16705
[5] 16706
[6] 0
[7] 0
[8] 0
[9] 0
[10] 0
[11] 0
[12] 0
[13] 0
```

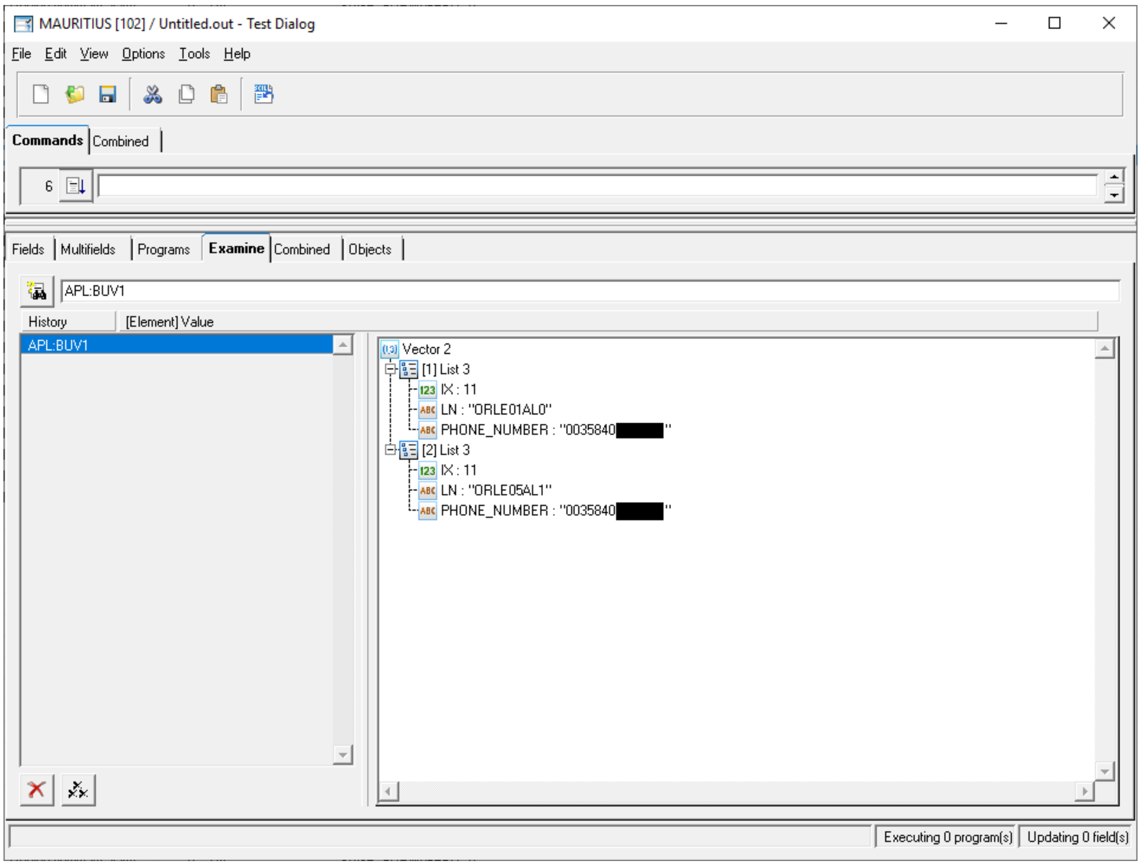
Figure 20. Inspecting "Add SMS" register values from MicroSCADA with Test Dialog.

The initial format for ”Add SMS” was unknown when inspected from MicroSCADA. The format was found by inspecting the registers using 16-bit integer format and hexadecimal format from TRB140’s Modbus master service and an ASCII character code chart. It was concluded that the hexadecimal values of the ASCII characters match with the Modbus request’s hexadecimal format’s values. This means that the values MicroSCADA reads are decimal interpretations of the hexadecimal ASCII character codes. The format was the same for the custom Modbus register block.

The next step was to find suitable SCIL statements to convert text to decimal format. During this period of plug-in development, a prototype command procedure for text message sending was made, as this allowed easier troubleshooting and establishing the basic logic. The fundamental idea of the command procedure is that it splits phone number and message into two-character long substrings using UNPACK_STR SCIL function which are then converted to a decimal form using RTU_INT function. In case the last substring contains only one character, a null character is added to the last substring to prevent error in RTU_INT function. (Appendix 1)

At this point, the phone number and message are a data type of vector which is the needed data type when writing multiple Modbus register from MicroSCADA [21, 53]. After converting the phone number to RTU_INT format, the vector is appended with zeroes to ensure that the vector is the length of 10, as the first 10 registers of “Add SMS” parameter are reserved for the phone number. Both vectors are then concatenated and written to the “Add SMS” registers and “Send SMS” register is written with the value 1, which sends the text message to the recipient. (Appendix 1)

After the basic logic of sending a text message was established, the logic was integrated to SMS_ALARM, the pre-existing command procedure of the serial SMS alarm plug-in. This command procedure was also modified so that it saves the sent alarm’s LN, IX and the recipient’s phone number as a data type of list (figure 21). The list is then appended to APL:BUV1. This attribute is used to search alarms in the alarm acknowledgement command procedure. UV (user variable) attribute can be used as a global variable in application programs [10, 57].


MAURITIUS [102] / Untitled.out - Test Dialog

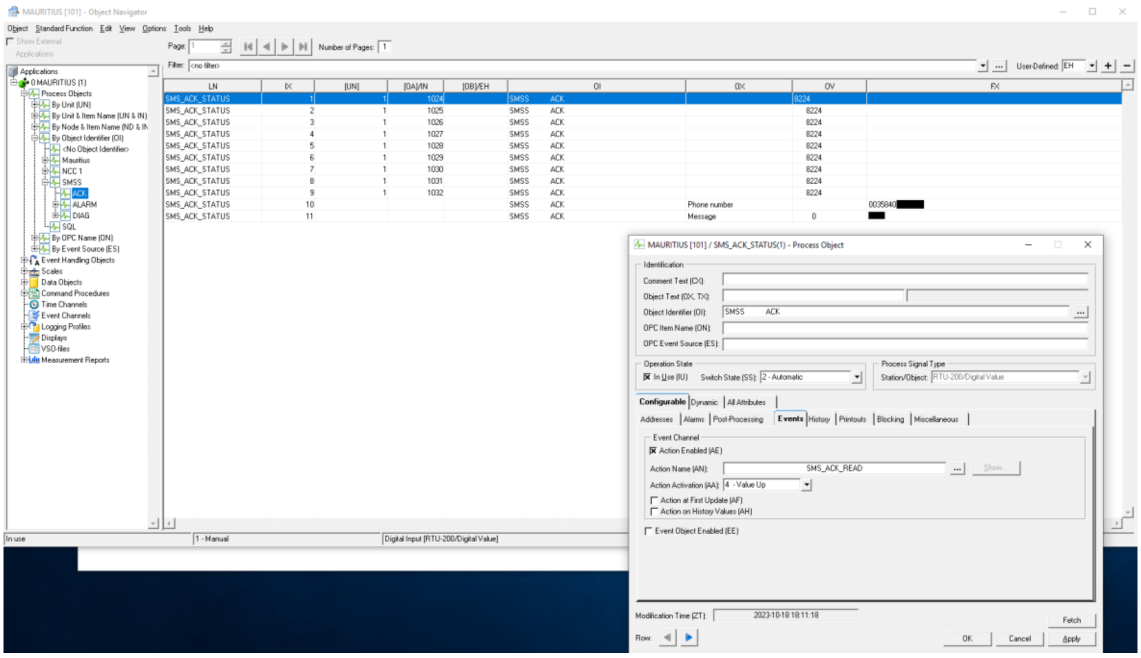
APL:BUV1

```
Vector 2
  [1] List 3
    IX : 11
    LN : "ORLE01AL0"
    PHONE_NUMBER : "0035840[redacted]"
  [2] List 3
    IX : 11
    LN : "ORLE05AL1"
    PHONE_NUMBER : "0035840[redacted]"
```

Executing 0 program(s) | Updating 0 field(s)

Figure 21. APL:BUV1 contents inspected with Test Dialog. Sent SMS alarms are saved here.

11 process objects were created to the process database called SMS_ACK_STATUS. The first nine indexes of this process object group read the custom Modbus register block values. The first index also has a command procedure attached it, SMS_ACK_READ, which handles the incoming text messages and alarm acknowledgement. This command procedure activates whenever the first index’s object value goes up, in other words, whenever a text message is received. SMS_ACK_READ reads the decimal values from the first 9 objects and transforms them to text form using RTU_AINT SCIL function, the reverse function for RTU_INT. The values are then written to the FX (free text) attribute of indexes 10 and 11. Index 10 contains the phone number and index 11 contains the message. Configured process objects are shown in figure 22.



Page: 1 Number of Pages: 1

Filter: <no filter>

User-Defined: EH

LN	IX	[UN]	[OA]/IN	[OB]/EH	OI		OX	OV	FX
SMS_ACK_STATUS	1	1	1024		SMSS	ACK		8224	
SMS_ACK_STATUS	2	1	1025		SMSS	ACK		8224	
SMS_ACK_STATUS	3	1	1026		SMSS	ACK		8224	
SMS_ACK_STATUS	4	1	1027		SMSS	ACK		8224	
SMS_ACK_STATUS	5	1	1028		SMSS	ACK		8224	
SMS_ACK_STATUS	6	1	1029		SMSS	ACK		8224	
SMS_ACK_STATUS	7	1	1030		SMSS	ACK		8224	
SMS_ACK_STATUS	8	1	1031		SMSS	ACK		8224	
SMS_ACK_STATUS	9	1	1032		SMSS	ACK		8224	
SMS_ACK_STATUS	10				SMSS	ACK	Phone number		0035840[redacted]
SMS_ACK_STATUS	11				SMSS	ACK	Message	0	[redacted]

In use | 1 - Manual | Digital Input (RTU-200/Digital Value)

MAURITIUS [101] / SMS_ACK_STATUS(1) - Process Object

Identification
Comment Text (CX):
Object Text (OX, TX):
Object Identifier (OI): SMSS ACK
OPC Item Name (ON):
OPC Event Source (ES):

Operation State
[✓] In Use (IU) Switch State (SS): 2 - Automatic

Process Signal Type
Station/Object: RTU-200/Digital Value

Configurable | Dynamic | All Attributes
Addresses | Alarms | Post-Processing | Events | History | Printouts | Blocking | Miscellaneous

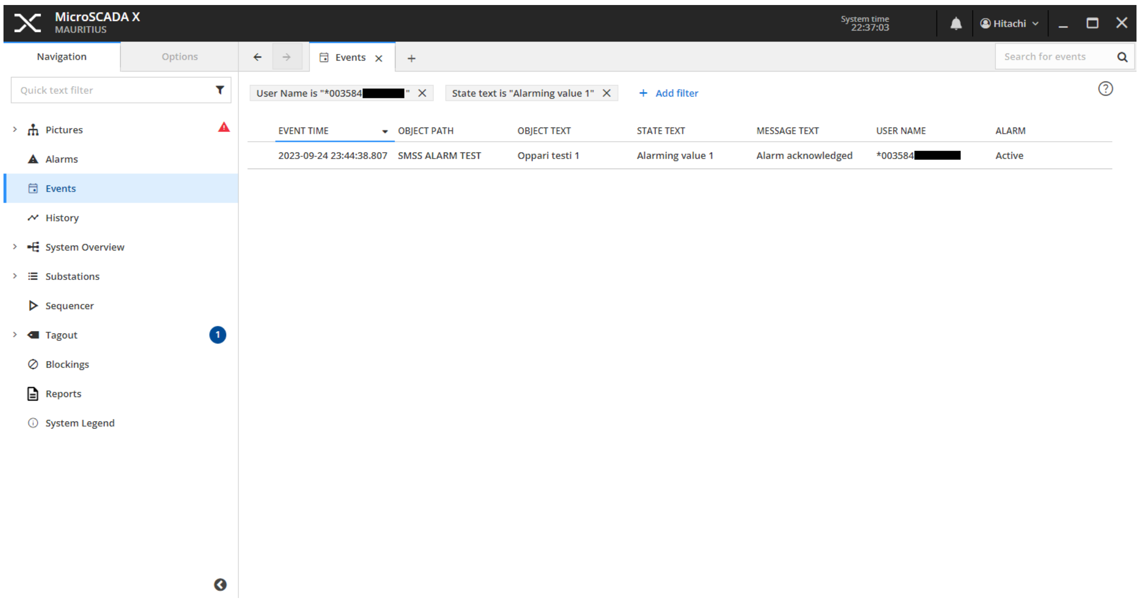
Event Channel
[✓] Action Enabled (AE)
Action Name (AN): SMS_ACK_READ
Action Activation (AA): 4 - Value Up
[ ] Action at First Update (AF)
[ ] Action on History Values (AH)

[ ] Event Object Enabled (EE)

Modification Time (ZT): 2023-10-18 18:11:18

Figure 22. Object Navigator view of process objects that contain the custom Modbus register block values. The selected process object activates SMS_ACK_READ, which handles the reading of received SMS messages and acknowledgement of SMS alarms.

SMS_ACK_READ checks if SMS_ACK_STATUS index's 10 FX attribute's value exists in the phone number list of the operators. It then checks the text message content. If SMS_ACK_STATUS index's 11 FX attribute contains the correct acknowledgement code, the command procedure fetches sent alarms from APL:BUV1 and sets the alarm's AR attribute to 1. The acknowledged alarm is then deleted from APL:BUV1. The alarm acknowledgement is also shown in MicroSCADA's event list as seen in figure 23.



User Name is "*003584[redacted]"

State text is "Alarming value 1"

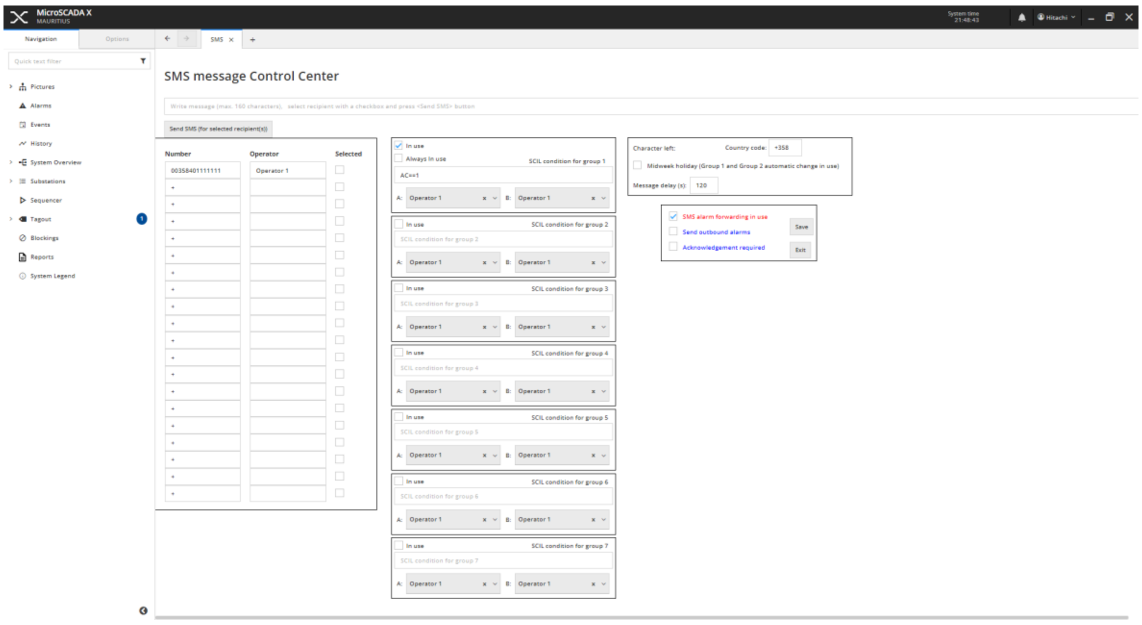
EVENT TIME	OBJECT PATH	OBJECT TEXT	STATE TEXT	MESSAGE TEXT	USER NAME	ALARM
2023-09-24 23:44:38.807	SMSS ALARM TEST	Oppari testi 1	Alarming value 1	Alarm acknowledged	*003584[redacted]	Active

Figure 23. Workplace X event list view of remotely acknowledged alarm. User name column contains the operator’s phone number.

Control commands other than alarm acknowledgements, are possible to implement with the previously mentioned command procedure, such as sending a control command to a breaker.

The reminder text message command procedures were modified to use the format conversion in a similar manner which was used in sending out the alarm messages.

SMS management page’s XML file was also modified so that the “Send SMS” button uses RTU_INT format conversion. Visual adjustments were also made to the user interface such as modifying the text of “Call delay” as “Message delay” as the new plug-in does not support making calls. The new user interface is presented in figure 24.



SMS message Control Center

Write message (max. 160 characters), select recipient with a checkbox and press <Send SMS> button

Number	Operator	Selected
00358401111111	Operator 1	☐

[✓ In use]
Always in use ☐
SCIL condition for group 1: AC==1
A: Operator 1 B: Operator 1

In use ☐ SCIL condition for group 2 — A: Operator 1 B: Operator 1
In use ☐ SCIL condition for group 3 — A: Operator 1 B: Operator 1
In use ☐ SCIL condition for group 4 — A: Operator 1 B: Operator 1
In use ☐ SCIL condition for group 5 — A: Operator 1 B: Operator 1
In use ☐ SCIL condition for group 6 — A: Operator 1 B: Operator 1
In use ☐ SCIL condition for group 7 — A: Operator 1 B: Operator 1

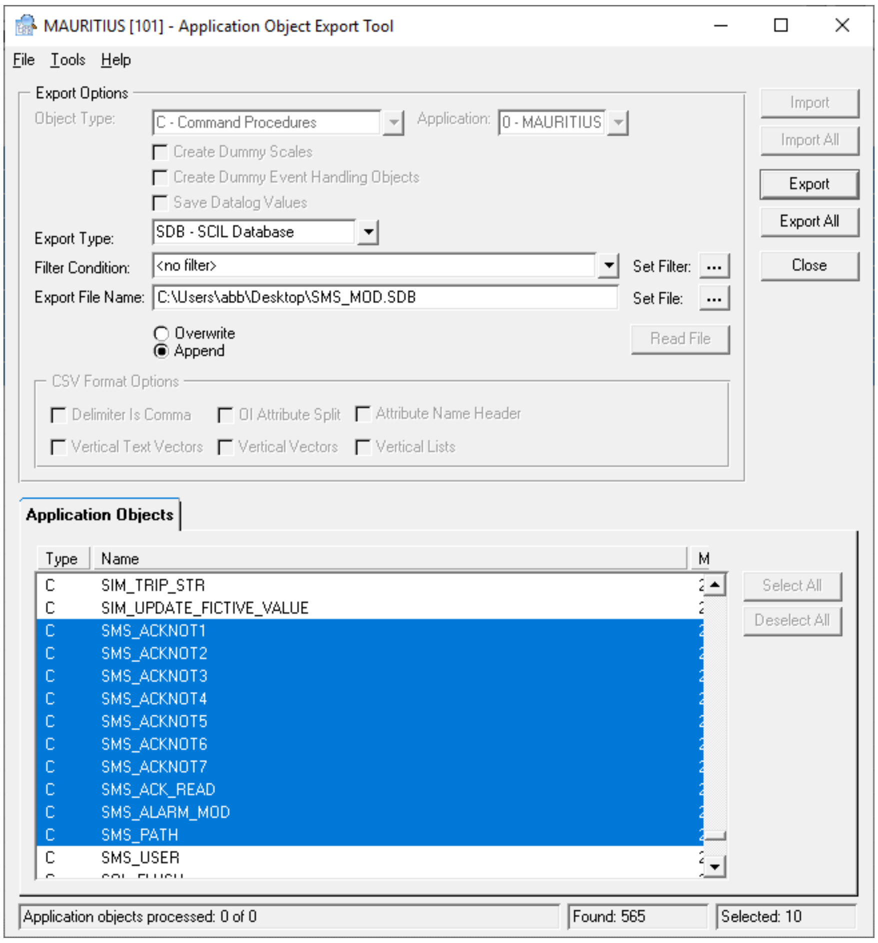
Character left:
Country code: +358
☐ Midweek holiday (Group 1 and Group 2 automatic change in use)
Message delay (s): 120

[✓ SMS alarm forwarding in use]
☐ Send outbound alarms
☐ Acknowledgement required

Figure 24. Modified SMS alarm plug-in management page.

3.6 Installation Packages

After the new SMS alarm plug-in was developed, installation packages were made to be available for engineers. Most components of the old plug-in were left out of the installation packages, such as the external programs and their initialization files, as these were not used anymore. Command procedures, process objects and other application objects related to the new plug-in were exported using Object Navigator as shown in figure 25. SMS user interface and its initialization file were also copied from hard drive.


MAURITIUS [101] - Application Object Export Tool

File Tools Help

Export Options

Object Type: C - Command Procedures

Application: 0 - MAURITIUS

Create Dummy Scales

Create Dummy Event Handling Objects

Save Datalog Values

Export Type: SDB - SCIL Database

Filter Condition: <no filter>

Export File Name: C:\Users\abb\Desktop\SMS_MOD.SDB

Overwrite [✓ Append]

CSV Format Options

Delimiter Is Comma, OI Attribute Split, Attribute Name Header, Vertical Text Vectors, Vertical Vectors, Vertical Lists

Application Objects

Type	Name
C	SIM_TRIP_STR
C	SIM_UPDATE_FICTIVE_VALUE
C	SMS_ACKNOT1
C	SMS_ACKNOT2
C	SMS_ACKNOT3
C	SMS_ACKNOT4
C	SMS_ACKNOT5
C	SMS_ACKNOT6
C	SMS_ACKNOT7
C	SMS_ACK_READ
C	SMS_ALARM_MOD
C	SMS_PATH
C	SMS_USER

Application objects processed: 0 of 0

Found: 565

Selected: 10

Figure 25. Exporting of SMS alarm plug-in application objects.

In addition to creating the installation files, installation manual was also written (appendix 2). This manual provides a general guideline for TRB140 and MicroSCADA configuration for the deployment of the plug-in for engineers.

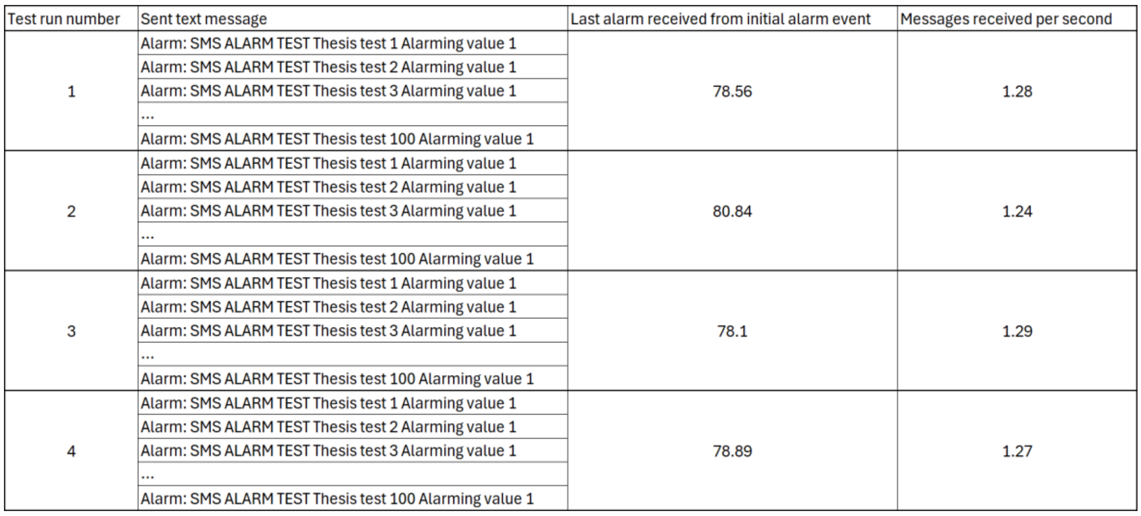
4 Testing of the New SMS Alarm Plug-in

4.1 Performance Evaluation

Initially the performance of the text message sending was poor, and when multiple alarms were generated in MicroSCADA, it resulted in errors related to the Modbus line. The first few alarms were sent, but then the Modbus line got congested and no more text messages were sent. It resulted in pauses of minute or two before the text message sending continued. This was fixed by increasing the Modbus line and station attributes related to performance, such as buffer size and enquiry limit. Most of the attributes' values were increased to maximum or near the maximum. The configuration for these attributes are presented in the installation manual (Appendix 2, 14-15). After these adjustments, text message sending was successful without hangups. If the TRB140 and other devices with many topics were under the same Modbus line, it could hinder the performance of the plug-in. To ensure the best performance of the plug-in, the performance attributes should be turned as high as possible and create a dedicated Modbus line for TRB140.

When a single alarm is generated in MicroSCADA, a text message is delivered to one recipient within around one second or at maximum two seconds with an adequate mobile signal strength using 4G network. Usually MicroSCADA's alarm text are designed to be short and concise. Simulated process objects with long alarm texts were created to evaluate whether it affected the delivery and the delivery speed of text messages. Long alarm texts did not seem to affect the delivery or delivery speed when a single text message was sent to a recipient.

If a station connected to a customer's MicroSCADA system encounters a catastrophic failure, multiple alarm signals can activate at the same time, sometimes exceeding 100 alarms. The current SMS alarm plug-in is often used in a customer system where there can be multiple operators. The amount of text messages to be sent is multiplied by the number of operators, for example 25 alarms to be sent to 4 operators means 100 text messages. It was important to evaluate the performance of the SMS alarm plug-in for these conditions to ensure that the message delivery was successful and that the delivery speed was adequate. For simulating these circumstances, 100 process objects were created, which were triggered to alarm at the same time using SCIL. A stopwatch was used to measure the time from the alarm trigger to the last received text message. TRB140's mobile connection type was 4G and the mobile signal strength 65 dBm. The recipient's phone connection type was also 4G. The mass sending of text messages was conducted four times as presented in figure 26.



Test run number	Sent text message	Last alarm received from initial alarm event	Messages received per second
	Alarm: SMS ALARM TEST Thesis test 1 Alarming value 1		
	Alarm: SMS ALARM TEST Thesis test 2 Alarming value 1		
1	Alarm: SMS ALARM TEST Thesis test 3 Alarming value 1	78.56	1.28
	...		
	Alarm: SMS ALARM TEST Thesis test 100 Alarming value 1		
	Alarm: SMS ALARM TEST Thesis test 1 Alarming value 1		
	Alarm: SMS ALARM TEST Thesis test 2 Alarming value 1		
2	Alarm: SMS ALARM TEST Thesis test 3 Alarming value 1	80.84	1.24
	...		
	Alarm: SMS ALARM TEST Thesis test 100 Alarming value 1		
	Alarm: SMS ALARM TEST Thesis test 1 Alarming value 1		
	Alarm: SMS ALARM TEST Thesis test 2 Alarming value 1		
3	Alarm: SMS ALARM TEST Thesis test 3 Alarming value 1	78.1	1.29
	...		
	Alarm: SMS ALARM TEST Thesis test 100 Alarming value 1		
	Alarm: SMS ALARM TEST Thesis test 1 Alarming value 1		
	Alarm: SMS ALARM TEST Thesis test 2 Alarming value 1		
4	Alarm: SMS ALARM TEST Thesis test 3 Alarming value 1	78.89	1.27
	...		
	Alarm: SMS ALARM TEST Thesis test 100 Alarming value 1		

Figure 26. Performance of sending text messages with SMS alarm plug-in.

During test run 4 it was observed that one alarm text was received two times for unknown reasons. This duplication of the messages is probably related to the lack of an alarm buffer. However, this is probably not a critical bug as all messages were received at an adequate speed and no messages were lost during the testing.

4.2 Bugs and Missing Features

The new TCP/IP-based SMS alarm plug-in contains the most essential functionalities of the previous solution, but some features were not implemented due to technical reasons and time constraints. Some bugs were also encountered during the testing which were not fixed due to previous reasons. These missing features and bugs, and the possible solutions for them, are documented in this chapter.

4.2.1 Character Encoding

Unlike the serial SMS alarm plug-in, the developed plug-in has a limited character support. If the message contained a character of which ASCII code exceeded 127, it resulted in a “?” character in the received text message. This was discovered when the alarm text contained characters such as ”Ä”, ”ä”, “Ö” and “ö”. This was remedied by replacing these characters with “A”, “a”, “O” and “o” before sending the text message. This means that if the alarm text contains the word “Hälytys”, the command procedure transforms the word into “Halytys”. When the alarm text contained Finnish text, it resulted in an intelligible alarm text message, although it was not in the most optimal form. However, this could be a problem with other languages when special characters are not supported, such as Cyrillic characters. When special characters are written to the registers as a data type of ASCII from the TRB140’s web user interface, the received text message is in proper form. However, instead of one character taking half of the Modbus register, it takes up the whole register, meaning the character size is 16 bits. When converting these register values in MicroSCADA to a text form using RTU_AINT SCIL function, it resulted in unintelligible characters.

If the created solution would be integrated a customer’s MicroSCADA system, it should be determined whether special character replacements affect the intelligibility of the sent text messages and whether the customer accepts the lacking character support.

There are no built-in SCIL functions to convert special characters to a proper form, but it could be possible to implement a custom character conversion command procedure. When the correct character encoding is figured out, the command procedure should be relatively easy to implement.

Character encoding problem is also present when the alarm acknowledgement command procedure tries to read the custom Modbus register block when it contains a special character. RTU_AINT function could not convert the integer values read from the registers to proper text. Creating a integer-to-text conversion command procedure would solve this.

4.2.2 Mobile Calls

Making an outgoing call is not a supported Modbus parameter and was not implemented in the SMS alarm plug-in which was also why the SMS management page was visually modified. This could be a problem from an operational view. The outgoing call in the serial SMS alarm plug-in is sometimes used as a “last resort” to ensure that the operator reacts to the alarm, as making an outgoing call is more audible than a text message. This could be an essential feature for some customers, but can be compensated with the following steps:

- Adjust the message sound for the text message. The sound should be clear and audible.
- Making sure the mobile phone volume is always relatively high.
- Using a shorter reminder message interval than what would be used for a wake-call delay. As the text message might not be as audible as an incoming call, the shorter interval could help compensate for the lack of a constant ringtone.

The new alarm plug-in cannot fetch inbound caller’s number, nor can it be seen from TRB140’s event logs. This could pose a problem if an unknown number tries to make calls to the TRB140 and will also be evaluated in the next chapter from cybersecurity perspective.

Another bug related to mobile calls was that if constant call were made to TRB140, it hindered the performance of the plug-in to an almost unusable state for a few minutes. It was observed from the TRB140’s WebUI that making multiple calls changed the mobile connection type from the used 4G to 2G and 3G connection. The constant fluctuation between different mobile networks could be the reason calls hinder the performance significantly. Although TRB140 has a “Reject incoming calls” option which was turned on from the WebUI, the bug persisted.

4.2.3 SCADA Server State Monitoring

The new SMS alarm plug-in does not monitor the state of running state of MicroSCADA software or server. If MicroSCADA software were to crash, no text messages are sent to operators to inform about the situation. However, a command procedure could be made which activates and immediately deactivates one of the TRB140's input sockets via Modbus, and is periodically activated by a time channel object. Then a shell script would be created for TRB140 which monitors the pin state changes and if it does not detect a state change within three minutes, for example, it sends a text message about this to the operator by using gsmctl command. TRB140's input socket states can be obtained from the command line interface of TRB140 [23].

4.2.4 Redundancy

The redundancy concept of the serial SMS plug-in was not researched for the new plug-in. However, if TRB140 Modbus connection were to fail because of power outage for example, a redundant TRB140 can be configured for MicroSCADA. MicroSCADA supports station redundancy which requires the configuration of a primary station and a secondary station. In case of a primary station connection failure resulted from a power outage for example, MicroSCADA performs a switch-over and automatically switches to the secondary station. [10, 128-129.] Compared to the serial SMS alarm plug-in, this is better as this means that two TRB140's could always be available in a single system MicroSCADA.

In addition to configuring a redundant TRB140, an application should receive information about the running state of TRB140. For this purpose, an alarming process object should be created in the process database which has an object address of a System Message. System Messages inform an application about changes in the station communication [10, 164].

4.2.5 Custom Modbus Register Block Issues

As mentioned before, the TRB140’s custom Modbus register block length is set to 18 characters which puts limits to the received message length. The outputted message read from registers will be four characters long assuming that the received number will be 14 characters long. Messages exceeding the register block length will be cut off and only the first four characters can be read from the registers. It is possible to receive 160-character SMS messages, but this was not investigated further in this thesis. This would require extending the register block length, modifying the shell script, and adjusting MicroSCADA system and application configurations.

Another problem related to the custom Modbus register and shell script is that if the received phone number is not 14 characters long, SMS_ACK_READ will not work correctly as it assumes that the areas from which the phone number and message are read stay constant. This problem is presented in figure 27. The upper section of figure depicts a normal situation when SMS_ACK_READ reads a 14-character-long phone number and text message from the Modbus registers. The lower section of the figure depicts an abnormal situation when the received phone number is 16 characters long. In this situation, the last part of the phone number is located at the area where the actual message should be read.

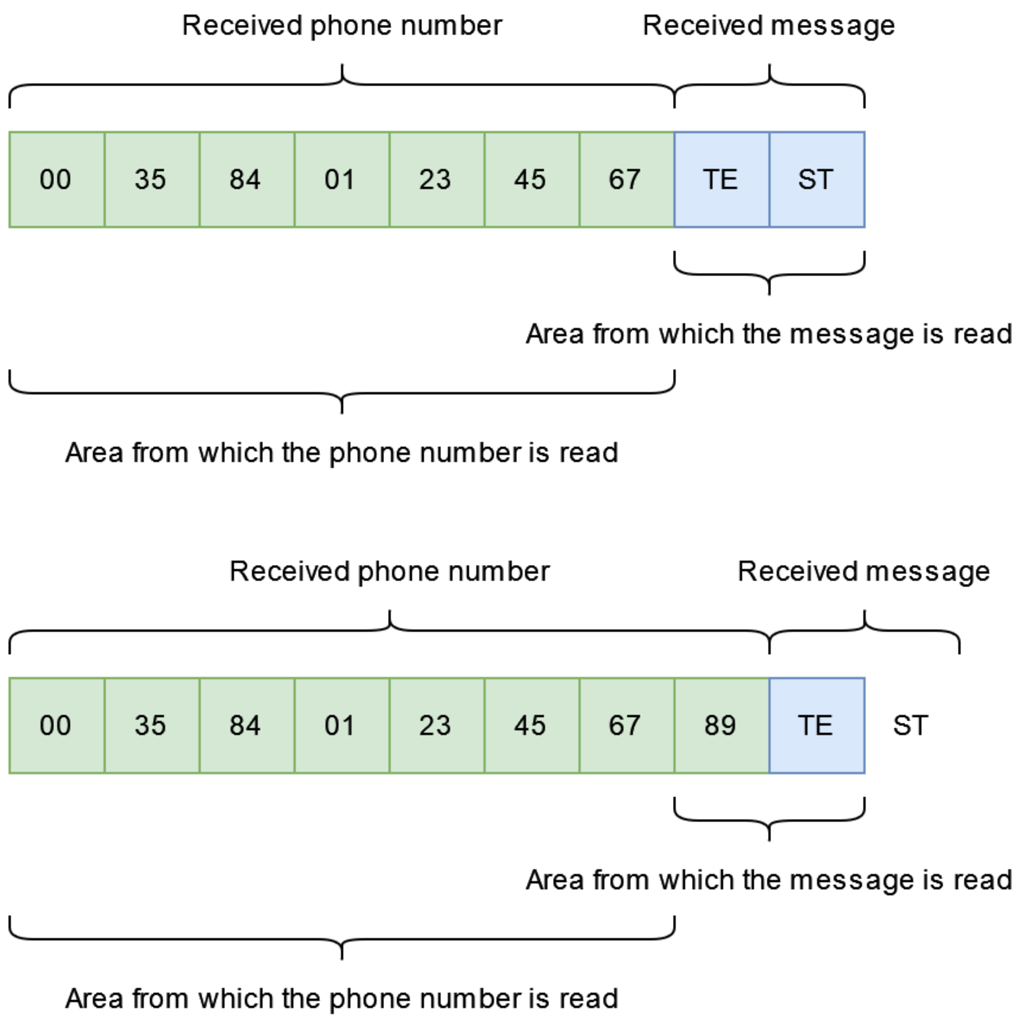

Received phone number
Received message
00
35
84
01
23
45
67
TE
ST
Area from which the message is read
Area from which the phone number is read
Received phone number
Received message
00
35
84
01
23
45
67
89
TE
ST
Area from which the message is read
Area from which the phone number is read

Figure 27. Contents of custom Modbus register block with 14-character-long and 16-character-long phone numbers.

SMS_ACK_READ should be modified so that it takes actions based on the entire Modbus register block, rather than reading the phone number and message content from certain register areas.

4.2.6 Other Missing Features

The serial SMS plug-in provides over 100 error codes related to the modem, which are useful in error diagnostics. In contrast, TRB140's Modbus service provides a limited information related to the device’s modem. If more information is needed from the modem, a shell script running on TRB140 could output more data to the custom Modbus register block by utilizing gsmctl commands, as these commands provide some additional information regarding the modem state [16].

The pre-existing control commands from serial SMS plug-in, such as turning the SMS alarm forwarding on or off, were not programmed, but should be relatively simple to add afterwards this would only require a few If-then SCIL statements to SMS_ACK_READ.

The new plug-in does not offer an alarm buffer for a situation where multiple alarms are generated. It is, however, probably a non-critical feature as the performance of the plug-in proved to be adequate during the testing with a proper network type and signal strength.

The serial SMS plug-in’s VSO dialog, which is used in Monitor Pro MicroSCADA user interface, was not modified. This means that the new plug-in can only be used in Workplace X interface. Using the new plug-in in a Monitor Pro environment would require modifying the code of the VSO dialog. As both the View and VSO dialog operate in similar ways, this would likely only require adding a few lines of SCIL to the “Send SMS” button of the VSO dialog to be able to work with the new plug-in.

5 Cybersecurity of SMS Alarm Plug-in

Modbus offers no authentication, encryption or integrity [24]. It is then important to assess what risks using this new SMS alarm plug-in can pose and what can be done to mitigate the risks.

There a few ways Modbus can be compromised. As Modbus does not offer authentication, any Modbus master can write commands to the Modbus slave, making unauthorized command executions. If a malware were to infect the network, it could possibly send malicious messages to the slave. Executing a Denial-of-Service as master could flood the Modbus slave making it inoperable. Man-in-the-Middle attacks stem from the lack of integrity, and an attacker can intercept and modify the messages before passing them on to the server. [25.]

The worst cyber-attack from the perspective of TRB140’s Modbus service might be unauthorized command execution. The attacker can for example reboot the modem, change the LAN IP, send text messages, and write values the custom Modbus registers. If an attacker knew the operators phone number and the acknowledgement commands, no actual text message would be needed in order to acknowledge alarms. If the text message reading command procedure would be made to support custom control commands such as controlling a critical device on the field, it could result in catastrophic consequences if an unauthorized entity wrote values to these Modbus registers. It is then important to assess whether a user would need the new SMS alarm plug-in using Modbus protocol at all if there were inherent cybersecurity risks in the SCADA network. Custom control commands should probably not be implemented in the command procedure to limit the possible damage to the SCADA system and network. At the very least, the control commands for the plug-in should be classified and be customer specific. However, this would probably not be very useful if there was an internal attack in the SCADA network, as the attacker could inspect the Modbus traffic. To minimize the security risks of Modbus is to design a secure SCADA network.

As mentioned above, TRB140 cannot fetch inbound caller’s number and making constant calls to TRB140 hinders its performance significantly. To mitigate these risks from cybersecurity perspective, the mobile subscription phone number should be hidden from directory enquiries and the SCADA operators should not leak the phone number to third parties. In case the phone number gets leaked, the alarm plug-in should be turned off immediately and a new mobile phone subscription should be acquired.

Another possibly useful security measure for when unknown numbers send text messages to the SMS interface, is that it should be trigger a SCADA alarm containing the unknown phone number. This would keep the operators informed and to take appropriate actions if someone tries to interfere with the SMS alarm plug-in.

TRB140 software updates should be taken care of, just like other software updates. If there was a vulnerability in the current software version, it must be updated to the latest software version. TRB140 has unnecessary features from the perspective of the new SMS alarm plug-in, such as mobile data connection which allows the host computer to connect to the Internet and should be disabled before installing TRB140 to SCADA. This procedure is presented in the installation manual (Appendix 2, 6). There are probably other features in TRB140 which should be disabled but they were not researched further in this thesis work.

6 Conclusion

The result of this thesis work is a TCP/IP-based SMS alarm plug-in for MicroSCADA which supports the most essential features of the currently used plug-in. Installation manuals and packages were developed, which engineers can use in the deployment of the plug-in.

To have a fully featured SMS alarm plug-in, the new solution should be developed further due to the missing features and bugs. These were documented and the possible solutions or workarounds for these were presented. Some of these features and bugs are not possible fix such as making outgoing calls. However, if the previous feature would be needed, Teltonika’s development team could implement this as it is most likely not a technical limitation. It is also possible that the bugs and other lacking features could be fixed in future device firmwares.

Due to cybersecurity issues mentioned above, the new plug-in’s usage in a customer system can be questionable. Cybersecurity has become a critical aspect not only in industrial systems but in everyday life and cyberattacks are done now more than ever. Before deploying this plug-in in a customer system, the cybersecurity aspect should be researched further.

References

1 Triggs Robert. 2023. What is SMS and how does it work? [online]. Android Authority. URL: https://www.androidauthority.com/what-is-sms-280988/. Accessed 17 August 2023.

2 Parziale Lydia, Liu Wei, Matthews Carolyn, Rosselot Nicolas, Davis Chuck, Forrester Jason, Britt David T. 2006. TCP/IP Tutorial and Technical Overview. IBM Redbooks.

3 Modbus organization. 2012. Modbus Application Protocol Specification V1.1b3. [online]. URL: https://modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf. 24 October 2023.

4 Modbus organization. Modbus Messaging On TCP/IP Implementation Guide V1.0b. 2006. [online]. URL: https://modbus.org/docs/Modbus_Messaging_Implementation_Guide_V1_0b.pdf. 24 October 2023.

5 Boyer Stuart A. 2004. SCADA: Supervisory Control and Data Acquisition. The Instrumentation, Systems, and Automation Society.

6 Hitachi Energy. 2022. MicroSCADA X SYS600 10.4 System Configuration. Manual.

7 ABB Oy, Substation Automation Products. 2002. Introduction to MicroSCADA Technology. Manual.

8 Hitachi Energy. 2022. MicroSCADA X SA Engineering W301. Presentation.

9 Hitachi Energy. 2022 MicroSCADA X SYS600 10.4 Application Objects. Manual.

10 Hitachi Energy. 2022 MicroSCADA X SYS600 10.4 System Objects. Manual.

11 Hitachi Energy. 2022 MicroSCADA X SYS600 10.4 Programming Language SCIL. Manual.

12 Teltonika Networks. Product Catalog 2023 [online]. URL: https://teltonika-networks.com/cdn/pages/2022/10/63479480a08ff8-64018958/teltonika-networks-product-catalog-2023-v11.pdf. Accessed 8 August 2023.

13 Teltonika Networks. TRB140 Industrial Rugged LTE Gateway [online]. URL: https://teltonika-networks.com/products/gateways/trb140. Accessed 13 August 2023.

14 Teltonika Networks. TRB140 [online]. URL: https://wiki.teltonika-networks.com/view/TRB140. Accessed 13 August 2023.

15 Teltonika Networks. QSG TRB140 [online]. URL: https://wiki.teltonika-networks.com/view/QSG_TRB140. Accessed 9 September 2023.

16 Teltonika Networks. Gsmctl commands [online]. URL: https://wiki.teltonika-networks.com/view/Gsmctl_commands. Accessed 18 May 2023.

17 Teltonika Networks. TRB140 Mobile Utilities. https://wiki.teltonika-networks.com/view/TRB140_Mobile_Utilities [online]. Accessed 4 May 2023.

18 Teltonika Networks. TRB140 Modbus [online]. URL: https://wiki.teltonika-networks.com/view/TRB140_Modbus. Accessed 18 May 2023.

19 Teltonika Networks. Modbus TCP Slave Send SMS example [online]. URL: https://wiki.teltonika-networks.com/view/Modbus_TCP_Slave_Send_SMS_example. Accessed 6 June 2023.

20 Hitachi Energy. 2022. MicroSCADA X SYS600 10.4 Communication Programming Interface (CPI). Manual.

21 Hitachi Energy. 2022. MicroSCADA X SYS600 10.4 Modbus Master Protocol. Manual.

22 Teltonika Networks. 2023 TRB140 modbus custom register block [online]. URL: https://wiki.teltonika-networks.com/view/TRB140_modbus_custom_register_block. Accessed 8 August 2023.

23 Teltonika Networks. 2023. TRB140 Input/Output [online]. URL: https://wiki.teltonika-networks.com/wikibase/index.php?title=TRB140_Input/Output. Accessed 13 September 2023.

24 Nardone Roberto, Rodríguez Ricardo J., Marrone Stefano. 2016. Formal security assessment of Modbus protocol. 142-147. 10.1109/ICITST.2016.7856685.

25 Fovino, Igor Nai, Andrea Carcano, Marcelo Masera and Alberto Trombetta. 2009. Design and Implementation of a Secure Modbus Protocol. Critical Infrastructure Protection.

Command Procedure Prototype for Sending Text Messages

```
#LOCAL num, msg, i

num = "00358401234567" ; recipient's phone number
msg = "Alarm" ; SMS message

; add null character if the message length is not an even number
; Otherwise RTU_INT will not work correctly
#IF LENGTH(msg) MOD 2 == 1 #THEN #BLOCK
   msg = PAD("'msg'", ASCII(0), LENGTH(msg)+1)
#BLOCK_END

; split contents for RTU_INT
num = UNPACK_STR(num,2)
msg = UNPACK_STR(msg,2)

; text conversion (for phone number)
; RTU_INT function: text -> hex -> integer. e.g. "00" -> 3030 -> 12236
#LOOP_WITH i = 1..LENGTH(num)
   num(i) = RTU_INT(num(i))
#LOOP_END

; add zeroes to "Add SMS" registers until the first 10 are filled
; first 10 registers are reserved for phone number
#LOOP_WITH i = LENGTH(num)+1..10
   num(i) = 0
#LOOP_END

; text conversion (for SMS message)
#LOOP_WITH i = 1..LENGTH(msg) ;
   msg(i) = RTU_INT(msg(i))
#LOOP_END

#SET sta1:sdv(1397..1486) = APPEND(num, msg) ; write SMS to registers
#SET sta1:sdv396 = 1 ; send SMS
```

Installation Manual for the Plug-in

Installation of TCP/IP-based SMS alarm plug-in

Contents

1 General

This manual will cover installation procedure of TCP/IP based SMS alarm plug-in for a single system MicroSCADA X SYS600. The plug-in and manual have been made for Teltonika Networks TRB140 (https://teltonika-networks.com/products/gateways/trb140). TRB140 supports sending and reading SMS via Modbus TCP. The alarm plug-in has been tested in single system MicroSCADA version 10.5 with TRB140 "TRB1_R_00.07.04.5 | 2023.07.24" firmware. Although not likely, newer and older software versions could result in the plug-in not working correctly and should be tested before installing in a customer system. The plug-in has not been tested in a HSB system and is also not guaranteed to work correctly.

This manual covers and overview of TRB140's and MicroSCADA's configuration. Installation files which are used in TRB140 or MicroSCADA can be found from

[redacted]

2 TRB140 configuration

Most configurations should be done from a personal computer before installing the TRB140 in MicroSCADA network for an easier deployment.

Follow the instructions in this article https://wiki.teltonika-networks.com/view/QSG_TRB140 and/or the video linked in the article https://www.youtube.com/watch?v=Os4L3wDvAbY. These links cover the hardware installation and connection setup.

2.1 Login

After the device is powered on and plugged via USB, type the default IP address 192.168.2.1 to Web browser to access the device's WebUI. The default username is "admin" and the password "admin01" (figure 1).

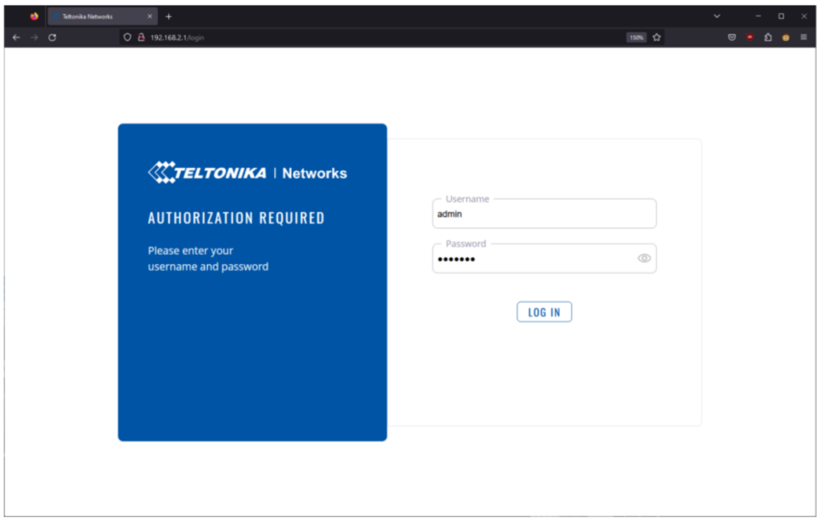
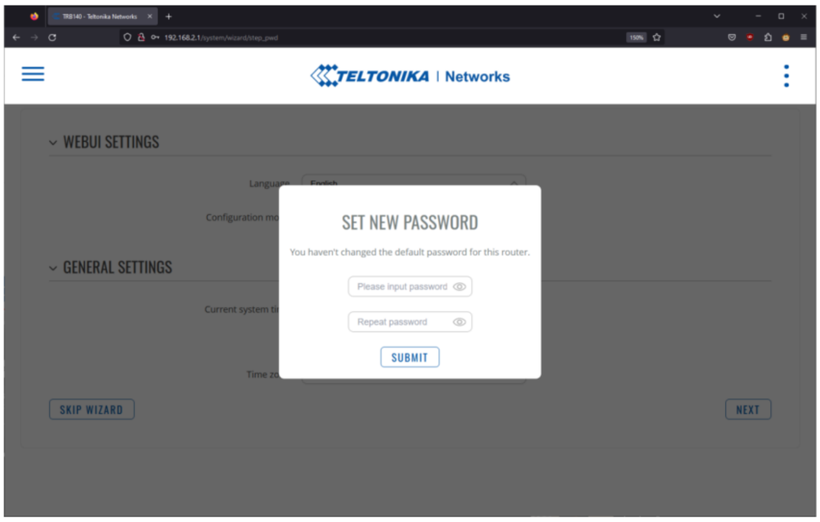


Figure 1. TRB140 log-in page

Insert new password according to security policies and press “Submit” (figure 2).



SET NEW PASSWORD

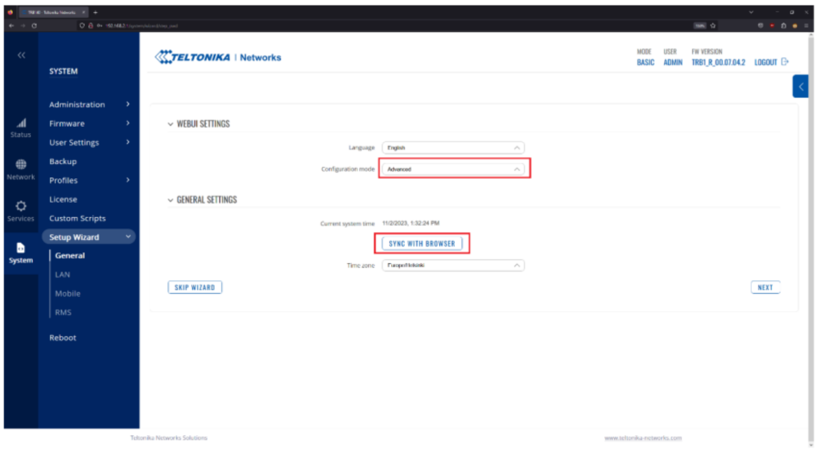
You haven't changed the default password for this router.

Figure 2. Password change

2.2 Setup wizard

Select “Configuration mode” as “Advanced” and click “Sync with browser” (figure 3).

4



MODE BASIC USER ADMIN FW VERSION TRB1_R_00.07.04.2 LOGOUT

WEBUI SETTINGS

Language: English

Configuration mode: Advanced

GENERAL SETTINGS

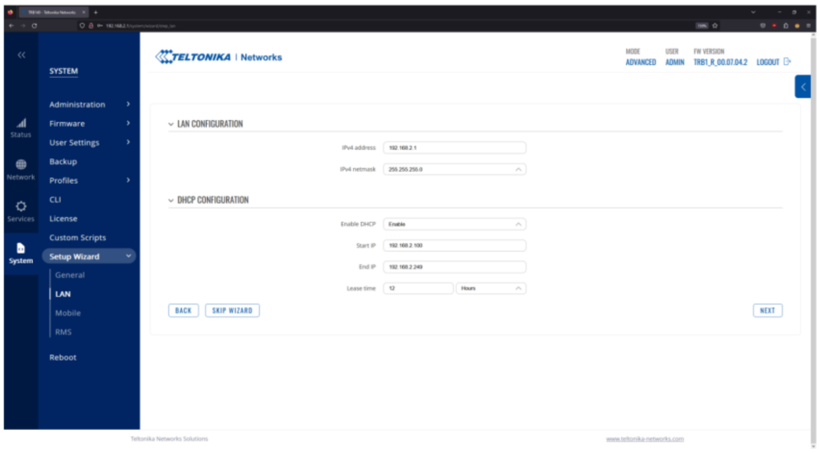
Current system time: 11/2/2023, 1:32:24 PM

SYNC WITH BROWSER

Time zone: Europe/Helsinki

Figure 3. Configuration mode change and time sync

IP address and DHCP settings can be changed from the next screen (figure 4). These can also be configured afterwards.



MODE ADVANCED USER ADMIN FW VERSION TRB1_R_00.07.04.2 LOGOUT

LAN CONFIGURATION

IPv4 address: 192.168.2.1

IPv4 netmask: 255.255.255.0

DHCP CONFIGURATION

Enable DHCP: Enable

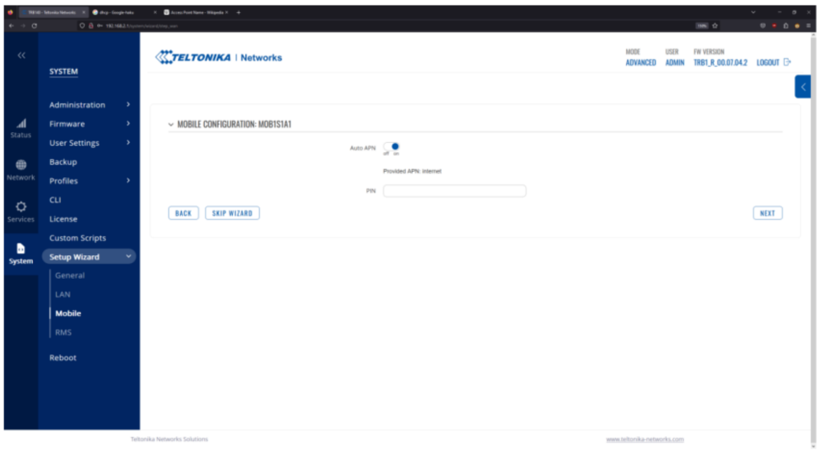
Start IP: 192.168.2.100

End IP: 192.168.2.249

Lease time: 12 Hours

Figure 4. Initial IP configuration page

“Auto APN” setting can be left as default as of now. Insert SIM card’s PIN code. (figure 5)



MOBILE CONFIGURATION: MOB1S1A1

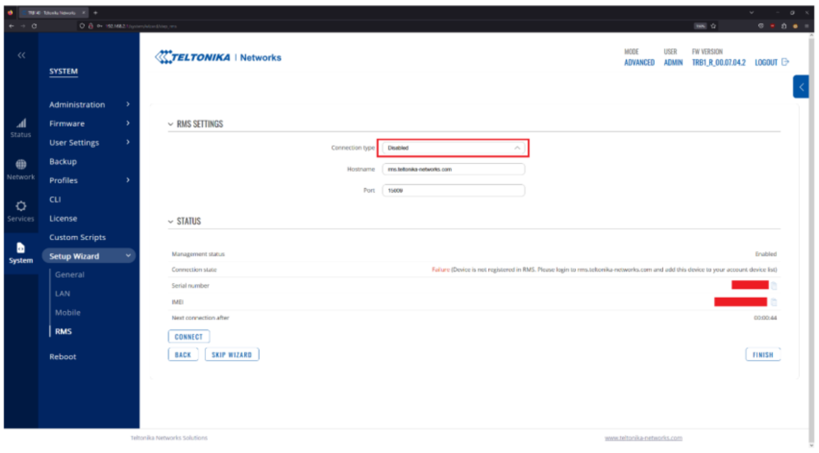
Auto APN: on

Provided APN: internet

PIN:

Figure 5. Mobile interface configuration page

Disable RMS service (figure 6). This is not a needed feature.



RMS SETTINGS

Connection type: Disabled

Hostname: rms.teltonika-networks.com

Port: 15009

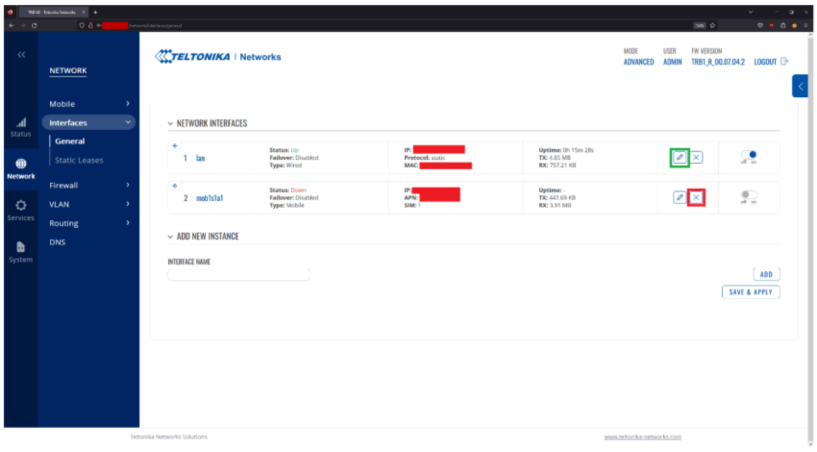
STATUS

Management status	Enabled
Connection state	Failure (Device is not registered in RMS. Please login to rms.teltonika-networks.com and add this device to your account device list)
Serial number	[redacted]
IMEI	[redacted]
Next connection after	00:00:44

Figure 6. Disabling of RMS service

2.3 Connection interface setup and IP configuration

After the previous steps, the WebUI is ready to be used. By default, TRB140 tries to initiate Internet connection and it is important to disable this feature before installing the device to MicroSCADA network. This is done by navigating to Network -> Interfaces -> General and removing “mob1s1a1” network interface and clicking the “X” button highlighted in red. (figure 7) LAN interface should not be disabled. Otherwise, the access to the device will be barred and TRB140 needs to be reset by pressing the physical button found in the device.



MODE ADVANCED
USER ADMIN
FW VERSION TRB1_R_00.07.04.2

NETWORK INTERFACES

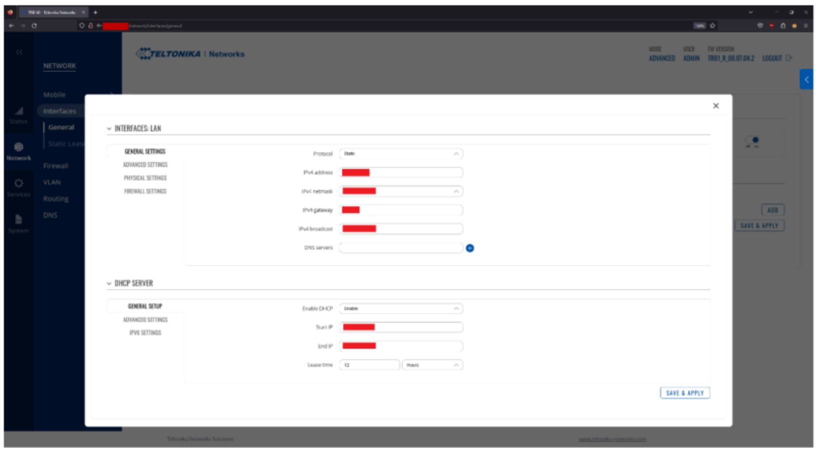
1	lan	Status: Up Failover: Disabled Type: Wired	IP: [redacted] Protocol: static MAC: [redacted]	Uptime: 0h 15m 28s TX: 4.85 MB RX: 757.21 KB
2	mob1s1a1	Status: Down Failover: Disabled Type: Mobile	IP: [redacted] APN: [redacted] SIM: 1	Uptime: - TX: 447.69 KB RX: 3.91 MB

ADD NEW INSTANCE

INTERFACE NAME

Figure 7. Interface configuration page

Configuring of LAN IP address and DHCP is done by clicking the button highlighted in green in the previous figure 7. The user is greeted with the IP address and DHCP settings (figure 8). Adjust these accordingly.



INTERFACES: LAN

Protocol	Static
IPv4 address	[redacted]
IPv4 netmask	[redacted]
IPv4 gateway	[redacted]
IPv4 broadcast	[redacted]
DNS servers	

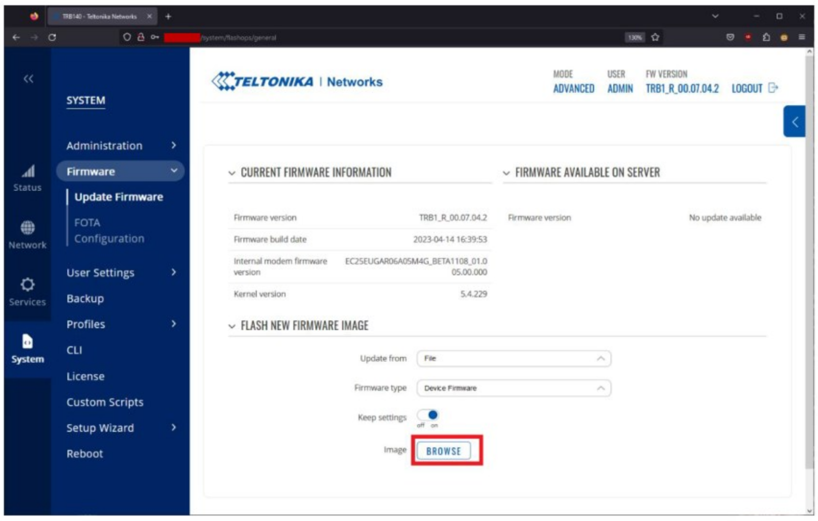
DHCP SERVER

Enable DHCP	Enable
Start IP	[redacted]
End IP	[redacted]
Lease time	12 Hours

Figure 8. Interface IP configuration dialog

2.4 Firmware update

TRB140’s firmware files can be found from https://wiki.teltonika-networks.com/view/TRB140_Firmware_Downloads. Download the correct firmware version. Go to TRB140’s WebUI and navigate to System -> Firmware -> Update Firmware. Select “Browse” from WebUI and upload the downloaded firmware file. (figure 9)



MODE: ADVANCED
USER: ADMIN
FW VERSION: TRB1_R_00.07.04.2

CURRENT FIRMWARE INFORMATION

Firmware version	TRB1_R_00.07.04.2
Firmware build date	2023-04-14 16:39:53
Internal modem firmware version	EC25EUGAR06A05M4G_BETA1108_01.0 05.00.000
Kernel version	5.4.229

FIRMWARE AVAILABLE ON SERVER

Firmware version	No update available

FLASH NEW FIRMWARE IMAGE

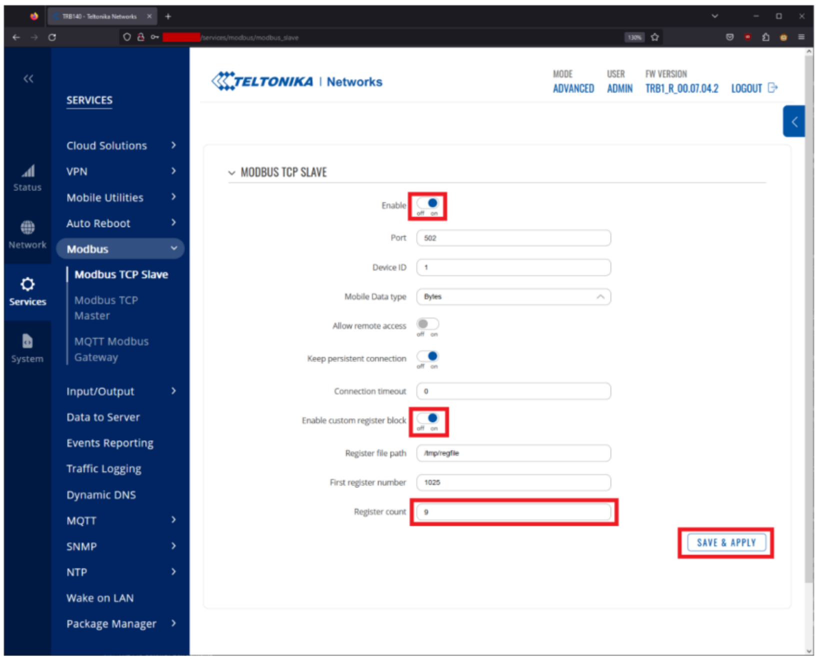
Update from: File
Firmware type: Device Firmware
Keep settings: on
Image: BROWSE

Figure 9. Firmware updating

2.5 Modbus

2.5.1 Enabling Modbus service

Modbus service needs to be enabled from Services -> Modbus -> Modbus TCP Slave. Enable Modbus slave option, custom register block and adjust the custom register block's register count to 9 and click "Save & Apply". (figure 10)



[redacted]/services/modbus/modbus_slave

MODE ADVANCED | USER ADMIN | FW VERSION TRB1_R_00.07.04.2

MODBUS TCP SLAVE

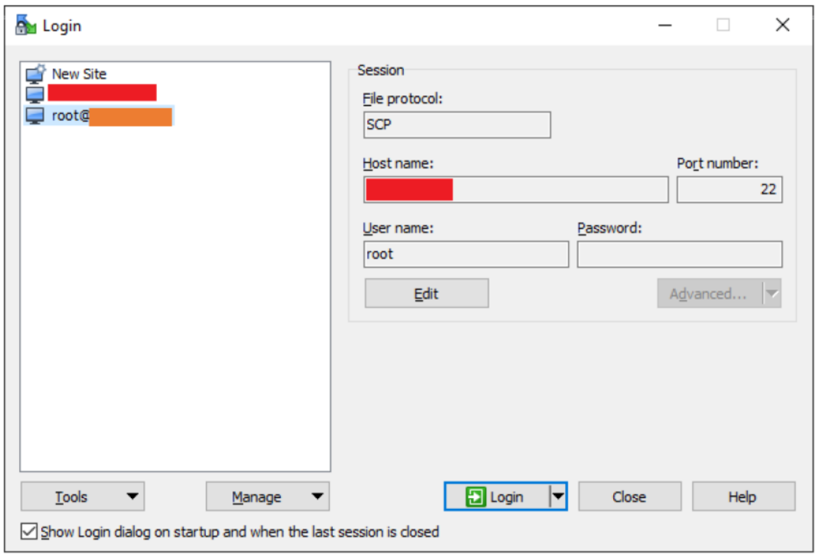
Enable	on
Port	502
Device ID	1
Mobile Data type	Bytes
Allow remote access	off
Keep persistent connection	on
Connection timeout	0
Enable custom register block	on
Register file path	/tmp/regfile
First register number	1025
Register count	9

Figure 10. Enabling of Modbus service and custom register block

2.5.2 Custom register block

User defined file path can be read from the custom register block. More information regarding this feature can be found from https://wiki.teltonika-networks.com/view/TRB140_modbus_custom_register_block. In this case, it is used to read received SMS. A shell script called “extramodbus” has been created that constantly scans received SMS’s and outputs the contents to /tmp/regfile, the path where the register contents are read.

After the shell script has been downloaded, initiate SCP connection with WinSCP, or similar program. The user name is “root” by and the password is the defined as in the device setup. (figure 11)



Session

File protocol: SCP

Host name: [redacted]

Port number: 22

User name: root

Password:

☑ Show Login dialog on startup and when the last session is closed

Figure 11. TRB140 log-in via WinSCP

Navigate to /bin folder. Drag the downloaded extramodbus to this folder. (figure 12)

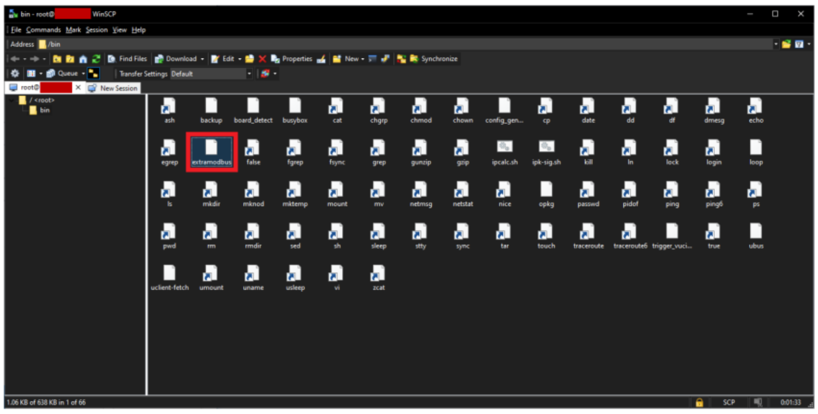
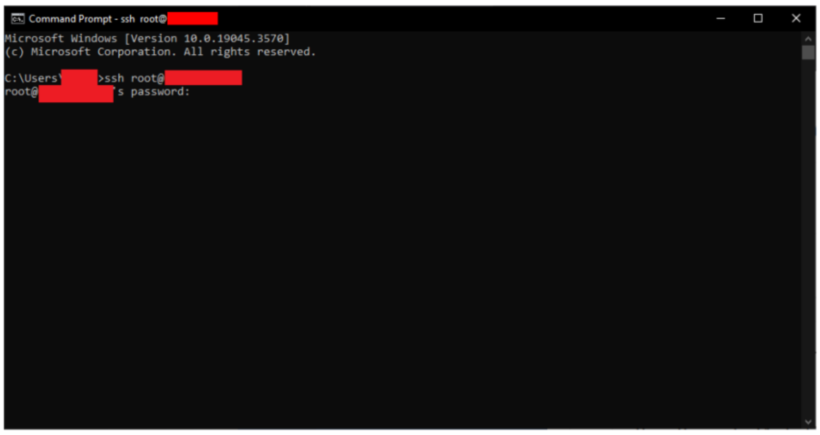


Figure 12. Adding extramodbus script to TRB140 via WinSCP

After this, initiate SSH connection with a suitable program or alternatively from WebUI found in System -> CLI. Initiating connection via Windows is done by opening cmd and typing "ssh root@ipaddress". After this type in the password. (figure 13)

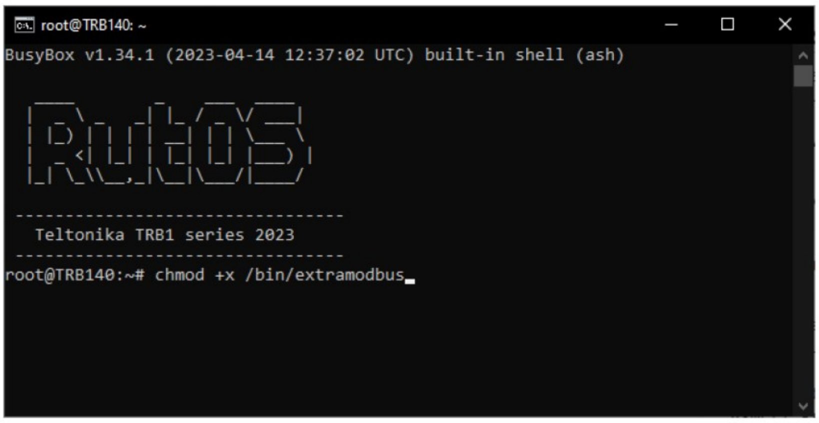


```
Microsoft Windows [Version 10.0.19045.3570]
(c) Microsoft Corporation. All rights reserved.

C:\Users\[redacted]>ssh root@[redacted]
root@[redacted]'s password:
```

Figure 13. TRB140 SSH log-in

Making the script executable is done with the following command: chmod +x /bin/extramodbus (figure 14).



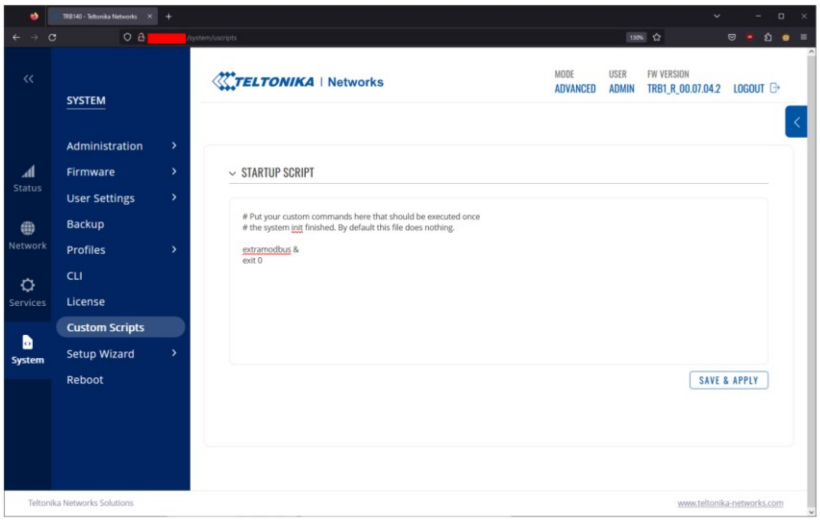
```
root@TRB140: ~
BusyBox v1.34.1 (2023-04-14 12:37:02 UTC) built-in shell (ash)

  ____        _    ___  ____
 |  _ \ _   _| |_ / _ \/ ___|
 | |_) | | | | __| | | \___ \
 |  _ <| |_| | |_| |_| |___) |
 |_| \_\\__,_|\__|\___/|____/

 --------------------------------
   Teltonika TRB1 series 2023
 --------------------------------
root@TRB140:~# chmod +x /bin/extramodbus
```

Figure 14. Enabling extramodbus shell script

SSH connection can now be closed. After the script is made executable, the script needs to start during the boot of TRB140. This is found from WebUI by navigating to System -> Custom Scripts. Insert “extramodbus &” before “exit 0”. (figure 15).



MODE ADVANCED
USER ADMIN
FW VERSION TRB1_R_00.07.04.2
LOGOUT

```
# Put your custom commands here that should be executed once
# the system init finished. By default this file does nothing.

extramodbus &
exit 0
```

Figure 15. Setting extramodbus to launch during the boot of the device

Reboot the device from System -> Reboot to make the script constantly run in the background.

At this point, TRB140 can be installed to MicroSCADA network.

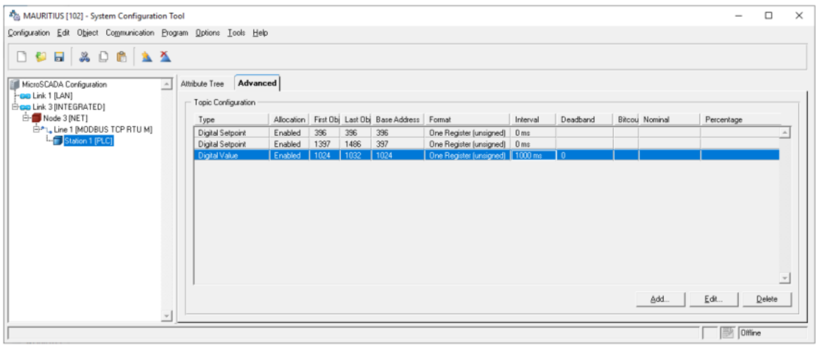
3 MicroSCADA configuration

3.1 System configuration

TRB140 needs to be configured as processs communication unit from MicroSCADA. It is assumed that the MicroSCADA system configuration does not contain a pre-existing Modbus TCP line or a station. Insert Modbus TCP line and a station under a NET node. The following Modbus parameters need to be configured as topics:

- "Add SMS". SMS message content. Requires 90 registers. Configure as Digital Setpoint.
- "Send SMS". This is used to send SMS message. Requires one register. Configure as Digital Setpoint.
- Custom register block. This is used to read received SMS messages. Requires 9 registers. Configure as Digital Value.

An example topic configuration is presented in figure 16.



Type	Allocation	First Obj	Last Obj	Base Address	Format	Interval	Deadband	Bitcou	Nominal	Percentage
Digital Setpoint	Enabled	396	396	396	One Register (unsigned)	0 ms				
Digital Setpoint	Enabled	1397	1486	397	One Register (unsigned)	0 ms				
Digital Value	Enabled	1024	1032	1024	One Register (unsigned)	1000 ms	0			

Figure 16. Example topic configuration

If different object addresses are used for the previous parameters, command procedures and SMS management page need also to be adjusted as these use the topic addresses as presented in figure 16.

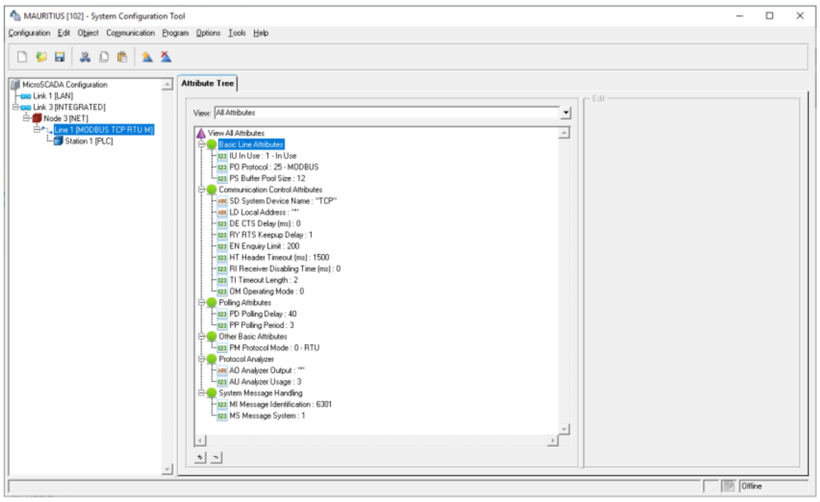
Other perhaps useful Modbus parameters related to TRB140 can be found from https://wiki.teltonika-networks.com/view/TRB140_Modbus.

Adjust the following settings in System Configuration tool:

Line:

- EN (Enquiry Limit): 200
- HT (Header Timeout [ms]): 1500

Example of Line configuration is presented in figure 17.


MAURITIUS [102] - System Configuration Tool
Configuration Edit Object Communication Program Options Tools Help
Attribute Tree
Edit
Offline

```
MicroSCADA Configuration
  Link 1 [LAN]
  Link 3 [INTEGRATED]
    Node 3 [NET]
      Line 1 [MODBUS TCP RTU M]
        Station 1 [PLC]
```

View: All Attributes

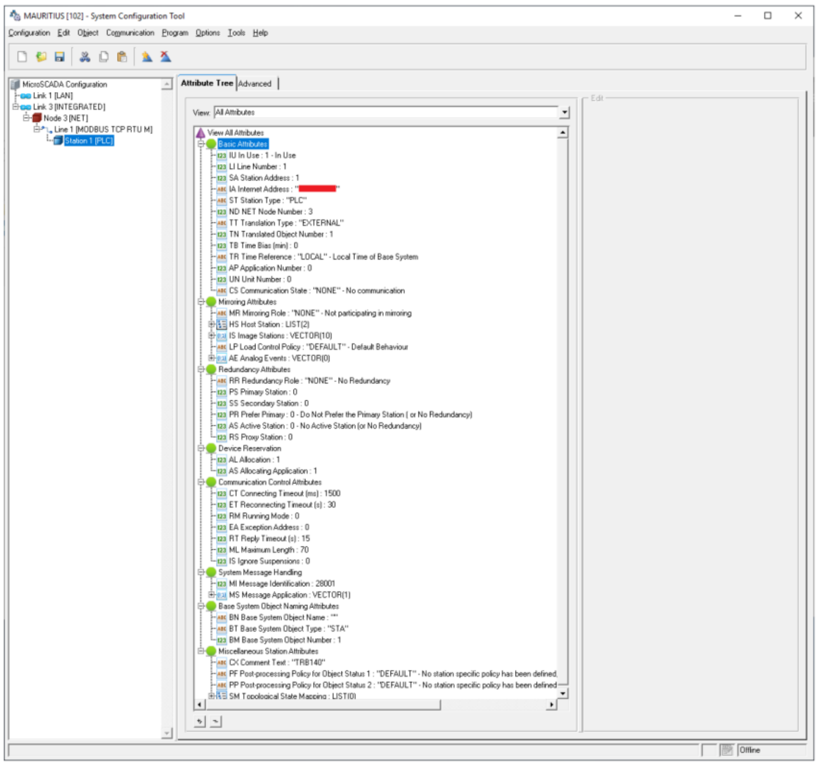
```
View All Attributes
  Basic Line Attributes
    IU In Use : 1 - In Use
    PO Protocol : 25 - MODBUS
    PS Buffer Pool Size : 12
  Communication Control Attributes
    SD System Device Name : "TCP"
    LD Local Address : ""
    DE CTS Delay (ms) : 0
    RY RTS Keepup Delay : 1
    EN Enquiry Limit : 200
    HT Header Timeout (ms) : 1500
    RI Receiver Disabling Time (ms) : 0
    TI Timeout Length : 2
    OM Operating Mode : 0
  Polling Attributes
    PD Polling Delay : 40
    PP Polling Period : 3
  Other Basic Attributes
    PM Protocol Mode : 0 - RTU
  Protocol Analyzer
    AO Analyzer Output : ""
    AU Analyzer Usage : 3
  System Message Handling
    MI Message Identification : 6301
    MS Message System : 1
```

Figure 17. Example line configuration

Station:

- IA (Internet address): <TRB140 IP address>
- ML (Maximum length): 70

Example of Station configuration is presented in figure 18.


MAURITIUS [102] - System Configuration Tool
Configuration Edit Object Communication Program Options Tools Help
Attribute Tree
Advanced
Edit
Offline

```
MicroSCADA Configuration
  Link 1 [LAN]
  Link 3 [INTEGRATED]
    Node 3 [NET]
      Line 1 [MODBUS TCP RTU M]
        Station 1 [PLC]
```

View: All Attributes

```
View All Attributes
  Basic Attributes
    IU In Use : 1 - In Use
    LI Line Number : 1
    SA Station Address : 1
    IA Internet Address : "[redacted]"
    ST Station Type : "PLC"
    ND NET Node Number : 3
    TT Translation Type : "EXTERNAL"
    TN Translated Object Number : 1
    TB Time Bias (min) : 0
    TR Time Reference : "LOCAL" - Local Time of Base System
    AP Application Number : 0
    UN Unit Number : 0
    CS Communication State : "NONE" - No communication
  Mirroring Attributes
    MR Mirroring Role : "NONE" - Not participating in mirroring
    HS Host Station : LIST(2)
    IS Image Stations : VECTOR(10)
    LP Load Control Policy : "DEFAULT" - Default Behaviour
    AE Analog Events : VECTOR(0)
  Redundancy Attributes
    RR Redundancy Role : "NONE" - No Redundancy
    PS Primary Station : 0
    SS Secondary Station : 0
    PR Prefer Primary : 0 - Do Not Prefer the Primary Station ( or No Redundancy)
    AS Active Station : 0 - No Active Station (or No Redundancy)
    RS Proxy Station : 0
  Device Reservation
    AL Allocation : 1
    AS Allocating Application : 1
  Communication Control Attributes
    CT Connecting Timeout (ms) : 1500
    ET Reconnecting Timeout (s) : 30
    RM Running Mode : 0
    EA Exception Address : 0
    RT Reply Timeout (s) : 15
    ML Maximum Length : 70
    IS Ignore Suspensions : 0
  System Message Handling
    MI Message Identification : 28001
    MS Message Application : VECTOR(1)
  Base System Object Naming Attributes
    BN Base System Object Name : ""
    BT Base System Object Type : "STA"
    BM Base System Object Number : 1
  Miscellaneous Station Attributes
    CX Comment Text : "TRB140"
    PF Post-processing Policy for Object Status 1 : "DEFAULT" - No station specific policy has been defined,
    PP Post-processing Policy for Object Status 2 : "DEFAULT" - No station specific policy has been defined
    SM Topological State Mapping : LIST(0)
```

Figure 18. Example station configuration

Save system configuration, restart PC-NET, and if necessary, restart MicroSCADA system.

3.2 Installation packages

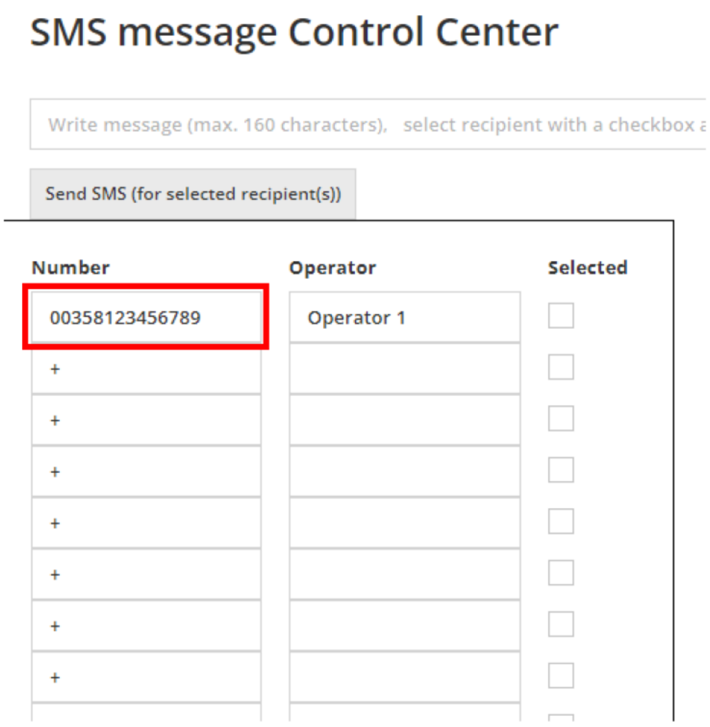
The installation packages for MicroSCADA contain application objects such as command procedures and process objects and a View page for SMS management. These can be found from [redacted] To install the plug-in to MicroSCADA system, the following steps are required:

- Insert **mod** folder to the root of **sc** folder. This contains the View page for SMS management
- Insert SMS_DATA.INI to "X:\sc\apl\<aplname>\PICT\"
- Adjust View file path for SMS_DATA.INI. This step allows the user interface to read and save operators' data
- Adjust View file "SET:SDVx(xxx...xxx)" SCIL statements according to your topic configurations, if necessary. This is required to send SMS to operators from the management page.
- Import application objects from SMS_MOD.sdb using Export/Import tool.
- Modify SMS_ACK_STATUS:PSS(1..9) (switch state) as 2 (automatic state)
- Open Test Dialog. Execute the following SCIL statement: #SET APL:BUV1 = vector(). This global variable / application object (APL:BUV1) is required for the alarm acknowledgement command procedure.
- Find APL_INIT_2 command procedure. Insert the following SCIL statement: #SET APL:BUV1 = vector(). The global variable now initializes during the boot of the MicroSCADA system.
- Connect SMS_ALARM_MOD command procedure to APL_ALARM event channel. This command procedure activates whenever APL_ALARM detects a change in alarm state.

4 Testing and usage

After these steps, you can insert your own phone number to the SMS management page and test whether the plug-in sends text messages. You should test the "Send SMS" from View page and trigger a simulated alarming object from the process database. You should also ensure that the reminder SMS command procedures and the acknowledgement work.

Note that the phone number's "+" sign needs to be replaced with "00" as seen in figure 19. When inserting new phone numbers or operator names, press "Save" and refresh the page.



SMS message Control Center

Write message (max. 160 characters), select recipient with a checkbox a

Send SMS (for selected recipient(s))

Number	Operator	Selected
00358123456789	Operator 1	☐
+		☐
+		☐
+		☐
+		☐
+		☐
+		☐
+		☐

Figure 19. Phone number format for SMS message control page