Navid Ahmad

Beacon Reader Simulator Software Implementation on Raspberry Pi

Metropolia University of Applied Sciences

Bachelor of Engineering

Electronics

Thesis

November 2018

Metropolia

Author Title Number of Pages Date	Navid Ahmad Beacon Reader Simulator Software Implementation on Raspberry Pi 33 pages + 1 appendices 28 November 2018
Degree	Bachelor of Engineering
Degree Programme	Degree Programme in Electronics
Professional Major	Electronics
Instructors	Iiro Koskinen, Software Designer Matti Fisher, Principal Lecturer

The purpose of this project was to design and implement a device capable of simulating beacon reader functionality. Beacon reader simulator can communicate with Automatic Selective Door Operation System (ASDO) and send a message according to the user's preference.

Beacon reader simulator software was designed using C/C++ and Qt framework. The software was implemented on a Raspberry PI 3 B+. A beacon file template was created to configure and customize beacon data. The simulator software was designed such that when it is started the beacon data is extracted from the file. User interface is displayed on the touchscreen and user can access the simulator. The serial communication was done through a USB-RS485 cable and the selected message can be sent through it.

The simulator was tested to see how it performs and how ASDO interacts with it. The results showed that simulator follows the actual beacon reader protocol and its core functionality was replicated.

Keywords	Qt, UART, Graphical User Interface

Metropolia

Contents

List of abbreviations and definitions

ASDO – Automatic Selective Door Operation

GUI – Graphical User Interface

HMI – Human Machine Interface

CRC – Cyclic Redundancy Check

USB – Universal Serial Bus

GPS – Global Positioning System

List of figures and tables

1 Introduction

Train operating companies are using longer trains on the existing infrastructure to reduce operation costs and increase the number of passengers [1]. As such it leads to situations where the length of the station platform does not match the length of the train at some stations along the trains route. This presents a risk for the passengers if the train doors are opened on the wrong side and part of the train is outside the platform. Automatic Selective Door Operation Systems (ASDO) are implemented to contain mechanisms that automate the selecting of the doors according to the platform length and side [2]. This prevents doors being released when there is no platform available.

EKE Electronics has developed an ASDO system which uses global positioning system (GPS), distance travelled between stations and trackside beacons to detect the station. These trackside beacons are a beam powered RFID device used in the UHF radio frequency band [3] and containing the station information. Beacon is read using an interrogation device used in the UHF radio frequency band known as beacon reader [4]. Beacon readers are mounted on the train and communicates with ASDO. As the train goes over the beacon, it is energized by the beacon reader and station platform information is read. The reader then transfers the contents of the beacon, which is processed by ASDO. The information is then displayed to the driver to be verified.

Testing ASDO software functionality with beacons can be a challenge. The beacon cannot be programmed in house without a beacon programming device as such, multiple stations platform cannot be tested without reprogramming the beacon. Furthermore, it is not possible to test scenarios when beacon reader is faulty and how ASDO respond to such an event. Therefore, the aim of this project was to develop a beacon reader simulator for ASDO.

2 ASDO Overview

2.1 Automatic Selective Door Operation System

Automatic Selective Door Operation system (ASDO) automates the functionality of train door operation. ASDO is developed primarily for scenarios where the platform is shorter than the train. Figure 1 illustrates a scenario where ASDO is not used. ASDO only enables doors which accommodates the platform length. ASDO improves passenger safety by preventing the doors to be opened on wrong side and/or outside the station platform.

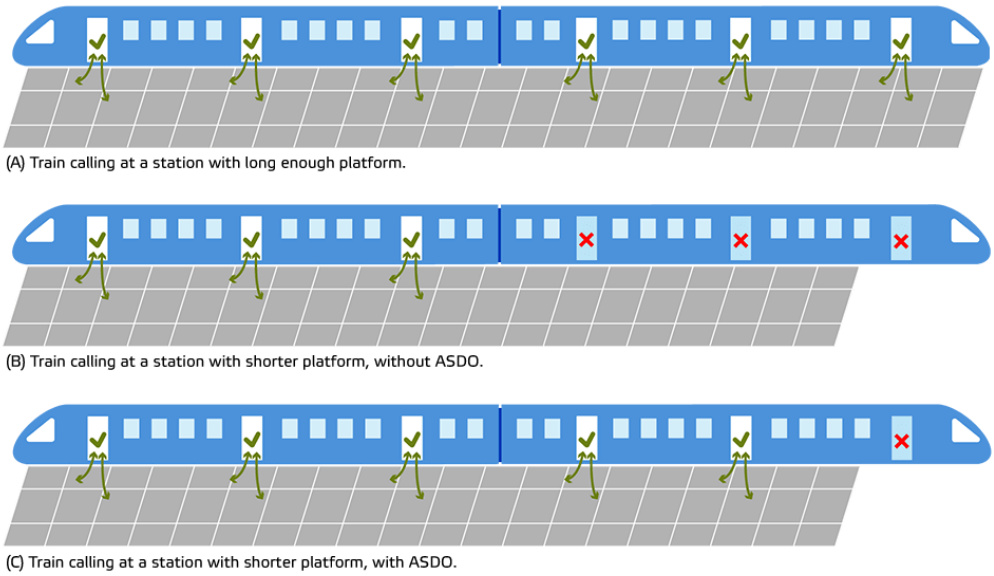


Figure 1. ASDO application on a platform shorter than the train [5]

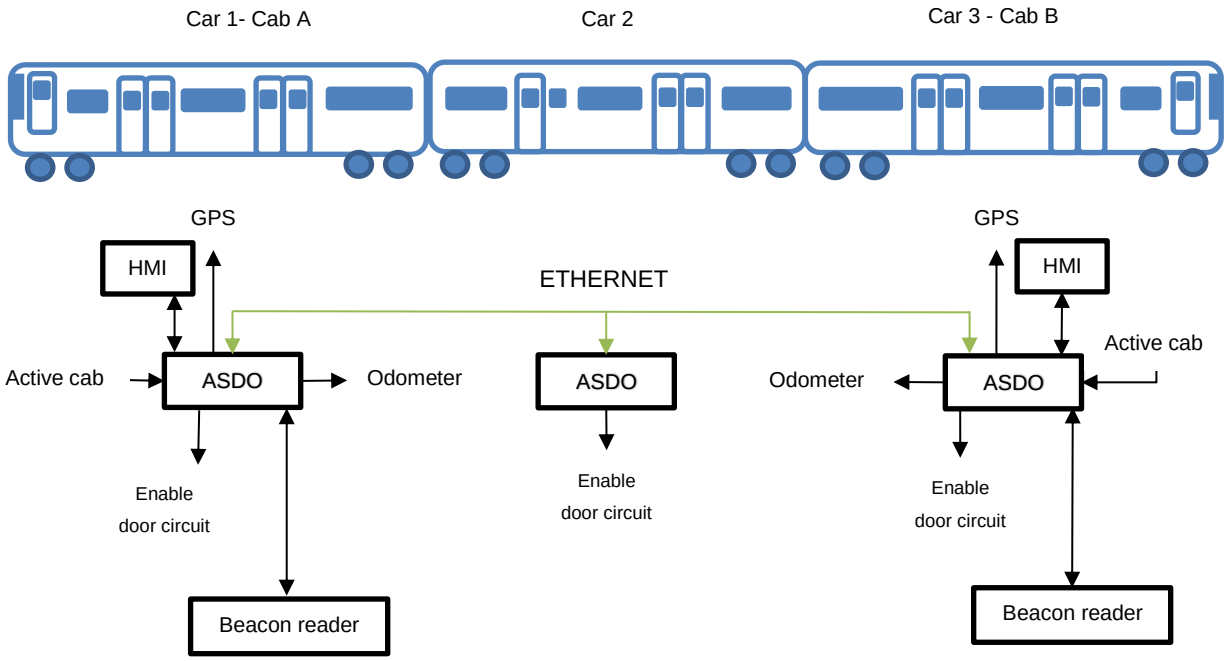


Figure 2. ASDO system configuration within a train unit [2]

The figure 2 displays how EKE-Electronics ASDO is usually set within a train unit. The HMIs are installed on the cabs where the train driver is located. The active cab is set when the driver key is inserted, and the signal is sent to ASDO. Active cab determines the leading car and the direction the train is moving to. Each car contains ASDO rack that is connected to the door control system. Train configuration is defined in the configuration database which is required by ASDO to detect the length of the train and car in addition to door position.

The driver or guard can use the HMI to interact with ASDO. HMI displays the station information, ASDO status and diagnostics data. Through the HMI the user can select a station and its platform. The user can also override the train doors if required.

GPS, Odometer and station database are used to determine train location. GPS receiver provides the GPS speed and position data. Odometer is a pulse sensor which measures the pulses per wheel revolution. ASDO then converts the pulses to distance travelled based on the train wheel diameter. Station database contains the location detail of every station and its platform in addition to the distance between stations in the rail network.

Station can be detected by matching the coordinates and distance travelled between stations.

§In addition to the train location ASDO can use a beacon to detect the correct station and platform so the correct doors can be opened. This can be applied for example on the terminal stations with different platform length, as shown in figure 3. Each station platform has a beacon/s. When the train drives over the beacon, the beacon reader scans the beacon and the station data is transferred to ASDO.

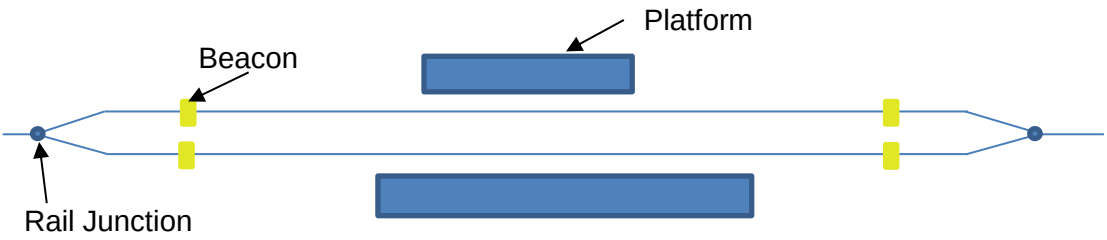


Figure 3. Terminal station with different platform length [5]

2.2 Beacon Reader

2.2.1 Beacon Reader System Functionality

Radio Frequency identification (RFID) uses radio waves to read and collect information stored within a tag or device. EKE-Electronics ASDO uses Tracklink III® Tag or beacon (figure 4) which is a beam powered RFID device used in the UHF frequency [3]. By default, a beacon can be factory programmed or it can be programmed by the user with a Tracklink III® beacon programmer [3]. Beacon contains station code, platform number, platform length, selective door operation (on/off), approach direction and correct side door to be enabled.

Beacons are passive RFID devices which are energised when the beacon is within a radio frequency field generated by Tracklink III® beacon reader (figure 4). It is an interrogation device used in the UHF radio frequency band which provides the radio field to the beacon [4]. The beacon contains an internal circuit and antenna which is activated by the radio field or beam generated by the beacon reader.

The activated beacon then transmits an encoded signal of the beacon data to the reader. The beacon reader decodes the signal and stores the information. It then transfers the information through RS485 communication link when requested by ASDO.

a) Beacon b) Beacon reader

Figure 4. Tracklink III® beacon communication system [2]

There are two modes of operation between ASDO and beacon reader unit which are Request and Reset mode [6,5]. During request mode ASDO polls the beacon reader and it responds with data stored in its buffer, if there is data available. Otherwise reader responds that the requested data is not available. Reset mode occurs when ASDO polls the reader to delete stored data and it responds to ASDO with a reset confirmation.

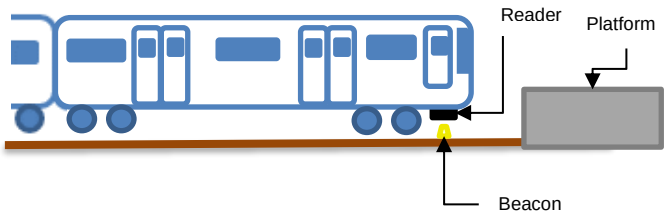


Figure 5. Train unit reading beacon

ASDO continuously requests for the beacon data and reader responds the status via RS485 communication link. If there is no beacon data available, the reader responds that data is unavailable. When the train arrives at the station the beacon is scanned (figure 5), and the data is stored in the reader buffer. The reader responds with confirmation that the requested beacon is available including the platform's data. The data is processed by ASDO to enable the correct doors to be opened. After the beacon data is received by ASDO, it requests the reader to delete the data and the reader responds with the delete confirmation. In case there are any internal faults within the beacon reader, it responds with an error message.

2.2.2 Poll Request Message Format

ASDO uses a poll request message to communicate between ASDO and the reader unit via the RS485 port. The poll request message frame is 17 bytes long and is constructed as illustrated in figure 6.

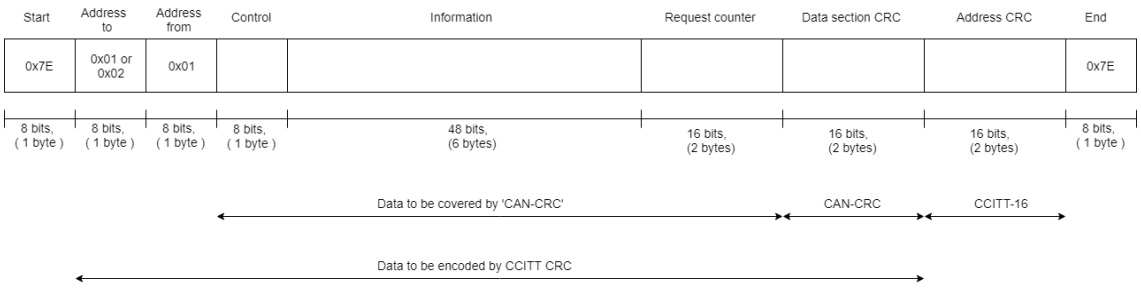


Figure 6. Poll request message format [6,7]

Poll request message start and end bytes have a fixed value of 0x7E which is used to define the message start and end. Address to section defines which reader unit the poll request message is sent to. Since there are usually two reader units in the train it is set as 0x01 or 0x02. Address from section refers to address where the poll request message is coming from. This value fixed to 0x01 because each reader unit is connected to one ASDO unit.

Control section is used to request status information from the reader unit. The possible Control byte values are expressed in hexadecimal format and used by ASDO is show in the table 1.

Table 1. Possible Poll request Telegram Control Byte Values [6,7]

Control byte value	**Request status information**
0x88	Request for beacon system page.
0x01 – 0x07	Request for one of beacon data page.
0x01	Request for beacon data page 1, standard setting for request beacon data.
0xD1 – 0xD7	Deletion of one of the beacon data pages.
0xD1	Deletion of beacon data page 1, standard setting for delete beacon data.
0xDA	Deletion of all beacon data pages.

The 48 bits in information section is set to logic “0” (see figure 6). This is because the poll request message is used for indicating the request by ASDO as such, no further information is required to activate a response from the beacon reader.

Current ASDO system uses the standard setting for request and delete beacon. This means that requests for beacon data page 1 (0x01) from the reader. Once ASDO receives the data it requests to delete page 1 (0xD1). Beacon reader stores the beacon data and only resets it after the delete request (0xD1 to 0xD7 or 0xDA) is received from ASDO.

Checksum calculations are carried out for the poll request telegram by ASDO. Checksums are used to ensure the integrity of the poll request message. Checksums are added to the poll request message before transmitting to beacon reader. Beacon reader then checks the checksum to determine if the data is valid. Cyclic redundancy check (CRC) is a form of checksum and is employed by the poll request telegram. There are two CRC encoding methods employed by the poll request message. The first method is CAN-CRC where the Control Byte, Information Bytes and Request Counter Bytes are encoded by using a 15-bit CRC and it is stored in the Data Section CRC [6]. CCIT-CRC is the second method where Address Bytes, Control Bytes, Request Counter Bytes and Data Section are encoded by using a 16-bit CRC stored in Address CRC [6].

2.2.3 Beacon Reader Reply Message Format

The beacon reader reply message is used to transfer information from reader RS485 port to ASDO. The poll request message frame is 17 bytes long (see figure 7).

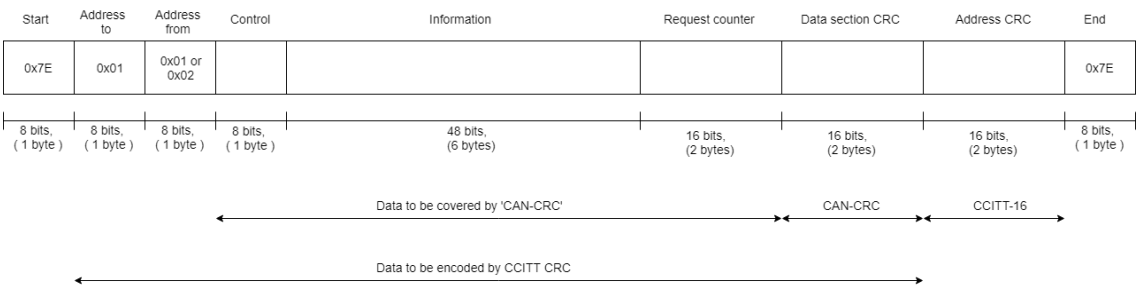


Figure 7. Beacon reader reply message format [6,8]

Beacon reader reply message format is the same as the poll request telegram. Only, the contents inside vary. The start and end section are set to 0x7E which is same as poll request message. The address to value section is set 0x01 because there is only one ASDO unit the reader is connected to. The Address from section value can be 0x01 or 0x02 because the reply can come from beacon reader one (0x01) or beacon reader two (0x02).

Control byte is used to reply the status of the beacon reader to ASDO. The possible Control byte values are expressed in hexadecimal format and used by the beacon reader is show in the table 2.

Table 2. Possible Beacon reader control Byte values [6,8]

Control Byte Value	Reply status information
0x88	Information contains the system page.
0x01 – 0x07	Information contains one of the beacon data pages.
0x01	Information contains data beacon page 1, standard setting.
0x3F	No Beacon data available.
0xFC	Condition mode signalling. This code will apply to either a system fault or a normal condition SPI timeout.

ASDO constantly requests for beacon page 1 (control byte 0x01), if data is unavailable, the reader responds with control byte 0x3F. The information section is set to logic “0”. When data is available the reader responds with control byte 0x01 and station information. The station information is constructed as shown in figure 8.

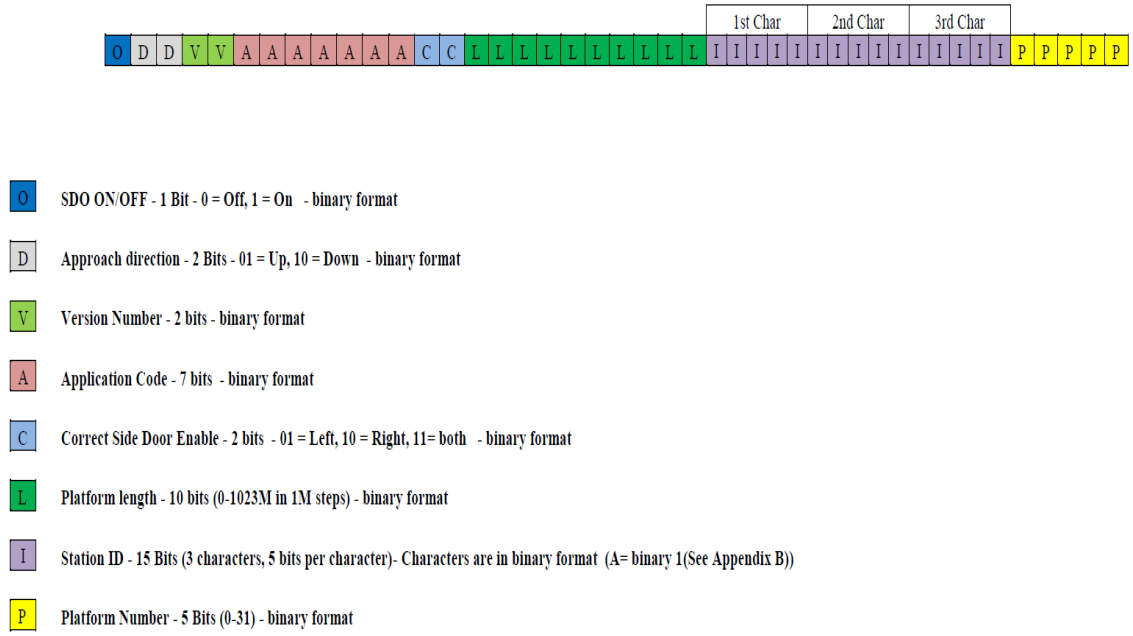


Figure 8. Beacon information [6,28]

If the reader detects a fault within its electronics, it uses the control byte to express the fault. Next time ASDO polls the reader, it responds with control byte 0xFC and the error code. The error code is set in the first byte of the information section in the reply telegram. The Error codes are expressed in hexadecimal format and their description are shown in the table 3.

Table 3. Beacon reader error codes [6,21]

Value	Description
0xC1	RFID module – Initialisation error.
0xC2	RFID module – Frequency error.
0xC3	Data Handler – RAM data register error.
0xC7	Data Handler – CPU Brown-out reset.
0xC9	Data Handler – Watchdog reset.
0xF7	Main CPU – Brown-out reset.
0xF9	Main CPU – Watchdog reset.
0xFA	Main CPU - Low voltage.
0xFC	Data handler communication error.

The same method of checksums calculations is implemented by the beacon reader which is mentioned in 2.2.2.

2.2.4 Serial Communication

Asynchronous serial interface is used to communicate between ASDO and the beacon reader. It is a form of communication where data is transferred one bit at a time without depending on an external clock signal. In order to have a clear and error free data transfer, the asynchronous serial protocol is dependent on four parameters as follows:

- Data bits
- Synchronization bits
- Parity bits
- Baud rate

The data is transferred in a well-defined frame which is a complete and non-divisible packet of bits. Data frame consists of the synchronization bits, data bits, and parity bits (see figure 9). Synchronization bits refer to start and stop bits which defines the beginning and end of a data packet. Parity bits are used to detect transmission errors. The parity bit is produced by the adding the 5-9 data bits and the evenness of the sum determines whether the bit is set or not. If the parity mode is even and the sum of data bits gives an odd number, the parity bit is set to 1. If the parity mode is set to odd and the sum of bits is odd the parity bit is set to 0. Baud rate is the speed of data transfer from transmitter and receiver in bits per second. The size of the data, parity and stop section can be configured (see figure 9). [7][8][9]

Figure 9. Serial data frame. [7]

Half duplex transmission mode is used to communicate between the beacon reader and ASDO. This transmission mode means that the data can be transmitted between the sender and receiver, but not at the same time. Serial Communication is done via two wire RS485. RS485 is the serial interface standard where data is sent in a differential pair, which allows greater distances and higher data rates than non-differential configuration such as RS232 [10].

3 Simulator System Design

3.1 Beacon Reader Simulator Requirements

There are limitations when using the beacon reader for ASDO testing. In a laboratory setting the beacons are scanned by swiping the beacon over the reader (see figure 10). At times it can be distracting because while scanning the beacon, the tester cannot check on ASDO to determine if the beacon data is received or not. The reader does not have LEDs or a display to indicate if beacon scan was a success or not. As such, it leads to a scenario where the beacon is scanned more than once, and the test needs to be done again.

Figure 10. Scanning beacon

Beacons are factory programmed and it cannot be programmed in house. It is not possible to reprogram the beacon to test all the stations in the rail network. To deal with this issue EKE-Electronics has added a beacon “overwrite” feature to ASDO. The tester creates a beacon file with the beacon data used in the rail network. Each beacon data will correspond to a station ID of a beacon which is available. When the beacon is scanned, ASDO checks the beacon ID and if it matches the station ID in the beacon file, ASDO processes the data according to the beacon file. The use of “overwrite” feature is still not a viable solution since the tester is limited by the number of beacons available in the lab and time is consumed writing the beacon file. Furthermore, the beacon “overwrite” feature is not added to ASDO which is delivered to the end-customer. ASDO needs to be certified without “overwrite” feature which is only used for debugging purposes. The

version of ASDO used for testing is not certified version so it is not good to depend on the beacon “overwrite” feature.

In section 2.2.3 table 3 shows error codes used to describe an internal fault within the beacon reader. These faults cannot be tested unless the reader is damaged. The beacon reader cannot be configured to provide the error codes because the system specification is not available. Overall the beacon reader is not flexible at all since all its features cannot be tested and tester has no control over what data is sent to ASDO. For example, Beacon reply message cannot be edited since it is constructed by the beacon reader. As such the effect of checksum errors, system faults and incorrect reply beacon frame size on ASDO cannot be tested.

The current beacon reader is limited because it is not easy to use and lacks configurability. A proper replacement is necessary, and it is possible to design a system which simulates the beacon reader functionality.

3.2 Hardware Implementation

Specific hardware is necessary to implement the beacon reader simulator software. The following components used:

- Embedded system
- Serial interface
- Display
- Input and outputs device
- Case

For the embedded system a Raspberry PI 3 B+ was used. Raspberry Pi 3 B+ is the latest single board computer (SBC) created by Raspberry Pi Foundation (see figure 11). Raspberry Pi runs on Debian based Linux distribution (Raspbian) which is provided by Raspberry Pi Foundation, as well as other third-party distributions e.g. Ubuntu, Windows IOT core etc. Raspberry Pi was used instead of PC, laptop or microcontroller due to its low cost, low power consumption and size. Furthermore, the device can fully support software functionality of simulator software. The file reader functionality can be developed since the user can easily create and modify the beacon file on Raspberry Pi. Beacon file

can be saved in a USB drive and file reader reads it from the USB port when it is connected. Qt supported on Raspberry Pi Linux platform the simulator user interface can be developed there.

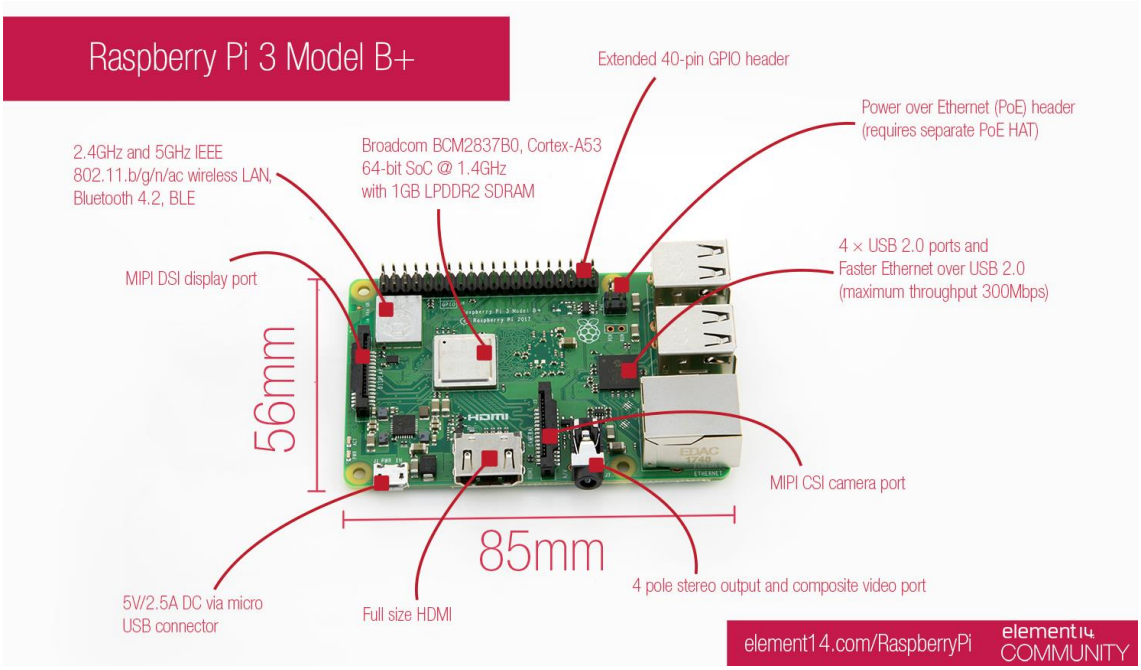
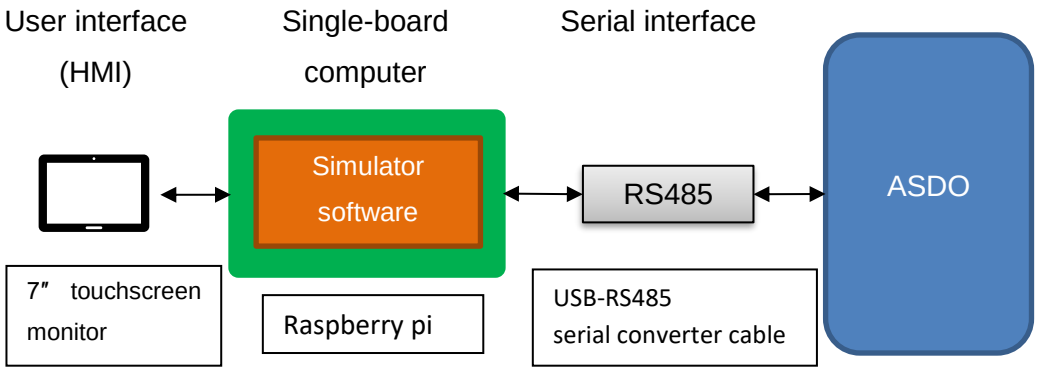


Figure 11. Raspberry Pi 3 B+ specifications [11]

The serial interface used is a USB to RS485 serial converter cable. This cable incoporates FT232RQ USB to serial UART interface IC device which handles all the USB signalling and protocols [14]. The USB end will be connected to Raspberry Pi USB port. A DB9 male connector will be added to the RS485 end and that end is connected the ASDO rack RS485 port. Serial communication functionality will initialize UART paraments and serial data communication.

A 7″ touchscreen monitor is used as human machine interface (HMI) for the end customer. The simulator software UI is displayed on the touchscreen and user can access the simulator. This way the user can select the type of beacon reply message by simply touching the screen.

The touch screen and Raspberry Pi is enclosed by a case. The figure 12 shows the beacon reader simulator system architecture.



Fiugure 12. Beacon reader simulator architecture

3.3 Software Implementation

The beacon reader software has the following functions:

- Beacon file reader
- Serial communication
- Poll request message analysis
- User interface

In order to simulate the beacon reader, the beacons need to be defined first. The current approach is to create a configuration file which contains the beacon information. The user will specify the beacon information and store it in the file. Application code, correct side door enables, platform length, station ID and platform number for each beacon must be defined by the user. Version number can be fixed to zero since it does not affect ASDO. Each line in the configuration file contains one beacon data. At start up the beacon configuration file is read.

Asynchronous serial interface is used to communicate between ASDO and the Beacon reader. The simulator software reads from and writes to RS485 port of ASDO. Serial communication function initialises and opens the serial port.

The poll request analysis is started after the serial communication is initialized and data is available from the serial port. During the poll request analysis, the poll request message from ASDO is read. The user can select the type of beacon reply message and

the message is constructed. When the simulator is polled by ASDO beacon reply message is sent.

The user interface (UI) displays the beacons which were stored in the beacon file. Also, the type of reply message can be selected. The system log screen is used to display the events occurred when using the simulator.

Qt is a cross-platform software framework for creating graphical user interfaces and applications. The simulator software is designed using the Qt framework.

3.3.1 Beacon File Reader

The beacons from the beacon file are extracted and stored in the beacon data structure. Beacon file is a comma-separated value (csv) file. A csv file is a text file that allows data to be saved in a table structured format. The data is separated by a semicolon and figure 13 illustrates the beacon file template.

```
#;;;;;;
# This file template is used to generate data for Hima-Sella TrackLink  III beacon simulator.;;;;;;
#;;;;;;
# Data format:  A;O;D;C;L;I;P;N
# A  -  Application code;;;;;;
# O  -  SDO ON/OFF:  0 = Off, 1 = On ;;;;;;
# D  -  Approach direction:  1 = Up (01 bin), 2 = Down (10 bin) ;;;;;;
# C  -  Correct Side Door Enable;  1 = LEFT (01 bin), 2 = RIGHT (10 bin), 3 = BOTH (11 bin);;;;;
# L  -  Platform length (0-1023 m in 1m steps) ;;;;;;
# I  -  Station(3 character string in uppercase);;;;;;
# P  -  Platform Number(0-31) ;;;;;;
# N  -  Name of beacon ;;;;;;
;;;;;;
1;1;1;2;25;EPS;5;GTY;;;;;;
1;1;1;2;50;EPS;6;MAU;;;;;;
1;1;2;2;204;EPS;4;EPS;;;;;;
1;1;1;2;75;EPS;7;HDG;;;;;;
1;1;1;1;204;KOE;4;KOE;;;;;
1;0;1;3;300;LON;1;LON;;;;;
1;1;2;2;204;OXF;4;OXF;;;;;
1;0;1;3;300;HKI;1;HKI;;;;;
1;0;1;3;300;HKI;1;HKI;;;;;
0;0;1;1;0;AAA;0;MIN;;;;;
65;0;2;2;512;MMM;15;MED;;;;;
127;1;2;2;1023;ZZZ;31;MAX;;;;;
```

Figure 13. Beacon csv file template

The file contains all the relevant information to construct a beacon reply message. Comments in the beacon file start with hash (#) character and comment lines are ignored. The user saves the beacon at a predefined file location in the Raspberry PI. When simulator is started the file is read. The lines which start hash, space, semicolon, return and newline character are ignored. During parsing process, each line is separated into a

series of tokens using delimiter ";". The token is stored in the beacon data structure according to the order of items in data format of set on the file.

3.3.2 Serial Communication

Serial communication function is started once data is extracted from beacon csv file. The USB-RS485 cable used by beacon reader simulator incoporates FT232RQ USB to serial UART interface IC device. Therefore UART settings are configured during the serial communication process. All Linux systems provide an access to serial port via device files. The serial port is accessed by opening the device file which is /dev/USB0tty since USB-RS485 cable is used.

The serial port is opened, and the serial interface parameters are changed to support communication with ASDO. The simulator uses the same serial settings as the beacon reader. The following serial control options are set:

- Baud rate 38400
- 8-bit character size
- 1 Start bit, (logic 0)
- 1 Stop bit, (logic 1)
- Even parity

The read timeout is set such that minimum of 17 character are read and waits 0.1 seconds for incoming character. After the serial interface parameters are configured, reading from and writing to the serial port is possible. The poll request message is read and passed on to be analysed.

3.3.3 Poll Request Analysis

The beacon reader reply message is used to communicate information from reader RS485 port to ASDO. Beacon reader simulator constructs the beacon reply message according to the actual beacon reader (see figure 7). First each beacon data item bits need to be packed according to the beacon reader protocol (see figure 8). Beacon information array is created for each beacon data which is six bytes (48 bits) long and fits in the information section. Beacon information section consists of 44 bits of data and the extra four bits are not used so it is filled by the Dummy header. Dummy header is set to one that is 0001 binary. The beacon data bits extracted and packed in each element of the array as illustrated by figure 14.

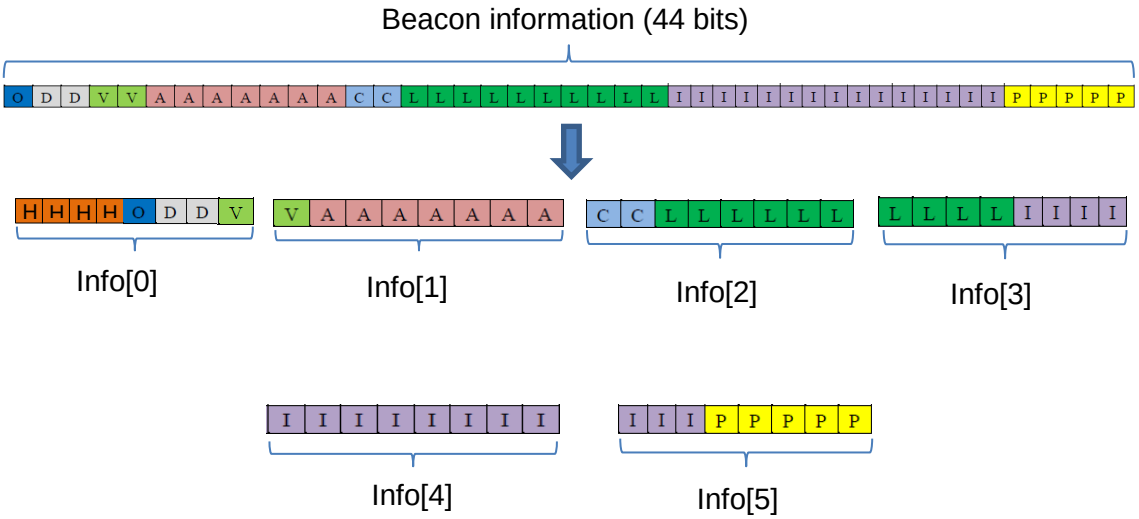


Figure 14. Beacon data bit packing

The beacon data is packed, and serial communication is started. Poll request message is read, and it is passed on to be processed. Beacon reply message construction is initiated, start and address sections are filled first. Control section of the Poll request message determines what information is needed from the beacon reader. The reply control byte and beacon information section are set based on the poll request control byte and user selected message type. By default, the reply control byte is set to 0x3F and beacon information section is set to zero. If user requests to send beacon data, control byte is set to 0x01 and beacon information section is filled with the packed beacon data. In the case where the user selects to send an error message, the control byte is set to 0xFC and beacon information section is set to error code requested by the user. When ASDO receives the data, it requests the beacon reader simulator to delete it. In that case reply control byte is set to 0x3F and beacon information is set to zero. The reply request

counter is set to the same value as the poll request message counter. Checksum calculations are done based on the CRC standards used by the actual beacon reader. The data section CRC and address section CRC of the reply message are set with the calculated checksum values. Checksum error can be applied to the data section CRC, address section CRC or both sections based on the user request. This is done by adding a random number to the CRC section/s according to the user request.

The user can send a reply message with incorrect size. In the case where user wants to send a short reply message, half of the message (nine bytes) is written to the serial port. When user wants to send a large reply message a double sized (34 bytes) message is written to the serial port. In normal cases 17 bytes are written to the serial port after beacon message is constructed.

Figure 15 illustrates the process of how the beacon reader reply message is constructed.

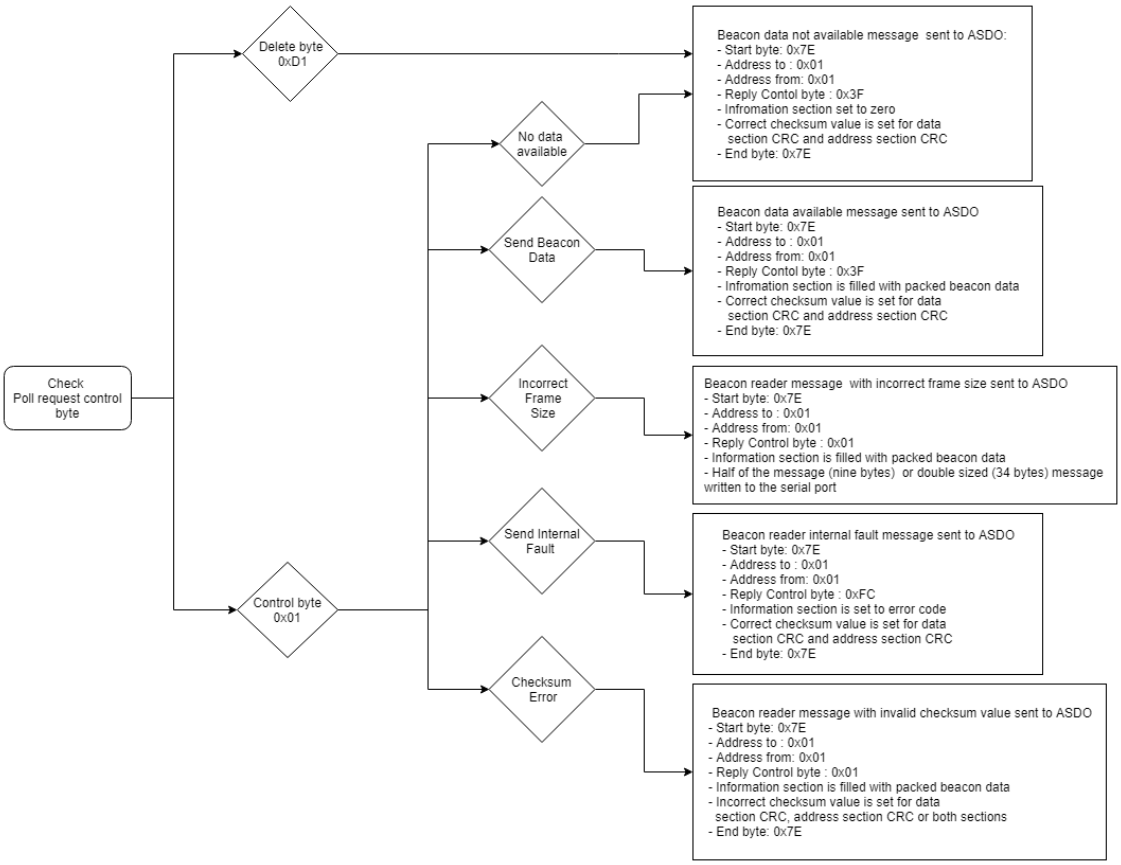


Figure 15. Simplified Reply message construction process diagram

3.3.4 Graphical User Interface

The beacon reader simulator needs to communicate with ASDO constantly. The simulator software is divided into two threads, GUI thread and serial communication thread. These threads interact using the simulator data structure that contains the beacon and user control data. Both threads have access to the members in the data structure and can update or edit them. Beacon data refers to data collected from the beacon. The user control data is used to control the serial communication thread. When the GUI is used the relevant control data members are updated. In addition to that they are used to notify the user the selected task is completed and its outcome. Figure 16 illustrates the simulator threads and the primary data structure members that are used by both threads.

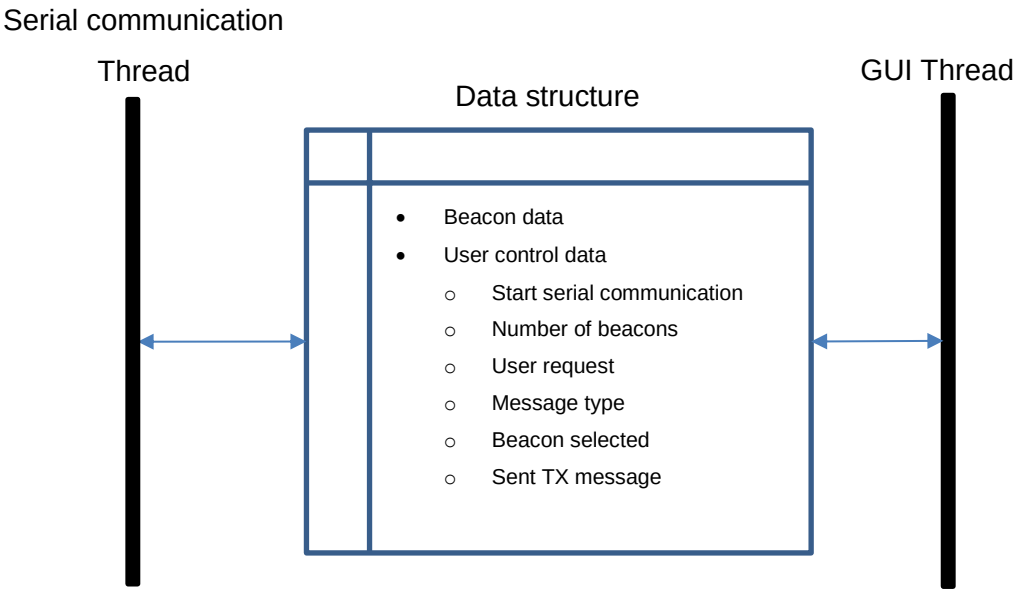


Figure 16. Beacon reader simulator thread

User interface was created for the beacon reader simulator software as shown in figure 17. The beacon file is read when the simulator is started, and the result of that process is displayed on the system log. When beacon file is read successfully, the number of beacons is shown on the system log screen. The serial interface is initialized and notifies the user in GUI. The serial communication thread starts running once beacon data is collected from the beacon file and serial interface parameters are configured.

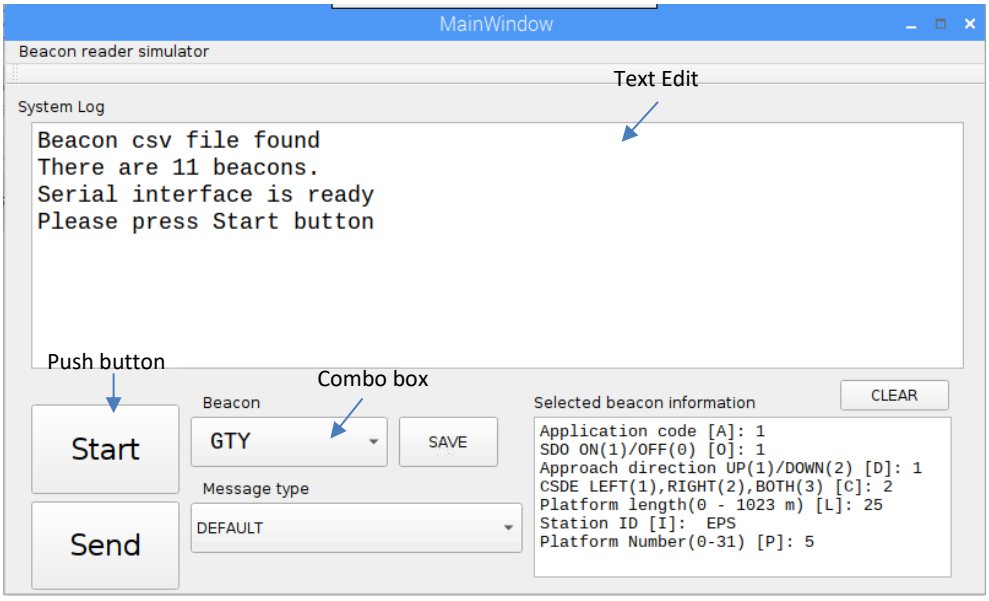


Figure 17. Beacon reader simulator GUI

Pressing the start button changes the start serial communication member value. Serial communication thread has access to it, so it notices the change and starts communication with ASDO. In the serial communication thread poll request message from ASDO is read, processed and reply message is sent to ASDO.

Beacon combo box is opened when it is pressed and lists all the beacons (see figure 18). A beacon can be selected from the combo box and its contents is displayed in the Selected beacon information screen (Figure 18). The index of current item in the beacon combo box is stored and updates Beacon selected member value in the data structure.

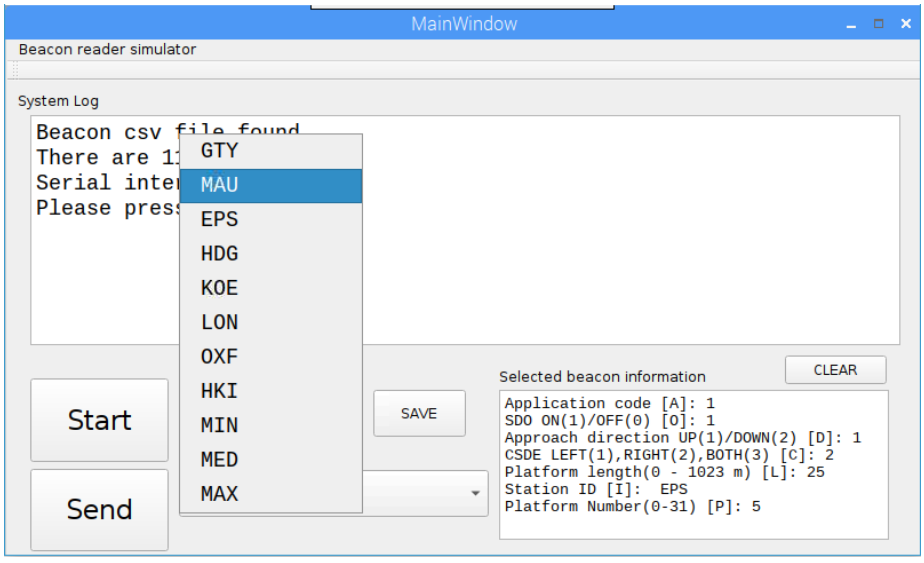


Figure 18. Selecting beacon from combo box

The user can select the reply message type from the Message type combo box and the options are shown in figure 19. Message type combo box is opened when it is pressed and lists all the types of reply message. The index of current item in the message type combo box is stored and updates Message type member value in the data structure.

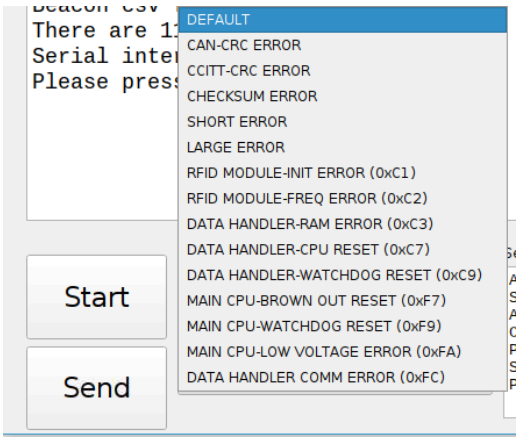


Figure 19. Message type options

The selected reply message is sent to ASDO when the send button is pressed, and User request data structure value is changed. This notifies the serial communication thread to start constructing the reply beacon message. The thread also checks what type beacon message is requested. Reply message is constructed based on Beacon requested and Message type value. Before the reply message is written into the serial port, it is copied to send TX message member in the data structure. Once the reply message has been sent to ASDO, it is printed in hexadecimal form in the system log screen as shown in Figure 20.

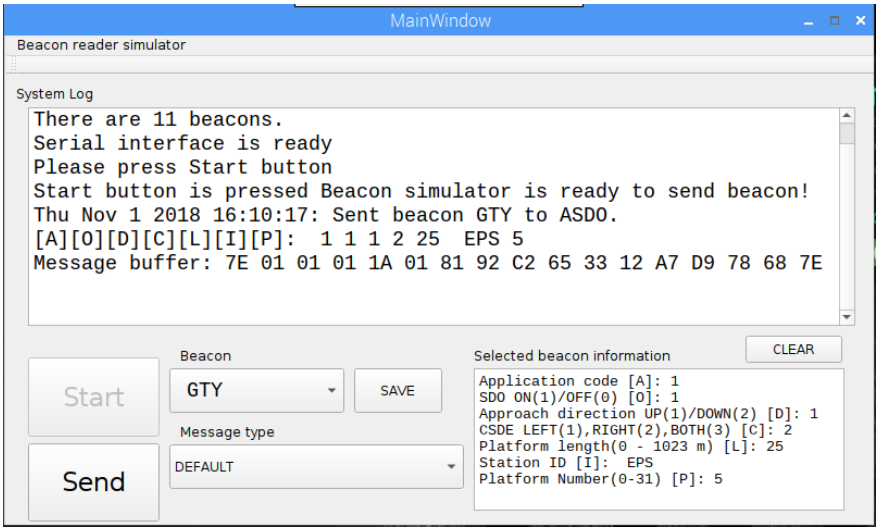


Figure 20. Beacon reply message sent to ASDO

The user can save the system logs by pressing the “save” button. The “clear” button clears text on the system log screen.

Figure 21 illustrates a normal sequence of events when using beacon reader simulator.

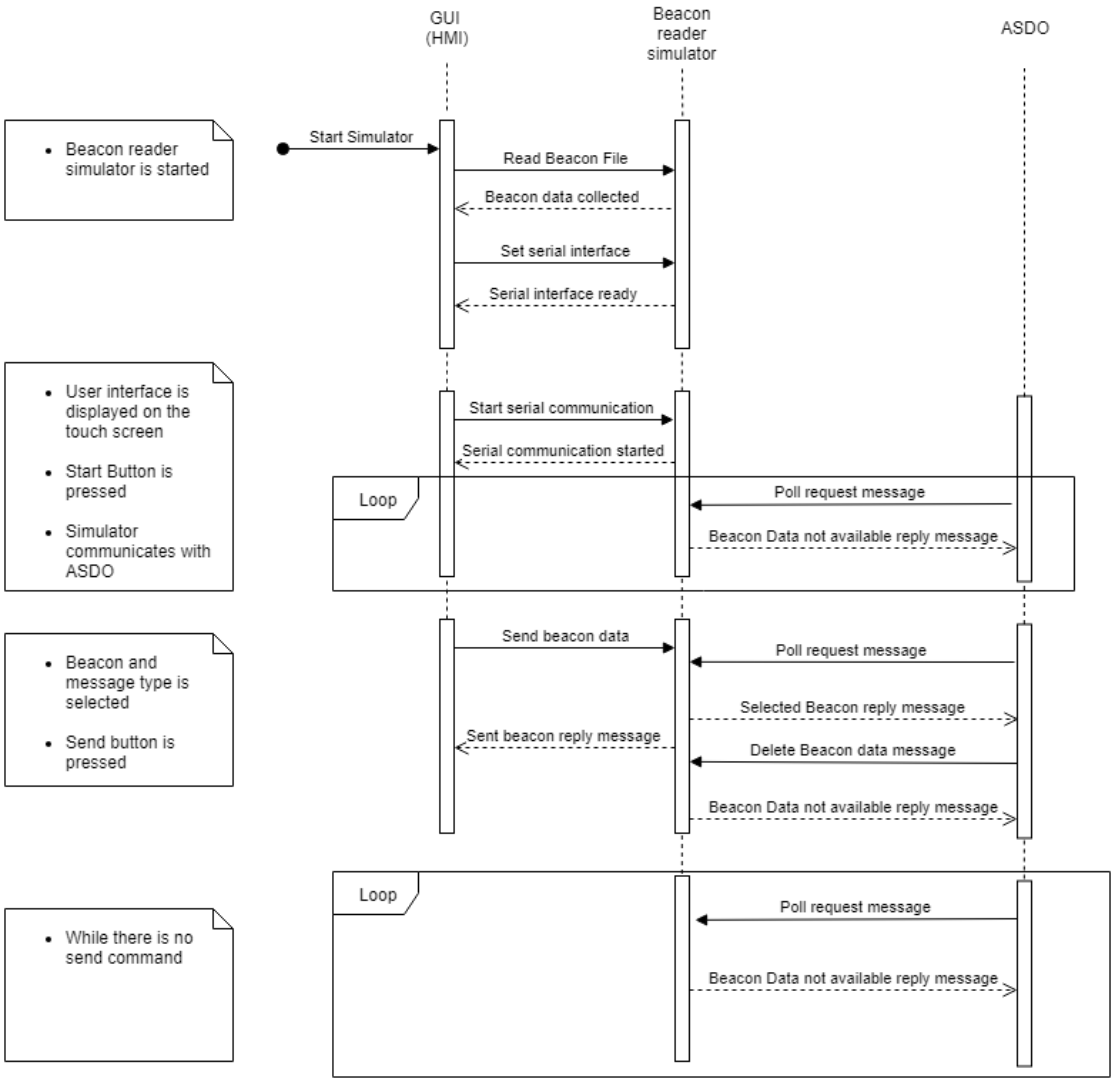


Figure 21. Beacon reader simulator sequence diagram

4 Beacon Reader Simulator Testing

4.1 Test System Configuration

The beacon reader simulator performance is tested on the ASDO test system. ASDO test system contains the displays, modules and ASDO software application used on one 3-car unit train. A digital input/output (DIO) simulator has the switches to supply digital signal to ASDO.

In addition to test rack the following software applications are used for testing:

- ASDO Test Tool
- ASDO Movement Simulator
- System log
- Serdump

ASDO Test Tool is used to monitor ASDO process data going through the car. The test tool shows the details of the train composition. Odometer, GPS and beacon condition can be observed on the test tool. Furthermore, test tool displays if the beacon data read is successful and the beacon data contents. If there are internal faults within the beacon reader the error codes are displayed.

The train movement is done using the ASDO Movement Simulator. The start location coordinates, end location coordinates and speed of a train route or segment is set. In addition to the GPS parameters, train wheel diameter and pulse per revolution is set. When movement simulator is started the GPS coordinates is provided to ASDO. Also, at the same time movement simulator sets the function generator to supply pulses to ASDO based on the train speed, wheel diameter and pulse per revolution setting.

System log contains a record of the events occurred within ASDO. The systems log shows informational, error and warning events related to the ASDO. If error is caused by the beacon reader simulator it can be investigated by checking the syslog.

Serdump is the ASDO serial monitoring software. When the software is executed the transmitted and received serial data is displayed. This is used to check the poll request

message transmitted from ASDO and the beacon reply message from beacon reader simulator.

These software applications are used in the test PC and the beacon reader simulator test set up is illustrated on figure 22.

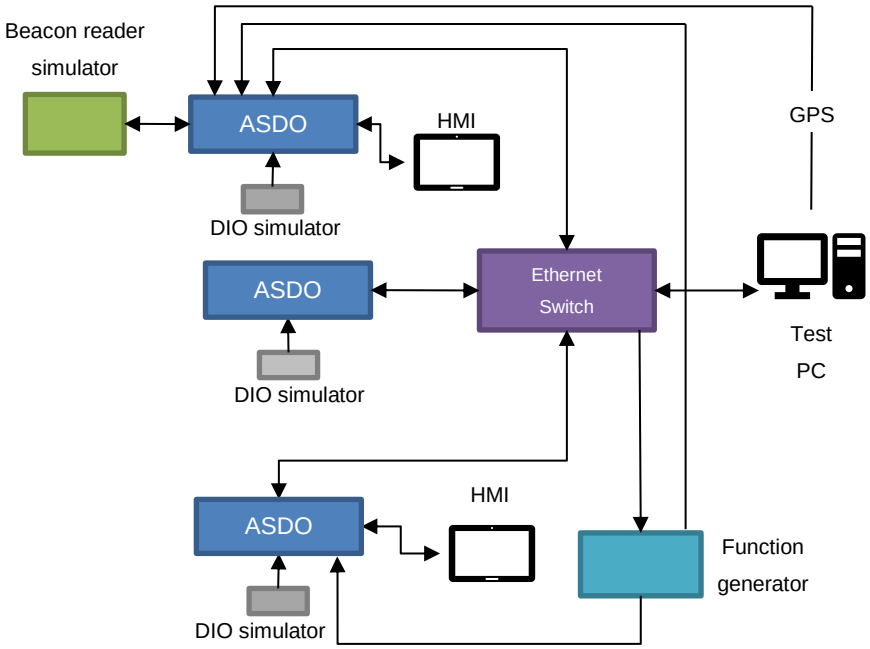


Figure 22. Beacon reader simulator test set up

4.2 Test Results

Four test case were carried out to see how the beacon reader simulator performs and how ASDO interacts. The following test cases were investigated:

- Simulator default state
- Sending a reply message
- Sending a reply message with incorrect size
- Sending a reply message with incorrect checksum value
- Sending a reply message with internal fault

4.2.1 Simulator Default State

Default state here refers to how the beacon reader behaves when the user does not send any data to ASDO. In default state the simulator reads the poll request message and responds that the data is not available. The transmitted and received data can be observed by using Serdump in ASDO. Serdump is started while the beacon reader simulator is running. Figure 23 shows the beacon reader simulator receives poll request message from ASDO and it responds accordingly. In addition to that it can be observed from ASDO Test Tool that beacon reader simulator is connected or equipped, and the received data is valid (see figure 24).

```
TX: 0.000000 [17]  7E 02 10 01 00 00 00 00 00 00 3D 80 60 1D 83 5C 7E
RX: 0.020004 [17]  7E 01 01 3F 00 00 00 00 00 00 3D 80 C7 A5 76 DB 7E
TX: 0.520030 [17]  7E 02 10 01 00 00 00 00 00 00 3D 81 EB 2F 65 1F 7E
RX: 0.546562 [17]  7E 01 01 3F 00 00 00 00 00 00 3D 81 4C 97 90 98 7E
TX: 1.043325 [17]  7E 02 10 01 00 00 00 00 00 00 3D 82 FD 4B B9 B8 7E
RX: 1.058040 [17]  7E 01 01 3F 00 00 00 00 00 00 3D 82 5A F3 4C 3F 7E
TX: 1.549992 [17]  7E 02 10 01 00 00 00 00 00 00 3D 83 76 79 5F FB 7E
RX: 1.569678 [17]  7E 01 01 3F 00 00 00 00 00 00 3D 83 D1 C1 AA 7C 7E
TX: 2.060026 [17]  7E 02 10 01 00 00 00 00 00 00 3D 84 D1 83 10 D7 7E
RX: 2.081184 [17]  7E 01 01 3F 00 00 00 00 00 00 3D 84 76 3B E5 50 7E
```

Figure 23. Serial data transmitted and received by ASDO

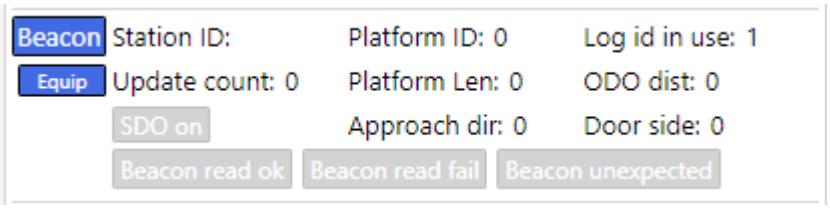


Figure 24. ASDO Test Tool showing Beacon reader simulator is equipped

4.2.2 Sending a Reply Message

The train needs to move in order to accept beacon data which is done using ASDO movement simulator. ASDO movement simulator is configured to move from station G to station H. Station H contains a platform where beacon GTY is used so when the train arrives the beacon GTY is sent to ASDO. The beacon data is accepted, and door pattern is shown in HMI according to the beacon (see figure 25).

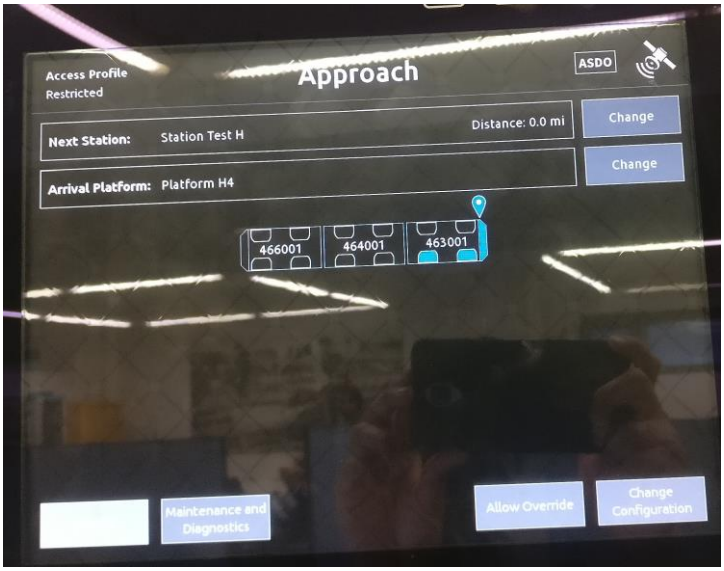


Figure 25. Beacon platform displayed at Station H

The beacon data is read successfully, and it can be verified in the ASDO test tool (figure 26). The configured beacon data is accepted, and its contents is shown in the test tool.

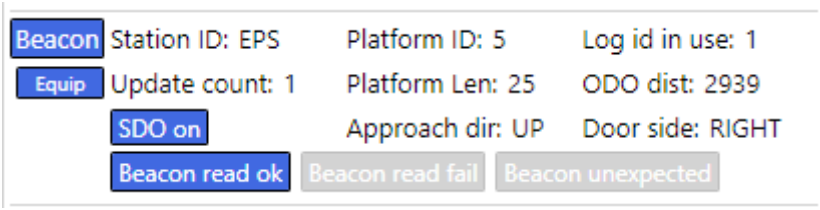


Figure 26. ASDO test tool showing beacon data is read

The beacon reply message is in the correct format. In the beacon information section, the bit is packed properly and matches the beacon data configured by the user. After the data is received by ASDO, it requests beacon reader simulator to delete the data. Beacon reader simulator responds data is not available as shown in figure 27.

```
TX: 2.576592 [17]  7E 02 10 01 00 00 00 00 00 00 44 2B 9A E7 CD 6C 7E
RX: 2.600054 [17]  7E 01 01 01 1A 01 81 92 C2 65 44 2B 0B 4F 8B 68 7E
TX: 2.603604 [17]  7E 02 10 D1 00 00 00 00 00 00 44 2C B4 C5 F1 77 7E
RX: 2.629365 [17]  7E 01 01 3F 00 00 00 00 00 00 44 2C 9A A5 77 C7 7E
```

Figure 27. Beacon reply message received by ASDO

```
Fri Nov 2 2018 14:25:12: Sent beacon GTY to ASDO.
[A][O][D][C][L][I][P]:  1 1 1 2 25  EPS 5
Message buffer: 7E 01 01 01 1A 01 81 92 C2 65 44 2B 0B 4F 8B 68 7E
```

Figure 28. Sent message is displayed on the GUI

4.2.3 Sending a Reply Message with Incorrect Size

Beacon reader simulator can send a message with incorrect frame size to ASDO. ASDO receives the data and requests the beacon reader simulator to delete it. The beacon reply message is not accepted as shown in figure 29. If the beacon data was accepted the updated counter is increased and its contents would be displayed on the test tool (see figure 30).

```
TX: 8.256432 [17]  7E 02 10 01 00 00 00 00 00 00 14 3E AF A1 C2 A6 7E
RX: 8.275431 [ 9]  7E 01 01 01 1A 01 81 92 C2
TX: 8.275476 [17]  7E 02 10 D1 00 00 00 00 00 00 14 3F AD 4B 57 D2 7E
RX: 8.294145 [17]  7E 01 01 3F 00 00 00 00 00 00 14 3F 83 2B D1 62 7E
```

Figure 29. Small Beacon reader reply message received by ASDO

Figure 30. Snippet of ASDO test tool showing beacon data is not accepted

4.2.4 Sending a Reply Message with Incorrect Checksum value

It is possible to send a Beacon reply message with checksum calculation errors. When checksum values are not valid the ASDO does not accept the beacon data. As shown in figure 31 ASDO received a message with wrong CAN-CRC value and the following message had a CCIT-CRC error. This shows that ASDO validates the Address CRC and Data section CRC value.

```
Nov  2 13:14:39 (none) local0.info ASDO-Core[421]: Beacon CAN-CRC failed: e1da <> 4c4c
Nov  2 13:14:52 (none) local0.info ASDO-Core[421]: Beacon CCITT-CRC failed: f577 <> 3e3e
```

Figure 31. Syslog displaying check sum error

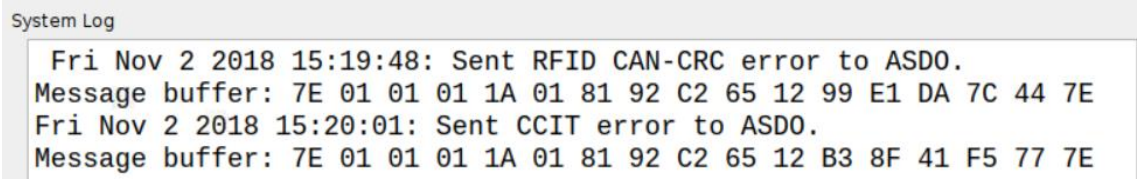


Figure 32. Message with checksum faults is displayed on the GUI

4.2.5 Sending a Reply Message with Internal Fault

In addition to erroneous messages the beacon reader simulator can send internal fault messages to ASDO. A RFID module Initialisation error message is sent to ASDO. The reply message constructed correctly and received by ASDO (see figure 33).

In the syslog ASDO detected the beacon reader fault and request the reader to delete the data (see figure 35). The error code is also displayed in ASDO test tool and it can be confirmed that ASDO detected the beacon reader internal fault (see figure 34).

```
TX: 7.730064 [17]  7E 02 10 01 00 00 00 00 00 00 AE 37 61 8F 35 BA 7E
RX: 7.750998 [17]  7E 01 01 FC C1 00 00 00 00 00 AE 37 B0 4F D1 B3 7E
TX: 7.751419 [17]  7E 02 10 D1 00 00 00 00 00 00 AE 38 A7 A3 D8 D5 7E
RX: 7.766255 [17]  7E 01 01 3F 00 00 00 00 00 00 AE 38 89 C3 5E 65 7E
```

Figure 33. Beacon reader reply message with internal fault received

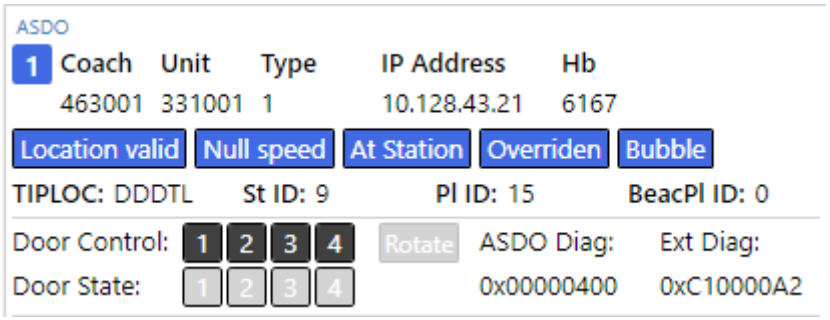


Figure 34 ASDO test tool showing the internal fault error code on Ext Diag

```
Nov  2 13:24:59 (none) local0.info ASDO-Core[421]: Beacon Internal Error - cause=0xC1!
Nov  2 13:24:59 (none) local0.info ASDO-Core[421]: Beacon reader internal fault - cause=0xC1!
Nov  2 13:25:05 (none) local0.info ASDO-Core[421]: Beacon reader internal fault deleted!
```

Figure 35. Syslog showing ASDO detected reader internal fault

5 Beacon Reader Simulator Evaluation

Beacon reader simulator is small, compact and light compared to the actual beacon reader. The simulator user interface is easy to use and stable according to the testers. GUI clearly indicates that the beacon data is sent to ASDO and prints the reply message. The components to the GUI are also responsive and it interact properly with the touch screen.

The simulator follows the actual beacon reader protocol and its core functionality is replicated. It is capable of communicate with ASDO the same manner as the actual beacon reader. Furthermore, the beacon reader simulator contains extra features which the beacon reader did not have. The user can create multiple beacon according to his/her preference and send the data to ASDO. In addition to that different types of messages can be sent from the beacon reader simulator.

6 Conclusion

The beacon reader simulator performs as planned. However, it is possible to add more features to the beacon reader simulator. The beacon reader simulator can be automated. This is done so that the user can configure the stations where the beacons are used, so when the train arrives at the station the beacon reader simulator automatically sends the data to ASDO. This way the user interaction is not necessary to send beacon data.

At the moment beacons cannot be added or modified through the beacon reader simulator. In order to modify or add beacons the user needs to shut down the simulator and edit the beacon file. It is possible to implement a feature where the user can edit existing beacons or add new beacons through the UI.

The beacon simulator can communicate one ASDO unit at the moment. The beacon reader simulator can be modified to communicate with two ASDO units. Therefore, when the train arrives at the station the beacon data will sent to the front car and then to rear car.

To conclude, the current limitations aside beacon reader simulator is good replacement for the actual beacon reader. The beacon reader simulator can test all the features of

the beacon reader. In addition to that the simulator can be improved further to carry out more functions.

References

[1] Chan K, Turner D. The application of selective door opening within a railway system [Internet]. Witpress.com. 2018 [cited 18 November 2018]. Available from: https://www.witpress.com/Secure/elibrary/papers/CR04/CR04016FU.pdf

[2] Automatic Selective Door Opening (ASDO) Correct Side Door Opening (CSDE) [Internet]. Sellacontrols.com. 2018 [cited 18 November 2018]. Available from: http://www.sellacontrols.com/data/pp/pp054/docs/ASDO_CSDE_0318.pdf

[3] Tracklink III® Beacon The RFID Solutions for Railways. [Internet]. Sellacontrols.com. 2018 [cited 18 November 2018]. Available from: http://www.sellacontrols.com/data/pp/pp054/docs/TracklinkIIIBeacon0318.pdf

[4] Tracklink III® Reader Track to Train Communication. [Internet]. Sellacontrols.com. 2018 [cited 18 November 2018]. Available from: http://www.sellacontrols.com/data/pp/pp054/docs/TracklinkIIIBeacon0318.pdf

[5] Bras D. EKE-Electronics - Automatic Selective Door Operating system (ASDO) [Internet]. Eke-electronics.com. 2018 [cited 18 November 2018]. Available from: https://www.eke-electronics.com/automatic-selective-door-operation-system-asdo

[6] TRACKLINK III HSd2300/025-Reader SOFTWARE REQUIREMENTS SPECI-FICATION. HIMA-SELLA LTD; 2013.

[7] Serial Communication - learn.sparkfun.com [Internet]. Learn.sparkfun.com. [cited 18 November 2018]. Available from: https://learn.sparkfun.com/tutorials/serial-communication/all

[8] What is Serial Communication and How it works? [Explained] [Internet]. Codrey Electronics. 2018 [cited 18 November 2018]. Available from: https://www.codrey.com/embedded-systems/serial-communication-basics/

[9] Lemmon M. What is a serial interface? [Internet]. Www3.nd.edu. 2009 [cited 18 November 2018]. Available from: https://www3.nd.edu/~lemmon/courses/ee224/web-manual/web-manual/lab9/node4.html

[10] Glossary Definition for RS485 [Internet]. Maximintegrated.com. [cited 18 November 2018]. Available from: https://www.maximintegrated.com/en/glossary/definitions.mvp/term/RS485/gpk/996

[11] Raspberry Pi 3 Model B Plus (B+) Technical Specifications [Internet]. www.element14.com. 2018 [cited 18 November 2018]. Available from: https://www.element14.com/community/docs/DOC-88853/l/raspberry-pi-3-model-b-plus-b-technical-specifications

[12] Supported Platforms | Qt 5.11 [Internet]. Doc.qt.io. [cited 18 November 2018]. Available from: http://doc.qt.io/qt-5/supported-platforms.html?hsCtaTracking=5da5e2cd-9eb3-4dc2-a917-d7419c6adca8%7C993e64d0-b3e8-40cd-b32c-b5eb8bf05f0a

[13] Raspberry Pi 3 Model B [Internet]. Wiki.seeedstudio.com. [cited 18 November 2018]. Available from: http://wiki.seeedstudio.com/Raspberry_Pi_3_Model_B/

[14] 15. USB TO RS485 SERIAL CONVERTER CABLE Datasheet [Internet]. Ftdichip.com. 2010 [cited 18 November 2018]. Available from: https://www.ftdichip.com/Support/Documents/DataSheets/Cables/DS_USB_RS485_CABLES.pdf

[15] The Qt Company [Internet]. Qt.io. 2018 [cited 18 November 2018]. Available from: https://www.qt.io/company

[16] Qt | Cross-platform software development for embedded & desktop [Internet]. Qt.io. [cited 18 November 2018]. Available from: https://www.qt.io/

[17] What is the System Log (Syslog)? - Definition from Techopedia [Internet]. Techopedia.com. [cited 18 November 2018]. Available from: https://www.techopedia.com/definition/1858/system-log-syslog

[18] BigCommerce Help Center [Internet]. Support.bigcommerce.com. [cited 18 November 2018]. Available from: https://support.bigcommerce.com/s/article/Import-Export-Overview

Beacon reader simulator