Master's Thesis

Miniature Internet of Things Gateway with Power over Ethernet

Adam Johansson

Department of Electrical and Information Technology,
Faculty of Engineering, LTH, Lund University, November 2014.

Miniature Internet of Things Gateway with Power over Ethernet

Adam Johansson
`johanssonadam@outlook.com`

Department of Electrical and Information Technology
Lund University

Advisers:
Mats Andersson
Markus Wejrot

Examiner:
Joachim Rodrigues

November 5, 2014

Printed in Sweden
E-huset, Lund, 2014

Abstract

The Internet of Things is on the rise, according to many, and the future prospect of having 30 billion or more devices connecting wirelessly to the internet by 2020 is creating new business areas. This master's thesis investigates the possibility of implementing a very small scale Internet of Things gateway through which Bluetooth Low Energy devices can access or be accessed from the internet. The proposed gateway is then implemented, from concept to a working prototype and a demonstration software is built to run on the hardware prototype. The result is a web page hosted on the gateway through which users can control and read data from a connected Bluetooth Low Energy device. The gateway is based on a multi radio module developed at u-blox Malmö.

Acknowledgements

I would like to start by thanking u-blox Malmö, formerly connectBlue, for providing this master's thesis. I would like to thank all employees at their Malmö office for all their help and support especially my advisers Mats and Markus. Everyone has been very friendly and made me feel like a part of the team.

I would also like to thank Joachim Rodrigues at LTH for acting as my examiner, he has been very patient.

Lastly, I would like to acknowledge all the work put into the FreeRTOS, lwIP and BTstack projects from all of their respective contributors. I would never have had a demonstration software without their hard work. I also thank Drew Greenwell for freely distributing his Metro JS JavaScript plugin that i used to make a much more interesting demonstration web page.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Project Description

Internet of Things (IoT) has been around since the late nineties[1], but until now has been bulky to implement. Emerging integrated circuits allow it to be used in increasingly smaller formats opening up new areas of implementation. This masters thesis investigates the possibility of implementing a small footprint IoT gateway with WLAN and Bluetooth Low Energy support. This device will be powered using Power over Ethernet and the prototype printed circuit board will be designed to fit inside a pre-fabricated casing. It will use an ARM Cortex-M4 based chip that is to be programmed to retrieve Bluetooth data and send it over WLAN or Ethernet to a database or other service.

This master's thesis was issued by u-blox Malmö, previously connectBlue, in collaboration with the institute of Electrical and Information Technology at Lunds Tekniska Högskola, Lund University.

1.1.1 Project Scope

- Analyse existing technologies that can be used in the project and gain knowledge of how they work.
- Set-up specifications for the hardware with regards to physical and electrical limitations, market demands, cost etc.
- Design a hardware prototype for the project, including all information needed for manufacturing.
- Set-up specifications for the software with regards to processing power, available hardware platforms, security etc.
- Implement a real-time kernel with limited Bluetooth and Ethernet support and a demonstration software capable of running on the hardware prototype.
- Verify and possibly benchmark the hardware prototype.

1.2 Report Outline

Chapter 1 - Introduction
: The project and this report is introduced to the reader.

Chapter 2 - Background
: Different theories and technologies required to understand the report are explained.

Chapter 3 - Design of a Prototype Printed Circuit Board
: The requirements of the design are explained along with a system level design approach to the solution.

Chapter 4 - Detailed Design
: The methods used to implement the solution and how it works, is explained to the reader.

Chapter 5 - Evaluation
: Testing of the implementation and the results. Discussion of continuation on the work done and how the resulting gateway compares to other implementations on the market.

Chapter 2

Background

This chapter gives detailed information on some of the technologies needed to implement the gateway, as well as some background theory on Internet of Things.

2.1 Internet of Things

Internet of Things (IoT) is a concept that has been trending for the past couple of years, and is simply put the idea of connecting everything and anything to the internet and/or each other. It is also known, or has been known, as Internet of Everything, Machine to Machine (M2M) and Device to Device (D2D). The term seems to continuously evolve, but the goal stays the same. When things start getting a common digital representation, collection of "big data" is possible and with it brand new applications.

Of course, this is not without problems. Some security analysts believe that introducing more IoT devices to the market would lead to a crisis[2], since a majority of embedded systems on the market are unpatchable and riddled with security holes. Developers of new technology must keep in mind that their systems must be updatable to get rid of security holes.

IoT devices could include pollution sensors used in big cities to give real time information on pollution levels to both citizens and city planners, moisture sensors planted in the ground to let farmers know how their crops are doing or your coffee maker and alarm clock that can be told by your on-line calendar that your morning meeting has been postponed by 30 minutes.

2.1.1 Available Technologies

While there are many ways to implement internet access it would be preferable if it could be made as low-cost, energy efficient, easy-to-use, secure, with as wide range as possible and with an already available ecosystem. Most of this can be achieved by moving the computational power required to maintain an IP stack etc. to a remote, more easily powered device and instead only implement the most basic requirements to communicate over radio. Available radio technologies include Bluetooth, Bluetooth Low Energy (BLE), WLAN, Zigbee, Near Field Communication, 3G/4G. Research done[3] shows that Bluetooth/BLE are strong contenders and suitable as target technology for this project.

2.1.2 In the Near Future

Today a little more than 10 billion devices, most of them computers and smart phones, are connected to the internet. The European Union, researchers and big IT companies predict that by 2020 somewhere around 30 to 50 billion devices will be connected to the internet, most of them wirelessly[4][3][5][6][7]. A majority of these will be IoT devices.

Current generation IoT devices are limited to applications within its own industry or sector. The next generation of connected devices will be able to go beyond these borders, opening up for completely new applications when data from the energy, IT, transport, education, healthcare and environment sectors can be combined.

2.2 Power over Ethernet

Power over Ethernet (PoE) is a technique where both data and power can be sent over a single Ethernet cable. There are many other interfaces that transfer both data and power in one cable; USB, Firewire, Thunderbolt etc. The latest version of USB 3.1[8] (april 2013) claims it can supply up to 100 W of power while maintaining a data rate of up to 10 Gbps. The electrical specifications[9] limit the length of a standard cable to about 3m. Firewire and Thunderbolt boast similar specifications. The latest Power over Ethernet specification (IEEE 802.3 at), PoE+, can supply 25.5 W of power over a 100 m standard (cat.5e) ethernet cable at 1 Gbps data rate[10](Table 33-18). The big advantage is the length of the cable and that it can utilize a cheap unshielded twisted copper pair cable such as a cat.5 or cat.5e Ethernet cable already present almost everywhere and run power along it when the need arises, while maintaining all present forms of communication.

Typical applications include IP phones, IP cameras, and wireless access points.

This section of the report, while in-depth, mainly focuses on implementation of the device that is being powered. Also gigabit Ethernet implementions is left out, but is very similar to 10/100 Mbit/s that is described below. The full specification for Power over Ethernet is available as a free download at the IEEE website[10] (section 2 chapter 33).

2.2.1 Standards

PoE can be implemented in different ways, currently two are standardized by the IEEE; 802.3af and 802.3at. The main difference is that IEEE 802.3at can provide more power, and is sometimes referred to as PoE+. Another technology called

Universal Power over Ethernet (UPoE) claims to be able to supply 51 W over a single cable[11].

2.2.2 Implementation

PoE requires both a Power Sourcing Equipment (PSE) typically a PoE compliant Ethernet switch, and a Powered Device (PD). In a standard ethernet cable (cat 5e) there are four twisted cable pairs connected with RJ-45 connectors. In 10BASE-T or 100BASE-TX only two pairs are used for data transfer. Power is applied to two pairs of the cable by the PSE by raising/lowering the common mode voltage of the cable pairs. The standard defines two modes for power transfer, A and B, where in mode A power is applied to the cables carrying data and in mode B power is applied to the two spare pairs, as is shown in Table 2.1. The PD extracts the common mode voltage to use for powering the device, leaving the data intact. A PSE may be implemented with capability to use both powering modes, but may only run in one mode at a time. A PD must be able to handle both modes.

Table 2.1: PoE pinout on an RJ-45 connector, note that mode B is the same for both MDI and MDI-X (crossover RX/TX) cabling.

Pin #	mode A (MDI)	mode A (MDI-X)	mode B (both)
1	V+ & data	V- & data	data
2	V+ & data	V- & data	data
3	V- & data	V+ & data	data
4	-	-	V+
5	-	-	V+
6	V- & data	V+ & data	data
7	-	-	V-
8	-	-	V-

2.2.3 Handshaking Protocol

To successfully establish a Power over Ethernet connection a handshaking protocol is used. This protocol ensures that a non-PoE device can be detected as such, and safely be connected. It also lets the Powered Device request the amount of power needed. This can be useful for the Power Sourcing Equipment when multiple PDs are connected and the available power has to be distributed among them. The handshaking is done at the physical layer, meaning success is based on the characteristics of the electrical signals rather than a correct sequence of data bits.

The protocol is split up into two stages; a detection stage and a power classification stage. A PSE may not apply operational power to a line until it has successfully detected a PD to prevent damaging legacy devices. A simplified state diagram of the handshaking process is shown in Figure 2.1 (a full state diagram is shown in the IEEE 802.3 Specification[10] section two, clause 33). V_{PD} is the voltage measured by the PD on the power lines. In the Idle state, the Powered Device waits for the voltage to rise above V_{det_min}, the limit at which the PD must enter the Detection state. In the Detection state, the PD will change the impedance of the power lines, described as changing the physical *signature* of the link by the PoE specification. Detection (and classification) is handled by the PSE by probing the power lines to measure impedance and current levels. Valid detection characteristics, or signatures, for a Powered Device are shown in Table 2.2.

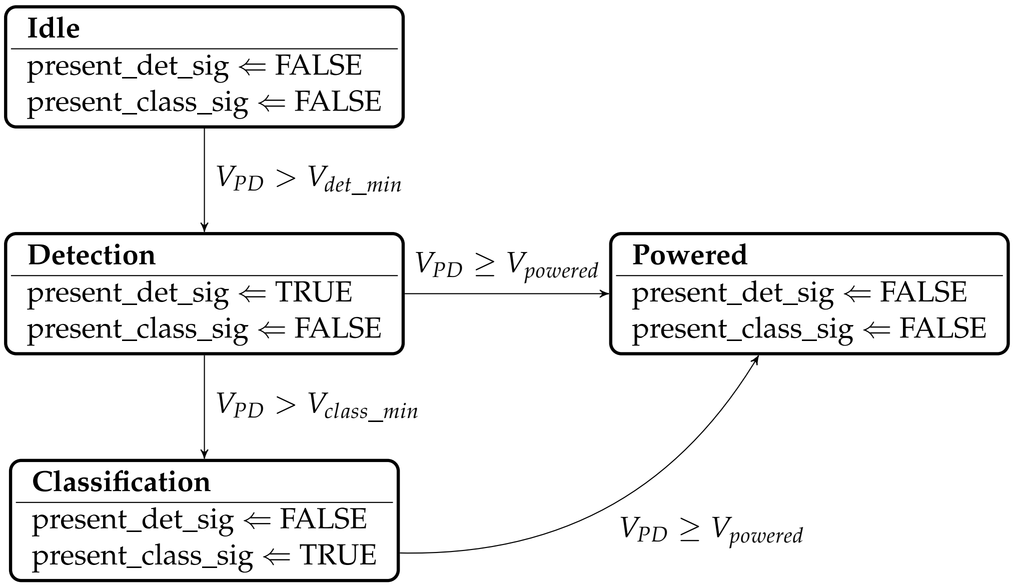


Figure 2.1: A simplified state diagram of the handshaking process in a PoE link set-up. V_{PD} is the voltage measured at the PD input.

Table 2.2: Valid detection signatures for a PoE Powered Device.

Parameter	Unit	min	max	Additional info
PSE test voltage (V_{det})	V	2.70	10.1	applied to the cable
PD resistance	kΩ	23.7	26.3	presented across PD input connector
PD capacitance	µF	0.050	0.120	presented across PD input connector

If a PSE successfully detects a PD it may choose to directly grant or deny power, though normally it will continue to a Classification stage in which it lets the PD ask for the amount of power it requires. To enter Classification state the supply voltage must be again be raised, this time to above V_{class_min}, or even further to above $V_{powered}$ to skip classification. In the 802.3af standard there are four different power classes, the newer PoE+ standard has an additional high power class (class 4). Classification is done by the PD presenting a specific current through it's input connectors, when the PSE applies classification voltage to the connection. The different classification currents are shown in Table 2.3. When classification is done, operational voltage is applied and the PD enters the Powered state.

Table 2.3: Valid PD classification currents. If a PD presents a non-valid current it may not get powered.

Parameter	Unit	min	max	Additional info
PSE test voltage (V_{class})	V	14.5	20.5	measured at PD input connector
class 0 current	mA	0	4	power limit: 13.0 W
class 1 current	mA	9	12	power limit: 3.84 W
class 2 current	mA	17	20	power limit: 6.49 W
class 3 current	mA	26	30	power limit: 13.0 W
class 4 current	mA	36	44	power limit: 25.5 W

The newer IEEE 802.3at (PoE+) standard requires a second classification stage in order to get more than 13.0 W of power, but the details are not of importance to this report since the project uses the legacy standard. The exact details of PoE implementation can be read in the IEEE 802.3 specification[10] section two, clause 33.

2.2.4 Maintaining the Link

Once a Powered Device is properly detected and classified the link is powered up. The Power Sourcing Equipment now applies a voltage between 37–57 V to the link, it is up to the manufacturer to choose. If a PD draws more power than advertised during classification the PSE may power down the link, so careful calculations must be made on the power consumption of a new PD device. There is also a lower limit of the minimum amount of current a PD must draw to keep the link alive. If the current drops below 10 mA for longer than 400 ms the PSE will power down the link. Valid PD Maintain Power Signature characteristics are shown in Table 2.4 below.

2.3 Bluetooth Low Energy

Bluetooth Low Energy (BLE), or Bluetooth Smart, was added to the Bluetooth standard in 2010 when version 4.0 was released. Like the name suggests it is targeted for applications that are battery powered and only need low data throughput. Like classic Bluetooth, BLE uses the 2.4 GHz industrial, scientific, and med-

Table 2.4: Valid PD Maintain Power Signature, both current and impedance is measured across the PD input.

Parameter	Unit	min	max
Input current	A	0.010	-
Input resistance	kΩ	-	26.3
Input capacitance	µF	0.050	-

ical (ISM) band for transmission meaning a single device can implement both versions along with WLAN etc. and only use a single antenna. The range for BLE varies between applications, but has been measured to have a range of up to 250 m[12].

The power consumption of a Bluetooth Low Energy device also greatly varies with implementation, but as the peak current consumption is under 15 mA and some implementations have reported average power consumptions of 1 μW[12] truly low power implementations are possible.

2.3.1 Bluetooth Stack

Bluetooth, like Ethernet etc. is defined as a layer protocol architecture. A Bluetooth stack is a software implementation of all or parts of the protocol suite. Figure 2.2 shows a Bluetooth protocol architecture that is of interest in this report.

- **Physical Radio & Baseband** - These are part of the controller and outside the scope of this report, thus they will not be explained in detail. In short they take care of signal processing and transmission.
- **Host Controller Interface (HCI)** - Allows for standardised communication between the host stack and the controller. The controller is often a separate radio module IC, while the host is the targeted microprocessor, smart phone, tablet etc. Communication between them is done over a serial interface. The HCI standard allows any controller IC and stack software to be interfaced with minimal adaptation. The standard allows the host to send any number of pre-defined HCI commands to access different areas and settings in the controller. The controller can send HCI events either as a response to a given command or to let the host know something needs its attention. The HCI layer is also responsible for data transmission from higher layers.
- **Logical Link Control and Adaptation Protocol (L2CAP)** - L2CAP serves as interface between the HCI layer and upper layers. It segments and reassembles data packets. Some stack implementations include a BLE version of L2CAP that only supports the ATT protocol (and profiles that use it).

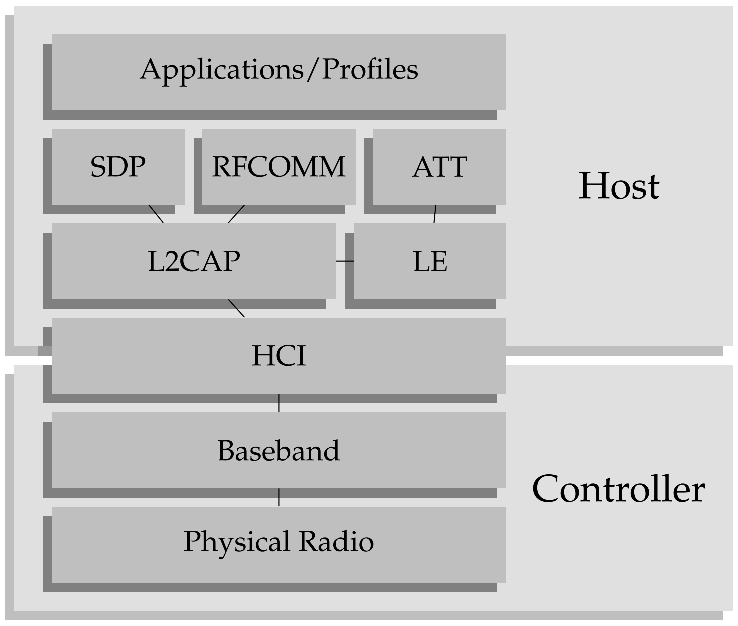


Figure 2.2: An architectural view of a Bluetooth stack implementation.

- **Service Discovery Protocol (SDP)** - Used to discover what profiles other Bluetooth devices support and what parameters to use to connect to them. Not used in BLE applications.

- **Radio Frequency Communication (RFCOMM)** - Used to create virtual serial ports between Bluetooth devices. Is easily interfaced to existing serial ports on the system. Not used in BLE applications.

- **Attribute Protocol (ATT)** - Serves as a lighter version of the Service Discovery Protocol for low energy applications. It is also responsible for sending and receiving data in LE devices.

These are the most common protocols used in a host implementation. There are other key protocols used in the controller stack, but they are not discussed in this report that focuses on the host side. Most other host protocols are sitting on top of L2CAP. On top of these protocols there is also defined a number of profiles that devices must comply to in order to be used correctly. These profiles can be seen as templates used to build applications that want to implement a commonly sought after feature. A Bluetooth headset for instance, might include the Headset Profile and Generic Audio/Video Disrubution Profile. The profiles are discoverable by SDP or ATT.

2.3.2 Generic Access and Attribute Profiles

The Generic Access Profile (GAP) is a low energy profile sitting on top of ATT that is used for connecting two low energy devices. It defines two types of devices: a central device, like a smart phone etc. and a peripheral device, like a low energy sensor. Peripheral devices advertise their presence by sending advertisement data packets in programmable intervals. Central devices scan for these packets, and uses the data inside to connect to the peripheral. The advertisement packets can also be used as a one way broadcast of small amounts of data to any central device in range.

Once a connection has been established the Generic Attribute Profile (GATT) is used to send data between devices. Like the name suggests GATT also sits on top of the Attribute Protocol. The profile defines a structure shown in Figure 2.3 that all BLE devices have to comply to.

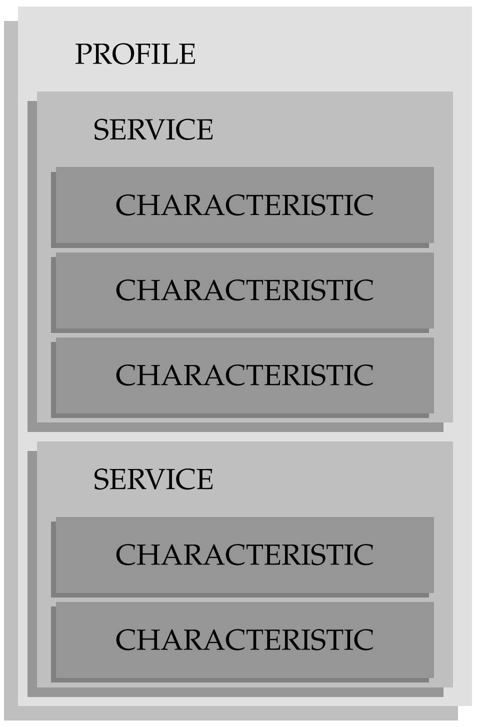


Figure 2.3: The structure of Profiles, Services and Characteristics used in GATT

The peripheral device runs a GATT server that the central device can query using a GATT client. The server can have various services running that expose different characteristics to the client. A characteristic can be a variable holding the current temperature in a temperature sensor, or a setting to send a warning if the temperature reaches above a certain point etc. Characteristics are grouped together in services and multiple services can be grouped together in a profile. The Heart Rate Profile for instance, includes the Heart Rate Service and the Device Information Service. A list of profiles and services that currently are adopted into the Bluetooth specification can be found on the Bluetooth homepage[13]. The GATT structure is meant to be easily implemented by resource constrained systems, where a characteristic might simply point to a hardware register.

2.4 Reduced Media Independent Interface

The reduced media independent interface (RMII), standardized by IEEE, is used to connect a Media Access Controller (MAC) hardware to a physical transceiver (PHY) and as it is media independent, twisted pair copper, fiber optic etc. can be used interchangeably. Table 2.5 shows the signals used in RMII. As two data bits are clocked with a 50 MHz clock, 100 Mbit/s is the max achievable transfer speed.

Table 2.5: RMII signals. The management signals can be connected to multiple PHYs.

Signal name	Direction	Description
TXD0	MAC to PHY	Transmit data bit 0
TXD1	MAC to PHY	Transmit data bit 1
TX_EN	MAC to PHY	Transmit Enable, high when data is being clocked onto TXD0 and TXD1
RXD0	PHY to MAC	Receive data bit 0
RXD1	PHY to MAC	Receive data bit 1
CRS_DV	PHY to MAC	Carrier Sense and Receive Data Valid signals multiplexed together.
RX_ER	PHY to MAC	Receive Error
REF_CLK	MAC to PHY	50 MHz reference clock
MDIO	bidirectional	Management data I/O line, push-pull
MDC	MAC to PHY	Management data clock line

2.5 ODIN-W260

ODIN-W260 is a multiradio module that was chosen as the target module to build this thesis around because of its small form factor along with both dual-band WLAN and dual-mode Bluetooth capabilities. It also comes with a reduced media independent interface (RMII) for easy connection to an Ethernet physical layer chip. The on-board microcontroller (MCU) is an ARM Cortex-M4 based

STM32F439 from STMicroelectronics. It has a 2 MB flash memory, 256 kB RAM and operating frequency up to 180 MHz.

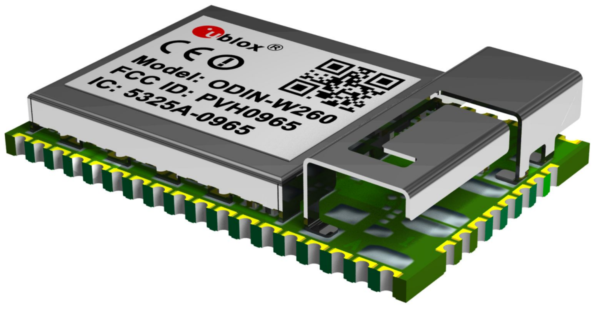


Figure 2.4: The ODIN-W260 multiradio module with a shield box and on-board antenna.

Chapter 3

Design of a Prototype Printed Circuit Board

This chapter describes in a general way the design process of this project, from an early concept to finished product. The Figure 3.1 shows an early concept design of how the implementation could look fitted inside the casing.

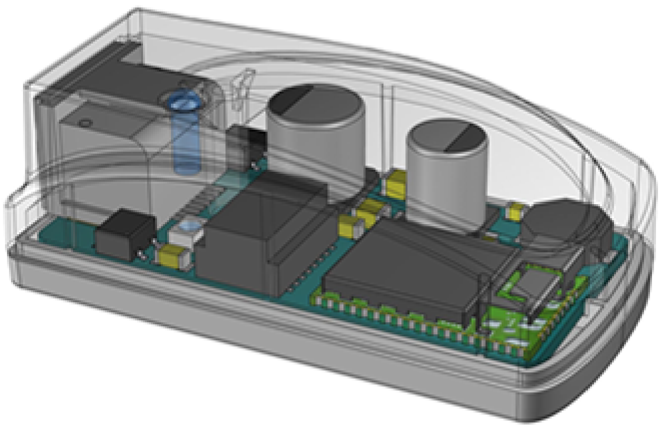


Figure 3.1: An early concept image of how a miniature gateway could be implemented.

To give the project some structure and a fixed starting point it was decided to base the hardware design on this concept which includes:

- A casing: this limits the size of the PCB to be 53.6 mm long and 26.2 mm wide. Components mounted on the bottom side of the PCB can have a maximum height of 1.6 mm.
- ODIN-W260 module: includes a host MCU, a dual-mode Bluetooth and dual-band WLAN radio chip, 1.8 V I/O voltage reference and RMII interface.

- RJ-45 modular jack: for connecting a cable capable of transmitting Ethernet data and supplying power to the design.
- RGB LED: for indication purposes.

The case also fixes the placement of the RJ-45 connector and RGB LED on the PCB, limiting the component placement and signal routing options during PCB design. A screw is needed to hold the case together which requires a fixed mounting hole in the circuit board which also limits routing options.

3.1 Requirements

The following requirements were included in the project proposal and were considered at the start of the prototype development:

- The solution should fit inside the casing.
- It will be powered using Power over Ethernet.
- It will act as a carrier board for the ODIN-W260 module.
- It will use BLE to gather data and move it using WLAN or Ethernet to a service or database on the intra/internet.

In addition to these initial requirements, further specifications were added to the project during prototype planning:

- The MCU will be programmed using JTAG/SWD and will include a UART interface for debugging purposes.
- The design will include a USB interface to connect the UART and JTAG signals to a PC. This can also act as an alternative power source.
- The design will include low profile push-buttons and LED indicators for debug purposes.
- The software will use GATT to move data to and from Bluetooth devices and a RESTful API to communicate with the internet.
- The capabilities of the gateway should be demonstrated in some way.

3.2 System Level Design

After considering the different requirements it was decided to completely drop WLAN support. As the gateway would be connected to the Internet through a cable in order to get power anyway, there would not be much to be gained for demonstration purposes. A graphical overview of how the gateway solution is thought to work is shown in Figure 3.2 and is explained below.

1. An HTTP GET request is sent over the Internet to access a resource on a Bluetooth sensor. The request can include data to be written to the sensor.

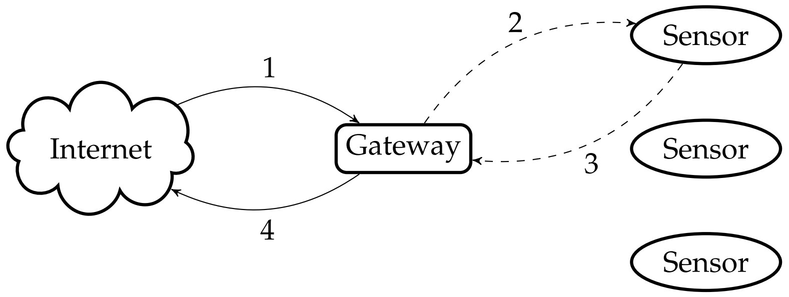


Figure 3.2: Graphical overview of the thesis.

2. If the request is valid, the gateway initiates a GATT characteristic read or write. Which sensor and service to be accessed can be included in the HTTP request or be decided by the gateway. Security measures can/may be taken.
3. Depending on the request, the sensor may send a response to be decoded by the gateway.
4. The gateway answers the request with an appropriate HTTP response e.g 200 OK or 404 not found.

The above is a typical use case, there can be many more uses for the proposed hardware that are discussed later. The originator of the HTTP request can be a web browser or a smart phone application etc. For this thesis the important part is that a BLE sensor can be accessed from a remote network.

3.2.1 Hardware

Since the prototype was being developed as a carrier board to mount the existing multi-radio module on, its major purposes was to supply power and act as a physical layer for Ethernet communications. In Figure 3.3, a block diagram shows an architectural view of the system. Notice that PoE supplies 37–57 V to connecting devices, therefore a power converter is needed to supply the 3.3 V required by the radio module and the rest of the system. There also needs to be some logic that can set-up and maintain the PoE link.

More logic is needed to implement an Ethernet transceiver capable of interfacing with the Media Access Control layer on the host microcontroller. A Reduced Media Independent Interface (RMII) is used to connect between the PHY and MAC layers. The Ethernet PHY layer is responsible for transmitting and receiving the electrical data signals between connecting devices. The MAC layer is responsible for addressing different devices in the network.

The previous two hardware blocks is essentially all hardware that is needed to fulfill the initial project requirements, however it is often preferable to include ways to test both hardware and software of prototypes. For this purpose it was

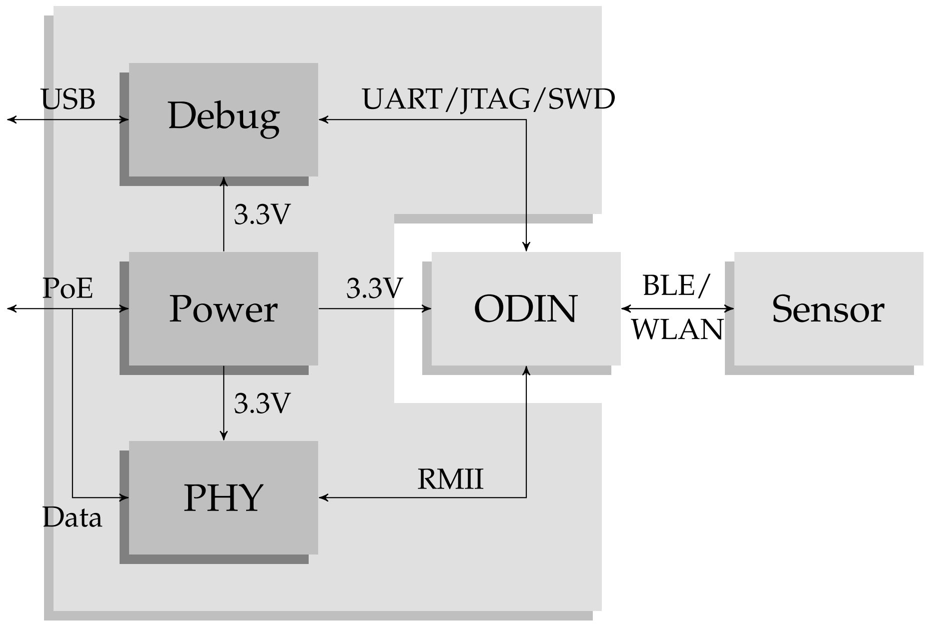


Figure 3.3: Block diagram of the hardware, arrows indicate power and data transitions. The shaded area indicates the scope of the project.

decided to include ways to access the JTAG/SWD and UART interface of the MCU. On most designs where these interfaces are present, they are connected with pin headers. In this design, to save space, the JTAG and UART signals were instead sent over a single USB interface. The SWD interface using less signals, was connected to pin headers. A few LEDs and push-buttons were included to help during software development and testing.

3.2.2 Software

The software of the system can be seen as split into three parts, as shown in Figure 3.4. Firstly there is a real-time operating system (RTOS) running on the STM32F439 microcontroller. The operating system is responsible for initiating the clocks, interrupts, mutexes, GPIO pins, DMAs and serial ports that are going to be used by the applications. It is also responsible for scheduling tasks, providing mailbox support for inter-thread communication, and provide debugging capabilities.

The second part is an IP stack. This will be responsible for setting up the PHY chip, negotiate Ethernet link parameters, setting up a structure for handling incoming and outgoing packets (of different layers and protocols) and manage the Ethernet traffic. It will also be responsible for running a webserver to be used for evaluation. It will communicate with the PHY chip using RMII.

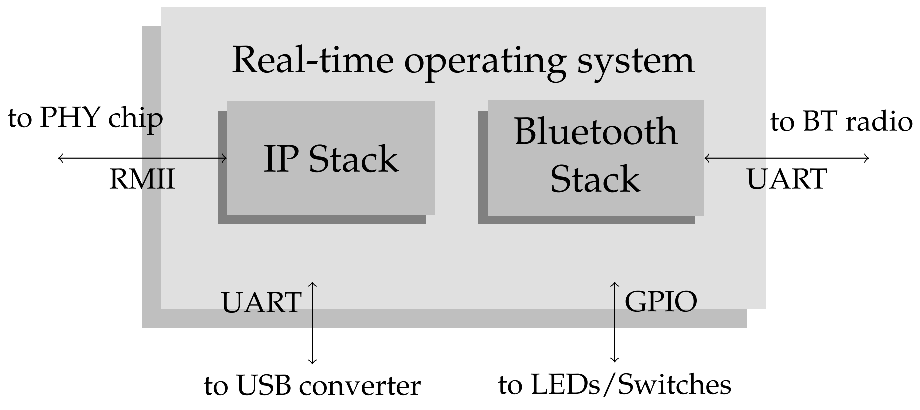


Figure 3.4: An architectural view of the software.

The third part is the Bluetooth stack. In similarity to the IP stack, this will be responsible for setting up the Bluetooth radio chip, setting up the Bluetooth Low Energy protocol suite and manage the Bluetooth traffic. It will communicate with the radio chip using a UART interface.

Chapter 4

Detailed Design

In this chapter a detailed description of the hardware and software implementation is given. It explains how the different parts of the design work, and the reasoning behind certain design decisions.

4.1 Hardware

Hardware design is an iterative process following a multitude of steps, where each new step is a bit closer to the final, physical product. The steps followed in this project include: mechanical CAD design, schematic design, and printed circuit board design. When working with new designs a lot of time is usually spent looking up components and reference designs, in fact most of the projects hardware schematics are based on or simply identical to the integrated circuit (IC) manufacturer's "typical application" design. The usage of ICs were preferred over discrete components to save space and keep the design as simple as possible.

IronCAD was used for the Mechanical CAD part. The Cadence Allegro suite was used for schematic and PCB layout design.

4.1.1 Mechanical CAD

Due to the size requirements of the project and irregular shape of the case a mechanical CAD model of the design was created to ensure that components would fit inside the casing. This model was used throughout the hardware design process to verify new component placements and help decide on different IC packages when there was more than one to choose from. Most ICs used in the design are variations of Quad-Flat No-leads (QFN) or Dual-Flat No-leads (DFN) packages or similar. These are often the smallest available packages. For common components such as general purpose resistors and capacitors 0402 packages were used and 0603 packages for 100 V rated components.

It quickly became clear when more and more features were added that a two sided design would be needed, meaning that components would be soldered on both sides of the board. While a single sided design is often cheaper, a double sided design could (in theory) result in almost half the pcb size, a big factor for production costs.

4.1.2 Power over Ethernet

Because of the limited space available on the PCB it was preferable to have both power and data in one connector. The drawback in this particular case is that the IEEE 802.3af version of Power over Ethernet supplies 37–57 V to the Powered Device, while the radio module requires 3.3 V. To convert this voltage requires large transformers, capacitors, inductors, power diodes and rectifiers that probably take up more space than an external power jack would. There are RJ-45 connectors that include a PoE transformer and bridge rectifiers but they did not fit in this design due to the placement of a mounting hole.

The Power over Ethernet circuit was based on a design from Texas Instruments[14] (TI). This design follows all the design constraints of a PoE Powered Device design according to the 802.3af specification. It uses two TI ICs, TPS2375 and TPS54160, for handling the PoE handshaking protocol and converting the voltage. All components connected directly between the power lines before the voltage converter must be 100 V rated.

Power

Figure 4.1 shows the input connection of the design, the Ethernet transformer separates the common mode voltage from the data signals. The data signals are then sent to the physical layer transceiver. Table 2.1 shows the possible pinouts of a PoE design; a Powered Device must be able to handle all available power modes so two bridge rectifiers are used to ensure correct polarity. The two ferrite beads blocks high frequency noise on the power lines and the zener diode protects the circuit from voltage spikes above 58 V.

TPS2375 - Powered Device Controller

Figure 4.2 shows the Texas Instruments TPS2375 IC that handles the handshaking process. The Power Sourcing Equipment changes the different stages of handshaking by gradually increasing/decreasing the voltage applied to the power lines. Internally, the TPS2375 uses comparators[15] to sense the voltage level and activate/deactivate different pins and logic circuits to be able to change function according to the current handshaking stage. Until the handshaking is complete VSS is completely isolated from GND, making sure the detection signatures from Table 2.2 can be maintained.

R107 that is connected to the CLASS pin of the IC determines the power class of the device. When classification is active, the CLASS pin outputs 10 V from an internal regulator. With an estimated power supply demand of under 3.84 W for the entire design Table 2.3 shows that for a class 1 device R107 should thus be

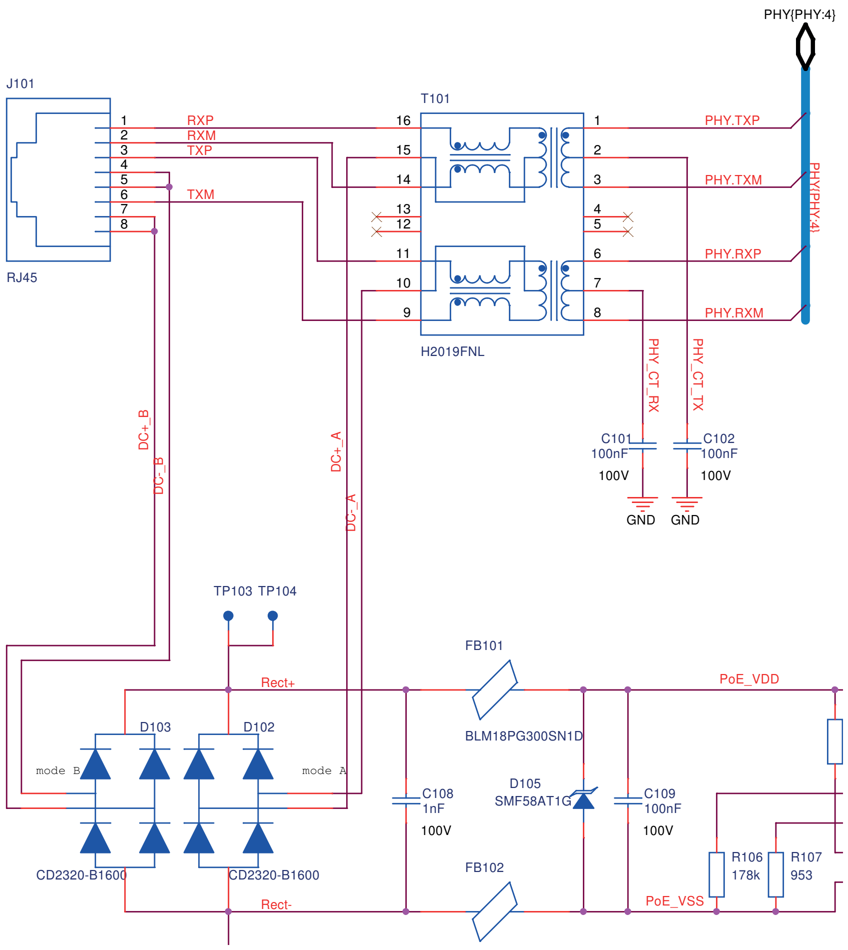


Figure 4.1: Data and power are separated in the transformer.

roughly 1 kΩ. The selected value of 953 Ω leaves a margin for some power to be dissipated in the voltage regulator etc. without the design drawing more than 12 mA of current from the PSE.

Once the link is up VSS and RTN are connected and the rest of the design is powered. The IC includes inrush current protection (ILIM) and a power good (PG) signal.

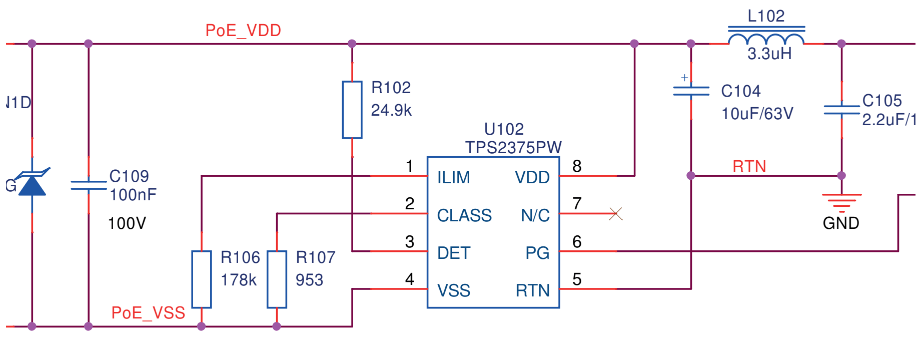


Figure 4.2: The Texas Instruments TPS2375 handles the handshaking process when setting up the PoE link.

TPS54610 - DC/DC Converter

Since the supply voltage of the system is 3.3 V a voltage converter was needed to lower the PoE voltage. The chosen TPS54610[16] chip has an input voltage range of 3.5–60 V. While this is perfect for the high voltage range of PoE, it also allows for lower voltage sources like the 5 V power supply from a USB connection. Looking at the schematic in Figure 4.3, there are two possible power sources going into VIN, one from the RJ45 connector and one from a USB connector (VBUS). There are two different mounting options here; either the pin header and diode (Debug), or the 0 Ω resistor (No Debug) is mounted. The latter is the same as having a jumper connecting pins 3 and 4 of the pin header, meaning that the power good signal from the handshaking chip is connected to the enable (EN) pin of the converter. As mentioned above this ensures no power is drained until a stable PoE link is up. If the jumper is moved to pins 1 and 2 USB power is connected and the enable pin is disconnected. Since the EN pin uses an internal pull-up resistor the voltage converter will be enabled as long as it is powered. The TPS54160 is a voltage step-down converter, also known as buck converter. Instead of getting rid of excess energy in form of heat dissipation it uses switching to convert the voltage. The key components to a switching power regulator is an inductor, a diode, and a switch. In this case the switch contained in the converter chip is a MOSFET. When the transistor is conducting, the entire voltage of VIN will be presented on PH, and current will start to build up in the inductor coil. Immediately the inductor will try to counter-act this change in current by producing an opposing voltage effectively lowering the voltage across the load. When the current in the

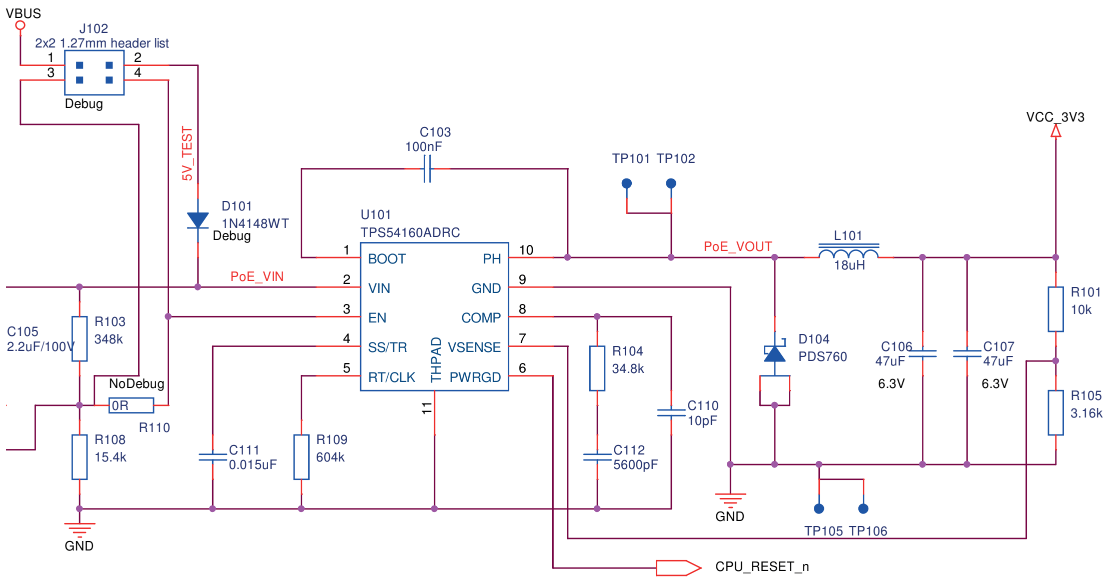


Figure 4.3: A DC/DC buck converter and supporting components. A pin header may be included to select between power sources.

inductor builds, the voltage across it starts dropping and the voltage across the load rises. If the MOSFET is turned off, the current in the inductor will be cut off from its source and will start decreasing. Since the built up energy will want to keep the current flowing, this will cause a change in voltage polarity across the inductor, aiding the current and opening the schottky diode. This smaller circuit will keep conducting until the current in the inductor reaches zero, or the regulator switches on the MOSFET. By switching between these two states and by using storage capacitors the load voltage can be kept at a stable level.

4.1.3 Ethernet Physical Layer

An Ethernet transceiver was needed to interface the Ethernet data signals to the MAC RMII interface of the MCU. 10/100 Mbit/s on twisted copper wire uses two differential data pairs, one for transmission and one reception. The four data signals TXP, TXM, RXP and RXM were connected from the Ethernet transformer (used to separate the power and data) to the PHY chip.

4.1.4 Printed Circuit Board Design

During the PCB design, or layout, there were a few key areas that needed extra consideration; the power converter, the USB data signals and the RMII signals. It was also important that every component had a good connection to ground. A six layered design was used; two layers for top and bottom component placement and signal routing, one extra signal layer underneath each surface layer, one ground plane and one power plane. Vias, drilled holes, are used to connect different layers together. Micro-vias were used to connect the surface to buried signal layers. These vias are smaller than ordinary vias so they are useful when there is not a lot of space for signal routing, as was the case in this project.

It was desirable to keep the PoE voltage separated from the system voltage as much as possible, to keep the system voltage stable at 3.3 V. Therefore the PoE and power components were kept together, so the power and ground planes for the rest of the design could be as large as possible. Switching voltage converters, as was used in this project can create a lot of noise if the layout is not carefully designed, as there are a few fast changing currents and voltages involved. The data sheet for the converter included layout guidelines to keep these from interfering with the rest of the circuit. The loop area formed by the input bypass capacitor, the anode of the catch diode (C105 and D104 in the schematic) and the VIN pin was kept as low as possible by placing those components close together. Also the ground and power line traces were made relatively large and extra vias to the ground and power planes were added to allow for good current flow.

The USB and Ethernet data signals are differential, meaning extra care has to be taken when routing them. The impedance of two lines in a differential pair is usually controlled to be as matched as possible, so that high frequency noise will affect both pairs equally to prevent distortion of the signal. This is controlled by adjusting the width of the signal trace, and to route the signals at a fixed distance to each other. Some differential signals, like the USB data pair, are standardised to have a set characteristic impedance (USB has 90 Ω). This can usually be controlled by adjusting the width of the trace and the thickness of the isolation material between the trace layer and ground plane (the distance to GND). For high-speed differential signals that have a set impedance, it is important that there is good ground connection along the trace.

Signal skew also has to be kept to a minimum to prevent both differential signals being low/high at the same time, as this will produce a lot of jitter. This is controlled by adjusting the total length of the individual traces. This is also important for signals belonging to a common interface, like the RMII signals, so that signal fronts arrive at the same time.

Lastly, another important design consideration is to keep decoupling capacitors close to the supply pins of digital circuits, to keep the supply voltage as stable as possible.

Appendix B shows the produced PCB copper layers of the design.

4.2 Software

There are a multitude of different solutions to implement the parts of the gateway software. Many available IP and Bluetooth stacks are written to be easily ported to everything from 8- or 16-bit systems to more advanced Linux kernels. Some include their own minimal operating systems and most of them are extremely modular, giving the user control over what parts to include in their projects. Writing own IP and Bluetooth stacks would be a whole new project in itself, so as much code as possible was used from open source projects and demonstration software. The scope of the software part of the thesis project was thus to find appropriate software solutions, port them to the STM32F439 processor and write a demonstration application.

As mentioned in section 3.2.2 the software can be split up into the following parts:

- Real-time operating system
- IP Stack
- Bluetooth Stack

There is also a hardware abstraction layer of processor specific drivers and routines used to access different hardware resources like UART, RMII, DMA buffers, timers and so on. These were freely available from the manufacturer. All code was available and written in C and compiled using the GCC ARM toolchain.

4.2.1 Real-time operating system - FreeRTOS

Some thought was given at the beginning of planning the software, as to how it was going to handle multiple inputs from multiple sources at as low delay as possible. Some situations where the gateway could be used, demanded the gateway to be able to respond within a guaranteed amount of time. An example would be that the IP stack would need to process incoming Ethernet packets before the receive buffers fill up and packets are dropped, even if the processor would be busy. It was decided that an operating system capable of multi-tasking and context switching based on priority would be used. A real-time operating system specifically written for embedded systems called FreeRTOS[17] was chosen. Since it had already been ported to an ST microcontroller in the same processor family, using the same or similar drivers, a lot of code could be borrowed.

The scheduler in FreeRTOS uses fixed-priority pre-emptive scheduling. This means that each thread/task will be given executable time based on its set priority. When a task is finished executing, the processor will be given over to the highest priority task ready to execute. Pre-emptive scheduling means that a higher priority task will be given processor time as soon as it is ready to execute, even if another lower priority task is running. The lower priority task will be pre-empted, but allowed to finish when the higher priority task is done. New tasks can be created dynamically during run-time, and delete themselves (or each other) from the scheduler list when/if they have completed their tasks.

4.2.2 IP Stack - lwIP

LwIP[18] is a popular open source IP stack, that like FreeRTOS, also had been ported to a similar microcontroller. It was therefore less work in making it work with the gateways MCU.

The IP stack starts by initiating and configuring the hardware required on the MCU; the MAC, GPIO pins for RMII and DMAs to move incoming and outgoing data to and from memory buffers. It also configures the PHY chip using the RMIIs management interface. It then initiates the different IP layers and registers threads with the operating system for incoming connections and traffic. A DHCP client is started when the stack is up and running.

Fortunately there was a web server in the contribution area for lwIP. This was easily added on top of the running lwIP stack. When browsing to the IP address of the gateway a web page was now displayed. The html files were stored as images in a file system. They were extracted and with a python script converted to ascii. The web page could now be re-written and used for demonstration purposes. The web server also included a Common Gateway Interface (CGI) and Server Side Includes (SSI) handler. This made it possible to call CGI-scripts from the web page and trigger application code on the gateway. This was used to call helper functions in the Bluetooth stack.

4.2.3 Bluetooth Stack - BTstack

BTstack[19] was chosen as the Bluetooth stack used for the project and proved to be simple, yet versatile. The stack runs in a single thread. The thread normally does not block, but was rewritten to block for 250 ms or until an incoming packet was signalled by the UART. The stack is completely event driven, using handlers. When an incoming packet is detected it is sent to the handler for the target protocol. The handler then parses the information in the packet and decides the next action to take. For example, a string of HCI commands can thus easily be chained by telling the HCI handler that when a $COMMAND\_COMPLETE\_EVENT$ for the first command is received, the next command should be sent.

Like lwIP it starts by initiating the required hardware, and setting up appropriate data structures. It also configures the off chip radio module using a UART connection. All Bluetooth packets are sent and received using this UART.

Chapter 5

Evaluation

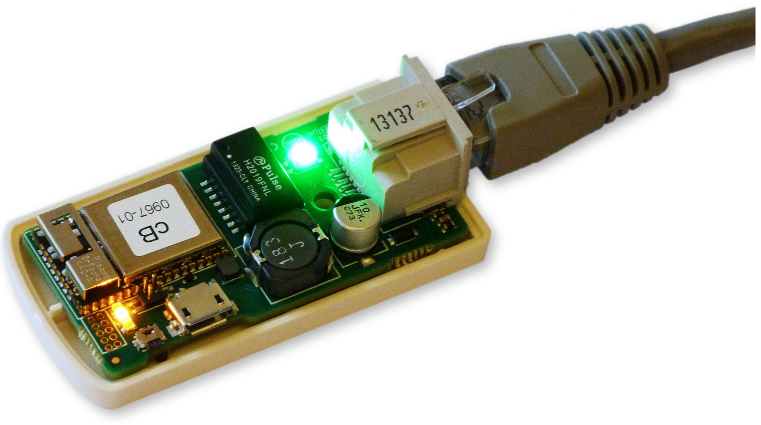


Figure 5.1: The finished prototype being powered by PoE.

After the hardware prototype had been produced it had to be verified. Before connecting anything, measurements had to be made to make sure the key power components were properly connected and that the power planes were intact. The next thing to do was to connect an Ethernet cable with PoE and measure the voltages at the different test points. Everything worked perfectly.

With power up and running the next step was to start testing USB, Ethernet, JTAG/SWD interfaces etc. Since these functions depend on instructions from the processor it was important to incorporate the evaluation of these modules into the software development process. To properly evaluate the gateway something to connect to was needed, a standard Ethernet router and a Bluetooth Low Energy sensor was provided and is shown in Figure 5.2. The sensor has a couple of LEDs, an accelerometer and a temperature sensor that can be controlled using a GATT client.

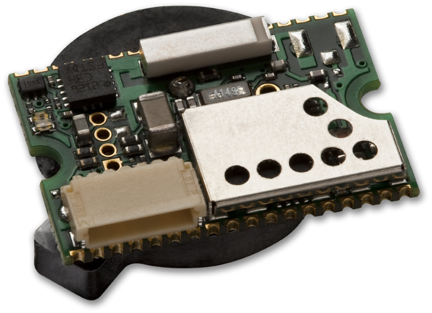


Figure 5.2: An OLP425 BLE sensor powered with a coin cell battery.

5.1 Tests

5.1.1 Power

Since the gateway powered up as soon as a PoE powered Ethernet cable was connected (when the jumper was set to PoE mode) it was obvious that the PoE circuit worked. However it was still of interest to see if it worked as intended, so an oscilloscope was used to measure the voltage between the V_{DD} and V_{SS} pins of the TPS2375 and over the buck converters storage capacitors.

5.1.2 Functionality

To test the functionality of the gateway and software, the small example described at the beginning of section 3.2 was expanded. A web browser is used to open the webpage hosted on the gateway. On the page the user can control an LED on the demonstration Bluetooth sensor. Javascript is used to provide updated information from the server along with Common Gateway Interface (CGI) scripts. These are common ways to provide interactivity on a webpage. Figure 5.3 is updated to show what should happen when the button to activate the sensor LED is clicked. During this test, the core system was clocked at 96 MHz.

1. Upon clicking the button to activate the sensor LED, a GET request is sent with a call to the CGI script *leds* with the parameter *LED*. Upon reception it travels up the IP stack and is passed to the webserver. The webserver calls its CGI handler that locates the *leds* script. The script recognises that LED number four is on a BLE sensor and promptly calls a helper function to queue a GATT characteristic write to the correct sensor.
2. The script then returns an HTTP response saying it went fine and some javascript code to trigger a response on the webpage.
3. When the context switches to the Bluetooth thread, the outgoing command is detected and sent through the Bluetooth stack to the radio module then out on the air.

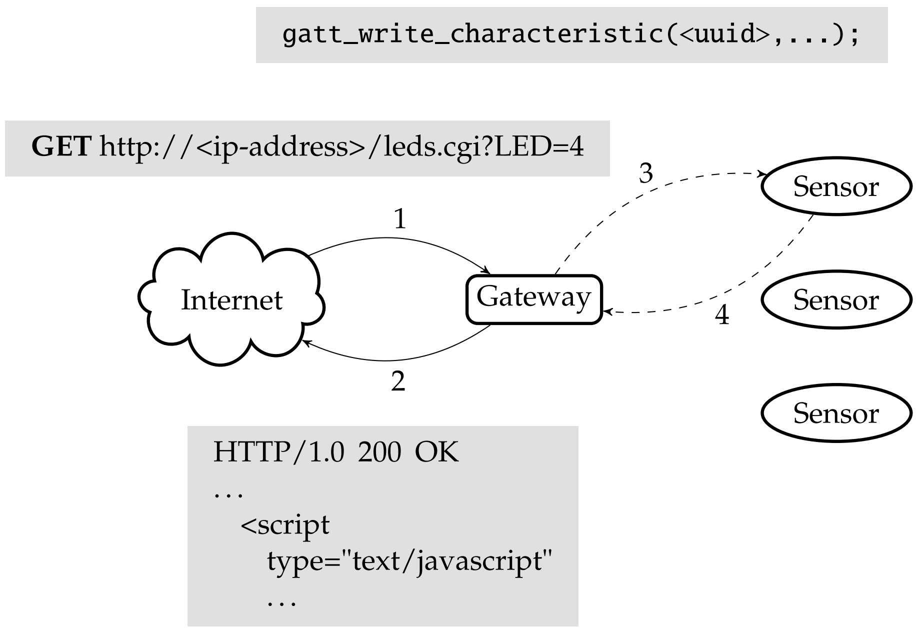


Figure 5.3: Evaluation of the gateway core functionality.

4. A response may be sent by the sensor, but will be ignored and discarded by the Bluetooth thread application. It is not interested in write responses.

5.2 Results

5.2.1 Power

Figure 5.4 shows the waveform of the voltage measured at the PD controller chip. First the voltage is stable at just below 2 V, then rises to about 15 V and then drops slightly for about 20 ms before operational voltage is applied. In Table 2.2 it says that during detection, at least 2.7 V must be applied to the cable. The reason this is not measured is probably due to losses in the cable and the forward voltage of the rectifiers. There is a noticeable decline in slew rate when the voltage reaches 12 V (marker "a" in the figure). This is when the internal 10 V regulator is switched on to drive the classification current. During the classification stage the voltage seems again a bit low, it should be at least 14.5 V. This can again be explained by line loss and forward voltage. The idle time before operational power is applied seems to be implementation specific, the only requirement in the PoE specification is that power must be applied within 400 ms so this is well within that limit. The first glitch, at 22 V, is when the internal 10 V regulator is turned off and the second, at 39 V, is when the operational voltage limit is reached and the buck converter is connected. At marker "b" in the figure we see that 47.2 V is being measured proving that the Power over Ethernet implementation is working.

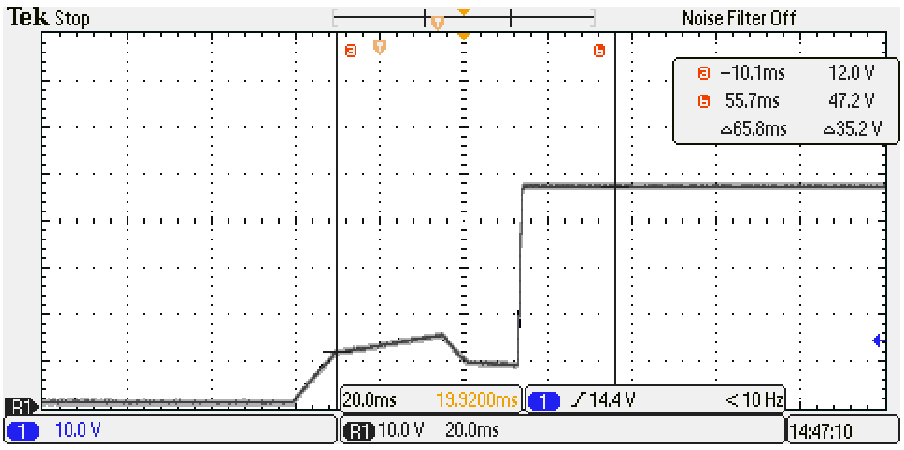


Figure 5.4: The voltage measured across V_{DD} and V_{SS} during the handshaking process.

Figure 5.5 shows the 3.3 V operational voltage for the rest of the gateway, measured across the buck converters storage capacitors. At this stage it is important that there are no voltage spikes on the power line as this can damage more sensitive components. There is a small spike when the regulator switches on, this pulse will contain a relatively small amount of energy and will be captured by decoupling capacitors and ferrite beads protecting sensitive chips in the design (see Appendix A). As can be seen in the figure the transient is a smooth step up to 3.36 V with no overshoot and no noticeable noise (that will affect performance).

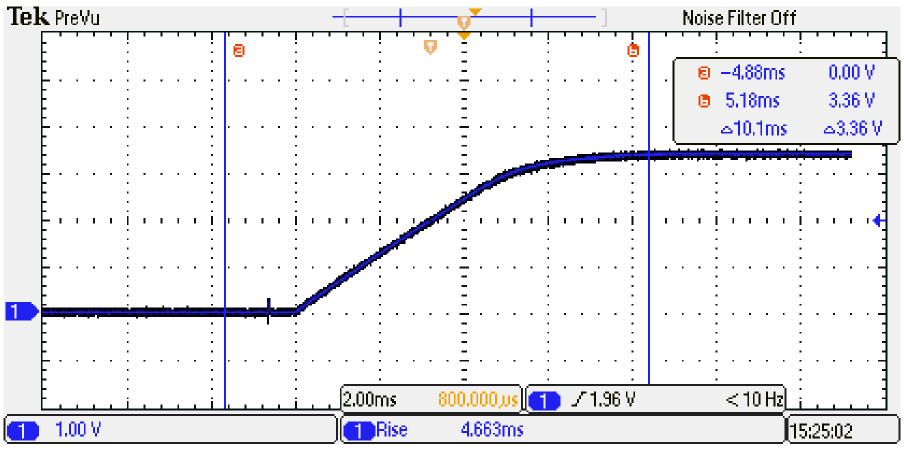


Figure 5.5: The voltage measured across the DC/DC converter after the PoE handshaking is done.

In conclusion it can be said that the power part of the design is working as expected.

5.2.2 Functionality

The functionality test worked perfectly. The browser successfully loaded the page and the LED could be toggled on/off repeatedly. Options to connect/disconnect the sensor was added and also a feature to continuously read the value of the temperature sensor and have it shown on the page. The sensor was moved further away from the gateway, and was fully functional up to about 15 meters away. This was due to the on-board antenna of the sensor. To get extended range an external antenna could be connected, but this was not attempted.

The time it took for the LED to respond varied from being almost instantaneous to up to over 3 s. Options to toggle the RGB indication LED on the gateway was added so both LEDs could be toggled at the same time, with the difference that the command to toggle the sensor LED also had to be sent over Bluetooth. The time difference was conclusively only a few milliseconds (not surprisingly the sensor LED responded slower), just noticeable when observing both devices. This proves that the BLE connection was working. The seemingly random delay in time remained, and could be either due to Ethernet traffic or the IP stack. To further test this, the test was attempted yet again but with the gateway and test computer connected by themselves on a separate network. The results were the same as before, meaning the IP-stack implementation is a probable cause for slowing down the connection. No further attempts to trace the source of this was attempted, since the application works as intended; fully demonstrating the capabilities of the gateway. However the webpage was expanded to include a feature that continuously read and displayed the value from the temperature sensor on the BLE device. A screenshot of the webpage can be seen in Figure 5.6

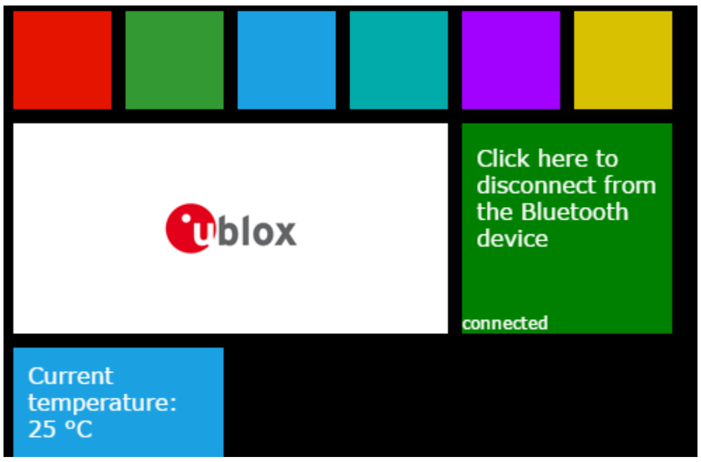

ublox
Click here to disconnect from the Bluetooth device
connected
Current temperature: 25 °C

Figure 5.6: A screenshot of the webpage hosted on the gateway.

5.3 Further Development

5.3.1 Hardware

The printed circuit boards will need some small modifications in a second hardware spin, if this project should be taken into production. Also further verification will be needed. Electrostatic discharge tests to make sure internal components are protected from high voltage discharges to connectors, push-buttons, PCB edges etc. The prototype will also need emissions testing, to make sure it does not radiate electromagnetic waves that can disturb other equipment or transmissions, or that itself is not affected by emissions from other sources. Acceptable emission levels are controlled and legislated by different organisations in different economic areas around the world and products meant to be sold in certain areas needs to comply to their standards. In the European Economic Area, Conformité Européenne (CE) compliance tests and marking are mandatory, while Federal Communications Commission (FCC) declaration of conformity is needed for electronic products intended to be sold in the Unites States.

5.3.2 Software

The code written for this project was only intended to demonstrate the capabilities of the hardware. However the gateway could easily be turned into a real product with a few optimizations of and additions to the software. Examples of improvement could be to include ways to power down both the processor and peripherals when they are in an idle state. Careful considerations would have to be made however, so that enough power will be drawn from the PoE link, if used, as can be read in section 2.2.4. More critical, is the power usage of the connected BLE peripherals. Since many of these devices are designed to be battery powered, and spend most of their time in a sleep state to save power, it is desirable to disconnect from the peripherals as soon as they are no longer needed.

Since the STM32F439 includes a hardware accelerated crypto engine, it should be a small step to add an SSL library etc. so that wireless and Ethernet traffic could be encrypted. This is probably the most pressing improvement needed. Some sort of authorization system could also be included if the connected sensors are meant to be accessible from outside the target network. For example, a recent (2014) Master's Thesis[20] details a very high security, server based, authorization system made for industrial applications that could be built into an embedded device.

As mentioned in section 5.2.2 the IP stack implementation should be looked over, as it is slowing down the system.

5.4 Contributions

- The original idea and project proposal, ODIN-W260 multi-radio module, the plastic enclosure and the OLP425 BLE module/sensor is the work of u-blox Malmö.
- FreeRTOS is the work of Real Time Engineers.
- lwIP is the work of many contributors, but was initially developed by Adam Dunkels at SICS.
- BTstack is the work of BlueKitchen.

The rest of the work described in this report is the contributions of its author. This includes hardware and software design and implementation choices, mechanical design, hardware schematics design, printed circuit board design and production documentation, porting and assembly of the different software components and writing the web page and supporting scripts.

5.5 Related Work

This section analyses similar solutions that are available on the market and compares them to this thesis work.

Intel recently (April 2014) released a range of IoT Gateways[21]that are intended to be used as evaluation kits for their Quark and Atom SoCs. These are powerful processors that run Linux distributions and support multiple wired and wireless interfaces including 2G/3G, WLAN, Bluetooth, Ethernet, CAN etc. In comparison with the thesis gateway they are much more powerful but also much more expensive. Intels gateways are more suitable for “big data” collection if the data needs to be heavily processed before being sent on, while the thesis gateway is more focused on transparently connecting sensor nodes to larger networks.

NXP’s Internet of Things gateway JN5168-RD6040[22] connects ZigBee networks to the internet. Like Intel’s gateways it also uses a more powerful processor to run a Linux operating system. Looking at functionality it offers the same things as the thesis gateway like a web server and user application code, except it does not have built in WLAN. Considering that it uses an ARM9 based processor this “lack” of functions is surprising, however this means it has a lot of processing power available if needed.

In conclusion, there are some similar devices on the market, but the big difference is size, price and functionality. Most of the other gateways found run some version of Linux that require more powerful processors and are therefore more expensive. The unique things with the thesis gateway is that it is very compact (53.6 x 26.2 mm) and still features Ethernet, Bluetooth and WLAN, without an external antenna. Because of its small PCB size and lower cost processor (ARM Cortex-M4) it is also probably a lot cheaper (the estimated price of the prototype is not publicly available) than for instance NXP's gateway. Whether it can perform as well as the others would have been an interesting test.

5.6 Conclusion

An Ethernet gateway that can be used to connect a Bluetooth Low Energy sensor to the Internet has been designed and built. It has been based on an existing multi-radio module, and the software was based on a combination of open and freely available source code. It can be powered using a Power over Ethernet connection or a USB cable and can be assembled into a plastic casing. A web page was written and is hosted on the gateway, enabling control of a connected sensor. The evaluation of the hardware prototype shows that it works for demonstration purposes, and with the discussed improvements the gateway could be turned into a consumer product.

Such a gateway could be used to connect Bluetooth and/or BLE devices to each other and the internet, or serve as a WLAN access point. Networks of relatively dumb sensor nodes could be formed, protected from unauthorised access by the gateway while still being accessible to the Internet.

This has been a very practical master's thesis, mainly focusing on the implementation of a concept. I worked using a set of requirements that was changed from time to time, both by me and by others, but I was always given much freedom of choice concerning what features to add, what components to use etc. and it was both liberating and daunting to be faced with this much creative responsibility. The porting and assembling of the operating system, IP stack and Bluetooth stack proved to be a big challenge, not only getting the different software components to talk to each other but also getting the entire project to link and build properly. I learned a lot about product design and development especially printed circuit board manufacturing, something completely new to me prior to this thesis work. I am very pleased with the resulting prototype and hope that my work will be built upon to make a real product.

Bibliography

[1] Kevin Ashton. *That 'Internet of Things' thing - RFID Journal*. 2009. URL: `http://www.rfidjournal.com/articles/view?4986`.

[2] Bruce Schneier. *The Internet of Things Is Wildly Insecure—And Often Unpatchable*. 2014. URL: `https://www.schneier.com/essay-468.html`.

[3] connectBlue Mats Andersson. *Short-range Low Power Wireless Devices and Internet of Things (IoT)*. 2014. URL: `http://www.connectblue.com/fileadmin/Connectblue/Web2006/Documents/White_papers/connectBlue-short-range-wireless-IoT.pdf`.

[4] The European Comission. *The Internet of Things*. 2012. URL: `http://ec.europa.eu/digital-agenda/en/internet-things`.

[5] ABI research. *More Than 30 Billion Devices Will Wirelessly Connect to the Internet of Everything in 2020*. 2013. URL: `https://www.abiresearch.com/press/more-than-30-billion-devices-will-wirelessly-conne`.

[6] Cisco. *The Internet of Things*. n.d. URL: `http://share.cisco.com/internet-of-things.html`.

[7] Ericsson. *CEO to shareholders: 50 billion connections 2020*. 2010. URL: `http://www.ericsson.com/thecompany/press/releases/2010/04/1403231`.

[8] USB Implementers Forum. *New SuperSpeed USB Developments Increase Speed and Power Delivery*. 2013. URL: `http://www.usb.org/press/presskit/USBIF_MomentumPR_IDF2013_FINAL.pdf`.

[9] USB Implementers Forum. *The latest (3.1) USB Specification*. 2013. URL: `http://www.usb.org/developers/docs/`.

[10] IEEE Standards Association. *IEEE Standard for Ethernet (IEEE 802.3)*. 2012. URL: `http://standards.ieee.org/about/get/802/802.3.html`.

[11] Cisco. *Cisco Universal Power Over Ethernet — Unleash the Power of your Network White Paper*. 2012. URL: http://www.cisco.com/c/en/us/products/collateral/switches/catalyst-4500-series-switches/white_paper_c11-670993.html.

[12] connectBlue. *Bluetooth Low Energy Technology*. 2014. URL: http://www.connectblue.com/technologies/bluetooth-low-energy-technology.

[13] Bluetooth SIG. *Adopted Bluetooth GATT profiles*. 2014. URL: https://developer.bluetooth.org/TechnologyOverview/Pages/Profiles.aspx#GATT.

[14] Texas Instruments. *Class 2- PD SIP Module, 3.3V Buck - Reference Design*. 2011. URL: http://www.ti.com/lit/df/slur225/slur225.pdf.

[15] Texas Instruments. *TPS2375 Datasheet*. 2008. URL: http://www.ti.com/lit/ds/symlink/tps2375.pdf.

[16] Texas Instruments. *TPS54160 Datasheet*. 2014. URL: http://www.ti.com/lit/ds/symlink/tps54160a.pdf.

[17] Real Time Engineers Ltd. *FreeRTOS home page*. 2014. URL: http://www.freertos.org.

[18] Swedish Institute of Computer Science. *lwIP development page*. 2014. URL: http://savannah.nongnu.org/projects/lwip/.

[19] BlueKitchen GmbH. *BTstack development page*. 2014. URL: https://code.google.com/p/btstack/.

[20] Niklas Hjern and Jonas Vistrand. "Authorization for Industrial Control Systems". MA thesis. EIT Lund University, 2014. URL: http://www.eit.lth.se/sprapport.php?uid=765.

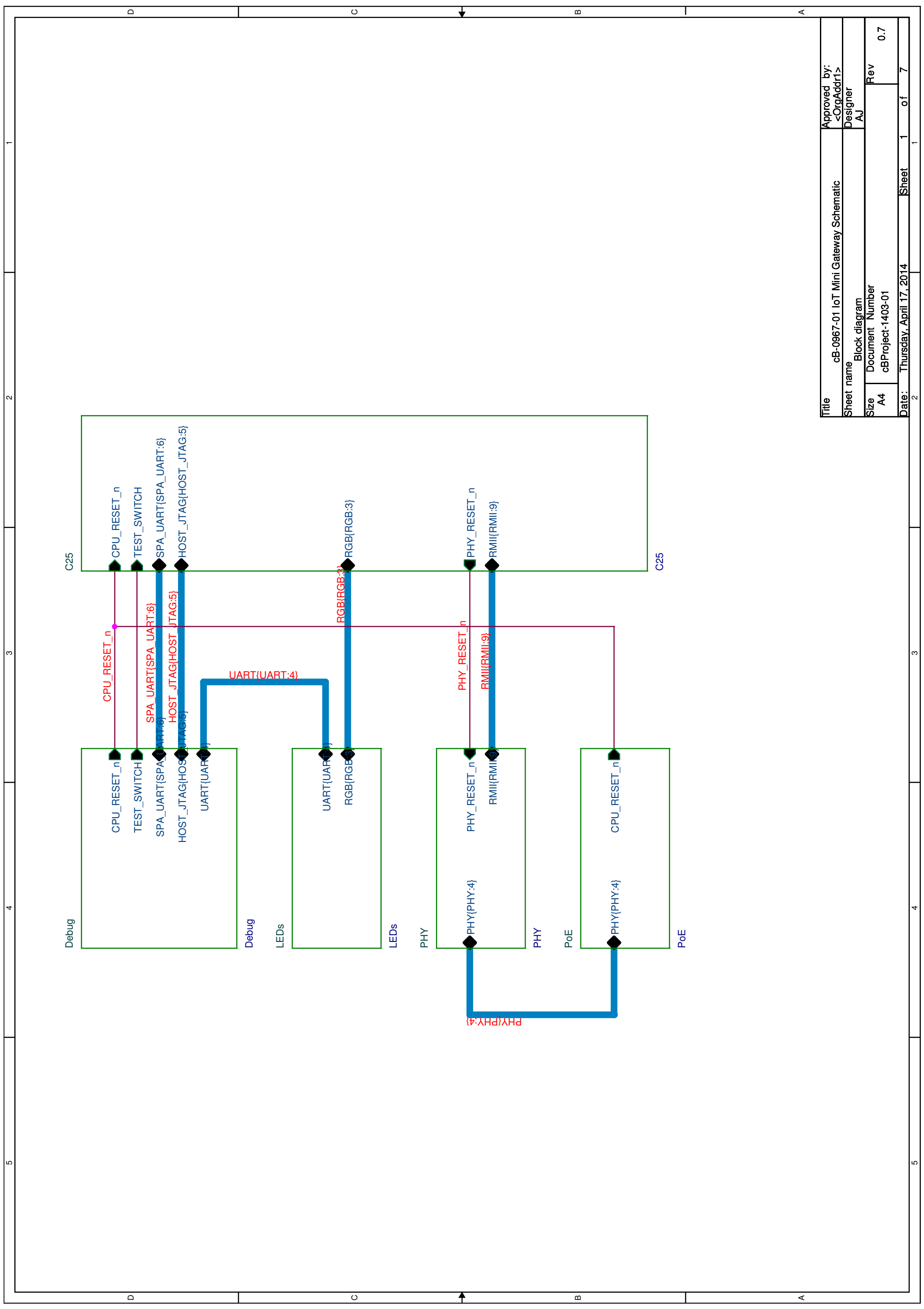
[21] Intel. *Transform Business with Intelligent Gateway Solutions for IoT*. 2014. URL: http://www.intel.com/content/www/us/en/internet-of-things/gateway-solutions.html.

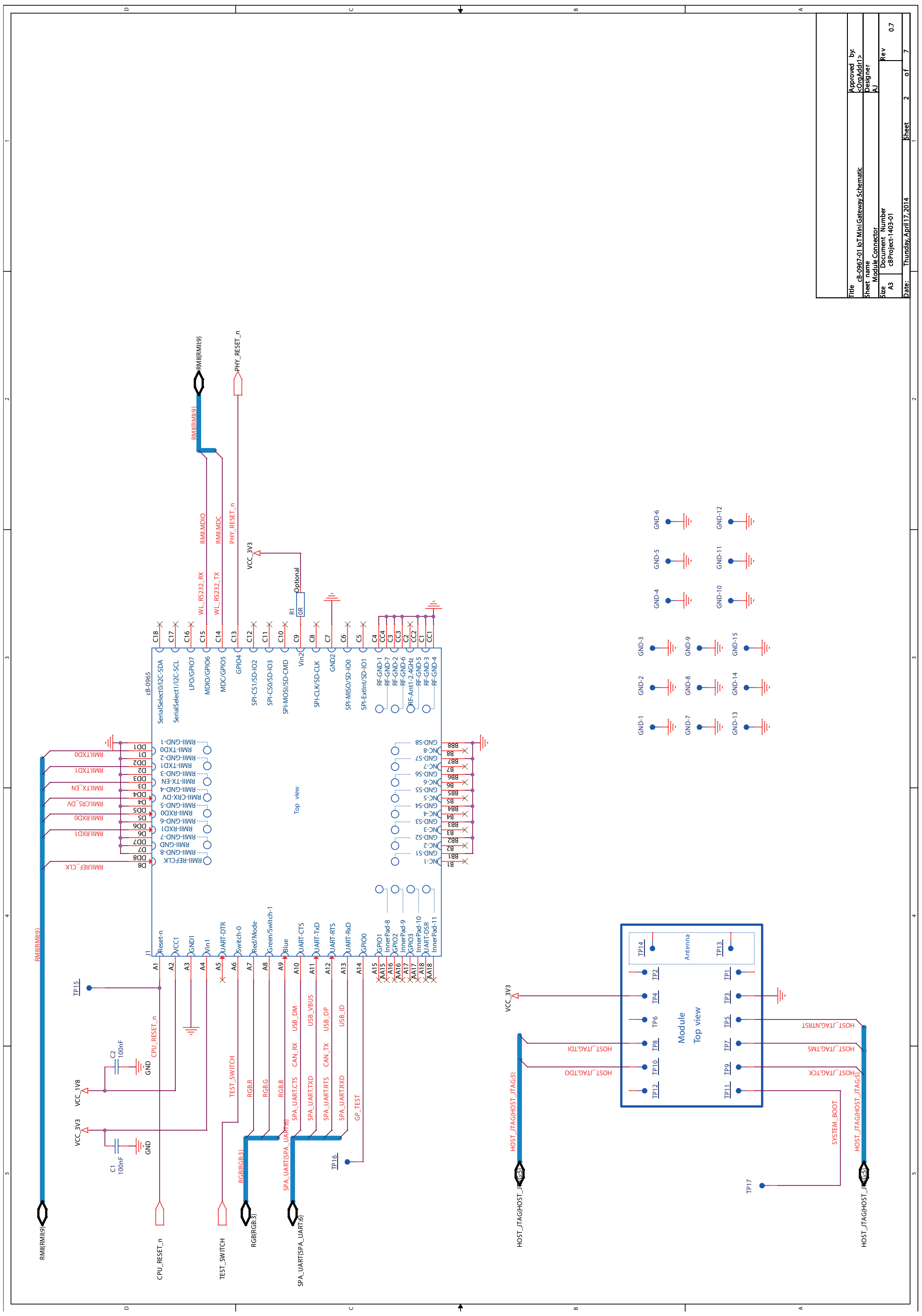
[22] NXP. *JN5168-RD6040 Internet of Things (IoT) Gateway*. 2014. URL: http://www.nxp.com/demoboard/JN5168-RD6040.html.

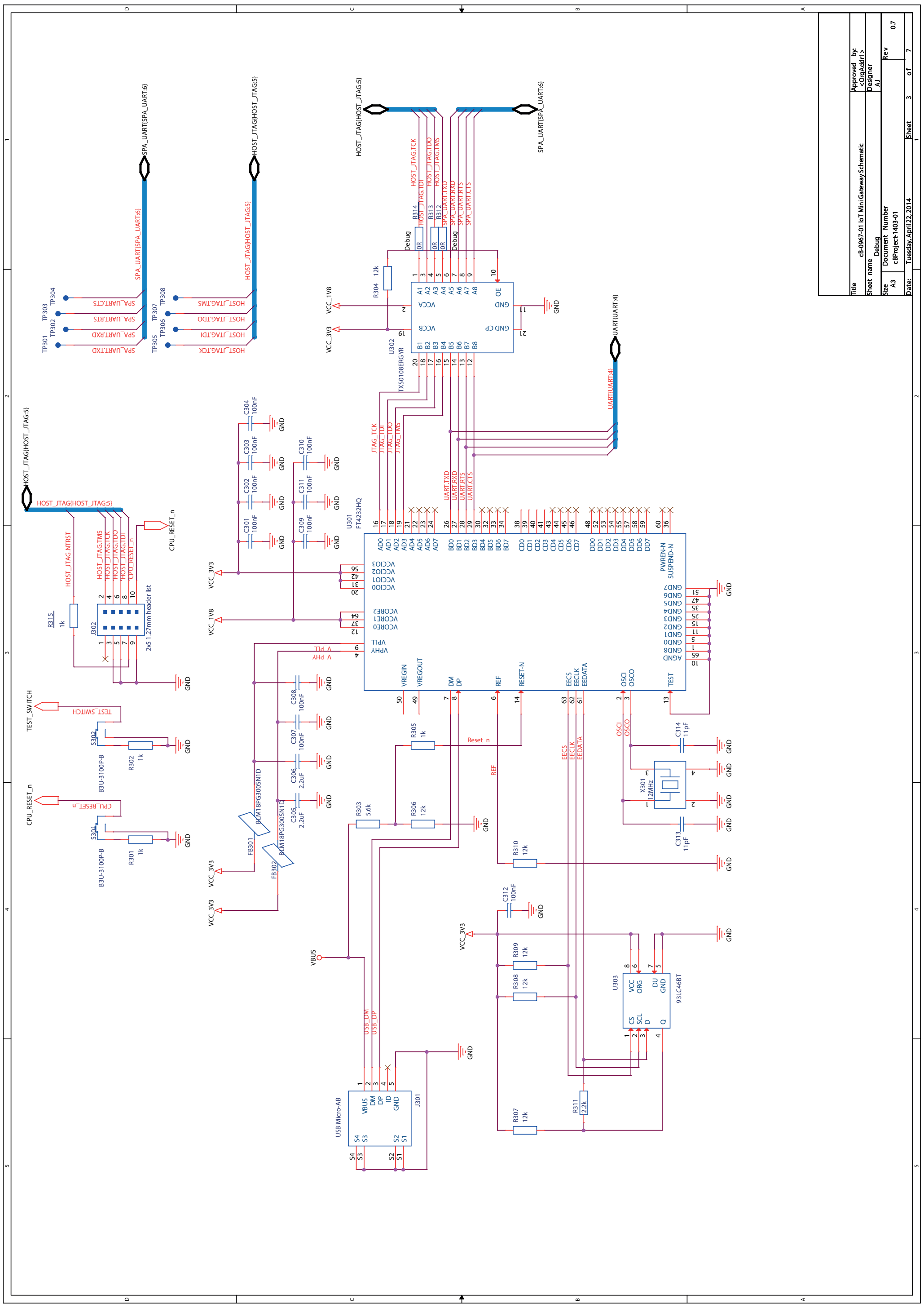
Appendix A

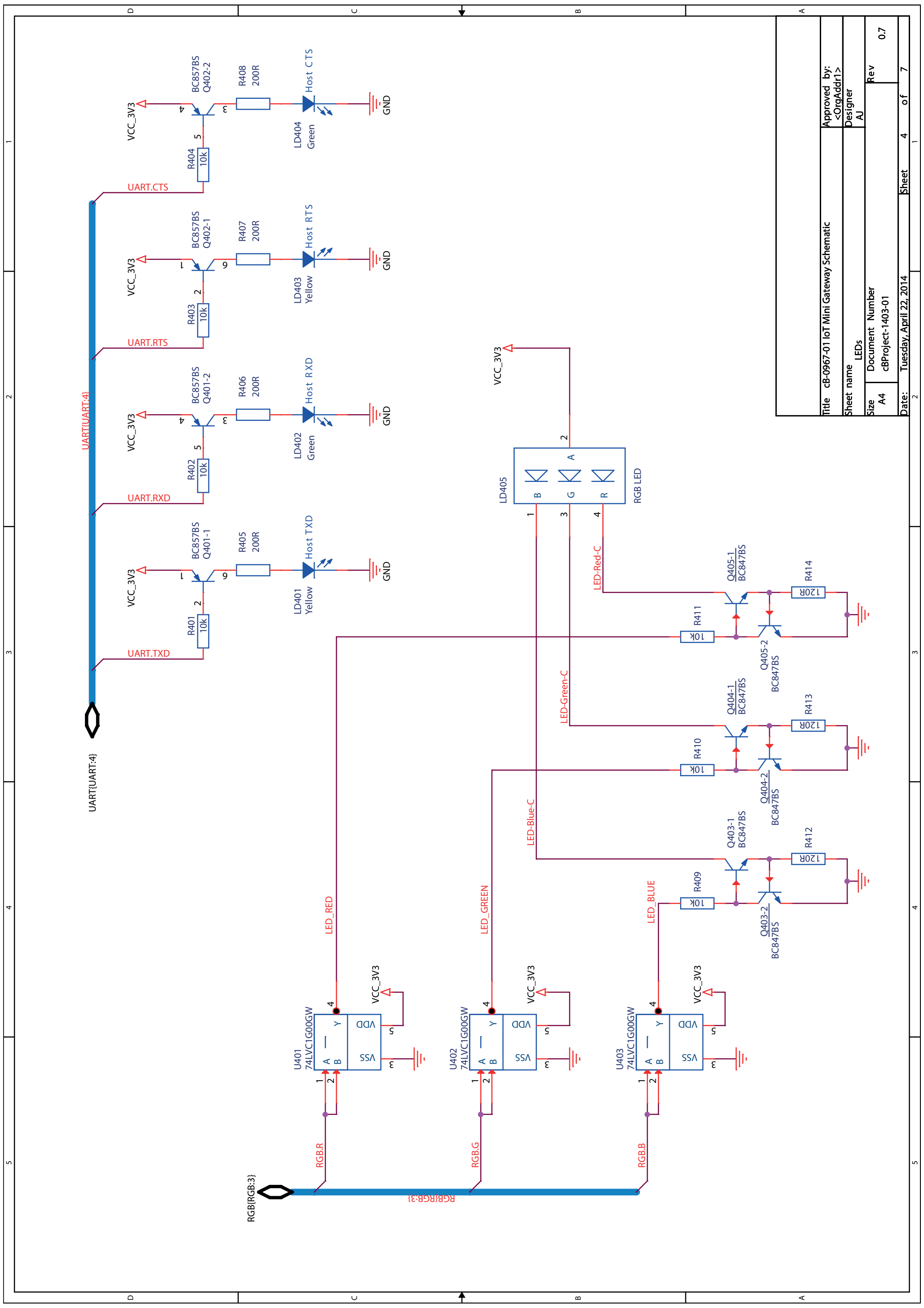
Schematics

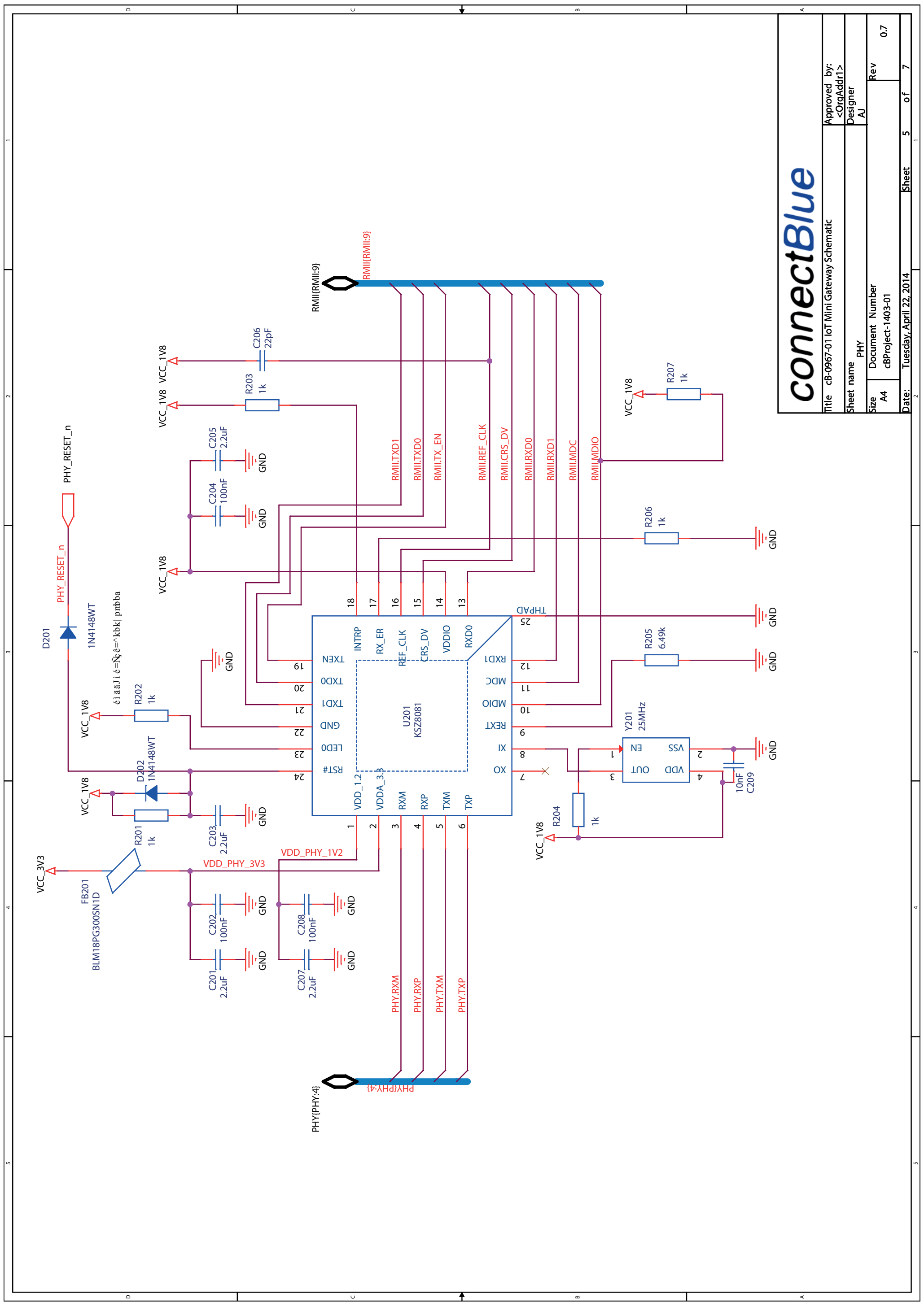
The produced hardware schematics for the thesis are shown below.


C25
CPU_RESET_n
TEST_SWITCH
SPA_UART{SPA_UART:6}
HOST_JTAG{HOST_JTAG:5}
RGB{RGB:3}
PHY_RESET_n
RMII{RMII:9}
UART{UART:4}
Debug
LEDs
PHY
PoE
PHY{PHY:4}
Title
cB-0967-01 IoT Mini Gateway Schematic
Sheet name
Block diagram
Size
A4
Document Number
cBProject-1403-01
Rev
0.7
Date: Thursday, April 17, 2014
Sheet 1 of 7
Approved by: <OrgAddr1>
Designer
AJ

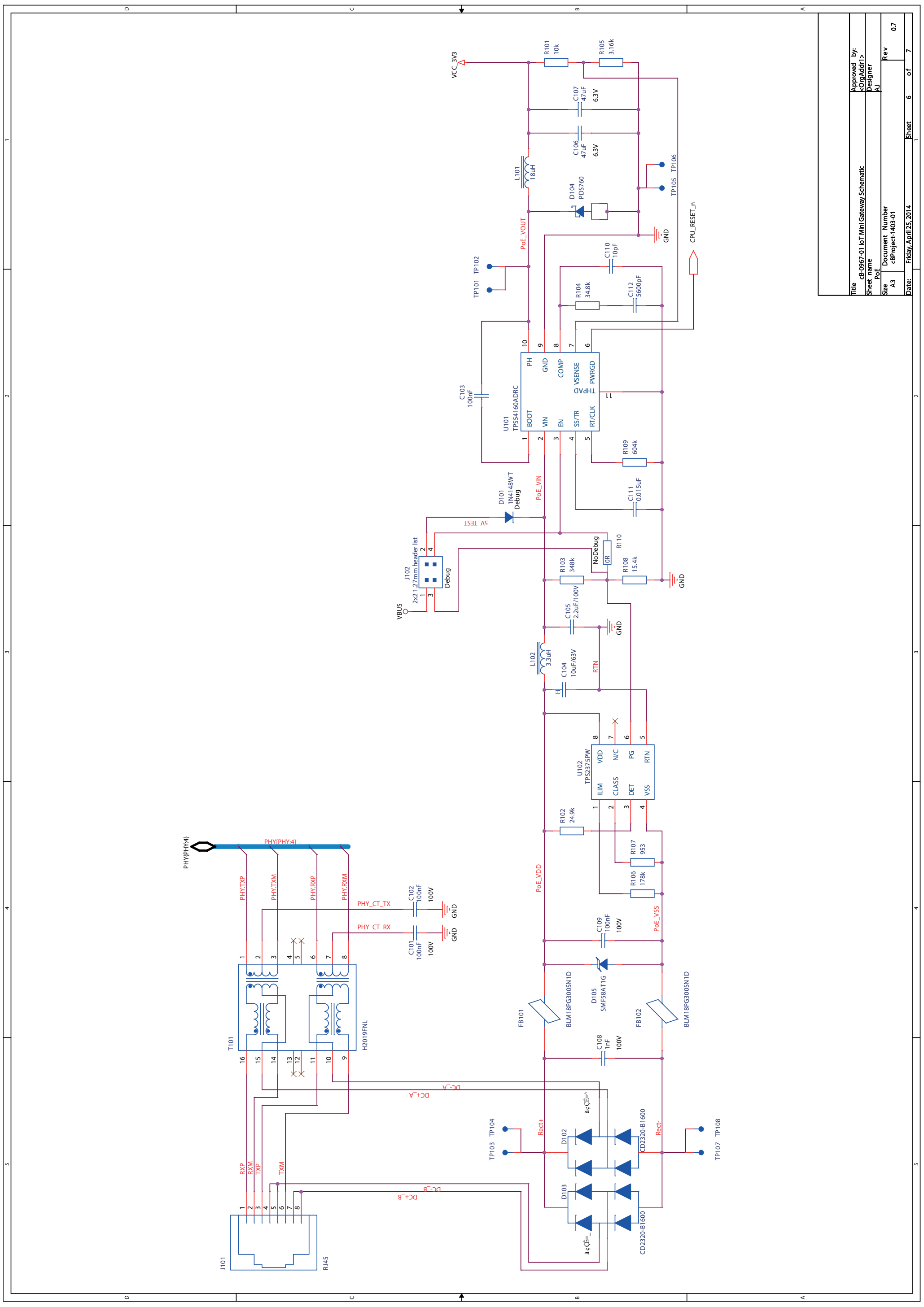



Title cB-0967-01 IoT Mini Gateway Schematic
Sheet name Debug
Size A3
Document Number cBProject-1403-01
Rev 0.7
Date: Tuesday, April 22, 2014
Sheet 3 of 7
U301 FT4232HQ
U302 TXS0108ERGYR
U303 93LC46BT
X301 12MHz
J301 USB Micro-AB
J302 2x5 1.27mm header list
S301 B3U-3100P-B
S302 B3U-3100P-B
FB301 BLM18PG300SN1D
FB302 BLM18PG300SN1D
HOST_JTAG(HOST_JTAG:5)
SPA_UART(SPA_UART:6)
UART(UART:4)
TEST_SWITCH
CPU_RESET_n
VCC_3V3
VCC_1V8
VBUS
GND


cB-0967-01 IoT Mini Gateway Schematic
Approved by: <OrgAddr1>
Designer AJ
Sheet name LEDs
Document Number cBProject-1403-01
Size A4
Rev 0.7
Date: Tuesday, April 22, 2014
Sheet 4 of 7
UART[UART:4]
UART.CTS
UART.RTS
UART.RXD
UART.TXD
VCC_3V3
GND
R401 10k
R402 10k
R403 10k
R404 10k
R405 200R
R406 200R
R407 200R
R408 200R
Q401-1 BC857BS
Q401-2 BC857BS
Q402-1 BC857BS
Q402-2 BC857BS
LD401 Yellow Host TXD
LD402 Green Host RXD
LD403 Yellow Host RTS
LD404 Green Host CTS
LD405 RGB LED
LED-Red-C
LED-Green-C
LED-Blue-C
Q403-1 BC847BS
Q403-2 BC847BS
Q404-1 BC847BS
Q404-2 BC847BS
Q405-1 BC847BS
Q405-2 BC847BS
R409 10k
R410 10k
R411 10k
R412 120R
R413 120R
R414 120R
LED_RED
LED_GREEN
LED_BLUE
U401 74LVC1G00GW
U402 74LVC1G00GW
U403 74LVC1G00GW
VDD
VSS
RGB.R
RGB.G
RGB.B
RGB[RGB:3]



connectBlue
Title cB-0967-01 IoT Mini Gateway Schematic
Approved by: <OrgAddr1>
Sheet name PHY
Designer AJ
Size A4
Document Number cBProject-1403-01
Rev 0.7
Date: Tuesday, April 22, 2014
Sheet 5 of 7
U201 KSZ8081
Y201 25MHz
D201 1N4148WT
D202 1N4148WT
FB201 BLM18PG300SN1D
R201 1k
R202 1k
R203 1k
R204 1k
R205 6.49k
R206 1k
R207 1k
C201 2.2uF
C202 100nF
C203 2.2uF
C204 100nF
C205 2.2uF
C206 22pF
C207 2.2uF
C208 100nF
C209 10nF
PHY_RESET_n
VCC_3V3
VCC_1V8
VDD_PHY_3V3
VDD_PHY_1V2
GND
RMII{RMII:9}
RMII.TXD1
RMII.TXD0
RMII.TX_EN
RMII.REF_CLK
RMII.CRS_DV
RMII.RXD0
RMII.RXD1
RMII.MDC
RMII.MDIO
PHY{PHY:4}
PHY.RXM
PHY.RXP
PHY.TXM
PHY.TXP


Title cB-0967-01 IoT Mini Gateway Schematic
Sheet name PoE
Approved by: <OrgAddr1>
Designer AJ
Size A3
Document Number cBProject-1403-01
Rev 0.7
Date: Friday, April 25, 2014
Sheet 6 of 7
VCC_3V3
R101 10k
R105 3.16k
C107 47uF 6.3V
C106 47uF 6.3V
L101 18uH
D104 PDS760
TP105 TP106
GND
CPU_RESET_n
PoE_VOUT
TP101 TP102
C110 10pF
R104 34.8k
C112 5600pF
U101 TPS54160ADRC
PH GND COMP VSENSE PWRGD BOOT VIN EN SS/TR RT/CLK THPAD
C103 100nF
R109 604k
C111 0.015uF
D101 1N4148WT Debug
PoE_VIN
5V_TEST
J102 2x2 1.27mm header list
Debug
NoDebug
VBUS
R110 DR
R103 348k
R108 15.4k
C105 2.2uF/100V
L102 3.3uH
C104 10uF/63V
RTN
U102 TPS2375PW
VDD N/C PG RTN ILIM CLASS DET VSS
R102 24.9k
R107 953
R106 178k
PoE_VDD
PoE_VSS
C109 100nF 100V
D105 SMF58AT1G
FB101 BLM18PG300SN1D
FB102 BLM18PG300SN1D
C108 1nF 100V
PHY[PHY:4]
PHY.TXP PHY.TXM PHY.RXP PHY.RXM
PHY_CT_TX
PHY_CT_RX
C102 100nF 100V
C101 100nF 100V
T101 H2019FNL
DC+_A DC-_A
DC+_B DC-_B
TP103 TP104
TP107 TP108
Rect+ Rect-
D102 CD2320-B1600
D103 CD2320-B1600
RXP RXM TXP TXM
J101 RJ45

Appendix B

PCB Layout

The PCB copper layers produced for the thesis are shown below. They are shown in a 1.3 increased scale.

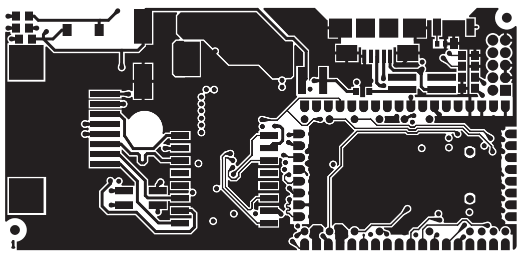


Figure B.1: Top component layer.

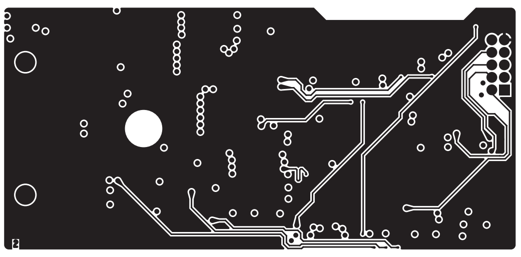


Figure B.2: Top buried signals layer.

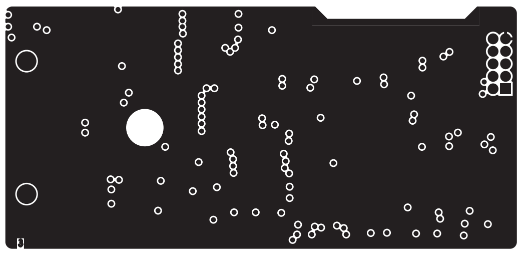


Figure B.3: Ground plane.

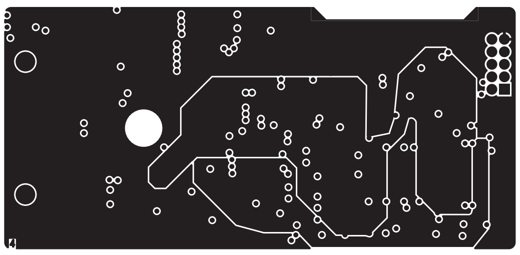


Figure B.4: Power plane.

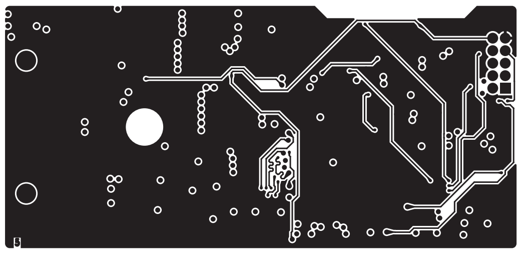


Figure B.5: Bottom buried signals layer

Figure B.6: Bottom component layer