Department of Computer Science

Tobias Gunnarsson
Hans Johansson

Visual implementation of computer communication

Computer Science
C-dissertation (15 hp)

Date/Term:	2010-06-08
Supervisor:	Stefan Alfredsson
Examiner:	Martin Blom
Serial Number:	C2010:14

Karlstads universitet 651 88 Karlstad
Tfn 054-700 10 00 Fax 054-700 14 60
Information@kau.se www.kau.se

Visual implementation of computer communication

Tobias Gunnarsson
Hans Johansson

This report is submitted in partial fulfillment of the requirements for the Bachelor's degree in Computer Science. All material in this report which is not our own work has been identified and no material is included for which a degree has previously been conferred.

Tobias Gunnarsson

Hans Johansson

Approved, June 8, 2010

Advisor: Stefan Alfredsson

Examiner: Martin Blom

Abstract

Communication is a fundamental part of life and during the 20th century several new ways for communication has been developed and created. From the first telegraph which made it possible to send messages over long distances to radio communication and the telephone. In the last decades, computer to computer communication at high speed has become increasingly important, and so also the need for understanding computer communication. Since data communication today works in speeds that are so high that the human eye cannot by any chance see the signals that are sent it is hard to, on a basic level, show how communication work. Therefore in this project a communication system is made of lasers, photo diodes and Morse code as the encoding. The lasers are used to create a communication link between two computers to visualise how communication work and the Morse code will be used as the language the lasers and the photo diodes use to understand each other. The goal was to create a communication system that allows two way communication and the results which was achieved are a system that allows half-duplex communication. The project resulted in that the hardware and software can be used to show how communication works in a controlled environment.

Acknowledgements

We would like to thank our advisor Stefan Alfredsson who has been invaluable when it comes to tips, help and gathering of the components. Without him the time needed to get as far as we did in the project would have taken several weeks of extra time. We would also like to thank Karlstad University for the facilities we have been able to use here. A big thank you will be directed to Simon Glyssner who have helped us to analyse the text to eliminate spelling errors and overall read the text to make the text more flowing.

We acknowledge the following sources for some of our figures:

Figure 2.1 is copied from `http://en.wikipedia.org/wiki/File:Photophony1.jpg` and since it is 130 years old we believe that copyright has expired and the picture is in the public domain.

Figure 2.2 is copied from `http://commons.wikimedia.org/wiki/File:T%C3%A91%C3%A9graphe_Chappe_1.jpg` and is claimed to be in the public domain because of copyright expiration. The image has been slightly modified for our report.

Figure 2.3 is copied from `http://en.wikipedia.org/wiki/File:Optical_telegraph01_4484.jpg` and the licence says: *"Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation".*

Figure 2.4 is copied from `http://en.wikipedia.org/wiki/File:J38TelegraphKey.jpg` and the licence says: " *This file has been (or is hereby) released into the public domain by its author, Lou Sander at the Wikipedia project. This applies worldwide.*" it also say " *Lou Sander grants anyone the right to use this work for any purpose, without any conditions, unless such conditions are required by law.*"

Figure 2.5 is a modified version of `http://commons.wikimedia.org/wiki/File:Spectre_visible_light.svg` originally created by Wikipedia user Tatoute and modified by Isometrik, and licensed under Creative Commons Attribution ShareAlike 3.0.

Contents

List of Figures

1 Introduction

In this project a basic laser communication system will be constructed. This will make it possible to send messages between two computers with the use of laser beams. This will then visually show how communication can work and this is striven for since it will be useful for educational purpose when demonstrating different data communication systems and how the different systems can work.

1.1 Implementation of the project

In order to create the communication system more knowledge had to be acquired about similar projects. A main characteristic of the system is the operation in free space without for example fibre cabling. Such technology is called Free Space Optics, FSO, and a need to study history of telecommunications had to be made to get more aware of the communication field. A set of laser pointers and USB Interface control board of the brand Velleman have been acquired for each computer. The motivation for using laser can be seen in section 2.3. The USB interface board can read input ports and control the outgoing voltage on the outgoing ports. This can then be used to control the laser. The program will be written in the programming language C and it will be a two-way chat program which can exchange text messages. Morse code will be used to encode and decode the messages. Morse code will be used since it makes it easy to comprehend for the human eye and the communication can even be read by a person that know Morse code. The overall structure of the project can be seen in figure 1.1. The figure shows two computers connecting to an separate USB Interface card that is connected to a laser or a photo diode with copper wires.

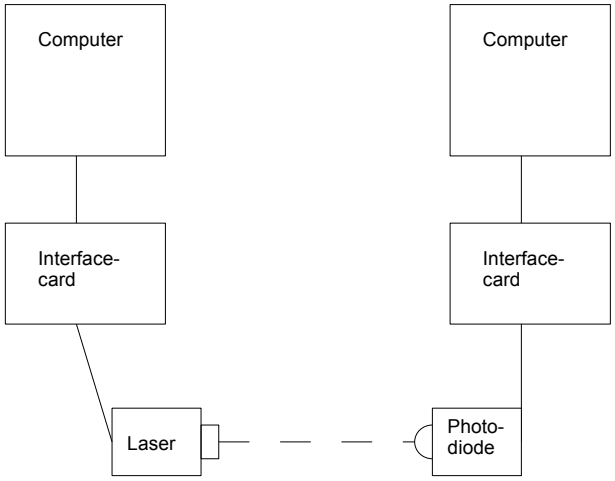


Figure 1.1: Overview of the communication system

1.2 Goals

The main goal of this project is to accomplish a laser beam communication link between two computers. The purpose is to visually demonstrate how communication between computers can be done.

The first goal is to be able to, by using a computer, turn on and off the laser and be able to detect at another computer. The method chosen to control the laser is through a USB interface board, and the method to detect the laser is by a photo diode. First the API of the USB interface board needs to be read and understood. The need to understand the API (Application Programming Interface) is needed to be able to control the laser and to be able to use the photo diode for detection. After being able to control the laser and have the detection working, the focus will be on sending and detecting simple bit-patterns. Further on when the basic hardware part of the project is working the start of creating a program that can control the data communication will be made. First a decision on what physical

encoding that will be implemented and then the encoding will be implemented according to the rules set for the specific physical encoding. When the encoding and decoding functions are working some simple one-way messages will be sent, in the beginning just by sending simple ASCII characters and then try out the performance by testing different distances and higher transmit rates. When all this is working a two-way link will be created and the developing of a basic chat application is possible.

Goal list

Step-by-step goal list in order of execution.

1. Reading and understanding the API of the interface board.
2. Control laser pointer beam from computer.
3. Detect active/inactive beam from computer.
4. Transmit a simple bit pattern.
5. Detect a bit pattern at receiver.
6. Create a Morse code communication.
7. Decide on the physical encoding.
8. Implement the physical encoding and decoding.
9. Make an one-way chat application (transmit characters, present at receiver).
10. Do a simple performance test
11. Increase the distance and find the limits of the software/hardware.
12. Increase the speed and find the limits software/hardware.
13. Implement two-way communication and chat application.

1.3 Report outline

The remainder of the report is structured as follows. In chapter 2, a brief history of telecommunication is presented that will include wired- and wireless communication and then some basic knowledge that is needed to understand the rest of the report. Then there is a discussion of existing projects and the purpose of why to use lasers to communicate. In chapter 3, a discussion of the construction of how the laser system have been built and how it has been implemented to create a communication link is made. There is also a section of what kind of technical characteristics that has been used. Chapter 4 describes the program and its implementation. Finally in chapter 5, 6 and 7 the results, a user manual of how to set-up and use the program, our experiences and what future work that can be made to improve the program are described.

2 Background

This chapter presents a brief history in both wired and wireless communication (see section 2.1) leading up to the high-speed communication used today, followed by our motivation for using laser to communicate in section 2.3. After that, a discussion of what physical encoding is and how it works are explained in section 2.4, followed by a description of how the state machine works in section 2.5. Then, a presentation of how old works and projects are related to this project and what difference it is (see section 2.6). Last, a description of what components that have been used and its features are explained in section 2.7.

In the late 19th century and early 20th century many new ways to communicate were invented, from the first telegraph transmissions to the use of the first wireless communications with the photo phone, which where invented by Guglielmo Marconi, where actual sound was transferred. To be able to send any information some sort of medium is needed to transfer the information sent by the transmitting side to the receiving side. May it be, air and drums to create and propagate sound or a transceiver with wires to send the information on. There also needs to be a language that both sides understand (can also be seen as a protocol between the sender and the receiver). That could be simple messages via drums e.g. that three repeated taps on the drums means forest fire, to complex binary systems used in the computers today. This is known as the physical encoding [1].

The different common physical encodings that are used today are Manchester code which is used in Ethernet, in both LAN and WAN. Other encodings in use today are Morse code for long distance short message communication or common unipolar encoding where a positive voltage is seen as a binary 1 and zero voltage as binary 0.

The purpose of this project is to create a communication link between two computers with the use of laser pointers then get them to communicate with each other through the communication link and to finally be able to see how communication work. The link is constructed by using an USB interface card to control the laser on the sender side and to

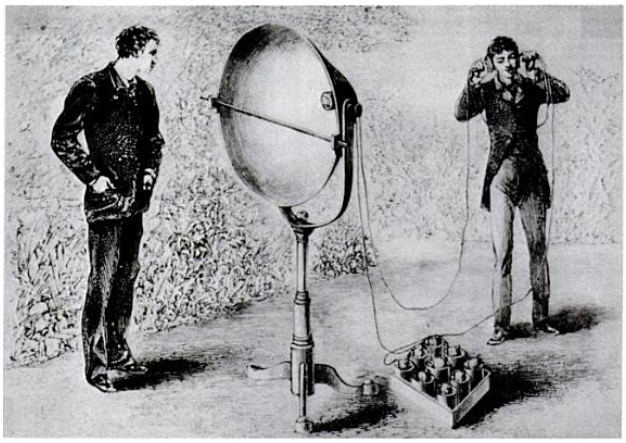


Figure 2.1: Bell and Tainter testing their photo phone receiver, of 1880.

detect the laser on the receiver side with a photo diode.

These components will serve as the connection but to be able to even communicate at all between the two interface cards there is a need for some sort of encoding (the protocol used between the to sending and receiving parties). All this so both sender and receiver know how to interpret the protocol that is used (how binary ones and zeros look like). The easiest way to do this would be laser on (positive voltage) equals binary one, and laser off (zero voltage) equals binary zero also known as basic line encoding. However this may lead to problems with synchronisation.

Problems that occur with synchronization and the reason that a basic encoding system is not used is that even though the sending computer can clock the signal and send it with very precise timing the existence of timing-drift still exist. This can lead to for example if the sender sends for example 100 zeros (the lights off for 100 time units) the receiver might not be able to clock those as exactly 100 but can see them as maybe 99 or 101. To solve this problem a protocol (physical encoding) must be used. Manchester-encoding (also known as phase coding, see section 2.4), which is a self-clocking encoding, is one of the existing encodings that helps the communication to stay synchronized. To get communication up and working Morse code as physical encoding will be used which in itself is self clocking

encoding. Morse Code is also good since the signals of Morse Code make them easy to understand while only looking at the signals when data is sent. To make the system resilient to de-synchronization, a state machine will be added (see section 2.5) this is made since the program cannot handle sending and receiving at the same time.

2.1 History of telecommunication

In the very beginning of telecommunication the way people communicate with each other, over long distances, were very basic. They used smoke signals, fires and drum sounds. These techniques were used to transmit news, signal danger, or gather people to a common area [2].

All these old ways of communicating relate to this project via the basics of communication. A sender, a receiver and a protocol that both sides understand so a communication can be made. The fires or the drums can be compared to the laser in this project and the different ways of tapping the drums or lighting the fires can be compared to the physical encoding. Physical encoding can easily be compared to the regular languages used in the world. All humans can make sounds (all working lasers can send light) but to understand a person that one needs to convert those sounds into words and the same goes for the laser. The sender needs to control the laser to send light with certain predetermined time intervals so they can be decoded on the receiving side using a a set of rules predetermined before the conversation started.

2.2 The telegraph

In the 18th century, a french engineer named Claude Chappe, built the first optical telegraph, also known as a semaphore system [3], see figure 2.2.

A variant of the optical telegraph was developed in 1794 by the swede Abraham Niclas Edelcrantz. The biggest difference from Chappe's semaphore system was that instead of three movable arms he used three fixed arms that were built on the mast, see figure

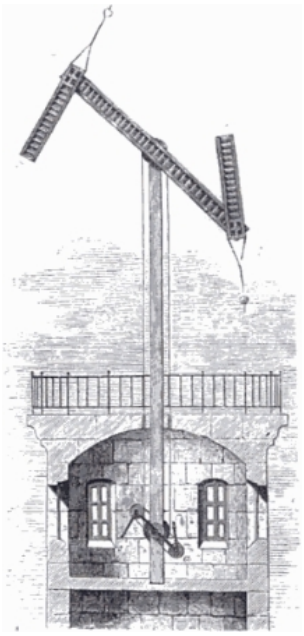


Figure 2.2: Claude Chappe's telegraph system.

2.3, on each arm there were three square gaps. These square gaps could either be placed horizontally or vertically. Any gap in the upper arm means 1, the middle 2 and on the bottom 4th.

The nine doors that these gaps would form were organized so that they also formed three vertical rows. The first vertical line of distinguished entities, the second line dozen and the third line hundred. From the gaps it was possible to distinguish any three-digit number from 0 to 777.

These gaps could be alternately opened and closed and thus create up to 1024 different combinations [4]. The system can be described as a forerunner of the number representation in modern computer systems, since it is based on a binary system with 10 signal elements. All this technology including smoke signals, beacons, reflection lights and semaphore systems are also called optic telegraphy.

Figure 2.3: The telegraph Abraham Niclas Edelcrantz made.

2.2.1 Wired communication

In the early 19th century electric telegraphy was invented and it made it possible to send messages via electrical wires. Messages were sent in form of short or long pulses of electrical current to control an electromagnet that was located at the receiving end of the telegraph wire. These pulses could represent Morse code. Figure 2.4 shows a picture of an electric telegraph, the U.S. model J-38. Morse code was invented by Samuel F. Morse and Alfred Vail at the start of 1836. For further description of the implementation of Morse-code used in this project is described in section 2.4.2.

2.2.2 Wireless Communication

The distances wireless communication can travel are from just a few meters (bluetooth, infra-red light) to millions of kilometres (space probe radio communication). The first wireless communication occurred in 1880 when Alexander Graham Bell and Charles Sumner Tainter did the first telephone call with their invention, the photo phone [5]. Figure 2.1 shows Bell and Tainter when they are testing their very first photo phone.

A pioneer in wireless communication was Guglielmo Marconi [1], an Italian inventor who won the Nobel prize in Physics 1909 together with Karl Ferdinand Braun "*in recognition of*

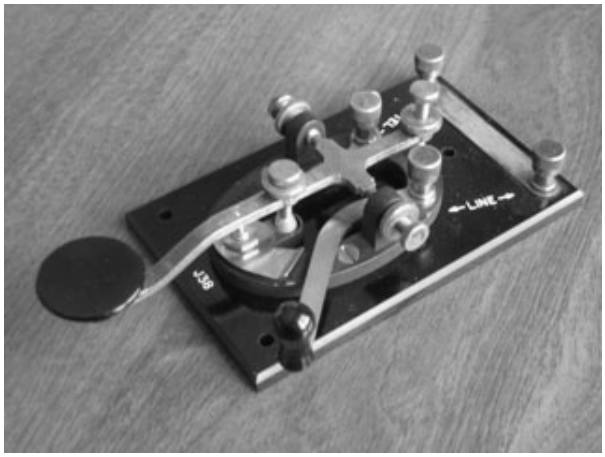


Figure 2.4: The J-38 electric telegraph.

their contributions to the development of wireless telegraphy"[6]. They built the first radio telegraph which served as basis for wireless communication world wide. Their discovery was also the base for the establishment of a great number of companies all around the world (from phone companies such as Ericsson and Nokia to electronics companies such as SONY and Toshiba). Wireless communication is now used all around the world in several types of equipment such as mobile phones, WiFi, Television remote controls, Computer Interface Devices and of lately Wireless energy transfers.

2.3 Motivation for using laser

When both the two sending and receiving parties are too far apart with no visible obstacles, such as space, pure distance or that the area is dangerous to navigate in, the use of wires to link the sender and the receiver together will be virtually impossible. For example in satellite to satellite, or satellite to earth communication or in war zones. Laser as a medium to propagate information on can be used to set-up quick communication links when there is no time to set-up a wired network such as in emergencies/natural disasters, or were the use of wires are impossible such as earth to space communication or space to space communication. The reason to use laser instead of a radio link is that the bandwidth you

get from laser is over 100 times higher then what a radio link can give. Since laser is a highly directional link the power usage of the link itself is lower then what would be needed for radio link. The size of the equipment is a lot smaller than what is needed in radio communication and the low-to-none spread of the light in laser beams makes them much harder to wire tap compared to radio communication. All this since regular radio communication spread the signals and makes them much easier to intercept and eavesdrop [7].

The laser itself is very concentrated and have a high power output per square centimetre. But since the light itself does not spread the actual energy needed to create the beam the power usage is very low. From as low as the non harmful lasers with less than 1mW to the lasers that are dangerous to both skin and eyes that are working over 0.5W. The beam compared to regular light bulbs, anything from the small flash light bulbs with 1.5W to regular light bulbs in lamps with 60W, and other devices that emits electromagnetic radiation does not spread and lose energy over distance. This gives the laser the property of creating a clear signal with very little use of power (less then 1W).

A big part of this project is the visual aspect of the laser. All the different visible colours, non visible colours such as infra-red or ultraviolet too, all depend on the frequency of the laser. The lower the frequency of the laser the closer to the ultraviolet spectrum and with really low frequencies the laser becomes a X-ray laser. All the colours of the laser are coherent with wavelength the light spectrum seen in figure 2.5. Since this project is based on visualisation a laser with a frequency within the visual, for the human eye, light spectrum is chosen. With a wavelength of 660nm the colour of the laser in this project becomes red.

One big problem with laser is the directional beam of light that it produces. This requires a clear line of sight, also known as LOS or path of sight, to let the laser propagate the information to its destinated receiver without interruption [8].

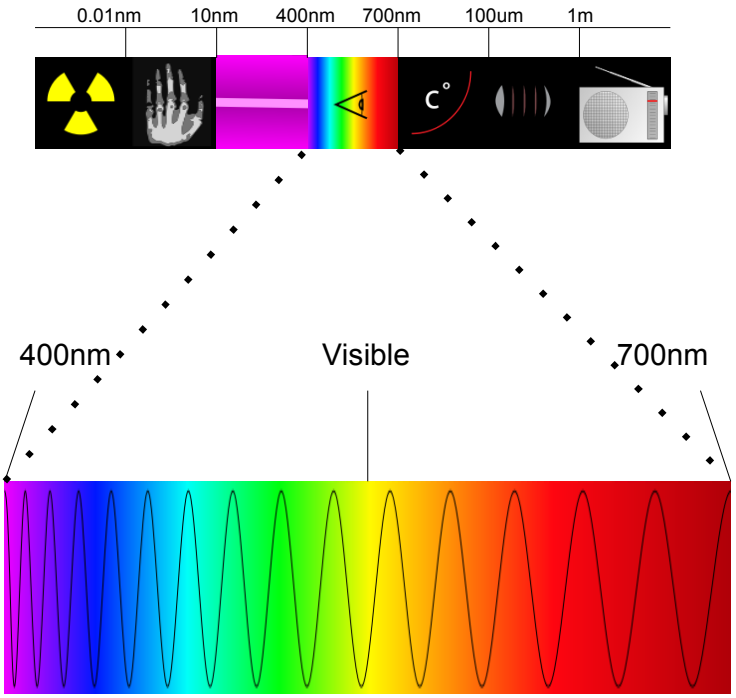


Figure 2.5: The light spectrum

2.4 Physical encoding

In computer architecture a waveform that switches between two voltage levels representing the two states of a Boolean value (0 and 1) is referred to as a digital signal, even though it is an analog voltage waveform, since it is interpreted in terms of only two levels.

A clock signal is a special digital signal that is used to synchronize digital circuits. Logic changes are triggered either by the rising edge or the falling edge of the signal.

Line coding is used within communication system for digital data transport. Line coding is represented by a digital signal to be transported by an amplitude and time discrete signal that is optimally tuned for the specific properties of the physical channel (and of the receiving equipment). The waveform pattern of voltage or current used to

represent the 1s and 0s of a digital signal on a transmission link is called line encoding. Common types of line encoding are unipolar[1], polar[2] and Manchester encoding [9]. Briefly, line coding is the process of arranging symbols that represent binary data in a particular pattern for transmission.

Following will be a brief explanation of two physical encodings, the first being Manchester Code and the second is Morse Code.

2.4.1 Manchester Code

Manchester code, also known as phase modulation (PM), is a line coding method in which the encoding of each data bit has at least one transition and occupies one clock cycle, it means that it is self-synchronising. When sending a bit through the system the sender is also sending the clock-time with each bit which result in that the receiver cannot lose bits because of system-clock variations.

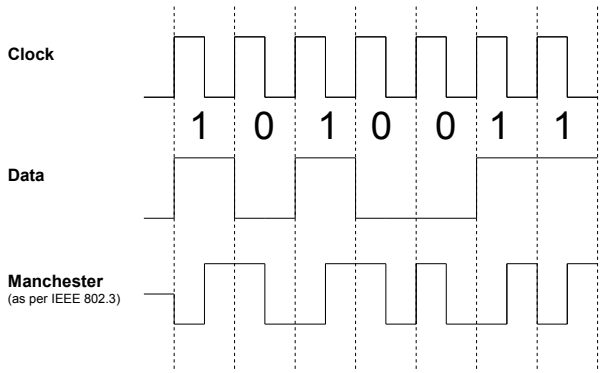


Figure 2.6: Manchester code

The coding represent binary values, 1 and 0, where 1 represent a pulse that has a positive

[1]If only one polarity of voltage level is used, positive or negative.
[2]If both polarity of voltage level is used, positive and negative.

pulse voltage during the first-half of the bit duration and a negative voltage during second half of the bit duration. The binary 0 represent a pulse that is negative during the first-half of the bit duration and positive during the second-half of the bit duration. Each bit in a data transition are coded with two bits, 01 or 10 (see figure 2.6). Manchester Code is the physical encoding used in Ethernet.

2.4.2 Morse code

A traditional physical encoding for a two-way half-duplex communication is Morse code. Morse code is a good encoding for small text transmissions since it works on time rules instead of on clock signals as for example Manchester code does. The time rules are based on the time between the off and on state of the transmitter. There are five different time rules in Morse code, these different rules are international known rules [10] and are as follows:

- Short mark, dot (.) - one "on" time unit long.
- Long mark, dash (-) - three "on" time units long.
- Intra-character gap (between the dots and dashes within a character) - one "off" time unit long.
- Short gap (between letters) - three "off" time units long.
- Medium gap (between words) - seven "off" time units long.

For example if the transmitter is on for one time unit (the time units are decided on both the receiving and the transmitting side before the transfer of information is started) and then off for one time unit the receiver decodes that signal as a short mark and then an intra character gap. These signals are then be saved for encoding at a later state and the process is repeated until a short gap signal appears. When a short gap appears all the

short-, long- and intra character signals that have been saved are decoded via the Morse to Latin/Latin to Morse translation table as the one seen in appendix A. A sent short message (SOS) and how it will be decoded can be seen in figure 2.7.

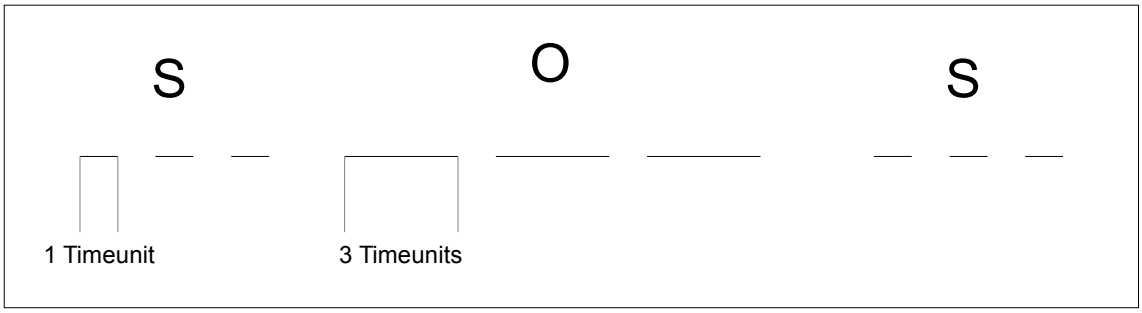


Figure 2.7: The SOS message and its signals in Morse Code.

2.5 State machine

The use of different states to decide if certain operations are viable or not during the specific state during the execution of the program is called a state machine. The basic idea is that you have different states so that when for example a program which do not have full duplex, is receiving data, it should not be able to send data. All to lower the chance of problems in the system and loss of information. The state machine is further described in section 3.2.

2.6 Related work

The following section will have a brief description of the related works to this project that already exist.

2.6.1 FSO

The systems that are used for space and satellite communication are called FSO, Free Space Optic. It was originally developed by the military and NASA and has been used for over three decades to provide fast communication links in remote areas. FSO offers full-duplex throughput and it is a line-of-sight[3] technology. It uses laser beams to provide optical connections and is capable of sending up to 1.25 Gbps of data, voice and video communication simultaneously through the air [11].

Laser communication is used all around the world, but much more often in large scale. Space and satellite communication is the area in which lasers are most often used. The reason for using laser communication is the energy saving property a laser has and that in space the chances that something will block the line of sight is minimal. [7]

2.6.2 IrDA

IrDA, Infra-red Data Association, is another wireless technology that uses the infra-red spectrum of light to send information. The signals are at a frequency that are just below what the human eye can perceive. IrDA is used mostly in the relatively short distance. The signals cannot go through walls and are susceptible to bad weather. Infra red technology is most often used in remote controls but also for data transmission in mobile phones and PDAs.

All these works and projects described above use line of sight dependent technologies. This special property is also used in this project and is a very special property since all information needs to be sent in a straight line with no obstacles in the way.

2.7 Components

The components that are used in the project are described here.

[3]Electro-magnetic waves travelling in a straight line.

2.7.1 Overview

An overview of the main components.

- Computer (USB controller): A computer with either Windows or Linux.
- Interface card: An USB controlled Interface card with both analog and digital inputs and outputs. Figure 2.8 shows the Interface card.
- Laser: A class 2 laser that is controlled by the computer through the interface card.
- Photo diode: A photo diode that control a current to the analog in port. With light emitting from the laser on to the diode the analog input gives a value depending on the intensity of the light, from 0 (0 V) when no light shines and up to 71 (1.39V) on full exposure of light on the diode (all this is with the components used in this project). This creates big enough change in the intensity of the light that hit the diode that the program can convert the different signals into information, which the computer can read and convert to a one or a zero depending on the physical encoding. Or as in the case of Morse, the amount of time the photo diode is detecting light from the laser.

2.7.2 USB Experiment Interface Board

The control of the laser and the photo diode detection is handled by a Velleman VM110/K8055 USB Interface Board. The board has the following features:

- 5 Digital inputs (0 = ground and 1 = open).
- 2 Analog inputs (0 - +5V).
- 8 Digital open collector output switches (max 50V/100mA).
- 2 Analog outputs.

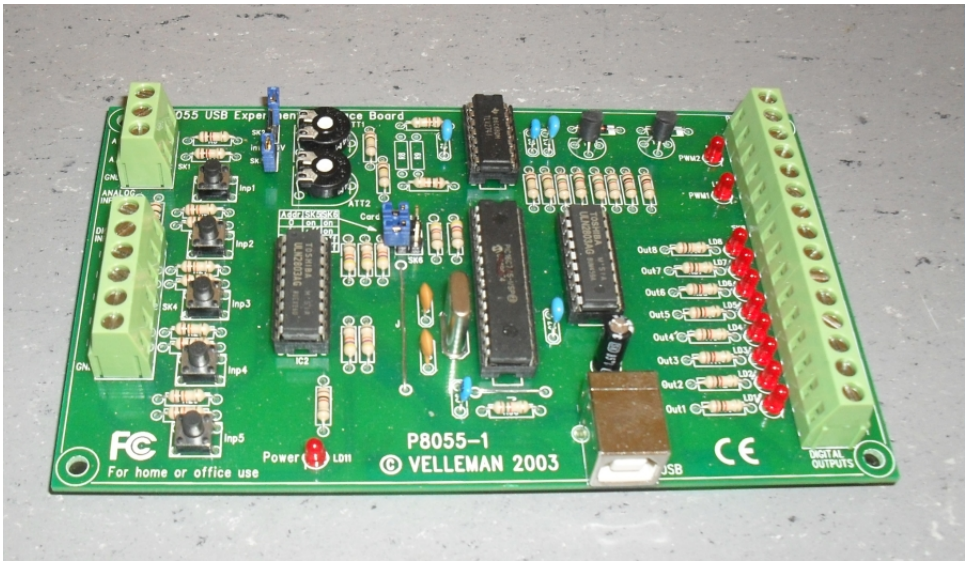


Figure 2.8: The Velleman interface card.

  - 0 to 5V (output resistance 1.5k ohm).
  - max 100mA / 40V.

- Powersupply through USB: maximum 70mA.
- Included software.
  - Diagnostic software.
  - Communication DLL.

2.7.3 USB Interface Board API

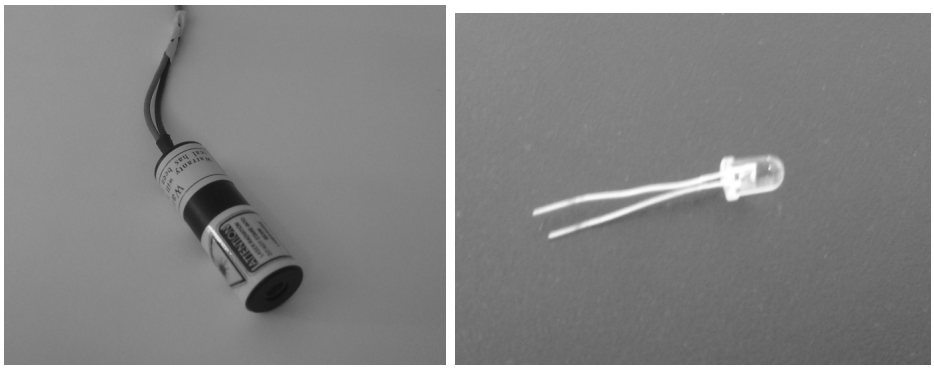
To control the USB board a program is needed to be developed that can control the interface board via the API. This program consist of several parts that help with the control of the API, from a state machine to the timers that decide which part of the functions from the API that should be called. The API consists of several control functions for the interface card. The ports being used in the project are analog-out and digital-in which results in that the only functions used in the API will be OutputAnalogChannel(Channel:

Longint, Data: Longint), ClearAnalogChannel(Channel: Longint) and ReadDigitalChannel(Channel: Longint)

Function-list with descriptions:

- Long OpenDevice(Long **ChannelAdress**):
  - **ChannelAdress:** Opens the connection between the interface card and the computer.
  - **Data:** To be able to distinguish between different cards if more than one are connected to the computer, two jumpers can be set on the interface card. This gives a possible combination of four different set-ups on the jumpers. The value 0 as a parameter is the card with both jumpers attached, 1 is the first jumper off and the second on, 2 is first jumper on and second off and 3 is both jumpers off.
  - **Description:** Sets up the connection with the interface card.
  - **Result:** If succeeded the return value will be the card address read from the K8055 hardware. Return value -1 indicated that K8055 card was not found.
- void CloseDevice():
  - **Description:** Sets all outgoing ports to 0V or digital Zero and closes the communication with the K8055 card.
- void SetDigitalChannel(Long **Channel**):
  - **Channel:** The value decides which channel to read (1 or 2).
  - **Data:** The value between 1 to 8 corresponds to the output channel that is to be set.
  - **Description:** The selected digital output channel is set.

- void ClearDigitalChannel(Long **Channel**):
  - **Channel:** The value between 1 to 8 corresponds to the output channel that is to be cleared.
  - **Description:** Sets the selected channel is cleared.
- long ReadAnalogChannel(long **Channel**):
  - **Channel:** The value decides which input channel to be read (1 or 2).
  - **Data:** The corresponding analogue to digital converter data is read.
  - **Description:** The input voltage of the selected 8.bit Analogue to Digital converter channel is converted to a value which lies between 0 (0V) and 255 (5V).



(a) Laser pointer (b) Photodiode

Figure 2.9: The Laser pointer and the photodiode.

2.7.4 Laser

(Light amplification by stimulated emission of radiation, see figure 2.9)

A mechanism for emitting electromagnetic radiation, usually in form of a visible beam of light. There are several different classes of lasers, simply named Class I, Class II and Class III (or Class IIIa) [12] [13]. There are even higher classified lasers but the size of

the power source needed to power these lasers make them hard to use as hand-held laser pointers. The most common class of laser that the average person come across in his everyday life are the class II lasers with visible light working in the wavelength of 400 to 700 nm and a power output lower then 1 mW. When you reach power levels of 1 mW or higher you start to reach levels of intensity on the laser that can be harmful to the eyes. But the natural response to protect the eyes, blink and turn the head away from the beam, will help to protect the eyes up to lasers with class 3A intensity.

Photo diode (see figure 2.9): All semiconductor diodes have light-sensitive properties but since this often is an undesired effect they are encased in light blocking materials. Photo diodes on the other hand are encased in materials that allow the light to pass through. Photo diodes are used in solar cells, photometry (the science of measurement of light) and for optical communication.

2.8 Summary

When it comes to communication many methods can be utilised, may it be drums or electrical impulses. All share the same basics, they need a medium to propagate the information in and some sort of protocol so both sides can understand each other. This project is using a light in form of a laser as the medium and a physical encoding called Morse code as the language, a self-synchronising time unit based encoding. The actual transfer of information is handled by the laser and a photo diode, both controlled by an USB Interface Card. The reason for using laser in this project is primarily for to visualizing two computers communication with each other.

3 Design

In this chapter the overall design of the project is discussed. How the construction of the hardware is made, see section 3.1. Then information about how the state machine works is discussed in section 3.2. Finally, section 3.3 discusses how the physical encoding is made.

3.1 Construction

To connect the laser and interface board a solderless breadboard is used, see figure 3.1. The construction is made simple because the main focus have been on the implementation and not the construction in this project. The interface card is placed inside a plastic box and figure 3.2 shows a complete sender/receiver component. On the side of the box a hole is made were the USB cable can connect with the interface card.

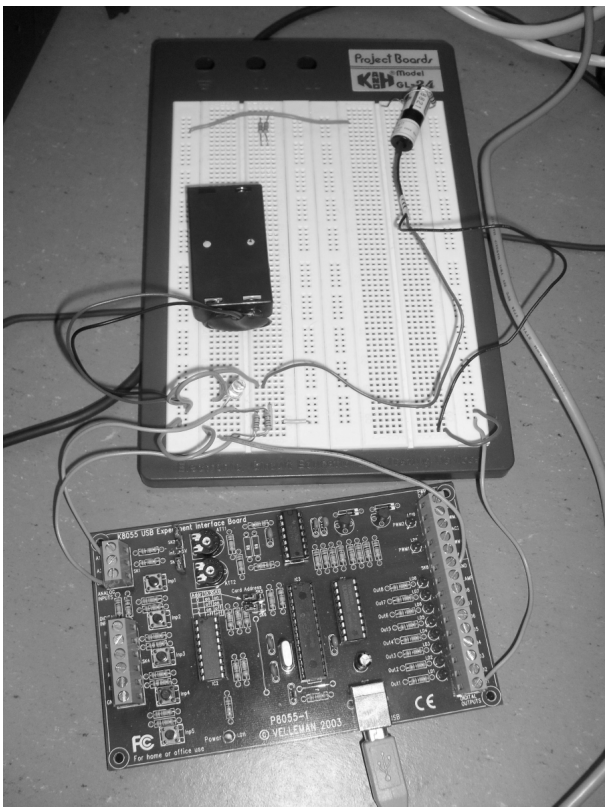
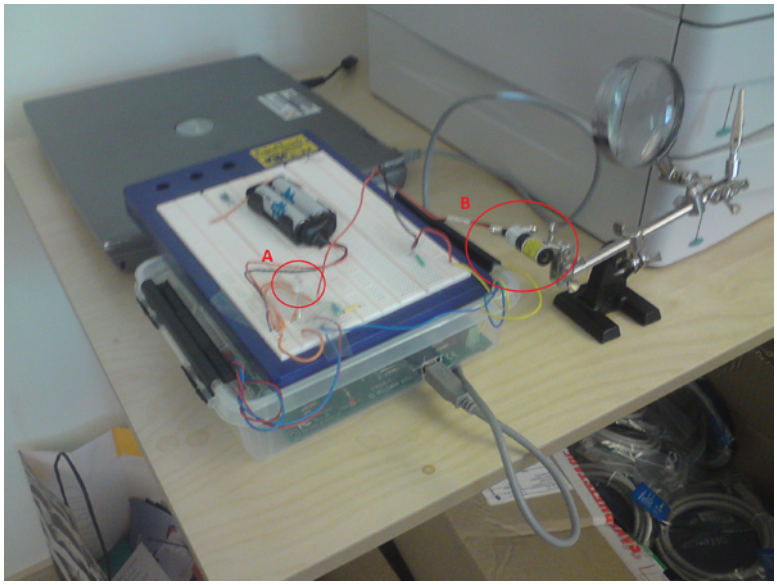


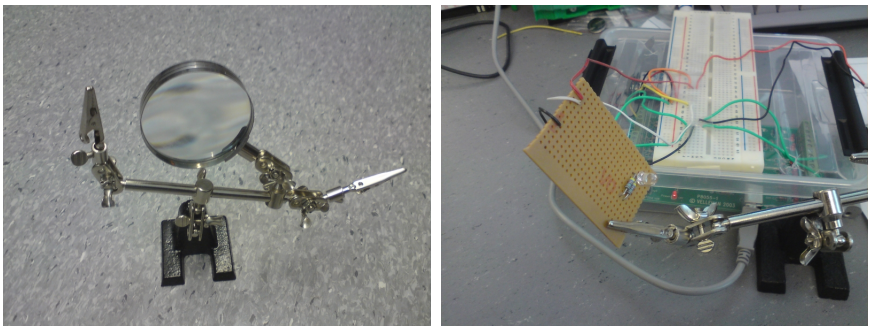
Figure 3.1: The circuit.



(a) Photo diode (b) Laser Pointer

Figure 3.2: A complete sender/receiver component.

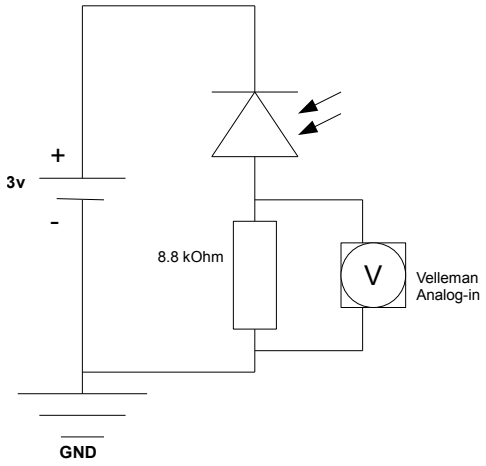
The laser is connected with the project board via copper wires lead through another hole made on the top of the box. To easily target the photo diode and laser a so-called "Helping Hand" have been used with alligator clips to hold both the diode and the laser, see figure 3.3(a). The photo diode circuit (photo diode and resistors) are soldered together on a tiny circuit board to make it easier to make the actual coupling, see figure 3.3(b).



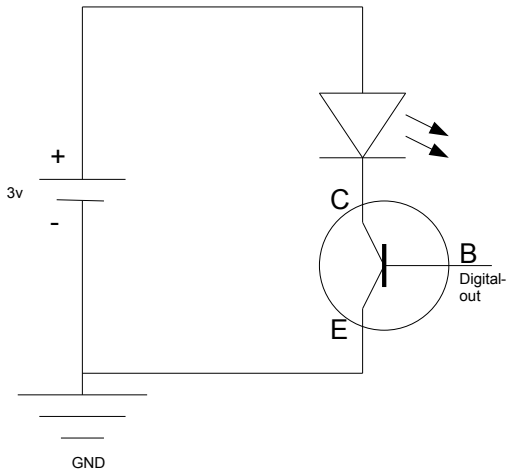
(a) Helping Hand (b) Hand Clip

Figure 3.3: A "helping hand" and a photodiode on one of the clip.

The circuit diagram for both the diode and laser can be seen in figure 3.4(a) and figure 3.4(b).



(a) The circuit diagram for the photo detector



(b) The circuit diagram for the laser.

Figure 3.4: The circuit diagrams for the diode and the laser.

See appendix B for component list and their cost.

3.2 State machine

The communication protocol are the rules that are needed to be followed so both the sender and the receiver know how to send and receive data. To implement this the rules of Morse code are needed to be followed and a state machine is needed to control the flow of the program (both which are implemented in the program).

A state machine is a virtual control mechanic used to perform certain actions depending on an event and all previous events leading up the the current state. The program, used to control the interface board via the API (see 2.7.3), uses half duplex when communicating which mean that the state machine in the program is used to prevent that the receiving side from sending during the time the sending side is sending. This mode of communication allows the program to only receive or send data at any one time. To start the state machine there is a need for a start-state, the state in which the program waits for some event to happen.

The start-state used in the program, independently if you are on the receiving or the sending side of the connection, is "Stop and Wait". In this state you may start sending information and then jump to one of the two other states that exists (sending or receiving state). In the sending state you may only encode and send the information specified by the user. To make sure that the receiving side do not send any information while the sending side is in this state the sending side will occupy the link by having the laser lit up (ButtonPressed event in figure 3.5) until the actual transmission of the data. When this occurs the sender will start out with a synchronisation blink with the laser and then send the actual information. After all information has been sent the program reverts back to the "Stop and Wait" state. The last state is the receiving state which is a state where the program can only receive data and decode it to the appropriate character. The program switches to this state when it registers that the other side of the communication link has

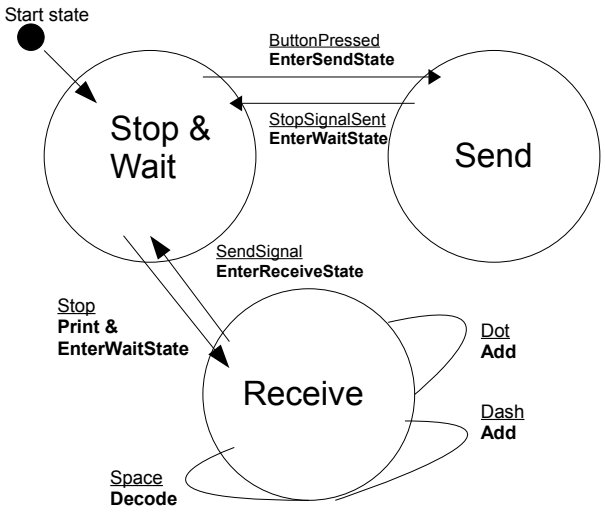


Figure 3.5: The three states of the program.

entered the sending state by lighting the laser(SendSignal event in figure 3.5). When in this state the program cannot do anything as long as the sending side have the laser lit and will stay in this state as long as needed for all the information to be sent. When all the information has been sent (using Dot Dash and Space event seen in figure 3.5) (indicated with the 20 time unit long "stop signal", Stop event in figure 3.5) the receiving side reverts back to the "waiting to receive or send" state.

3.3 Morse code as physical encoding

Except the international rules of Morse code (seen in section 2.4.2) two extra rules were needed as control rules for the interface card. The first being a synchronisation signal to reset the timers so when the actual information is sent all timers start from zero. The second rule is a stop rule telling the program to stop receiving and decode the message transferred. The two control rules looks as follows:

- Synchronisation signal - one half ”on” time unit long.
- End signal - twenty ”on” time units long. If this signal is lost the text buffer will not be reset but since this only repeats the last message again when the next text is received no fall-back has been made for this.

A photo diode was used to solve the problems of measuring the time units correctly and to determine if any light has been sent. The photo diode readout gives different readouts depending on the amount of light that hit it, and different values depending on which photo diode has been used. Zero when covered with something as thick as two regular sheets of paper, with the room light on, and 71 (1.39V, this number was specific for the diode used in this project) when directly hit with the laser. When the change in the intensity of light changed the number indicated by the photo diode with over 10 (0.02V) the time since the last change was taken and compared to the above Morse code rules. As long as a short-, long- or intra character gap signal is registered the corresponding signal is added to a buffer that will be decoded later on. This decoding will only happen when a new character signal is sent or at the end when the stop signal is sent. The decoding will then decide what character the corresponding signal was by comparing it with the already defined Morse to Latin alphabet chart (see appendix A).

3.4 Summary

In this chapter the construction and compilation of the hardware has been discussed, from the laser connected to the interface board into the plastic box to the photo diode and how the wiring that connects it all are connected. The basics of the state machine has been discussed, from what state (also know as a start-state) that will be entered at the start of the program to what state that should be entered depending on the different events that can occur. The last section has explained how Morse code has been implemented as a physical encoding in this project.

4 Implementation

This chapter explains the program from an implementation viewpoint and explains how the basics of the program is done. When the basics have been explained, from what different start-up functions are made to which state the program enters at the start, an in-depth explanation will be about how a message is transmitted. This is made through the explanation of the code used in the transceiver. When this is done an explanation of the function used to control the USB interface board is made.

At the very start of the program there is a need to check if there is a connection with the USB Interface card. This is done with by the following instructions, which also gives an error message if the connection cannot be found. See figure 4.1. It will also send back an error message if no interface card was found. After this the `signal()` function is initiated so the CTRL + C command will not only kill the process but also close the connection to the interface board. When the CTRL + C command is done the program enters the start state and then enters the event loop, polling both the photo-diode and the keyboard for events. To explain the polling and usage of photo diode and keyboard the next section will describe the course of a message.

```
1  test = OpenDevice(0);
2  if (test == -1)
3  {
4          perror("OpenDevice");
5          exit(0);
6  }
7  signal(SIGINT, sigproc);
```

Figure 4.1: Connection control.

4.1 Message transmission

The program starts of in a start state which both polls the keyboard and the photo diode to see if any changes has been made since last reading. If a change has been made at

the keyboard the program will immediately enter the sending state. The sending side will then activate the laser and send a constant emission of light to tell the receiving side to get ready to accept incoming transmission from the sending side. When this has started the sender will enter the message via the keyboard and then the message is encoded from Latin letters to Morse signals via the encode function (figure: 4.2) when the sender press the enter-key. Code[] is the already defined Latin alphabet to Morse signal struct used to convert between Latin letters and Morse signals.

```
1  void encode(char *s)
2  {
3          size_t i, j, k, l = 0;
4          char mrs[MAX];
5          mrs[0] = '\0';
6
7
8          for ( i = 0; s[i]; ++i )
9          {
10                 for ( j = 0; j < sizeof Code / sizeof * Code; ++j )
11                 {
12                         if ( toupper(s[i]) == Code[j].letter )
13                         {
14                                 strcat(mrs, Code[j].morse);
15                                 printf("%c: ", Code[j].letter);
16                                 printf("%s\n", Code[j].morse);
17                                 break;
18                         }
19                 }
20         }
21         putchar('\n');
22         sendMessage(mrs,l);
23 }
```

Figure 4.2: Encode function.

When the text to be sent has been encoded it will be sent via the `sendMessage()` function (figure: 4.3). This function will first synchronise the clocking on both sides via the sync function which sends pulses of pre-defined length light so both sides know that sending is starting. The message will then go through the morsesignal-buffer sent from the encode function and send the corresponding sequence of light depending on if its a dot

(short signal), a dash (long signal), a new character signal (represented as white space), a white space (represented as underscore) or the stop signal at the end of the message. When this is done the sending side reverts back to the Stop and Wait state and just waits for a new event.

```
1  void sendMessage(char msg[], size_t length)
2  {
3          size_t i;
4          int time;
5          printf("Size of message: %d\n", length);
6
7          sync(); //Syncronize, set up timer
8
9          for(i=0; i < length; i++)
10         {
11                 if(msg[i] == '.')
12                         shortLight();
13                 else if (msg[i] == '-')
14                         longLight();
15                 else if (msg[i] == ' ')
16                 {
17                         newletterLight();
18                 }
19                 else if (msg[i] == '_')
20                 {
21                         newwordLight();
22                 }
23         }
24         printf("Last Letter sent\n");
25         printf("Message: %s was sent\n", msg);
26         endLight();
27 }
```

Figure 4.3: sendMessage function.

4.2 Message reception

When the sending side starts sending any signal (transmit light) the receiving side enters the receive state and then waits for the sending side to transmit. As soon as the light intensity threshold is reached on the photo diode a timer is clocked and compared to the

different set of time rules already defined in the program. Depending on what time rule is matched different states will be generated, see event example 4.4. The if-statements in this code segment are the actual time rules for the off value of the laser.

```
1  gettimeofday(&t3, NULL);
2  diff = timediff(t3, t4);
3  gettimeofday(&t4, NULL);
4  printf("The light was off for %lld useconds\n", diff);
5
6  if ( diff > SPACE_SIGMIN && diff < SPACE_SIGMAX )
7  {
8          printf("\tSignal was WHITESPACE\n");
9          return SPACE_SIG;
10 }
11 else if ( diff > INTRA_CHARMIN && diff < INTRA_CHARMAX )
12 {
13         printf("\tSignal was Intra-Character\n");
14         return IDLE;
15 }
16 else if ( diff > NEW_CHARMIN && diff < NEW_CHARMAX )
17 {
18         printf("\tSignal was New-Character\n");
19         return CHAR_SPACE;
20 }
```

Figure 4.4: Event example.

The events corresponds to the different signal sent by the sending side. From short signals to the stop signal. As soon as the newchar event is hit the now Morse coded character is decoded and added to a buffer via the decode function (figure 4.5).

Then the whole process is repeated and the textbuffer storing the already decoded Morse signals is printed by entering the idle function when a stop-event is called.

It is also possible to stop the program if wished for with the CTRL and C key combination. This starts the sigproc-function (figure 4.7). In this function the connection to the USB Interface board is closed and then the program exits like normal.

```
enum state decode(const char *morse)
{
        size_t j;
        for ( j = 0; j < sizeof Code / sizeof *Code; ++j )
        {
                size_t size = strlen(Code[j].morse);
                if ( memcmp(Code[j].morse, &morse[0], size) == 0 )
                {
                        strcat(text, Code[j].letter);
                        morsesig[0] = '\0';
                        printf("text = %s \n", text);
                        return RECIEVE_STATE;
                        break;
                }
        }
        strcat(text, '\$');
        morsesig[0] = '\0';
        return RECEIVE_STATE;
}
```

Figure 4.5: Decode function.

```
enum state idle(const char *placeholder)
        {
                addnewcharsignal(morsesig);
                morsesig[0] = '\0';
                printf("Message recieved: %s \n", text);
                text[0] = '\0';
                return WAIT_STATE;
        }
```

Figure 4.6: Idle function.

```
void sigproc()
{
        ClearAllAnalog();
        CloseDevice();
        exit(0);
}
```

Figure 4.7: Sigproc function.

4.3 Control of the interface board

For the actual control of the Interface card an API was needed and was linked to sourceforge.net from the Velleman homepage[4]. After downloading and installing it the use of the

[4] http://www.velleman.eu/distributor/home/?country=se&lang=en

above described functions was available and since the project only use the digital output ports and the analog in port no more functions were needed. To manage these functions a toggle function was needed to control the laser, see figure 4.8.

```
1  void ToggleDigitalChannel(long c) {
2          static int cur = 0;
3
4          if (cur == 0) {
5                  ClearDigitalChannel(c);
6                  cur = 1;
7          } else {
8                  SetDigitalChannel(c);
9                  cur = 0;
10         }
11
12 }
```

Figure 4.8: Toggle function.

4.4 Summary

The basic process of the transmission of a message have now been explained, from the creation of Morse-code signals translated from regular characters via the encode-function to the detection of the signals and decoding of the signals back to characters.

5 Results

In this chapter the results of the project are discussed starting with a walk-through of the goal list from the first chapter. After that, there will be a discussion of what kind of tests have been used in the project to evaluate the system, and last there will be a short evaluation of the tests.

5.1 The goal-list

Follow-up of the goal-list from the introduction. This list describe what has been learned, how this experience has been gained and if the goals of the project have been reached.

1. **Reading and understanding the API of the interface board:** After just a few hours of searching and reading all the knowledge needed to control the systems hardware were known.

2. **Control laser pointer beam from computer:** After the initial connection to the interface board was done the actual control of the laser was made by connecting the laser to a 3V battery and the interface board via an open collector transistor giving the interface board the control of the laser. The laser was then controlled with the Set- and ClearDigitalChannel function described above (see section: 2.7).

3. **Detect active/inactive beam from computer:** When all the hardware was connected to the interface board the photo-diode was read using the `ReadAnalogChannel()` function. The photo-diode was positioned (so it blocked the current) between the analog-in channel one and the ground on the interface board. The analog-in channel is now a basic voltmeter, being able to read the voltage over the photo diode. When light shines on the diode it will stop blocking the current and a voltage can be read (more light is equal to higher voltage up to 1.39V for the photo diode used in this

project). The current is the same 3V battery used to power the laser. For more information on the photo diode see section 2.7.

4. **Transmit a simple bit pattern:** Since Morse code was used no actual bit-pattern was sent but instead a regular character using Morse code. The transmission was just an encoding with a defined Morse code alphabet(appendix A), using the Morse code time rules seen in section 2.4.2.

5. **Detect a bit-pattern at the receiver:** The detection of a bit-pattern, or as in this case, Morse code was made by comparing the time difference between the signals the sender sent. The most important thing done here was lining up the laser so it hit the photo-diode.

6. **Morse code communication:** Since the encoding and decoding from the two previous goals was working, the Morse code communication was up and running as soon as the two previous paragraphs were working.

7. **Decide on physical encoding:** After a brief discussion the use of Morse code as physical encoding was made over Manchester code because of the way Morse code is used to communicate. Since this project is more about the visual aspect of communication, Morse code felt much more comprehendible to see and understand then just simple on and off signals that Manchester would give.

8. **Implement physical encoding and decoding:** Morse code was decided to be the physical encoding in the project. The physical encoding was implemented as soon as the project progressed to this goal on the goal-list.

9. **Make one-way chat application:** The one-way chat application is working if only one laser is aligned and this is the way the program works now and how the program worked before the merging of the sending and receiving parts of the program.

10. **Simple performance test:** With the first communication up, some simple transmission were made with just some basic text sent which went through correctly. There were no problems in finding the actual limit of the system but because of the time pressure at the end of the project the bottleneck in the system could not be found.

11. **Increase distance and see limits:** 20 cm (centimetres) was the first test that was made. In this test no detection problems were found and the test range was increased to one meter. In the one meter test no problems were detected and a further increase of the range was made to five meter and here some interesting results started to show. The amount of light that hit the photo diode started to get so low that certain combinations of signals such as "H" started to get divided into "E" and "S". This was solved by increasing the time unit time to 50 milliseconds which resulted in almost no errors in the decoding. Due to time pressure any longer distances then five meter were not possible to make.

12. **Increase speed and see limits:** With the communication up and working a speed test could be made with ease and the lowest number one time unit (one "dot" signal etc.) in the system could found at 35 milliseconds with the exception of longer distances of five meters or more. When any higher speed were tried data started to get lost.

13. **Implement two-way communication and chat application:** The merging was easy as soon as the state machine was in place and working.

5.2 Tests

To validate the reliability of the communication-link several different range tests were made. These tests varied from 20 centimetres up to 5 meters. There was also a speed test to see what the max bandwidth was on the link.

Test 1 (20 cm): The first test made was a regular distance test of 20 cm. This test was made with a 100 000 microseconds (100 milliseconds) time unit at first. This worked without any problem or information loss. Since the distance of this test was fairly short, a speed test to see the bandwidth on the link was made since the distance would not be a problem for the detection. By stepping down from 100 milliseconds to 50 milliseconds the hardware and software clearly had no problem in detecting the laser. From here the the time unit (tu) were lowered to only 35 milliseconds and all worked fine. But if the time units were lowered any more than 35 milliseconds the receiver started to have problem detecting the transmitted signals and thus information was lost.

Test 2 (100 cm): Now the biggest problem of laser communication became apparent, the need of millimetre precision to hit the photo diode. The problem became quite obvious as soon as both lasers needed to be aligned to each photo diode, but on only 1 meter it only took about 1 to 2 min to line up the lasers and the photo diodes. When the Laser was lined up (how to line up, see section 6) several different messages was sent to test the different combinations of signals that could be sent. All to see if there was any loss of data on the increased range. As expected on this short distance no difference at all was detected and the messages was sent correctly.

Test 3 (500 cm): As soon as the range started to increase the amount of light that hit the photo diode dropped to levels that gave effect on the signals sent by the sending side. The solution to this was to increase the time units from 35 milliseconds to 50 milliseconds. This removed the most of the errors but still signals that looked the same such as several "H" characters (four "dot" signals) in a row gave some errors were one part of the "H" were translated into "S" (three "dot" signals) and the other into "E" (one "dot" signal).

Assumption: With no time to build a proper receiving unit with a lens to gather the laser so it would hit the diode spot on an assumption has been made. The assumption is that the reason information is lost due to the laser not being concentrated enough, since the laser is not a perfect laser and spreads with increased range. The spread of the laser

only looked at by the naked eye with the different distances can be seen on picture 5.1. The lens on the helping hands equipment would have been possible to use it would be needed to be removed from the base of the helping hands to be able to align it in front of the diode.

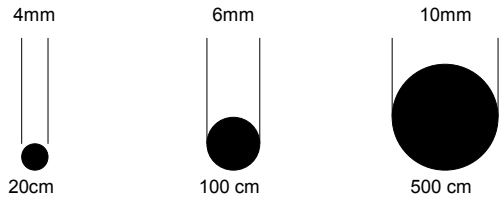


Figure 5.1: The three different sizes of the beam depending on range.

5.3 Evaluation

With the program (see the code in appendix C and appendix D) working and the tests made a pattern emerged pretty quickly. As predicted the laser worked as long as it was aligned and concentrated. As soon as the time unit length were increased the hardware had more time to detect the change on the photo diode which gave a more accurate readout from the photo diode. With the quality the lasers being somewhat low the range of the communication could not be fully tested since some sort of lens will be needed to concentrate the laser for longer distances. The laser as a medium to send information visually has been a great technology since it can use light and still operate at longer distances.

5.4 Summary

In this chapter the results of the project has been discussed. The range and speed limits of the software and hardware have been narrowed down through tests and the evaluation of the tests. There are different limits in the system, the biggest being the laser since the beam of the laser widens with only a couple of meters of distance. Another limit of the system is the time needed to control and detect the laser beam. The transmission- and detection time determine the throughput of the link. It was found that a minimum beam power-on delay of 35 milliseconds was needed for the receiver to correctly detect the beam. This theoretically limits the throughput to one bit per $2 * 0.035$ seconds, or $1/2 * 0.035 \approx 14$ bits per second (assuming Manchester encoding with two transitions per bit). The major part of the delay was experimentally determined to be caused by the USB communication or the internal operation of the USB card. Consecutive readings of the analog port showed that 20 ms was spent from the API function call to its return.

6 User manual

This section is a description of how to set-up and how to use the laser communication system. The components that are used to make the set-up are described in appendix B.

6.1 Set-up

Start of by deciding the general positions of the lasers. Since the USB cable only is 1 meter long a position close to a computer is needed. When a position is chosen for the laser two things are needed to be taken into consideration. First of all the laser is fairly hard to adjust vertically, with the current construction, this makes it hard to collect the light that the laser emits if the height of the laser has been adjusted too high or too low. A position that has a free line of sight to the receiver is necessary (and very important).

When the position of the both transceivers (receiver and sender are called transceiver, the name is a combination of transmitter and receiver) is set the first thing that needs to be done is to point the laser in a rough direction towards the other transceiver. When this is done on both sides the photo diodes should be moved so the plastic of the diode blocks the beam, the beam can be turned on by starting the program and pressing any key on the keyboard for easier positioning. When the position of the diode is aligned so that it captures as much light as possible the programs can be started (if they are not already on) to test the link. One side can start by entering the sending state and just block the beam to see if any messages are printed on the console window on the receiving side. This means that if the light is hitting the photo diode and then is blocked a message it will be given on the screen for how long the beam was on (and off if the beam is applied to the photo-diode again). If nothing happens check if the laser is actually hitting the photo diode (and is emitting light). When the laser is lined up correctly it is time to start communication via the laser-link. This is done by simply pressing any key on the keyboard and then typing in the desired message and sending it with the ENTER-key. As long as one side is in the

send state, by pressing any key on the keyboard, the receiving side cannot send and will have to wait for the message to be received from the sending side.

6.2 Check-list (step by step-list)

1. Decide positions of the lasers (Remember line of sight and height positioning).
2. Point the lasers in the direction of the receiver (Roughly).
3. Move the photo-diode into the laser beam.
4. Test the connection by starting the program on both sides and enter the send-state. If no connection is made redo the previous step and continue to this one again.
5. Start communicating by pressing any key on the keyboard, type the desired message, and press enter.

7 Conclusion

This chapter contains the conclusions of the project, why this project was made and what have been learned and also a future work section to describe what can be improved/changed on the project.

7.1 The purpose of the project

When the project was finished, the major goal had been reached, the possibility to send and receive data in a visual appealing way in a two-way communication. To achieve this, knowledge in a wide span of both technical and theoretical fields needed to be touched, from electrical circuits and program code (in this case C) to the physics behind electromagnetic radiation (light).

7.2 Experience

This project has touched on several technical subjects. From the construction of the electrical circuits and code needed to control the laser and read the photo diode to the basic physics behind light. Regarding the software implementation and program code, the experience learned here was the possibility to set the flags controlling if a command should block the process of the program or not, and how to use a state-machine to control the flow of the program depending on which state and which specific event that happens. Reading and understanding Manchester-code and the implementation of Morse code and how to use it as a physical encoding gave a good insight in how different encoding protocol works. The use of flags was a crucial part of the program to be able to enter different states without stopping the actual polling of the photo diode. The state machine was useful to stop the program from being able to enter the sending state when it was receiving and also to decode the different signals to what they were supposed to be, from short signals to the stop signal. Since the project was all about lasers a greater understanding of this

technology was attained. From the dangers of lasers to the big differences between all the classes of lasers. Also learned is how and were to utilise the special property of the laser, from space communication to simple straight line open air communication.

The project worked out well and could easily be used to show how data-communication works even thou some basic computer knowledge is needed to get the program up and working.

7.3 Future

With the main goal of the program completed, several new ideas of how to improve and develop the program were invented. Since the project time limit was closing in fast, the theory behind these ideas are only barely touched. The following section gives a short description on these ideas and gives suggestions for implementation.

Light collector: The first idea that should be implemented is a light collector. A device that basically consist of a convex lens that gathers the light and concentrates it to the photo diode. This is needed since the laser is not a perfect laser. A perfect laser is a laser which do not spread the light it emits. This means that the laser will always have the same size on the beam independent on the distance. Since the the photo diode needs a minimum amount of energy in form of light landing on it to detect any change this will be a problem when the laser is used in longer distance (5 meters or more with the laser used in this project) communication.

Diode casing: Build some sort of casing for the photo diode to block the non-wanted incoming light (such as roof lamps, desktops lamps and the sun). This to make the detection of the laser easier since less energy is needed to reach the threshold for the detection.

Manchester-Code: Implement the use of Manchester code as an alternative encoding to Morse code. The purpose of using Manchester code is to make the system resemble a Ethernet connection. This can be made by switching out the time rules of the Morse code to different time rules which are based on either the ”on and off” or ”off and on”

combination of signals and only wait for two signals a time.

Acknowledgements: Create acknowledgements (acks) for the receiver to send using checksums on each message to create a reliable connection.

Bottlenecks: Search the system for bottlenecks. If the bottlenecks can be found in the system future upgrades can be made to the hardware and software to be able to send information at a higher rate. This can be made by doing a simple program that calculates the time it takes from doing different operations such as calculating the average time it takes for the system to complete a USB system call.

Construction: Create a more robust and easier to handle support for the laser and the photo diode. This will make the process of lining-up easier and to make the construction more stable, lowering the risk of breaking the connection when it is up and working.

7.4 Last Thoughts

With the project completed and the program working the only thing that is needed to be done now is the process of setting up the link easier. That the communication is working and that the number of bugs were so low and easy to fix was something that felt good. Generally speaking the project worked out well and the problems in the project were almost non-exsistent compared to what was believed on beforehand.

References

[1] A. Michael Noll. *Principles of modern communications technology*. Artech House, 2001. http://books.google.com/books?id=6tDEZlwiMK0C.

[2] Carl Nassar. *Telecommunications demystified*. Newnes, 2001.

[3] Anton A. Huurdeman. *The worldwide history of telecommunications*. Wiley-IEEE, 2003. http://books.google.com/books?id=SnjGRDVIUL4C.

[4] Friedrich Georg Wieck and Otto Wilhelm Aolund. *Uppfinningarnas bok*. Andra bandet. L. J. Hierta, 1874. http://runeberg.org/uppfinn/2/.

[5] Mary Kay Carson. *Alexander Graham Bell: Giving Voice to the World*. Sterling, 2007. http://books.google.ca/books?id=a46ivzJ1yboC.

[6] Nobelprize.org. The official web site of the nobel prize, [retrieved 2010-05-25]. http://nobelprize.org/nobel_prizes/physics/laureates/.

[7] David E. Smith. Two-way laser link over interplanetary distance. *Science*, (311):53, 2006. http://www.sciencemag.org/cgi/content/abstract/sci;311/5757/53.

[8] Hamid Hemmati. *Near-Earth Laser Communication*. CRC Press, 2009. http://books.google.com/books?id=hiFPrD55GD8C.

[9] Jerry D. Gibson. *The mobile communications handbook*. Number 2. CRC Press, 1999. http://books.google.se/books?id=9gmfVq1Jt8wC.

[10] United States. War Dept. *International Morse code*. Govt. Print. Off, 1945. http://books.google.com/books?id=gsJRGwAACAAJ&dq.

[11] FSO. Official website, [retrieved 2010-05-25]. http://www.freespaceoptics.org/freespaceoptics/default.cfm.

[12] Jerry D. Gibson. *Safety of Laser Products. Equipment Classification and Requirements*. B S I Standards, 2007.

[13] FDA U.S. Food and Drug Administration. Laser products and instruments, 2010. http://www.fda.gov/Radiation-EmittingProducts/RadiationEmittingProductsandProcedures/HomeBusinessandEntertainment/LaserProductsandInstruments/default.htm.

A Morse alphabet

Letter	Code	Letter	Code
A	. -	1	. - - - -
B	- . . .	2	. . - - -
C	- . - .	3	. . . - -
D	- . .	4	. . . . -
E	.	5	. . . . .
F	. . - .	6	- . . . .
G	- - .	7	- - . . .
H	. . . .	8	- - - . .
I	. .	9	- - - - .
J	. - - -	0	- - - - -
K	- . -	?	. . - - . .
L	. - . .	!	. . - - .
M	- -	,	- - . .
N	- .	.	. - . - . -
O	- - -	=	- . . . -
P	. - - .	-	- . . . . -
Q	- - . -	(	- . - - .
R	. - .	)	- . - - .
S	...	+	. - . - .
T	-	@	. . . - . -
U	. . -		
V	. . . -		
W	. - -		
X	- . . -		
Y	- . - -		
Z	- - . .		

B Component list

Prices in the table is what the components cost when they were purchased.

Component	Description	Cost/unit	Units	Subtotal
Laser pointer	Red, 1.5 mW	SEK 199	2	SEK 398
Velleman Interface board	VM110 complete, K8055	SEK 599	2	SEK 1198
IR-photo diode	Wavelength 700nm	SEK 10	2	SEK 20
Solders breadboard		SEK 79	2	SEK 158
Helping hand	With magnifying glass	SEK 69	2	SEK 138
Resistors		SEK 19	2	SEK 76
			Total cost	SEK 1991

C Morse.h

```
1  #ifdef linux
2  #define LASERCOM
3
4  #include <stdio.h>
5  #include <unistd.h>
6  #include <signal.h>
7  #include <stdlib.h>
8  #include <string.h>
9  #include <ctype.h>
10 #include <SDL/SDL.h>
11 #include <K8055DLL.h>
12 #include <sys/time.h>
13 #include <fcntl.h>
14 #include <termio.h>
15
16
17 #define MAX 255
18 #define TIMECONST 50000
19
20 #define INTRA_CHAR 1
21 #define NEW_CHAR 3
22 #define SPACE_SIGDEF 7
23 #define SHORT_SIGDEF 1
24 #define LONG_SIGDEF 3
25 #define STOP_SIG 20
26
27 #define INTRA_CHARMIN (INTRA_CHAR - (INTRA_CHAR/3))*TIMECONST
28 #define INTRA_CHARMAX ((NEW_CHAR - INTRA_CHAR)/2 + INTRA_CHAR)*TIMECONST
29
30 #define NEW_CHARMIN (NEW_CHAR - (NEW_CHAR - INTRA_CHAR)/2)*TIMECONST
31 #define NEW_CHARMAX (NEW_CHAR + (SPACE_SIGDEF - NEW_CHAR)/2)*TIMECONST
32
33 #define SPACE_SIGMIN (SPACE_SIGDEF - (SPACE_SIGDEF - NEW_CHAR)/2)*TIMECONST
34 #define SPACE_SIGMAX (SPACE_SIGDEF + (SPACE_SIGDEF - (SPACE_SIGDEF - NEW_CHAR)/2))*
       TIMECONST
35
36 #define SHORT_SIGMIN 0.8*TIMECONST
37 #define SHORT_SIGMAX (SHORT_SIGDEF + (LONG_SIGDEF - SHORT_SIGDEF)/2)*TIMECONST
38
```

```
39 #define LONG_SIGMIN (LONG_SIGDEF - (LONG_SIGDEF - SHORT_SIGDEF)/2)*TIMECONST
40 #define LONG_SIGMAX (LONG_SIGDEF + (STOP_SIG - LONG_SIGDEF)/2)*TIMECONST
41
42 #define STOP_SIGMIN (STOP_SIG - (STOP_SIG - LONG_SIGDEF)/2)*TIMECONST
43 #define STOP_SIGMAX (STOP_SIG + (STOP_SIG - (STOP_SIG - LONG_SIGDEF)/2))*TIMECONST
44
45 struct termios stored_settings;
46
47 static const struct
48 {
49         const char letter, *morse;
50 } Code[] =
51 {
52         { 'A', ".- " },{ 'B', "-... " },{ 'C', "-.-. " },{ 'D', "-.. " },
53         { 'E', ". " },{ 'F', "..-. " },{ 'G', "--. " },{ 'H', ".... " },
54         { 'I', ".. " },{ 'J', ".--- " },{ 'K', ".-.- " },{ 'L', ".-.. " },
55         { 'M', "-- " },{ 'N', "-. " },{ 'O', "--- " },{ 'P', ".--. " },
56         { 'Q', "--.- " },{ 'R', ".-. " },{ 'S', "... " },{ 'T', "- " },
57         { 'U', "..- " },{ 'V', "...- " },{ 'W', ".-- " },{ 'X', "-..- " },
58         { 'Y', "-.-- " },{ 'Z', "--.. " },{ '1', ".---- " },{ '2', "..--- " },
59         { '3', "...-- " },{ '4', "....- " },{ '5', "..... " },{ '6', "-.... " },
60         { '7', "--... " },{ '8', "---.. " },{ '9', "----. " },{ '0', "----- " },
61         { '?', "..--.. " },{ '!', "..--. " },{ ',', "--..-- " },{ '.', ".-.-.- " },
62         { '=', "-...- " },{ '-', "-....- " },{ '(', "-.--. " },{ ')', "-.--.- " },
63         { '+', ".-.-. " },{ '@', "...-.- " },{ ' ', "_" },
64 };
65
66 void reset_keypress(void);
67 void set_keypress(void);
68 char getch();
69 void sigproc();
70 enum state decode(const char *morse);
71 void ToggleDigitalChannel(long c);
72 long long timediff(struct timeval t2, struct timeval t1);
73 enum state addnewcharsignal();
74 enum state stop();
75 enum state sendstate();
76 enum state addspacesignal();
77 enum state addlongsignal();
78 enum state addshortsignal();
79 enum state nop();
```

```
80 enum state idle();
81 enum event getEvent();
82
83 void sigproc();
84 void sync();
85 void sendMessage(char* msg, size_t length);
86 void encode(char *s);
87 void ToggleDigitalChannel(long c);
88 long long timediff(struct timeval t2, struct timeval t1);
89 void endLight();
90 void newletterLight();
91 void newwordLight();
92 void shortLight();
93 void longLight();
94 void send();
95
96 #endif
```

D Morse.c

```
1
2  #include "morse.h"
3
4  enum event { SHORT_SIG = 0, LONG_SIG, SPACE_SIG, CHAR_SPACE, IDLE, STOP, SEND };
5  enum state { WAIT_STATE=0, RECEIVE_STATE, SEND_STATE};
6
7  enum event e;
8  enum state s;
9  char morsesig[MAX];
10 char text[MAX];
11 struct timeval t1, t2, t3, t4 = {0,0};
12 double in=0;
13 double oldin=0;
14 int quit = 0;
15
16 typedef enum state (*fp)(const char *placeholder);  /* deklarera funktionspekartyp */
17
18 /* tabell med eventhanterare  */
19 fp eventhandlers[][7] =  {{addshortsignal, addlongsignal, addspacesignal, nop, nop, stop,
       sendstate}, /* state WAIT_STATE */
20                         {addshortsignal, addlongsignal, addspacesignal, addnewcharsignal,
                               nop, stop, sendstate}, /* RECEIVE_STATE */
21                         {nop, nop, nop, nop, nop, idle, nop}}; /* SEND_STATE */
22
23
24
25 // Called on ctrl+C
26 void sigproc()
27 {
28         int flags;
29         ClearAllDigital();
30         CloseDevice();
31         printf("\n Digital out: Off \n Connection to USBInterfaceCard: off \n");
32         flags = fcntl(0, F_GETFL, 0); /* get current file status flags */
33         flags &= !O_NONBLOCK;            /* turn off blocking flag */
34         fcntl(0, F_SETFL, flags);
35         exit(0);
36 }
37
```

```
38 // Takes the complete morsesignal buffer and decodes it to the corresponding message
39 enum state decode(const char *morse)
40 {
41         size_t j;
42         printf(" IN DECODE MORSE BUFFER: %s\n", morse);
43         for ( j = 0; j < sizeof Code / sizeof *Code; ++j )
44         {
45                 size_t size = strlen(Code[j].morse);
46                 if ( memcmp(Code[j].morse, &morse[0], size) == 0 )
47                 {
48                         text[strlen(text)+1] = '\0';
49                         text[strlen(text)] = Code[j].letter;
50                         morsesig[0] = '\0';
51                         printf("text = %s \n", text);
52                         return RECEIVE_STATE;
53                         break;
54                 }
55         }
56         text[strlen(text)+1] = '\0';
57         text[strlen(text)] = '$';
58         morsesig[0] = '\0';
59         return RECEIVE_STATE;
60 }
61
62 // Sets the digital channel to off if on and vice versa
63 void ToggleDigitalChannel(long c) {
64         static int cur = 0;
65
66         if (cur == 0) {
67                 ClearDigitalChannel(c);
68                 cur = 1;
69         } else {
70                 SetDigitalChannel(c);
71                 cur = 0;
72         }
73
74 }
75
76 // Return the time difference in microseconds between t1 and t2
77 long long timediff(struct timeval t2, struct timeval t1)
78 {
```

```
79    return 1000000LL*(t2.tv_sec-t1.tv_sec) + t2.tv_usec-t1.tv_usec;
80  }
81
82  // Exits the program
83  enum state stop(const char *placeholder)
84          {
85                  putchar('\n');
86                  text[0] = '\0';
87                  return WAIT_STATE;
88          }
89
90  // Enters the receivestate
91  enum state sendstate(const char *placeholder)
92          {
93                  if( s != RECEIVE_STATE)
94                  {
95                          return RECEIVE_STATE;
96                          printf("Entering receivestate!\n");
97                  }
98          }
99
100 // Adds a spacesignal at the end of the morsebuffer
101 enum state addspacesignal(const char *placeholder)
102         {
103                 morsesig[strlen(morsesig)+1] = '\0';
104                 morsesig[strlen(morsesig)] = ' ';
105                 decode(morsesig);
106                 morsesig[strlen(morsesig)+1] = '\0';
107                 morsesig[strlen(morsesig)] = '_';
108                 decode(morsesig);
109                 return RECEIVE_STATE;
110         }
111
112
113 // Adds a new char character to the morsebuffer
114 enum state addnewcharsignal(const char *placeholder)
115         {
116                 morsesig[strlen(morsesig)+1] = '\0';
117                 morsesig[strlen(morsesig)] = ' ';
118                 decode(morsesig);
119                 return RECEIVE_STATE;
```

```
120         }
121
122 // Adds a long character to the morsebuffer
123 enum state addlongsignal(const char *placeholder)
124         {
125                 morsesig[strlen(morsesig)+1] = '\0';
126                 morsesig[strlen(morsesig)] = '-';
127                 return RECEIVE_STATE;
128         }
129
130 // Adds a short character to the morsebuffer
131 enum state addshortsignal(const char *placeholder)
132         {
133                 morsesig[strlen(morsesig)+1] = '\0';
134                 morsesig[strlen(morsesig)] = '.';
135                 return RECEIVE_STATE;
136         }
137
138 // Does nothing/waits
139 enum state nop(const char *placeholder)
140         {
141                 /* Stay in the current state */
142                 return s;
143         }
144
145 enum state idle(const char *placeholder)
146         {
147                 morsesig[0] = '\0';
148                 printf("Message received: %s \n", text);
149                 text[0] = '\0';
150                 return WAIT_STATE;
151         }
152
153 // Gets the event for the event handler
154 enum event getEvent()
155 {
156         long long diff;
157         int flags;
158         oldin = in;
159         /* Reception */
160         in = ReadAnalogChannel(1);
```

```
161         if (abs(oldin-in) > 10)  /* only react to sudden changes */
162         {
163                 if (in > oldin)
164                 {
165                         gettimeofday(&t3, NULL);
166                         diff = timediff(t3, t4);
167                         gettimeofday(&t4, NULL);
168                         printf("The light was off for %lld useconds\n", diff);
169
170                         if ( diff > SPACE_SIGMIN && diff < SPACE_SIGMAX )
171                         {
172                                 printf("\tSignal was WHITESPACE\n");
173                                 return SPACE_SIG;
174                         }
175                         else if ( diff > INTRA_CHARMIN && diff < INTRA_CHARMAX )
176                         {
177                                 printf("\tSignal was Intra-Character\n");
178                                 return IDLE;
179                         }
180                         else if ( diff > NEW_CHARMIN && diff < NEW_CHARMAX )
181                         {
182                                 printf("\tSignal was New-Character\n");
183                                 return CHAR_SPACE;
184                         }
185                 }
186                 else
187                 {
188                         gettimeofday(&t3, NULL);
189                         diff = timediff(t3, t4);
190                         gettimeofday(&t4, NULL);
191                         printf("The light was on for %lld useconds\n", diff);
192                         if ( diff > STOP_SIGMIN && diff < STOP_SIGMAX )
193                         {
194                                 printf("\tSignal was STOP\n\n");
195                                 return STOP;
196                         }
197                         else if ( diff > SHORT_SIGMIN  && diff < SHORT_SIGMAX )
198                         {
199                                 printf("\tSignal was SHORT\n", diff);
200                                 return SHORT_SIG;
201                         }
```

```
202                         else if ( diff > LONG_SIGMIN && diff < LONG_SIGMAX )
203                         {
204                                 printf("\tSignal was LONG\n", diff);
205                                 return LONG_SIG;
206                         }
207                 }
208
209
210
211                 oldin=in;
212
213
214         }
215
216         if(getch() != EOF && s != RECEIVE_STATE)
217         {
218                 flags = fcntl(0, F_GETFL, 0); /* get current file status flags */
219                 flags &= !O_NONBLOCK;           /* turn off blocking flag */
220                 fcntl(0, F_SETFL, flags);
221
222                 send();
223
224                 flags = fcntl(0, F_GETFL, 0); /* get current file status flags */
225                 flags |= O_NONBLOCK;            /* turn on blocking flag */
226                 fcntl(0, F_SETFL, flags);
227         }
228
229         if(in > 55 && s != RECEIVE_STATE)return SEND;
230
231         return IDLE;
232 }
233
234 /********************** SEND ************************/
235
236 //Sends the message msg[] with the length size_t
237 void sendMessage(char msg[], size_t length)
238 {
239         size_t i;
240         int time;
241         printf("Size of message: %d\n", length);
242
```

```
243         sync(); //Syncronize, set up timer
244
245         for(i=0; i < length; i++)
246         {
247                 if(msg[i] == '.')
248                         shortLight();
249                 else if (msg[i] == '-')
250                         longLight();
251                 else if (msg[i] == ' ')
252                 {
253                         newletterLight();
254                 }
255                 else if (msg[i] == '_')
256                 {
257                         newwordLight();
258                 }
259         }
260         printf("Last Letter sent\n");
261         printf("Messeage: %s was sent\n", msg);
262         endLight();
263 }
264
265 // Sends a quick synchronisation signal to reset timers before the actuall message is sent
266 void sync()
267 {
268         int time = 0.2 * TIMECONST;
269         ToggleDigitalChannel(1);
270         usleep(time);
271         ClearDigitalChannel(1);
272         usleep(time);
273 }
274
275 // Encodes the message s to morsecode so it can be sent over the link
276 void encode(char *s)
277 {
278         size_t i, j, k, l = 0;
279         char mrs[MAX];
280         mrs[0] = '\0';
281
282
283         for ( i = 0; s[i]; ++i )
```

```
284         {
285                 for ( j = 0; j < sizeof Code / sizeof * Code; ++j )
286                 {
287                         if ( toupper(s[i]) == Code[j].letter )
288                         {
289                                 strcat(mrs, Code[j].morse);
290                                 l = l + strlen(Code[j].morse);
291                                 printf("%c: ", Code[j].letter);
292                                 printf("%s\n", Code[j].morse);
293                                 break;
294                         }
295                 }
296         }
297         putchar('\n');
298         mrs[l+1] = '\0';
299         sendMessage(mrs,l);
300 }
301
302 // The last signal sent before termination of the sending cycle
303 void endLight()
304 {
305         int time = STOP_SIG * TIMECONST;
306         SetDigitalChannel(1);
307         usleep(time);
308         ClearDigitalChannel(1);
309 }
310
311 // Sent when the a new letter is found in the buffer
312 void newletterLight()
313 {
314         int time = 3 * TIMECONST;
315         printf("New Letter\n");
316         ClearDigitalChannel(1);
317         usleep(time);
318 }
319
320 // Sent when a whitespace is found in the buffer
321 void newwordLight()
322 {
323         int time = 4 * TIMECONST;
324         printf("New Word\n");
```

```
325         ClearDigitalChannel(1);
326         usleep(time);
327 }
328
329 // Sent when a '.' is found in the buffer
330 void shortLight()
331 {
332         printf("Shortlight \n");
333         int time = 1 * TIMECONST;
334         SetDigitalChannel(1);
335         usleep(time);
336         ClearDigitalChannel(1);
337         usleep(time);
338 }
339
340 // Sent when a '-' is found in the buffer
341 void longLight()
342 {
343         printf("Longlight \n");
344         int time = 3 * TIMECONST;
345         SetDigitalChannel(1);
346         usleep(time);
347         time = 1 * TIMECONST;
348         ClearDigitalChannel(1);
349         usleep(time);
350 }
351
352 void send()
353 {
354         char sendtext[MAX];
355
356         SetDigitalChannel(1);
357         printf("Enter the message you want to send and then press Enter: ");
358         fgets(text, MAX, stdin);
359
360         ClearDigitalChannel(1);
361         encode(text);
362         ClearDigitalChannel(1);
363 }
364
365 //Resets the flags for input
```

```
366
367 void reset_keypress(void) {
368     stored_settings.c_lflag |= 16;
369     tcsetattr(0,TCSANOW,&stored_settings);
370 }
371
372 //Sets the flags to be ready for input
373 void set_keypress(void) {
374     struct termios new_settings;
375     tcgetattr(0,&stored_settings);
376     new_settings = stored_settings;
377     new_settings.c_lflag &= (~ICANON);
378     new_settings.c_cc[VTIME] = 0;
379     tcgetattr(0,&stored_settings);
380     new_settings.c_cc[VMIN] = 1;
381     new_settings.c_lflag&=0x15;
382     tcsetattr(0,TCSANOW,&new_settings);
383 }
384
385 //Returns a char without using ECHO
386 char getch()
387 {
388   char c;
389   set_keypress();
390   c=getchar();
391   reset_keypress();
392   return c;
393 }
394
395 int main( int argc, char** argv )
396 {
397         long test = 0;
398         int flags = fcntl(0, F_GETFL, 0); /* get current file status flags */
399         flags |= O_NONBLOCK;            /* turn off blocking flag */
400         fcntl(0, F_SETFL, flags);
401
402         if(argc < 2)
403         {
404                 printf("Usage: ./morse # (where # is the interface ID)\n");
405                 exit(0);
406         }
```

```
407             gettimeofday(&t4, NULL);
408             signal(SIGINT, sigproc);
409             test = OpenDevice(atoi(argv[1]));
410             if (test == -1)
411             {
412                     perror("OpenDevice");
413                     exit(0);
414             }
415             in = ReadAnalogChannel(1);
416             s = WAIT_STATE; //Start state
417             while (!quit)
418             {
419                     e = getEvent();
420                     s = eventhandlers[s][e](morsesig);
421             }
422
423             return 0;
424     }
```