TU Delft

Electrical Engineering, Mathematics and Computer Science

DC systems, Energy conversion & Storage

eBike Charging Station - Communication Software

Author:
Pavel Purgat

December 15, 2015

Contents

The objective of this document is twofold. To provide introduction into the eBike Charging station communication software and to describe the hardware used. The document is divided into sections. Starting with general descriptions and driving ideas behind the creation of the software. Then it continues with short description of the hardware. The purpose is to give all essential technical data on one place for the ease of reference. The part describing the software itself follows. The document also contains the code documentation in the Appendix.

1 Introduction

The goal is to create a stand-alone solar powered eBike charging station which is power neutral. A prototype of the eBike charging station is to be build and demonstrated on the university campus. By placing the eBike charging station directly on the campus, which allows direct access and use, greatly facilitates the ongoing research and experience with new green technologies. Only by really building and using, one can demonstrate the effective use of green energy in a sustainable environment. The twofold use of the solar powered eBike charging station makes it an excellent milestone in the road towards a sustainable world.

- By using the solar powered eBike charging station on the university campus, experience in this new application field can be gathered and used for future ideas to move towards a more sustainable green environment. The prototype will serve as a demonstrator for sustainable mobility within cities and will promote the use of green technologies for short-range mobility.

- By including measurement devices and logging all measurements and storing them, makes the eBike charging station an ideal playground experiments and theoretical verifications

1.1 Extra Project role

The goal of the Extra Project was to design a communication system for the eBike charging station. The eBike charging station consists of several electrical parts. For correct operation of the station it is vital that these parts can communicate together and cooperate. Since the use of station is twofold, it is also necessary to collect data about the station itself, but also about e.g. weather.

1.1.1 System Overview

In the following figure Fig. 1.1 is the an overview of the system. The schematic clearly shows from which parts of the system are data being processed and where it is being send. The hearthstone of the system is Single on Board Computer (SBC). The computer was chosen to be Odroid C1+, which is a Linux based computer.

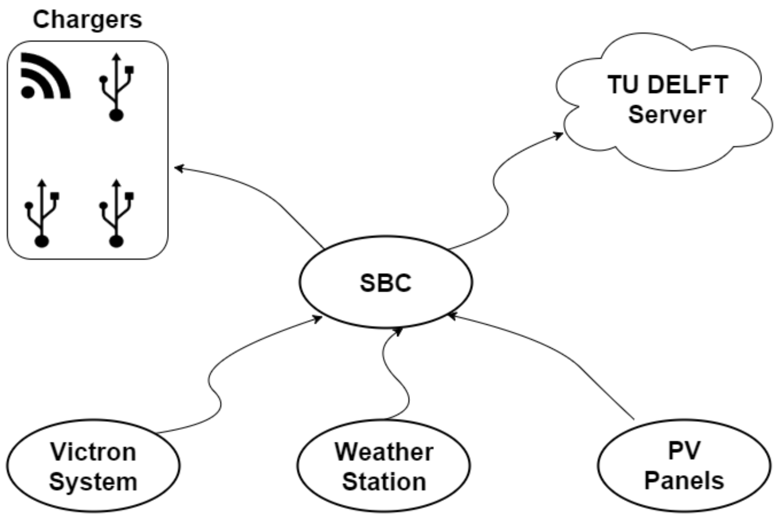

Figure 1.1: System overview.

What is not explicitly shown in the Fig. 1.1 is the fact that all parts use different communication protocols. However, the software running on the Odroid C+1 needs to understand all of the system parts. Therefore the language C was chosen as most powerful and robust, to be able to decompose and compose any type of message.

2 Hardware Overview

The eBike Charging station is equipped with various types of hardware. In this section very brief description of the equipment will be presented. The description will be given for all parts which are participating in the communication. The only part of equipment omitted is the router.

2.1 Lightweight Linux Controller

The ODROID-C1+ is very powerful low-cost single board computer, as well as an extremely versatile device. The heart of the device is a quad-core Amlogic processor. Further it contains advanced Mali GPU and Gigabit Ethernet. Thanks to the computational power available it can function as a prototyping device for hardware tinkering, a controller for home automation or as a workstation for software development etc..
The operating system that is running on our ODROID-C1+ is Ubuntu. Important advantage of Ubuntu is myriad of free open-source software packages. The ODROID-C1+ is an ARM device which is the most widely used architecture for mobile devices and embedded 32-bit computing. The ARM processor' small size, reduced complexity and low power consumption makes it very suitable for the eBike charging station.

2.1.1 Technical Specifications

Specifications	
Processor	Amlogic S805 SoC ARM® Cortex®-A5 (ARMv7) 1.5GHz Quad Core ARMv7 architecture @28nm wafer
Memory	1Gbyte DDR3 RAM 792Mhz
3D Accelerator	ARM ®Mali™-450 MP2 OpenGL ES 2.0 / 1.1
Flash Storage	MicroSD Card Slot : 8 GB MicroSD UHS-1
USB2.0 Host	High speed standard A type connector x 4 ports
USB2.0 Device/OTG	High Speed USB standard A type connector x 1 port
Ethernet/LAN	10/100/1000Mbps Ethernet with RJ-45 Jack (Auto-MDIX support)
Video/Audio Output	HDMI
Real Time Clock	On-board RTC function with a backup battery connector
IO Expansion	40pin port (GPIO/UART/SPI/I2C/ADC)
System Software	Ubuntu 14.04 + OpenGL ES on Kernel 3.10 LTS

2.2 Victron system

The Victron system consists of inverter *Phoenix Inverter 3000VA - 5000VA*, solar charge controller *EasySolar*, control panel *Color Control GX*, battery monitor *BMV-700 series*, battery protection *Battery Protect*. The system naturally consists of other protective equipment, however it is not in the scope of this work to provide extensive description of the power electronic devices and its use inside the eBike Charging Station.
For the communication system the most important part is the controller *Color Control GX*. The Color Control GX provides intuitive control and monitoring for all products connected to it. In the next figure Fig. 2.1 is depicted the Victron system overview. As is clear from the figure, the only connection from computer is to the common bus, MB/TCP. This connection is done via above mentioned controller.
The only important setting on the Victron System side is to set the IP address correctly. The rest is done inside the SBC.

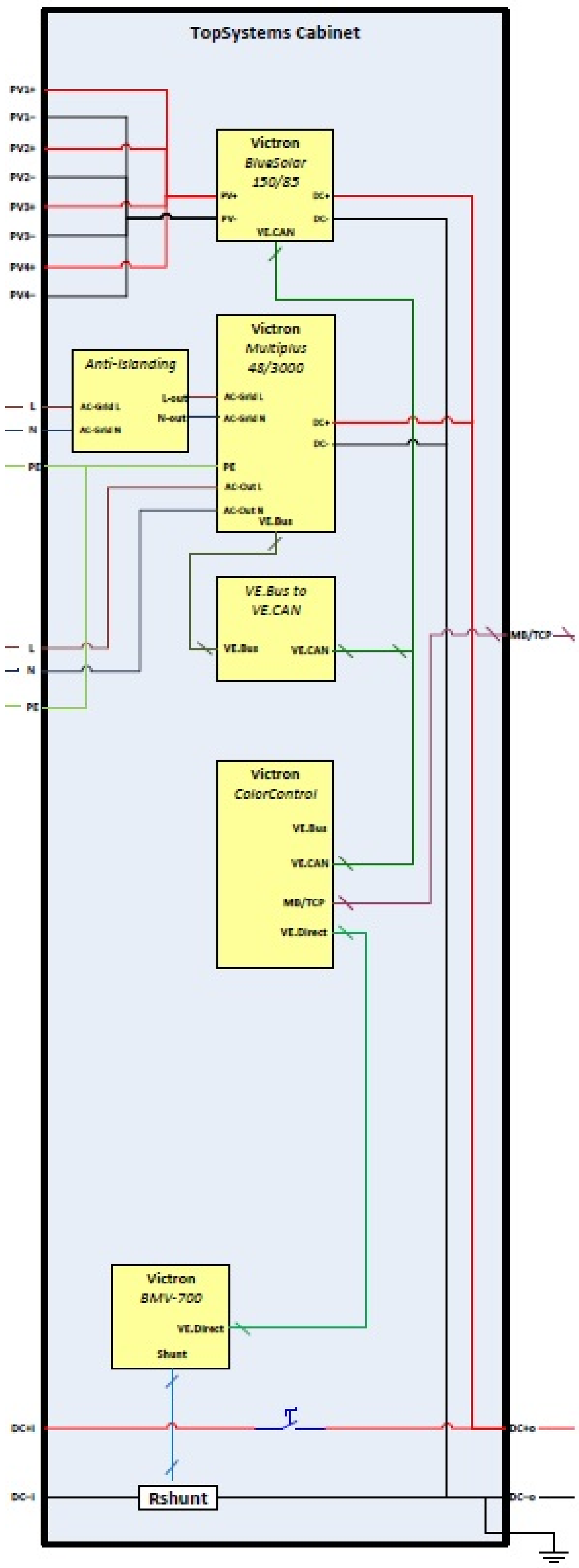

Figure 2.1: Victron system overview.

2.3 Weather Station

WS600-UMB Smart Weather Sensor for measuring of air temperature, relative humidity, precipitation intensity, precipitation type, precipitation quantity, air pressure, wind direction and wind speed. The weather station is connected to the SBC via Modbus/RTU. Before first connecting to the SBC, the weather station needs to be connected to the personal computer using the producer specified ANSCII communication protocol. Then after successful connection, and installation of proper software (again producer' software), the weather station can be set to use Modbus/RTU context. The initial connection posses the only major drawback of the weather station, since there is support only for Windows machines.

2.3.1 Technical Specifications

Specifications	
Dimensions	diameter approx. 150mm, height approx. 343mm
Weight	Approx. 1.5kg
Interface	RS485, 2-wire, half-duplex
Power supply	4...32 VDC
Operating temperature	-50...60 ℃
Operating rel. humidity	0...100 % RH
Heating	40VA at 24VDC
Cable length	10m
Protection level housing	IP66
Mast mounting suitable for	mast diameter 60 - 76mm

2.4 Module Adam-4015T

Adam-4015T is a data acquisition module which provides ideal measurement solution for harsh environments and demanding applications. With wide operating temperature ranges and multiple mounting methods, Adam 4015T, which is RS-485-based, can be implemented in diverse applications. The main advantages of Adam 4015T are built-in watchdog timer and individual wire burned-out detection.

2.4.1 Technical Specifications

Specifications	
Display	LED Indicators
Power Supply	10 30 VDC
Channels	6
Power supply	4...32 VDC
Operating temperature	-10 70 ℃
Operating rel. humidity	5 95 % RH
Temperature (Storage)	-25 85 ℃
Watchdog Timer	System (1.6 second) & Communication
Interface	RS-485
Communication Protocol	ASCII command & Modbus/RTU[1]

2.5 RS485/USB Coverter

The USB-RS485 cable is a USB to RS485 levels serial UART converter cable incorporating FTDI' FT232RQ USB to serial UART interface IC device which handles all the USB signalling and protocols. Each USB-RS485 cable contains a small internal electronic circuit board, utilising the FT232R, which is encapsulated into the USB connector end of the cable. The integrated electronics also include the RS485 transceiver plus Tx and Rx LEDs which give a visual indication of traffic on the cable (if transparent USB connector specified). The other end of the cable is bare, tinned wire ended connections by default.

3 Software Description

In this chapter all part of the software will be described. Also a general description of the communication protocols will be given. However, the aim is definitely not treat the protocols in great detail. The chapter concludes with some remarks, and strong and weak points of the code.
The software is written in C. C is a general-purpose, imperative computer programming language, supporting structured programming, lexical variable scope and recursion, while a static type system prevents many unintended operations. C provides constructs that map efficiently to typical machine instructions, and therefore it has found lasting use in applications that had formerly been coded in assembly language, including operating systems, as well as various application software for computers ranging from supercomputers to embedded systems.
The software it self contains in total 4 sub-programs. Each sub-program deals with communication with different part of the system. The main program is run in threads, which were used to ensure speed of the operations and also to ensure good memory management. The only disadvantage of the C for this type of software are the missing exceptions. Since 'communication timeout errors' are quite frequent *goto* statements had to be used.

3.1 Communication Protocols

A short overview of the communication protocols used. The most common TCP/IP is being used for communication with the server. The Modbus/TCP which is very common in embedded system is used to communicate with the Victron system. The RTU/TCP protocol is used to communicate with the Weather Station and the ADAM-4015T utility. The protocols are described in this order.

3.1.1 TCP/IP

TCP/IP is the Internet protocol suite, which is the computer networking model and set of communications protocols used on the Internet and similar computer networks. The Transmission Control Protocol (TCP) and the Internet Protocol (IP) were the first networking protocols defined in this standard. TCP/IP provides end-to-end connectivity specifying how data should be packetized, addressed, transmitted, routed and received at the destination. This functionality is organized into four abstraction layers which are used to sort all related protocols according to the scope of networking involved. From lowest to highest, the layers are the link layer, containing communication methods for data that remains within a single network segment (link); the internet layer, connecting independent networks, thus establishing internetworking; the transport layer handling host-to-host communication; and the application layer, which provides process-to-process data exchange for application.

3.1.2 Modbus/TCP

Modbus is a serial communications protocol originally published by Modicon (now Schneider Electric) in 1979 for use with its programmable logic controllers (PLCs). Simple and robust, it has since become a de facto standard communication protocol, and it is now a commonly available means of connecting industrial electronic devices.
Modbus enables communication among many devices connected to the same network, for example a system that measures temperature and humidity and communicates the results to a computer. Modbus is often used to connect a supervisory computer with a remote terminal unit (RTU) in supervisory control and data acquisition (SCADA) systems. Many of the data types are named from its use in driving relays: a single-bit physical output is called a coil, and a single-bit physical input is called a discrete input or a contact.
The development and update of Modbus protocols has been managed by the Modbus Organization since April 2004, when Schneider Electric transferred rights to that organization. The Modbus Organization is an association of users and suppliers of Modbus compliant devices that seeks to drive the adoption and evolution of Modbus.

3.1.3 RTU/TCP

A remote terminal unit (RTU) is a microprocessor-controlled electronic device that interfaces objects in the physical world to a distributed control system or SCADA (supervisory control and data acquisition) system by transmitting telemetry data to a master system, and by using messages from the master supervisory system to control connected objects. Another term that may be used for RTU is remote telecontrol unit.

3.2 eBike Charging Station - Communication software

The program overview is shown in the Fig 3.1. The idea is, that there are in general requests which need to be evaluated very often and very fast. But there are also those which are more time consuming, but do not need to be evaluated very often. Therefore the program runs several threads, which are not consecutive but the time is given by the main thread. If the thread takes to long to finish, it is most likely caused by some error. Therefore it is forced to exit, and then the program continues.
In following subsections a few words to each part of the program will be given. For thorough description the reader is invited to read the documentation attached to this document.

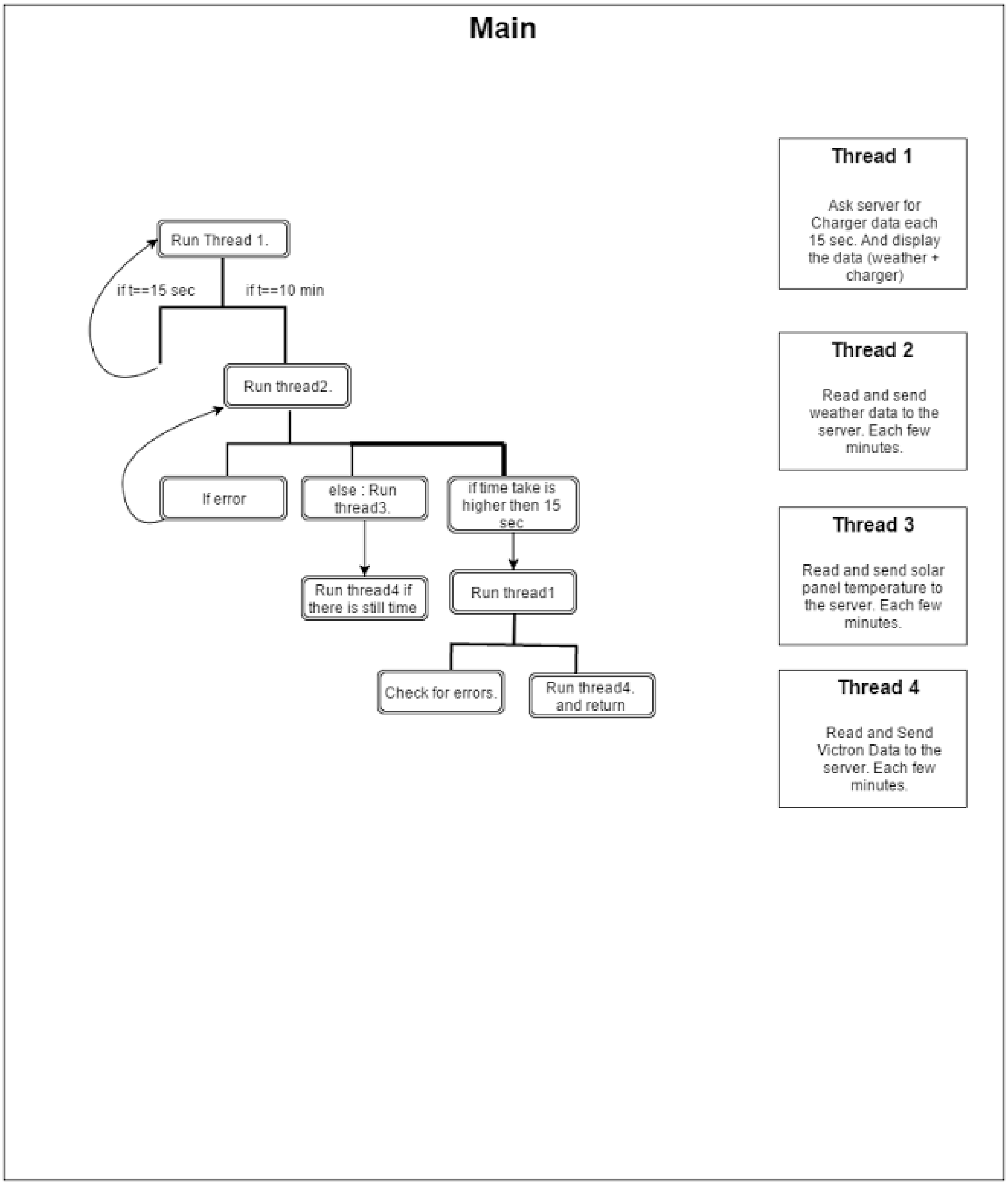
Main
Run Thread 1.
if t==15 sec
if t==10 min
Run thread2.
If error
else : Run thread3.
if time take is higher then 15 sec
Run thread4 if there is still time
Run thread1
Check for errors.
Run thread4. and return
Thread 1
Ask server for Charger data each 15 sec. And display the data (weather + charger)
Thread 2
Read and send weather data to the server. Each few minutes.
Thread 3
Read and send solar panel temperature to the server. Each few minutes.
Thread 4
Read and Send Victron Data to the server. Each few minutes.

Figure 3.1: Main Program Overview

3.2.1 BicycleShed.c

The is the main program for the eBike Charging station. The program handless the comunication between different parts of the charging station. Namely: weather station, chargers, web server, converters and solar panels temperature measurement. The

BicycleShed.c contains threads which operate on different parts of the system.
The overview of this part is given in Fig. 3.1.

3.2.2 Victron.c

The Victron system is connected via one Modbus link to which all of its parts are connected. The messages are constructed via *libmodbus* library.

3.2.3 Weather.c

In this part the communication with the utility ADAM-4015T and Weather station is handled. Both of these utilities use RTU contex, therefore they are merged under one .c file. The only important difference is that the Weather station is using *Holdin Registers* while the ADAM-4015T uses *Input Registers*. User wise this makes no difference, however the message construction is different. The difference can be easily found in the documentation of the used library.

3.2.4 Display.c

This .c file contains whole .html page. In the first part of the execution needed data are read from the weather station and the Server (states of the charger). And then the .html page is printed. This can be done thanks to the fact that the software is running on Linux. On Windows this would not be possible, since in the same time the .html file is also open by a web browser.
This part is maybe the most optimal way of displaying data. However, it has distinct advantages. Since the page is local no internet connection is required. Even in case of failure of the program, he html page should still show the last known values. Therefore the user does not have to look on annoying error messages.

3.2.5 ServerCom.c

This part deals with the communication with the server. There are two operations needed: send to server and read from the server. Both are performed using the *libcurl* library. All communication with the server is performed via http links. This is maybe not the best way. However, it is quite secure and also easy to implement. The amount of data is not so large to use FTP or other means of communication.

3.3 Web Server

On the web server MySQL databases are prepared. These database are divided to save different types of data from the eBike Charging station. In order to be able to save and read from these databases several php scripts were prepared. Which are part of the attached documentation.