Development of a Smart Reflective Surface-enabled System for Wireless Networks

Yernur Kalikhan, B. Eng. in Electrical and Computer Engineering

Submitted in fulfilment of the requirements

for the degree of Master of Science

in Electrical and Computer Engineering

School of Engineering and Digital Sciences

Department of Electrical and Computer Engineering

Nazarbayev University

53 Kabanbay Batyr Avenue,

Astana, Kazakhstan, 010000

Supervisors: Dr. Annie Ng, Dr. Galymzhan Nauryzbayev

April 2024

Declaration

I hereby, declare that this manuscript, entitled “Development of a Smart Reflective Surface-enabled System for Wireless Networks”, is the result of my own work except for quotations and citations which have been duly acknowledged.

I also declare that, to the best of my knowledge and belief, it has not been previously or concurrently submitted, in whole or in part, for any other degree or diploma at Nazarbayev University or any other national or international institution.

Name: Yernur Kalikhan

Date: 02.05.2024

Abstract

In the modern world of rapidly evolving technologies, wireless communications development is focused on facilitating higher data rates and better quality of service. Reconfigurable Intelligent Surfaces (RISs) allow control beyond the physical layer, the wireless medium, which is a big leap in developing sub-5GHz technology. An extensive review of RIS aspects, its working principles, and applications in wireless networks is provided in this thesis. A literature survey was done to identify the RIS prototyping challenges and controller selection factors. The study describes the prototype of the RIS system integrated with the controller, highlighting the hardware limitations and measurement setup requirements. Although the accurate measurement of the RIS was not done promptly in this study, the initial testing and experimental design give valuable insight into future studies, which may include scaling the RIS system, improving the indicating board, and optimizing the algorithm.

Acknowledgments

I would like to express my gratitude to my supervisors Dr. Annie Ng and Dr. Galymzhan Nauryzbayev for their guidance and support during this research. I am thankful to professors Dr. Akhan Almagambetov, and Dr. Nursultan Kabylkas for their assistance. I am also thankful to my colleagues Tolegen Oraz, and Adilkhan Abdikarim for their collaboration.

Finally, I would like to thank my family for their love and support.

Table of Contents

List of Abbreviations

RIS	Reconfigurable Intelligent Surface
FD	Full-Duplex
HD	Half-Duplex
AF	Amplify-and-Forward
DF	Decode-and-Forward
VLC	Visible Light Communication
FSO	Free-Space Optics
RFID	Radio Frequency Identification
SWIPT	Simultaneous Wireless Information and Power Transfer
MEC	Mobile-Edge Computing
NOMA	Non-Orthogonal Multiple Access
PLS	Physical Layer Security
PIN	Positive-Intrinsic Negative
GPIO	General Purpose Input/Output
FPGA	Field Programmable Gate Array
SPI	Serial Peripheral Interface
RCK	Register Clock
SRCK	Shift Register Clock
Tx	Transmitter
Rx	Receiver
EMI	Electromagnetic Induction

PCB	Printed Circuit Board
IDE	Integrated Development Environment
HPS	Hard Processor System
UART	Universal Asynchronous Receiver/Transmitter
SSH	Secure Shell
API	Application Programming Interface
VNA	Vector Network Analyzer
NLOS	Non-Line-Of-Sight
PL	Path Loss
CIR	Channel Impulse Response
PDP	Power Delay Profile

List of Tables

List of Figures

Chapter 1 – Introduction

1.1 Background

In the modern world of wireless communications with rapidly developing technology, it became clear that further development in networking requires a higher quality of service and extremely high data rates. Current technologies of sub-5GHz frequency bands are facing challenges due to signal interference, blockage problems, and fluctuations in transmission conditions [1]. The mentioned propagation limitations are commonly mitigated using relays to fill coverage holes between the base station and the receiver [2].

Reconfigurable Intelligent Surfaces (RISs) are a new technology in wireless networks, specially designed surfaces that are made of metamaterial to reflect radio signals. RISs enable a new approach in wireless networks where the environment is also part of the signal optimization process due to their tunable characteristics. It can be adjusted using electronic signals to manipulate electromagnetic waves, i.e. it can imitate various physical shapes. For example, the surface can simulate a metal plate with such an angle to focus waves on the desired area [3]. RIS technology is known by various names, including software-controlled metasurfaces [3-4], intelligent reflecting surfaces, reconfigurable reflectarrays, etc. [5]. The main difference of RIS technology is that the surface is positioned between the transmitter and the receiver of the wireless signals rather than in proximity. This creates new opportunities for use cases and presents new signal-processing difficulties related to the efficient use of the partially controlled channel [3].

RIS is built of metamaterials that are comprised of precisely engineered subwavelength-sized structural elements arranged in periodic patterns. These metamaterials have special electromagnetic properties, like perfect absorption, anomalous reflection/scattering, and negative

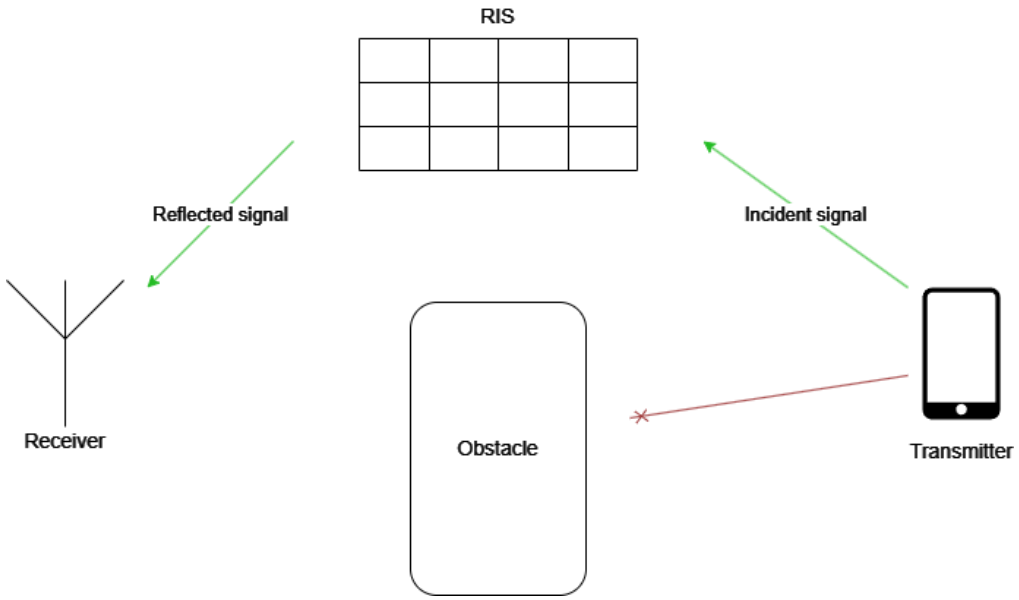

Figure 1.1: RIS-aided communication model for cellular network

refraction, that are not found in nature [2]. To achieve considerable gains, the RIS needs to have many elements with controllable properties. The capacity to dynamically manipulate how electromagnetic waves behave at the surface level allows to control signal propagation at much higher levels than traditional technologies. The example implementation is illustrated in Figure 1.1.

In [3], the authors state that unlike the conventional wireless network design, where the technology focuses on the physical layer transmission, the RIS allows control beyond the physical layer, the wireless medium. This presents an unprecedented leap forward in the development of wireless systems beyond the 5GHz frequency. The passive nature of RIS significantly lowers production costs and power consumption [1]. It is lightweight and thus can be installed virtually everywhere. Its natural full-duplex (FD) operating mode mitigates the risk of thermal noise and self-interference. By tuning the RIS cells' phase shifts, the reflected signals can enhance the signal power or reduce the multiuser interference by superimposing with the direct signals. The RIS applications include, but are not limited to:

- RIS-aided mmWave systems

- RIS-aided visible light communication (VLC)
- RIS-aided free-space optics (FSO)
- RIS-aided Radio Frequency Identification (RFID)
- RIS-aided Multicell networks
- RIS-aided simultaneous wireless information and power transfer (SWIPT) systems
- RIS-aided mobile edge computing (MEC) networks
- RIS-aided non-orthogonal multiple access (NOMA) systems
- RIS-aided Multicast systems
- RIS-aided physical layer security (PLS) systems

1.2 Literature Review

RIS is conceptually different from typical beamformers in various aspects. The most notable difference is the absence of an RF chain, as seen in phased arrays [6]. As a result, the RIS directs the reflected wave vectors completely, which is classified as an RF-chain-free wireless system. In electromagnetic theory, if a single EM-aperture has a periodic physical profile, it can do beam-forming without the assistance of many RF chains. The RIS unit cells, the smallest repeating geometry, change the structure's periodicity [7].

The RIS elements comprise unit cells of conductive printed patches that consist of the number of positive-intrinsic negative (PIN) diodes that surround metasurface cells made of metamaterial. Each patch can be tweaked so that the surface can restore the desired electromagnetic waves [7]. Here, the PIN diodes are equipped to tweak patches. By controlling the diodes, each unit cell can change its properties. The performance of RIS depends on the surface configuration that is made of individual panels, e.g. angle of reflection, the amplitude, and other characteristics of the signal can be different for various combinations of panel states

[1]. This feature allows to achieve great results in beamforming, reduced interference, and improved coverage. The permittivity of the substrate, its thickness, and the printed patch's geometry are the three variables that define the unit cell performance. The first two criteria for an RIS have some restrictions, therefore getting the right reaction from the unit cell depends in large part on the shape of the printed patch [7].

So, if RIS has *N* number of diodes, there are 2^*N* possible states [8]. In addition, the unit cell's reflection coefficient changes between 0° and 360°. In the most recent implementation, the 2-diode model had a 90° phase increment per the configuration of diodes [8]. However, the reflection angle of RIS is a complex combination of patterns of each cell pattern. When the cell pattern is selected using diodes, the reflection angle can be changed to the desired direction [9].

In RIS-enabled systems, the RIS components are usually modeled to function independently of one another, that is, without any mutual coupling between them. Furthermore, the reflectors are frequently represented as linear arrays in modeling. The array geometry barely affects the RIS reflection behavior in the performance studies. Nevertheless, to apply the techniques for two-dimensional arrays, the array geometry must be considered [9].

The controller is used to switch diodes "ON" and "OFF", thus controlling the RIS states. As the controller will be connected to RIS using its general-purpose input/output (GPIO) pins, we need to determine the number of pins required for the output. RIS may consist of many cells e.g. 12x12 in [8], or 16x16 in [10], meaning the controller should have a minimum of 144 or 256 output pins, respectively. When the switch toggles, the controller will generate the corresponding sequence to output [11].

Several solutions were proposed to increase the number of effective output pins. In [12], the authors created a high-speed control board consisting of two cascaded field programmable

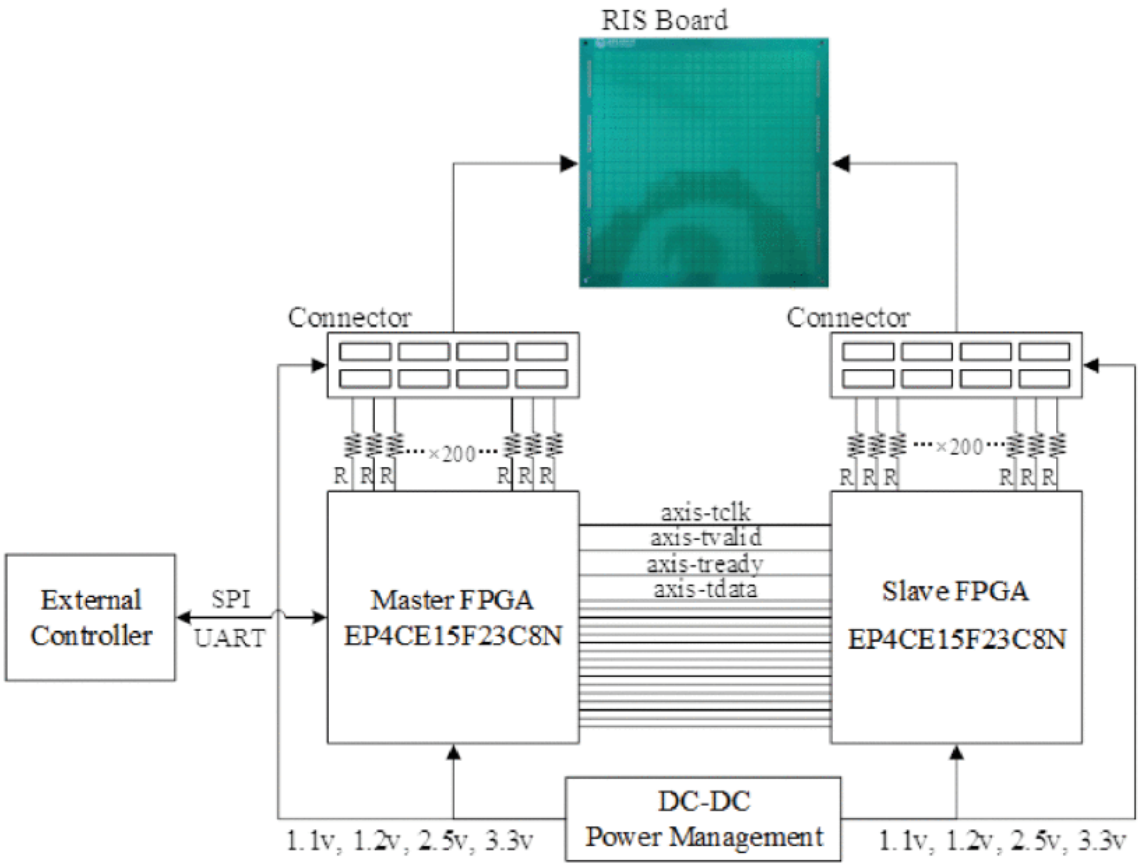

Figure 1.2: Schematic diagram of the control board [12]

gate array (FPGA) controllers with Intel Cyclone IV chips controlled using the Serial Peripheral Interface protocol (SPI), which allowed the control of 400 output pins with 100 MHz frequency. The control board receives control signals from the PC through the SPI and outputs signals to RIS diodes. The proposed setup's schematic diagram is demonstrated in Figure 1.2.

In [12], the authors were able to realize FPGA communication through a parallel 16-bit AXI-stream-4 bus consisting of 4 signal lines: 16-bit data, 1-bit ready, 1-bit valid, and 1-bit clock signals. The 100 MHz frequency clock rate allows to transfer of data between FPGA chips at high speed. The cascaded design presented in the paper is reusable for RIS with a larger number of cells. However, the system presented in the paper requires the PC to be constantly connected to the RIS system. The paper introduces three operating modes of the control board – index control mode, dynamic codebook (configurations of RIS) mode, and codebook download mode.

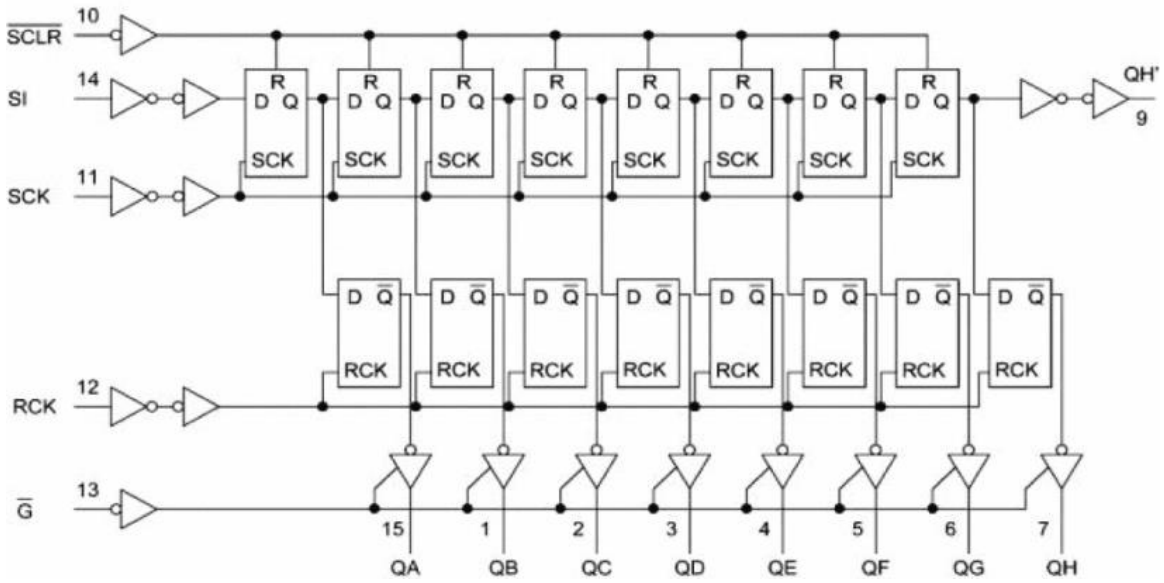

Figure 1.3: Shift register operating logic schematics [13]

All three modes work through an external connection to the PC, i.e., the SPI connection limits the operating speed of the system.

Another approach is to implement serial to parallel conversion using shift registers, as proposed in [13-14]. In [13], the authors were able to control 128 output pins using eight pins of an FPGA. They connected two shift registers cascaded, realizing 16-bit output using one pin of FPGA. Similar to [14], 160 output pins were controlled using eight shift registers. A shift register operates using two clock signals for synchronization, register clock (RCK) and shift register clock (SRCK). It works as follows: the control board sets the latch signal high, then serial input receives 8 bits and stores in the buffer. The received information is not sent to output until the latch signal is set to low [9]. Its logic is presented in Figure 1.3.

The power consumption factor should be considered as well when choosing the controller. Generally, FPGA-based systems show more power consumption and are more expensive than microcontrollers [15]. Despite this fact, authors suggest that FPGA is more suitable than microcontrollers for real-time applications as they can handle parallel processing

and high data throughput, thus providing scalability for RIS-aided systems. They also discussed the usage of microcontrollers such as ESP32, as they can support wireless Bluetooth or Wi-Fi connections. According to the authors, microcontrollers are well suited to indoor experimental setups.

Although the RIS development had quite a jump in recent years, most of the attention was towards theoretical modeling, such as in [16-17]. Few papers have proposed test-bed systems for real-world performance evaluation of the RIS.

In [10], the authors have proposed two prototypes of a RIS, one operating at 2.3 GHz and, the other operating at 28.5 GHz. Each unit cell comprises five PIN diodes, which leads to a more complex manufacturing process. PIN diodes are biased using a capacitor and inductor. This allowed the authors to change the reflection angle of the RIS. Their test setup consisted of the transmitter (Tx), placed close to the surface aperture, which makes the RIS operate as a reflectarray. In the real-world mobile network application the base station is placed kilometers away from the reflecting surface.

A much more credible experiment can be found in [18], where the Tx is set to a distance of approximately $80\lambda_0$ from the RIS. Here λ_0 is the signal wavelength at the RIS frequency. The authors of [19] presented a programmable surface that operates at roughly 11 GHz with a customizable polarization response and a focused beam. Each unit cell includes a PIN diode and a biasing circuit. The distance between Tx and the surface can be adjusted to provide a variety of responses. In line with this, when the structure is designed to produce a focused beam, the Tx should be placed near the surface aperture so that it can reflect the waves to the proper focal point. A longer Tx, on the other hand, results in a dynamic polarization response.

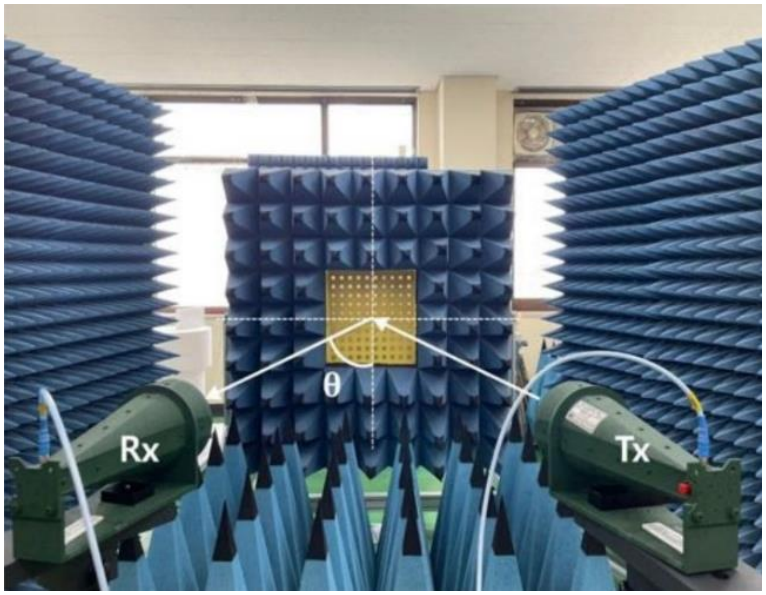

Figure 1.4: Placement of absorbers around the experimental setup [23]

In [20], the authors proposed a tunable impedance surface that is formed by adding four varactor diodes to four sides of a square-patch scatterer in each unit cell, resulting in a beam-reconfigurable reflectarray operating at 3.5 GHz. In [21-22], each unit cell had a couple of varactor diodes to create a programmable surface with an operating frequency of 4.25 GHz and a dual-polarized structure, respectively. Here dual polarization means that the RIS can alter electromagnetic waves reflected waves in two polarizations. In both papers the Tx was set close to the RIS and the receiver (Rx) was set farther.

The authors in [23] stated that special electromagnetic induction (EMI) absorbers should be placed on walls and around the RIS. The Tx and Rx should be positioned to avoid direct signal propagation. An absorber can be placed between the Tx and Rx as an obstacle. Their setup is shown in Figure 1.4.

This way the radiation from outside and wall reflections will have less effect on the receiver. Then, the Tx and the Rx should be placed at a certain angle toward the RIS. Configurations of the RIS are to be tested and measured to determine the best-performing one for

certain angles of incidence. The performance can be evaluated by measuring the effective channel gain and the reflection of radar cross-sections in the desired frequency range.

1.3 Aims and Objectives

RIS utilization is beneficial with its low loss, low cost, and flexible beamforming therefore it has been considered as a substitute method of configuring the wireless propagation environment [24]. However, prototyping of systems using the mentioned technology has been lacking. Even fewer prototypes were developed suitable for real-time applications.

In this research, I aim to implement a prototype of RIS for real-time applications using a suitable controller to test and measure it to determine the codebook for different scenarios The research objectives are:

- To evaluate the challenges of RIS prototyping using a controller for factors such as connection speed, synchronization and stability issues, and hardware and software development limitations.
- To develop an algorithm to efficiently control RIS boards to enable fast refresh rates of configurations.
- To design and construct a functional system consisting of RIS, controller, and LED board for demonstration, housed in a robust and highly compatible case.
- To measure the power consumed by the RIS system and its components.
- To build a chamber room suited for high-frequency signals and that absorbs wall reflections providing accurate results.
- To test the RIS performance under different scenarios using a predetermined codebook for various environmental conditions and use cases.
- Finally, come up with the best-performing configurations for different scenarios.

1.4 Thesis Structure

This thesis is organized as follows. Chapter 2 presents the methodology of prototype development, namely the processes done to identify the suitable controller, the development of the code for the controller, printed circuit board (PCB) design details, and chamber room description. Chapter 3 presents the results of the tests conducted. Chapter 4 offers a conclusion of the thesis and future work.

Chapter 2 – Method development

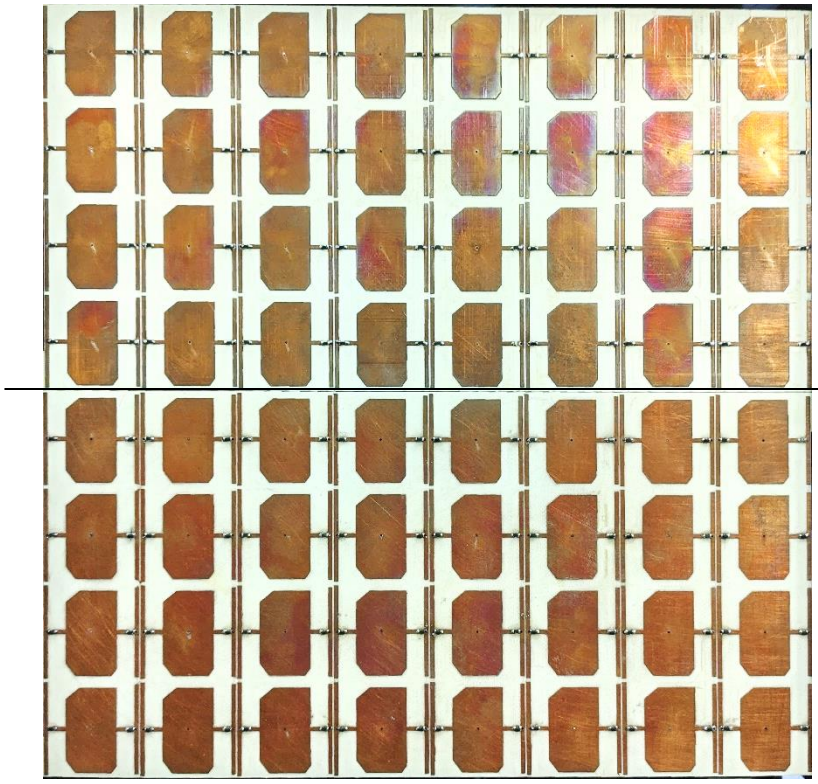

Figure 2.1: Two RIS boards stacked

2.1 Controller Identification

A key part of the thesis was to identify a suitable controller for its objectives. The lab has already designed and printed two RIS boards of 32 2-bit unit cells each, 128 diodes total, demonstrated in Figure 2.1.

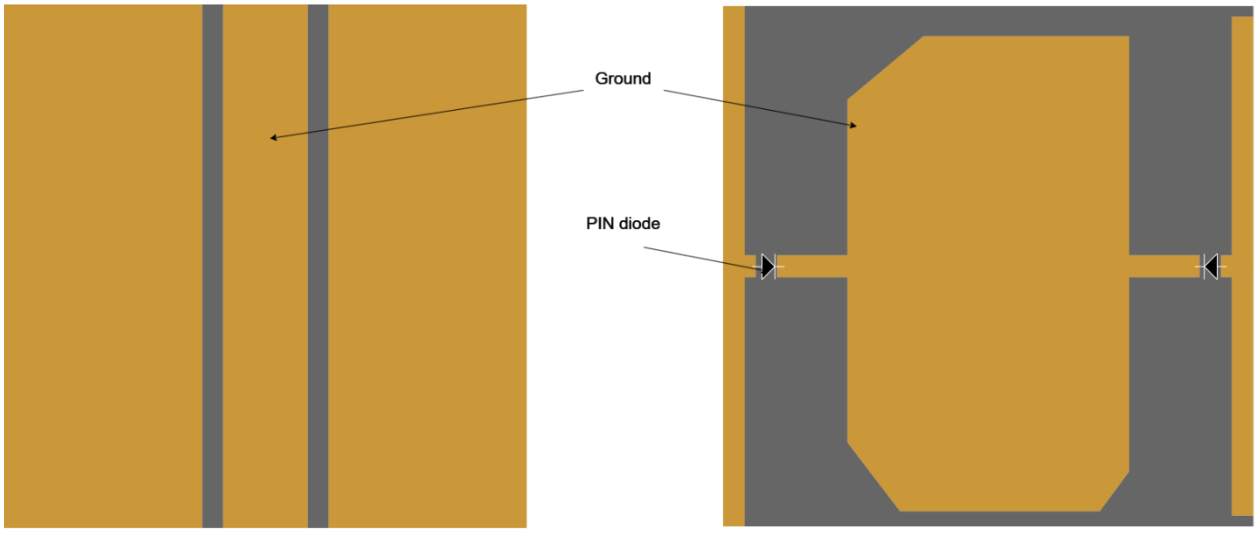

Figure 2.2: RIS unit cell bottom and top structure

Figure 2.2 displays the unit cell structure of the RIS implementation. It consists of a conducting scatterer of three patches on the FR4 substrate, where one patch is grounded, and two side patches are connected to it by SMPA1320-079LF PIN diodes. FR4 substrate proved to be very cost-effective [23]. The unit cells are 2-bit phase resolution, each cell can be set to states “00”, “01”, “10”, and “11”.

To make the prototype more robust and universal, the desired controller needs to support more than 256 general output pins. However, existing FPGA boards can not handle that requirement. As the authors stated in [13-14], serial data can be utilized to increase output pins for control. The next requirement was the clock frequency of the controller. There are lot of data to be transmitted, so a higher clock frequency is better.

The Altera DE1-SoC development board was a good choice since it has a higher clock speed (50 MHz) and more general-purpose output pins, combined with relatively low power consumption. However, it has only 36 output pins, which will be extended using shift registers, which convert serial input to parallel, allowing to drastically increase controllable pins.

Shift registers were to be carefully chosen as well since they have different clock frequencies. 74HC595 was chosen as it can run at 25 MHz and can output eight bits in parallel.

They can be combined in series to transmit up to 32 bits in parallel. This way cascaded connection of multiple shift registers was realized. four shift registers were connected in series and controlled using one pin on the board. The realization of cascaded shift registers in combination with parallel output allows powering 128 bits of the required output to control two RIS boards using only eight pins, with each pin connected to four cascaded shift registers. The minimum time to update RIS can be calculated using the formula adapted from [9], where n_p is the number of bits in parallel, N_s is the number of shift registers in series, and f is the frequency of the shift register.

$$t = \frac{n_p N_s}{f} = \frac{8 * 4}{25\ MHz} = 1.28\ \mu s$$

The De1-Soc code base was written in Verilog language and compiled in Altera Quartus Prime, which is a designated integrated development environment (IDE) for this SoC. It allows an assignment of pins of the board to variables in the code and simulates it using the Questa tool, which is useful to debug the code before connecting to RIS. Verilog is a hardware description language that uses behavioral-level modeling to describe the functionality of a design. It is compiled into a configuration of hardware, FPGA in this case, that allows control of its behavior [25].

In the progress of development of the software to control RIS behavior, it was discovered that “hard-coding” (i.e. allocating data in the memory inside the code before compilation) the codebook in the code leads to memory issues as there are hundreds of arrays that can not be compiled. To resolve this issue it was decided to write codebook configurations in separate files on the SD card and read from the HPS side of the De1-SoC. The HPS stands for Hard Processor System and consists of two ARM A9 processors, 1 GB SDRAM, various controllers, and an SD card socket. HPS can be controlled using Linux OS installed on the SD card. Connection to the

HPS was handled using a mini-USB UART port and by establishing a serial connection using PuTTY software. Files on the SD card are accessed using a Linux file management system [26].

The HPS was then connected to FPGA using the Qsys bus design tool integrated with Quartus Prime software. Qsys allows connections to the Intel/Altera Avalon bus and provides bridges to the HPS via the AXI bus. This tool allows the abstraction of bus width, timing, arbitration, and domain bridges to make design easier. In this design, Qsys was used as a memory-mapped, or address-mapped, system between the HPS and FPGA. Communication between HPS and FPGA was handled by AXI and lightweight AXI buses that can be 128 and 32 bits wide respectively [26].

The second problem that surfaced during development was slow interaction speed between the FPGA and the HPS memory, as well as restricted memory access on the FPGA side. After Qsys is integrated and the AXI bus is used to communicate between the HPS and FPGA, the memory access speed is not ideal, which makes it difficult to process and exchange data in real-time. 32-bit wide lightweight AXI bus requires segmenting 128 bits of data transmission from the HPS to FPGA, significantly reducing the interaction speed. The full-size AXI bus is used by the key components of the SoC, thus it is not suitable for frequent read-write operations. Moreover, the problem is made worse by the FPGA side's limited memory capacity, a size of 64 kB limits the amount of data that can be processed and stored locally [26].

One possible solution to the memory access and interaction speed limitations was to consider UART communication between the computer and FPGA. Nevertheless, it was discovered that UART packets are limited to nine bits, with only six bits of actual communication possible after one start bit, one parity bit, and one stop bit are preserved [26]. As such, this places a limit on the amount of data that can be sent in a single packet. Because of the

segmentation, each configuration must be transmitted over a minimum of 16 cycles, which adds complexity to the communication process and may cause latency problems.

The challenges raised by slow interaction speed and restricted memory access led to a move from the FPGA-HPS architecture towards the Raspberry Pi 4 controller. It has a quad-core ARM Cortex A72 processor running at up to 1.5 GHz, 8 GB of LPDDR4 RAM, a 40-pin GPIO header, and SD card storage that runs Linux-based OS [27-28]. Pi controller utilizes high-level structural languages like Python, C, etc. Since our application requires quick data read and write operations speed and fast loading into GPIO, higher processing speed enables us to achieve more responsive real-time interaction. Larger memory capacity and faster RAM frequency are also a crucial advantage over De1-SoC for the manipulation of large amounts of data. Furthermore, the memory architecture of the Raspberry Pi offers more abstract access to memory, which streamlines the development process and frees developers from having to worry about the complexities of low-level memory management. The switch to the Raspberry Pi 4 as the controller, taken together, offers a viable way to address the issues caused by slow interaction speed and restricted memory access. The user interacts with the Raspberry Pi 4 using the Secure Shell (SSH) protocol. This improves the system's mobility and availability. Codebooks can be precalculated and stored in the board’s SD card or dynamically calculated using its processing resources.

2.2 Software Development

Since shift registers were utilized in the system, the overhead delay in sending serial data is different, and a huge delay can be introduced without code optimization. To measure how fast the Pi controller sets up the RIS, a codebook with 128-bit long random sequences was generated and written to a text file on its storage. Then a code was developed to read sequences from the

text file and load them one by one into the GPIO where PCB headers were connected earlier. Time efficiency was measured using the “time” module.

Although Python is not traditionally associated with real-time applications since it is an interpreted language, still it can offer some advantages for some cases on the Raspberry Pi platform. It is easy to use, versatile, and has large library support [29]. Its readability and simplicity allow rapid prototyping of real-time applications which is crucial during the development process. Python has an extensive library of various modules many of which are excellent for implementing serial data output to GPIO. For example, the Raspberry built-in ‘RPi.GPIO’ module offers a simple and intuitive Application Programming Interface (API) to configure GPIO pins for output operations. However, this module is not suitable for real-time applications [30], but its simple use allows for the quick development of an algorithm for the code.

To determine the quickest algorithm for switching data and outputting to shift registers connected to GPIO, several languages, and their various modules were compared with each other.

First, the mentioned module ‘RPi.GPIO’ in Python was tested. This module and the overhead delay introduced by this module proved it. An alternative approach was to test the custom libraries built for Python. One of them is the ‘gpiozero’ module [31], which also did not show effective GPIO output speed. As a next step, the code was translated to C language, to utilize custom libraries ‘pigpio’ [32] and ‘pi-gpio’ [33]. The code excerpt can be found in Figure A.1.

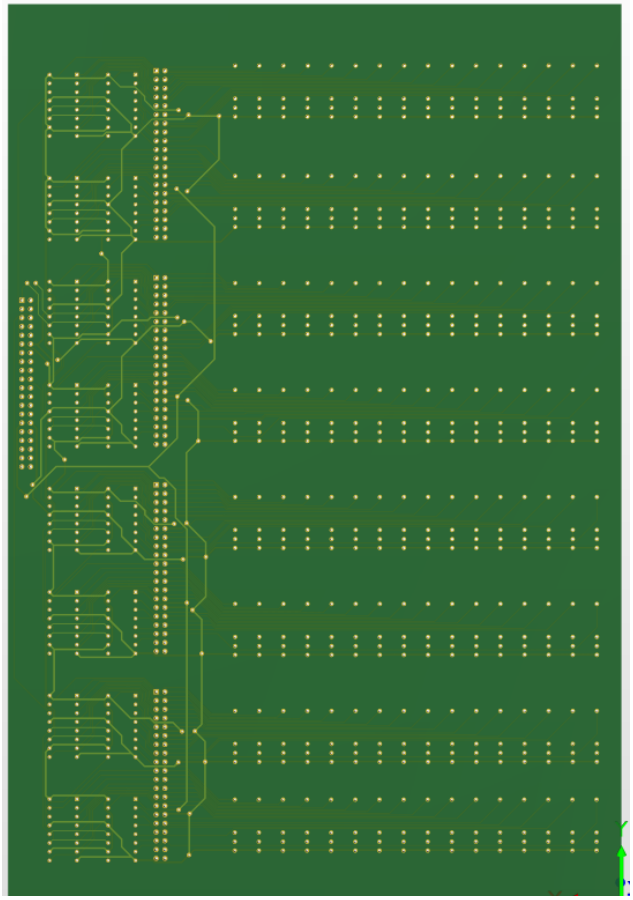

Figure 2.3: LED board model rendered in 3D in Altium Designer

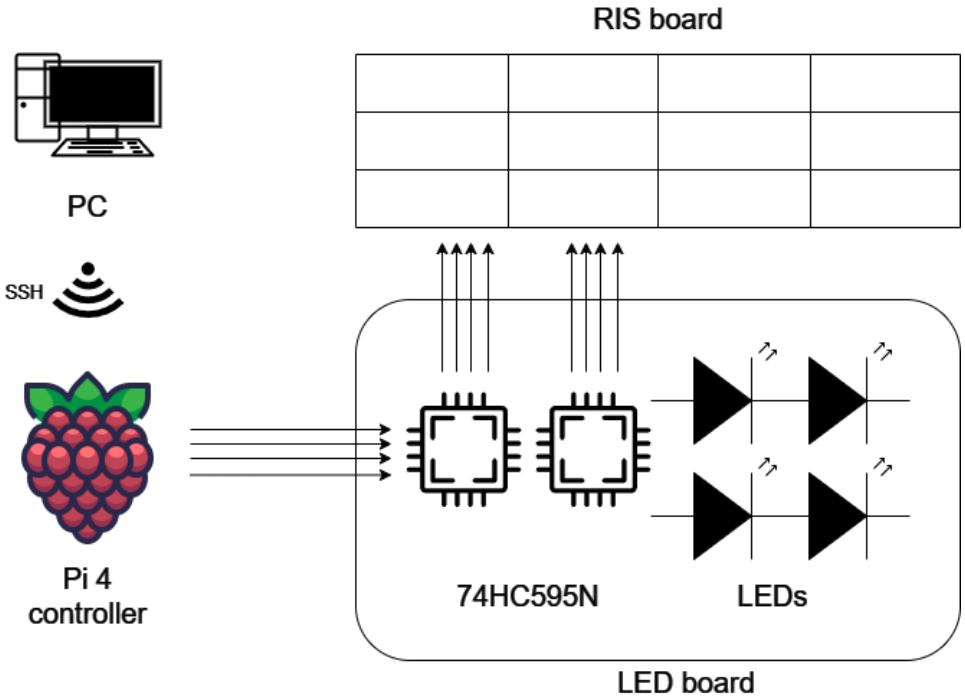

Figure 2.4: RIS system connection schematics

2.3 Designing the LED indicator board

The output of the RIS state to a special screen was developed to monitor which configuration was set to compare with measurements. LEDs were connected to show which cell diodes were turned on and off. Since RIS diodes are connected inversely, when the LED turns on, the corresponding RIS diode turns off. Resistors of 150 Ohm were connected in series to provide 2 V across LEDs and 0.8 V across RIS diodes. A further step was to design a PCB on Altium Designer to implement a 4x 20x2 GPIO extension header connection to the RIS board, displayed in Figure 2.3. Shift registers were also placed on this board and connected to the RIS. Here, four shift registers were connected in series so that 16 registers can output 128 bits using four serial input pins and two synchronization signal pins, six pins in total. The schematic for the board can be found in Figure B.1. The RIS system connection is presented as demonstrated in Figure 2.4.

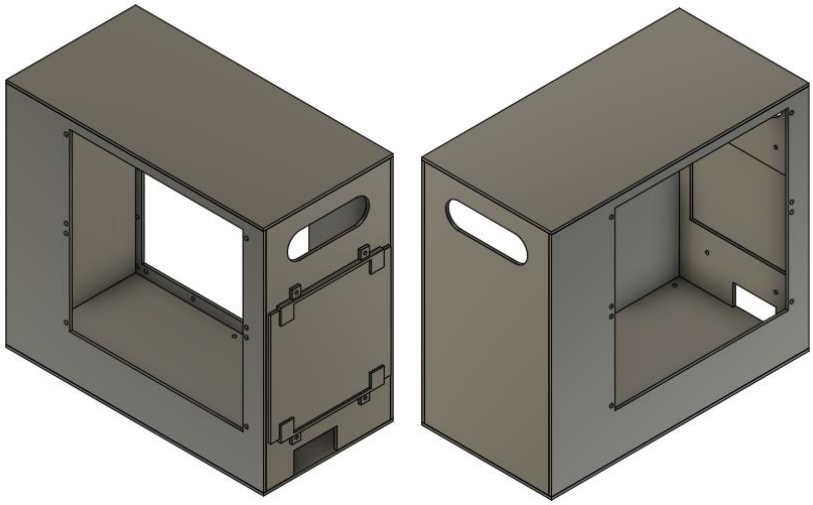

Figure 2.5: Rendered view of the case in 3D from two sides

2.4 Designing the System Case

Once the PCB model was printed and assembled, the RIS system case to hold together the RIS, LED indicating board, and the controller was constructed. To make the system portable and secure to hold all components a case was designed. The design of the case was developed in the Autodesk Fusion 360 software. The main criteria for the case materials were electromagnetic reflection and rigidity. The material is required to reflect minimum electromagnetic waves at a 6 GHz frequency. Additionally, it needs to be able to hold two RIS boards, the LED board, and the controller securely in place. Plexiglass proved to be the material of choice because plexiglass is both rigid and non-reflective of 6 GHz signals. One side of the case houses the two RIS boards and faces the measurement setup in the laboratory. The opposite side of the case accommodates the LED board and faces the observers. The final case design is demonstrated in Figure 2.5. The case dimensions and drawings can be found in Figure C.1 and C.2.

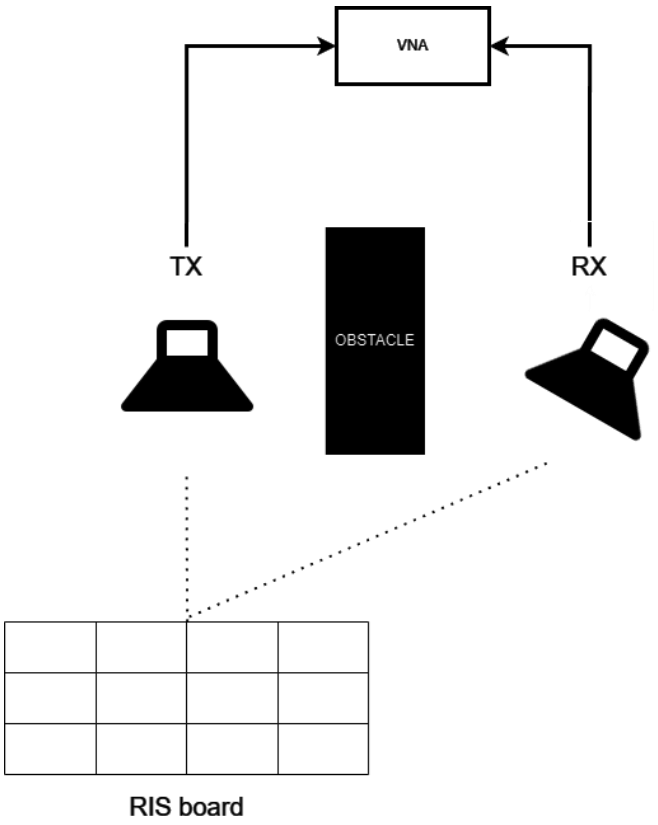

Figure 2.6: Channel measurement setup schematic

2.5 Measurements Setup

The channel measurement setup schematic is illustrated in Figure 2.6. Here Keysight PNA-X vector network analyzer (VNA) was utilized to measure the $S_{21}(f)$ parameter. The Tx and Rx were HF907 horn antennas connected to ports 1 and 2 of the VNA, aligned to the RIS center. The Tx incident angle was 0° but a variable distance, while the Rx had a variable angle and distance. The RIS operates at a frequency of 6 GHz. The non-line-of-sight (NLOS) path needed to be established between the Tx and Rx using an absorber.

The RIS-assisted channel path loss (PL) for the frequency domain between the Tx and Rx was adapted from [34]. It can be calculated from $S_{21}(f)$ as follows:

$$L_C[dB] = L_{cable} + L_{exp} - L_{VNA} + (G_t + G_r)$$

Here, L_{cable} is connection cable loss, L_{exp} is experimental data measured by the VNA, L_{VNA} is the internal loss of the VNA, when two ports are connected directly to each other, G_t and G_r are Tx and Rx amplification gains.

The channel impulse response (CIR) can be calculated by taking the Inverse Fourier Transform of $S_{21}(f)$ [34]:

$$h(t,\tau) = IFT(S_{21}(f) \times W_{hann})$$

Here, W_{hann} is Hann-window. The power delay profile (PDP) can be obtained as follows

$$P_h(\tau) = |h(t,\tau)|^2$$

The PDP gives the channel's time domain characteristics.

2.6 Preparation of the Chamber Room

Setting up the measurements setup required careful investigation and research of models available in the market to determine which are best within the intended frequency range of 5-30 GHz. Particularly, special focus on the material to cover the walls, ceiling, and floor of the room. Absorbers reduce unwanted reflections and enable anechoic conditions in the measurement room, which is a key factor in RIS testing.

The key characteristics are insertion loss, reflection coefficient, and absorption efficiency over the working frequency range. The models in the market were assessed to have at least -40 dB reflection loss. Also, the dimensions of the absorbers and their installation requirements were considered to be compatible with the 5 m * 5 m * 3 m dimensions of the chamber room.

Chapter 3 – Testing & Results

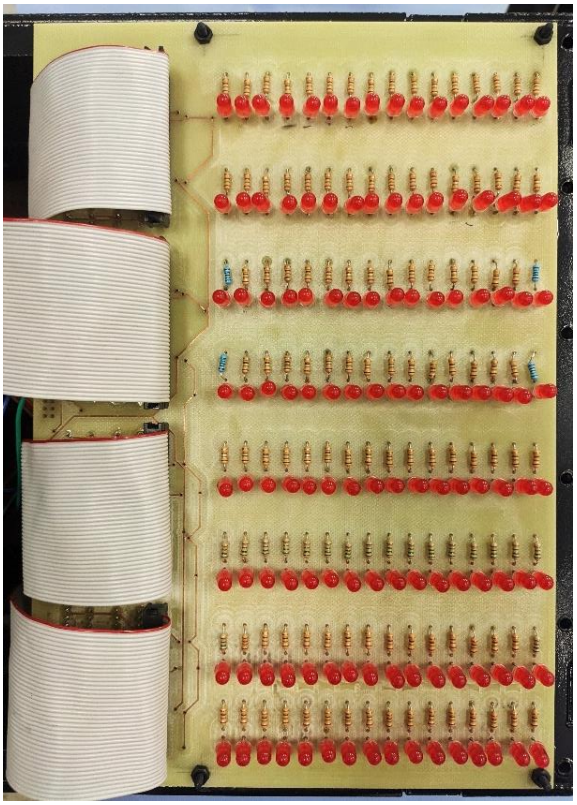

Figure 3.1: PCB assembled and connected to the system

This study's results section attempts to offer a thorough examination of the RIS prototyping procedure utilizing a controller.

3.1 System Assembly

The LED board was assembled using the components described above. It comprises 128 pc. red colored LED, 128 pc. 150 Ohm resistors, 5 pc. 20x2 GPIO headers, and 16 pc. 74HC595 shift registers soldered as demonstrated in Figure 3.1. Then all components, including RIS boards, LED board, and Raspberry Pi, were installed on the case, as shown in Figure 3.2. Nylon screws were used to fix components in place to minimize reflections, as metallic materials greatly reflect signals. The case design proved to be successful – even though it was not tested for electromagnetic reflections, the correct material made the system durable and strong.

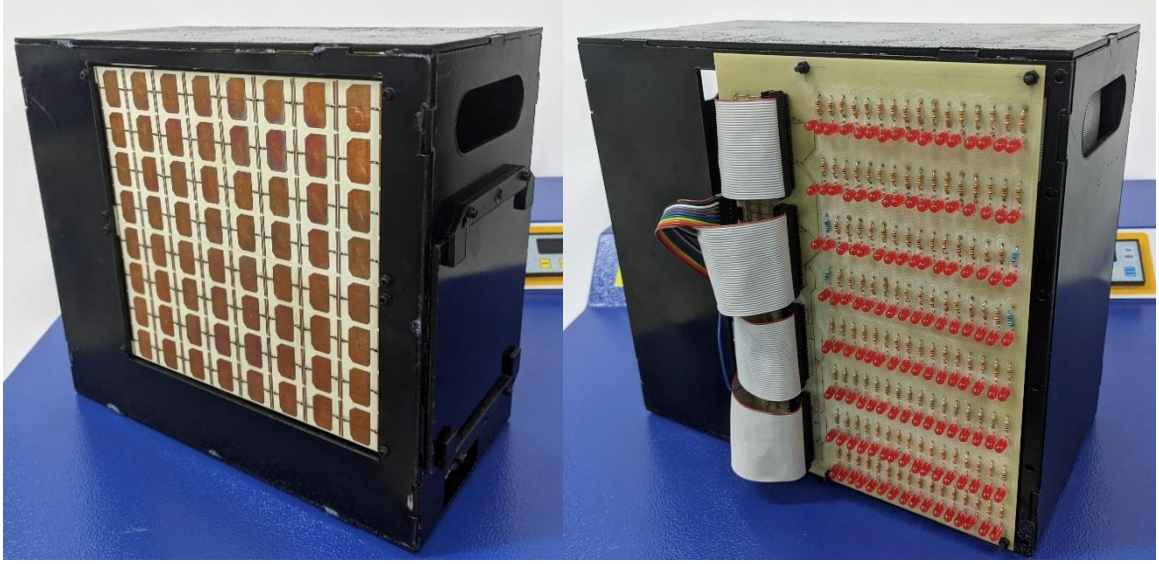

Figure 3.2: Assembled case from two sides

3.2 Code Optimization

The delay was measured by switching data sequences that were read from a text file and then sent to GPIO pins. The first step involved measuring the amount of time needed to load a single 128-bit word. Next, the time spent loading 10,000 128-bit word elements was evaluated. 10,000 consecutive elements were read from the text file and sent to GPIO. The results for different libraries are shown in Table 3.1.

The time taken to load a single 128-bit word is a basic indicator of how well the system performs at handling data transmission tasks. The fastest response rate was achieved using the ‘pi-gpio’ library on C. It allows to output data to the GPIO at a scale of microseconds. It was able to complete the task of loading 10000 words within 70 milliseconds, which gives a thorough insight into the system’s ability to operate in real-time demands when the RIS states need to be changed as fast as possible for seamless connection with the receiver.

Table 3.1: Comparison of delay using different libraries

Module (Language)	Loading a single word of 128-bit (ms)	Loading 10000 words of 128-bit (ms)
RPi.GPIO (Python)	2.44	13240
gpiozero (python)	1.81	11840
Pigpio (C)	0.02	137
Pi-gpio (C)	0.011	70

As a next step, alternating sequences were sent to GPIO pins using the 'RPi.GPIO' library on Python and 'pi-gpio' on C, and LED switching frequencies were measured using a digital oscilloscope. The first library frequency was indicated to be 4.091 kHz, while the second library frequency reached 97.02 kHz. The frequencies were more or less correlated with the delay calculated using internal methods. The second library frequency is more than 20 times greater than the Python library frequency. The oscilloscope measures can be found in Figure D.1 and Figure D.2. The overhead delay that was present when using Python was completely mitigated when the algorithm was translated to C.

However, the LED board showed that shift registers are unable to support such high refresh rates, which confirms the limitations presented in [10]. The 'gpiozero' module in Python testing showed that LEDs are indeed switching, while the C libraries could not demonstrate it. A possible solution is to artificially delay the output part in the algorithm. The instability in the system could be also affected by the Pi 4 board's GPIO timing issues and LED internal switching delay.

3.3 Power Consumption Measurement

The power consumption of the RIS system and its components need to be accurately measured. The Raspberry Pi is powered by an electrical socket adapter that can output 20 W. Although in theory, the RIS and LED board can not consume such amounts of power, exceeding

Table 3.2: Comparison of power consumption of RIS and PCB

Board	All diodes turned ON (W)	All diodes turned OFF (W)
RIS	3.524	0
LED	4.25	0.015

it may negatively affect the RIS performance. The current was measured using a digital multimeter at different points across the RIS system with all diodes both turned on and off, which allows to estimate its average power consumption. The RIS and LEDs were measured when activated and deactivated. The results are displayed in Table 3.2.

The system showed moderate power consumption. The RIS board consists of 64 unit cells, thus the average RIS unit cell power consumption is approximated to be 55 mW. Since the RIS and LED boards are connected inversely, overall power consumption did not exceed 5 Watts, which is acceptable for systems with Raspberry Pi 4.

3.4 RIS Measurements

Unfortunately, the chamber room was not built in the thesis period, and the RIS measurements could not be done accurately. The main obstacle – the chamber room planning process was delayed by the De1-SoC controller issues described in Chapter 2.1.

Figure 3.3 displays the measurement setup featuring the VNA, two horn antennas, and an assembled RIS system, controlled by Raspberry Pi 4.

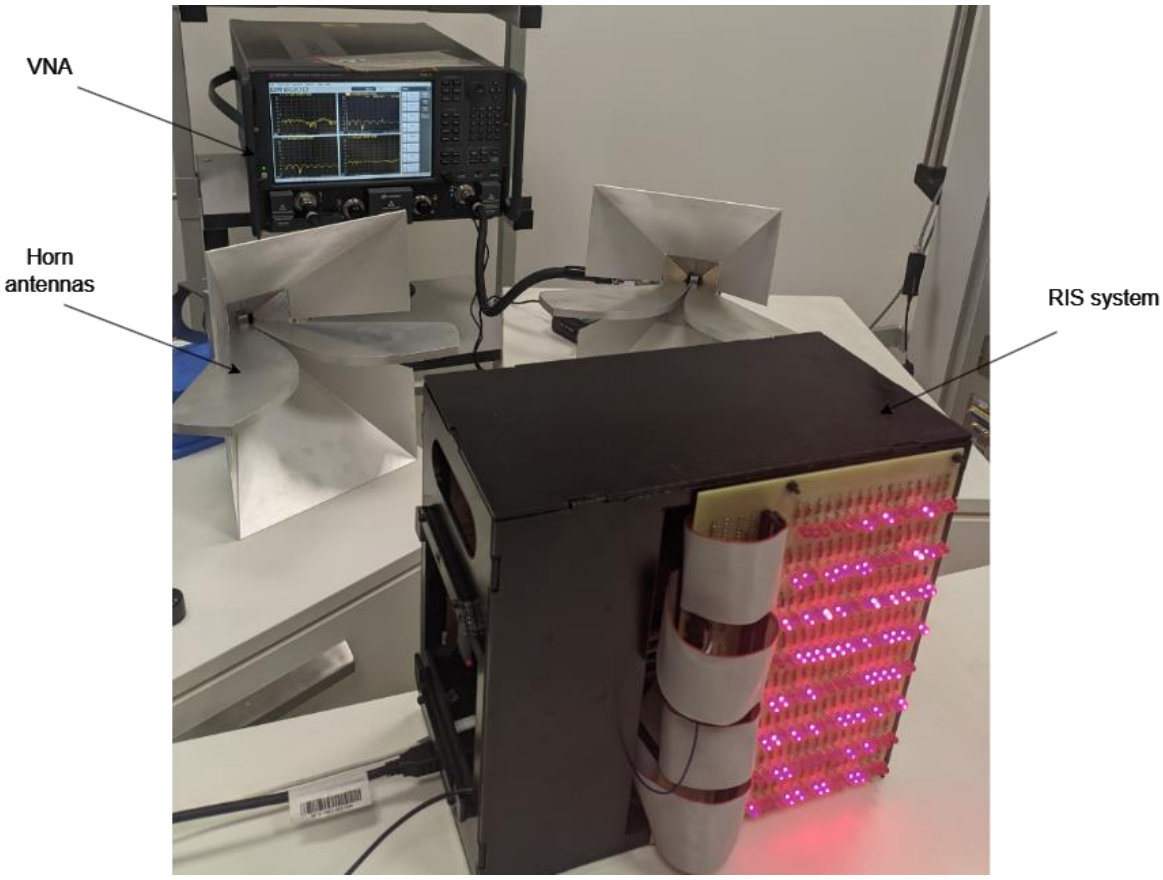

Figure 3.3: Channel measurement setup

Chapter 4 – Conclusion & Future Work

In summary, the thesis's ambitious scope to test the RIS prototype in a special chamber room was not fully achieved. Still, it established a strong framework for further development of the RIS prototype. The initial literature review, hardware and software development, and experimental setup procedure provide a strong basis for subsequent work. The original goals set for this thesis were met partially, the challenges being the time constraints and limitations in resources.

The designed system can meet real-time demands for seamless connection with the receiver using the Raspberry Pi 4 as a controller, which was coded in C language. The codebook sequences can be sent to the RIS with 97 kHz frequency. The overall power consumption of the system was 5 W at its peak, which is acceptable for the chosen controller.

The LED board and system case can be optimized in future work, for example by using surface-mounted devices for the board, hiding the connectors, and making the case more universal for different types of controllers. Alternative approaches using different protocols and controllers to resolve LED timing issues can be developed and tested in future research directions. Although the accurate measurement of the RIS was not completed in this study, it can be addressed in future works when the chamber room is fully assembled. Also, outdoor performance can be measured, where noise signals have more impact on measurements, which gives a more accurate insight into how the device performs in the real world.

Going forward, the knowledge acquired from the completed tasks can guide future research endeavors targeted at improving measurement techniques, RIS configuration optimization, and further investigating the possible uses of this technology in various real-world

situations. The field of RIS technology has immense potential and the development of its prototypes can accelerate the growth of wireless network quality and availability.

Bibliography

[1] C. Pan *et al.*, "Reconfigurable Intelligent Surfaces for 6G Systems: Principles, Applications, and Research Directions," in *IEEE Communications Magazine*, vol. 59, no. 6, pp. 14-20, June 2021, doi: 10.1109/MCOM.001.2001076.

[2] X. Pei *et al*., "RIS-Aided Wireless Communications: Prototyping, Adaptive Beamforming, and Indoor/Outdoor Field Trials," in *IEEE Transactions on Communications*, vol. 69, no. 12, pp. 8627-8640, Dec. 2021, doi: 10.1109/TCOMM.2021.3116151.

[3] E. Björnson, H. Wymeersch, B. Matthiesen, P. Popovski, L. Sanguinetti and E. de Carvalho, "Reconfigurable Intelligent Surfaces: A signal processing perspective with wireless applications," in *IEEE Signal Processing Magazine*, vol. 39, no. 2, pp. 135-158, March 2022, doi: 10.1109/MSP.2021.3130549.

[4] C. Liaskos, S. Nie, A. Tsioliaridou, A. Pitsillides, S. Ioannidis and I. Akyildiz, "A New Wireless Communication Paradigm through Software-Controlled Metasurfaces," in *IEEE Communications Magazine*, vol. 56, no. 9, pp. 162-169, Sept. 2018, doi: 10.1109/MCOM.2018.1700659.

[5] Q. Wu and R. Zhang, "Towards Smart and Reconfigurable Environment: Intelligent Reflecting Surface Aided Wireless Network," in *IEEE Communications Magazine*, vol. 58, no. 1, pp. 106-112, January 2020, doi: 10.1109/MCOM.001.1900107.

[6] S. Payami *et al*., "Developing the First mmWave Fully-Connected Hybrid Beamformer With a Large Antenna Array," in *IEEE Access*, vol. 8, pp. 141282-141291, 2020, doi: 10.1109/ACCESS.2020.3013539.

[7] A. Araghi *et al*., "Reconfigurable Intelligent Surface (RIS) in the Sub-6 GHz Band: Design, Implementation, and Real-World Demonstration," in *IEEE Access*, vol. 10, pp. 2646-2655, 2022, doi: 10.1109/ACCESS.2022.3140278.

[8] J. Hu *et al.*, "Reconfigurable Intelligent Surface Based RF Sensing: Design, Optimization, and Implementation," in *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 11, pp. 2700-2716, Nov. 2020, doi: 10.1109/JSAC.2020.3007041.

[9] V. Tapio, I. Hemadeh, A. Mourad, A. Shojaeifard, and M. Juntti, “Survey on reconfigurable intelligent surfaces below 10 GHz,” *EURASIP Journal on Wireless Communications and Networking*, vol. 2021, no. 1, Sep. 2021, doi: 10.1186/s13638-021-02048-5.

[10] L. Dai et al., “Reconfigurable Intelligent Surface-Based Wireless Communications: Antenna Design, Prototyping, and Experimental Results,” in *IEEE Access*, vol. 8, pp. 45913-45923, 2020, doi: 10.1109/ACCESS.2020.2977772.

[11] T. J. Cui, M. Q. Qi, X. Wan, J. Zhao, and Q. Cheng, “Coding metamaterials, digital metamaterials and programmable metamaterials,” *Light: Science & Applications*, vol. 3, no. 10, pp. e218–e218, Oct. 2014, doi: 10.1038/lsa.2014.99.

[12] M. Ouyang, F. Gao, Y. Wang, S. Zhang, P. Li, and J. Ren, "Computer Vision-Aided Reconfigurable Intelligent Surface-Based Beam Tracking: Prototyping and Experimental Results," in *IEEE Transactions on Wireless Communications*, doi: 10.1109/TWC.2023.3264752.

[13] X. Zeng *et al.*, "High-Accuracy Reconfigurable Intelligent Surface Using Independently Controllable Methods," *2021 IEEE International Workshop on Electromagnetics: Applications and Student Innovation Competition (iWEM)*, Guangzhou, China, 2021, pp. 1-3, doi: 10.1109/iWEM53379.2021.9790555.

[14] G. C. Trichopoulos *et al.*, "Design and Evaluation of Reconfigurable Intelligent Surfaces in Real-World Environment," in *IEEE Open Journal of the Communications Society*, vol. 3, pp. 462-474, 2022, doi: 10.1109/OJCOMS.2022.3158310.

[15] J. Herbst *et al.*, "Reconfigurable Intelligent Surfaces: About Applications and Their Implementation," *Mobile Communication - Technologies and Applications; 26th ITG-Symposium*, Osnabrueck, Germany, 2022, pp. 1-7.

[16] Q. Wu and R. Zhang, "Intelligent Reflecting Surface Enhanced Wireless Network via Joint Active and Passive Beamforming," in *IEEE Transactions on Wireless Communications*, vol. 18, no. 11, pp. 5394-5409, Nov. 2019, doi: 10.1109/TWC.2019.2936025.

[17] Z. Chu, P. Xiao, M. Shojafar, D. Mi, J. Mao and W. Hao, "Intelligent Reflecting Surface Assisted Mobile Edge Computing for Internet of Things," in *IEEE Wireless Communications Letters*, vol. 10, no. 3, pp. 619-623, March 2021, doi: 10.1109/LWC.2020.3040607.

[18] M. Dunna, C. Zhang, D. Sievenpiper and D. Bharadia, "ScatterMIMO: Enabling virtual MIMO with smart surfaces", *Proc. 26th Annu. Int. Conf. Mobile Comput. Netw.*, pp. 1-14, Apr. 2020.

[19] H. Yang, X. Cao, F. Yang, J. Gao, S. Xu, M. Li, et al., "A programmable metasurface with dynamic polarization scattering and focusing control", *Sci. Rep.*, vol. 6, Oct. 2016.

[20] D. F. Sievenpiper, J. H. Schaffner, H. J. Song, R. Y. Loo and G. Tangonan, "Two-dimensional beam steering using an electrically tunable impedance surface," in *IEEE Transactions on Antennas and Propagation*, vol. 51, no. 10, pp. 2713-2722, Oct. 2003, doi: 10.1109/TAP.2003.817558.

[21] W. Tang *et al*., "Wireless Communications with Programmable Metasurface: New Paradigms, Opportunities, and Challenges on Transceiver Design," in *IEEE Wireless Communications*, vol. 27, no. 2, pp. 180-187, April 2020, doi: 10.1109/MWC.001.1900308.

[22] X. Chen *et al*., "Design and Implementation of MIMO Transmission Based on Dual-Polarized Reconfigurable Intelligent Surface," in *IEEE Wireless Communications Letters*, vol. 10, no. 10, pp. 2155-2159, Oct. 2021, doi: 10.1109/LWC.2021.3095172.

[23] B. Rana, S.-S. Cho, and I.-P. Hong, “Parameters and Measurement Techniques of Reconfigurable Intelligent Surfaces,” *Micromachines*, vol. 13, no. 11, p. 1841, Oct. 2022, doi: 10.3390/mi13111841.

[24] K. Tang *et al*., "Reconfigurable Intelligent Surface Enhancing In-door Wireless Communication," *2021 IEEE International Workshop on Electromagnetics: Applications and Student Innovation Competition (iWEM)*, Guangzhou, China, 2021, pp. 1-3, doi: 10.1109/iWEM53379.2021.9790360.

[25] D. Thomas and P. Moorby, *The Verilog® Hardware Description Language*, 5th ed. New York, NY Springer, 2008, pp. 11–13.

[26] “DE1-SoC User Manual” 2014. Available: http://mems.ece.dal.ca/eced4260/DE1.pdf

[27] A. Nimje, S. Joshi, P. Ghutke, W. Patil and S. Sorte, "RasPiChrono: A Raspberry Pi Based Economical 3-in-1 Countdown Timer," *2023 11th International Conference on Emerging Trends in Engineering & Technology - Signal and Information Processing (ICETET - SIP)*, Nagpur, India, 2023, pp. 1-6, doi: 10.1109/ICETET-SIP58143.2023.10151479.

[28] S. J. Johnston and S. J. Cox, “The Raspberry Pi: A Technology Disrupter, and the Enabler of Dreams,” *Electronics*, vol. 6, no. 3, 2017, doi: 10.3390/electronics6030051.

[29] R. Krauss, "Real-Time Python: Recent Advances in the Raspberry Pi Plus Arduino Real-Time Control Approach," *2020 American Control Conference (ACC)*, Denver, CO, USA, 2020, pp. 2088-2093, doi: 10.23919/ACC45564.2020.9147236.

[30] B. Croston, “RPi.GPIO: A Module to Control Raspberry Pi GPIO Channels,” *PyPI*. Available: https://pypi.org/project/RPi.GPIO/

[31] “gpiozero — gpiozero 2.0.1 Documentation,” *gpiozero.readthedocs.io*. Available: https://gpiozero.readthedocs.io.

[32] “pigpio library,” *abyz.me.uk*. Available: https://abyz.me.uk/rpi/pigpio/

[33] I. Binnie, “Milliways2/pi-gpio,” *GitHub*. Available: https://github.com/Milliways2/pi-gpio/

[34] J. Lan *et al.*, "Measurement and Characteristic Analysis of RIS-assisted Wireless Communication Channels in Sub-6 GHz Outdoor Scenarios," *2023 IEEE 97th Vehicular Technology Conference (VTC2023-Spring)*, Florence, Italy, 2023, pp. 1-6, doi: 10.1109/VTC2023-Spring57618.2023.10200072.

Appendices

Appendix A

```
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <unistd.h>

#include <pi-gpio.h>

#define PIN_DATA1 16
#define PIN_DATA2 17
#define PIN_DATA3 27
#define PIN_DATA4 22
#define PIN_LATCH 20
#define PIN_CLOCK 21

#define LINE_SIZE 129

int main() {
    setup();

    setup_gpio(PIN_DATA1, 1, 0);
    setup_gpio(PIN_DATA2, 1, 0);
    setup_gpio(PIN_DATA3, 1, 0);
    setup_gpio(PIN_DATA4, 1, 0);
    setup_gpio(PIN_LATCH, 1, 0);
    setup_gpio(PIN_CLOCK, 1, 0);

    FILE *file;
    char line[LINE_SIZE];
    int x;
```

Figure A.1 Code excerpt

Appendix B

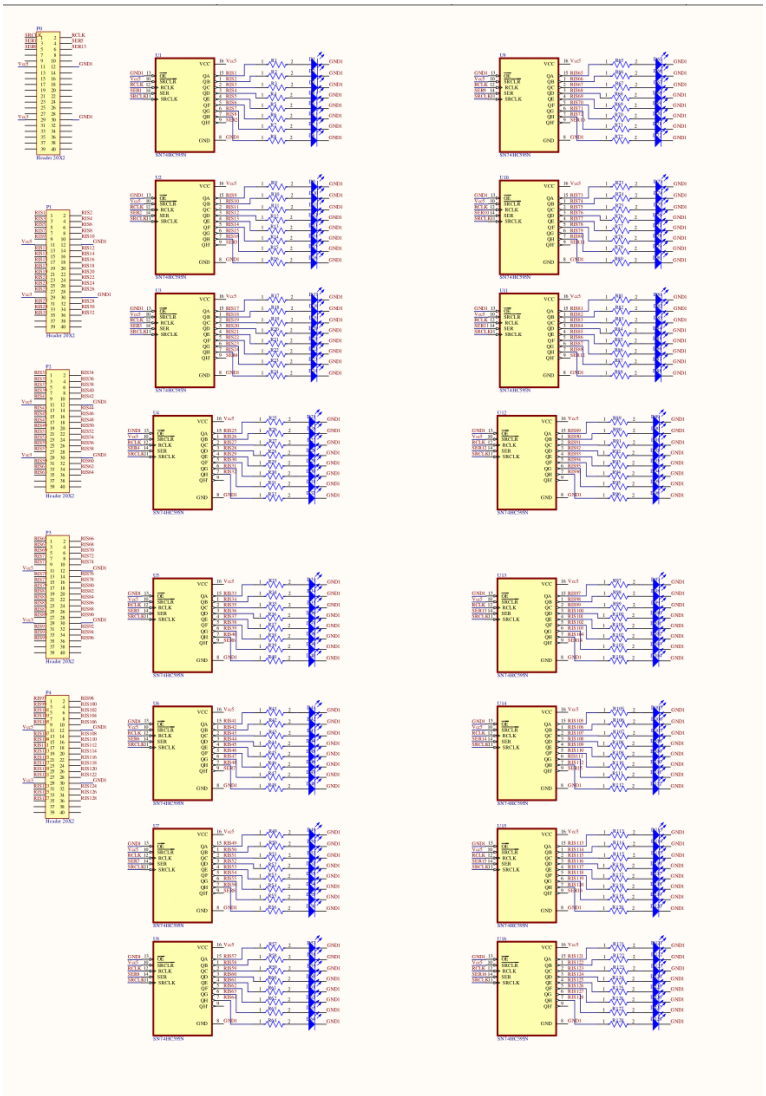

Figure B.1 LED board schematics

Appendix C

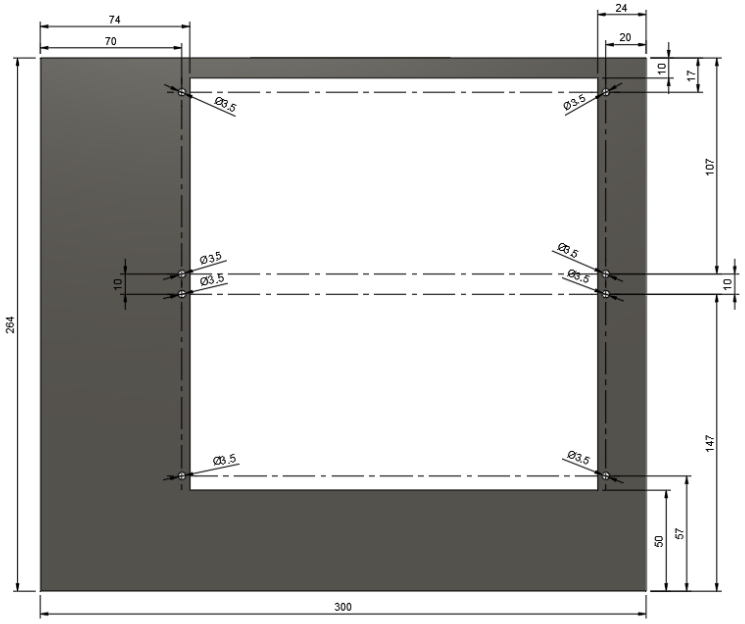

Figure C.1 Case front dimensions

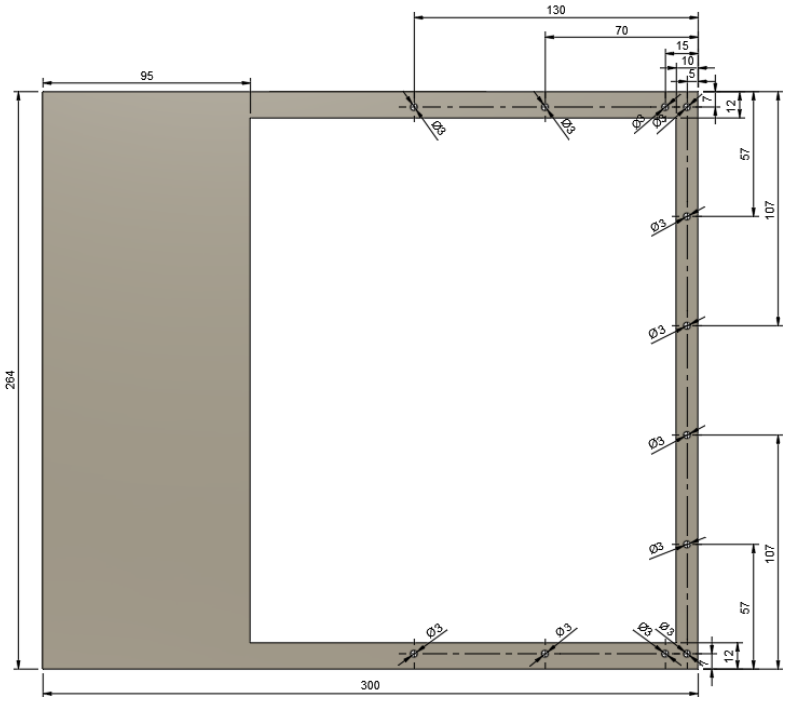

Figure C.2 Case back dimensions

Appendix D

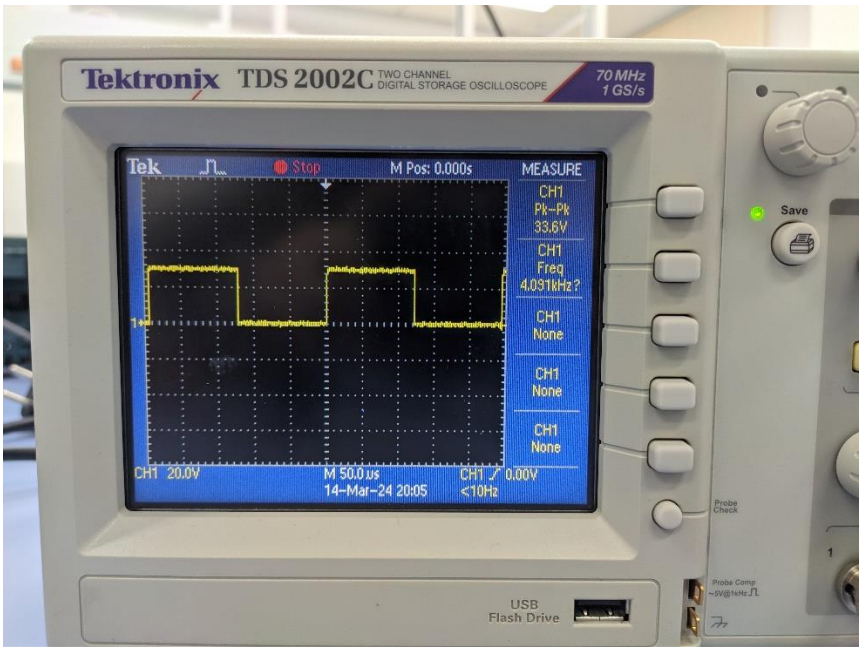

Figure D.1 LED switching frequency using 'RPi.GPIO' on Python

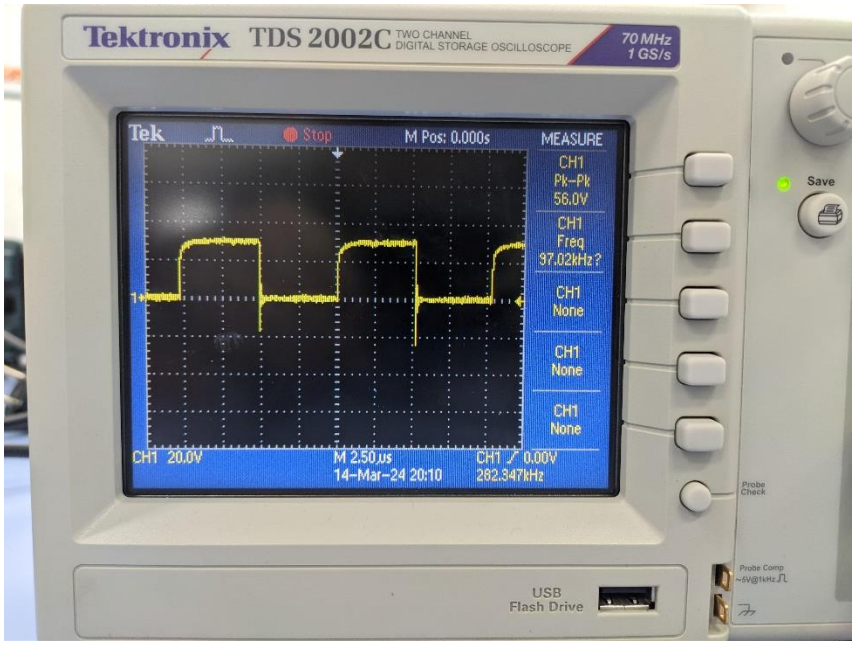

Figure D.2 LED switching frequency using 'pi-gpio' on C