RFID Inventory System (Final Report)

Avi Yadav (20D070018)
Hemant Hajare (20D070037)
Kaustubh Chourasia (20D070046)

March 2023

1 Introduction

The primary aim of this project was to develop a hand-held Radio Frequency Identification (RFID) reader to track the inventory in the Wadhwani Electronics Lab (WEL). Each item in the lab will be associated with a corresponding RFID tag. This tag will be scanned either during checking in of the item or checking out, to update the item's status on the tag. We have completed our project, and here is the summary of the steps taken during the project.

We divided the work into three milestones: Conceptual Design Review, Preliminary Design Review, and Critical Design Review. In order to meet the timeline and perform better management of the progress of our project, we followed systems engineering practices. We divided our system into three subsystems: The communication subsystem, the Onboard computing subsystem, and the PCB Design subsystem marking precise task distribution among them.

Next, we noted down the system's requirements from our subsystems and requirements among different subsystems. After this, we set up the timeline of the entire project considering several factors like lead times of components, availability of testing equipment, availability of faculty advisor, and academic schedule. We ensured that the deadlines were hard enough to ensure the project's completion and simultaneously provide room for rectifying errors that might arise during our project.

The following report is a detailed account of the decisions taken during various project reviews. This report is also a detailed documentation of our RFID-based inventory management system. We have covered as many details as possible to rectify issues, if any, during the functioning of our system. We have also added some of our thought based on our experience during the project, although they are subjective and change from one user to another.

2 Preliminary Design Review (PDR)

We did a Preliminary Design Review to address our project's fundamental questions. This involved choice of components, design of the flow of information, type of user interface, and plans for preliminary tests, which we have explained in further detail in the subsequent subsections.

2.1 Component Selection

During the project, we realized that performing a thorough analysis of different components in the market is crucial. A good component has detailed documentation and satisfies the requirements one has set up for the project. It should have an extensive user network to address basic queries one may have during his/her project. It should be readily available at a reasonable lead time and cost. Compromising one or more of these criteria would lead to a stall in the progress of our project.

2.1.1 Computing unit

Several micro-controller units are available in the market. Among the popular ones which we came across are Arduino, RP2040 (by Raspberry Pi Foundation), ESP32, STM32, TivaC (by Texas Instruments), ATxmega, and ATmega (Microchip Studios), among others.

A suitable micro-controller has sufficient General Purpose Input Output (GPIO) pins that suit your application. It should have good documentation with example codes for interfacing components like ADCs and DACs. While most micro-controller units can be programmed using embedded C and controller-specific assembly languages, others can be programmed using higher-level languages like C/C++ and Circuit Python/Micro Python. Higher-level languages often have built-in library functions for interfacing simple components using UART, SPI, and I2C protocols that are handy while developing any embedded system project. Some micro-controller systems come with onboard ADCs, which might seem attractive. However, the architecture ties down these ADC with the microprocessor. With well-commented example codes, it becomes easier to deal with them. Another feature that many controllers develop is a software emulator, which acts like a debugger giving one access to the memory and probing the GPIOs. These are some of the considerations that one should make, which will help immensely throughout the project.

The micro-controller we chose for our project was Raspberry Pi Pico. Several programming languages, like MicroPython/CircuitPython and C/C++, can program Raspberry Pi Pico. Our project had a strict requirement for using only C/C++/embedded C for programming, which made it into our list. Following are other features of Raspberry Pi Pico that make it a popular choice of micro-controller.

1. An official C/C++ Software Development Kit compatible with Windows and Linux-based operating systems
2. 26 GPIOs which support UART, SPI, and I2C interface
3. an extensive user network with a dedicated forum that addresses almost all fundamental problems
4. A support for programming using Arduino IDE and official documentation for the same

Arduino is known for its extensive documentation and support, along with many libraries for various components. The pico-arduino-compat available for Raspberry Pi Pico and Arduino compatibility exploits these features.

2.1.2 RFID High Frequency IC

During our PDR, to maintain novelty in our design among other groups, we decided to use High Frequency (13.56MHz) transmission for our data. Thus, we chose a popular MFRC522 IC available in the market. MFRC522 is a Quad Flat Package with a total of 32 pins. Several considerations come up when working with such an IC.

Testing QFIC is only possible by soldering it on a PCB; preliminary testing using a breadboard or perforated board is not an option. One immediate solution to this problem is using a breakout board for the IC. A breakout board makes it easier to carry out multiple tests without having to solder and desolder the IC on the final PCB. Soldering a QFIC requires skills, and one might burn the IC internally while using the hot air rework gun.
Standard breakout boards are available in the market in various dimensions. Choosing the proper dimension and shape for the breakout board is essential by considering the testing setup. Most square-shaped boards do not fit on breadboards conveniently, which might make the breakout board obsolete.

Our project group learned this lesson the hard way. We could not foresee this and thus could only carry out any tests once we received our PCBs. We had to replace faulty ICs repeatedly, causing inconvenience. However, we rectified our flaw and later performed all our tests on the perforated board, causing a setback in our timeline.

2.1.3 Liquid Crystal Display

In order to have an interactive user interface, we decided to use a Hitachi-based 10-pin Liquid Crystal Display. An LCD is a standard component widely used with many micro-controllers. Almost all the significant controllers we have discussed have an elaborate library for LCD interfacing. We exploited the Arduino library for interfacing the LCD with Raspberry Pi Pico using the pico-arduino-compat.

One immediate drawback of any LCD interface is the number of GPIOs on the controller it uses up. Though we were not concerned regarding this problem, it was pointed out to our faculty advisor to use a more elegant LCD-I2C module to interface the LCD. This module maps ten LCD pins to 4-pin, saving many pins on the micro-controller. Another advantage of having less number of pins is that more pins come with more connections which increases the chances of failure of the system in electronic systems.

2.1.4 Antenna

Several antenna designs were available for use with our systems. We specifically wanted an antenna whose operational frequency is 13.56MHz and is minimal in size. We went ahead with a trace antenna. One particular reason to choose a trace antenna was the availability of a well-documented RFID module in the market, which had the matching circuit given. We considered only after consulting our faculty advisor to verify if it was feasible to work with, as RF circuits need some care while working with them. z

2.1.5 RFID Tags

Though almost all RFID Tags are similar, some libraries may not be compatible with certain types of tags. Our reader only works with MIFARE Classic RFID cards. The developer must ensure that the working of a particular type of tag is verified by performing tests, and any assumption might be perilous.

2.1.6 Power Source

We decided to make our system entirely hand-held for ease of use. We wanted two kinds of power sources for our application. One power source of 5V to power our LCD., and another power source of 3.3V is for powering the MFRC IC. The Raspberry Pi Pico could be powered using a 5v power supply. The Raspberry had an output of 3.3V to handle the MFRC IC once powered on. We decided to use AMS1117-5V voltage regulator IC to meet our requirements. The power input to this IC was to come from 9V Conceptual Design Review single-use battery.

2.2 Design

The initial block diagram of our project is given in the figure below.

2.3 Testing

While we were actively working on making our PCB for MFRC522 and the antenna, we could not figure out any way to test it. So our team mainly focused on preliminary testing with the MFRC module. Most of our preliminary testing went into interfacing the LCD and MFRC522 with the Raspberry Pi Pico, and we have described them in the following sections below.

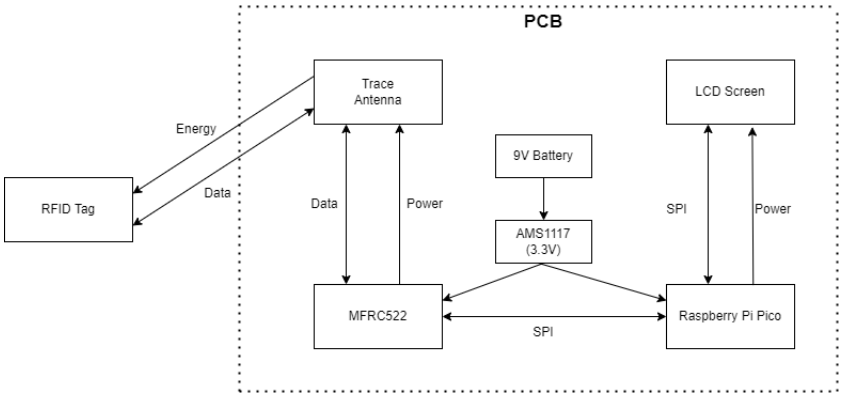


Figure 1: System Flowchart

2.3.1 Software Setup

Using the pico-arduino-compat documentation, we set up the Arduino IDE to program our Raspberry Pi Pico. We downloaded the required libraries for interfacing LCD and micro-controller, MFRC522 module, and micro-controller. We used the Windows operating system to conduct all our testing, as Arduino has better support for Windows. The choice of the operating system becomes crucial in the longer term as specific libraries might run into compatibility issues. Though most modern software is made for different operating systems, some subtle differences and problems can require another turnaround that might be time-consuming. Before discovering the pico-arduino-compat, we tried using C/C++ SDK, which took much time to set up. Even after setting up, there needed to be more support for interfacing different components.

2.3.2 Power Test

Raspberry Pi Pico is powered and programmed through the USB port. However, we wanted our RFID reader to be portable and hand-held; thus, we needed to find a way to power our Raspberry Pi Pico using an onboard power supply. Our main challenge was finding an alternative power supply for the micro-controller.

After referring to the datasheet of the Raspberry Pico Pi, we found that supplying an input voltage in the range of 3.3-5V to the VSYS pin of the micro-controller could power it. Since our MFRC522 IC needed a 3.3V supply, our first guess was to use 3.3V as this supply input.

We built a simple setup using a power supply available in the lab, initially set to 3.3V. We gave this supply voltage to the micro-controller's VSYS pin and the MFRC522 IC. and connected the LCD screen to the micro-controller's I2C VDD pin. We found that the LCD screen did not light up. We realized that the LCD screen requires a 5V input voltage and thus changed the supply voltage to

5V to rectify this issue.

A block diagram of our setup used in shown in the figure below:

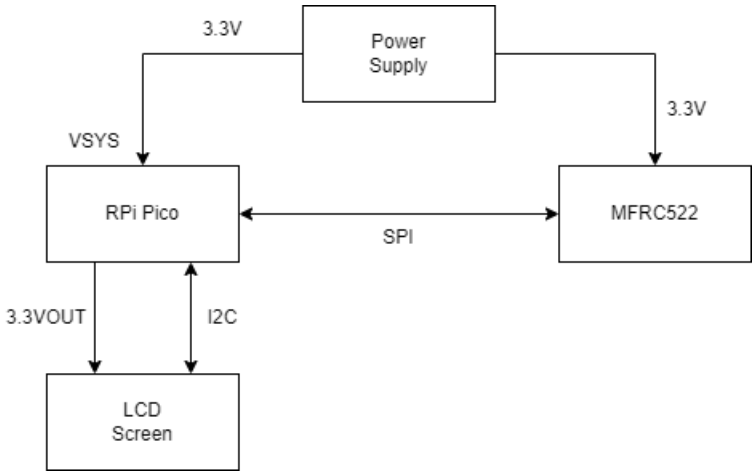


Figure 2: The block diagram of the volatge test setup

2.3.3 Interfacing MFRC522 and Raspberry Pi Pico

Several libraries are present to interface the MFRCC522 module with Arduino. We used the pico-arduino-compat to interface with Raspberry Pi Pico using Arduino IDE. The documentation clearly states how to port the codes in C++ to Raspberry Pi Pico. Even though several libraries are available in embedded C working with C++ is more convenient and thus preferred the same. The MFRC522 is ready to use module with RC522 QFIC along with the antenna and its corresponding matching circuitry. It can perform preliminary interfacing and check the module's functionality. The MFRC522 library provides several functions to detect, read and write to the RFID tags. We could read and write data to the RFID tags for our testing. The module can be interfaced with the micro-controller using the SPI protocol.

2.3.4 Interfacing LCD with Raspberry Pi Pico

We performed the LCD interfacing using a Two Wire Interface (I2C). Several I2C and non-I2C libraries are present for interfacing the LCD with Arduino. We were able to port these codes to Raspberry Pi Pico using pico-arduino-compat after facing some initial problems with channel 1 of I2C on Raspberry Pi Pico. Finally, we use channel 0 to carry out our interfacing.

3 Critical Design Review

3.1 Design Changes

After the first milestone, we carried out extensive testing to determine errors in the initial design, and finding any potential point of failures. The testing method to do the same has been explained in detail in later sections, as well as in our previous reports. In this section, we focus on some of the challenges we faced while testing, and our solutions to the problems faced.

3.1.1 Challenge: Taking User Input

The design for the user interface that was proposed during the PDR was insufficient to meet a fundamental requirement of our project. An RFID tag needed to be programmed with the item details before it can be used to track down inventory items. During our tests we programmed the card using the Serial Monitor to take user inputs. However, we had not considered any such functionality in our design. The current iteration required the access of the personal computer and was inconvenient. We though it was a good idea to add a functionality of alpha-numeric keypad to our system which will give a more interactive usage as compared to the initially proposed two button interface to check items in and out.

3.1.2 Challenge: PCB Errors

We already had made a circuit schematic and a PCB layout keeping the push buttons in mind. These PCBs had also been sent for printing to an external vendor. The original circuit schematic and the PCB layout that we planned to use in our project have been included in the previous reports. After this milestone, we made numerous in a new PCB. We will explain the differences between this new PCB (PCB2.0) and the original PCB (PCB1.0) along with the schematic and layout of PCB2.0.
As mentioned before, PCB1.0 had a lot of point of failures, along with adding mechanical constraints on how we packaged our project. In this section, we focus only on errors in PCB1.0, and the solutions implemented in PCB2.0. In the end, we also give the circuit schematic and the layout of PCB2.0.

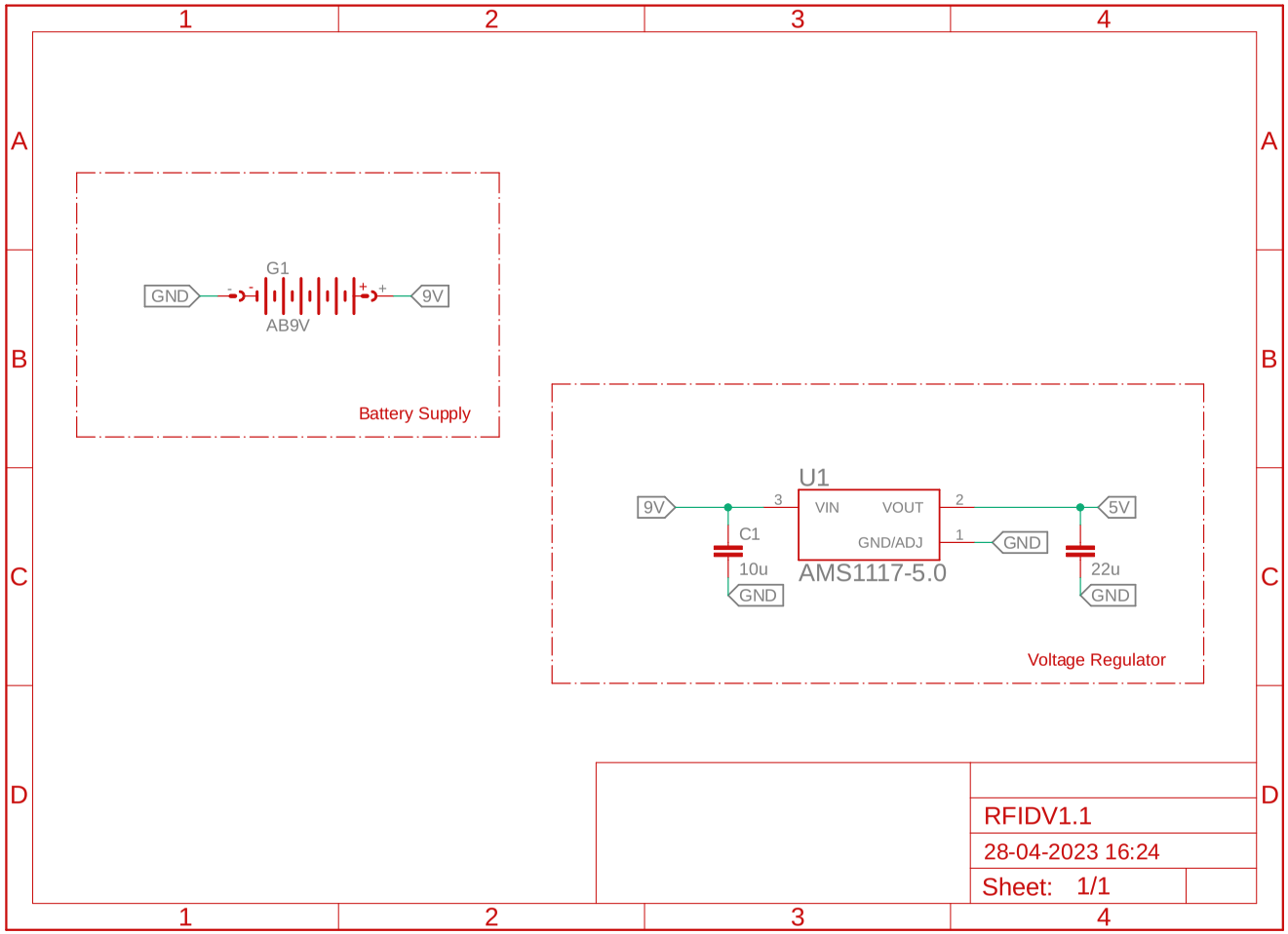
The first issue was the floating pin of the MFRC522 IC. Pin number 15 of the IC was left floating in PCB1.0, whereas its actual value should be kept at 3.3V. This issue was detected when we soldered the MFRC522 IC on PCB1.0, and tested the code given for checking the firmware. The firmware check never passed on PCB1.0, but the same IC passed the test on the module as well as the breakout board. This indicated an error on our PCB, and prompted us to check the circuit schematic of PCB1.0. The floating pin was found when we compared the schematic to the reference circuit in the datasheet. This error has been corrected in PCB2.0.

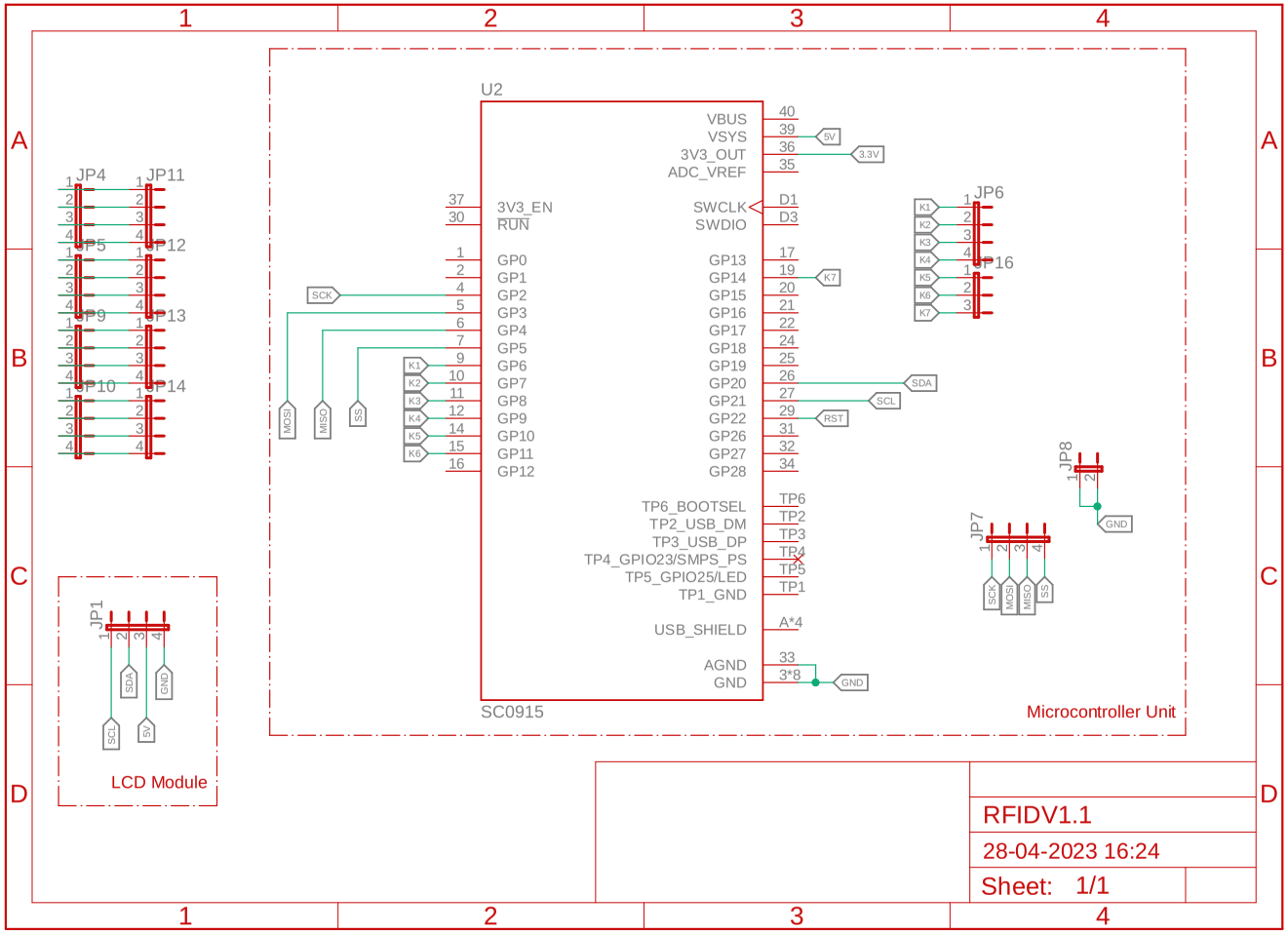
As mentioned earlier, PCB1.0 introduced multiple mechanical constraints on our design. One such constraint was caused by the incorrect placing of the Raspberry Pi Pico in PCB1.0. It was placed in the centre, so we could not program it as its USB port was not easily accessible. This was a major issue, as we needed to update any changes to the micro-controller on PCB1.0. We tried to convert the SMD package pads into a through-hole by drilling holes in PCB1.0, however this impacted the SPI traces that were routed in the bottom layer. In PCB2.0, we have placed the micro-controller keeping this constraint in mind, as well as sufficient spacing between connections for the LCD screen and the keypad.

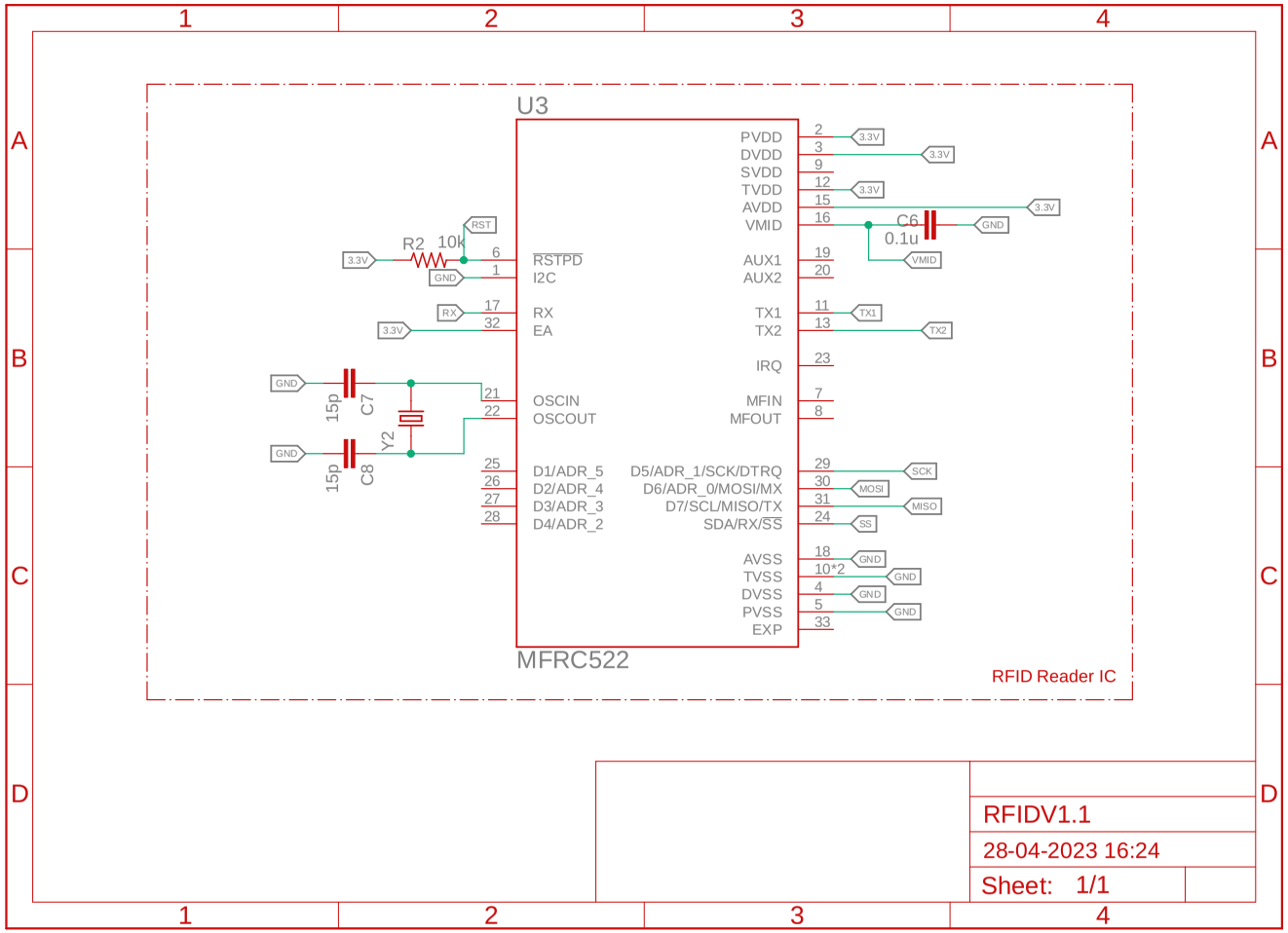
PCB1.0 used an AMS1117-5V IC as part of the power subsystem of our project. However, we had observed that for a few times when this IC was soldered on PCB1.0, it would overheat after only a few minutes of operation. This significantly impacted the entire project, as after overheating, AMS1117-5V would stop producing a constant supply of 5V, and would instead generate random voltages. This problem led to us using a wired USB connection to the micro-controller to power the entire project. The RFID reader was no longer hand-held. In PCB2.0, we still have connections for AMS1117-5V as a redundancy measure.

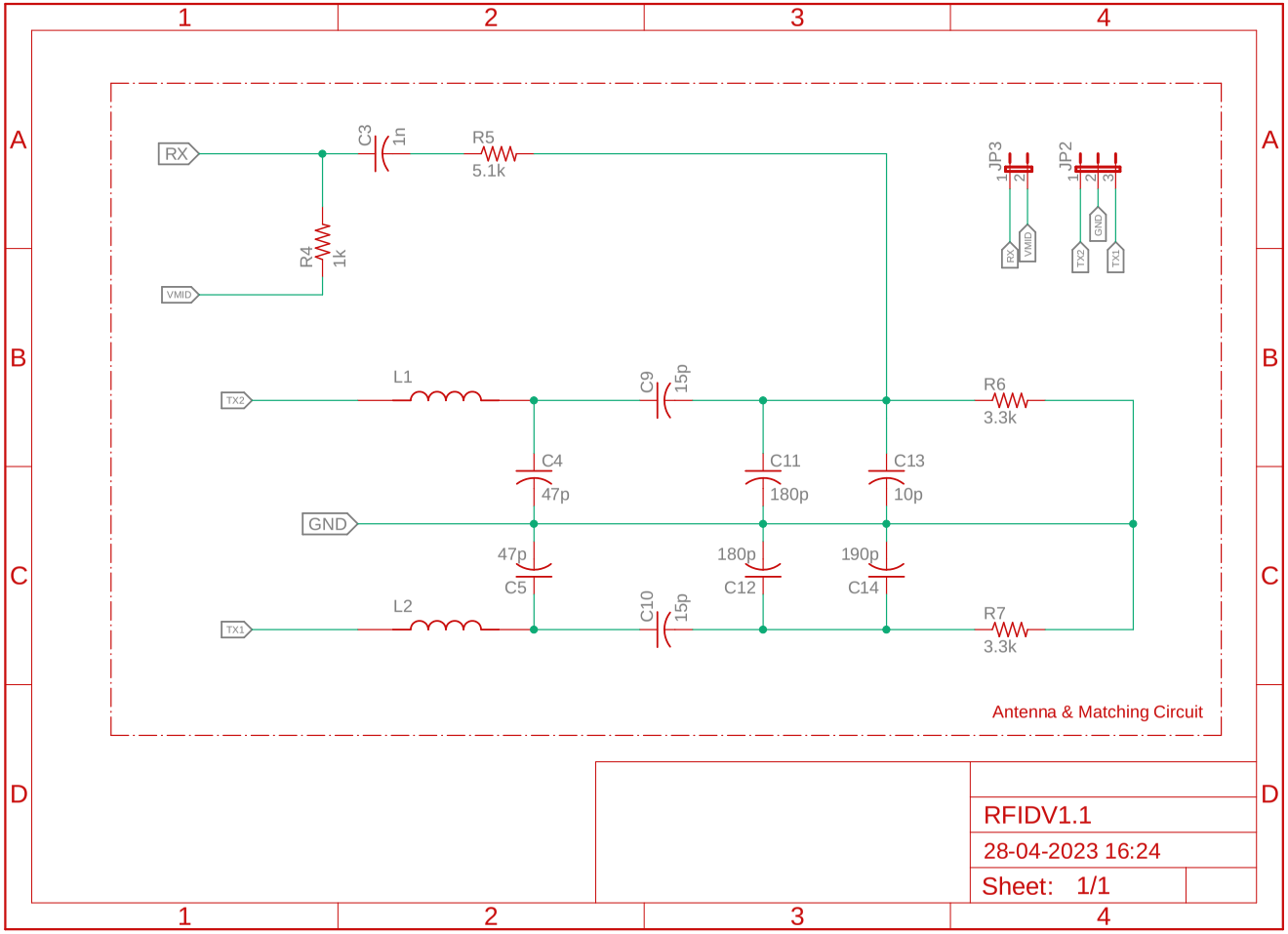
In PCB1.0, there were also incorrect corrections in the matching circuit of the trace antenna. This was detected by complete testing of the antenna along with the matching circuit and MFRC522. This error has been rectified in PCB2.0. We have also removed the 0 ohm resistors added in PCB1.0, as they interfered with the SPI lines and the matching circuit.

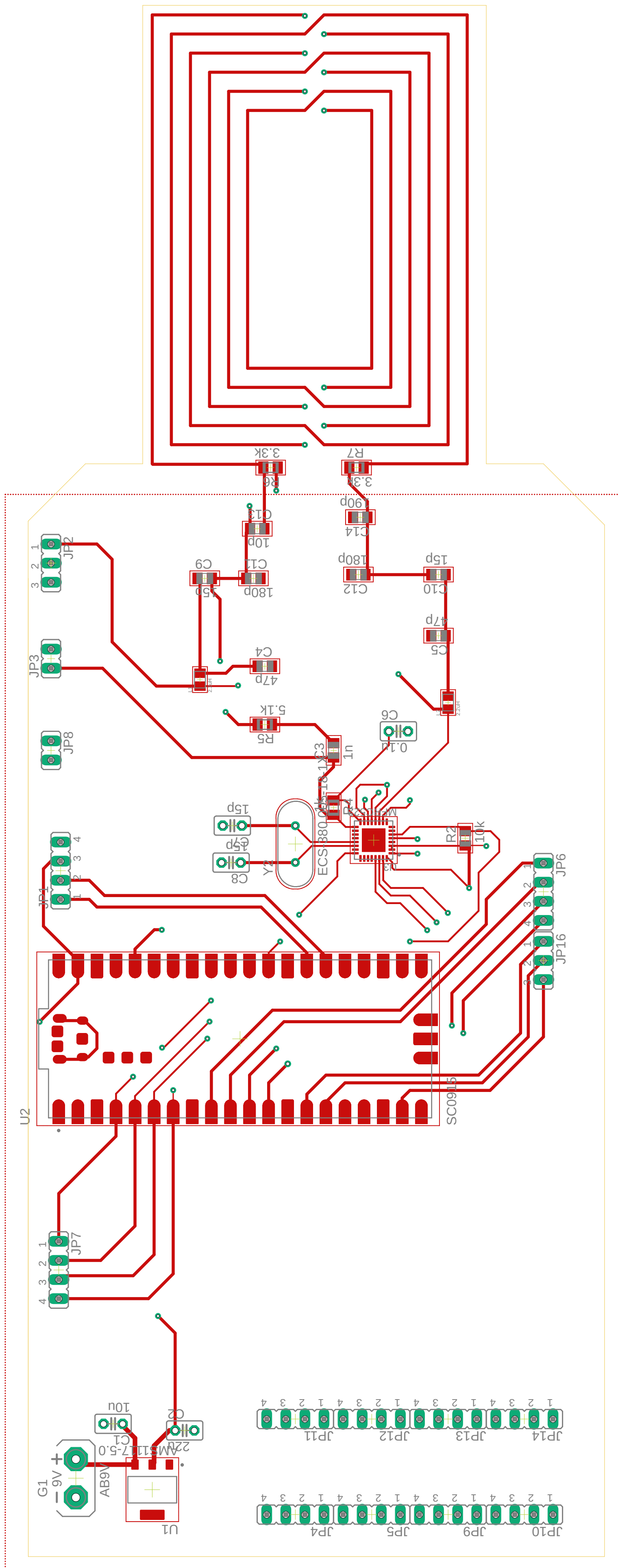
PCB2.0 could not be sent out to the vendor, as they could not promise to deliver the boards on time. So, our final implementation (which will be discussed in the next section) was completed on perforated boards, where we soldered the traces and components by hand. In the following few pages, we have attached the circuit schematic and the PCB layout of PCB2.0.

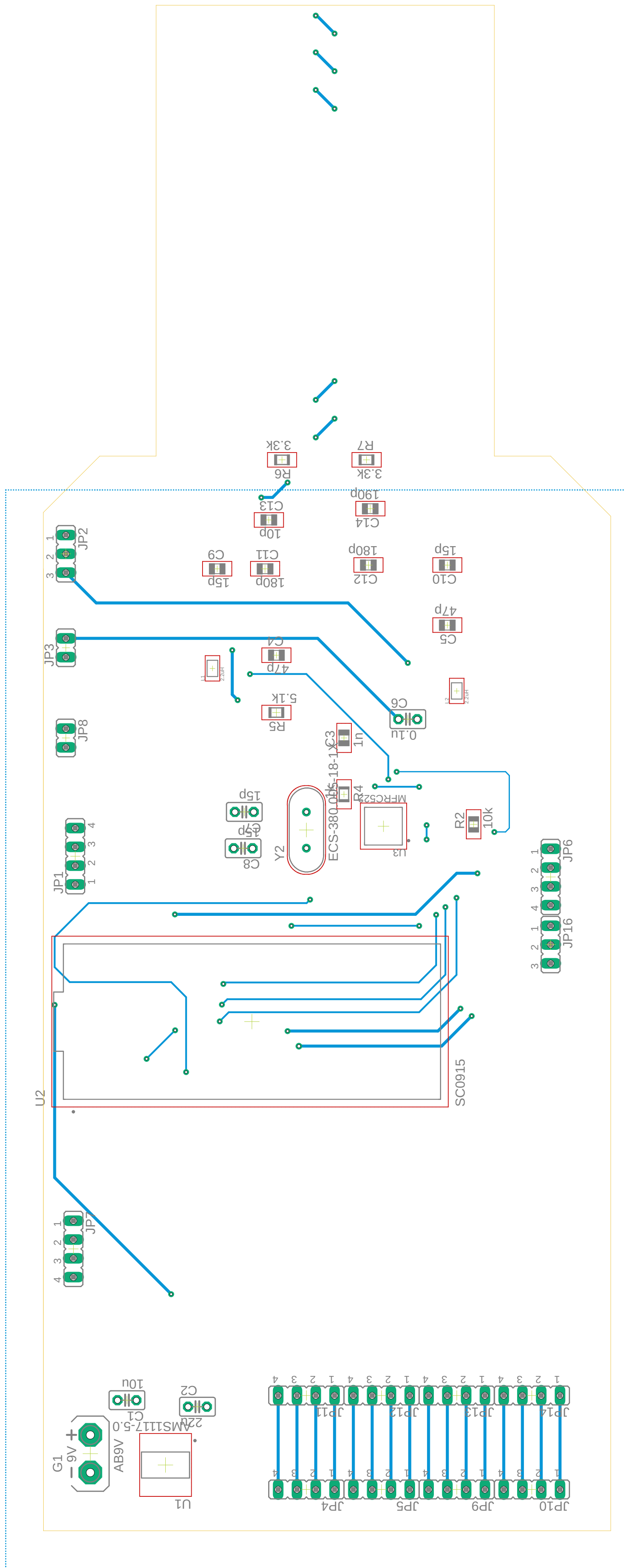

1
2
3
4
A
B
C
D
G1
GND
9V
AB9V
Battery Supply
U1
9V
3
VIN
VOUT
2
5V
C1
GND/ADJ
1
GND
10u
AMS1117-5.0
22u
GND
GND
Voltage Regulator
RFIDV1.1
28-04-2023 16:24
Sheet: 1/1


U2
SC0915
Microcontroller Unit
LCD Module
JP1
JP4
JP5
JP9
JP10
JP11
JP12
JP13
JP14
JP6
JP16
JP7
JP8
VBUS
VSYS
3V3_OUT
ADC_VREF
3V3_EN
RUN
SWCLK
SWDIO
GP0
GP1
GP2
GP3
GP4
GP5
GP6
GP7
GP8
GP9
GP10
GP11
GP12
GP13
GP14
GP15
GP16
GP17
GP18
GP19
GP20
GP21
GP22
GP26
GP27
GP28
TP6_BOOTSEL
TP2_USB_DM
TP3_USB_DP
TP4_GPIO23/SMPS_PS
TP5_GPIO25/LED
TP1_GND
USB_SHIELD
AGND
GND
5V
3.3V
SCK
MOSI
MISO
SS
SDA
SCL
RST
K1
K2
K3
K4
K5
K6
K7
RFIDV1.1
28-04-2023 16:24
Sheet: 1/1


U3
MFRC522
RFID Reader IC
PVDD
DVDD
SVDD
TVDD
AVDD
VMID
RSTPD
I2C
AUX1
AUX2
RX
EA
TX1
TX2
IRQ
OSCIN
OSCOUT
MFIN
MFOUT
D1/ADR_5
D2/ADR_4
D3/ADR_3
D4/ADR_2
D5/ADR_1/SCK/DTRQ
D6/ADR_0/MOSI/MX
D7/SCL/MISO/TX
SDA/RX/SS
AVSS
TVSS
DVSS
PVSS
EXP
R2 10k
C6 0.1u
C7 15p
C8 15p
Y2
3.3V
GND
RST
RX
VMID
TX1
TX2
SCK
MOSI
MISO
SS
RFIDV1.1
28-04-2023 16:24
Sheet: 1/1


1
2
3
4
A
B
C
D
RX
C3
1n
R5
5.1k
R4
1k
VMID
JP3
JP2
RX
VMID
GND
TX2
TX1
L1
C9
15p
R6
3.3k
C4
47p
C11
180p
C13
10p
GND
47p
C5
180p
C12
190p
C14
L2
C10
15p
R7
3.3k
TX2
TX1
Antenna & Matching Circuit
RFIDV1.1
28-04-2023 16:24
Sheet: 1/1


JP2
JP3
JP8
JP1
JP6
JP16
JP7
U2
SC0915
R6
R7
3.3k
C13
10p
C9
C11
180p
C14
C12
C10
15p
C5
47p
C4
R5
5.1k
C3
1n
C6
0.1u
R2
10k
Y2
C7
C8
C1
10u
C2
22u
AMS1117-5.0
U1
G1
9V
AB9V
JP4
JP5
JP9
JP10
JP11
JP12
JP13
JP14



R6
3.3k
R7
3.3k
C13
10p
C14
190p
C9
15p
C11
180p
C12
180p
C10
15p
C5
47p
C4
47p
R5
5.1K
C6
0.1u
C3
1n
R4
MFRC522
U3
R2
10k
Y2
ECS-380
C7
15p
C8
15p
JP2
JP3
JP8
JP1
JP6
JP16
U2
SC0915
JP7
G1
9V
AB9V
U1
AMS1117-5.0
C1
10u
C2
22u
JP11
JP12
JP13
JP14
JP4
JP5
JP9
JP10

3.2 Testing

3.2.1 Microcontroller Interfacing

Throughout the project, we used SPI communication for interfacing the MFRC522 with the Raspberry Pi Pico micro-controller. Along with this, the LCD screen communicates with the micro-controller through I2C. We followed the following testing strategies in our project.

- First, we tested the SPI communication between the micro-controller and the MFRC522 IC. The code was used to simply transfer the data from the RFID tag read by the MFRC522 IC and initially display the data on a centralized computer such as a laptop. We used the Serial Monitor that comes along with Arduino IDE to test the SPI.
- Second, we tried to establish the I2C communication of the micro-controller with the LCD screen.
- Lastly, We planned to display the data we obtained from the MFRC522 IC on the LCD screen in order to test if both were working correctly together.

The results for the following two tests will be visible in the following subsection, where we have talked about the code functionality and user interface in detail. We successfully completed the interfacing the MFRC522 module and LCD with the Raspberry Pi Pico on the breadboard. We specifically worked on incorporating three major functionality for carrying out inventory management by our system.
We wrote microcontroller code implement following functions on board.

1. addCard()
2. readCard()
3. increment()
4. decrement()

addCard()
This function is used to program a new MIFARE RFID card for an item in the inventory. For current iteration, the programming can be done using the Arduino Integrated Development Environment using the Serial Monitor only. The user is prompted for the name of the item in the inventory and the number of items in the inventory. Once the card is programmed the it can used to add and remove item from the inventory.

readCard()
This function is used to read the information on the card, specifically, the name of the item for which the card is assigned and the corresponding number of units

according to the last update.

increment()
This function is used to add the item corresponding to the card in the inventory. This function is used along with the readCard() function so that the user is able to see the update in the count.

decrement()
This function is used to remove the item corresponding to the card in the inventory. This function is used along with the readCard() function so that the user is able to see the update in the count.

3.2.2 User Interface

Following are the snapshots form the User interface for different modes at various stages.

Programming a card

A new card can be programmed for new item as follows:

1. Select the ADD CARD mode. A message "Scan Your Card" is visible on the LCD.
2. Keep the card in the close vicinity of reader. Do not remove the card anytime during the procedure.
3. Enter the name of the item in the Serial Monitor once prompted and press Enter.
4. Enter the count of the item in the inventory in the Serial Monitor once prompted and press enter.
5. Wait till you see the card information just added displayed on the LCD screen.
6. Remove the card for future use.

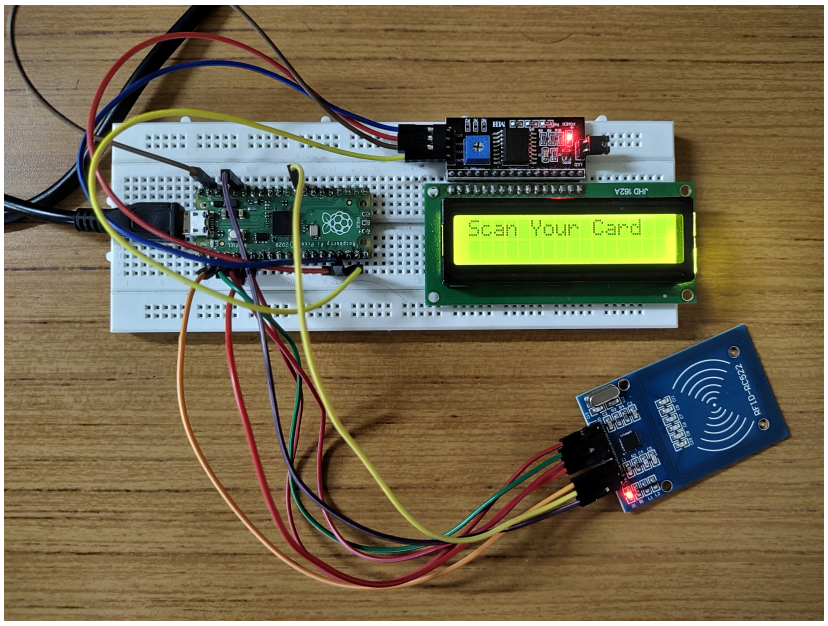


Figure 3: Scanning the RFID card

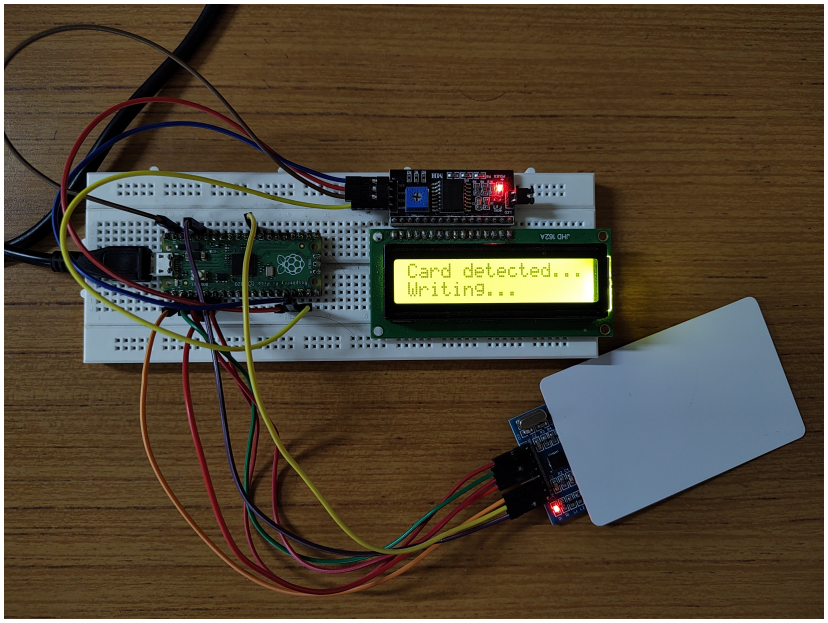


Figure 4: Card detected and writing

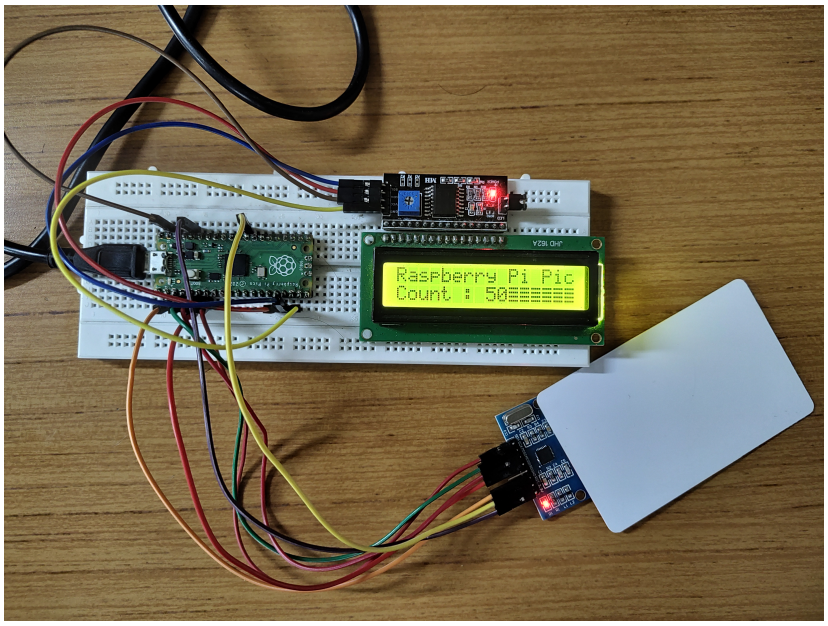


Figure 5: Item added successfully

Reading the current status of the card

An existing card can be read to know the status of the item in the inventory as follows:

1. Select the READ CARD mode. A message "Scan Your Card" is visible on the LCD.
2. Keep the card in the close vicinity of reader. Do not remove the card anytime during the procedure.
3. Wait till you see the item information on the LCD screen.
4. Remove the card for future use.

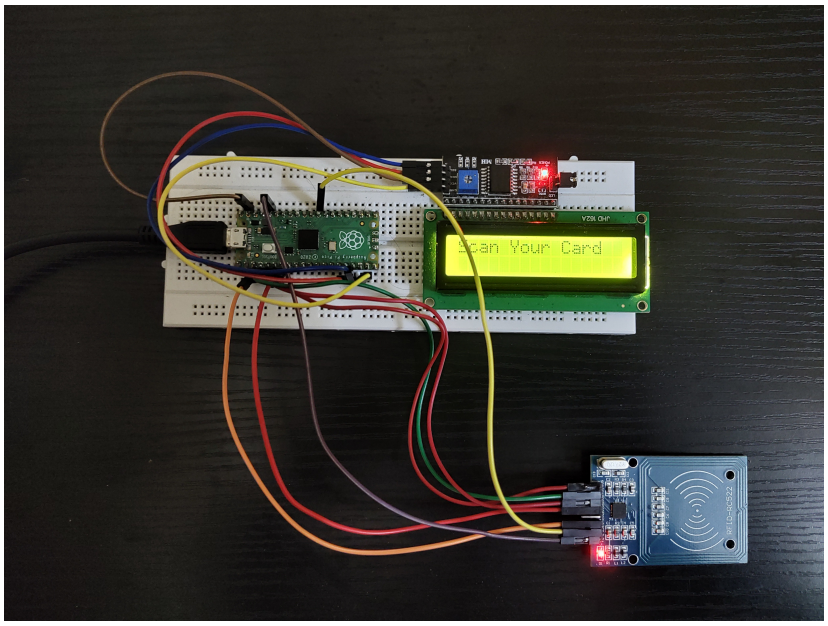


Figure 6: Scan the RFID card

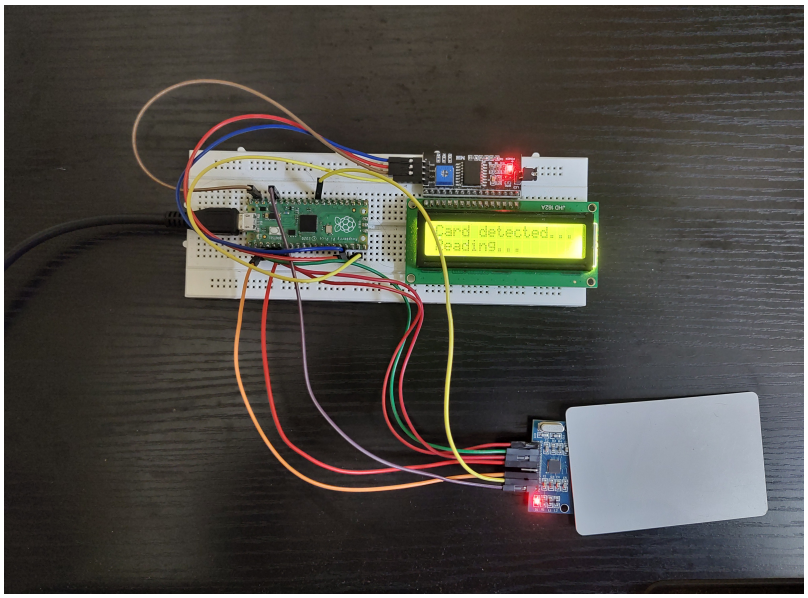


Figure 7: Card detected and reading

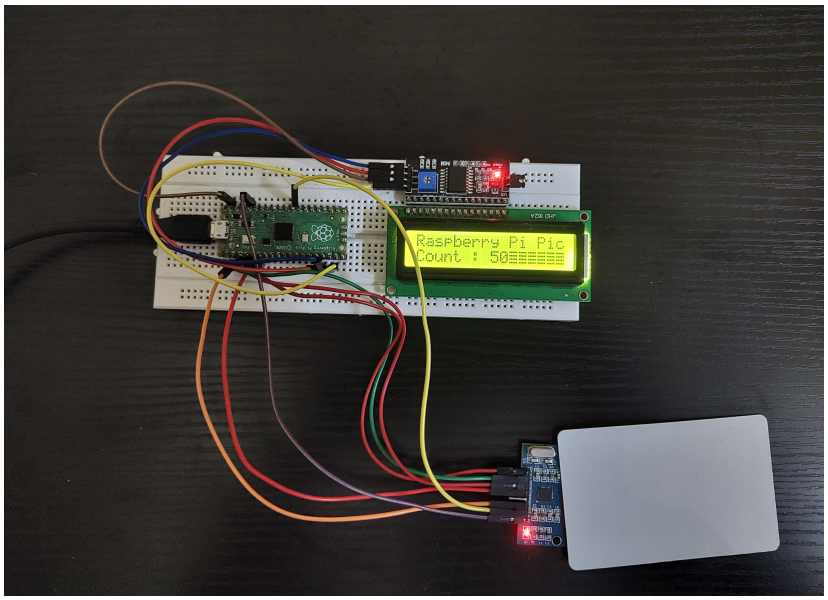


Figure 8: Item details

Adding and Removing items to and from the inventory

An item can be removed from the inventory or added to the inventory as follows:

1. Select between ADD ITEM or REMOVE ITEM mode by entering '1' or '2' respectively on the Serial Monitor and hit Enter. A message "Scan Your Card" is visible on the LCD.
2. Keep the card in the close vicinity of reader. Do not remove the card anytime during the procedure.
3. Wait while the information is being updated.
4. Remove the card once you can see the updated information on the LCD Screen.

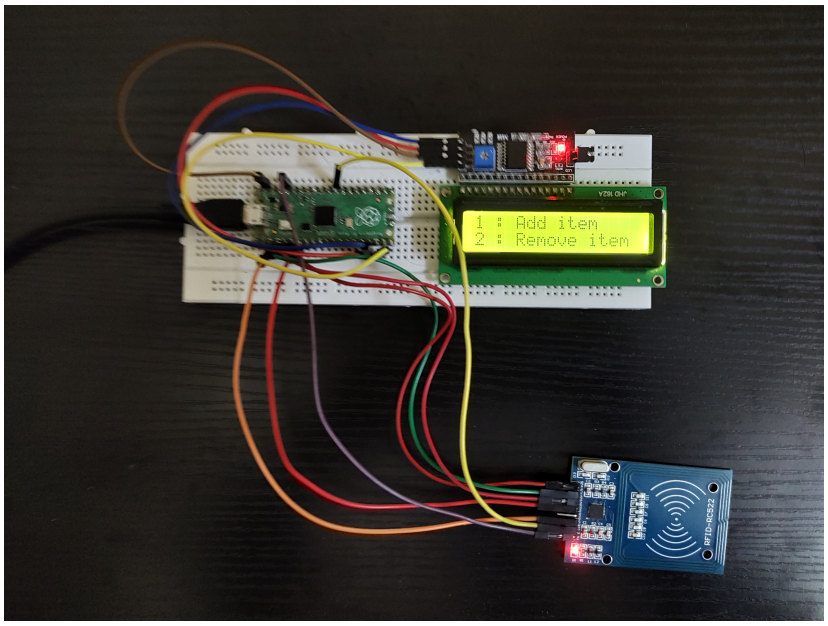


Figure 9: Add or remove item

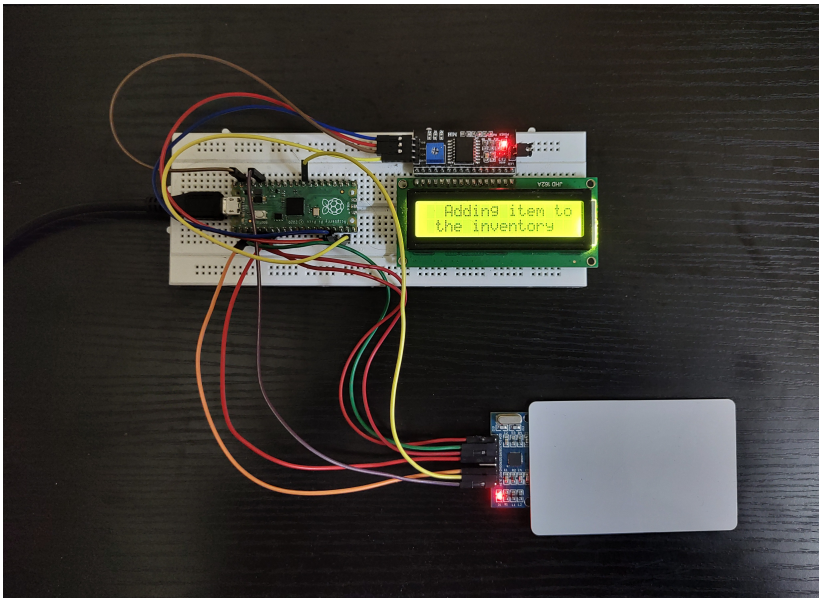


Figure 10: Adding item to the inventory

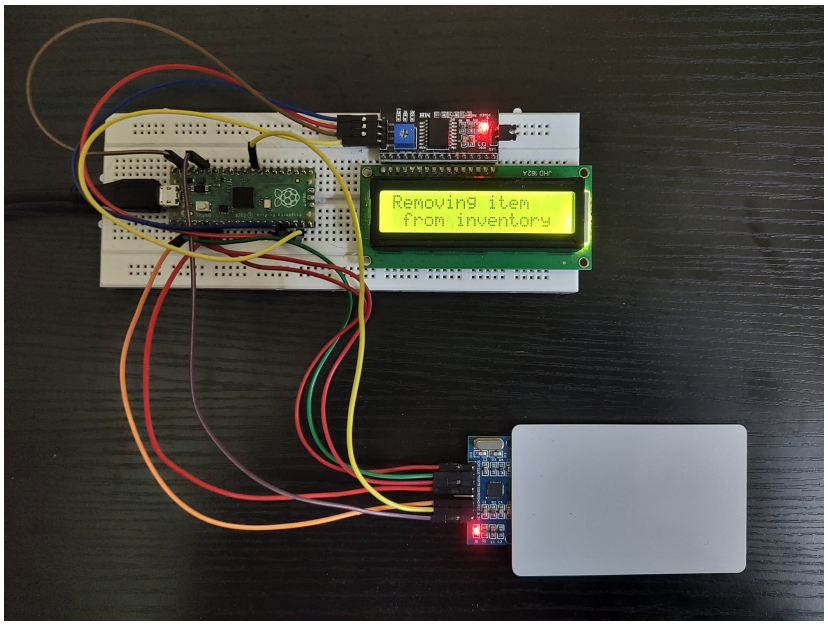


Figure 11: Removing item from inventory

In order to carry out preliminary testing of the switch functionality we have tried to interface 4×3 keypad with our Raspberry Pi Pico. However, due to insufficient documentation we were not able to figure out the functionality.

4 Final Implementation and Results

Since our implementation has shifted from a single PCB to a perforated board structure, we changed the CAD of our packaging. Since there was a lot of circuitry to be replicated on the perforated boards, we chose not to implement the entire project on a single board. Instead, we chose to develop a three-layer structure, with each subsystem on a single layer.

The alpha-numeric keypad was successfully interfaced with the micro-controller making our system completely independent of the personal computer.

Since we decided to power the project using the wired USB connection, there was no need for a separate power subsystem. The remaining subsystems from the initial block diagram were arranged in three layers. The antenna and its matching circuit were present at the bottom-most layer. In the middle layer, we connected the micro-controller. The top layer consisted of the LCD screen along with the keypad.

The choice of components on each layer was made to ensure that connections are easily made. The micro-controller connected to both the matching circuit as well as the LCD and keypad, so it was kept in the middle layer. The LCD screen and keypad had to be visible to the user, so it was kept on the top layer. The antenna was present on the bottom layer, as a result we designed a slot for the RFID card to be inserted into the package, so that it could be read by the antenna and the MFRC522 IC in the bottom layer.

The CAD has been designed as a simple box from the outside. However, inside the box we have designed small steps at appropriate heights in order to support each layer of the perforated boards. Such a design allows us to easily remove the layers, if needed, and place them back. It requires the bottom-most layer to have the least width, and the the top layer to have the most width. The box was 3D printed at WEL, and the cover of the box was laser cut.

The final implementation block diagram and the final CAD picture is shown below. The CAD files have been attached in the same folder as this report. The CAD was developed using Solidworks.

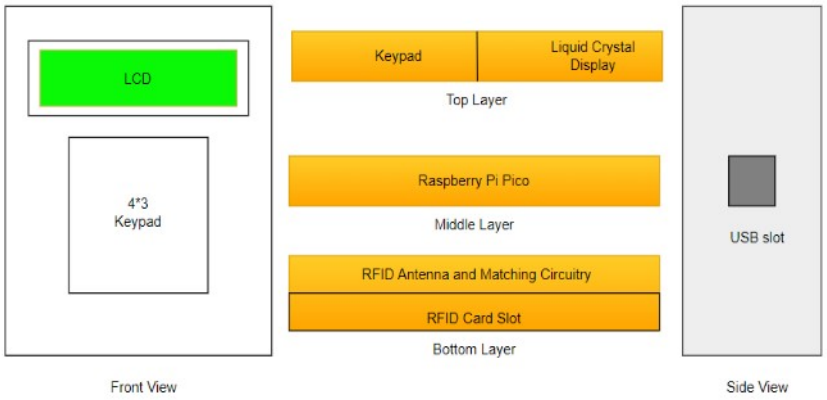


Figure 12: Block diagram of final implementation

Figure 13: CAD of the final implementation

4.1 Testing on Perforated Boards

Once we developed the three layer structure, it was important for us to test the individual components on each layer.

4.1.1 Micro-controller Layer

The micro-controller layer required the least effort in terms of any new testing while shifting to the perforated board structure. We added header pins next to the micro-controller pins to make them more accessible for any connections. All the connections to the micro-controller to the LCD screen and keypad were tested while these components were still on the breadboard, and any constraints were noted and considered while making the CAD.

4.1.2 Keypad and LCD Layer

Once we soldered the Keypad and LCD onto the perforated board, we checked our soldering for any shorts. The first time we did not detect any shorting between the connections. However, when we connected the keypad and the LCD screen to the micro-controller, we observed that the LCD screen did not switch on. On further testing, we detected a minute soldering error which led to two pins of the connection being shorted to each other. We then extensively tested the remaining connections and then proceeded with the connections.

4.1.3 Antenna and Matching Circuit Layer

To test the antenna and the matching circuit along with MFRC522 IC, first we needed to complete all connections. Then, there were two tests that confirmed whether the setup worked. First was the firmware check code (mentioned above). This was a default code given in the MFRC522 library that checked the firmware of the IC along with the connections and returned whether a defect was detected.

The second code was a more extensive test for the MFRC522 IC. A particular register of the IC stores the value at 128 as a default. In order to enable the IC to drive the antenna, the value in this register needs to be changes to the value 131. We wrote a SPI code to write to this register and change its value to 131, while displaying the value of the register as an output. We were able to successfully determine if the IC was faulty or not based on running the code and checking the output.

In the next section, we give the Bill of Materials (BOM) used in the project (as generated by the PCB software), along with any added components used.

5 Problems and Future Work

After presenting our project we got a descriptive feedback from the judges about our system. Some problems that were found in our system are stated as follows.

1. Though the design was wonderful the judges pointed out that the system design was not as expected. They suggested we could have gone with a design like a bar code scanner with faster check-in and check-out functionality.

2. The system was not secure it was pointed out we come up with some sort of strategy to make it secure.

Based on the feedback we tried to incorporate these design changes to some extent. Security feature was added for the system. The user interface was divided into two parts, namely Register Mode and Login Mode. Register Mode allowed new items to be assigned tags while providing them with a default password. On the other hand the Login Mode had functionality to add and remove item from the inventory, read tag, and change password.
A number of add-ons can be done on this system that increase its reliability and speed of tracking items in the inventory. The current system does not have a inventory logging mechanism. In cases if the user loses the card all the important data would be lost. The system does not provide a fast tracking of inventory movement and the user interface can be simplified further. There problems can be handled by incorporating SD card in the system which will allow more flexibility of storing data and thus improve the tracking.

5.1 Bill of Materials (BOM) & Components Used

Name	Description	Quantity	Price (Rs.)	Product Link (if any)
MFRC522 Module	RFID Module (13.56 MHz)	3	83	https://www.electronicscomp.com
Keypad	Numeric keypad	1	WEL	WEL
LCD Screen	LCD Screen	1	WEL	WEL
LCD I2C Module	To interface LCD	1	WEL	WEL
PCB1.0	First iteration of PCB	5	590	EPS PCB
SC0915	Raspberry Pi Pico	1	WEL	WEL
AMS1117-5.0	5V regulator	15	11.80	https://www.robu.in
9V Battery	9V power supply	1	22	https://robocraze.com
Battery Connector	Connects battery to PCB	1	10	https://www.electronicscomp.com
Crystal	27.12 MHz Crystal	1	35	https://www.robu.in

The components that we have used in this project (generated by EAGLE) are:

Partlist exported from C:/Users/kaust/OneDrive/Documents/EAGLE/projects/RFID/RFIDV1.1.sch at 28-04-2023 19:08

Part	Value	Device	Package	Description
C1	10u	C-EU025-025X050	C025-025X050	CAPACITOR, European symbol
C2	22u	C-EU025-025X050	C025-025X050	CAPACITOR, European symbol
C3	1n	C-USC0805	C0805	CAPACITOR, American symbol
C4	47p	C-USC0805	C0805	CAPACITOR, American symbol
C5	47p	C-USC0805	C0805	CAPACITOR, American symbol
C6	0.1u	C-EU025-025X050	C025-025X050	CAPACITOR, European symbol
C7	15p	C-EU025-025X050	C025-025X050	CAPACITOR, European symbol
C8	15p	C-EU025-025X050	C025-025X050	CAPACITOR, European symbol
C9	15p	C-USC0805	C0805	CAPACITOR, American symbol
C10	15p	C-USC0805	C0805	CAPACITOR, American symbol
C11	180p	C-USC0805	C0805	CAPACITOR, American symbol
C12	180p	C-USC0805	C0805	CAPACITOR, American symbol
C13	10p	C-USC0805	C0805	CAPACITOR, American symbol
C14	190p	C-USC0805	C0805	CAPACITOR, American symbol
G1	AB9V	AB9V	AB9V	9-V BATTERY CLIP
JP1		JP4E	JP4	JUMPER
JP2		JP2E	JP2	JUMPER
JP3		JP1E	JP1	JUMPER
JP4		JP4E	JP4	JUMPER
JP5		JP4E	JP4	JUMPER
JP6		JP4E	JP4	JUMPER
JP7		JP4E	JP4	JUMPER
JP8		JP1E	JP1	JUMPER
JP9		JP4E	JP4	JUMPER
JP10		JP4E	JP4	JUMPER
JP11		JP4E	JP4	JUMPER
JP12		JP4E	JP4	JUMPER
JP13		JP4E	JP4	JUMPER
JP14		JP4E	JP4	JUMPER
JP16		JP2E	JP2	JUMPER
L1	2.2uH	AIMC-0805-39NJ-T	INDC2012X105N	IND 39nH 0.3A 650mΩ Check availability
L2	2.2uH	AIMC-0805-39NJ-T	INDC2012X105N	IND 39nH 0.3A 650mΩ Check availability
R2	10k	R-US_R0805	R0805	RESISTOR, American symbol
R4	1k	R-US_M0805	M0805	RESISTOR, American symbol
R5	5.1k	R-US_M0805	M0805	RESISTOR, American symbol
R6	3.3k	R-US_M0805	M0805	RESISTOR, American symbol
R7	3.3k	R-US_M0805	M0805	RESISTOR, American symbol
U1	AMS1117-5.0	AMS1117-5.0	SOT229P700X180-4N	Check availability
U2	SC0915	SC0915	MODULE_SC0915	Check availability
U3	MFRC522	MFRC522	QFN50P500X500X100-33N	Check availability
Y2	ECS-380.005-18-1X	ECS-380.005-18-1X	XTAL_ECS-200-20-1X	Crystal HC49/U T/H Check availability

6 Demonstration Video

The final video of the current state of the system with explanation can be found here

7 Resources

Raspberry Pi Pico Datasheet: here
Arduino-Pico-compat: here