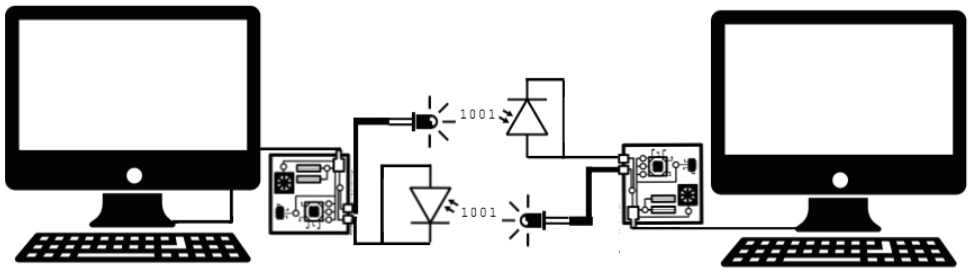
CHALMERS
UNIVERSITY OF TECHNOLOGY



An implementation of a visible light communication system based on LEDs

ADAM BÖCKER, VIKTOR EKLIND
DANIEL HANSSON, PHILIP HOLGERSSON
JAKOB NOLKRANTZ & ALBIN SEVERINSON

Department of Signals and Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2015

Bachelor thesis 2015

An implementation of a Visible Light Communication system based on LEDs

Adam Böcker, Viktor Eklind
Daniel Hansson, Philip Holgersson
Jakob Nolkrantz & Albin Severinson

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Signals and Systems
Division of Communication Systems
Chalmers University of Technology
Gothenburg, Sweden 2015

An implementation of a visible light communication system based on LEDs
Adam Böcker, Viktor Eklind, Daniel Hansson
Philip Holgersson, Jakob Nolkrantz & Albin Severinson

Supervisor: Cristian Bogdan Czegledi
Examiner: Erik Agrell

Bachelor Thesis SSYX02-15-09
Department of Signals and Systems
Division of Communication Systems
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Sketch of a visible light communication system. Computer created by Patrick Morrison and chip created by Michael Thompson from the Noun Project.

Typeset in LaTeX
Printed by Chalmers University of Technology
Gothenburg, Sweden 2015

An implementation of a visible light communication system based on LEDs
Adam Böcker, Viktor Eklind, Daniel Hansson
Philip Holgersson, Jakob Nolkrantz & Albin Severinson
Department of Signals and Systems
Chalmers University of Technology

Abstract

Wireless communication using visible light is an exciting prospect that, for various reasons, has never become popular. However, this branch of communication has several advantages, such as the fact that the infrastructure already exists and it can offer the extra security that traditional wireless communication lack of. All that needs to be done is to replace the current light bulbs with intelligent and efficient bulbs capable of data transmission.

This project explores visible light communication through a prototype implementation. The implementation consists of a Linux network driver running on a single-board computer, using a light-emitting diode for transmission and a photodiode for reception. Further details on the implementation and design are presented in this thesis.

The implementation is capable of full duplex communication at a bit rate of roughly 25 kbit/s, and is capable of compensating for errors that occur during transmission to achieve reliable communication.

In closing, the thesis presents suggestions for further work to increase system performance and capabilities, speculates on the limitations of the implementation and compares it to other methods of wireless communication.

Keywords: VLC, communication, free-space, wireless

Sammanfattning

Trådlös kommunikation över synligt ljus är en spännande möjlighet som av olika anledningar aldrig blivit populär. Detta trots att denna typ av kommunikation har flera fördelar, som att infrastrukturen redan är uppbyggd och att det finns möjlighet till extra säkerhet utöver den som finns i traditionell trådlös kommunikation. Allt som behövs är att byta ut de lampor som används idag mot intelligenta och effektiva lampor som kan hantera kommunikation.

Det här projektet utforskar trådlös kommunikation över synligt ljus genom en prototypimplementation. Implementationen består av en nätverksmodul för Linux som körs på en enkortsdator, med en sändare bestående av en lysdiod och en fotodiod som mottagare. Denna tes presenterar implementationen och dess design.

Implementationen klarar av full duplex kommunikation med en bit rate på 25 kbit/s, och klarar av att kompensera för eventuella problem under överföring för att uppnå pålitlig kommunikation.

Avslutningsvis presenterar tesen förslag på framtida utveckling för att öka hastigheten och systemet funktioner, spekulerar över implementations begränsningar och jämför med andra metoder för trådlös kommunikation.

Nyckelord: VLC, kommunikation, trådlös

Acknowledgements

This is a bachelor project thesis at Chalmers University of Technology, and we primarily want to thank our supervisors Cristian B. Czegledi and Erik Agrell. We would also like to thank ETA, Elektrosektionens teletekniska avdelning, for letting us use their instruments and lab. Finally, we want to thank the Xenomai, OpenVLC and Linux projects for their work on open source software that have made this project possible.

Adam Böcker,
Viktor Eklind,
Daniel Hansson,
Philip Holgersson,
Jakob Nolkrantz &
Albin Severinson
Gothenburg, May 2015

Glossary

ADC Analog-to-Digital Converter.

AGC Automatic Gain Control.

BBB BeagleBone Black.

CAD Computer-Aided Design.

CPU Central Processing Unit.

CRC Cyclic Redundancy Check.

FSM Finite State Machine.

GPIO General Purpose Input Output.

LED Light Emitting Diode.

MOSFET Metal Oxide Semiconductor Field Effect Transistor.

OOK On-Off Keying.

OP Operational Amplifier.

PAM Pulse Amplitude Modulation.

PCB Printed Circuit Board.

SBC Single Board Computer.

TCS-ADC-SS Touch Screen Controller and Analog-to-Digital Subsystem.

VLC Visible Light Communication.

Contents

List of Figures

1
Introduction

Visible light communication (VLC) is an exciting prospect, with a long historical background, but has never become popular for various reasons. However, currently interest for this kind of communication is increasing, and the technology for making it possible is constantly becoming more easily available.

The main technological development that made VLC possible is cheap, high-powered light-emitting diodes (LED) of high quality, capable of switching at high frequencies. Furthermore, the infrastructure for VLC is already available. All that needs to be done is to exchange the already deployed light bulbs with intelligent and efficient LED bulbs. One interesting application is vehicle-to-vehicle communication, where they communicate with each other through their head- and/or tail lights. When the vehicle in front suddenly brakes, it can communicate this to the vehicle behind and a potential multiple-vehicle accident is avoided.

1.1 Background

This chapter briefly details the background of communication using visible light, and communication systems in general.

1.1.1 History

The history of wireless communication based on electromagnetic waves dates back to the Photophone, invented by Alexander Graham Bell in the late 19th century. It was the first device used to communicate without wires and Bell's invention used the light produced by the sun to carry the information [1]. Although this was a great achievement at the time, the Photophone was never a big hit and had to give way to other wireless communication systems based on lower frequency waves. During the 20th century, almost all data sent through the air was carried by waves with frequencies lower than those of the visible light. However, today light is a common carrier of data, such as in fiber optical networks. A not so common way to transfer data is to, just as Bell did, send the information through the air using visible light. Unlike Bell, we now have more sophisticated technology, like the light-emitting diode and the transistor, to make this work in a significantly more efficient way.

1.1.2 Physics

Any kind of data can be broken down to single bits of *ones* and *zeros*, which in turn can be represented as a low or high signal. Using a light source this can be achieved by turning the light on and off, this is called on-off keying (OOK). More advanced methods exist to increase speed and stability, which are described in Chapter 2. When producing a light beam, a key aspect is how quickly the light source can move between the on and off state. This is crucial for the data transfer rate because it limits the achievable data rates. An LED is ideal for this purpose since it has a short rise- and fall-time, thus the rate at which it can switch between on and off is high. The typical rise and fall time for a communication specific LED has is a few nanoseconds. Since the LED needs a certain time to reach a stable value it limits the highest possible data transfer rate. The theoretical limit is somewhere below 1 GHz, that is <1 Gbit of data/second using OOK [2].

1.1.3 Visible Light Communication

Light has indeed been used for some time to transfer data. An everyday example is the infrared (invisible) light in remote controllers, used only to send a short control signal. When data transfer is the main intention, the transmitting frequency must be very high, hence visible light will be perceived as a continuous light rather than an irritating flicker. Data transfer with light, VLC, uses the same principles as the well established fiber optical technology, but for wireless transmission.

The most obvious property of light is that it can not move through opaque objects (e.g., walls and floors) thus the communicating devices must be in line-of-sight, or at least be located in the same room. This property may appeal in a situation where the flexibility of wireless is needed, but the security of traditional radio wave communication is insufficient. The distance data can be sent is limited by the intensity of the light. For short distances through air (e.g., within a building) the intensity can be approximated as

$$I = \frac{P}{4 \cdot \pi \cdot r^2}, \tag{1.1}$$

where P is the power emitted by the light source and r is the distance to the source [3]. Possible applications of this technology could be data transfer in environments sensitive to radio waves, such as airplanes and hospitals. Furthermore, vehicles could communicate with other vehicles using their head and/or tail lights, aiding the current development of autonomous vehicles.

1.1.4 Systems Model

In a communication system of this kind, the way the data is handled has to be structured in a certain way. One way to achieve this is to use the TCP/IP model. TCP/IP is the most common model for communication systems, such as the Internet. The model can be described in terms of four levels, where each level describes how the data is represented in its way from one point to another. In Figure 1.1 [4], the

TCP/IP layers in the case of an Internet connection is shown. The uppermost layer can, for example, be the Internet browser and the bottom layer, also know as the physical layer, could be the computer's Ethernet connection, or in this case, the VLC connection [5].

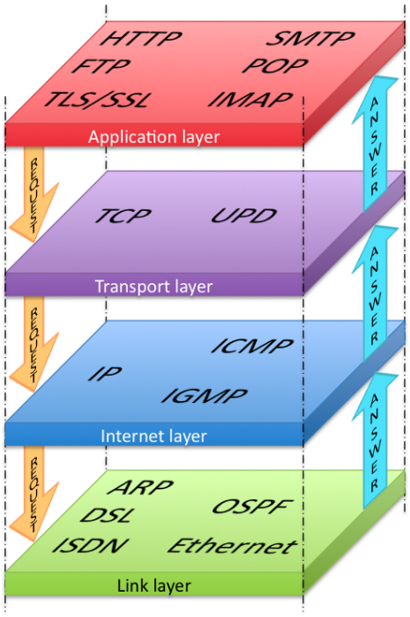


Figure 1.1: The layers of the TCP/IP system model, lower levels are closer to the physical world.

1.1.5 Hardware

The typical hardware used to construct a VLC system includes a wavelength specific LED and photodiode. The photodiode is doped to enhance its capability to turn light into a current, usually constructed as a so called P-I-N diode [6]. Besides the component responsible of generating and capturing the light signals, additional hardware is needed to filter and interpret these signals. For example, a suitable band pass filter can be used to filter out incoming signals with frequencies other than those desired; an optical filter can be a good way to filter out light of other wavelengths; a lens to focus incoming light can be suitable if the light intensity is low. In the case of digital data transmissions, a digital signal processor at both ends of the system is required to process incoming and outgoing data.

1.1.6 Software

A good software implementation is essential to achieve an efficient communication system. At the transmitting end, the data that will be sent is structured according to a chosen communication protocol, and at the receiving end the same protocol must be used to correctly interpret the incoming data. Some form of two-way communication is necessary to guarantee a loss-less transmission, in which the receiving

end performs a check to verify that all data was obtained correctly and then informs the transmitter whether it has to re-send the data packet or not.

1.2 Purpose and Scope

The purpose of this project is to perform digital, point-to-point, communication over visible light and to illuminate the area where the transmission is intended. The transmission must be stable enough to allow for reliable communication and must be capable of compensating for transmission errors. The distance and rate at which data can be sent is also parameters that will be taken into account since these limits its applications. Furthermore, the project must be easily extensible to allow for further development.

1.3 Task

This section presents the project tasks and required specifications.

1.3.1 Main Task

The main task of this project is to develop and build a pair of devices capable of sending respectively receiving data over visible light. This can be done using any light source emitting visible light that is capable of switching quickly. This project will focus on the use of LEDs for transmission.

In order for the system to be useful, it must be capable of performing a continuous transfer of a 10 MB file. The transmission speed is interesting for comparison with other technologies, but speed is not a goal in this project.

1.3.2 Subtasks

The task consists of two major parts. The first part is to design and build a transmitter and a receiver. The second part is to implement software to control these devices. Figure 1.2 shows a more detailed structure of the task.

1.3.3 Primary Specifications

The system must be able to:

- Send arbitrary data over visible light.
- Receive arbitrary data over visible light.
- Transfer data while producing 450 lumen (equivalent to a 40 W light bulb).
- Send data over at least 1 meter in a brightly lit room.
- Send data in one direction.
- Perform an uninterrupted transfer of 10 MB.

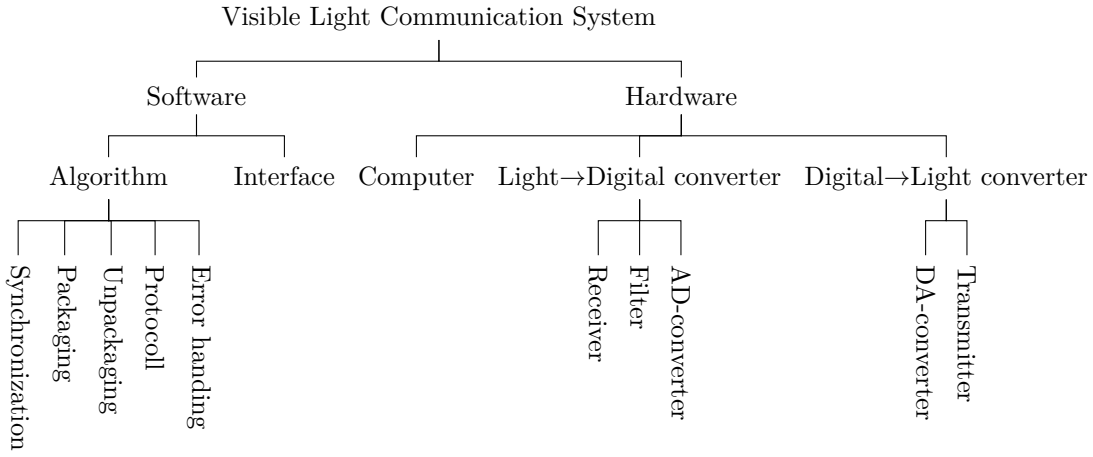


Figure 1.2: Task divided into subtasks.

1.3.4 Desired Specifications

The desired specifications, in order of priority are as follows:

1. The device can connect to any computer over Ethernet.
2. The device can manage half-duplex transfer.
3. The device can perform reliable and loss-less data transmission.
4. The device is packed in a case.
5. The device should manage to send data over a distance of 2 meters in a brightly lit room.
6. The device can manage full-duplex transfer.
7. The device supports light intensity variation during transfer (dimming the light source).
8. The device can connect to multiple units simultaneously, similar to a WiFi router.
9. The device should support multiple data steams in parallel (multiplexing).

1.3.5 Challenges

This project includes many challenges. For example:

- Disturbances affecting the transfer. For example luminaries and the sun.
- Variations in transmission distance.
- Detecting and handling errors.
- Hardware limitations such as LED rise time.
- Timing between the transmitter and the receiver.

1.4 End Product

The end product consists of a communication system, with a transmitter and receiver capable of transferring data using visible light. The device consists of two single

board computers (SBC) with additional hardware for the sender respectively the receiver. The device can communicate by its own, or can be configured to connect to other devices over Ethernet, enabling the device to communicate over visible light. The device simultaneously functions as a luminary, both when idle and when transmitting, and can be embedded into a light fixture.

1.5 Limitations

Limitations of the project scope are set such that as little time as possible will be spent on activities not concerned with VLC. This includes limitations on the software and hardware implementation.

This project will be limited to developing the link layer for transmitting data over visible light. Existing, open source, code will be used for all other layers. On the hardware side, the project will be limited to using an SBC with an already working operating system as development platform. The light used for transmission must be safe to look at, and must not be annoying to the human eye.

2
Methods

The methods chapter presents how the VLC system was developed and implemented. It includes both a software implementation, from now on referred to as the VLC driver, and a hardware transmitter-receiver implementation. Firstly, a high level overview of the developed communication system is presented, after which the hardware and software implementation is presented in turn.

2.1 Overview

This section will briefly present the basics of the communication system used to transmit data over light. A high-level overview of the communication system implemented in this project can be seen in Figure 2.1, where the devices intended to communicate can be connected with the SBCs.

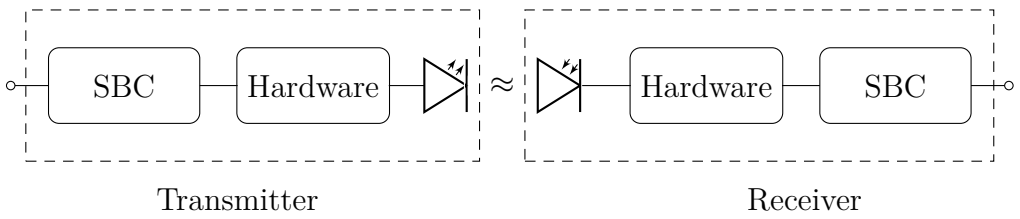


Figure 2.1: High-level overview of the communication system implemented.

2.1.1 Development platform

During the project a Beaglebone Black (BBB) was used as development platform. It is a small, inexpensive and relatively powerful single board computer. For a complete description of the development platform, see [7].

The BBB is a single-board computer, which means it can run a variety of Linux distributions, such as Debian, Ubuntu and Ångström. That means the project can use the functionality already present in the operating system, such as the network stack and TCP/IP protocol suite. Furthermore, the operating system makes it easy to run several execution threads in parallel. These reasons should make development easy, and is the reason why an SBC was chosen over a micro controller.

The BBB in particular was chosen over other SBCs due to its diverse set of features. For example, the BBB has both a programmable real-time subsystem and an analog-to-digital subsystem which can be used to synchronize software and read analog inputs respectively.

2.2 Hardware

The physical layer is the hardware used between the two BBBs, i.e., all the electrical components. The hardware was implemented using LEDs transmitting data via light to a photodiode which converts it back to an electrical signal so that the BBB can interpret it as data. To process the electrical signal correctly the transmitter and receiver have to be designed and optimized for this purpose. All the separate parts of the hardware are explained in detail in their separate subsections.

A complete component list is available in Appendix A with references to the components' datasheets.

All active components need a power supply to work properly. In this project 5 V was used due to it being the maximum voltage the BBB can deliver.

2.2.1 Transmitter

The transmitters task is to convert digital data into visible light. An LED was a suitable component because of its relatively linear relation between current and light intensity [8]. The general idea was to modulate the light intensity of the LED i.e., the intensity of the light corresponds to the symbol transmitted. The BBB ports are not capable of delivering the right amount of current to make the light intensity strong and fast enough. To get around this problem a transistor was used as a switch, which made it possible to switch a larger current faster. In Figure 2.2 the general design is shown to give an overview of the transmitter.

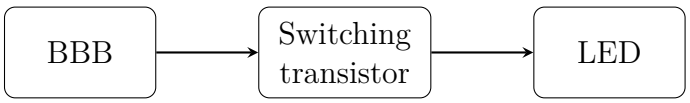


Figure 2.2: Transmitter's general design.

Transistor

The BBB can only convert data into a voltage and therefore a voltage operated transistor was needed. The Metal Oxide Semiconductor Field Effect Transistor (MOSFET) was suitable for this application because it operates using a positive voltage on the gate. The MOSFET can be seen in Figure 2.3.

When a voltage is applied to the gate of the transistor, it generates an electrical field which lowers the internal resistance to increase the current from drain to source. Due to the high input resistance, the MOSFET can handle high currents with almost no current on the gate. This makes it possible for almost any driver to handle the MOSFET, including the BBB. This means that a BBB can be used to operate the MOSFET for switching an LED. To increase the current when applying a voltage to the gate, an enhancement MOSFET was chosen over a depletion MOSFET, otherwise the signal would be inverted. The MOSFET also have a low drain-source resistance which makes it good for switching since it acts like a short circuit when fully on [9].

The MOSFET used is an N-channel named *IRLR3715ZPBF*. It is a switching power transistor and was chosen for the capability of switching high current. With a capability of switching 49 A at a gate voltage of 10 V at 25°, the transistor makes no limit for further development with more LEDs. The transistor is also designed with a ultra low gate resistance, which makes it optimal to use with a driver like the BBB.

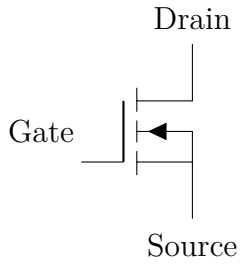


Figure 2.3: MOSFET symbol with gate, drain and source.

LED

An LED is a semiconductor that produces light. When electrons enter the semiconductor they bond with holes in the substrate and energy is released in the form of photons. There are several variables that need to be considered when choosing an LED and these variables have to be weighted against each other. The maximum intensity of the LED affects the rise time. A low intensity makes it possible to have a short rise time and increasing the intensity increases the rise time. The more intensity an LED generates, the more power it needs and the more heat it produces. The BBB has a maximum voltage output of 5 V, maximum current output of 1 A and can send a square wave with the maximum frequency of 50 kHz. To meet these criteria the *CREE MCE4WT-A2-0000-000HE7* was chosen. This is a white LED made up of four separate LEDs with an intensity of 240 lm each. The LEDs have a forward voltage from 3.1 V to 3.9V. The LED has its peak wavelength at 450 nm and 610 nm. Manufacturers of LEDs do not specify switching characteristics in their datasheets and therefore the diodes had to be tested in order to check its function with the frequencies that were used. The chosen LED was tested and can perform well at frequencies over 25 kHz.

Final Design

The transmitter was designed after the specification of the BBB, in other words making use of the limited power output. The transmitter was made using three transistors, as can be seen in Figure 2.4. These three independent transistors are each controlled using the GPIO ports on the BBB, and thus it is possible to regulate the three transistors independently. This means that the current through the LED, and thus the intensity, can be switched in eight steps. Having multiple steps makes it possible to use different encoding styles, like Pulse Amplitude Modulation (PAM) or OOK. Flickering can be avoided by having one transistor constantly open and cycling the other two on and off. This also made it possible to use regular resistors with a maximum tolerant power rating of 0.25 W. Two ripple capacitors, one large of 100 μF, and one small of 100 nF were used to remove ripple from the power source. Firstly, a prototype board were built for testing the design of the circuit, and then a single layer PCB was made using the Eagle CAD software. The prototype and the PCB can be seen in Appendix B.

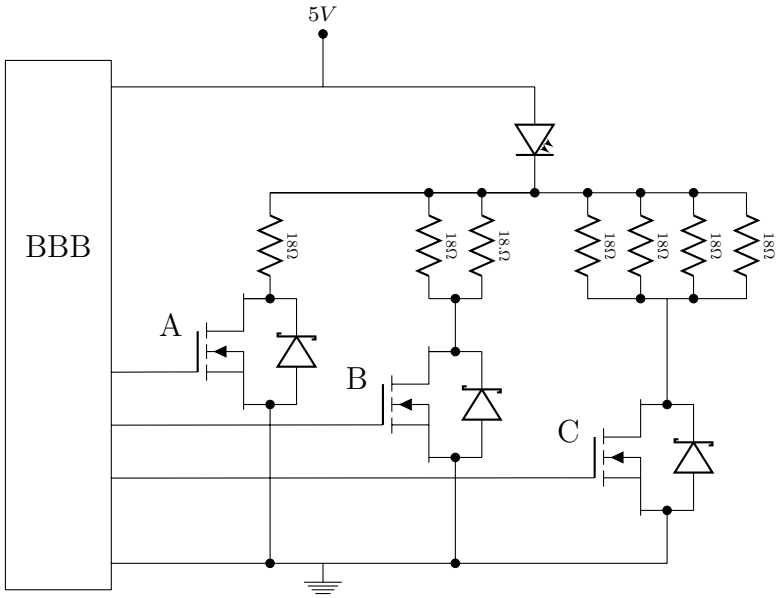


Figure 2.4: Schematics of the transmitter.

DC characteristics

Electrical components have power ratings that cannot be exceeded without destroying the component. Therefore this has to be accounted for when using high currents and voltages. In Table 2.1 the power dissipation was calculated by measuring the current and voltage through the components, with the LED constantly on. It can be noted from the table that the resistors are close to their maximum power rating of 0.25 W. However, since Manchester encoding is used, the resistors will only have an applied voltage half of the time and thus the average power is lower.

Table 2.1: The transmitters voltage, current and power dissipation.

Components	U[V]	I [A]	P [W]
Transistor A	2 m	112 m	224 μ
Transistor B	3.2 m	224 m	717 μ
Transistor C	6.8	448 m	3046 μ
Resistors	2.1	112 m	0.224
LED	2.686	784 m	2.1

2.2.2 Receiver

The receiver converts the incoming light into current using a photodiode. A photodiode is a semiconductor converting light into an electrical current. This electrical current needs to be converted into a voltage such that the BBB can process it while not exceeding the BBB's restrictions. For a digital signal the BBB cannot receive a voltage above 3.3 V. Therefore, the electrical circuit between the photodiode and the BBB needs to process the electrical signal so it can be interpreted correctly.

The receiver's electronics need to convert the current to voltage in order to amplify and filter it. To be able to vary the distance between the transmitter and the receiver without risking getting a too small signal or a too high signal, an automatic gain controller (AGC) was designed. This component amplifies or reduces the input voltage to a selected output voltage. To make sure the signal is digital and stable before the BBB, a transistor was used as an analog-to-digital converter (ADC). The general design is shown in Figure 2.5 and the final design is shown in Figure 2.11.

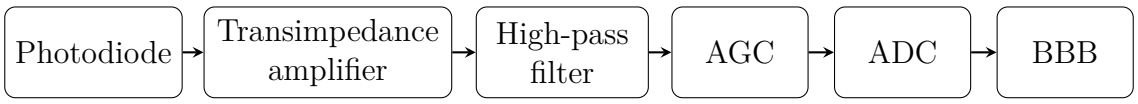


Figure 2.5: Receiver's general design.

Photodiode

As mentioned before, the photodiode is a semiconductor converting light into an electrical current. Most of the photodiodes on the market are produced for the purpose of fiber optics. In applications concerning fiber optics, the radiant sensitive area is small and the rise respectively fall time is short. With increased radiant sensitive area, the component's response time will be slower. Without fiber optics a larger radiant sensitive area allows for more light to be captured by the receiver. Therefore, the choice of photodiode is limited.

The requirements of the photodiode was a quick response time, a spectral sensitivity in the visible spectrum and a large radiant sensitive area. The size of the radiant sensitive area is crucial and therefore the photodiode used was a *VISHAY*

BPW21R. It has a suitable wavelength peak sensitivity at 565 nm. The spectral bandwidth is from 420 nm to 675 nm and gives a perfect range for the intended application. It has a linear light intensity to current ratio and the radiant sensitive area is 7.5 mm^2, which was larger than most photodiodes found. It has a rise and fall time of 3 μs each, which provides a switching frequency of 166 kHz. This was enough since it is above the capabilities of the rest of the hardware and software.

The photodiode converts all the light hitting the radiant sensitive area into a current proportional to the intensity of the light, including all other light sources. The surrounding light sources flicker in different frequencies depending on, e.g., the power grid or internal hardware. These light sources produce noise in the form of electrical current in the receiver. To maintain a good, stable signal the noise needs to be removed. Therefore a filter is crucial and the filter implemented is described the high-pass filter section below.

To amplify the current from the photodiode, a reverse voltage can be applied. But with a large reverse voltage the reversed leakage current, also know as dark current, increases and creates unwanted noise. This gives a trade-off. The maximum reverse voltage of the VISHAY BPW21R is -10 V, but in this application -5 V was used. This makes it possible to use the power supplies of the BBB and gives a lower dark current and enough amplification.

Transimpedance amplifier

Instead of using a single resistor to convert current to voltage, a transimpedance amplifier was used. The transimpedance amplifier is a current-to-voltage converter and is commonly used with photodiodes [10]. It is suitable because it increases both gain and speed, which are needed in this application.

A photodiode has an internal capacitance C, and an internal resistance R. Together they make the time constant $\tau = RC$, and if the resistance is large then the gain can be large, but the response becomes slow. To increase the speed a smaller resistor is needed, but then the gain will be low. To avoid this phenomenon, the photodiode was connected directly to the transimpedance amplifier, making the time-constant no longer be determined by R or C, and the gain can be determined by the feedback resistor R_f, seen in Figure 2.6.

The transimpedance amplifier is an operational amplifier (OP) with a feedback resistor. Because the transimpedance amplifier has a closed-loop amplifier, oscillations can occur if the phase margin is not sufficient. To solve this, phase compensation can be done with the feedback capacitor C_f, added parallel to the feedback resistor R_f, seen in Figure 2.6. This capacitor needed to be optimized for the specific circuit to not overcompensate. Even if a small overcompensation often is preferable, a large overcompensation would make the signal oscillate again.

The OP chosen for this circuit was *UA741CD* from TEXAS INSTRUMENTS. It is a general purpose OP with a bandwidth of 1 MHz. It is a cheap, commonly

used and is therefore suitable for this project. The feedback resistor was chosen to 470 kΩ to produce a suitable amplification. To find a suitable feedback capacitor measurements were conducted until oscillation decreased, $C_f = 5.6$ pF.

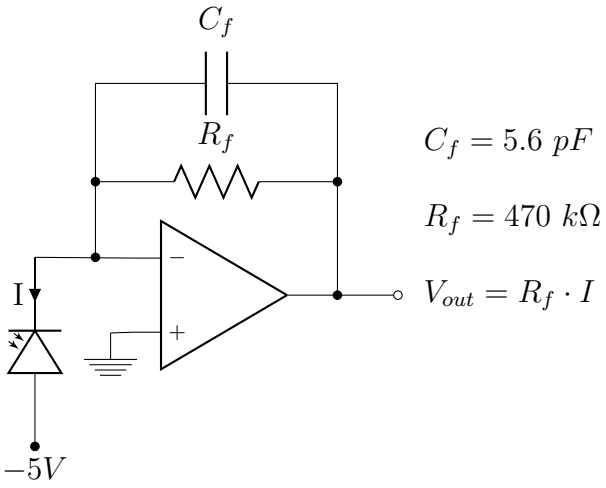


Figure 2.6: Transimpedance amplifier.

High-pass filter

To remove unwanted noise from surrounding light sources, e.g., the sun or luminaries, a high-pass filter was included in the design. A passive first order analog high-pass filter was used for the simplicity of the design, realized using an RC circuit as shown in Figure 2.7. A passive instead of an active filter was chosen because both the transimpedance amplifier and the AGC include gains, and more gain was not needed.

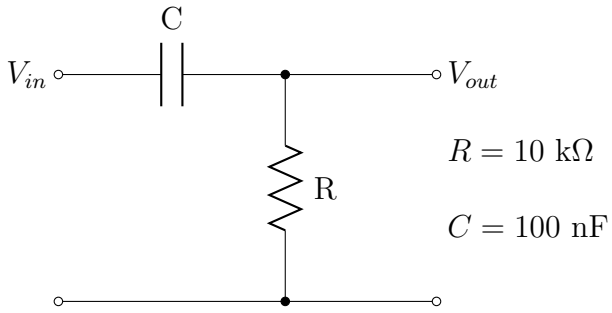


Figure 2.7: A passive, analog, first order high-pass filter.

The surrounding light sources have a relatively low frequency compared to the frequency used by the transmitter. To avoid distortion of the sent data, the filter was designed with a relatively low cut-off frequency. Otherwise the square wave would be distorted and hard to interpret. The chosen values for the resistance was 10 kΩ and the capacitance was chosen to 100 nF. The 3 dB cut-off frequency is

$$f_c = \frac{1}{2\pi RC} = \frac{1}{2\pi \cdot 10 \cdot 10^3 \cdot 100 \cdot 10^{-9}} = 159 \text{ Hz}. \tag{2.1}$$

The chosen filter gives the transfer function $G(s) = \frac{s}{s+1000}$ and in Figure 2.8 the transfer functions is plotted in a Bode diagram. The figure describes the filters impact on the signal depending on its frequency. When a sinusoidal signal reaches over 1 kHz the signal is nearly untouched. The carrier frequency used by the system is 25 kHz and theoretically a square wave includes unlimited frequencies. This means that the square wave will be affected by the filter and a trade-off where a part of the square wave will be attenuated in the favor of removing noise.

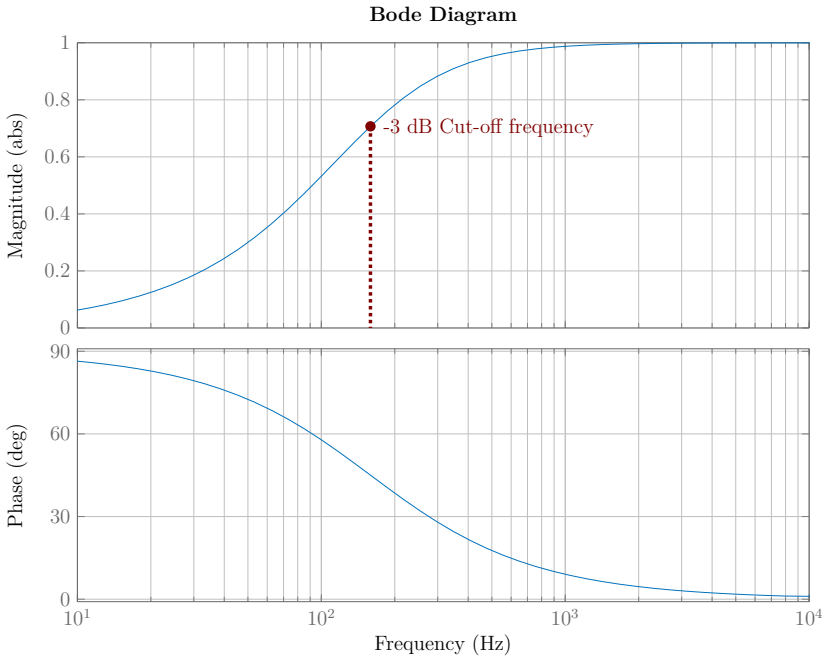


Figure 2.8: Transfer function for the high-pass filter.

Automatic gain control

An AGC was used to accommodate for the variations in distance between the transmitter and receiver, without changing the receiving signals amplitude. The AGC amplifies the varying signal to a defined value based on the gain factor created by the feedback circuit. In this case the only AGC available was the *TL026C* and with a bandwidth of 50 MHz and a peak gain of 38 dB it was suitable for this application. The circuit design is from the AGC's datasheet and can be seen in Figure 2.9. The same values as in the datasheet was used since it provides an adequate amplification. This could also be done in software, but due to the limited processing power in the BBB it would slow down the transmission significantly.

Analog-to-digital converter

To make sure it was a digital signal with the right voltage at the input of the BBB, the AGC's output was followed by a transistor which acts as an AGC, which can be seen in Figure 2.10. Firstly a 5 V supply was voltage divided down to 2.5 V, which was a suitable voltage for the BBB without any risk of breaking it. When the AGC

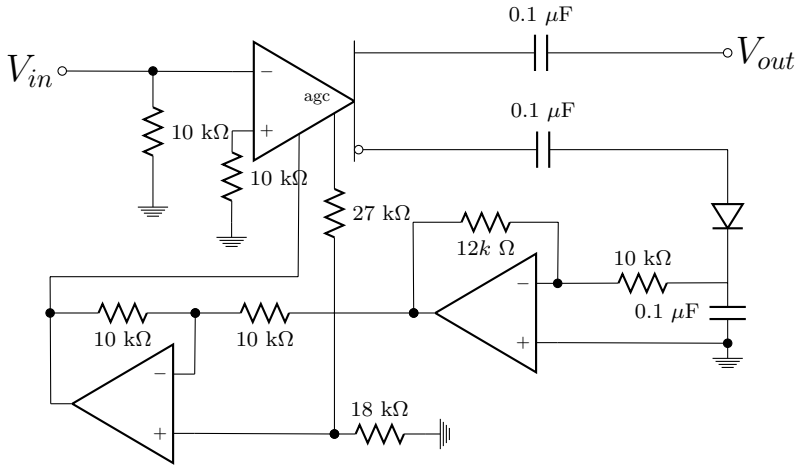


Figure 2.9: Schematic of the automatic gain control.

gives a high voltage the transistor is closed and the potential at the BBB is 2.5 V, a logic *one*. When the AGC gives a low voltage the transistor is opened and the BBB will be grounded and sample a logic *zero*. The diode was used to remove the negative voltage in the signal from the AGC. This design does not work with the encoding style PAM and needs to be changed if PAM is intended to be used. The transistor used was a bipolar NPN *BC547A*. The value of the resistors was chosen to 10 $k\Omega$, this minimizes the current and thus the power dissipation.

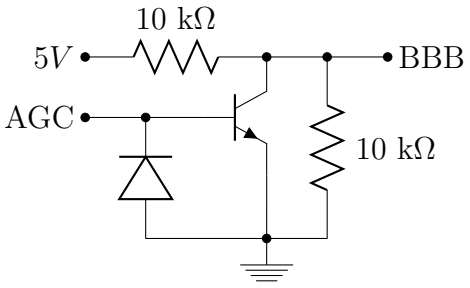


Figure 2.10: Voltage division over a transistor acting as an AGC.

Final design

The schematics of the receiver with all parts can be seen in Figure 2.11. The prototype product of the receiver can be seen in Appendix B, Figure B.4.

2.2.3 Design verification

Tests were conducted to verify the design and see if the right properties were achieved. These tests were mainly done using an oscilloscope and a function generator. Sending square waves trough the systems made it easy to look at different parts of the circuit and see how the signal propagates trough the components. The final design verification was done using the complete system, transmitter, receiver

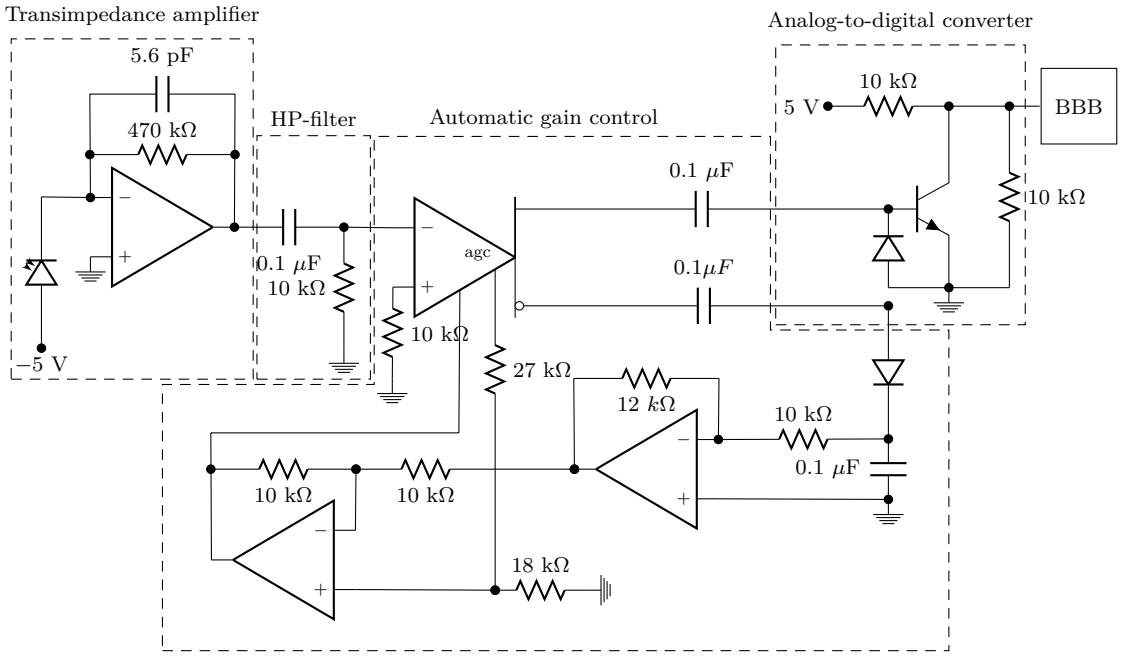


Figure 2.11: The schematics of the receiver.

and software, and the results are shown in chapter 4, Results. The instruments used can be found in Appendix A.

2.3 Software

The software implementation of the project consists of the following separate parts:

- Data packaging
- Hardware control
- Transmission synchronization
- Transmission encoding and decoding
- Error handling

Implementation Framework
The VLC driver is implemented as a Linux network module and it has to be taken into account during all parts of the software implementation.

The reason for choosing to implement it in this way is that it allows any application, that can be run on a Linux computer, to transmit data over visible light. This is because the VLC driver will integrate seamlessly with the Linux kernel, and will be displayed as a network card in the operating system which makes the implementation much more useful. Furthermore, the existing TCP/IP protocol suite will handle address resolution and transport protocols.

The downside is some loss in flexibility. The driver has to conform to the specifications of Linux modules, which means some functionality can not be used. However,

this is a small problem in comparison to the advantages.

Network Modules
The network module works by providing an interface to the operating system kernel through a set of specific functions. The kernel can then use those functions to interchange data with the module. For example, whenever a user wants to send data to another user, the following steps will take place:

1. The application used by the user will notify the kernel that it wants to send data, what the data is, and where it should be sent.
2. The kernel will take the data and hand it to the network module.
3. The network module will buffer it and start transmitting it over the channel.
4. The network module on the receiving computer will collect the packet from the channel, and hand it to the kernel of the receiving computer.
5. If the data is intended for an application, such as a web browser, the data will be handed to that application.

2.3.1 Design

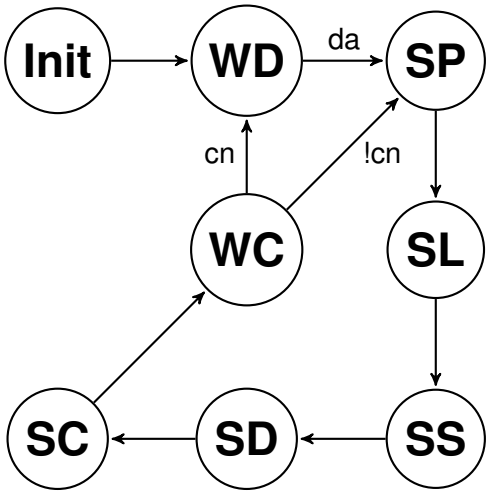
Transmission in a communication system is performed by encoding the data to be sent into a sequence of symbols well-suited for transmission. For example, the data can be viewed as a long sequence of logical *ones* and *zeroes*, and transmission is made by turning the LED on and off to send a *one* and a *zero* respectively. However, to achieve reliable and efficient communication the data has to be packaged in some way described in Section 2.3.2. This section describes how the software was designed to send the packets in the VLC driver.

The transmission was made by representing the data as a sequence of bits. Before the file is sent, the file length[1] and the sequence number[2] have to be sent. However, the whole sequence is preceded by a preamble[3]. The software was designed as finite state machines (FSM) described bellow.

Transmitter side

After the transmitter is initialized it is set to wait for data. When data is to be sent the transmission is started by sending a preamble in order to synchronize the receiver. This is followed by the packet length and the sequence number. Then the data followed by the checksum is transmitted. In the last state the transmitter is waiting for the receiver side to send a confirmation. If the transmission was successful the transmitter side returns to wait for data, otherwise the packet is retransmitted. This is displayed in Figure 2.12.

[1] This was needed for the receiver to read the correct number of bits.
[2] This was needed to put the packets back together to a file at the receiver side.
[3] A predetermined sequence of ones and zeros that was transmitted before every packet to *wake-up* the receiver.

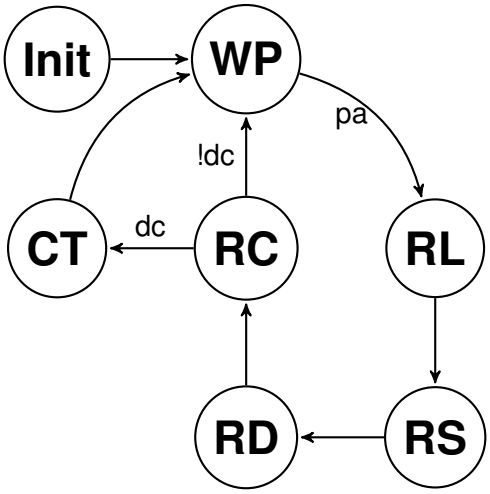


	Meaning
WD	Wait for Data
SP	Send Preamble
SL	Send Length
SS	Send Sequence number
SD	Send Data
SC	Send CRC
WC	Wait for Confirmation
cn	Confirmation
!cn	Confirmation Timeout
da	Data

Figure 2.12: FSM chart describing the transmitter software

Receiver side

The receiver is in standby[4] until it detects the preamble. When it is detected the data length and the sequence number is read and then the data transmission begins followed by the CRC. If the transmission is successful the receiver side sends a confirmation signal to the transmitter side. This is displayed in Figure 2.13.



	Meaning
WP	Wait for Preamble
RL	Receive Length
RS	Receive Sequence number
RD	Receive Data
RC	Receive CRC
CT	Confirm Transfer
pa	preamble
dc	Data Correct
!dc	Data Corrupted

Figure 2.13: FSM chart describing the receiver software

2.3.2 Packaging

Digital data transmissions works by sending data in packets. For example, when sending a file between two computers, the file would be split into packets of smaller size that are transmitted. The receiving side will collect all packets, and once the transmission is completed, the packets will be combined into the original file again.

[4]Doing nothing but listen for the preamble.

If any one packet is corrupted, only that packet will have to be retransmitted instead of having to retransmit the entire file. This section describes how this packaging is performed in the VLC driver.

Linux Kernel

Whenever an application wants to send data over the network, it will notify the Linux Kernel. The transport and the IP-layer of the communication stack is contained in the Kernel and the Kernel will split the data into packets. The Kernel will then notify the network driver that there are packets ready for transmission and will hand over the packets to the driver in sequence.

The packets, as they are presented to the driver, contain the following sections that are used by the VLC driver:

- Data payload
- Data payload length

VLC Driver

Whenever the VLC driver receives a packet from the kernel, the driver will store the packet in a buffer and will add the following to the packet:

- Preamble
- Sequence number
- Data checksum

The end result is a packet that looks as in Fig. 2.14. Further details on the different parts of the packet are presented below.

32-bits	32-bits	32-bits	0:12 000-bits	32-bits
Preamble	Length	Sequence	Data	Checksum

Figure 2.14: The packet layout.

Preamble

To notify the receiver that a packet is about to be sent that needs to be collected, the transmitter will send a preamble. The preamble is a predetermined sequence of bits that the receiver is looking for continuously while not already receiving a packet.

The VLC driver uses a preamble that consists of two square waves with different period in sequence, and looks as follows: 00110011001100110011001100111010. Since the receiver and transmitter might not be synchronized when the preamble is sent, the preamble is not Manchester encoded.

Length

The number of bytes contained in the data payload of the packet is sent to inform the receiver on how long to listen. The receiver will store the payload length and will collect octets of bits until it has collected the amount of bytes specified by the length.

Sequence

All packets sent are numbered, and the sequence number is transmitted along with the packet. This handles duplicate transmissions of packets. If, for example, the acknowledgement is corrupted due to noise, the transmitter will retransmit that packet despite it being received correctly. Having the sequence number available means the receiver knows it is a duplicate and can take care of it accordingly.

Data Payload

The actual data intended for the receiver. This data is handed to the VLC driver from the Linux kernel, and can vary in size between 0 and 12 000 bits [11]. This is the only part of the packet that is not overhead.

Checksum

The checksum of the packet is sent last and is used in error detection and handling. See the section on error handling for further details.

2.3.3 Hardware Interface

This section describes how the VLC driver connects to the hardware of the sender and receiver.

GPIO

General purpose input-output (GPIO) is a method for controlling the physical world through software. The GPIO is a set of hardware pins present on the BBB. The GPIO pins can be configured as inputs or outputs and the software can read from or write to them. For example, reading from a digital GPIO pin configured as input would return a *zero* or *one* depending on if there is a voltage present over the pin or not. Tests show that the BBB will interpret the value as *one* if the voltage is higher then 3 mV and *zero* otherwise. Writing a *one* to a GPIO pin configured as an output will set a voltage of 3.3 V over that pin, and writing a *zero* will set a voltage of 0 V.

The GPIO is connected to the CPU of the board, either directly or through some subsystem present on the board. This means that the capabilities of the GPIO vary

between processors and boards. The exact specifications of the BBB GPIO can be found in [7].

Linux GPIO Interface

Linux provides several ways of controlling the GPIO, which vary greatly in performance. The simplest form is through a set of files that are linked to the GPIO pins. Writing and reading to the files translates into reading and writing to the GPIO. However, this implementation has the additional overhead of opening and closing files for every read and write.

The more efficient way to interface with the GPIO is through memory. Since the GPIO is connected to the CPU, the pins are mapped into physical memory addresses available to the VLC driver. Reading and writing to these memory addresses is roughly 1000 times faster than writing through files and allows for a maximum toggle rate of about 2.8 MHz [12]. The only downside is slightly more complex code. The VLC driver uses this method.

Reading and Filtering

The BBB has the capabilities of reading both analog and digital values, and this is done through different sets of GPIO pins. The analog values are represented over 12 bits, which allows for a more precise filtering and gain control to be performed in the software. Reading digital values severely diminishes these capabilities since this method only provides a one-bit representation of the input value.

The BBB has several subsystems that work independently of the main CPU and that cooperate through shared memory. One of these is the touch screen controller and analog-to-digital subsystem (TCS-ADC-SS), which handles analog reading. This system contains a micro controller, and to read analog values, it must be programmed to read analog values continuously and place them in shared memory where the CPU can access them. There are differences in what functionality is available to user programs and kernel modules, and existing ways of programming the TCS-ADC-SS require functionality not available to kernel modules. For this reason reading analog values in the VLC driver would require a lot of work. Furthermore, the VLC driver would not be able to run on a different board without this subsystem. However, the pins for digital reading are connected directly to the CPU and are easy to interface with. For this reason the driver only uses digital GPIO. All filtering and gain control is instead done in hardware, which has the added benefit of offloading the CPU of the board.

2.3.4 Synchronization

Communication must be performed in a synchronized manner. For example, the following functionality is required for reliable communication:

- The receiver must listen whenever something is being sent to it but should otherwise ignore incoming noise.
- The receiver must collect bits at the same pace as they are being sent, and vice versa.
- The receiver should try to read the bit currently being sent at the half-way point between two bits being sent. That is, if one bit is sent every second, the receiver should read the bit half a second after the sender has transmitted it.

The problem consists of two parts; software real-time constraints and symbol synchronization, detailed below. The software real-time constraints are met through the use of the Xenomai kernel extension. The early-late method of symbol synchronization was implemented but not used in the final product due to poor performance. Both are detailed below.

Real-time constraints

To transmit data between two devices they must operate in a synchronized manner. The receiving device must collect bits at the same pace as they are sent. This puts, what is called, hard real-time constraints on the application.

This is a problem when using Linux since Linux is not an operating system designed to handle real-time constraints. Linux will, for example, interrupt a task if another event occurs, or a certain task might be delayed in favor of some other task. This is desireable in most cases since the overall performance increases, but if deadlines which must be met exist, as in this case, it is a problem. Another example of a system with real-time constraints is the computer handling the ABS breaks in a car. No-one would want the breaking to be delayed because the computer is busy doing some other work.

Xenomai

Real-time constraints can, however, be met in Linux by using the Xenomai framework. Xenomai is an open source project with the goal of running real-time applications on Linux. The core of Linux is the Linux kernel, which manages the computer's resources, and decides what task to run and when. Xenomai introduces another kernel into Linux, which always runs before the regular kernel. This separate Xenomai kernel handles all real-time tasks and runs the regular Linux kernel only when there is no real-time constrained task waiting. This enables applications with real-time constraints to run on Linux while not losing the flexibility and functionality that Linux offers.

The downside of this approach is that the Linux kernel runs more slowly since the Xenomai kernel takes priority. The upside is that this approach does not depend on the specific hardware platform used to run the code. Any platform with support for Xenomai can be used.

Specialized Hardware

Another way to meet real-time constraints, and the way they are most often handled, is by using specialized hardware for the timing sensitive code. This specialized hardware would only run the timing sensitive code and can therefore guarantee the real-time constraints. This could, for example, be a USB WiFi network card that plugs into a laptop.

The CPU on BBB has a built-in programmable real-time subsystem (PRUSS) which could be used for this. This is likely a high-performance solution that would work well once implemented since it frees up the Linux kernel. However, it comes with the obvious downside of making the code dependent on using this specific CPU. Furthermore, the PRUSS is complicated and implementing this solution would require a lot of work.

Symbol Synchronisation

Two separate clocks can never be perfectly synchronized. One of the clocks is always ticking slightly faster than the other. This causes a problem that the two devices could drift out of sync. This has to be compensated for in some way, either by adjusting the clocks whenever they start to drift or by detecting when an error occurs and asking the sender to retransmit whatever data that was damaged.

The early-late method of synchronization is based on taking several samples per bit and using this to detect clock drift. In this case three samples per bit was used. Ideally the three samples is centered on the currently transmitted bit. If the clocks drift apart, one of the edge samples is taken on either the next or the previous bit.

The system attempts to detect this scenario by calculating the moving average of each sample value and compare it to the combined average of all three samples. If the average of one of the edge samples deviates from the combined average, the system determines that the clocks are drifting apart and will compensate for it by adjusting its clock. The method is displayed in Figure 2.15.

Since the early-late method collects three samples, and calculates some averages to detect if one sample point is deviating from the others, the method is significantly more expensive computationally compared to receiving bits without using early-late. For this reason this method is not used in the VLC driver. It is cheaper to allow the clocks to drift apart occasionally, and then compensate for the errors. Early-late is, however implemented and can be activated in the VLC driver.

2.3.5 Encoding

Data can be represented using various methods of encoding. In a computer everything is represented as *ones* and *zeros*. When data is sent through a channel those *ones* and *zeros* must be represented in some physical way. There are many different

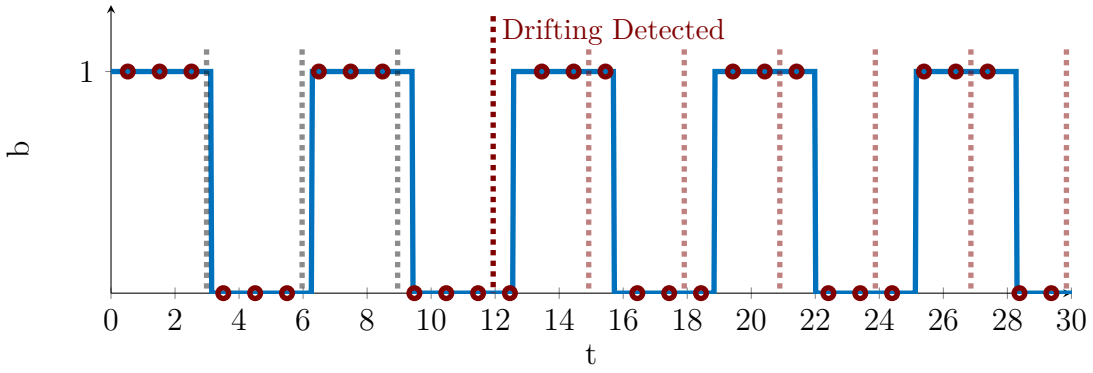


Figure 2.15: The Early Late symbol synchronization method. The groups of three samples are separated with the dotted lines. Drifting is detected when one of the edge samples differ from the other two in its group. When drifting is detected the dotted lines are marked red in the diagram.

encoding styles. This chapter briefly describes some of them, and the choice made in this project.

Encoding Styles

This section describes the most commonly used encoding styles that was discussed during this project.

On-off keying (OOK) is the simplest method to represent data. The logic value *zero* correspond to *LOW* and the logic value *one* to *HIGH*. In the VLC case, this means the LED is turned off to transmit a *zero* and turned on to transmit a *one*.

Manchester encoding is a system used in many communication systems in conjunction with OOK. This method encodes a *zero* into the sequence 01 and the *one* into the sequence 10. This has two clear advantages in a VLC systems. Firstly, even at high frequencies a long sequence of *zeros* followed by a long sequence of *ones* will be perceived as annoying flicker of the LED. Manchester encoding solves this by always sending an equal number of *ones* and *zeroes*.

Secondly, the AGC of the receiver uses the average value of the input to calculate the amplification, which is disturbed by long sequences of the same value. For example, after observing a long sequence of *zeros* the AGC will increase the amplification and thus corrupt the signal. Manchester encoding guarantees that such a sequence never occurs.

The disadvantage of Manchester encoding is that every logical bit is sent using two physical bits, and as a result the transfer speed is halved compared to OOK without Manchester encoding. How the bits are encoded is displayed in Figure 2.16, 2.17 and 2.18.

Pulse-amplitude Modulation (PAM) works in almost the same as OOK, but

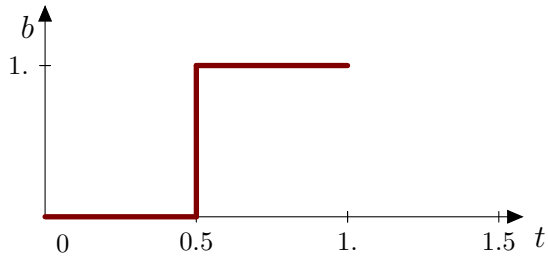


Figure 2.16: Manchester encoding. The boolean sequence representing a digital one.

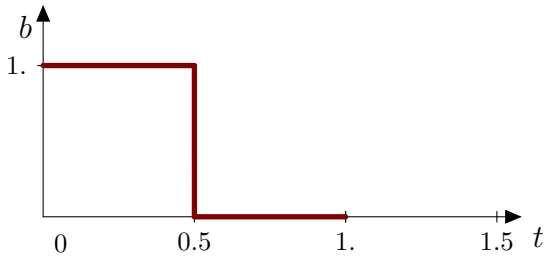


Figure 2.17: Manchester encoding. The boolean sequence representing a digital zero.

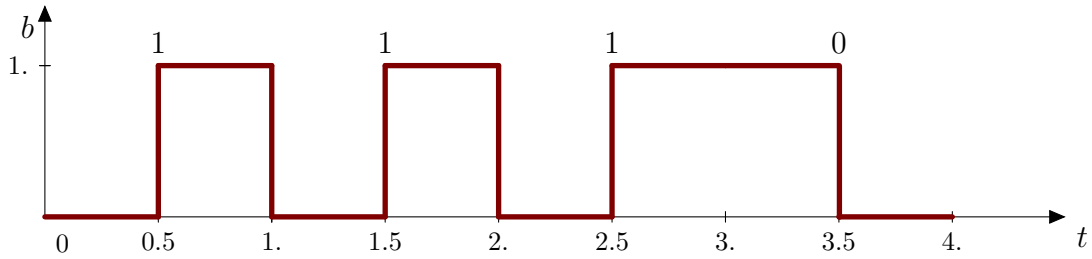


Figure 2.18: Manchester encoding. The boolean sequence representing a the digital sequence 1110.

with more levels of signal amplitude. For example a 4-level PAM has 4 possible levels, representing the values 0,1,2,3, and encodes two bits of data per level. The receiver detects the input amplitude and from that infers what data was sent.
Note that OOK is a 2-level PAM, representing the values *zero* and *one*, a one bit value. The transmission rate increases logarithmically with the number of levels. For digital transmission the number of levels should be a power of two to represent an integer sequence of bits.

Encoding in this project

The main focus in this project has been reliable communication. Manchester encoding was chosen since it was easiest to achieve reliable communication using this scheme. Manchester encoding is also the slowest and other methods such as PAM were considered. However, it was not implemented.

2.3.6 Error Handling

To achieve stable communication, a communication system must have the ability to detect and correct errors that occur during transmission. This can be done both with and without feedback.

Error handling with feedback

Error handling with feedback is achieved through the use of checksums and acknowledgment of correctly received packets. After transmitting a packet, the transmitter

waits for an acknowledgement before sending the next packet. If the transmitter has not received an acknowledgment after waiting for a set amount of time, the packet is retransmitted. This process is repeated until the acknowledgement is received. This method was implemented in the VLC driver.

Error handling without feedback

Error handling without feedback can be achieved by encoding the data being sent in a clever way which allows the receiver to calculate how any eventual noise has affected the transmission, and thus recover the data as it was transmitted. This method has not been explored in this project, but the curious reader is recommended to look up the 7-4 Hamming code for a simple example of such a method.

Checksum Algorithms

Checksum Algorithms is a method for error handling with feedback. A checksum is a many-to-one mapping of an arbitrarily large amount of data into a sum of fixed size, for example 16 or 32 bits. Prior to sending a packet, the transmitter will calculate the checksum of the data contained in the packet, and will append the computed checksum to the end of the packet.

The receiver also computes the checksum over the packet data, and compares it to the checksum appended to the transmitted packet. If the checksums match, the receiver sends an acknowledgement to the transmitter. If, on the other hand, the checksums do not match, there has been an error during transmission and no acknowledgment is sent. In this case the transmitter retransmits the packet. The CRC and Adler-32 checksum algorithms are described below.

Checksum Limitations

Checksum limitations have to be considered when relying on them for communication. Since checksums is a many-to-one mapping, several versions of the same packet will have the same checksum. As such, the receiver might think a correct packet has been received when, in fact the packet is corrupted. However, the checksum algorithms are designed to minimize the probability of this happening by being constructed in such a way that a small change in the source data produces large change in the output checksum.

Cyclic Redundancy Check or CRC is a method to calculate control sums for files. There are different kinds of CRC. The simplest from of CRC is the *parity bit*. This method uses the sum of logical ones and zeros in a sequence and adds a logical one or zero at the end depending on if the sum is *even* or *odd*. The CRC-16, and CRC-32 algorithms are more sophisticated and compute a parity consisting of 16 and 32 bits, respectively. The drawback of the CRC algorithms is that the whole data source must be available when starting to compute the checksum which takes more time. For this reason, the CRC algorithms are not used in the VLC driver.

Adler-32 is a faster but less reliable method to detect errors than CRC. The main advantage compared to the CRC is the speed. However, this affects the reliability negatively and is known to have a weakness with short messages. Furthermore, the Adler-32 checksum is a rolling checksum that can be calculated incrementally as the packet is received. This means that the receiver will start calculating the checksum as soon as it starts receiving data, and that it will be computed instantly when the transmission is completed. The receiver can then send the acknowledgement without waiting for the checksum to be computed sometime after packet transmission. For this reason, this is the algorithm chosen in the VLC driver. Adler-32 is demonstrated in Appendix C.1.

Differences

Adler-32 differs from the CRC algorithms in how they are implemented. CRC checksums are computed by dividing the source data until it has reached the correct size. Adler-32 is computed by adding together the bytes of the source data.

The probability of a false positive on random errors is $2.3283 \cdot 10^{-10}$ for CRC-32, and the slightly larger $2.3294 \cdot 10^{-10}$ for Adler-32. The amount of data needed for the algorithms to be reliable is 4 kb for CRC-32 and 500 kb for Adler-32 [13].

Since packets can be smaller than 500 kb, the VLC driver should either use another checksum algorithm, or should concatenate packets to guarantee that they are always at least 500 kb if it is used in a production system.

Acknowledgment Errors

If a packet is received correctly, but the acknowledgment is corrupted during transmission, the transmitter will retransmit the packet even though it has already been received. This causes a duplicate packet. The VLC driver handles this case by including a sequence number in every packet.

Whenever a packet is received, the receiver compares the included sequence number with the number of the previous packet. If the sequence number is identical, the receiver sends an acknowledgement to let the transmitter know it has been received correctly, and then destroys the packet instead of passing it to the kernel.

2.3.7 Channel Model

A communication channel is used to convey information. The communication channel in this project can be modelled as in Figure 2.19 where X models the transmitter (encoder) and Y is the receiver (decoder). The possibility of incorrect decoding is of high importance. This model is based on Manchester encoding described in section 2.3.5. If the crossover possibility is p the channel is characterized by the probabilities in Equation 2.2.

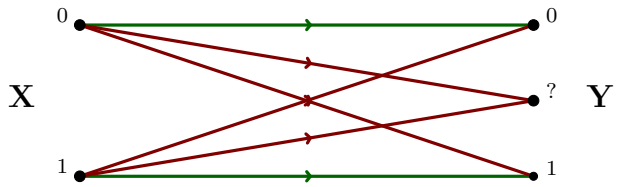


Figure 2.19: Bit level communication channel model based on Manchester encoding.

$$\begin{aligned} P(Y=0|X=0) &= (1-p)^2 \\ P(Y=?|X=0) &= 2p(1-p) \\ P(Y=1|X=0) &= p^2 \\ P(Y=0|X=1) &= p^2 \\ P(Y=?|X=1) &= 2p(1-p) \\ P(Y=1|X=1) &= (1-p)^2 \end{aligned} \tag{2.2}$$

Encoding for a channel

Data can be encoded to minimize the risk of errors during transmission for a specific channel. The highest transmission speed that allows for an arbitrarily small error probability is called the capacity of the channel. The possibilities of these methods of encoding is not explored in this project, and the curious reader is referred to "A. El Gamal and Y.-H. Kim, Network information theory. Cambridge, U.K.: Cambridge Univ. Press, 2011".

3
Results

This chapter presents a summary of the design choices made during the project and describes the achieved results.

3.1 Hardware

The transmitter and receiver results are described in the sections below. All figures, except Figure 3.4 and 3.10, are measurements with a signal of 25 kHz, at a distance of 1.5 meter. Figure 3.1 displays the signal from the BBB when it is propagated through the transmitter and receiver. This shows the results of the transmitter-receiver pair. The resulting propagated signal is nearly identical to the transmitted signal. The difference in duty cycle between the received and the transmitted signal is due to the capacitance in the circuits.

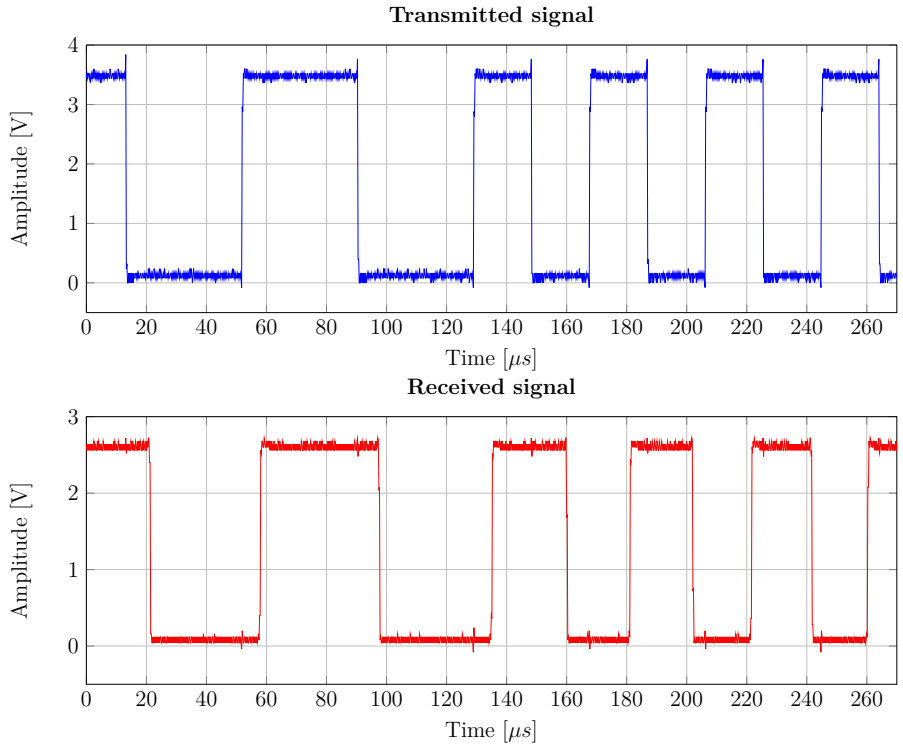


Figure 3.1: Example of a signal propagated from the transmitter to the receiver.

3.1.1 Transmitter

The transmitter receives a signal from the GPIO pins on the BBB, this signal controls the three transistors which open and close the power supply to the LED.

BBB

The BBB transmits a square wave with enough stability and a fast rise and fall time, as can be seen from the eye pattern in Figure 3.2. The eye pattern gives insight to the quality of the signal and the *ones* and *zeros* provides a stable signal for the transistor. The small amount of jitter and the small disturbance on the rising edge of the square wave does not create any problems. A sequence of the signal can be seen, as mentioned before, in Figure 3.1.

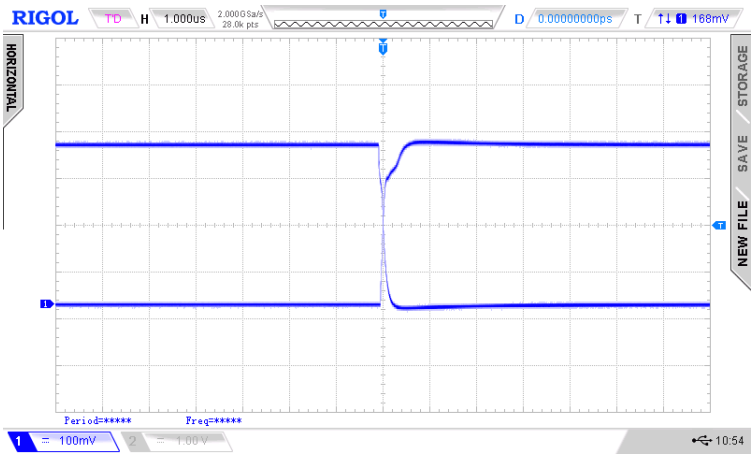


Figure 3.2: Eye pattern measured at the output of the BBB at the transmitter.

Transistors and LED

As can be seen from Table 2.1, showed earlier in the methods section, the LEDs receives a total current of 784 mA. This means it is 196 mA for each of the four LEDs. According to the datasheet, this means that the four LEDs generates a total intensity of 300 lumen during transmission. This light is enough to illuminate a small work area. The voltage over the LED can be seen in Figure 3.3 and contains large ripples from the power supply. Capacitors are used to reduce the ripple, but does not completely remove it. The transistors work without problems with the used frequency.

Maximum frequency

The highest frequency the transmitter can operate at is roughly 500 kHz. Above this frequency, the quality of the square wave becomes very poor, as can be seen in Figure 3.4.

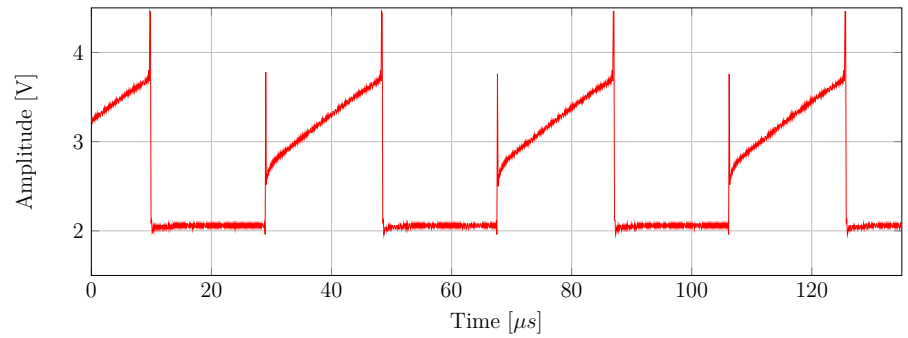


Figure 3.3: Signal measured over the LED.

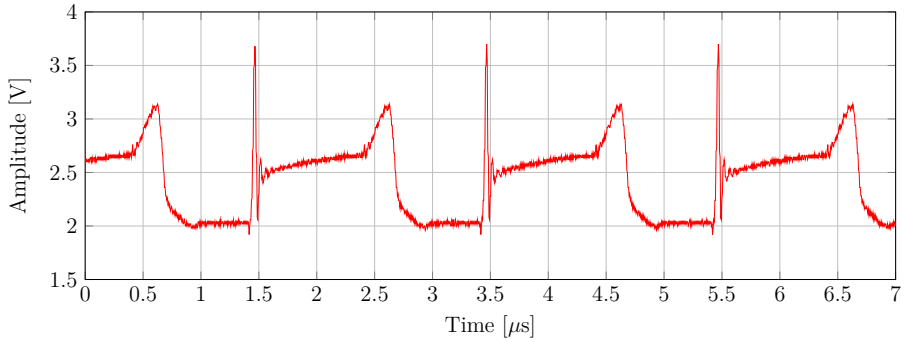


Figure 3.4: Maximum frequency measured at the output of the transmitter.

3.1.2 Receiver

The receiver converts the incoming light into an electrical signal and feeds it into the BBB. The receiver filters and performs AGC on the signal prior to feeding it into the BBB.

Photodiode and transimpedance amplifier

The photodiode chosen in the project has a large radiant surface area, which increases its sensitivity and allows for transmission over greater distances. This reduces the response time of the LED, which results in that the rise and fall time of the received square wave was increased. The signal is shown in Figure 3.5 and as can be seen from the figure the intercepted signal is no longer a square wave. The problem with change in duty cycle can be seen between 60 μs and 120 μs, which occurs when transmitting a sequence of two *ones* or two *zeros* in a row.

High-pass filter

The high-pass filter works as intended and removes the biased voltage noise from surrounding light sources. The signal is seen in Figure 3.6. Note that it is not the same sequence as previous figures.

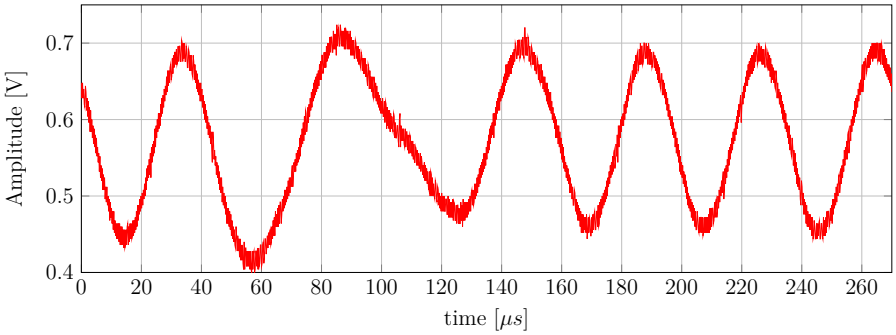


Figure 3.5: The signal measured at the output of the transimpedance amplifier.

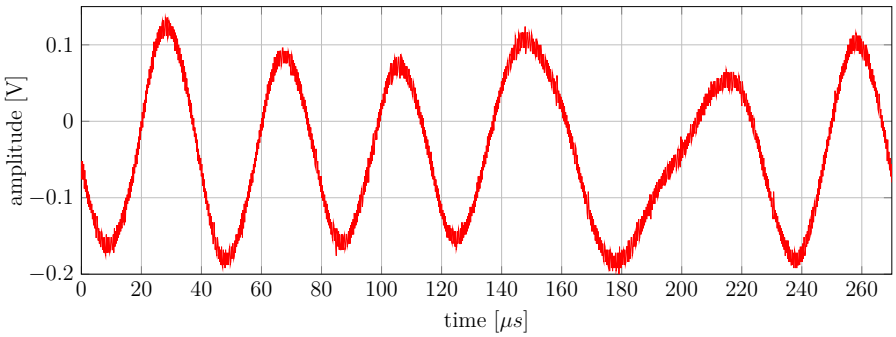


Figure 3.6: The signal measured at the output of the high-pass filter.

Automatic gain control

The remaining parts of the receiver converts the signal in Figure 3.6 into a square wave. During normal operation the AGC creates a square wave by amplifying and dampening the incoming signal such that the average value has a desired amplitude. The AGC is crucial to the design since it restores the square wave that was transmitted and makes up for the slow response time of the photodiode. The signal after the AGC can be seen in Figure 3.7.

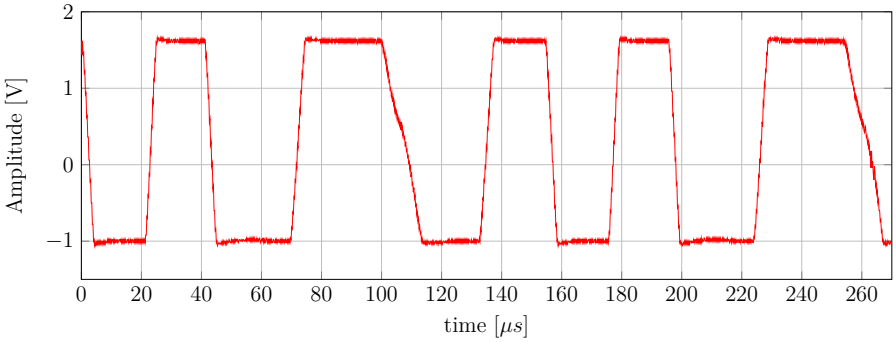


Figure 3.7: Signal measured at the output of the AGC.

Analog-to-digital converter

The ADC cleans up the square wave coming from the AGC, as can be seen in Figure 3.8. The eye pattern of the signal can also be seen in Figure 3.9. In the transitions, some jitters can be observed but the *ones* and *zeros* are stable and gives plenty of sample time.

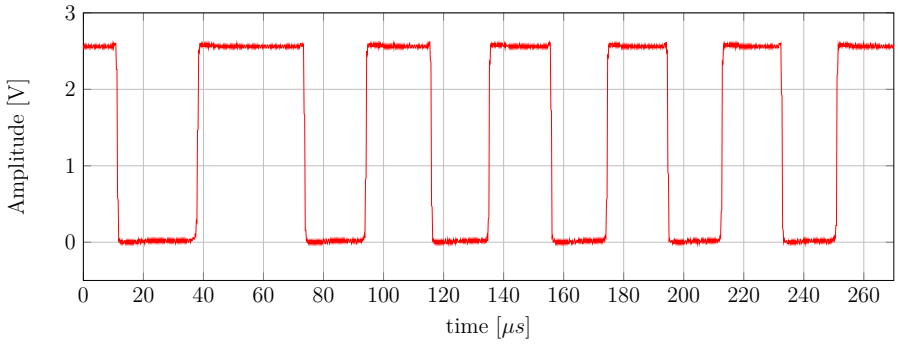


Figure 3.8: Signal measured at the output of the ADC.

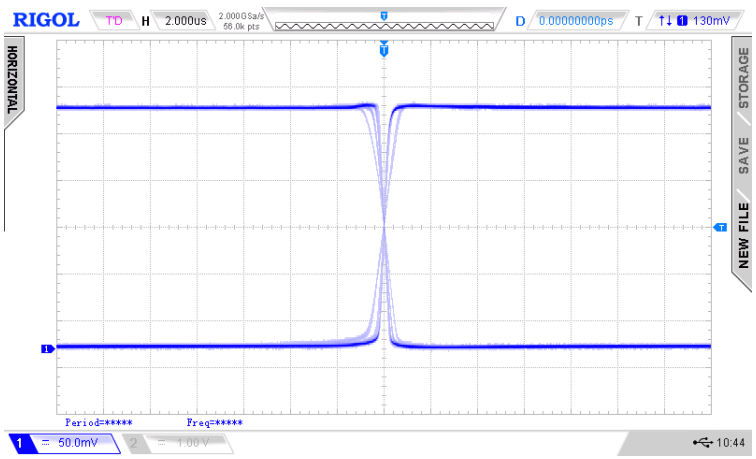


Figure 3.9: Eye pattern measured at the output of the ADC.

Maximum frequency

The maximum frequency for the receiver is tested to 80 kHz, where the output signal is still an undistorted square wave. The output can be seen in figure 3.10.

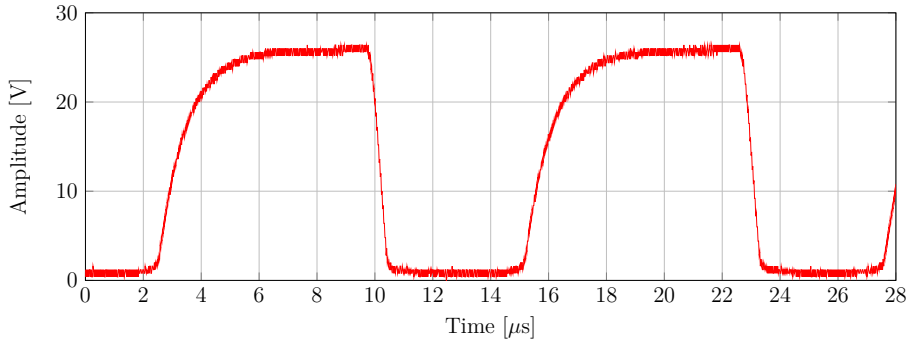


Figure 3.10: Maximum frequency measured at the output of the receiver.

3.2 Software

The resulting VLC driver is a software suite consisting of three parts that all run in parallel:

- Receiver
- Transmitter
- Packet management

The packet management system receives packets from the kernel for transmission and places them in a buffer that the transmitter will read from. Whenever there is a packet in the buffer, the transmitter will pick it up and send it over the channel. The receiver on the other end is always looking for the preamble when it is not currently receiving a packet. The received packet will be placed in a buffer, from which the packet handler will pick it up and pass it to the kernel. The kernel handles what to send, in what order, and most of the data packaging. The VLC driver is functioning according to the kernel instructions.

3.2.1 Design

The software is implemented as a state machine. Both receiver and transmitter keep track of its current state, and move to the next state when the current state has been completed. In some states a decision, such as whether to retransmit a packet or send the next packet, is taken. The complete state model is shown in Section 2.3.1.

The packet management runs in the background and will wait for packets to be handed to it from either the kernel or the receiver.

3.2.2 Synchronization

Data must be sent at precise intervals for reliable communication. To achieve this in the Linux environment, Xenomai is used. Xenomai is an extension to the Linux kernel which provides real-time functionality. Whenever a bit is to be sent over the channel, the VLC driver will wait for the next time slot where it is allowed to

transmit, and will then send it. The same applies for the receiver.

Nothing is done for symbol synchronization. The early-late method is implemented and has been tested, but it is cheaper to let the clocks drift apart occasionally, and simply retransmit the packet where the error occurred.

3.2.3 Data Packaging

The Linux kernel does most of the data packaging, but the VLC driver makes some additions.
The following sections are added:

- Preamble: To signal the receiver that a packet is incoming.
- Data length: To notify the receiver on how long the transmissions is.
- Checksum: For error handling purposes.

3.2.4 Hardware Interface

The VLC driver interfaces with the hardware of the transmitter and receiver through the GPIO pins present on the board. The data to be sent is written to one of the GPIO pins and the transmitter takes care of turning on or off the LED according to what bit is sent.

The VLC driver reads the GPIO pin configured as the input from the receiver. All gain control and filtering is done in hardware which increases the performance compared to software filtering.

3.2.5 Encoding

Data is sent using OOK and Manchester encoding. This is a requirement for the AGC used in this project. If transmissions are performed without Manchester encoding, the AGC will be confused by long strings of the same symbol. For example, a long string of *zeroes* will cause the AGC to increase its amplification, and it will start flipping bits. Furthermore, Manchester encoding significantly reduces the perceived flicker of the light source.

Manchester encoding can also be used for synchronization purposes, since a Manchester symbol always has a transient in the middle, which can be detected. However, this feature is not used in the VLC driver. For the reasons behind this, see Section 2.3.5.

The downside of Manchester encoding is that it effectively halves the rate compared to system without it, since every logical bit is sent using a 01 or 10 sequence.

3.2.6 Error Detection and Handling

Error detection is done through the use of checksums. The checksum of the data to be sent is computed by the transmitter, and sent along with the data. The receiver will also compute the checksum of the packet upon reception, and compare it to the checksum sent along with the data. If the checksums are identical, the receiver will send an acknowledgment to the transmitter, which will then send the next packet. If the checksums differ, the receiver will not send an acknowledgment, and after a timeout, the transmitter will resend the packet. This is repeated until the packet has been successfully received. Specifically, the Adler-32 checksum is used.

The main advantages of the Adler-32 checksum is that it is computationally cheap and that it is a rolling checksum, which means it can be computed incrementally over the packet as it is received. For this reason, the receiver will know if the checksums match instantly when the packet has been received, and does not have to wait for it to be computed. This reduces the delay between packet transmission and acknowledgment, which is important since the transmitter will not send the next packet until the current packet has been acknowledged.

Errors can still occurr when using checksums. Since a checksum is a many-to-one mapping several packets will have the same checksum. This is mitigated by how checksum algorithms are implemented. The algorithm is designed to create large variations in the checksum from small variations in the input data. This means that, for another packet to have the same checksum as the one currently being transmitted it has to be an entirely different packet. In that case the packet is most likely not even a valid packet, and the error will be detected in the Linux kernel.

4

Discussion and Conclusions

The project has completed its goal of designing and implementing of a VLC system. There are still many areas of improvement, but the system works!

This final chapter offers some closing discussions regarding the project, compares the end result with the goals set up at the beginning of the project, and suggests areas of further work on the implementation. The VLC driver is open source, and is available on Github [14].

4.1 General Discussion and Limits

Development platform

The project is using a single board computer for development. This platform has been useful for development and for constructing a prototype. However, this has proven to lack enough processing power needed for an implementation of a network interface device.

The advantages of the platform are that it has an ADC subsystem capable of reading analog values and it has a real-time subsystem for running timing-critical tasks, such as communication. However, these features are not used in the end product, and the platform could be exchanged for one with increased processing power, such as the Raspberry Pi 2.

Channel comparison

Communication over visible light compared to communication over frequencies outside the visible spectra looks very promising. The frequency spectrum used is significantly wider than what is used in radio wave communication systems. For example, the visible spectra is 10 000 times wider than what is used for WiFi [15] and provides significantly more freedom and headroom for implementation. As a result VLC is not nearly as sensitive to bandwidth crunch. For example, if there are more than three WiFi access points within range of each other, their frequency bands will start to overlap and they will interfere with each other [16].

Encoding and modulation

To increase the transmission speed a different encoding style is preferable. Manchester encoding provides a stable signal with an equal amount of *ones* and *zeros* over time which is suitable for the AGC. Without Manchester encoding, for example, PAM could be used. To use PAM the transistor used as ADC needs to be redesigned and instead use a ADC with a larger resolution, so multiple values of PAM can be detected.

Hardware Availability

Most communication systems using light either use infrared light, such as remote controllers, or lasers, as in fiber systems. This means that there are few available hardware components for implementing a system for VLC. Hopefully this will change in the future.

Light output

The perceived light strength could be increased in three ways. The first way is to increase the light output when transmitting a *zero*. For example, representing a *zero* by turning the LED to output 500 lumen. This would increase the perceived light strength to 550 lumen. However, reducing the difference between *zero* and *one* will increase bit errors since the signals will become more similar. The alternative would be to increase the current to the LED. This will require cooling of the LED, because of the increase in power. The more robust solution to increase the perceived light strength is to use a stronger LED. The downside of this approach is an increased power usage for transmission, and most likely the optimal solution is some combination of these solutions.

Signal Reception

Since photodiodes for visible light are designed to detect a broad spectrum of wavelengths, the receiver is sensitive to surrounding light sources. With a narrower detection spectrum less noise would enter the receiver circuit.

To increase the used carrier wave, and thus the transmission rate of the system, a photodiode with short rise and fall time is needed. The currently used photodiode is not quick enough and is the bottleneck in the hardware. As mentioned before a big radiant sensitive area entails a slower response time. A possible solution may be using a photodiode with a smaller radiant sensitive area and direct the light with an optical lens. The rest of the hardware in the receiver is capable of handling frequencies up to the order of MHz.

4.2 Conclusions

When comparing the end result with the project goals all the required specifications, and some of the desired specifications were met. As a reminder, the project goals

were as follows.

4.2.1 Required Specifications

Here, the end result is compared with the required specifications defined at the beginning of the project.

Send and receive arbitrary data over visible light

The device is capable of both transmitting and receiving arbitrary data over visible light, due to being implemented as a Linux network device. The device will receive arbitrary data from the Linux kernel for transmission, and the contents of that data does not affect the behaviour of the system.

Transfer data while producing 450 lm

The system uses a high-power LED with an average of 300 lm, which does not meet the requirements. To increase the intensity a larger external power supply is needed to support the increased current from another LED.

Send data over at least 1 meter in a brightly lit room

Tests show that the system can easily transfer data over 1 meter in a lit room.

Send data in one direction

The system is capable of transmission in one direction. Due to only having built one transmitter-receiver pair, the transmissions in the other direction is performed via a copper wire.

Perform an uninterrupted transfer of 10 MB

The system can perform an arbitrarily long uninterrupted transfer. The system is built to retransmit a packet until it has arrived successfully, so any interruptions will be compensated for in the VLC driver.

4.2.2 Desired Specifications

Here, the end result is compared with the desired specifications defined at the beginning of the project.

The device can connect to any computer over Ethernet

The device can connect to any other computer over Ethernet by simply running on a Linux computer. Linux has the capabilities of easily bridging the Ethernet interface already present on the board with the VLC network interface.

The device should manage to send data over a distance of 2 meters in a brightly lit room

The device can transmit data over a distance of 2 meter in a brightly lit room.

The system can perform half- or full duplex transmissions

The system can perform full duplex transmissions by using a wire for transmission in one of the directions, and light in the other. This is due to having built only one transmitter-receiver pair. But there is no reason the wire could not be replaced with another VLC transmitter-receiver pair. However, this has not been tested.

The device can perform reliable and lossless data transmission

The system will compensate for any packet loss by using checksums to verify the integrity of the packet, and will retransmit any corrupted packets. Checksums are not guaranteed to always work, but the risk of having a packet corrupted with an identical checksum is the very slight $2.3294 \cdot 10^{-10}$ for packets of at least 500 kb [13].

The device supports light intensity variation during transfer

Dimming is not supported, but could easily be implemented since no amplitude modulation is used. Furthermore, the hardware is designed with eight intensity steps. These are not currently used, but could be controlled either through the software or through a dial on the transmitter. However, the probability of error will increase and the distance the system can transmit over will decrease by lowering the intensity.

The device can connect to multiple units simultaneously

Router functionality is not supported, and most likely significant work is required to implement it. The data being transmitted must, in that case, be encoded in a way that does not interfere with other transmissions.

The device should support multiplexing

Multiplexing is not supported either. This could be implemented by stacking several transmitters and receivers. However, the VLC driver is not designed for this and performance will be poor. The BBB is already at its limits of processing power running one channel at 50 kHz.

4.3 Further Work

Further work on the VLC implementation includes increasing the system capacity, and adding useful features.

4.3.1 Increasing System Capacity

For increasing the systems capacity, the recommended first step is to switch to a more powerful development board for running the VLC driver. Especially, the software will scale very well into a system with four processor cores. This is because, in that case, the transmitter, receiver, packet management, and operating system can run on separate cores, significantly reducing the overhead from switching between the tasks.
The software could also be adapted to run on specialized hardware without an operating system, to further reduce the overhead. However, this requires significantly more work.
Furthermore, the following areas could be explored to increase capacity:

- Data compression prior to transmission
- Encoding data to reduce retransmissions
- Optimization of the VLC driver

Data Compression

The packets received from the kernel are uncompressed, which is inefficient for transmission. If the VLC driver compressed the packets before transmitting them, every packet could be sent using fewer bits.

Furthermore, data can be compressed to contain an equal number of *ones* and *zeroes*, which might make Manchester encoding unnecessary.

Data Encoding

The number of packet retransmissions can be reduced by using error handling without feedback. This is done by encoding the data in a way that allows the receiver to calculate how the noise has affected the transmission. This is a complex area, but it can potentially allow for significantly more robust transmission.

VLC Driver Optimization

There are several parts of the VLC driver that can be rewritten to increase performance. For example, the time needed to move between packets in the packet buffer could be reduced.

4.3.2 Additional Features

The VLC driver is currently designed to work in a system with two units communicating, but the system can be adapted to work in larger networks consisting of many transmitter-receiver pairs communicating simultaneously.

Bibliography

[1] Anonymous, "Visible light communications: Back story light-bulb moment," *The Engineer*, p. 1, 2011.

[2] L. Grobe, A. Paraskevopoulos, J. Hilt, D. Schulz, F. Lassak, F. Hartlieb, C. Kottke, V. Jungnickel, and K.-D. Langer, "High-speed visible light communication systems," vol. 51, no. 12, p. 60, 2013. [Online]. Available: http://ieeexplore.ieee.org/xpls/icp.jsp?arnumber=6685758

[3] E. F. Schubert, *Light-Emitting Diodes (2nd Edition)*. Cambridge University Press, 2006, book section 22, p. 379.

[4] Bughunter, "Internet protocol stack," May 2015. [Online]. Available: http://commons.wikimedia.org/wiki/File:InternetProtocolStack.png

[5] Wikipedia, "Internet protocol suite — wikipedia, the free encyclopedia," 2015, [Online; accessed 11-February-2015]. [Online]. Available: http://en.wikipedia.org/w/index.php?title=Internet_protocol_suite&oldid=644032151

[6] K. K. Ng, *Complete Guide to Semiconductor Devices*. New York: John Wiley & Sons, Inc, 2002, book section 2, p. 30.

[7] BeagleBoard, "Beagleboard:beagleboneblack," May 2015. [Online]. Available: http://elinux.org/Beagleboard:BeagleBoneBlack

[8] E. F. Schubert, *Light-Emitting Diodes (2nd Edition)*. Cambridge University Press, 2006, book section 23, p. 391.

[9] K. Jeppson, *Microelectronics*. Chalmers University of Technology, Department of Microtechnology and Nanoscience, 2012.

[10] T. Wang and B. Erhman, "Compensate transimpedance amplifiers intuitively," Texas Instruments, Application Report, 1993. [Online]. Available: http://www.ti.com/lit/an/sboa055a/sboa055a.pdf

[11] I. O'Reilly & Associates. (2015, May) Linux device drivers, 2nd edition. [Online]. Available: http://www.xml.com/ldd/chapter/book/ch14.html

[12] C. Nagpal, "Beaglebone black gpio through /dev/mem," March 2015. [Online]. Available: http://chiragnagpal.com/examples.html

[13] zlib. (2015, May) Adler-32 versus crc-32. [Online]. Available: http://www.zlib.net/zlib_tech.html

[14] Group, "Vlc driver source code," March 2015. [Online]. Available: https://github.com/Zeverin/VLC-driver

[15] H. Haas, "Wireless data from every light bulb ted talk," May 2015. [Online]. Available: http://www.ted.com/talks/harald_haas_wireless_data_from_every_light_bulb#t-205142

[16] Wikipedia, "Wifi frequency bands — wikipedia, the free encyclopedia," 2015, [Online; accessed 18-May-2015]. [Online]. Available: https://en.wikipedia.org/wiki/List_of_WLAN_channels

A

Appendix

Here follows a list of the active components used in the project, also a list of the instruments used during testes.

A.1 Component list

Component	Name and Datasheet
AGC	TI TL026CD IC http://www.farnell.com/datasheets/1835114.pdf
LED	CREE MCE4WT-A2-0000-000HE7 http://www.farnell.com/datasheets/1821872.pdf
Operational amplifier	TI μA741CD http://www.farnell.com/datasheets/1834254.pdf
Photodiode	VISHAY, BPW21R http://www.vishay.com/docs/81519/bpw21r.pdf
MOSFET transistor	IRLR3715ZPBF http://www.farnell.com/datasheets/107852.pdf
NPN transistor	BC547A http://www.arduino.cc/documents/datasheets/BC547.pdf

Table A.1: Active components used in the project.

A.2 Instrument list

Instruments	Name
Function generator	Agilent, 33220A
Oscilloskop	RIGOL, DS2072A
Power supply	TTI, PL303QMD

Table A.2: Instruments used during tests.

B

Appendix

Figure B.1 and B.2 shows the final transmitter. The prototype transmitter is shown i Figure B.3. The prototype over the transmitter is in Figure B.4.

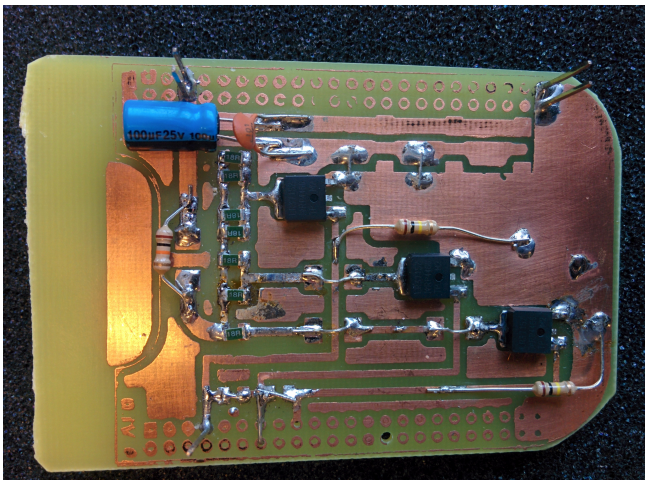


Figure B.1: Picture of the final transmitter.

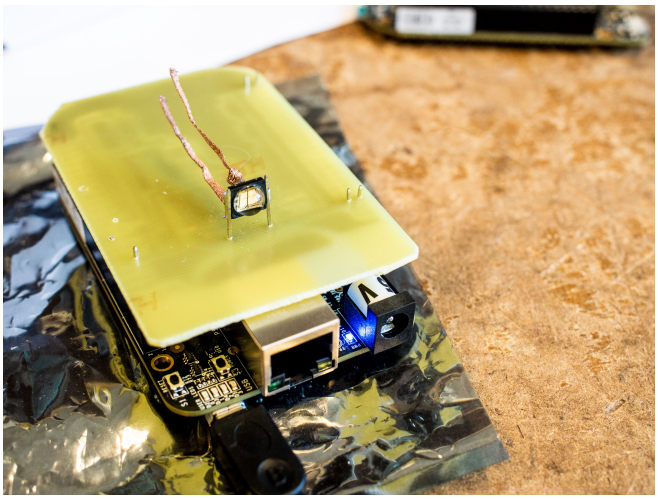


Figure B.2: Picture of the final transmitter mounted on the BBB.

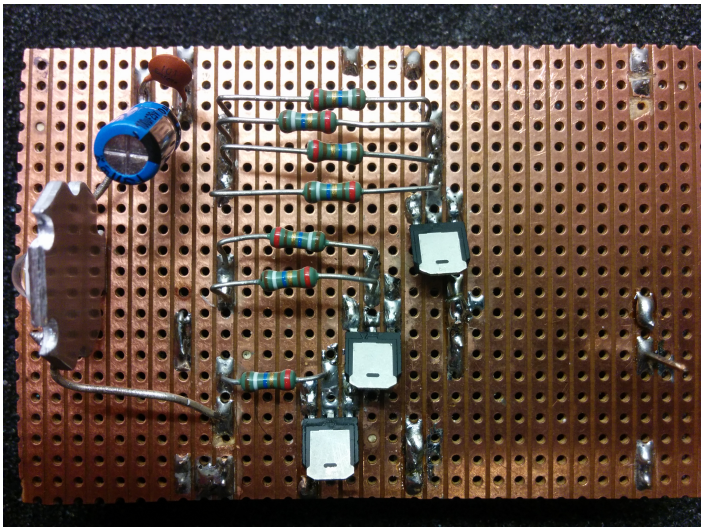


Figure B.3: Picture of the prototype transmitter.

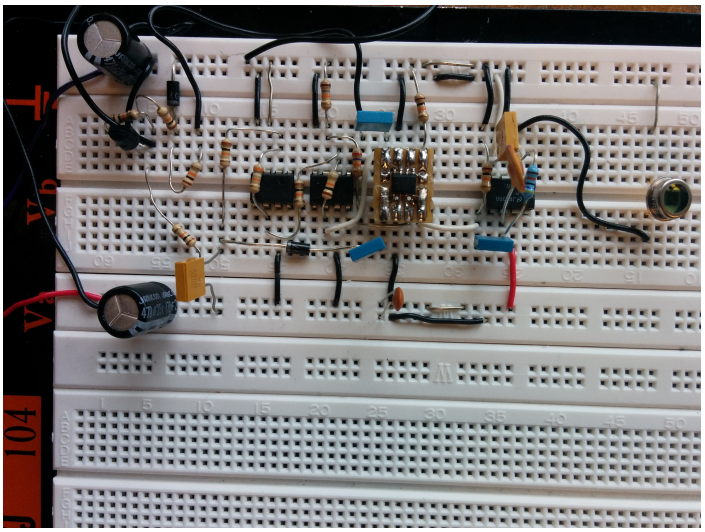


Figure B.4: Picture of the prototype receiver.

C
Appendix

C.1 Adler-32 Demonstration

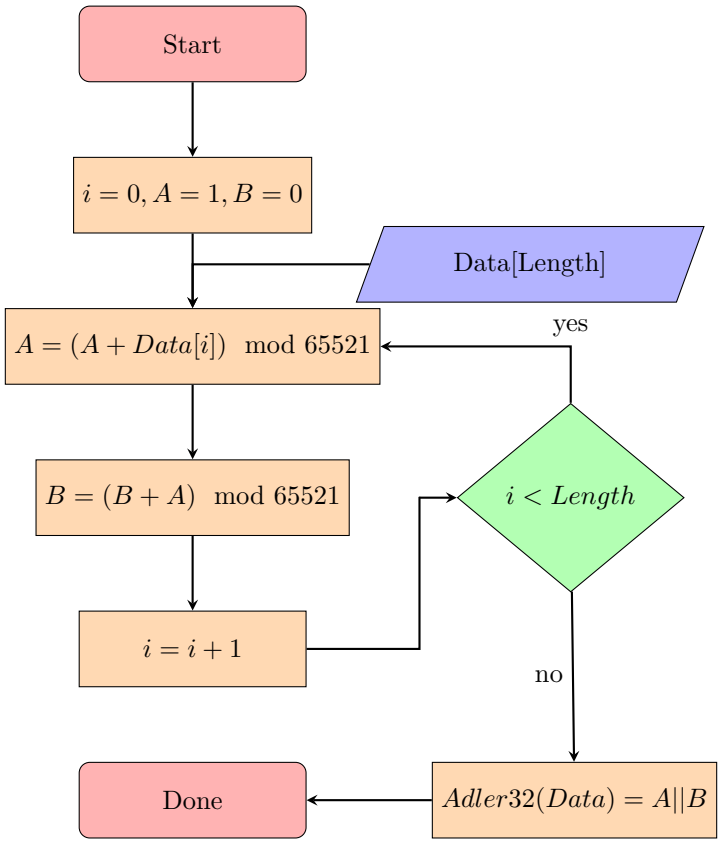


Figure C.1: Adler-32 sequentially described. Two variables A and B is created and the data is looped through bytewise. Dependent of the file content A and B get different values which is concatenated. This represent the Adler-32 checksum of the file.