

Henoke Benebaru Hailegiorgis

Implementing a Data Logger for Home Automation

Helsinki Metropolia University of Applied Sciences

Bachelor of Engineering

Information Technology

Thesis

5 May 2015

Author(s) Title	Henoke Hailegiorgis Implementing a data logger for home automation
Number of Pages Date	44 + 2 appendices 5 May 2010
Degree	Bachelor of Engineering
Degree Program	Information Technology
Specialization option	Software Engineering
Instructor(s)	Keijo Lämsikunnas, Senior Lecturer
<p>Today's technology provides a smart and secure ability to a home automation system. However to overcoming drawbacks and improving the system in a cost effective way needs to be done in a broader sense by paying attention to the most important purpose of a house automation system.</p> <p>The purpose of this thesis is to discuss a data logger, an automatic data collecting device, extension to a home automation system to improve the performance of the system. The focus is on the data logger design and implementation. The main objective is to propose a data logger addition to be practiced by home automation vendors.</p> <p>To justify the data logger addition, a prototype data logger design for an automated ventilation system is carried out as a case study. The design consists of a microprocessor which runs the designed software, a serial communication cabling and home automation system simulator. A serial communication protocol software plugin to open-source software is designed to provide a working data logger system.</p> <p>The finding of this paper shows with a low cost device that a data logger can be designed and used to improve the home automation system. It is recommended for home automation system vendors to include industry-wide open standard data logger design in the system.</p>	
Keywords	data logger, home automation, RS-485 serial communication

Contents

1	Introduction	1
2	Project Overview	2
2.1	Home Automation	2
2.1.1	Home Automation Objectives and Challenges	2
2.1.2	Home Automation Building Blocks	4
2.2	Data Logger	5
2.2.1	Data Logger Benefits	5
2.2.2	Data Logger Types and Application Area	6
2.3	Embedding Data Logger on Home Automation	7
3	Data Logger Hardware Design	8
3.1	Data Logger Overview	8
3.2	Design Components	9
3.2.1	Raspberry Pi	9
3.2.2	Arduino Vallox Digit2 SE Simulator	11
3.2.3	Serial-USB Adaptor	14
4	Communication Protocol	17
4.1	Overview of Communication Protocol	17
4.2	Serial Communication	19
4.2.1	RS-232 Serial communication	19
4.2.2	RS-485 Serial Communication	20
4.2.3	Serial Communication Protocol	22
5	Data Logger Software	25
5.1	Overview of Data Logger Software	25
5.2	TaloLogger	25
5.3	TaloLogger Data Source Plugin Software Design	28
6	Project Implementation and Result	30
6.1	Overview of the Design	30
6.2	Vallox Digit Serial Protocol Plugin	31
6.2.1	Packet Reading and Arranging Software Component	31
6.2.2	Packet Verification Software Component	34

6.2.3	Data Extraction Software Component	35
6.3	TaloLogger Core Configuration	35
6.3.1	Data Source Configuration	35
6.3.2	Data Store Configuration	36
6.3.3	VALLOXDIGITSerial Configuration	36
6.3.4	Measurement Points and Data Store Key Configuration	36
6.4	Testing	37
6.4.1	Testing The Controller using a script	37
6.4.2	Testing Using TaloLogger Core Configuration	38
7	Conclusion	39
	References	40
	Appendices	
	Appendix 1. Temperature Conversion Table	
	Appendix 2. Variable Table	

1 Introduction

Nowadays it seems that the growth in technology focuses on making the living conditions easy for human beings. A good example of the ease of life brought by technology is a home automated system. A home automation system avoids tedious and time consuming practice, bring security, reduce cost of living. The system uses sensor technology, automation, computer or microprocessor technology and telecommunication technology. In the current technology it became possible to make a living home even smart and entertaining.

Basically home automation systems are considered a controlling system for parameters such as temperature, humidity and ventilation. The system reads data from the environment via sensors then a microprocessor device analyses it and forwards an appropriate parameter to a control device. Although the automated system fulfils its purpose by hiding the details in the change of parameters, sometimes it is important to know what is going on around the environment. The performance of the home automation system can be enhanced by using an additional data logger system. A data logger is an automatic data collecting tool to further analyze the data and to understand the changes around the environment.

Data loggers are electronic devices capable of recording data from a data source, a home automation system, into a data store or memory as a file or a database object. A typical data logger requires a communication interface from the data source, a microprocessor or computer to analyze, process and record data to a data source, either a database or log file. [1]

The goal of this thesis is to design and implement a prototype data logging functionality as an extension to a home ventilation automation system. The data logger consists of a hardware component a Raspberry Pi, a serial to a USB adapter and a serial cable to connect to a Vallox Digit2 SE ventilation system. This data logger implemented is by designing a proprietary standard RS-485 serial communication protocol software as a plugin to an open source data logger application which runs on Raspberry Pi.

2 Project Overview

2.1 Home Automation

2.1.1 Home Automation Objectives and Challenges

Home automation is a technology of integrating home appliances into a network and controlling them automatically for living convenience. This technology engages switching on/off or setting a value at a desired condition to home appliances like an air conditioner, lighting or heaters. In addition to algorithmic automation a user can adjust to a suitable setting. The home automation system brings a better quality of living condition for residents. The purpose of home automation to fulfil the following objectives. [2]

- **Comfort:** Residents in a conventional home switch light on/off light or adjust it to a favorable temperature manually. This tedious task, checking every time when a change appears at home is inconvenient for residents. Sometimes people choose to continue sleeping in uncomfortable hot room instead of waking up to switch on the air conditioner because both waking up and continuing sleeping are uncomfortable. A well designed automation system with a user interface increase the comfort of the user by reducing the tedious tasks.
- **Security:** When a storm is expected or protection from burglary is needed, residents check every door and window in a conventional home by walking through all the rooms. Burglary might happen anyway by breaking in or a resident might miss something during checking. The level of security is enhanced by cooperating and automating devices such as alarm equipment, a sensor-controlled roller shutter or programmable timer for power outlet. Additionally early notification of danger detected by a sensor helps to prepare for appropriate reaction.
- **Safety:** Home automation improves the safety of a resident particularly in the aspect of health. The sense of safety is to prevent an accident from happening avoiding the root of accident that is unfavorable situation. Since the purpose of home automation is to reduce unfavorable situation, therefore it increase safely.
- **Minimize cost:** one of the main expenses in household is energy. In automated system the excess loss of energy is prevented.

Although most home automation systems try to provide these objectives, there are a few challenges to be addressed. Based on the research conducted by Microsoft Research and the University of Washington four main challenges of home automation are illustrated as follows: [3]

- **High cost of ownership:** The cost of hardware alone and the cost of installation and maintenance both in money and time aspect are high. In addition when an automated system is installed the previous household hardware should be replaced in most cases which is an extra cost.
- **Inflexibility:** Vendors who provide home automation systems may integrate with each other or be inflexible to integrate. Although the integration eases in some occasions, sometimes it becomes complicated enough, so that it is easier to keep in separate too.
- **Poor management:** In order to save energy, unnecessary cost and time, good management is required. The research shows that in order to achieve a better management iteration of the setup of the system continuous assistance from a consultant is required. It is difficult to fulfill these requirements since unreliable and unpredictable behavior of hardware is frustrating or even consultants sometimes fail to solve their own product failures.
- **Security:** Most home automation systems have a control panel on the wall that any person at home can access. It may be mandatory to be accessed only by a partial group of users or a temporary access only by guests are required. So this raises a barrier in fulfilling security.

The research finally proposed a solution to overcome these challenges. The proposal suggests a new design layout that implements a low-cost and open-source home automation system. The most important requirement in the proposed system are user friendly interface, security and authentication, low cost per node/ high node count, large area coverage and system scalability.

The main modules of this proposed system are the server, hardware interface module and software package. The software package runs on the server which is a computer and the hardware interface module wired with sensors and actuators. A secure Wi-Fi technology used in communication between the server and hardware interface module.

Although a new home automation system layout design like previously mentioned design improves home automation, the design in this thesis approaches the challenge differently. There are a lot of design layout different vendor's use. The objective of this thesis is not to improve the performance of already designed home automation system by adding a data logger.

2.1.2 Home Automation Building Blocks

From a technical viewpoint typical home automation systems have at least three components. These components are either compacted in one device or are a separate device connected through wired or wireless communication. Some core components are as follows: [4]

- **Sensors and actuators:** Sensors and actuators are mechanical or electrical component which interact with the physical environment. Sensors collect information from the physical environment and feed to the system. Actuator on the other hand influence the environment based on the information have got from the system. For example infrared sensor detects the presence of a person while an actuator acts by switching on the lights.
- **Main control unit:** This main control device acts on environment by means of actuators based on the information read from the sensors. The information extracted from the sensors is processed by this control device according to the setting parameter provided by the user. The setting parameters are provided through a control panel.
- **Control panel:** Control panel is the user interface to manually set parameters or configure the automation to a personal suitable setting.

2.2 Data Logger

2.2.1 Data Logger Benefits

Data logging engages collection of environmental data such as temperature, relative humidity, or CO₂ percentage from the surroundings. A data logger is an electronic device which records measurement data at configured interval over a time period. A typical data logger consists of one or more sensors, an internal microprocessor and memory compacted in one device to record data unattended for about months. Although a typical data logger is a standalone device, the component can be separated for the need of a wide implementation.

Data loggers are used by researchers, environmental analysts or building managers to collect time stamped data for the purpose. The data collected by a data logger is important or mandatory for the research or performance improvement. Data logger are the best option to collect a data. [5]

- Low cost: the cost require to buy a data logger is very low as compared with the time and cost required to collect data manually by being at the place. Researchers can deploy a data logger to collect data for a period of time, transfer the data to computer and analyze the data according to their research purpose.
- Ease of use: A well designed data logger can be easily used by elementary students in addition to Engineers or research technicians. The main task of using data logger is deploying the data logger at appropriate place and transferring the collected data to a computer for further analysis of the data. These simple tasks make easy to use a data logger.
- Reliability: Once configured and deployed properly the data collected by the data logger are accurate and free of human error. The ability to stand wide range of environmental changes such as steady air flow to strong wind or refrigeration to constant sunlight makes it a reliable way to collect data.
- Time stamped data: for a researcher or data analyst to make a decision one time data is not enough for a better research a time stamped data are required to evaluate the data through different time. Since a data logger collect data in a time stamped manner for a period of time it shows the change in a data in the long run.

2.2.2 Data Logger Types and Application Area

Types of data logger

As mentioned in the previous section data loggers have three most basic parts; one or multiple sensors, internal microprocessor and data storage. Using a data logger includes configuring a piece of software, deploying in a secure place, downloading collected data and processing or analyzing the downloaded data. All data loggers have those three parts and characteristics. On the extent of how the data is downloaded or transferred by a user, a data logger is divided into three types: stand-alone, web-based data logging and wireless data node. [5]

Standalone data logger

In a standalone data all basic parts are compacted in one device or sometimes an external sensor plugged into the port. The data collected for a period of time will be downloaded to a computer through a USB interface. In some cases instead of taking the data logger to a computer and downloading data a pocket size device called data shuffle is used to retrieve data from the data logger and transfer it into the PC without interrupting data logging process.

Web-based data logging system

In a web based data logging system the collected data is transferred to a secure web server using an internet-based data access. Real-time data is accessed remotely through a secure website. The benefit of this system is that data collection can be performed in a long distance range via Wi-Fi, cellular GSM or Ethernet communication.

Wireless data node

A wireless sensors from multiple nodes transmits accurate and real-time data to a centralized location within a short distance range. This type of data transfer eliminates manual off-loading of data like a standalone data logger, but unlike web-based data logging system it transfers the data within a limited short distance range.

The required data logger is selected based on the following parameters: the budget, the scope of data logging need, the distance between data source and centralized location and the frequency to access data. Data loggers can be implemented to different kind of other systems with custom configuration.

Data logger application area

The two most used application area are system performance improvement and environmental research. The performance of a system for example buildings are improved by tracking building electricity usage, monitoring equipment runtimes to ensure efficient operation or better manage peak energy demand. Environmental research is a wide area to be performed by analyzing data collected from different environmental area for a wide period of time. [5]

2.3 Embedding Data Logger on Home Automation

Data loggers usually refer to a standalone data collecting device from different kind of sensor modules through a communication interface to a memory. In this design the sensor module is replaced by a data source which is a home automation control system and the device is a small low-cost Linux-based computer called Raspberry Pi.

Typical home automation has a set of sensors, an actuator, a control panel and a mainboard with a microcontroller. Both in wired and wireless connection there is communication protocol between this components of the system. A serial communication protocol between a computer panel and mainboard can be an example of wired communication protocol while an RF-module, Wi-Fi or Bluetooth are examples of wireless communication between sensors to mainboard.

So there is a possibility to add a data logger component to an already designed home automation system which can integrate with a standardized data communication protocol, for example serial communication. As vendors allow a standardized communication protocol for modularity and integration extension purposes, a data logger installed device or computer are possible to be extended to make the home automation system smarter and improve performance.

Imagine a ventilation system like Vallox Digit2 SE whose worth is 2300€ before tax, for the purpose of clean indoor air and quality life. Since the money spent on this product is quite high it is important to get what is paid for. As any other home automation system there are still the previously mentioned drawbacks and a need for improvement. The addition of a data logger with for less than 100€ not only solves the drawback but also improves the lifespan of the ventilation system.

The data extracted by the data logger can be analyzed for a desired objective. Whether it is energy efficiency or the health of the ventilation system it can be enhanced by analyzing the data obtained by the software installed on the data logger. Besides the data logger helps to understand what is going on in a home environment and inside ventilation control equipment.

3 Data Logger Hardware Design

3.1 Data Logger Overview

Usually data loggers usually refer to a standalone data collecting device from different kind of sensor modules through a communication interface to a memory for further analysis. In this design the sensor module is replaced by a data source which is a home automation ventilation system. As the prototype Vallox digit2 SE from Vallox Oy. is used. As mentioned in chapter two the purpose of this thesis is to add a data logger design to a standalone ventilation system to improve its performance.

The data collecting and processing is designed on a small low-cost Linux-based computer called Raspberry Pi. Since a computer is flexible to develop a software with a broader functionality, Raspberry Pi was chosen. Raspberry Pi provides a data logging platform and an ability to communicate with data source and it processes, analyzes and stores data on a configured data store.

In order for two different devices to work together there should be communication both in physical and logical ways. The communication protocol used in this Rs-485 serial communication protocol which is provided by Vallox Oy Company. While recent computer systems does not have a serial interface peripherals, a serial-USB adaptor should be used to replace the serial interface.

The data source, Vallox Digit2 SE, is too big in size and expensive so that working this thesis on the real equipment is unreliable. Therefore, a simulation software is installed on Arduino. In the following section the hardware components required to design the desired data logger will be explained.

3.2 Design Components

3.2.1 Raspberry Pi

For most people experimenting on a laptop or desktop computer is not easily affordable. Eben Upton created an affordable and small computer with the desire for people to have fun in computing and understand how a computer works. The Raspberry Pi foundation founded by Eben Upton with the assistance of Professor Alan Mycroft, Dr. Rob Mullins and Jack Lang from Cambridge University; hardware expert Pete Lomas and David Braben. The foundation aimed to create a computer as charity so that it would be affordable by anyone to experiment computing without any worry. After five years of clever engineering the first Raspberry Pi was built to be sold for 25\$ in 2012. [6]

Currently two models are available known as Model A and Model B. Both models are similar in most cases except Model A has less functionality than Model B in order to reduce cost and power consumption.

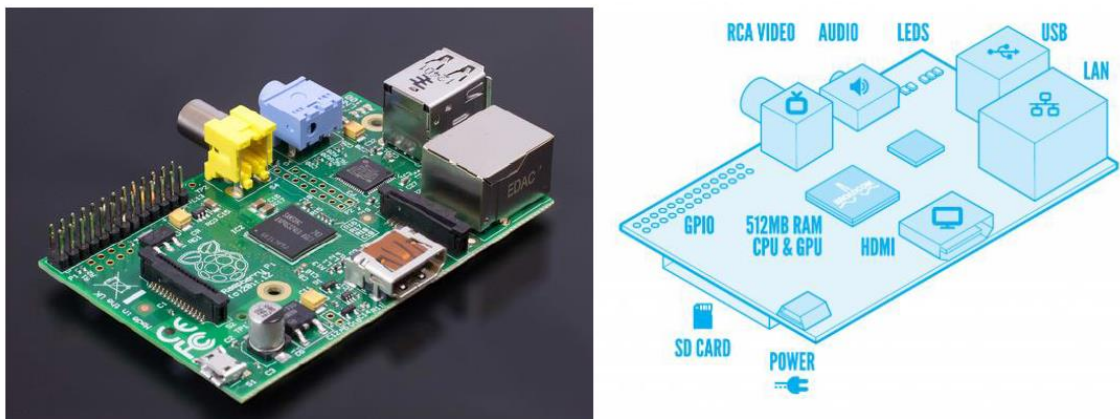


Figure 1. Raspberry Pi Model B. Reprinted from White Paper (2013) [7].

The Raspberry Pi Model B has a Broadcom BCM2835 system-on-chip (SoC) module located at the center of the board. This semiconductor provides general purpose computing and graphics. The RAM temporary memory storage for program and data is

stacked on top of the SoC. A high Definition Multimedia Interface (HDMI) port and composite video port located on the opposite sides of the board are connected to a modern TV and older TV respectively. While HDMI provides high definition video and audio output, the composite video port is a lower quality video output for TV without HDMI. On the right of the composite video port a 3.5 mm audio jack provides audio output for a composite video socket. [5]

On top of the Raspberry Pi underside of the board there is the SD card slot. The SD card is a memory to store operating system, files and non-volatile data. The GPIO pins are another part of the board which is usually used to add an embedded system to Raspberry Pi. Raspberry Pi has significant tools to make the design easy such as the ability to remotely access the desktop, efficient processing power and other design benefits. Table 1 below summarizes the technical specifications of Raspberry Pi which has enough features to be the main component for data logger hardware design. [6]

Table 1. Raspberry Pi model B feature summary. Reprinted from Raspberry Pi Project (2013) [6]

Type	Single-board computer
Operating System	Linux (Debian GNU/Linux)
Power ratings	3.5W (700mA)
Power Source	5V via MicroUSB or GPIO header
SoC	Broadcom BCM2835 (with CPU, GPU,DSP, SDRAM)
CPU	700 MHz ARM1176JZF-S core (ARM11 family, ARMv6 instruction set)
GPU	Broadcom VideoCore IV
Memory (SDRAM)	512MB (shared with GPU)
Onboard Storage	SD Card slot
Low-level peripherals	8 GPIO, UART, I2C bus, SPI bus with 2 chip selects, I2S audio, +3.3V, +5.5V,GND
USB 2.0 ports	2
Video Input	CSI input connector for RPF camera module
Video Outputs	Composite RCA, HDMI
Audio Outputs	3.5mm jack, HDMI, I2S audio

3.2.2 Arduino Vallox Digit2 SE Simulator

Vallox Digit2 SE: the data source

In this literature a data source refers to the source of information that the data logger should record. It can be any device capable of communicating periodically with a data logger through a supported interface. The data source for this data logger system is a standalone home automation ventilation control system. A home ventilation system is automatically controlled by Vallox Digit2 SE. The controlled parameters such as temperature, CO₂ and fan speed will be extracted automatically from this data source by the designed data logger.

Vallox digit is not just only a design parameter but it is the pillar of the design. The importance of the data logger is for the performance enhancement of the data source Vallox digit2 SE. Figure 2 below shows a sample installation design of a Vallox Digit SE ventilation system in a residential house. The ventilation system removes impurities in indoor air, add comfort and improve energy efficiency. This purpose is fulfilled by controlling the following key airflow temperature parameters. The four colors in the figure shows the piping installation of these airflows, which are supply air, extract air, outdoor air, and exhaust air.

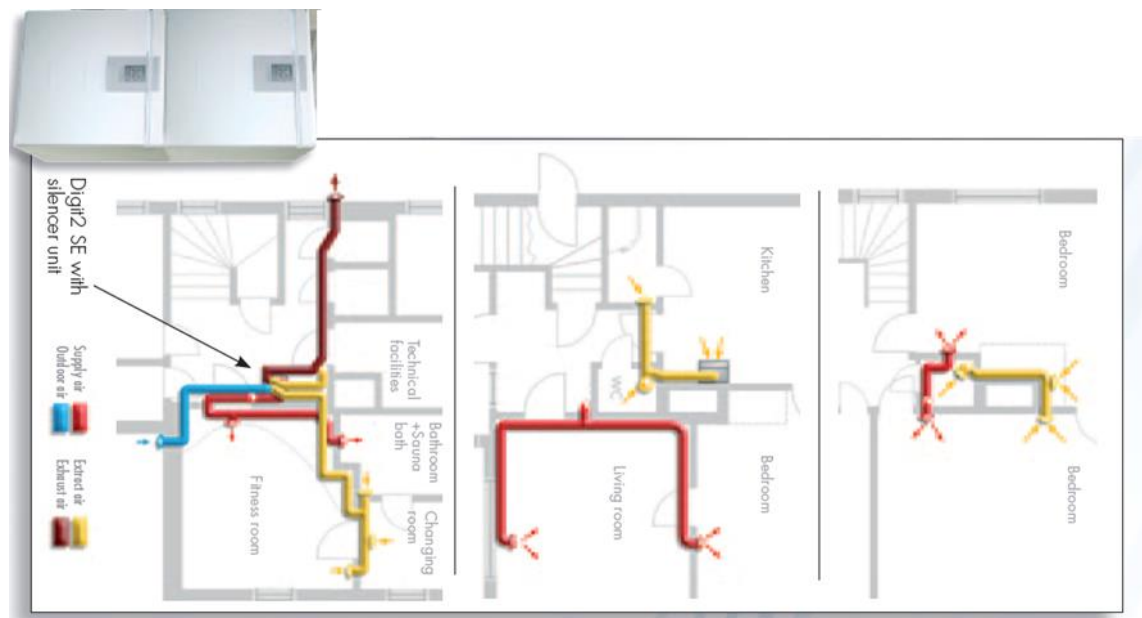


Figure 2. Vallox Digit2 SE home installation reprinted from Operational and Installation Manual (2014) [8].

The ventilation system is controlled based on user setting either manually or automatically to suit personal preference and to ensure the purity of indoor air. The other key data when an automatic control system used are humidity and Carbon dioxide.

Vallox Digit2 SE allows to extract more than 50 data variables from the system but as mentioned earlier in this section the very important data variables are four temperature value at supply air flow, outdoor air flow, extract air flow and exhaust air flow. In addition to the key temperature values percentage of humidity and carbon dioxide are important data wherever automatic control used. [8]

Vallox simulation application on Arduino

Since the Vallox Digit2 SE home automation ventilation system equipment size is quite large it is not reliable to work on it. Therefore, an application which simulates the data sending and receiving protocol of the Vallox Digit2 SE device is designed on Arduino. This design is not part of the thesis. The simulator has been provided by the thesis advisor. Figure 3 below illustrates an Arduino with additional electronic circuits to simulate the exact operation principle of Vallox Digit2 SE. The simulation software flowchart is shown in figure 4.

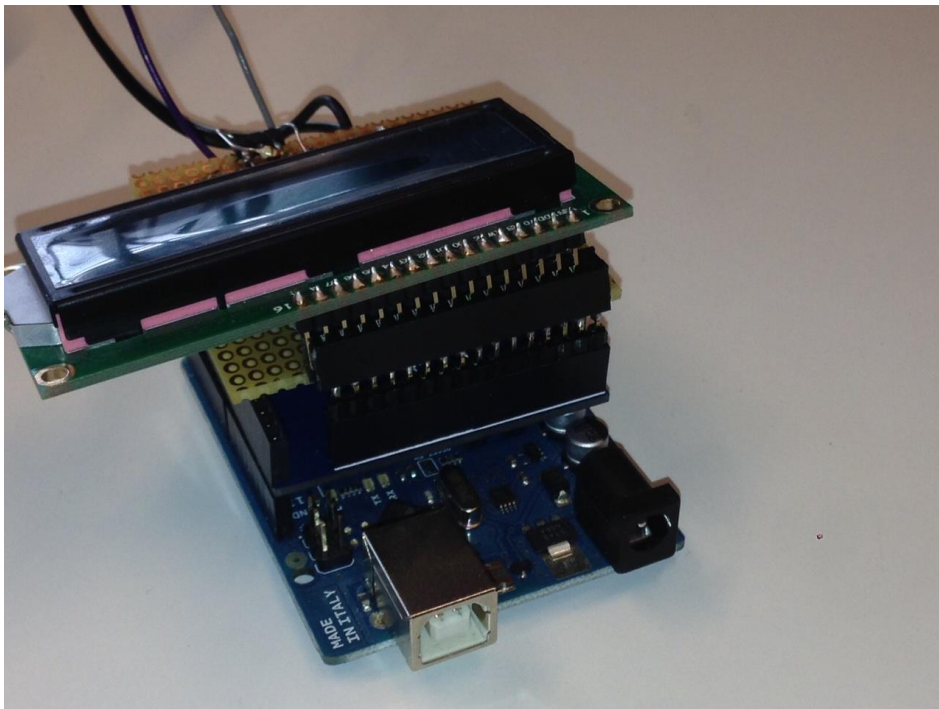


Figure 3. Snapshot of Arduino simulator

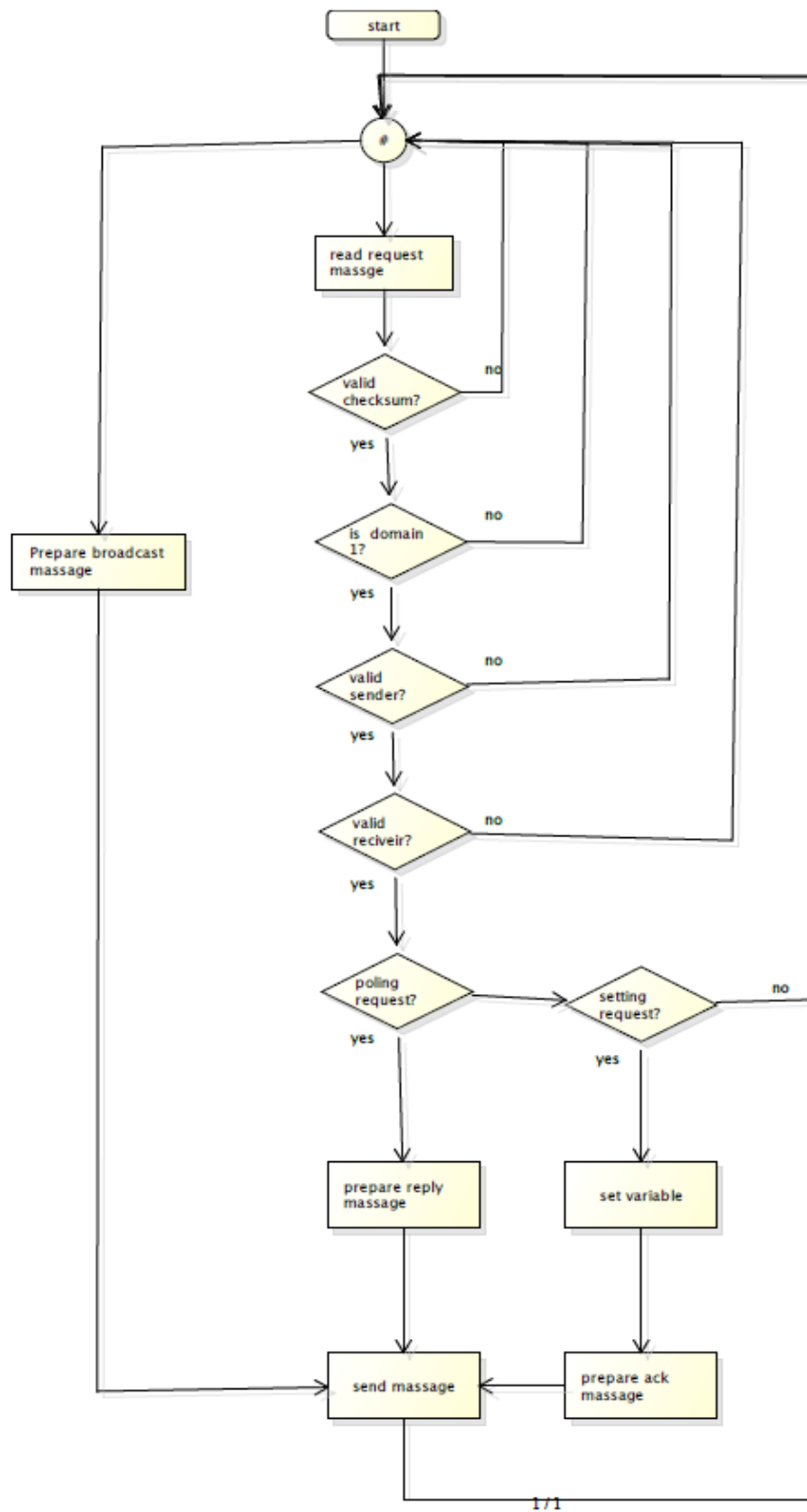


Figure 4. Flowchart of Vallox digit2 SE simulator

3.2.3 Serial-USB Adaptor

Although the universal serial bus (USB) took hold of serial communication by replacing serial port peripherals with USB starting in the late 1990s, the use of serial port would not be obsolete. Since the use of serial ports is inexpensive and less complex to program, it suits to use it for many embedded system and for devices that cannot use USB. The most important serial communication is the ability to use longer cable than USB allows, RS-485 serial interface is used for network communication for control and monitoring application.

While computers have no longer have built-in serial port, it can be easily added by a USB adaptor. The serial to USB adaptors can replace the serial port interface without changing the serial communication protocol and without losing the serial communication physical cabling advantage.

USB-RS485-PCB

USB-RS485-PCB is a cheap PCB adaptor which provides a fast and simple way to connect devices with RS-485 to USB. The PCB contains internal electronic circuit Tx and Rx LEDs to give a visual indication of the data flow. [9]



Figure 5. USB-RS485-PCB. Reprinted from datasheet 2011 [10]

Figure 5 illustrates USB-RS485 adaptor which shows the USB interface. Tx (blue) and Rx (red) LEDs shows sending and receiving traffic respectively.

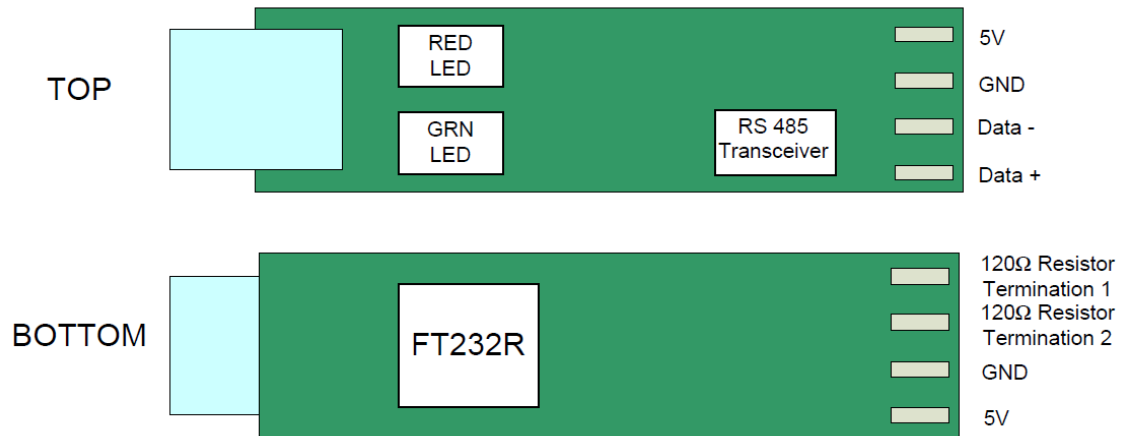


Figure 6. USB-RS485-PCB pin out. Reprinted from datasheet 2011 [10].

Figure 6 illustrates the top and bottom view of the connector pin-out in both sides of the PCB. Each connector pin out is described in table 1.

Table 1. USB-RS485-PCB signal description

Name	Type	Description
GND	GND	Device ground supply pin
Terminator 1	Input	Pin 1 of 120R Terminating Resistor
POWER	Output	Power output +5V when active by default
Data +	Bi-Direction	Data + signal
Data -	Bi-Direction	Data - signal
Terminator 2	Input	Pin 2 of 120R Terminating Resistor

Table 1 shows a list of pin outs and their signal description. Terminator 1 and 2 are only required if the PCB is the first or last device in a multi-drop RS-485 system to meet RS-485 termination requirement.

The circuit schematic inside the Serial-USB PCB board is shown in figure 7.

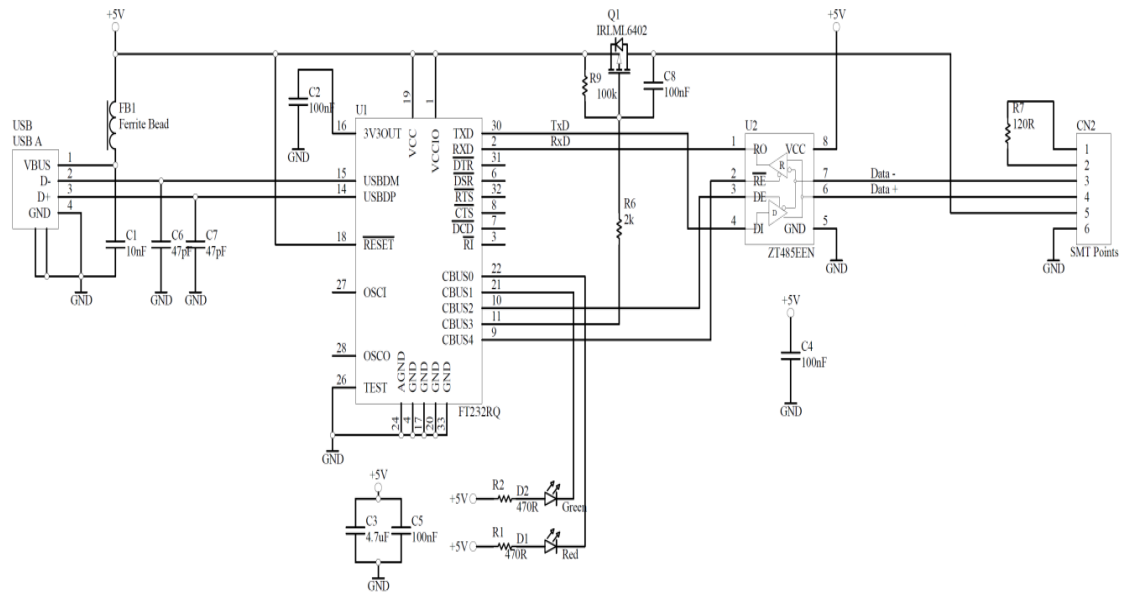


Figure 7. Circuit schematic of USB-RS485- PCB. Reprinted from data sheet 2011 [10].

4 Communication Protocol

4.1 Overview of Communication Protocol

While it is possible to run the whole physical computing project show with only one microprocessors, it is more common and sometimes mandatory to use multiple microprocessor for different part of the project. Therefore, in the design where it is necessary to make a couple of microprocessor devices talk to each other, the most common and easiest way is to use a communication method. For a hardware developer there are many options for computer interfaces.

Asynchronous and synchronous communication

The serial communication protocol used in this project is asynchronous communication which needs identical clock speed on both sides of communication. In asynchronous communication the data transfer frequency should approximately match the clock frequency of both receiver and transmitter. To do that the transmitter start bit adjusts the clock frequency of each computer or device. In synchronous communication the serial interface includes a clock data line which is controlled by one of the computers to insure both the receiver and transmitter to have the same clock. [11]

Half-duplex and full-duplex serial cabling

A half- duplex serial cabling uses only one wire to transmit and receive data, while in a full-duplex cabling there are two wires each dedicated to transmit and receive data.

Based on the desired communication specification one can select the required interface from the list of interfaces provided in table 2 below. [13]

Table 2. Comparison of popular computer interfaces. Reprinted from

Interface	Format	Number of device (max)	Distance (max ft)	Speed (max bps)	Typical Use
RS-232 (TIA-232)	asynchronous serial	2	50 - 100	20k	modem, basic com.
RS-485 (TIA- 485)	asynchronous serial	32 unit load (256 devices)	4000	10M	data acquisition and control system
Ethernet	serial	1024	1600	10G	pc network comm.
IEEE-1394b (FireWire 800)	serial	64	300	3.2G	Video, mass storage
IEEE-488 (GPIB)	parallel	15	60	8M	Instrumentation
I ² C	synchronous serial	40	18	3.4M	Microcontroller communication
Microwire	synchronous serial	8	10	2M	Microcontroller communication
MIDI	serial current loop	2	50	31.5k	Music, Sow control
Parallel Printer	parallel	2 (8 with daisy-chain support)	10 - 30	8M	Printer
SPI	synchronous serial	8	10	2.1M	Microcontroller communication
USB	asynchronous serial	127	16	1.5M	PC peripherals

4.2 Serial Communication

4.2.1 RS-232 Serial communication

RS-232 is an interface that is suitable for many basic communication tasks between two computers or microcontrollers. This section describes RS-232 signals and interfacing options.

RS-232 is designed to handle communications between two devices with a distance limit of around 80 to 130 ft, depending on the bit rate and cable type. RS-232 uses unbalanced, or single-ended, lines. Each signal in the interface has one dedicated line whose voltage is referenced to a common ground. In popular use, RS-232 refers to a serial interface that complies with much of the standard TIA-232-F: Interface between Data Terminal Equipment. The name RS-232 dates to an earlier edition of the standard.

The standard's current publisher is the Telecommunications Industry Association (TIA). Previous versions were a product of the Electronics Industries Association (EIA). A similar standard is encompassed by V.24 and V.28 from the International Telecommunication Union (ITU) and ISO 2110 from the International Organization for Standardization (ISO). The standard defines the names and functions of signals, electrical characteristics of the signals, and mechanical specifications, including pin assignments. Earlier versions did not include all of these items. The addition of new material, such as recommended connectors, documented what had become standard through popular use.

Much of the RS-232 terminology reflects its origin as a standard for communications between a computer terminal and an external modem. A “dumb” terminal contains a keyboard, a display, a communications port for accessing a remote computer. An RS-232 link connects the terminal to a modem, which in turn accesses the phone lines that connect to the remote computer. PCs with modems and network interfaces have made this type of terminal connection nearly obsolete.

These days, an RS-232 port is more likely to connect a PC to an embedded system or to connect two embedded systems. Much of the original RS-232 terminology thus does not apply to modern applications, but the hardware interface remains useful. [12]

4.2.2 RS-485 Serial Communication

The Electronic Industry Association (EIA), the main standard organization for data communication defines RS-series serial communication protocols, since the well-known RS-232 serial interface has many downsides such as:

- Single transmitter and receiver
- Cable length limitation, maximum 20m
- Limited baud rate, max 20kbps

Therefore, the RS-232 standard is replaced by RS-422 and RS-423 standard for better data transmission and multipoint connection support. The RS-485 standard is an advanced version of RS-422, which allows more peripheral interfaces per line.

RS-485 allows bidirectional multipoint transmitter and receiver line communication and eases half-duplex communication. Although its maximum data rate is unlimited, usually it is bounded to 10 Mbps due to the rise time pulses setting. The maximum length possible is 1.2 Km and 32 transmitter/receiver communication is allowed. RS-485 interface is defined by TIA-485-A: Electrical Characteristics of Transmitters and Receivers for Use in Balanced Digital Multipoint Systems. A similar standard is ISO/IEC 8482.1993. A supplementary document is TSB-89-A: Application Guidelines for TIA/EIA-485-A. [13]

RS-485 uses a balanced line so that it can transmit over long distances and have good noise immunity. In a balanced line each signal has a dedicated pair of wires where the voltage on one wire is the negative, or complement, of the voltage on the other wire. The logical 1 or 0 bit of the signal is determined by the difference between the voltages. This type of transmission is called differential signaling. TIA-485-A designates the two lines in a differential pair as A and B. Unlike RS-232 a single voltage +5V or lower voltage supply is required for RS-485, so therefore the receiver and driver chip are inexpensive compared to RS-232. [12, 13]

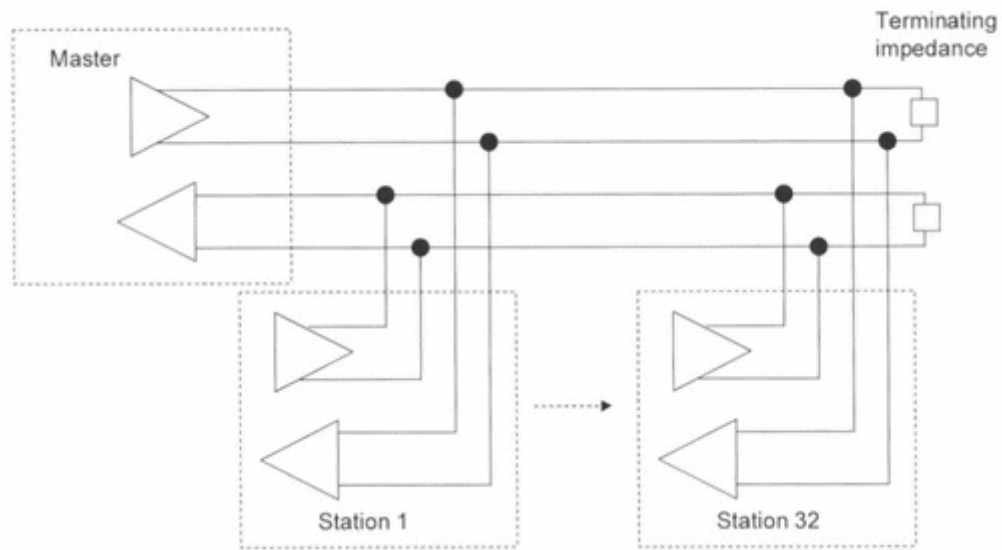


Figure 8. Four wire RS-485 connection (full-duplex). Reprinted from White Paper (2015) [14].

Figure 8 illustrates the master communicating with a slave (station 1 up to 32). Only the master communicates to all the slaves.

Half duplex RS-485 communication

The serial communication protocol used in this project is a two-wire half-duplex protocol. A connection of a half-duplex serial communication is illustrated in Figure 9.

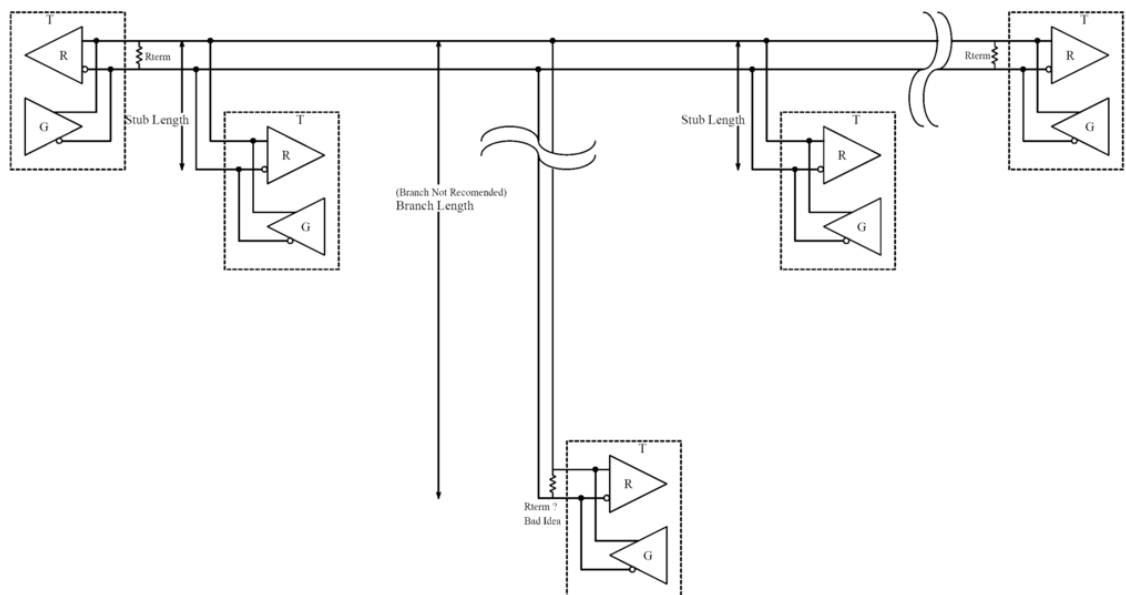


Figure 9. Two wire RS-485 connection (half-duplex). Reprinted from White Paper (2015) [14].

In an RS-485 connection the generator or driver, receiver and transceiver are the defined interfaces. RS-485 is inherently half-duplex. It means that when one driver or transceiver transmits the other drivers must be passive or disconnected from the RS-485 network. Moreover, a higher level of addressing schema in the software protocol is required to implement multiple devices communication allowed by the RS-485 network to ensure the message transmits to the right device. [14]

Collision detection is not part of the RS-485 standard. The message might be corrupted, if multiple drivers start to transmit at the same time. The RS-485 standard protocol does not detect whether the message is corrupted or not. Therefore, a higher-level software protocol is needed to define a way for a device to verify that the correct data has been sent. A simple example is a query/response protocol. [14]

Collision detection and data corruption are detected by the software protocol by using a query/ response protocol, for example device A sends a query to device B in an RS-485 network for a data by its address and device B responds to the requested data. If device A does not receive the requested data because of collision or any other error mechanism, the query will be sent again. Adding a CRC or checksum at the end of the data being transmitted is another error detection mechanism. The CRC is generated by similar software logic in both sender and receiver software side to verify validity of the data. It is engineer who writes the software that has responsivity to implement collision detection and data validity, since it is beyond the RS-485 standard. [14]

4.2.3 Serial Communication Protocol

Protocol is the agreement about how messages are sent required to communicate between two devices. In section 2.2.2 lower layer agreements, RS-232 and RS-485, were discussed. Physical connection agreement, number and purpose of each wire and connector types are established. The agreement for the electrical characteristics of the voltage used, timing, meaning, grouping and formatting of pulses are also set.

The communication protocol is organized in a layer of agreement to focus on the upper layer after selecting a convenient lower layer protocol. Beside the lower layer physical and electrical specification, an RS-485 software protocol at the upper layer protocol assigns each bit a meaning. Although it is possible to use all layers of the protocol, each

project requires its own message syntax and format. In order to achieve full communication, protocol software on both sides of the device should be created.

The RS-485 protocol defines many specifications, but the content and the formatting of the message are not specified. The protocol software, by studying the documentation of the end devices, can invent any schema for formatting message as long as both sides are consistent. The two techniques to achieve agreement are Punctuation and Call-and-response.

Punctuation

When data is transmitted at the physical layer, an offset might occur. The offset bit causes the whole data to have a different meaning. Therefore by putting a punctuation between a set of messages as a separation, the message is understood in an organized way. This method works by adding a character or delimiter in every end of transmission of a set of signals, so that the offset or delay at the beginning does not affect the whole data.

Call-and-response

A better way to communicate is to take a more organized approach. In a call and response method the device sends a call and the other device responds to the requested call. The sequence of the process is

1. the sender waits with a blocking function
2. the receiver sends on arbitrary request call
3. the sender sends bytes of data
4. the receiver reads the data

This process loops infinitely. In a smarter way the receiver sends a specific request data and the sender sends the required parameter.

When working on an existing device, it may not be possible to design protocol software. Therefore, instead of creating software on both sides the device should have to use either an industry-wide open standard or a proprietary standard given by the company. The Modbus communication protocol and Hayes protocol from Hayes Corporation are

examples of an industry-wide standard used by many industries. In this project a proprietary standard from Vallox Oy is used to demonstrate a higher-level protocol.

Vallox serial protocol

The message in this protocol is organized in a packet. There are two types of packets: polling packet and setting packet. There are also one byte long ack packets, but these are only sent by the master mainboard to inform a sender that the packet is received. [15]

Typical polling packet

Domain	Sender	Receiver	Poll request	Polled variable	Checksum
--------	--------	----------	--------------	-----------------	----------

Typical Setting packet

Domain	Sender	Receiver	Variable	Data	Checksum
--------	--------	----------	----------	------	----------

Domain: always 1

Sender: a sending module's address: 11 = mainboard, 21-29 = different individual control panels

Receiver: a module targeted by packet: 10 = to all master and slave mainboards, 11 = to master mainboard, 20 = to all control panels, 21 – 29 = different individual control panels

Poll Request: is always 0. It defines the packet as a polling packet.

Polled Variable: defines polled variable.

Variable: a variable described by the variable table in the appendix.

Data:

Checksum: is always automatically calculated by software. Every 8-bit byte in the packet is added together and carry bits are discarded (only 8 lower bits from 16-bit result are used).

Master and Slave mainboards are inside the ventilation machine. Master mainboard's address is always 11. There can be only one master mainboard in the same RS-485 line. Only the master mainboard can send ack bytes when its variables are set or answer to poll requests. Slave mainboards do not have individual addresses, and their common

address is 10. Slave mainboards never transmit anything, so they cannot be polled either.

Control panel's addresses can be 21-29. Address 20 targets every module in 21-29. Control panels do not reply to polls or send ack bytes and they can poll variables from the master mainboard.

The LON gateway's address is always 28. The LON gateway does not reply to polls or send ack bytes. It only polls variables from mainboard.

5 Data Logger Software

5.1 Overview of Data Logger Software

For every different type of data logger there are also many different data analyses and graphing software packages. In general, the software is selected for the desired platform that is Windows or Mac based, depending on the requirements. The software makes the data logger operational by quickly and easily performing tasks such as configuring parameters and timestamp length, deploying the device, and offloading data. [5]

In addition to enabling communication between other devices, the data logging software package should also allow data plotting capabilities, by easily merging, appending, and cropping data, and enabling one to easily export data to other programs, such as Microsoft Excel, for further analysis. [5]

5.2 TaloLogger

TaloLogger is a donation ware, distributed as freeware for non-commercial use. TaloLogger is a configurable freeware data logger application for a home automation system. Based on the core configuration it activates the selected data source, retrieves the data point values at configured intervals from multiple sources, maps the retrieved data to data labels and stores in a data store using the data label. [15]

TaloLogger is a modular application developed in Python. Although the application could be usable in all platforms, TaloLogger is mainly developed to run on a UNIX platform.

Since the modularity of the application additional software package required for some data source and data store module. [15]

A list of currently supported data source is as follows:

- Runnable shell script or executable program.
- OneWire sensors using either OWFS OneWire file system, DigiTemp or EDS OW-SERVER-ENET-2 device
- Modbus devices, serial RTU and ASCII modes and Modbus TCP over Ethernet
- Ouman controllers through Ouman serial cable connection
- Ouman EH800 series controllers through Ethernet connection
- Thermia or Danfoss heat pump controllers connected with serial connection or ThermlQ interface.
- Rego600-series controllers connected with serial connection
- Rego800 and Rego1000 series controllers equipped with CAN bus, using CAN232/CANUSB interface.
- Ekowell heat pump controllers using serial connection
- Nibe heat pump controller through serial connection, modbus or OpenHab NibeGW (UDP over network).
- Siemens SmartWeb module using network (web) connection
- Telldus Tellstick Duo and wireless sensors (temperature, humidity, rain, wind) using telldus-core library.
- Telldus TellStick Net wireless sensor devices, using Telldus Live API (over network from the remote service).
- Raspberry Pi GPIO inputs (with taloLoggerPi).
- Enervent EDA using Modbus Serial RTU from device bus and ModbusTCP over network from Enervent Freeway WEB interface.

A list of currently supported data stores is as follows

- MySQL database table,
- File log with configurable format (example Excel CSV-format),
- RRDTool database file and
- SQLite database table

TaloLoggerGraph

TaloLoggerGraph is a graphing application which draws graphical plots of a log data from supported databases. The application provides a plot of history and recent data from a database on a time axis to the user. The data must contain a timestamp of the recorded data from one database table. The image graphs are viewed in a web browser. [15]

TaloLoggerGraph is developed in PHP and implements the JpGraph and D3 Data-Driven Documents library to draw the plot graphs. MySQL and SQLite are the currently supported databases. TaloLoggerGraph can also be used to draw graphs from any suitable data other than the TaloLogger application.

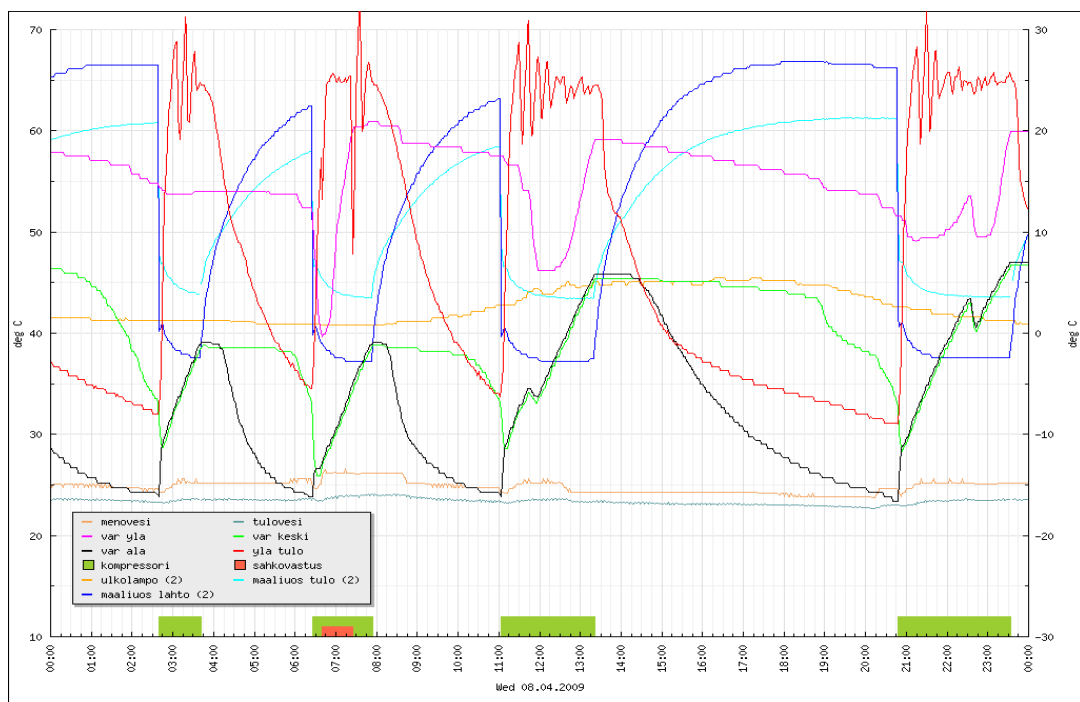


Figure 10. TaloLogger graph. Reprinted from [15]

TaloLoggerPi

TaloLoggerPi is a logging system for Raspberry Pi devices intended to be installed on a clean Raspbian operating system distribution. TaloLoggerPi consists of TaloLogger and TaloLoggerGraph applications and a set of the required supporting applications to run on a Raspberry Pi computer as a simple data logger. [15]

The TaloLoggerPi install script will install the following Raspbian packages using the apt-get command:

- daemon tools
- python-serial, python-imaging, python-mysqldb, python-sqlite
- apache2
- php5, php5-mysql, php5-sqliteand php5-gd
- sqlite3
- data source related software package such as owfs and digitemp
- data store related software packages such as rrdtool
- i2c-tools
- python-oauth

5.3 TaloLogger Data Source Plugin Software Design

The framework of the software is adopted from open-source home automation data logger software called TaloLogger. TaloLogger is modular open source software developed in python programming language.

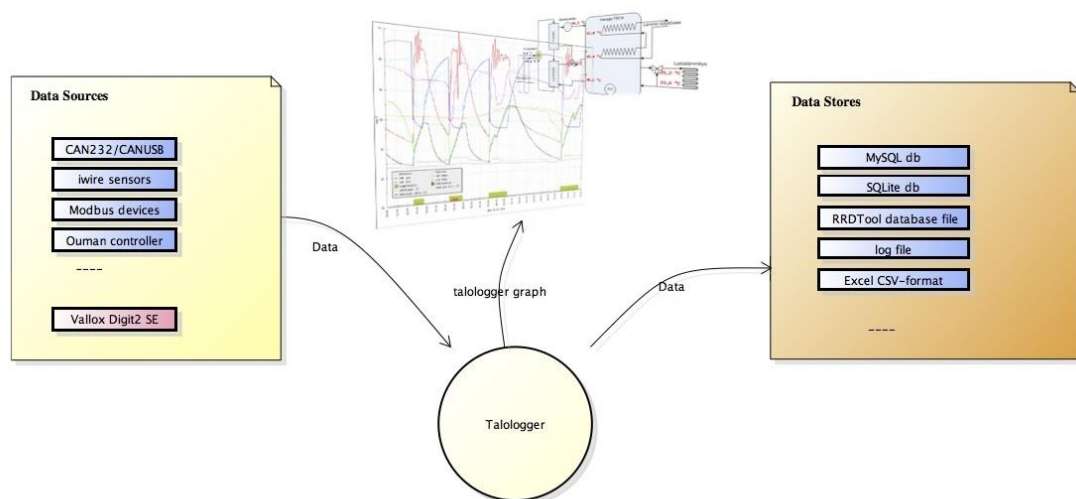


Figure 10. TaloLogger software framework

Figure 10 illustrates the framework of the software and the core software components. The main software components are:

1. Data source related software components: these software components contain the communication protocol to read a data from the data source device which is connected through a serial communication protocol, for example to the Raspberry Pi where the TaloLogger is installed.
2. Data store related software component: the data extracted from the data source will be stored into a data store selected in the TaloLogger configuration file. TaloLogger Configuration file allows to select a data store from the list supported data store.
3. The core TaloLogger: On this part of software component the data extracted from the data source is labeled and mapped to the configured data store. Additionally the TaloLogger graph demonstrates the data in a graphical way from the data store.

6 Project Implementation and Result

6.1 Overview of the Design

The prototype data logger is designed to be embedded on is home automation ventilation system. Figure 11 shows the overall design layout of data logger embedded in Vallox Digit2 SE. Of course the Vallox Digit2 SE machine is replaced by system simulating device built on Arduino for design reliability.

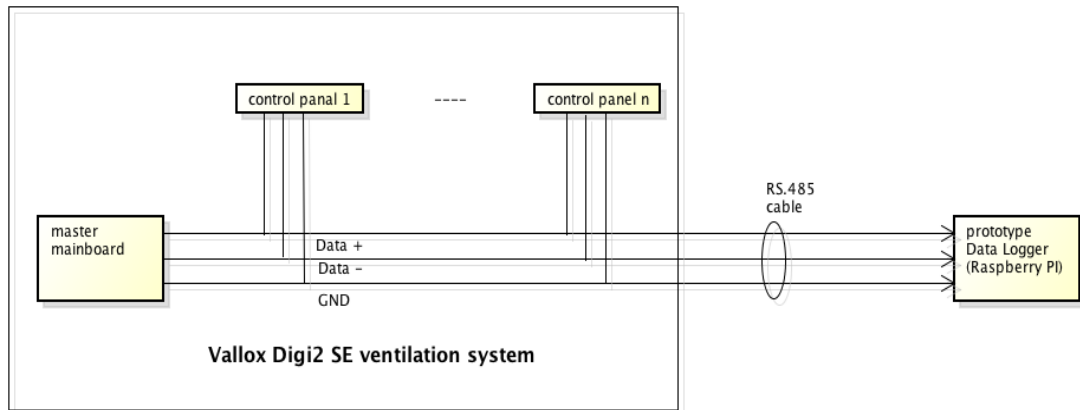


Figure 11. Design layout of the data logger

In order to implement data logger functionality in the home automation system a software plugin on the TalLogger open source software is designed. Usually data loggers have multiple communication with sensors for multiple data types but in this prototype design all data types can be accessed through a single RS-485 communication protocol from a single data source with multiple sensors which is a Vallox digit2 SE home ventilation automation system.

Using RS-485 serial communication protocol given by the company the data source plug in is designed. As mentioned in chapter 4 the data sent to RS-485 trunk by the Arduino, Vallox Digit2 SE simulator, is a mixture of broadcast message, acknowledge message, and polling reply message. The software plugin is developed to analyze the data read from the data source, verify data integrity and validity, process the error, send polling and setting a request. In section 6.2 the major functionalities of the plugin are described.

While designing and coding the plugin software different approaches are considered. TaloLogger allows to add a data source module by implementing plugin software either in Python similar to other data source modules or by designing any runnable shell script using C language. Since TaloLogger is developed in Python and for easy integration to the core software component, designing the plugin in Python is decided.

Following a similar pattern to the already supported data source plugin the software component related to Vallox Digit2 SE is designed in two Python files. Moreover the data source is introduced in core TaloLogger software files. The simulator Arduino device can be connected to a laptop to be tested by a terminal program. In addition to the laptop used to test each plugin components software before implementing in to TaloLogger the following resources are used:

Required Resource Components

- Raspberry Pi
- HP L2245wg 22' monitor
- HDMI to DVI convertor to connect Raspberry Pi with the monitor
- Vallox Digit2 simulator device built in Arduino
- USB port extension for extra port to connect the keyboard, mouse and the simulator to a Raspberry Pi
- Serial to USB convertor.

6.2 Vallox Digit Serial Protocol Plugin

6.2.1 Packet Reading and Arranging Software Component

Packet reading and arranging software component of the plugin sorts the data in a proper way for further analysis. Although more than 50 variables can be read from the data source, in this prototype the focus is on temperature values in four important places. The core temperature values such as exhaust air temperature, extract air temperature, outdoor air temperature and supply air temperature discloses the overall condition of the ventilation and the machine itself.

In addition to broadcast messages Vallox Digit2 SE main board responds to polling requests from a control panel or the data logger. The core air temperatures may not be

retrieved properly from the broadcast message and therefore by sending a polling request to the mainboard it is possible to receive the data for the requested variable. This software component sends a polling request until finding the requested variable data from the response.

Initially the software component designed in such a way that it reads many bytes which contain broadcast messages, acknowledge messages for setting requests, reply message for polling requests and sometimes fault bytes. After reading these mixed data the desired packet is retrieved using a software logic. The flowchart shown in Figure 12 is the initial packet retrieving software logic which is not implemented in the plugin.

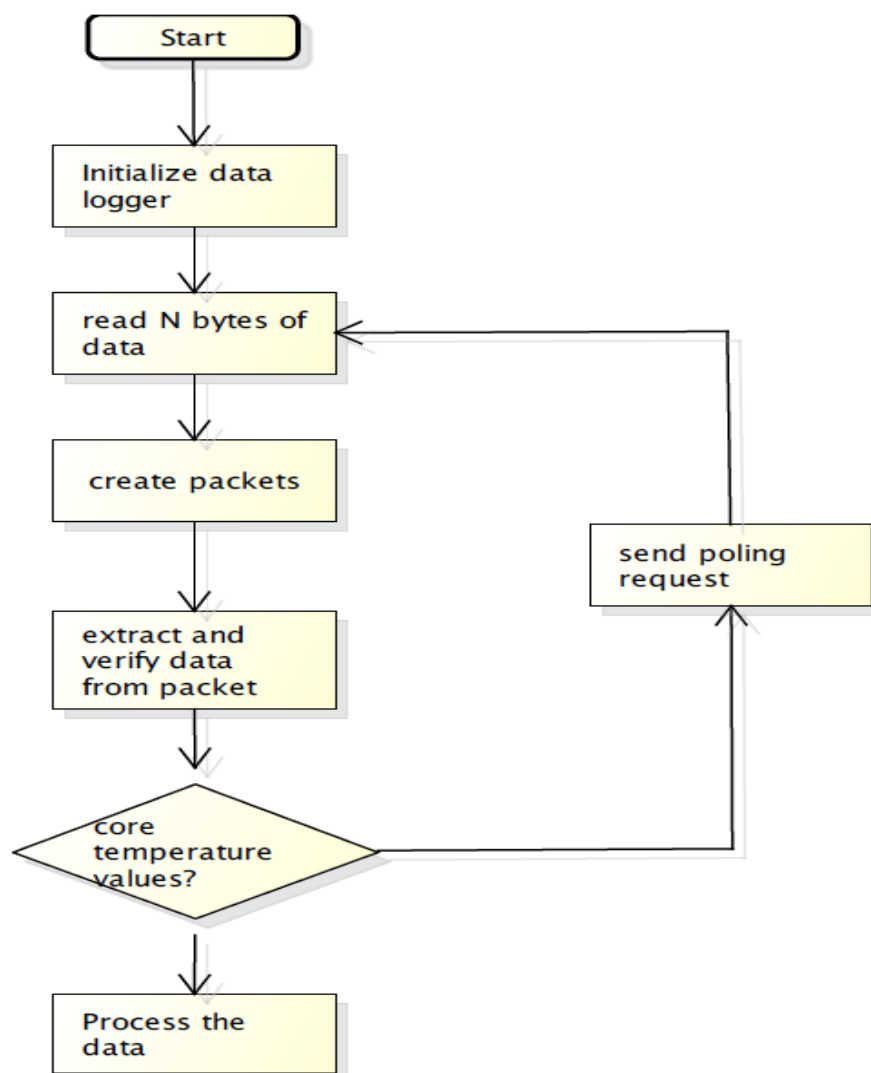


Figure 11. Initial design flowchart of polling request

Afterwards it is understood that a better way to retrieve an arranged packet is by sending a polling request and process the response without trying to read the broadcast message. Figure 12 below illustrates packet reading and the processing software component used in the plugin software.

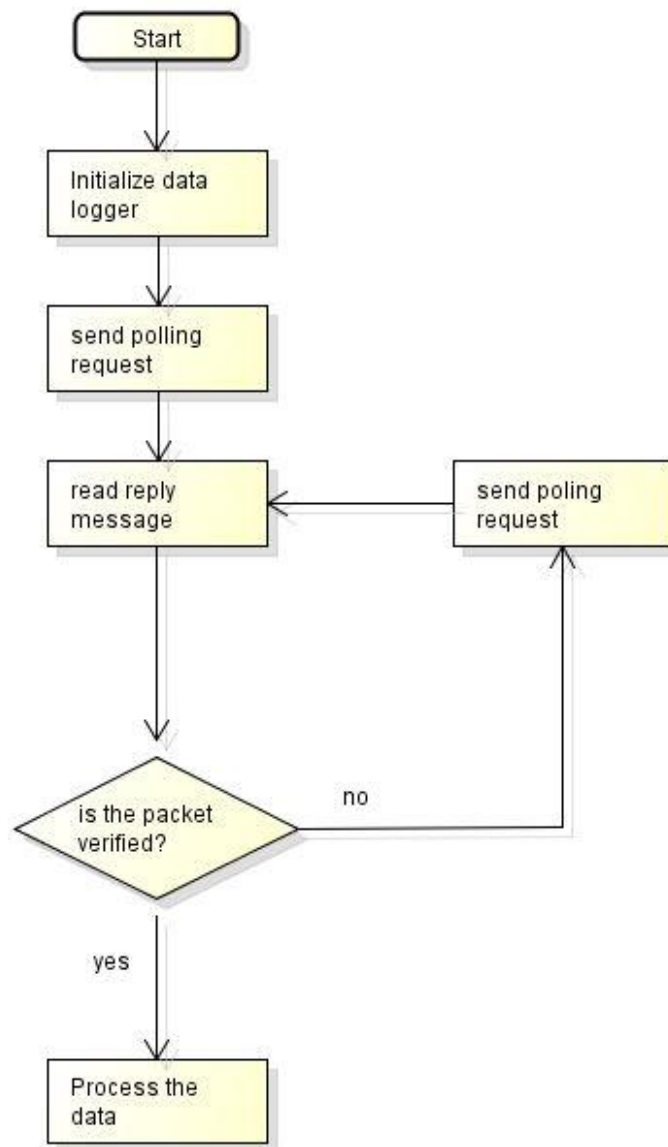


Figure 12. Simplified polling request flowchart

6.2.2 Packet Verification Software Component

As mentioned in chapter 4 in Vallox Digit2 SE the Vallox packet has six bytes. The following tables show a typical polling packet and typical setting packet respectively:

Polling packet:

Domain	Sender	Receiver	Poll request	Polled variable	Checksum
--------	--------	----------	--------------	-----------------	----------

Setting packet:

Domain	Sender	Receiver	Variable	Data	Checksum
--------	--------	----------	----------	------	----------

The last byte, checksum, is a value automatically calculated by the sender software, so that the receiver can check the integrity of the data. Listing 1 shows a snippet of code used to check the validity of Vallox data received.

```
# returns integer 0 = ok, but not ready 1 = ok 2 = error
def checkvalloxMessage(data):
    if len(data) <= 0:
        return 0
    #check if the domain or the first byte is 0x01
    if data[0] != chr(0x01):
        return 2
    # if packat is already longer or less than it should be
    if len(data) > 6:
        return 2
    if len(data) < 6:
        return 0
    # if packat checksum does not match
    crc = 0
    for i in range(1, len(data)-1):
        crc = crc + ord(data[i])
    if (crc & 0xFF) != ord(data[len(data)-1]):
        return 2
    return 1
```

Listing 1. Packet verification checker function

6.2.3 Data Extraction Software Component

The Vallox packet contains six bytes so that each byte represents a different component value. For example the first byte represents, the domain which is always 01, and the second and the third bytes represent the sender and receiver address respectively. In a setting packet the fourth byte is a unique variable representing a data point and the fifth byte in the array is the corresponding data value of the variable. As mentioned in section 6.2.2, the last byte is checksum which is used by the software to check data integrity. After checking the domain and checksum for packet verification and integrity, the variable and data part as the key and value from the setting packet are extracted by this software component.

The data read by the logger is not always the final understandable data. For example the temperature value is a hexadecimal value to be converted to degree Celsius based on the table in appendix 1.

6.3 TaloLogger Core Configuration

6.3.1 Data Source Configuration

The TaloLogger application supports multiple data sources and it is also possible to add an additional data source such as Vallox Digit2 SE plugin added in this thesis. The preferred data source can be enabled in a configuration file. The default configuration file delivered in the application archive as a base for the application configuration is *configuration.conf*. Multiple data sources can be configured to use the same data source module that have separate configuration values.

Syntax: @DATASOURCE=sourcemodule:modulename

Sourcemodule - Data source module. Each data sources has its own source module name

Modulename - Identifying a unique name given to this module in this configuration. Module specific configuration parameters refer to this module name. Measurement points refer to this module name.

To configure 1 Vallox Digit2 SE as a data source

@DATASOURCE=VALLOXDIGIT:VALLOXDIGIT1

6.3.2 Data Store Configuration

The desired data store can be added as syntax.

Syntax: @DATASTORE=storemodule:modulename

Storemodule - Data store module. Possible values are:

FILESTORE - File store module

RRD - RRD store module

MYSQldb - MySQL database store module

SQLITEDB - SQLite database file store module

Modulename - Identifying unique name given to this module in this configuration. Module specific configuration parameters refer to this module name.

If no data stores are configured, TaloLogger will output the measured data to the application log.

6.3.3 VALLOXDIGITSerial Configuration

The VALLOX serial device type can be configured in the following syntax. For the time being we have one device type which is Digi2 SE.

Syntax: VALLOXDIGIT:DEVICE = DIGIT

Serial port address/name where the VALLOXDIGIT device is connected:

Syntax: VALLOXDIGIT:SERIAL_PORT = /dev/ttyUSB0

6.3.4 Measurement Points and Data Store Key Configuration

Measurement points are the parameters required to be logged. In this thesis the four key temperatures are measurement points.

Syntax: @MEASURE=key:source.point

Key - title or key for the measured value, key is used as label in file logging, as a database column name in DB logging and as ds-name in rrdtool logging

Source - modulename for retrieving the value. Source modules and their names are configured using the DATASOURCE configuration file directives.

point - name of the module specific measurement

According to the syntax the required four temperature data point in this prototype design are added in the following way

```
@MEASURE=outemp:VALLOXDIGIT.Outside temp
```

```
@MEASURE=intemp:VALLOXDIGIT.Inside temp
```

```
@MEASURE=Exhtemp:VALLOXDIGIT.Exhaust temp
```

```
@MEASURE=inctemp:VALLOXDIGIT.Incoming temp
```

The plugin is designed to allow to monitor all measurement points and variables included in the protocol, although only the four important temperatures are included in the configuration as shown above. For example if anyone wants to monitor high byte CO₂ percent or relative humidity percent from sensor one, he/she can add the measurement configuration as below in the configuration file. All measurement points are included in the source code and appendix 2 to be used by the configuration.

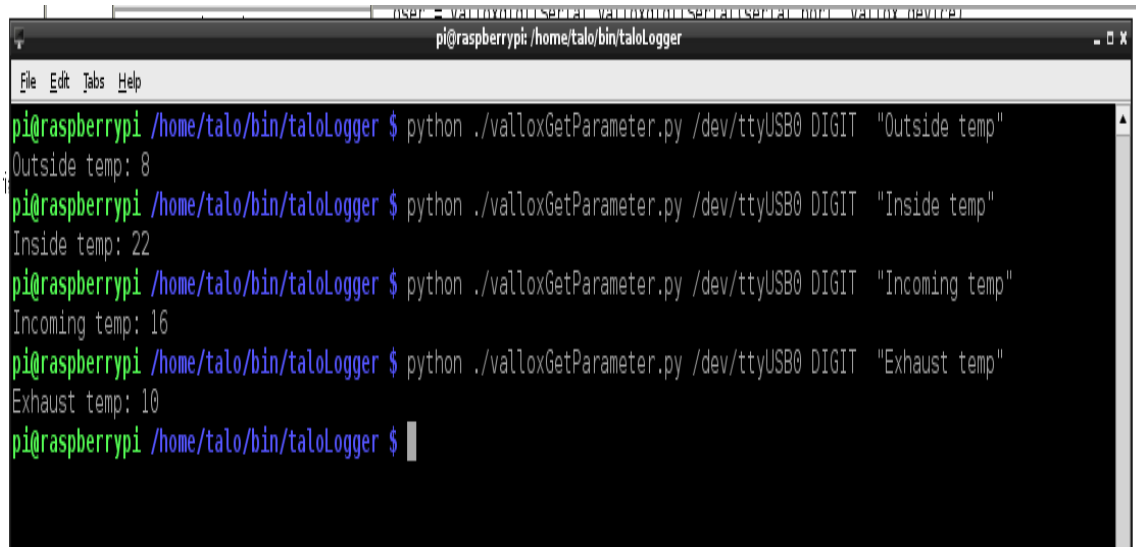
```
@MEASURE=CO2hb:VALLOXDIGIT.co2 level hbyte
```

```
@MEASURE=1RHper:VALLOXDIGIT.%RH from sensor 1
```

6.4 Testing

6.4.1 Testing the Controller Using a Script

A script to read Vallox Digit2 values and parameters using the serial connection to a simulated Vallox Digit2 SE controller is designed to test the reply from the controller. Figure 13 illustrates the respond from the controller to the requested data point temperature values. The designed tool script takes the device type, serial port and desired data point as a parameter respectively.



```

pi@raspberrypi /home/talo/bin/taloLogger
File Edit Tabs Help
pi@raspberrypi /home/talo/bin/taloLogger $ python ./valloxGetParameter.py /dev/ttyUSB0 DIGIT "Outside temp"
Outside temp: 8
pi@raspberrypi /home/talo/bin/taloLogger $ python ./valloxGetParameter.py /dev/ttyUSB0 DIGIT "Inside temp"
Inside temp: 22
pi@raspberrypi /home/talo/bin/taloLogger $ python ./valloxGetParameter.py /dev/ttyUSB0 DIGIT "Incoming temp"
Incoming temp: 16
pi@raspberrypi /home/talo/bin/taloLogger $ python ./valloxGetParameter.py /dev/ttyUSB0 DIGIT "Exhaust temp"
Exhaust temp: 10
pi@raspberrypi /home/talo/bin/taloLogger $

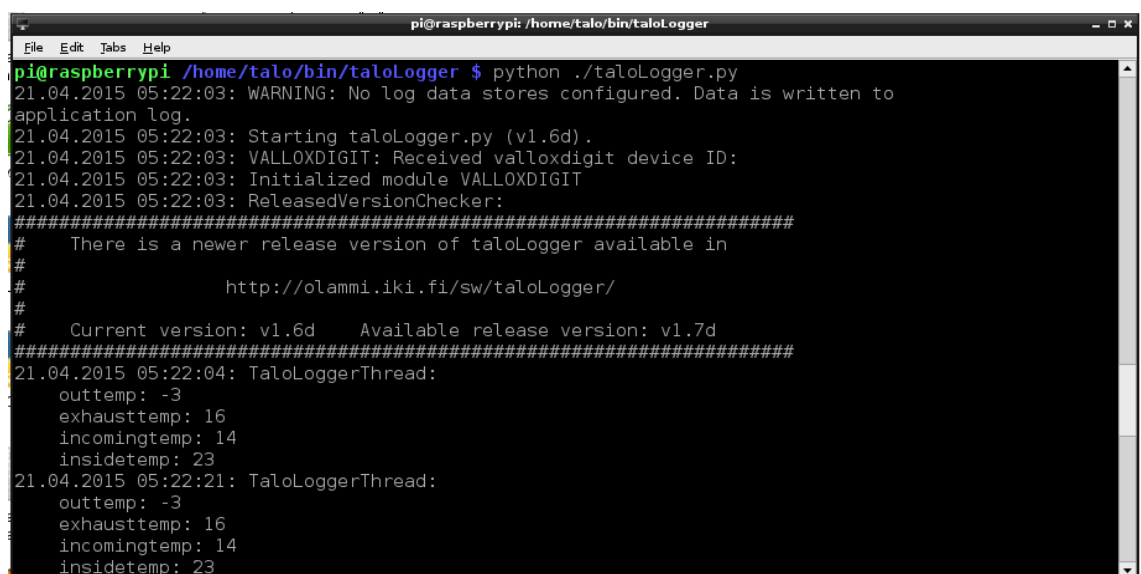
```

Figure 13. Testing using custom script

The initial testing using a custom script shows the Vallox data can be read from the simulator. Therefore, it is possible to configure the TaloLogger software in order to read Vallox data in a proper way.

6.4.2 Testing Using TaloLogger Core Configuration

After the Vallox Serial protocol software plugin has been developed, all data source and data point measurements configuration parameters are configured in a core TaloLogger configuration file.



```

pi@raspberrypi /home/talo/bin/taloLogger
File Edit Tabs Help
pi@raspberrypi /home/talo/bin/taloLogger $ python ./taloLogger.py
21.04.2015 05:22:03: WARNING: No log data stores configured. Data is written to
application log.
21.04.2015 05:22:03: Starting taloLogger.py (v1.6d).
21.04.2015 05:22:03: VALLOXDIGIT: Received valloxdigit device ID:
21.04.2015 05:22:03: Initialized module VALLOXDIGIT
21.04.2015 05:22:03: ReleasedVersionChecker:
#####
#   There is a newer release version of taloLogger available in
#
#       http://olammi.iki.fi/sw/taloLogger/
#
#   Current version: v1.6d   Available release version: v1.7d
#####
21.04.2015 05:22:04: TaloLoggerThread:
    outtemp: -3
    exhausttemp: 16
    incomingtemp: 14
    insidetemp: 23
21.04.2015 05:22:21: TaloLoggerThread:
    outtemp: -3
    exhausttemp: 16
    incomingtemp: 14
    insidetemp: 23

```

Figure 14. Output of TaloLogger logging response

Figure 14 shows the logging procedure where the data is written in an application log since no data store is configured. The configured measurement data point values are printed in the application log which shows the plugin works properly based on the configured parameters.

7 Conclusion

The data logger that was designed on Raspberry Pi is running successfully. The primary goal of the project was to design a prototype data logger, to be embedded into a home automation system to prove the home automation system could be improved by adding a data logger to the home automation system. How an automated data collecting device improves a home automation system was not discussed in this project, since it is beyond the scope of the project goals. However a cost-effective and powerful data logger was designed and tested successfully.

The findings in this study provide enough information to create an economical, modular and extensible data logger device with a better functionality compared to available data loggers on the market considering the costs as reference. It is recommended that users add a data logger to their home automation system, to get valuable data to understand what is going on inside the system.

Although the prototype design is an independent data logger which communicates using serial communication protocol with the home automation system, it is possible to include the design inside home automation system. Instead of users the designing or buying designed data logger independently, it is recommended that the home automation company include the data logging functionality into the system. So the cost of hardware and software design could be reduced more and the system would become smarter.

References

- 1 Sio-lin Ao, Burghard Rieger. Machine Learning and System Engineering. Dordrecht Heidelberg London New York: Springer science + business media B.V; 2010.
- 2 Juing-Huei Su, Chyi-Shyong Lee, Wei-Chen Wu. The Design and Implementation of a Low-cost and Programmable Home Automation Module; IEEE Transactions on Consumer Electronics, NOV 2006; 52(4): 1239-1244.
- 3 A.J. Bernheim Brush, Bongshin Lee, Ratul Mahajan, Sharad Agarwal, Stefan Sarioiu, and Colin Dixon. Home Automation in the Wild: Challenges and Opportunities. CHI 2011, May 7–12, 2011, Vancouver, BC, Canada.
- 4 A. Alheraish. Design and Implementation of Home Automation System. IEEE Transactions on Consumer Electronics, Nov. 2004; 50(4): 1087-1092.
- 5 Data Logger Basics
URL: http://www.onsetcomp.com/learning/white_papers
Accessed 28 of March 2015.
- 6 Robinson Andrew, Cook Mike: Raspberry Pi Projects. Somerset, NJ, USA: Wiley; 2013.
- 7 Raspberry Pi
URL: <https://www.raspberrypi.org>
Accessed 28 of March 2015.
- 8 Vallox Ventilation Renovation
URL: http://www.vallox.com/tiedostot/4/documents/Esitteet_EN/030311_pk-esite_E-web.pdf. Accessed 2 March 2015.
- 9 USB to RS485 UART serial convertor PCB Datasheet: Future Technology Devices International Limited (FTDI); Glasgow G41 1HH United Kingdom; 2008,
URL: http://www.ftdichip.com/Support/Documents/DataSheets/Cables/DS_USB_RS485_PCB.pdf. Accessed 2 April 2015.
- 10 FTDI USB-RS485-PCB
URL: http://shop.clickandbuild.com/cnb/shop/ftdichip?productID=102&op=catalogue-product_info-null&prodCategoryID=91. Accessed 2 April 2015.
- 11 Buchanan WJ. The handbook of data communication and networks (2nd edition). Boston, Dordrecht, London: Kluwer Academic Publication; 2004.
- 12 Igoe Tommy, O'Sullivan Dan. Physical Computing. Boston, MA, USA: Course Technology; 2004.
- 13 Axelson, Jan. Serial Port Complete: COM Ports, USB Virtual COM Ports, and Ports for Embedded Systems. 2nd Ed. Madison, WI, USA: Lakeview Research; 2007.

- 14 Serial Programming/RS-485
URL: http://en.wikibooks.org/wiki/Serial_Programming/RS-485
Accessed 2 April 2015.
- 15 TaloLogger How To
URL: <http://olammi.iki.fi/sw/taloLogger/>
Accessed 2 April 2015.

NOTE: The URLs cited in this list were functional in May 2015.

Appendix 1: Temperature Conversion Table

APPENDIX A

CONVERSION TABLE: NTC SENSOR VALUES <--> °C

HEX	DEC	°C	HEX	DEC	°C	HEX	DEC	°C	HEX	DEC	°C
00	0	-74	40	64	-12	80	128	9	C0	192	34
01	1	-70	41	65	-12	81	129	9	C1	193	34
02	2	-66	42	66	-12	82	130	9	C2	194	35
03	3	-62	43	67	-11	83	131	10	C3	195	35
04	4	-59	44	68	-11	84	132	10	C4	196	36
05	5	-56	45	69	-11	85	133	10	C5	197	36
06	6	-54	46	70	-10	86	134	11	C6	198	37
07	7	-52	47	71	-10	87	135	11	C7	199	37
08	8	-50	48	72	-9	88	136	11	C8	200	38
09	9	-48	49	73	-9	89	137	12	C9	201	38
0A	10	-47	4A	74	-9	8A	138	12	CA	202	39
0B	11	-46	4B	75	-8	8B	139	12	CB	203	40
0C	12	-44	4C	76	-8	8C	140	13	CC	204	40
0D	13	-43	4D	77	-8	8D	141	13	CD	205	41
0E	14	-42	4E	78	-7	8E	142	13	CE	206	41
0F	15	-41	4F	79	-7	8F	143	14	CF	207	42
10	16	-40	50	80	-7	90	144	14	D0	208	43
11	17	-39	51	81	-6	91	145	14	D1	209	43
12	18	-38	52	82	-6	92	146	15	D2	210	44
13	19	-37	53	83	-6	93	147	15	D3	211	45
14	20	-36	54	84	-5	94	148	15	D4	212	45
15	21	-35	55	85	-5	95	149	16	D5	213	46
16	22	-34	56	86	-5	96	150	16	D6	214	47
17	23	-33	57	87	-4	97	151	16	D7	215	48
18	24	-33	58	88	-4	98	152	17	D8	216	48
19	25	-32	59	89	-4	99	153	17	D9	217	49
1A	26	-31	5A	90	-3	9A	154	18	DA	218	50
1B	27	-30	5B	91	-3	9B	155	18	DB	219	51
1C	28	-30	5C	92	-3	9C	156	18	DC	220	52
1D	29	-29	5D	93	-2	9D	157	19	DD	221	53
1E	30	-28	5E	94	-2	9E	158	19	DE	222	53
1F	31	-28	5F	95	-2	9F	159	19	DF	223	54
20	32	-27	60	96	-1	A0	160	20	E0	224	55
21	33	-27	61	97	-1	A1	161	20	E1	225	56
22	34	-26	62	98	-1	A2	162	21	E2	226	57
23	35	-25	63	99	-1	A3	163	21	E3	227	59
24	36	-25	64	100	0	A4	164	21	E4	228	60
25	37	-24	65	101	0	A5	165	22	E5	229	61
26	38	-24	66	102	0	A6	166	22	E6	230	62
27	39	-23	67	103	1	A7	167	22	E7	231	63
28	40	-23	68	104	1	A8	168	23	E8	232	65
29	41	-22	69	105	1	A9	169	23	E9	233	66
2A	42	-22	6A	106	2	AA	170	24	EA	234	68
2B	43	-21	6B	107	2	AB	171	24	EB	235	69
2C	44	-21	6C	108	2	AC	172	24	EC	236	71
2D	45	-20	6D	109	3	AD	173	25	ED	237	73
2E	46	-20	6E	110	3	AE	174	25	EE	238	75
2F	47	-19	6F	111	3	AF	175	26	EF	239	77
30	48	-19	70	112	4	B0	176	26	F0	240	79
31	49	-19	71	113	4	B1	177	27	F1	241	81
32	50	-18	72	114	4	B2	178	27	F2	242	82
33	51	-18	73	115	5	B3	179	27	F3	243	86
34	52	-17	74	116	5	B4	180	28	F4	244	90
35	53	-17	75	117	5	B5	181	28	F5	245	93
36	54	-16	76	118	5	B6	182	29	F6	246	97
37	55	-16	77	119	6	B7	183	29	F7	247	100
38	56	-16	78	120	6	B8	184	30	F8	248	100
39	57	-15	79	121	6	B9	185	30	F9	249	100
3A	58	-15	7A	122	7	BA	186	31	FA	250	100
3B	59	-14	7B	123	7	BB	187	31	FB	251	100
3C	60	-14	7C	124	7	BC	188	32	FC	252	100
3D	61	-14	7D	125	8	BD	189	32	FD	253	100
3E	62	-13	7E	126	8	BE	190	33	FE	254	100
3F	63	-13	7F	127	8	BF	191	33	FF	255	100

Appendix 2: Variable table

Variable table

#	Name	Type	Description	
29	Fan speed	fanspeed	Current fan speed. Legal values: HEX 1 = speed 1 HEX 3 = speed 2 HEX 7 = speed 3 HEX F = speed 4	HEX 1F = speed 5 HEX 3F = speed 6 HEX 7F = speed 7 HEX FF = speed 8
2A	Current relative humidity	humidity	higher measured relative humidity from 2F and 30. Translating Formula: $(x-51)/2,04$ (Data is in hex).	
2B	Current level of CO2 high byte	dec		
2C	Current level of CO2 low byte	dec	Last measured amount of CO2. 2B=upper byte and 2C=lower byte. You can use 16-bit conversion tool to translate this. Just write both hex numbers to HEX box, high byte first.	
2F	%RH from sensor 1	humidity	formula: $(x-51)/2,04$	
30	%RH from sensor 1	humidity	formula: $(x-51)/2,04$	
32	Outside temp	temp	Measured temperature from outside air duct.	
33	Exhaust temp	temp	Measured temperature from exhaust air duct.	
34	Inside temp	temp	Measured temperature from inside air duct.	
35	Incoming temp	temp	Measured temperature from incoming air duct.	
A3	SELECT	state flag	bit 0=Power state 1=CO2 adjust state 2=%RH adjust state 3=Heating state 4=Filterguard indicator 5=Heating indicator 6=Fault indicator 7=service reminder	
A4	Heating setpt.	temp	Sets the temperature of the incoming air.	
A5	MAX fan speed	fanspeed	Sets the maximum fan speed. Use same values as in variable 29.	
A6	Service reminder	dec	Sets the interval of service reminder in months.	
A7	Preheating setpt	temp	Preheating is turned on when exhaust air temperature drops under this setpoint.	
A8	Input fan stop	temp	Input fan stops when exhaust air temperature drops under this setpoint.	
A9	MIN fan speed	fanspeed	Sets the minimum fan speed. Use same values as in variable 29.	
AA	PROGRAM	state flag dec	PROGRAM has two parts: lower nibble (bits 0-3) set the adjustment interval of CO2 and %RH in minutes. bit 4 = automatic RH basic level seeker state bit 5 = Boost switch mode(1=boost sw. 0=fireplace sw.) bit 6 = radiator type: 0 = electric 1 = water bit 7=cascade adjust 0=off 1=on	
AE	Basic humidity lev.	humidity	Displays apartment's normal humidity level when there aren't any humidity sources.	
AF	HRC bypass	temp	Heat recovery cell bypass setpoint.	
B0	DC fan input adj.	dec	DC-type input fan adjustment %	
B1	DC fan output adj.	dec	DC-type output fan adjustment %	
B2	Cell defrosting	temp	Defrosting routine starts when exhaust air drops below this setpoint.	
B3	CO2 setpt upper	dec	CO2 adjustment's setpoint upper byte.	
B4	CO2 setpt lower	dec	CO2 adjustment's setpoint lower byte. See variable 2C.	
B5	PROGRAM2	state flag	bit 0 = Function of max speed limit, 0=with adjustments 1 = always	