Bern University
of Applied Sciences

Inductive Short-Range Communication Channel

Course of study	Bachelor of Science in Computer Science
Author	Severin Kaderli
Advisor	Dr. Reto König
Expert	Thomas Jäggi

Version 1.0.0 of January 17, 2023

► Engineering and Computer Science

Abstract

In this thesis, I researched an alternative short-range wireless communication method that uses magnetic induction instead of conventional radio waves for transmission of data, and pointed out the advantages and disadvantages of magnetic induction as a communication channel.

For demonstrating the communication method, I created a prototype implementation of a simple communication protocol that utilizes magnetic induction and that can be run using components found in everyday devices such as a smartphone and a laptop.

At the end, I benchmarked the prototype implementation under multiple conditions and configurations to illustrate how feasible and robust the communication is and if it is viable as an alternative to radio wave communication.

Contents

1. Introduction

In our lives we use many communication technologies that work over a short range: for connecting our mice and keyboards to our computers, listening to music wirelessly and to pay contactlessly. Those technologies include, but are not limited to: Bluetooth, Radio-frequency identification (RFID), Near-field communication (NFC) and many others.

What many of those technologies have in common is that they work on top of radio waves that utilize electromagnetic induction. Some problems of radio waves are the difficulty of passing through conductive materials such as water or metals [1] and they are susceptible to interference [2] and jamming [3]. An alternative to those technologies would be communication over magnetic induction.

Other research projects in this area already exist and some have covered the use of the magnetic fields emitted from CPUs as a communication channel [4] and other cover the general use of magnetic induction communication in body area networks [5].

This thesis researched the concept of communication using magnetic induction and created a prototype implementation of a protocol that works through attempted manipulation of magnetic fields using a CPU in a laptop called MagSend. A simple user interface in the form of a website is created that allows a user to send messages over the protocol and an Android application is used to receive messages using the protocol.

2. Preliminaries

2.1. Magnetic Induction

Magnetic Induction (MI), also known as magnetic flux density or simply the magnetic field, is a physical quantity measured in Tesla T. The movement of electric charges in a conductor produces a magnetic field around the conductor. This phenomena also happens when in a CPU [6].

In Figure 2.1 the difference of the magnetic field of a CPU in idle (first 10 seconds) and under 100% load can be seen.

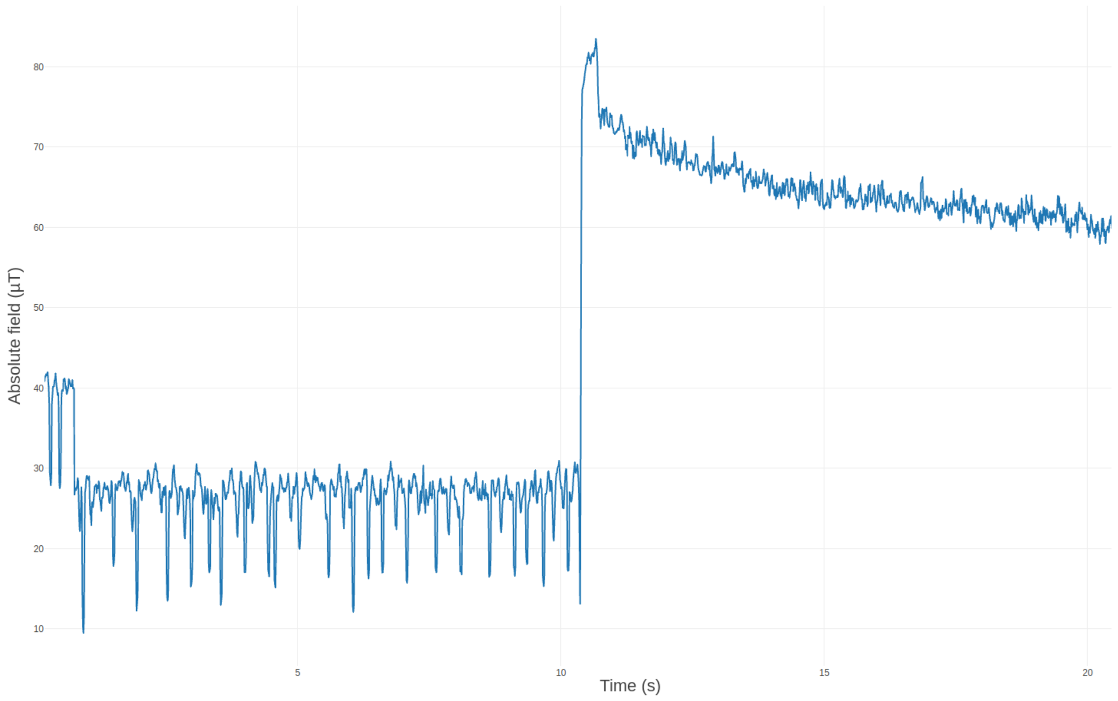


Figure 2.1.: Magnetic Field of a CPU

2.2. Hall Effect Sensor

A Hall Effect Sensor is a type of sensor to measure the strength of magnetic fields. This type of magnetometer is found in many smartphones today [7] and is generally used for compass capabilities and for positioning [8].

It utilizes the so-called hall effect to do so. The hall effect is the voltage between two sides of a current-carrying conductor inside a magnetic field [9, 10].

2.3. Thermal Throttling

CPUs utilize a lot of power when under load, and thus they get hotter. When they get too hot, they throttle by reducing clock speeds. This process is designed to protect the CPU from overheating [11].

When stress is put on the CPU, the CPU begins to throttle and the resulting CPU loads gets dampened after a while (see Figure 2.2).

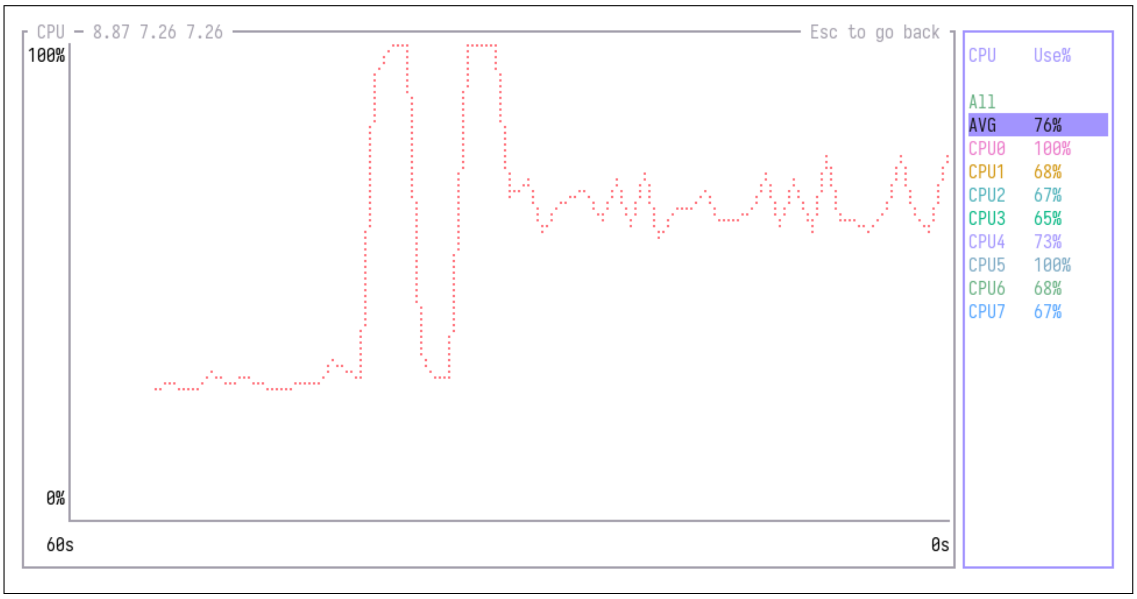

```
CPU - 8.87 7.26 7.26                         Esc to go back
100%                                                          CPU    Use%
0%                                                            All
60s                                                      0s   AVG    76%
                                                              CPU0   100%
                                                              CPU1   68%
                                                              CPU2   67%
                                                              CPU3   65%
                                                              CPU4   73%
                                                              CPU5   100%
                                                              CPU6   68%
                                                              CPU7   67%
```

Figure 2.2.: CPU Utilization with Thermal Throttling

2.4. CRC

Cyclic Redundancy Code (CRC) is a family of error detection codes designed to check if a message has been corrupted during transmission. To check if any errors ocurred during transmission, the checksum that was transmitted with the message, and the checksum that is calculated from the received message are compared, if they don't match, the message contains an error.

For calculating a CRC value, the message, which is treated as a binary number, is divided by another value, the so-called generator polynomial. The remainder of this division is the checksum value. The order of the generator polynomial defines the bit-length of the resulting checksum [12].

Depending on the chosen generator polynomial, different groups of errors can be detected [13]:

- A single, and double bit errors can always be detected
- Depending on the polynomial, more bit errors can be detected inside specific message lengths
- Using an n-bit generator polynomial can detect burst errors of up to n-bits in length
- A generator polynomial with an even number of terms can detect errors with an odd number of bits

Because of these small variations, there are many implementations of CRC algorithms in use and defined by different parties [14].

The specific algorithm used in this thesis is the CRC-8-AUTOSAR algorithm with the polynomial $x^8 + x^5 + x^3 + +x^2 + x + 1$ and an initial value of `0xFF`. This algorithm was created by AUTOSAR [15] and according to the CRC Polynomial Zoo, it can detect up to 4 single bit errors in a message length of 119 bits [16]. As our maximum payload size is 128-bit, this is a fitting CRC algorithm to use.

CRC is widely used in networks for detecting transmission errors and is simple to implement. [17]

2.5. Manchester Code

Manchester is a simple encoding code for the transmission of binary data. The data is encoded in the transitions of the signal instead of simple high or low values. A transition from low to high indicates a 0 and a transition from high to low indicates a 1 in the original definition by G. E. Thomas. In the definition from IEEE the values of the transitions are swapped (see Figure 2.3).

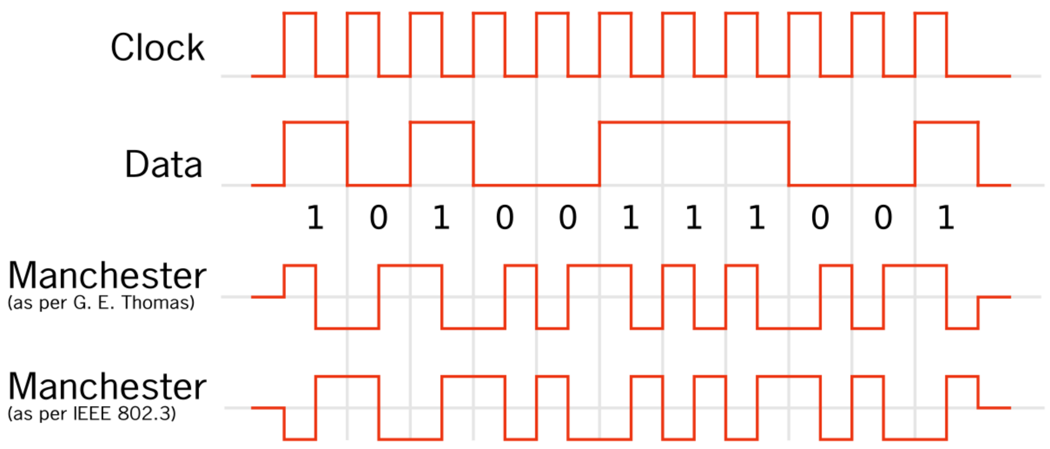


Figure 2.3.: Manchester Code[18]

The advantages of manchester encoding are, that there are no prolonged high or low signals. Furthermore, the original clock signal can be extracted from the signal and doesn't need to be synchronized between the sender and receiver. On the other side the bandwidth requirements are doubled as two code bits in the signal are needed to encode one bit of the original data [19].

For the work in this thesis, I use the definition of Manchester Code by G.E Thomas.

2.6. Web Worker

Web Workers are a web API that allows running JavaScript code in background threads in the browser. This way you can run CPU intensive processes without stalling the main thread and locking up the UI.

There are a few limitations with Web Workers. A Web Worker doesn't have access to the Document Object Model (DOM), to the window object, and to some web APIs. The communication between the main thread, and the web workers occurs via messages [20].

3. Personas

This chapter describes the personas with their motivations, goals, and problems that they want to have solved. These motivations are then later used to create and describe the requirements in chapter 4.

Alice
Alice wants to be able to receive a small piece of data over a website, similar to a QR code, with her smartphone but without other people being able to see the code over her shoulders. She also would like to be able to receive data to her smartphone while it has connection available, for example when it's in airplane mode.

Charlie
Charlie would like to enable two-factor authentication (2FA) on a website. To do so, he normally needs to scan a Quick Response (QR) code to register the website to his authentication app, but the camera on his smartphone is broken. Thus, he wants to be able to receive the 2FA registration token wirelessly to his smartphone without the use of his camera.

4. Requirements

4.1. Functional Requirements

ID	F01
Title	Calibration on Website
Application	Website
Description	The user visits the website and starts the calibration mode on the website. While in calibration mode, the website continuously puts load on the CPU.

Table 4.1.: Functional Requirement 01

ID	F02
Title	Calibration on App
Application	Smartphone App
Description	In the app, the user can start the calibration process. By putting their smartphone on different locations on their laptop, the app will tell them the current strength of the magnetic field. The point with the strongest magnetic field is the optimal place for data transmission.

Table 4.2.: Functional Requirement 02

ID	F03
Title	Text Entry
Application	Website
Description	The website contains a text input field. The user can enter any ASCII string in this text input.

Table 4.3.: Functional Requirement 03

ID	F04
Title	Sending Data
Application	Website
Description	The user visits the website, enter a text and then clicks on the send button. The website will attempt to send the string in the input field using MI. The data will be sent continuously until the user stops the sending by clicking on the stop button.

Table 4.4.: Functional Requirement 04

ID	F05
Title	Receiving Data
Application	Smartphone App
Description	The user can start the receiving mode in the app. By putting their smartphone on the laptop, they can receive the data that is sent by the website. The app displays the received text on the screen.

Table 4.5.: Functional Requirement 05

4.2. Non-Functional Requirements

ID	NO1
Title	Working in Airplane Mode
Description	The user can receive data using the app on their smartphone, even if they have put their phone in airplane mode and normal connectivity is not working.

Table 4.6.: Non-Functional Requirement O1

ID	NO2
Title	Transfer Rate
Description	The transfer speed between the laptop and smartphone should at least be 1 bit/s. That means an ASCII character should take 8 seconds to transmit.

Table 4.7.: Non-Functional Requirement O2

5. Solution

The solution consists of two separate applications: a website used for transmitting data, and a smartphone application used for receiving data. Transmission between the two application happens by a simple protocol on top of signals that use magnetic induction. I call the system MagSend.

5.1. Protocol

The following section describes how the MagSend protocol works. First I explain how the data gets encoded on the sender site, then the transmission, and finally how the data is received.

5.1.1. Sending

This section explains how a user-provided message gets encoded and finally transmitted using MI.

Data over MagSend gets sent in a packet. For the structure of the packet I tried to follow other packet formats that already exist. Most of the structure is similiar to the structure of a LoRa packet. [21] I choose LoRa as example as I am familiar to it was a part of my studies and as LoRa is also a protocol designed with a low data-rate in mind. The packet contains three parts.

Header

The first part of the packet is the header. The header is 4 bit long and contains the length of the payload in bytes. The length is mainly used to detect the end of the payload.

Payload

Next is the actual payload. As the length of the payload is specified in the header, the payload is limited to a size of 2^4 bytes. While the contents of the payload can be any bit stream, the sender and receiver should assume that the bit stream is interpreted as ASCII text.

Checksum

For verifying the integrity of the payload, a checksum of the payload is calculated and placed at the end of the packet. The algorithm used is the 8-bit CRC-8-AUTOSAR algorithm (see section 2.4).

Encoding

The contents of the packet is then encoded using a manchester code (see section 2.5) for transmission.

Transmission

The transmission works over magnetic induction. By putting a load on a CPU the resulting magnetic field can be changed. High signals should be interpreted as a 1 and low signals should be interpreted as a 0.

To indicate the start of a message in the signal, a preamble is used. The preamble used in this protocol is 3 high signals, followed by 3 low signals, followed by 3 high signals. The duration of one signal in the preamble is 1 second. That means the entire preamble takes 9 seconds.

After the preamble, the contents of the manchester encoded packet is sent. The entire transmission is repeated until the user stops the process.

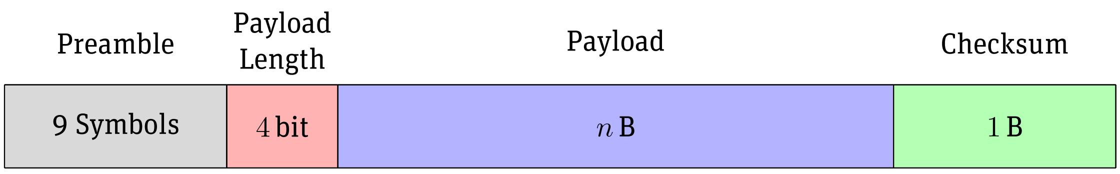


Figure 5.1.: Packet Structure, with Preamble at the Start

The following pseudocode in Listing 5.1 explains step by step how a message should get encoded and transmitted using MagSend.

```
// Define constants
preamble = [1, 1, 1, 0, 0, 0, 1, 1, 1]

// Set clock speed
clock_speed = 500

// Get the text from the user
text = get_input()

// Convert the ASCII text to a bit stream
payload = ascii_to_bits(text)

// Calculate the length of the payload in bytes
```

```
header = byte_length(payload)

// Calculate the CRC-8-AUTOSAR checksum of the payload
checksum = calculate_checksum(payload)

// Combine the header, payload, and checksum into a packet
packet = to_bits(header) + payload + to_bits(checksum)

// Encode packet using manchester encoding
encoded_packet = manchester_encode(packet)

while transmission_not_stopped:
    // Start the transmission by sending the preamble
    for symbol in preamble:
        if symbol is 1:
            stress_cpu(1000)
        else:
            idle_cpu(1000)

    // Transmit the manchester encoded bit stream of the packet
    for code_bit in encoded_packet:
        if code_bit is 1:
            stress_cpu(clock_speed)
        else:
            idle_cpu(clock_speed)
```

Listing 5.1: Pseudocode for Sending

Example

As an example the message ”Test” is shown in the MagSend packet structure in Figure 5.2.

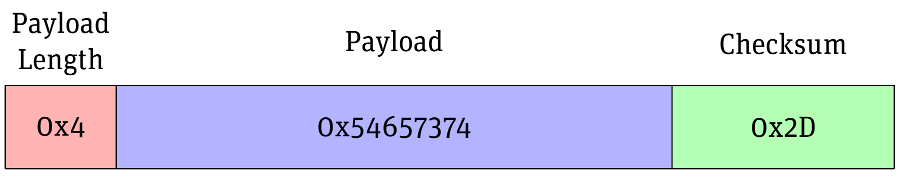


Figure 5.2.: Packet Structure of the message ”Test”

When that previous packet is transmitted, the CPU utilization should look similiar to the one in Figure 5.3 with the included preamble.

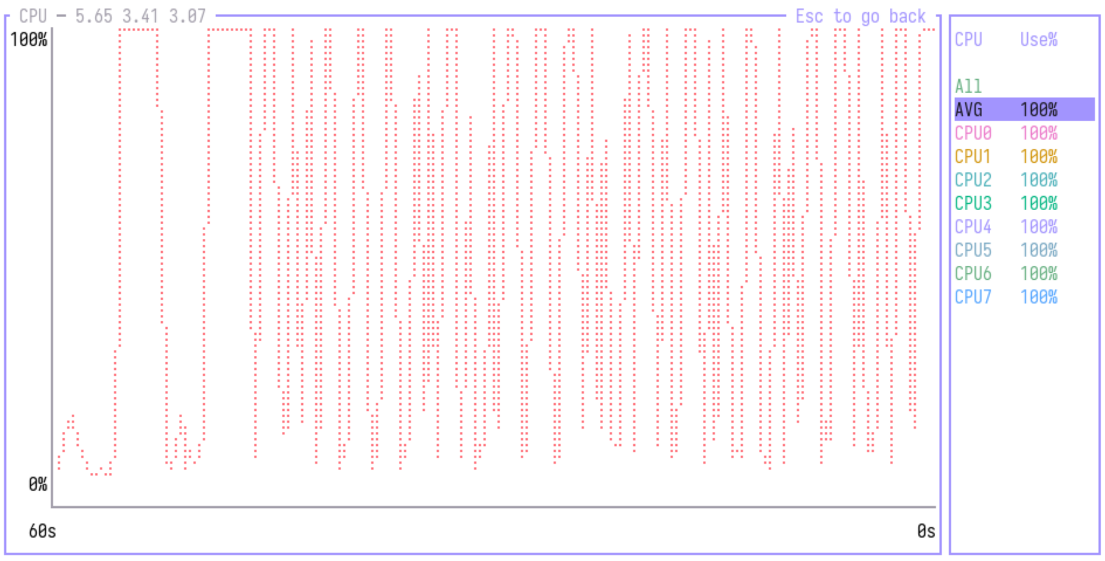


Figure 5.3.: CPU Utilization while sending a Packet

5.1.2. Receiving

The receiver utilizes a magnetometer to measure the magnetic field and interprets the received sensor values as a signal.

The following pseudocode in Listing 5.2 explains step by step how the signal should get interpreted and decoded using MagSend.

```
// Define constants
preamble = [1, 1, 1, 0, 0, 0, 1, 1, 1]

// Get sensor values
sensor_values = get_data_from_sensor()

value_range = range_of_data(sensor_values)

// Determine for sensor values if they are a high or low signal
signal = []
for value in sensor_values:
    // Compare the value with the median of the values with some threshold
    if value < median_with_threshold(value_range):
        signal += low
    else if value > median_with_threshold(value_range):
        signal += high
    else:
        // Discard value

// Filter out single high or low signals
signal = clean_signal(signal)

// Only keep the first of consecutive same signals, as the transitions hold the data
signal = compress_signal(signal)

```

```
// Check for preamble in data
if contains(signal, preamble):
    // Decode the received manchester encoded bitstream
    packet = signal[index(preamble)]

    // Read the first four bits of the bitstream to get the header
    header = read_bits(packet, 0, 4)
    display_header(header)

    // Read the next n bytes of the bitstream, according to the payload length, to get the payload
    payload = read_bits(packet, 4, header)
    display_payload(payload)

    // Read the next byte to get the CRC-8-AUTOSAR checksum of the payload
    checksum = read_bits(packet, 4 + header, 8)
    show_checksum_result()
```

Listing 5.2: Pseudocode for Receiving

Example

In Figure 5.4 the same packet from the sending example can be seen. This time it is the MI signals that the smartphone is receiving from the magnetometer.

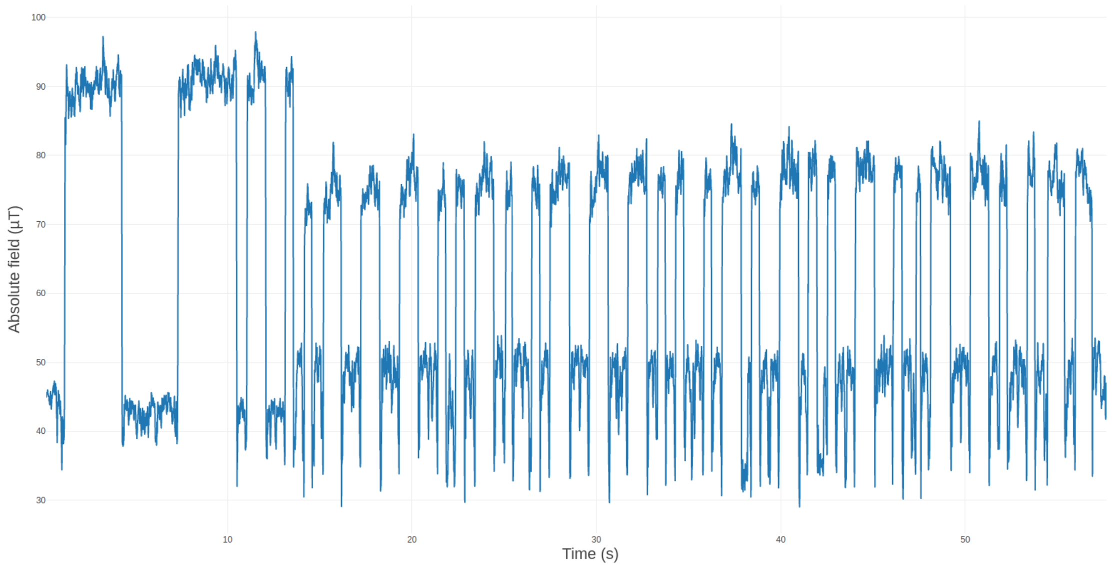


Figure 5.4.: Magnetic Induction Signal while receiving a Packet

5.2. Website

The website provides the user with an interface to transmit text over the MagSend protocol.

When the user opens the website they are greeted by a simple interface. It contains an input field for entering text and two buttons. The first button will send the entered text, and the second one will start the calibration process.

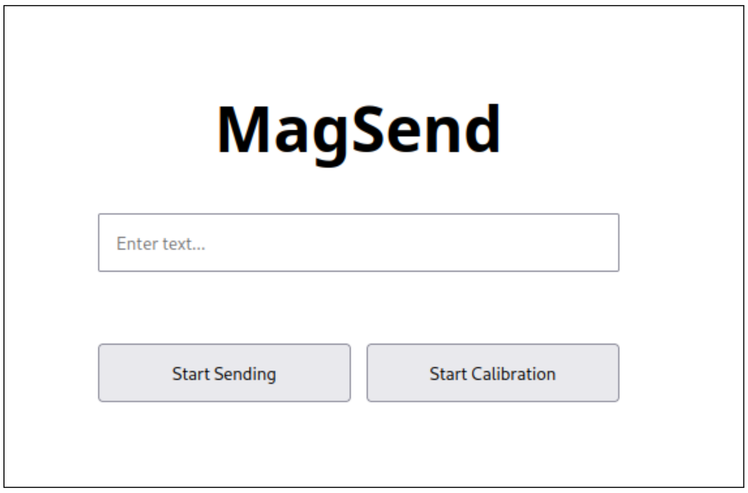
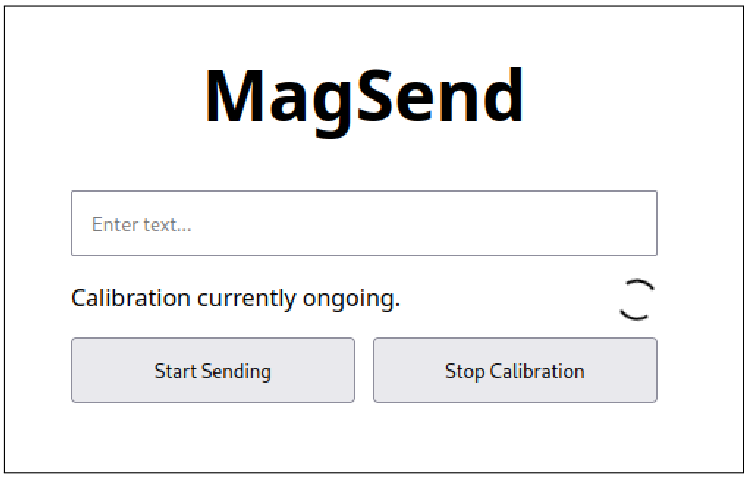


Figure 5.5.: Website - Home

When the calibration process is started, a message is displayed, so it's clear that we are in the calibration process. What happens during calibration is, that the website puts a full load on the CPU for the duration of the calibration. The actual calibration process happens in the smartphone application.



Calibration currently ongoing.

Figure 5.6.: Website - Calibration

When the user presses the sending button, the text will be sent by the website using the protocol described in section 5.1, by putting stress on the CPU to generate a signal. A message is displayed that indicates that a sending process is ongoing.

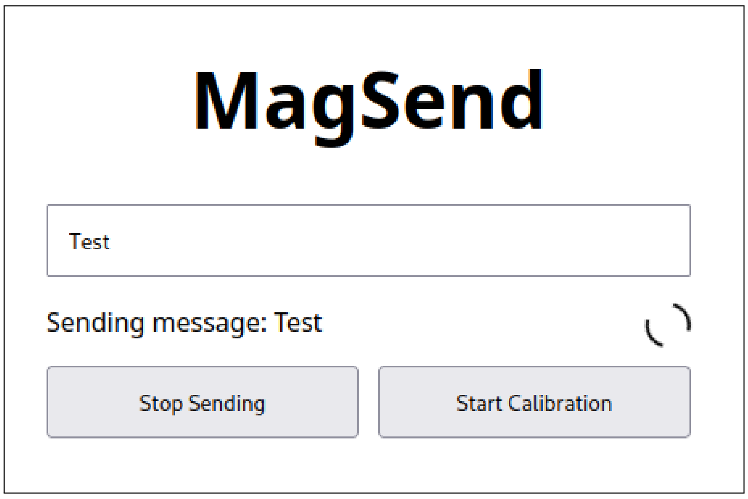
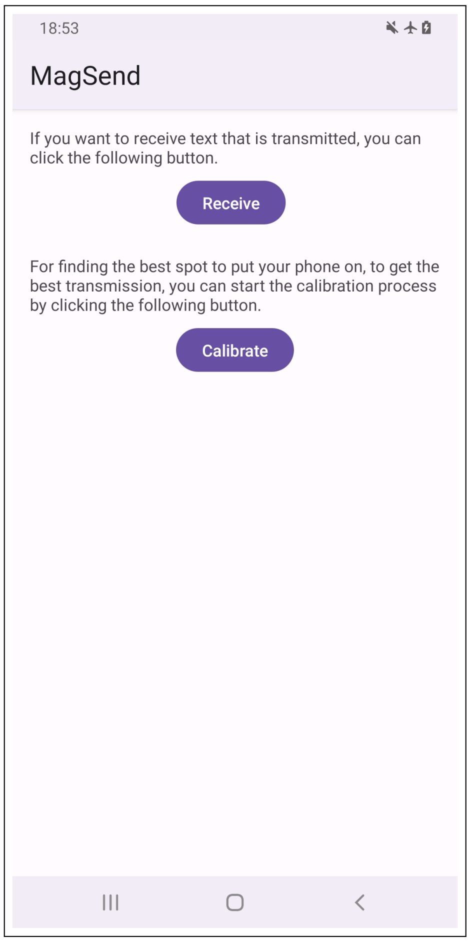


Figure 5.7.: Website - Sending

5.3. App

The app allows the user to receive messages over the MagSend protocol.

On the homescreen you have two buttons. One button is for receiving messages and one is to start the calibration process. By clicking one of the buttons the user will be switched to the corresponding screen.

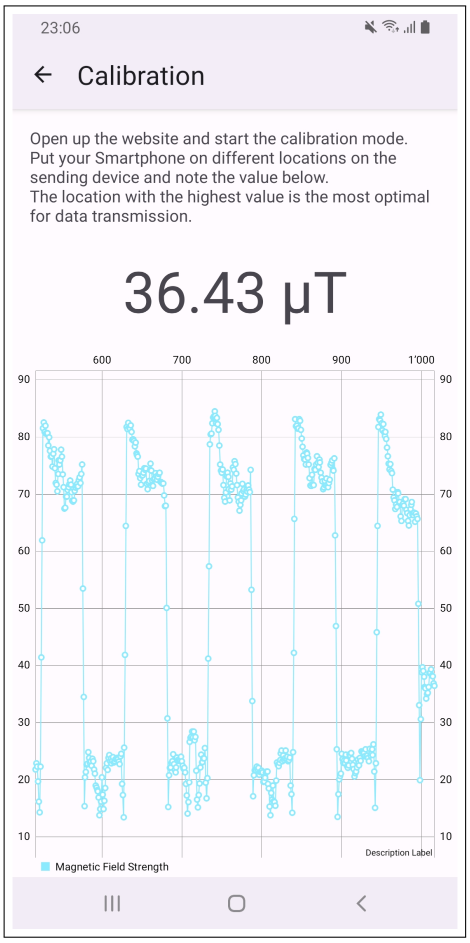


If you want to receive text that is transmitted, you can click the following button.

For finding the best spot to put your phone on, to get the best transmission, you can start the calibration process by clicking the following button.

Figure 5.8.: App - Home

The calibration mode is used to find the most optimal location where the user should put the smartphone on the sending device. The website will continously put a load on the CPU. The app will then show the current strength of the magnetic field. The location where the strength is the highest is the one where you will get the best transmission results.

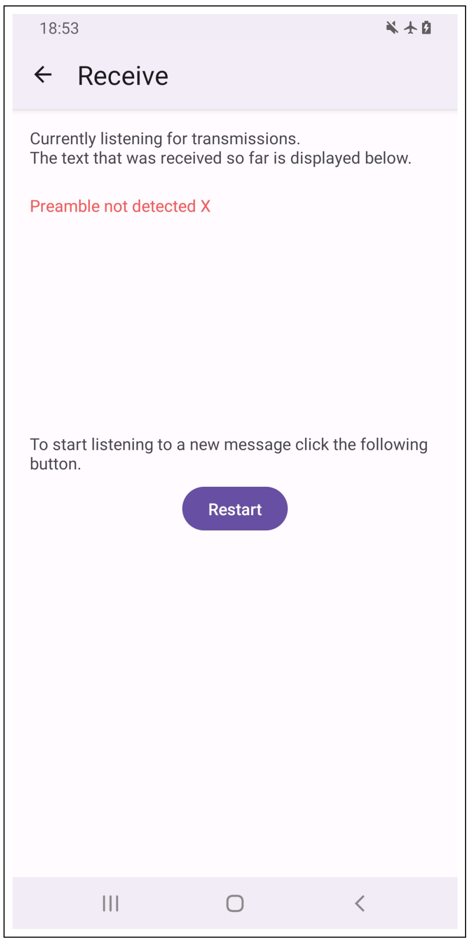


Open up the website and start the calibration mode. Put your Smartphone on different locations on the sending device and note the value below.
The location with the highest value is the most optimal for data transmission.

36.43 µT

Figure 5.9.: App - Calibration

During the reception process, the currently received text is displayed on the screen while a loading indicator indicates that the process is still ongoing. When the process is finished, the complete text is displayed. At any time, the user has the possibility to restart the reception process for a new message.

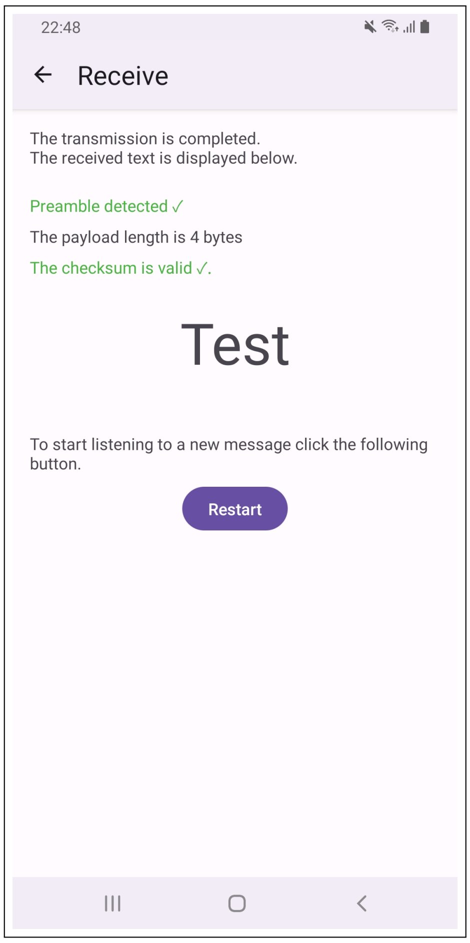


Currently listening for transmissions.
The text that was received so far is displayed below.

Preamble not detected X

To start listening to a new message click the following button.

Figure 5.10.: App - Receiving in Progress



The transmission is completed.
The received text is displayed below.

Preamble detected ✓

The payload length is 4 bytes

The checksum is valid ✓.

Test

To start listening to a new message click the following button.

Figure 5.11.: App - Receving Completed

6. Testing

6.1. Devices

The following devices were used for the development and testing of MagSend. While theoretically MagSend should work on other laptops and smartphones, I have only tested it on the specified devices. For each device, the most important specifications are listed.

6.1.1. Laptop

The laptop was used as the sending device for MagSend. The Website was run on the Firefox browser. The laptop was always plugged in to the power grid to ensure that the CPU throttles less often.

Device	Lenovo Thinkpad T470P
Operating System	Arch Linux
CPU	Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz
Core Count	4
Memory	16GB
Web-Browser	Firefox 108.0.1

Table 6.1.: Testing Device - Laptop[22]

6.1.2. Smartphone

The smartphone was used as the receiving device for MagSend. The app was run normally as a native Android application.

Device	Samsung Galaxy S9+
Operating System	Android 10 with One UI 2.5
SoC	Samsung Exynos 9810
Memory	6GB
Sensor Rate	Limited to 200 Hz by Android[23]

Table 6.2.: Testing Device - Smartphone[24]

6.2. Test Cases

The following test cases are all structured in the same way. The table lists the related requirements from chapter 4 and describes the test case briefly. Then the expected result row lists the steps and the result that should happen for the test case to be successful. The last row describes the actual result from testing.

ID	T01
Related Requirement	F01
Description	The user is provided with the option to start the calibration on the website.
Expected Result	1. The user opens the website in a web browser 2. The user is provided with a button to start the calibration
Actual Result	Upon opening the website, the user is greeted with an option to start the calibration (see Figure 6.1).

Table 6.3.: Test Case 01

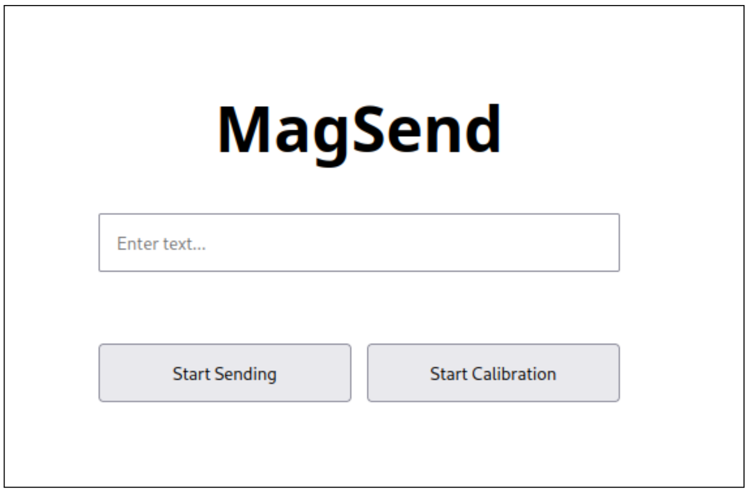
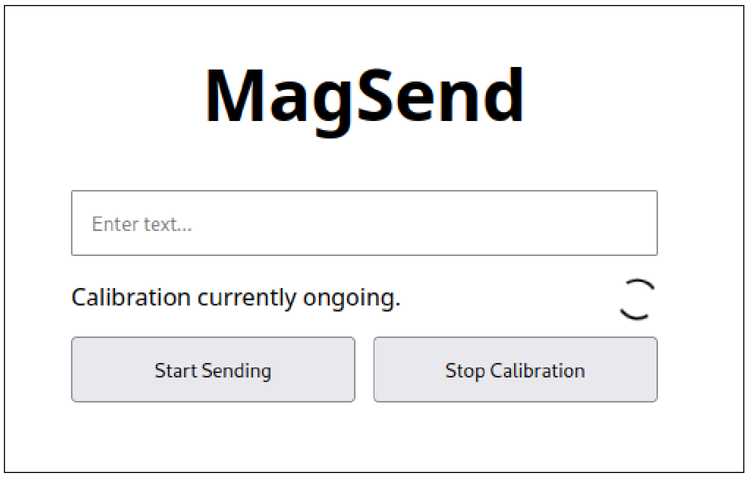


Figure 6.1.: Test Case 01 - Start Calibration Button on Website

ID	T02
Related Requirement	F01
Description	The user can start the calibration process on the website.
Expected Result	1. The user opens the website in a web browser 2. The user clicks on the ”Start Calibration” button 3. The calibration process is started, and the user is notified about it 4. During calibration, the CPU is periodically stressed
Actual Result	When the user starts the calibration process, the ”Start Calibration” button changes to ”Stop Calibration” and a message appears that indicates that calibration is in process (see Figure 6.2). While calibrating, the CPU is periodically stressed as seen in the CPU utilization graph (see Figure 6.3).

Table 6.4.: Test Case 02

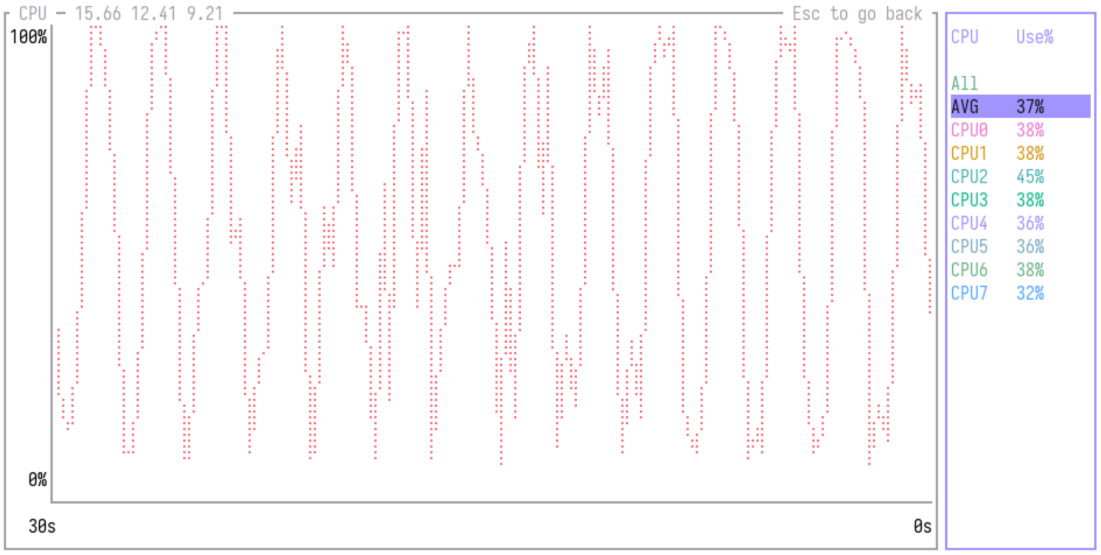


MagSend

Enter text...

Calibration currently ongoing.

Figure 6.2.: Test Case 02 - Calibration in Process on Website



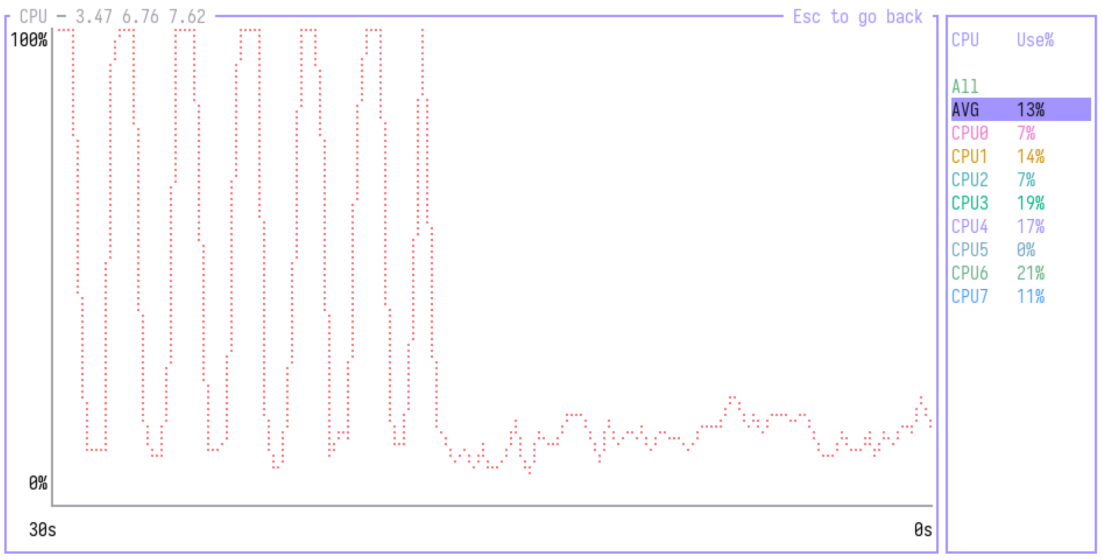
```
CPU     Use%

All
AVG     37%
CPU0    38%
CPU1    38%
CPU2    45%
CPU3    38%
CPU4    36%
CPU5    36%
CPU6    38%
CPU7    32%
```

Figure 6.3.: Test Case 02 - CPU Utilization

ID	T03
Related Requirement	F01
Description	The user can stop the calibration process on the website.
Expected Result	1. The user opens the website in a web browser 2. The user clicks on the ”Start Calibration” button 3. The user then clicks on the new ”Stop Calibration” button 4. The calibration is stopped, and the CPU should not be utilized anymore
Actual Result	When the user clicks on the ”Stop Calibration” button, the message about the calibration disappears and the button switches back to ”Start Calibration”. In the end, the website should look like if it has just opened. After stopping the calibration, the CPU utilization also goes back down to normal levels (see Figure 6.4).

Table 6.5.: Test Case 03



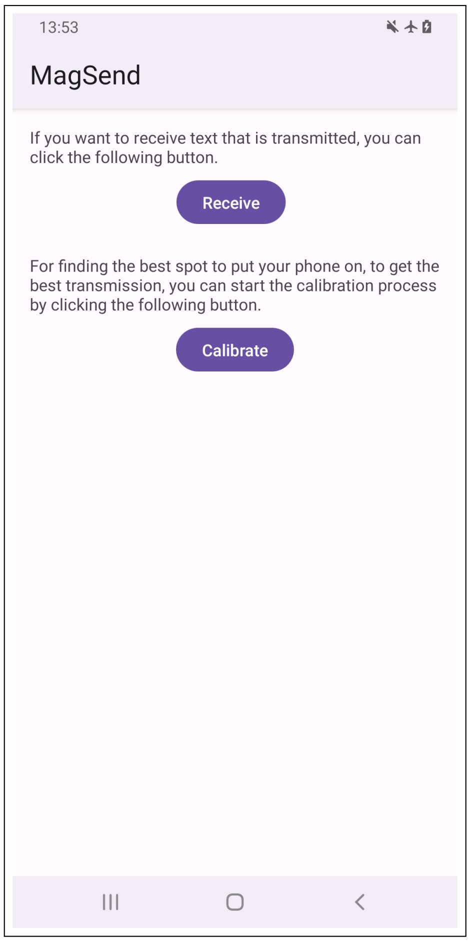
```
CPU     Use%

All
AVG     13%
CPU0    7%
CPU1    14%
CPU2    7%
CPU3    19%
CPU4    17%
CPU5    0%
CPU6    21%
CPU7    11%
```

Figure 6.4.: Test Case 03 - CPU Utilization after Stopping the Calibration

ID	T04
Related Requirement	F02
Description	The user is provided with the option to start calibration in the app.
Expected Result	1. The user opens the app 2. The user is provided with a button to start the calibration
Actual Result	Upon opening the app, the user is greeted with an option to start the calibration (see Figure 6.5).

Table 6.6.: Test Case 04



13:53

MagSend

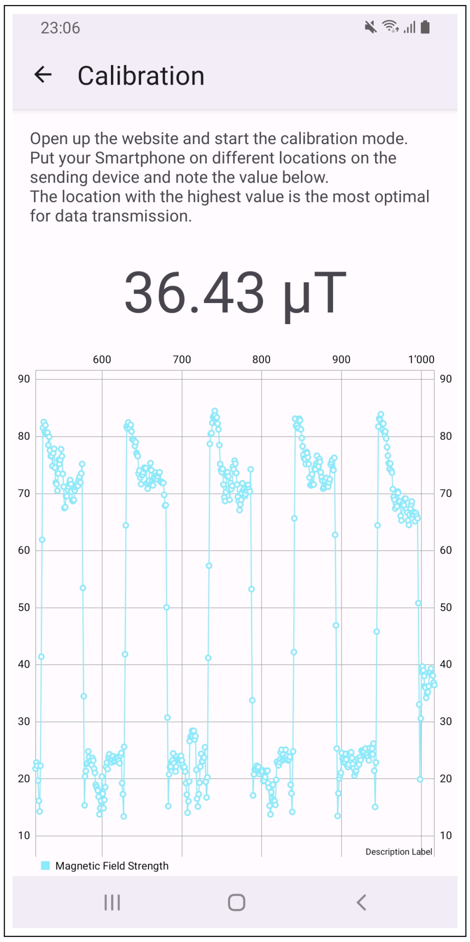
If you want to receive text that is transmitted, you can click the following button.

For finding the best spot to put your phone on, to get the best transmission, you can start the calibration process by clicking the following button.

Figure 6.5.: Test Case 04 - Home Screen in the App

ID	T05
Related Requirement	F02
Description	The user can start the calibration process in the app.
Expected Result	1. The user opens the app 2. The user clicks on the ”Calibrate” button 3. The user is shown the current measured magnetic flux density as well as a chart of the historical values
Actual Result	After starting the calibration in the app, the user can see the magnetic flux density and historical data in a chart. (see Figure 6.6).

Table 6.7.: Test Case 05



23:06

Calibration

Open up the website and start the calibration mode.
Put your Smartphone on different locations on the sending device and note the value below.
The location with the highest value is the most optimal for data transmission.

36.43 µT

Figure 6.6.: Test Case 05 - Calibrate Screen in the App

ID	T06
Related Requirement	F02
Description	The user can stop the calibration process in the app.
Expected Result	1. The user opens the app 2. The user clicks on the ”Calibrate” button 3. While on the calibration screen, the user clicks on the back arrow in the top left corner 4. The user should be back on the home screen of the app
Actual Result	After clicking the back arrow inside the calibration screen, the user arrives back on the home screen and can choose again between ”Receive” and ”Calibrate”.

Table 6.8.: Test Case 06

ID	T07
Related Requirement	F03
Description	The user can enter text on the website.
Expected Result	1. The user opens the website in a web browser 2. The user clicks into the text input 3. The user can enter any text between 0 and 16 characters in length
Actual Result	The user can enter any character in the input they want, but they are unable to enter more than 16 characters (see Figure 6.7).

Table 6.9.: Test Case 07

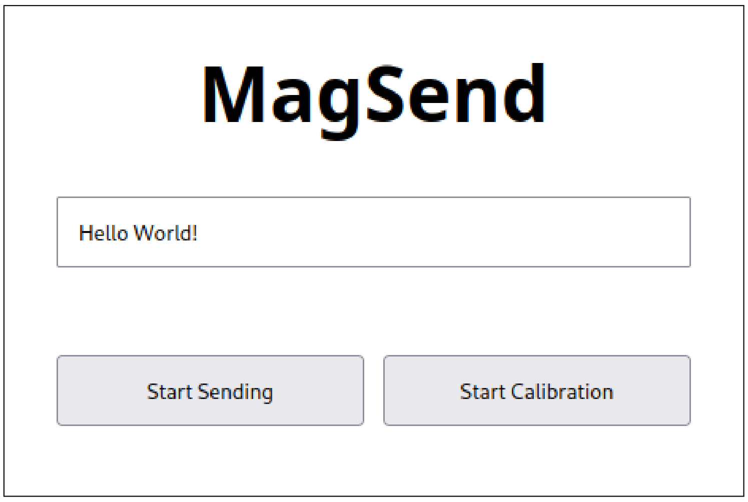
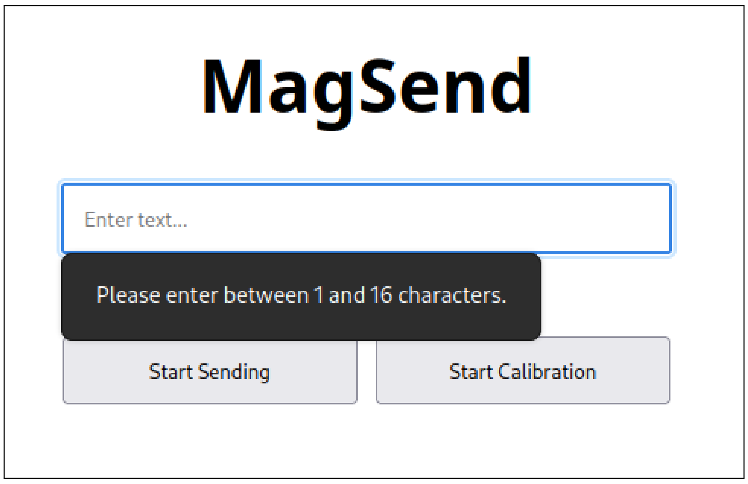


Figure 6.7.: Test Case 07 - Entering Text on the Website

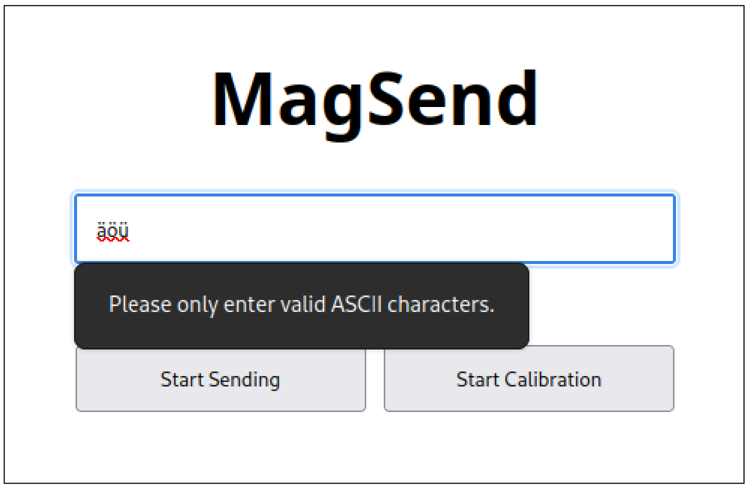
ID	T08
Related Requirement	F04
Description	The entered text is validated on the website when sending.
Expected Result	1. The user opens the website in a web browser 2. The user clicks into the text input 3. The user enters any text between 0 and 16 characters 4. The user clicks on ”Start Sending” 5. When the user hasn't entered any text or the text contains non-ASCII characters, a fitting message is displayed, and the sending process is not started.
Actual Result	When the user attempts to start sending when they haven't entered any text, a message is displayed (see Figure 6.8). A message also appears, when the user attempts to start sending when they have entered any non-ASCII characters (see Figure 6.9). In both cases, the sending process is not started.

Table 6.10.: Test Case 08



Please enter between 1 and 16 characters.

Figure 6.8.: Test Case 08 - Validation of Empty Input

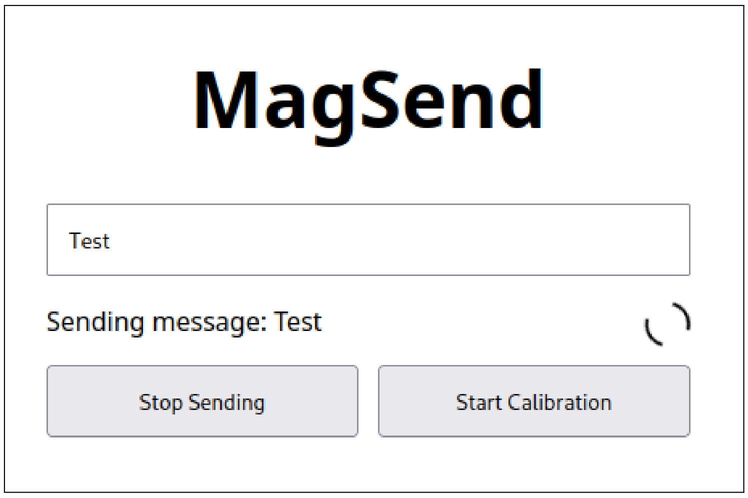


Please only enter valid ASCII characters.

Figure 6.9.: Test Case 08 - Validation of Invalid Characters

ID	T09
Related Requirement	F04
Description	The user can send text using the website.
Expected Result	1. The user opens the website in a web browser 2. The user clicks into the text input 3. The user enters any text between 0 and 16 characters 4. The user clicks on ”Start Sending” 5. The user is notified that the sending has started and the text that was entered is sent using the MagSend protocol by stressing the CPU according to the payload
Actual Result	When the user starts the sending process, the ”Start Sending” button changes to ”Stop Sencing” and a message appears that indicates that the sending is in process (see Figure 6.10). In the CPU utilization during the sending, the structure of the preamble and the packet itself can be seen (see Figure 6.11).

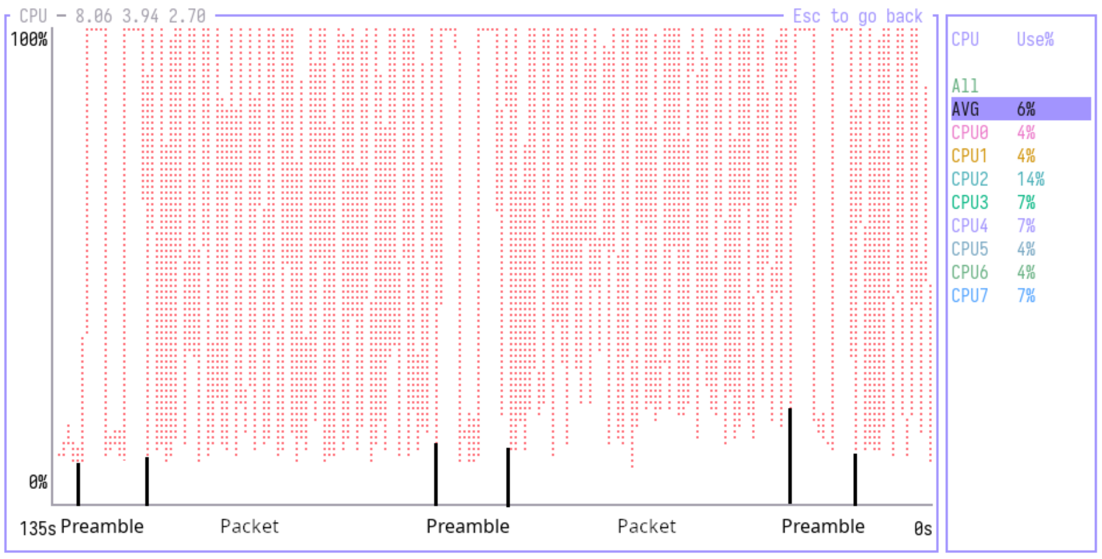
Table 6.11.: Test Case 09


MagSend

Test

Sending message: Test

Figure 6.10.: Test Case 09 - Website when Sending a Message

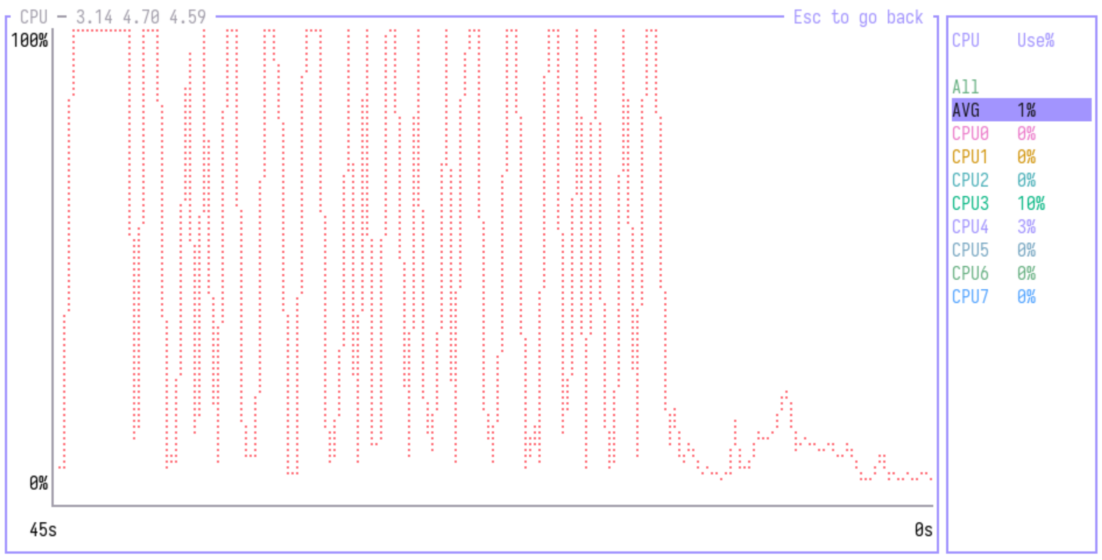


CPU	Use%
All	
AVG	6%
CPU0	4%
CPU1	4%
CPU2	14%
CPU3	7%
CPU4	7%
CPU5	4%
CPU6	4%
CPU7	7%

Figure 6.11.: Test Case 09 - CPU Utilization when Sending a Message

ID	T10
Related Requirement	F04
Description	The user can stop the sending process.
Expected Result	1. The user opens the website in a web browser 2. The user clicks on the ”Start Sending” button 3. The user then clicks on the new ”Stop Sencding” button 4. The sending is stopped, and the CPU should not be utilized anymore
Actual Result	When the user clicks on the ”Stop Sending” button, the message about the sending disappears and the button switches back to ”Start Sending”. In the end, the website should look like if it has just opened. After stopping the sending, the CPU utilization also goes back down to normal levels (see Figure 6.12).

Table 6.12.: Test Case 10



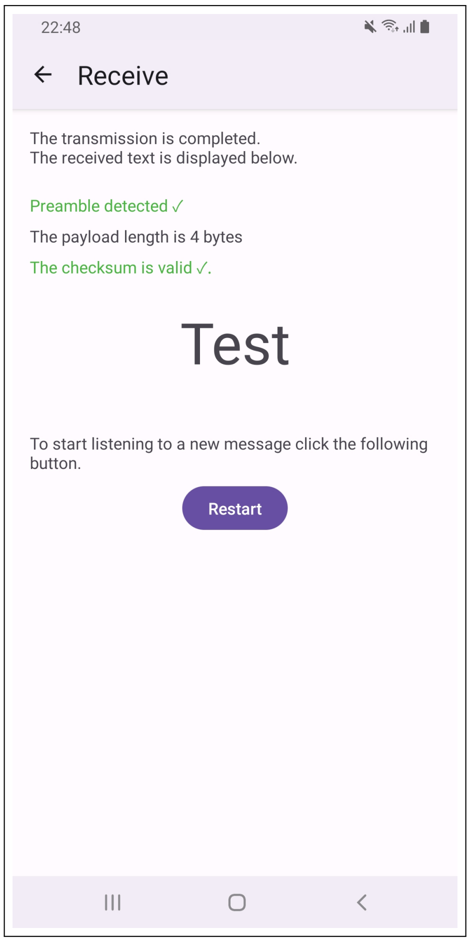
```
CPU     Use%

All
AVG     1%
CPU0    0%
CPU1    0%
CPU2    0%
CPU3    10%
CPU4    3%
CPU5    0%
CPU6    0%
CPU7    0%
```

Figure 6.12.: Test Case 10 - CPU Utilization after Stopping the Sending

ID	T11
Related Requirement	F05
Description	The user can receive data using the app.
Expected Result	1. The user opens the app 2. The user puts the smartphone on the laptop 3. The user clicks on the "Receive" button 4. The user starts the sending process on the website 5. The app will start the reception process and will indicate if the preamble was detected, the length of the received payload, and if the checksum is valid. The received text is also displayed.
Actual Result	When the users starts the reception process they are informed about the current process of the receive status. They are informed whether the preamble was detected, about the length of the payload, and if the checksum is valid. They can also see the currently received text at all times (see Figure 6.13). When the checksum is invalid the user is informed about it (see Figure 6.14).

Table 6.13.: Test Case 11



22:48

Receive

The transmission is completed.
The received text is displayed below.

Preamble detected ✓

The payload length is 4 bytes

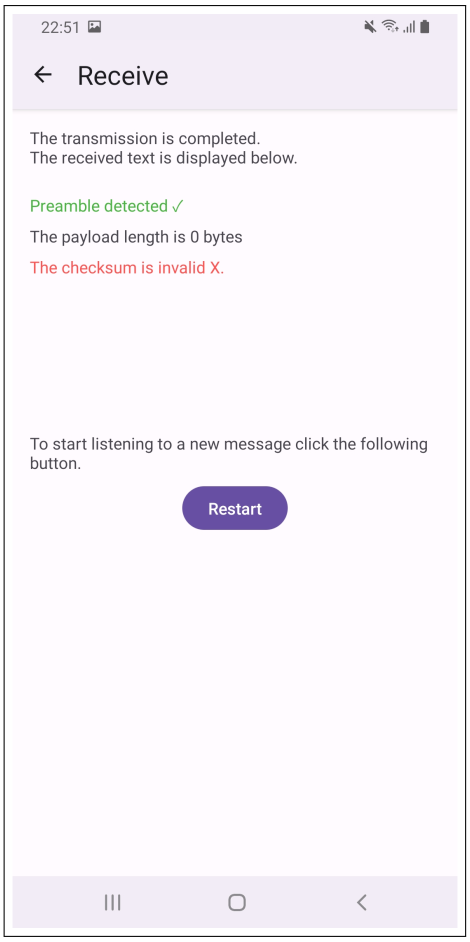
The checksum is valid ✓.

Test

To start listening to a new message click the following button.

Restart

Figure 6.13.: Test Case 11 - Reception Process without any Errors



22:51

Receive

The transmission is completed.
The received text is displayed below.

Preamble detected ✓

The payload length is 0 bytes

The checksum is invalid X.

To start listening to a new message click the following button.

Restart

Figure 6.14.: Test Case 11 - Reception Process with Invalid Checksum

ID	T12
Related Requirement	F05
Description	The user can restart the reception process.
Expected Result	1. The user opens the app 2. The user puts the smartphone on the laptop 3. The user clicks on the "Receive" button in the app 4. The user starts the sending process on the website 5. The user clicks on the "Restart" button in the app 6. The app should reset the reception progress to a clean state
Actual Result	When the user clicks on the "Restart" button, the received text gets removed and all status information is reset to their default state.

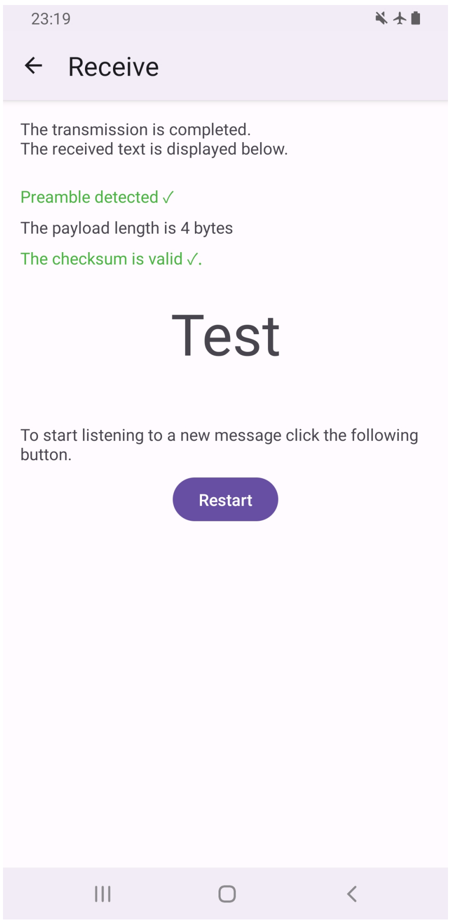
Table 6.14.: Test Case 12

ID	T13
Related Requirement	F05
Description	The user can stop the reception process.
Expected Result	1. The user opens the app 2. The user puts the smartphone on the laptop 3. The user clicks on the "Receive" button in the app 4. While on the reception screen, the user clicks on the back arrow in the top left corner 5. The user should be back on the home screen of the app
Actual Result	After clicking the back arrow inside the reception screen, the user arrives back on the home screen and can choose again between "Receive" and "Calibrate".

Table 6.15.: Test Case 13

ID	T14
Related Requirement	N01
Description	The app can receive data in airplane mode.
Expected Result	1. The user enables airplane mode on his smartphone 2. The user opens the app 3. The user puts the smartphone on the laptop 4. The user clicks on the "Receive" button in the app 5. The app should still be able to receive messages over MagSend
Actual Result	Even in airplane mode, the app can still receive data over MagSend as this process doesn't use any connectivity but only the inbuilt sensors of the smartphone (see Figure 6.15).

Table 6.16.: Test Case 14


23:19

Receive

The transmission is completed.
The received text is displayed below.

Preamble detected ✓

The payload length is 4 bytes

The checksum is valid ✓.

Test

To start listening to a new message click the following button.

Figure 6.15.: Test Case 14 - Receive Process in Airplane Mode

ID	T15
Related Requirement	N02
Description	The transfer speed between laptop and smartphone is 1 bit/s.
Expected Result	1. The user opens the app 2. The user puts the smartphone on the laptop 3. The user clicks on the ”Receive” button 4. The user starts the sending process on the website 5. The transfer speed between laptop and smartphone should at least be 1 bit/s
Actual Result	The clock speed of the transmitted signal is 500ms. The MagSend payload is encoded using a manchester code, which doubles the data size. This means every second 1 bit gets transmitted. For more information about data rate benchmarks, please refer to chapter 7.

Table 6.17.: Test Case 15

7. Benchmarking

This chapter will research the real world capabilities of MagSend. I attempt to transmit a message using different speeds of MagSend under different system CPU loads to see how it performs and which transfer speed is the most reliable to use.

The following benchmark tests are all structured in the same way. I run each test 5 times to get a more accurate view of the test. For each test, I wrote down if MagSend can detect the preamble, the received payload length, the received message, and whether the CRC is valid.

The message used in the test transmissions is ”**Test**”.

7.1. Idle CPU Load

The following benchmarks were run during idle CPU Load, which was about 1-5% utilization.

7.1.1. 0.5 bit/s

Run	Preamble Detected	Payload Length	Is CRC Valid	Received Message
1	Yes	4 B	No	Vêÿ
2	Yes	4 B	Yes	Test
3	Yes	4 B	Yes	Test
4	Yes	4 B	Yes	Test
5	Yes	4 B	No	_rÛ

Table 7.1.: Benchmark - Idle CPU Load - 0.5 bit/s

7.1.2. 1 bit/s

Run	Preamble Detected	Payload Length	Is CRC Valid	Received Message
1	Yes	4 B	Yes	Test
2	Yes	4 B	No	Test
3	Yes	4 B	Yes	Test
4	Yes	4 B	Yes	Test
5	Yes	4 B	No	TÊæé

Table 7.2.: Benchmark - Idle CPU Load - 1 bit/s

7.2. 25% CPU Load

The following benchmarks were run with 25% CPU utilization. The tool `stress` was used to put a load on the CPU [25].

7.2.1. 0.5 bit/s

Run	Preamble Detected	Payload Length	Is CRC Valid	Received Message
1	Yes	N/A	No	N/A
2	Yes	4 B	No	N/A
3	No	N/A	No	N/A
4	No	N/A	No	N/A
5	No	N/A	No	N/A

Table 7.3.: Benchmark - 25% CPU Load - 0.5 bit/s

7.2.2. 1 bit/s

Run	Preamble Detected	Payload Length	Is CRC Valid	Received Message
1	No	N/A	No	N/A
2	Yes	N/A	No	N/A
3	No	N/A	No	N/A
4	No	N/A	No	N/A
5	No	N/A	No	N/A

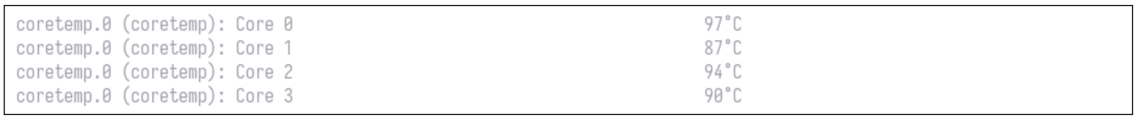
Table 7.4.: Benchmark - 25% CPU Load - 1 bit/s

7.2.3. Results

As one can see in the benchmarks, the most successful configuration was running MagSend with 1 bit/s at with an idle CPU followed by 0.5 bit/s at an idle CPU.

The benchmarks with 25% CPU load were mostly failing, with a few being able to detect the preamble, and one even being able to receive the payload length from the message. There might be a few reasons why the results are like this. The first one is thermal throttling (see section 2.3).

The laptop that was used for running the benchmarks already reaches CPU temperatures of over 90 °C when only running at around 25% CPU load (see Figure 7.1).



```
coretemp.0 (coretemp): Core 0                                  97°C
coretemp.0 (coretemp): Core 1                                  87°C
coretemp.0 (coretemp): Core 2                                  94°C
coretemp.0 (coretemp): Core 3                                  90°C
```

Figure 7.1.: CPU temperatures under 25% CPU load

The other problem with the failing benchmarks is on the receiving side. The interval at which the sensor provides data to the application is not guaranteed and thus introduces some jitter [26], which is not yet handled perfectly in the app.

8. Materials & Methods

To develop, test and benchmark MagSend the following devices and software are needed:

A laptop, which acts as the sender. This laptop needs to be able to run a modern web browser which supports the Web Workers API (see section 2.6). The operating system is irrelevant as MagSend is web-based.

An Android smartphone, which acts as the receiver. This phone needs to run at least Android version 10 to be able to run the MagSend client application. Furthermore, it is required that the phone is equipped with a magnetometer. Most modern smartphones should contain a hall effect sensor for that purpose (see section 2.2).

For developing, the official IDE for Android development, Android Studio [27] was used. While this is not strictly required to run MagSend, it is the most convenient way to do so.

9. Future Work

The goal of this thesis was to research and develop a communication method which uses magnetic induction, while using standard components and devices.

The result was a working prototype that allows non-technical users to transmit a message from their laptop to their smartphone wirelessly using magnetic induction, called MagSend.

For future improvements surrounding MagSend, the most important one is to enhance the robustness of the system. While the system generally works, there are any many cases where the message doesn't get received, the preamble doesn't get detected, or the received message is incorrect. Furthermore, when the system gets more robust, it might also be possible to increase the transfer speed of MagSend. In the end, the user experience and user interface applications can also be improved by giving more feedback and guidance to the user on how to use MagSend.

10. Project Management

10.1. Project Plan

At the beginning of the project, a project plan was created to plan all of the work. That original plan can be found in subsection 10.1.1. During the lifetime of the project, the plan was updated to reflect the actual state of the project. That adjusted plan can be found in subsection 10.1.2.

For the first few weeks, I managed to match my initial plan pretty well. I spent a bit more time than I'd like on the concrete selection of my subject. First I planned to use a speaker as the sending device in MagSend, as a magnetic field is generated, when a sound is played through the speaker. After multiple tests and experiments I wasn't able to get useful results from the speakers and settled on using a CPU after some quick prototypes.

In October, after defining the requirements for the project, my plan fell apart as I fell ill with COVID-19. I was sick for about 3 weeks and in that time I wasn't able to work on my thesis. This whole event caused my plan to shift behind a few weeks. I was generous in my initial planning, which helped alleviate this issue.

In November, I picked up the pace again and started with the planning and documentation of the solution, and then started to work on the prototypes for the sender and receiver. I spent a bit longer on the prototype for the receiver as in early December I was busy with some other projects and exams.

During the winter break and in early January, I in the process of finishing up the thesis report and the additional deliverables like the poster and movie. In the end it was a bit of a crunch to finish everything but thanks to some generous planning in the beginning it all worked out.

10.1.1. Initial Plan

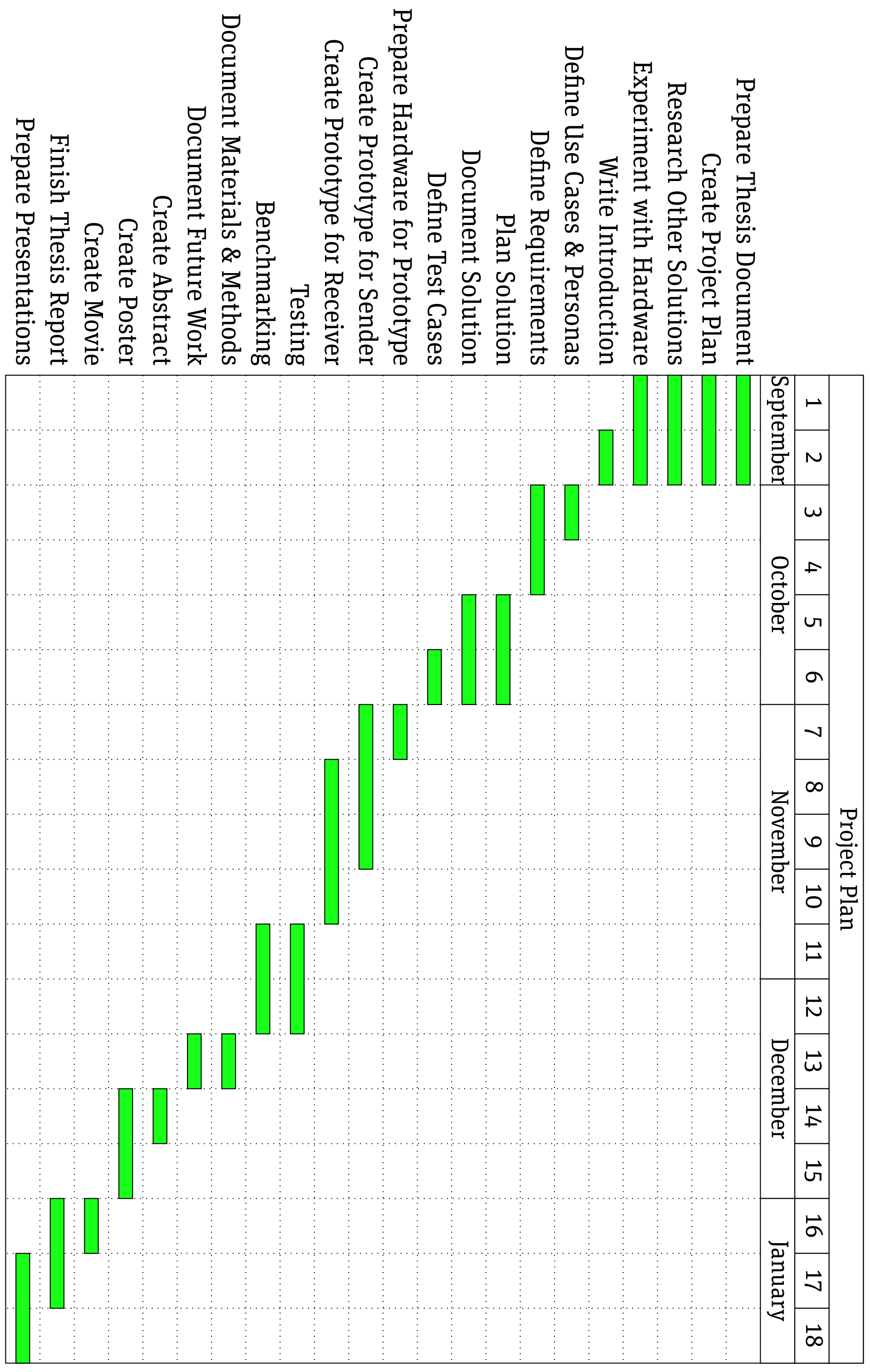


Figure 10.1.: Project Plan

10.1.2. Actual Plan

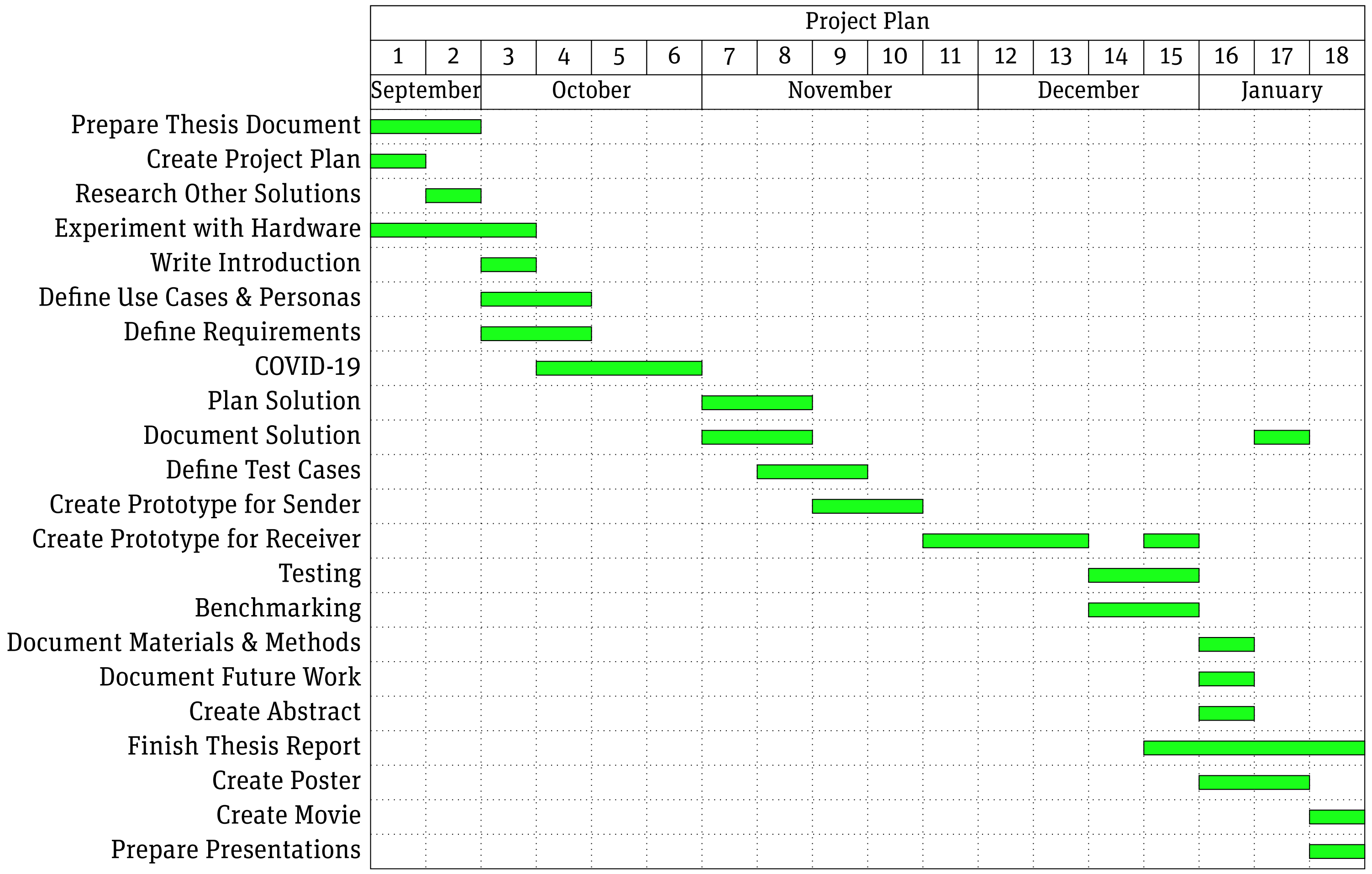


Figure 10.2.: Project Plan

10.2. Organization

All data for the project, including documentation, images, data, and source code was kept in a single Git repository [28]. The documentation and presentations were written using LaTeX with the official BFH templates.

10.2.1. Meetings

In the first week I agreed on weekly meetings with my advisor, unless it wasn't needed. For the big meetings I created a meeting notes document in advance, where I wrote down my progress since the last week and questions that I might have for my advisor. In hindsight, weekly meeting were a bit too frequent and bi-weekly meeting would have sufficed, but it was useful to have regular feedback from my advisor.

10.2.2. Mockups

During the planning of the solution, I created multiple mockups of the planned website and app screens. To create the mockups, I used the open-source design tool Penpot [29].

10.3. Conclusion

The project, albeit with some issues, was finished in time. The initial plan was only met for the beginning as the sickness delayed the rest of the tasks. This delay caused the last few of weeks of the project to be more stressful as I'd have liked, but thanks to generous planning it worked out.

One of the biggest things in terms of project management I can improve for the next project is to plan everything a bit more. While I had some rough planning with my project plan, the individual points, especially the development of the prototypes were not fully planned. By splitting those tasks up in smaller ones, I could have planned around them better and might have found some flaws in the plan.

The other problem was with the debugging and developing of the Android application. Many times I was testing the receiving of data over MagSend, when it didn't work, while it worked just in the try before. After much trial and error I found out that one of the main issues was that the logging capabilities of Android were quite slow and sometimes delayed the values from the magnetic sensor and caused wrong results, that didn't appear in normal utilization of the app. My takeaway from this is, to invest early in good logging and debugging utilities, especially when dealing with "invisible" data, such as magnetic fields and to use separate devices for running the applications as those used for developing them.

In the end, I'm happy with the project and my results. I managed to develop a custom communication protocol and finish a working prototype in the allotted time.

Declaration of Authorship

I hereby declare that I have written this thesis independently and have not used any sources or aids other than those acknowledged.

All statements taken from other writings, either literally or in essence, have been marked as such.

I hereby agree that the present work may be reviewed in electronic form using appropriate software.

S. Kaderli

January 17, 2023

Severin Kaderli

Bibliography

[1] Encyclopædia Britannica. (2022) Radio waves. [Online]. Available: https://www.britannica.com/science/electromagnetic-radiation/Radio-waves

[2] New Zealand Government. Radio interference. [Online]. Available: https://www.rsm.govt.nz/business-individuals/interference/radio-interference/

[3] everything RF. (2022) What is RF Jamming? [Online]. Available: https://www.everythingrf.com/community/what-is-rf-jamming

[4] P. Hao, C. Yi-Chao, X. Guangtao, and J. Xiaoyu, "MagneComm: Magnetometer-based Near-Field Communication," 2017.

[5] N. Thilak and R. Braun, "Near field magnetic induction Communication in Body Area Network," 2012.

[6] S. Zurek. (2021) Magnetic flux density? [Online]. Available: https://e-magnetica.pl/magnetic_flux_density

[7] GSMScore. Mobiles with Hall Effect Sensors available in Market. [Online]. Available: https://www.gsmscore.com/model-finder/sensors/hall-effect/

[8] D. Nield. (2020) All the Sensors in Your Smartphone, and How They Work. [Online]. Available: https://gizmodo.com/all-the-sensors-in-your-smartphone-and-how-they-work-1797121002

[9] R. Keim. (2015) Understanding and Applying the Hall Effect. [Online]. Available: https://www.allaboutcircuits.com/technical-articles/understanding-and-applying-the-hall-effect

[10] C. R. Nave. (1998) Hall Effect. [Online]. Available: http://hyperphysics.phy-astr.gsu.edu/hbase/magnetic/Hall.html

[11] T. Brookes. (2021) What Is Thermal Throttling? [Online]. Available: https://www.howtogeek.com/739732/what-is-thermal-throttling/

[12] R. N. Williams. (1993) A Painless Guide to CRC Error Detection Algorithms. [Online]. Available: https://zlib.net/crc_v3.txt

[13] R. Tervo. (2016) Error Detection with the CRC. [Online]. Available: https://www.ece.unb.ca/tervo/ece4253/crc.shtml

[14] G. Cook. (2022) Catalogue of parametrised CRC algorithms. [Online]. Available: https://reveng.sourceforge.io/crc-catalogue/all.htm

[15] *Specification of CRC Routines*, AUTOSAR, 2021.

[16] P. Koopman. (2018) CRC Polynomial Zoo. [Online]. Available: https://users.ece.cmu.edu/~koopman/crc/crc8.html

[17] Techopedia. (2020) Cyclic Redundancy Check (CRC). [Online]. Available: https://www.techopedia.com/definition/1793/cyclic-redundancy-check-crc

[18] S. Schmidt, "Manchester Code," 2006, License: Public Domain. [Online]. Available: https://commons.wikimedia.org/wiki/File: Manchester_encoding_both_conventions.svg

[19] R. Oed. (2022) Old, but Still Useful: The Manchester Code. [Online]. Available: https://www.digikey.ch/en/blog/old-but-still-useful-the-manchester-code

[20] MDN contributors. (2022) Web Workers API. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/Web_Workers_API

[21] R. Lie. (2018) LoRa Packet Format, Time on Air and Adaptive Data Rate. [Online]. Available: https://www.mobilefish.com/download/lora/lora_part17.pdf

[22] Lenovo. (2018) ThinkPad T470p Platform Specifications. [Online]. Available: https://psref.lenovo.com/syspool/Sys/PDF/ThinkPad/ThinkPad_T470p/ThinkPad_T470p_Spec.PDF

[23] Android Open Source Project. (2022) Sensor Rate-Limiting. [Online]. Available: https://developer.android.com/guide/topics/sensors/sensors_overview# sensors-rate-limiting

[24] GSMArena. (2018) Samsung Galaxy S9+. [Online]. Available: https://www.gsmarena.com/samsung_galaxy_s9+-8967.php

[25] A. Waterland. (2021) stress(1). [Online]. Available: https://man.archlinux.org/man/community/stress/stress.1.en

[26] Android Open Source Project. (2022) Monitoring Sensor Events. [Online]. Available: https://developer.android.com/guide/topics/sensors/sensors_overview.html# sensors-monitor

[27] Android Studio. [Online]. Available: https://developer.android.com/studio

[28] S. Kaderli. (2022) MagSend. [Online]. Available: https://git.kaderli.dev/severinkaderli/bachelor-thesis

[29] (2022) Penpot - Design Freedom for Teams. [Online]. Available: https://penpot.app/

Glossary

2FA	two-factor authentication.
CRC	Cyclic Redundancy Code.
DOM	Document Object Model.
MI	Magnetic Induction.
NFC	Near-field communication.
QR	Quick Response.
RFID	Radio-frequency identification.

List of Figures

List of Tables

Listings

A. Task Description

Berner Fachhochschule
Haute école spécialisée bernoise
Bern University of Applied Sciences

Bachelor Thesis

for	Severin Kaderli
Subject Area	Computer Science
Supervisor	Reto Koenig

Inductive Short-Range Communication Channel

Nowadays, communication channels between electronic devices are based on the properties of electromagnetism. But what if the surrounding medium blocks electromagnetic waves?
In this bachelor thesis, a communication channel will be created that uses the properties of magnetic induction. The standard components of today's smartphones shall be used to turn the smartphone into a receiver.
The aim of the bachelor thesis is to create a proof of concept so that a chat application can be developed on it in which at least one path of communication is inductive. The generation or distortion of the magnetic near field for communication should take place with minimal technical effort and be based exclusively on standard components.

Advisor:	Head of Department:
R.E. Koenig	[illegible]