Department of Electronic & Telecommunication Engineering
University of Moratuwa, Sri Lanka.

Design Document:
RFID based asset tracking device for forklift

210610H Sirimanna N.T.W.

Submitted in partial fulfillment of the requirements for the module
EN 2160 Engineering Design Realization

06/07/2024

Overview

The design document serves as a comprehensive blueprint for the development of our RFID-based asset tracking system designed specifically for forklift operations. It outlines the market analysis, design specifications, system architecture, and implementation plan for the project. This document provides a detailed overview of the product, its functionalities, and the technical aspects of its design, including the RFID reader module, antenna design, power management, and data communication. It also includes testing and validation strategies, implementation timelines, and budget considerations. This document is intended to guide the development team throughout the project lifecycle, ensuring a systematic and efficient approach to delivering a high-quality asset tracking solution for forklifts.

Contents

1 General (10/02/2024)

1.1 Marketing Needs

The proposed RFID-based product for tracking assets loaded onto forklifts addresses a critical need in Sri Lanka's market. With the absence of major companies offering similar solutions, there is a clear opportunity to introduce a cost-effective product that enhances warehouse management efficiency. This solution will enable warehouse managers to easily locate assets within the warehouse, improving overall operational efficiency and asset tracking accuracy.

1.1.1 Existing Product Analysis

- **Link -** A Case Study of Smart Forklift Integration with RFID
- **Link -** An RFID data-based forklift traffic management system by Research gate

Solutions from industry leaders in forklift-mounted RFID technology, such as Honeywell and Impinj, were analyzed through product documentation, technical specifications, and online resources. Key features such as compatibility with forklift mounting, ruggedness for industrial environments, and integration capabilities with existing warehouse management systems were highlighted. Importing these products into Sri Lanka would incur significant costs due to shipping, taxes, and import duties, adding to the overall expense of implementing such solutions.

1.1.2 User Profile

The primary users of this device are warehouse managers, logistics supervisors, and inventory control personnel. These individuals are responsible for overseeing the efficient movement, storage, and tracking of assets within a warehouse or distribution center.

1.2 Market Segment Analysis

The market segment for the product is the logistics and warehouse management sector, which includes businesses seeking to optimize asset tracking and inventory management processes.

1.3 Final Specifications

The final specifications for the device outline the key requirements and features of the device.

1. Functionality: The device should be able to track assets loaded onto forklifts within a warehouse environment.
2. Technology: The device should utilize RFID technology for asset tracking.
3. Range: The device should have a sufficient range to track assets within a warehouse or distribution center.
4. Accuracy: The device should provide accurate asset tracking information to improve operational efficiency.
5. Integration: The device should integrate with existing warehouse management systems for seamless asset tracking and management.
6. User Interface: The device should have a user-friendly interface for warehouse managers, logistics supervisors, and inventory control personnel.
7. Durability: The device should be rugged and able to withstand the demands of an industrial environment.
8. Cost-Effectiveness: The device should be cost-effective to implement, considering the potential costs of importing similar solutions.

2 RFID Reader Design

2.1 RFID chip Selection (24/02/2024)

The selection of the RFID IC (Integrated Circuit) is a crucial aspect of the forklift asset tracking device project. RFID ICs play a vital role in enabling the device to accurately and efficiently track assets within a warehouse or distribution center environment. The choice of RFID IC impacts the device's performance, compatibility with existing systems, and overall cost-effectiveness. Therefore, careful consideration must be given to selecting the RFID IC that best meets the project's requirements and objectives.

First, I selected three alternatives as the IC.

- NXP MFRC522
- NXP PN532
- Texas TRF7970A

2.1.1 NXP MFRC522

The MFRC522 chip is a highly integrated contactless reader IC (Integrated Circuit) for 13.56 MHz RFID (Radio Frequency Identification) communication. It is commonly used in projects requiring RFID technology, such as access control systems, payment systems, and asset tracking devices. The MFRC522 chip supports ISO/IEC 14443 Type A and Type B protocols and MIFARE Classic RFID tags. It provides a simple and effective way to add RFID functionality to projects, offering features like anti-collision and 7-byte unique identifier (UID) reading. The MFRC522 chip's versatility, ease of use, and robust performance make it a popular choice for RFID applications.

1. Frequency Compatibility: The MFRC522 operates at 13.56 MHz, which is a standard frequency for RFID applications, including asset tracking. This ensures that it can communicate effectively with RFID tags used for asset identification.
2. Protocol Support: The MFRC522 chip supports ISO/IEC 14443 Type A and Type B protocols, which are widely used in RFID applications. This allows it to communicate with a variety of RFID tags, including MIFARE Classic tags, which are commonly used for asset tracking.
3. Versatility: The MFRC522 chip is versatile and can be easily integrated into my project. It provides features such as anti-collision and 7-byte UID reading, which are essential for reliable and efficient asset tracking.
4. Availability and Cost: The MFRC522 chip is widely available and cost-effective.

Figure 1: NXP MFRC522 IC

2.1.2 NXP PN532

The PN532 chip is a highly integrated NFC (Near Field Communication) controller that supports various modes of NFC communication, including ISO/IEC 14443 Type A and Type B, FeliCa, and NFCIP-1 (ISO/IEC 18092) standards. It is commonly used in applications such as access control, payment systems, and interactive devices.

The PN532 chip offers a range of features, including support for peer-to-peer communication, card emulation mode, and reader/writer mode. It operates at 13.56 MHz, which is a standard NFC operating frequency, and provides a flexible and easy-to-use interface for integrating NFC functionality into projects.

1. Frequency Compatibility:
    - PN532: The PN532 chip operates at 13.56 MHz, which is a standard frequency for NFC communication.
2. Protocol Support:
    - PN532: Supports ISO/IEC 14443 Type A and Type B, FeliCa, and NFCIP-1 standards, providing a wide range of compatibility.
3. Versatility:
    - PN532: The PN532 chip offers a high level of versatility, supporting peer-to-peer communication, card emulation mode, and reader/writer mode.
    - MFRC522: The MFRC522 chip is less versatile compared to the PN532, mainly focusing on RFID applications and lacking support for peer-to-peer communication and card emulation mode.
4. Availability and Cost:
    - The PN532 chip is widely available and cost-effective, making it a practical choice for projects with budget constraints. However, it is more expensive than the MFRC522.

Figure 2: NXP PN532 IC

- **Link -**PN532/C1 datasheet
- **Link -**PN532 and MFRC522 Antenna Design Guideline

2.1.3 Texas TRF7970A

The Texas TRF7970A is a highly integrated RFID reader/writer IC designed for 13.56-MHz RFID systems. It supports various RFID standards, including ISO/IEC 14443A/B, ISO/IEC 15693, ISO/IEC 18000-3, and NFCIP-1, making it suitable for a wide range of RFID applications. The TRF7970A offers advanced features such as simultaneous protocol support, automatic modulation index adjustment, and selectable output power levels, providing flexibility and reliability in RFID communication. Its versatility and performance make it an excellent choice for implementing RFID functionality in my asset tracking device project.

1. **Frequency Compatibility:**
   - **TRF7970A:** The TRF7970A operates at 13.56 MHz, which is compatible with NFC and RFID standards.
   - **PN532:** The PN532 chip also operates at 13.56 MHz, making it compatible with the same NFC and RFID frequency.
2. **Protocol Support:**
   - **TRF7970A:** Supports a variety of RFID standards, including ISO/IEC 14443A/B, ISO/IEC 15693, ISO/IEC 18000-3, and NFCIP-1.
   - **PN532:** Supports ISO/IEC 14443 Type A and Type B, FeliCa, and NFCIP-1 standards.
3. **Versatility:**
   - **TRF7970A:** The TRF7970A offers advanced features such as simultaneous protocol support, automatic modulation index adjustment, and selectable output power levels, providing flexibility in RFID communication.
   - **PN532:** The PN532 chip also offers versatility, supporting peer-to-peer communication, card emulation mode, and reader/writer mode.
4. **Availability and Cost:**
   - This chip is widely available, but it is more expensive than both other chips.

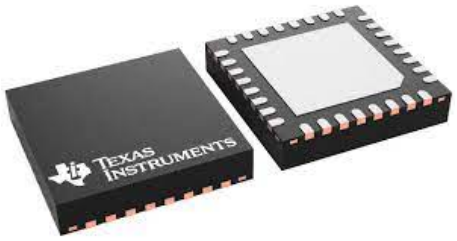


Figure 3: TEXAS TRF7970A IC

- **Link -**TRF7970A Datasheet
- **Link -**TRF7970A Antenna Design Guide

2.1.4 Selected IC – NXP PN532

The PN532's overall functionality can be separated into three functions:

- **Generate the RF field:** The generated magnetic field has to be maximized within the limits of the transmitter supply current and general emission limits.

- **Transmit data:** The coded and modulated data signal has to be transmitted in a way that every card and PN532 device is able to receive it. The signal shape and timing according to relevant standards have to be considered.
- **Receive data:** The response of a card or PN532 device has to be transferred to the receive input of the PN532 considering various limits, e.g., maximum voltage and receiver sensitivity.

2.2 RFID Reader Circuit Diagram (26/02/2024)

The PN532 (PN5321A3HN/C106) supports SPI (Serial Peripheral Interface) communication, allowing microcontrollers like the ESP32 to interact with RFID tags.

Here is the final circuit diagram for operating the Pn532 chip for RFID card reading under SPI communication.

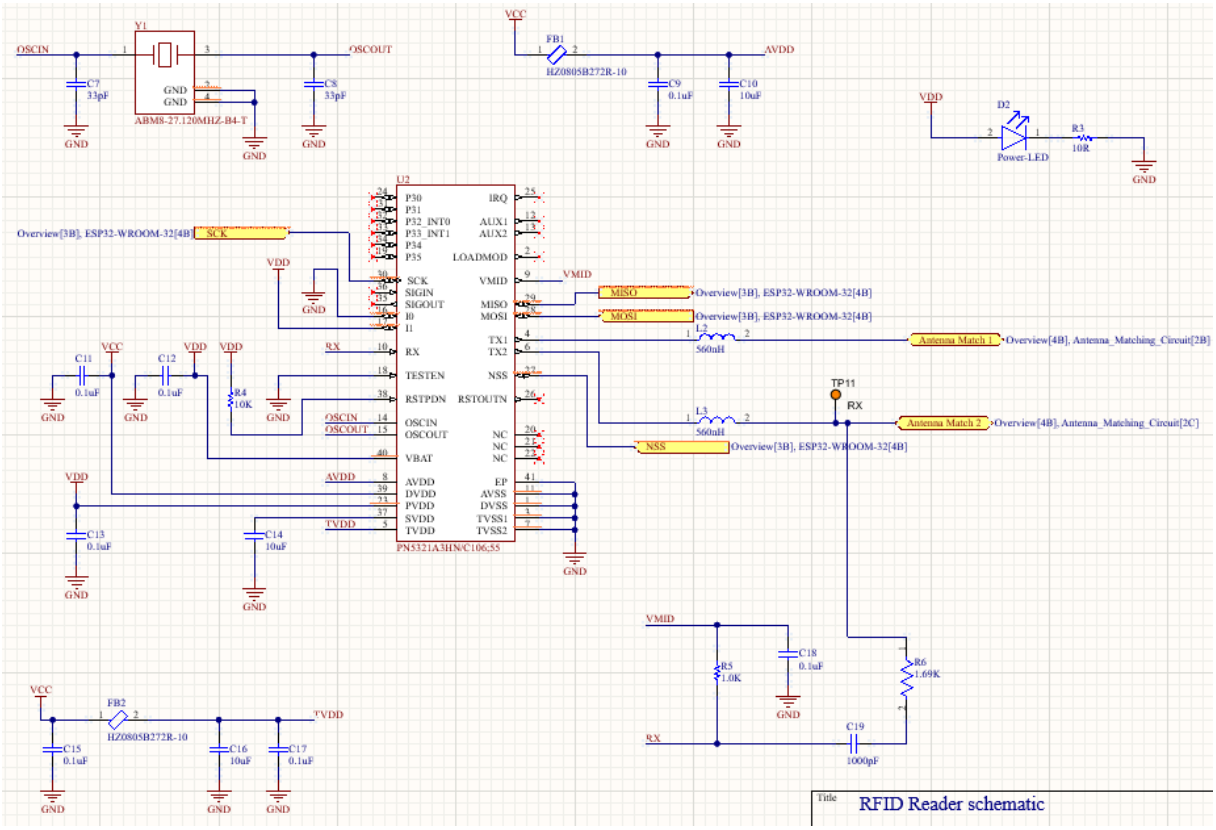


Figure 4: PN532 Circuit Diagram

To transmit data to the ESP32 using SPI, the following pins on the PN532 module should be connected to the corresponding pins on the ESP32:

- PN532 **pin 30** (SCK) to ESP32 SCK pin
- PN532 **pin 28** (MOSI) to ESP32 MOSI pin
- PN532 **pin 29** (MISO) to ESP32 MISO pin
- PN532 **pin 27** (SS) to ESP32 SS pin

The D1 LED turned on when the PN532 IC powered up. Also the PN532 IC operates using a 27.12MHz crystal oscillator for its internal timing requirements. This oscillator provides the necessary clock frequency for the IC to function correctly.

2.3 Antenna and Matching Circuit Design (11/03/2024)

The NXP Semiconductors PN532 chip, which enables contactless communication, is present in our PCB. When combined with the PCB antenna, it functions as an NFC reader for the ISO 14443 standard. The antenna has to match the PN532 chip in order to extend the reader's communication range. Aside from that, the signal that the tag reflected and the antenna detected needed to be optimized.

Since the PN532 chip outputs a 13.56MHz square wave, a low-pass filter is used to follow the output pins and produce a pure sine wave. The majority of power is sent in the reader-antenna configuration below when the antenna's impedance matches the chip's impedance.

The RF block diagram in Fig 7 shows a recommended circuitry design with all relevant components required to connect an antenna to the PN532. It also ensures the transmission of energy and data to the target device as well as the reception of a target device answer.

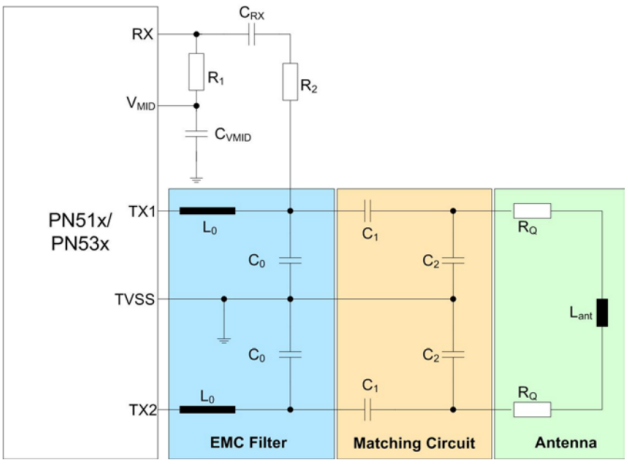


Figure 5: Block Diagram of complete RF part

- **EMC filter:** The EMC filter reduces 13.56 MHz harmonics and performs an impedance transformation.
- **Matching Circuit:** The Matching Circuit acts as an impedance transformation block.
- **Antenna coil:** The antenna coil itself generates the magnetic field.

The receiving part provides the received signal to the PN532 internal receiving stage.

Basically, this complete RF circuitry consists of at least 8 capacitors, 2 inductors, 2 resistors (the part size determines the maximum power which the resistor can withstand) and the symmetrical antenna coil as shown in Fig 1.

2.3.1 Equivalent circuit

There are two alternatives for equivalent circuit,

- **Series equivalent circuit:** The antenna loop has to be connected to an impedance or network analyzer to measure the series equivalent components.
- **Parallel equivalent circuit:** The parallel equivalent circuit of the antenna together with the added external damping resistor RQ.

2.3.2 Determination of parallel equivalent circuit

The parallel equivalent circuit of the antenna together with the added external damping resistor RQ has to be measured. The quality factor should be checked again to be sure to achieve the required value of Q=35.

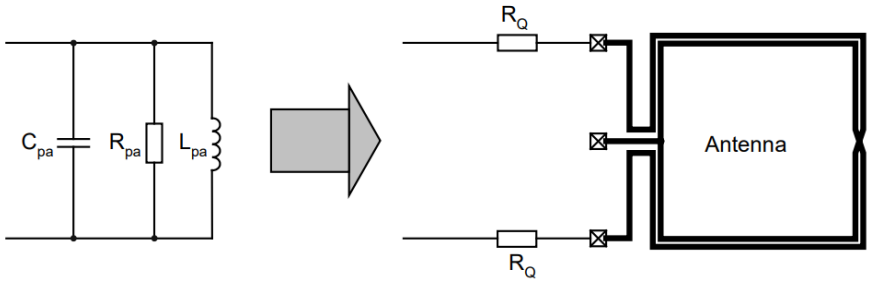


Figure 6: Parallel equivalent circuit

2.3.3 Circuit Diagram (Antenna, Matching Circuit, EMC Filter)

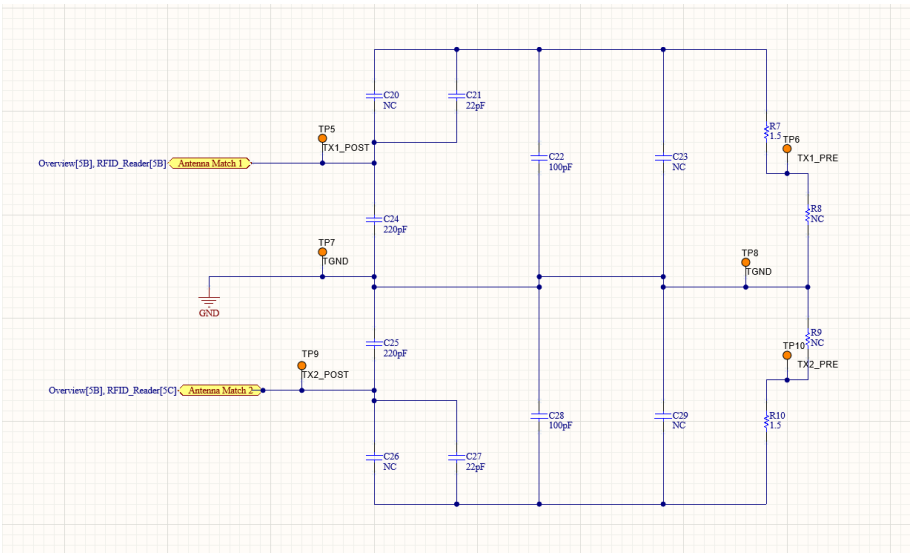


Figure 7: Circuit Diagram (Antenna, Matching Circuit, EMC Filter)

The circuit diagram for the PN532 RFID IC operating at 13.56MHz includes the antenna, matching circuit, and EMC filter. The antenna is designed to resonate at the operating frequency of 13.56MHz and is typically a coil or loop structure. The matching circuit is used to match the impedance of the antenna to the impedance of the PN532 IC for maximum power transfer. The EMC filter helps reduce electromagnetic interference (EMI) from the circuit, ensuring reliable operation and compliance with regulatory standards. This circuit diagram is crucial for understanding the RF front-end design of the RFID system using the PN532 IC.

According to the PN532 chip datasheet, the receive pin of the chip must not exceed AVDD +1 V, where AVDD is the analog supply voltage at 3.3 V, resulting in a maximum allowed voltage of 4.3 V on the RX pin. This limit applies when the chip is in continuous communication mode and the RF-field is continuously active. Even if the antenna is detuned by a tag, the voltage on the RX pin must not exceed 4.3 V. The antenna transfers the most power when its input impedance matches 50 Ω and the reactance is 0 Ω at the resonance frequency.

To ensure accurate measurements, it's important to keep the ground connection short and close to the measurement point, as any ground wire within the electromagnetic field can induce voltage in the probe, leading to unreliable measurements.

3 How to access WIFI connection, ESP32 WROOM 32 (25/03/2024)

The selection of the ESP32 Wroom 32 chip for WiFi connectivity in our project was a strategic choice driven by several key factors. Firstly, the ESP32 Wroom 32 offers robust and reliable WiFi connectivity, allowing us to seamlessly upload data to cloud storage. This is crucial for our project, as it enables real-time monitoring and tracking of assets in the warehouse environment.

The ESP32 Wroom 32 is highly versatile and supports a wide range of applications, making it suitable for our asset tracking device. Its compatibility with popular development platforms such as Arduino and its extensive documentation and community support were also significant factors in our decision-making process.

Additionally, the ESP32 Wroom 32 offers low power consumption, which is essential for the project's requirements. Its dual-core processor and ample memory resources provide the necessary processing power for handling data uploads to the cloud efficiently. The ESP32 Wroom 32 also features three power modes (active, sleep, and deep sleep), allowing us to optimize power consumption based on the device's activity. This feature is crucial for our application, as it helps us maintain power efficiency and avoid high-speed discharge of the forklift battery, ensuring reliable operation over extended periods.

Figure 8: ESP32 WROOM 32

- **Link -**ESP32 WROOM 32 datasheet

Overall, the ESP32 Wroom 32 chip stood out as the ideal choice for WiFi connectivity in this project, offering a combination of reliability, versatility, and performance that align perfectly with our project goals.

3.1 SPI Communication with ESP32 WROOM 32

The SPI Bus is a synchronous communication protocol used to transfer data between a microcontroller and other devices, such as sensors, displays, memories, and peripherals.

Unlike other protocols, the SPI Bus uses multiple communication lines, allowing for high transfer speeds and exceptional flexibility.

The SPI port is mainly used for its high data transmission capacity. For this reason, you will find it in devices that require a large amount of data per second, such as storage devices (memories, SD cards) and displays.

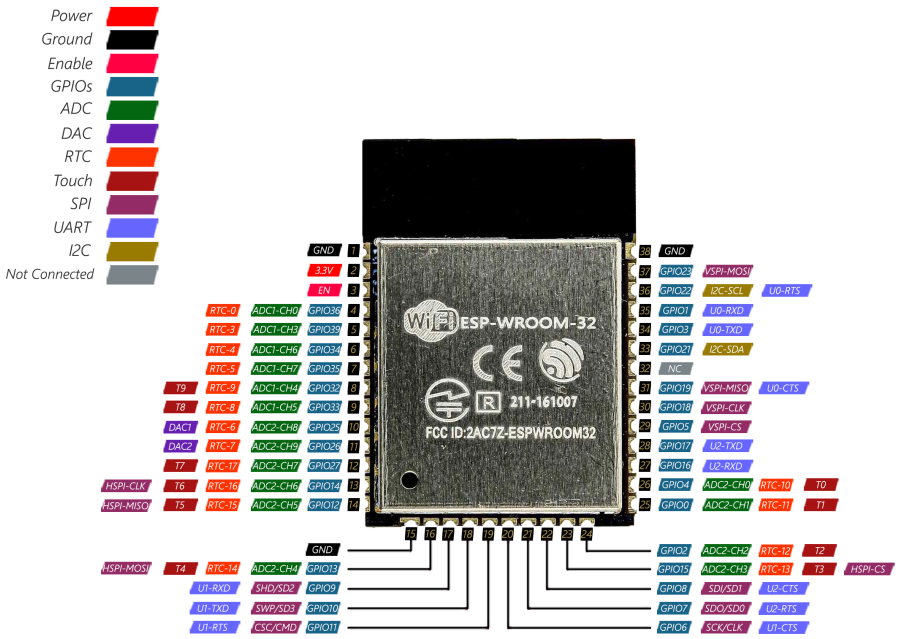


Figure 9: ESP32 WROOM 32 Pinout

The ESP32 WROOM 32 microcontroller supports two SPI (Serial Peripheral Interface) buses: VSPI (Virtual SPI) and HSPI (Hardware SPI). VSPI is a software-based SPI bus that can be mapped to any set of GPIO pins on the ESP32, providing flexibility in pin selection but may have slightly lower performance compared to HSPI. But in this case, performance is not a critical thing. So I am going to use VSPI communication.

- **VSPI MISO (Master In Slave Out):** GPIO 19
- **VSPI MOSI (Master Out Slave In):** GPIO 23
- **VSPI SCK (Serial Clock):** GPIO 18
- **VSPI CS (Chip Select):** GPIO 5

3.2 Programming ESP32 IC

Two important pins for programming the ESP32 chip are RX0 (Receive Data) and TX0 (Transmit Data). These pins are used for serial communication between the ESP32 and a programming device. RX0 is the input pin on the ESP32 for receiving data, while TX0 is the output pin for transmitting data. When programming the ESP32, these pins are typically connected to the corresponding RX and TX pins on the programming device. By using RX0 and TX0, developers can easily upload firmware and communicate with the ESP32 IC.

3.3 Final Circuit Diagram - ESP32 WROOM 32

Here is the final circuit diagram for ESP32 IC for connect with PN532 over VSPI communication.

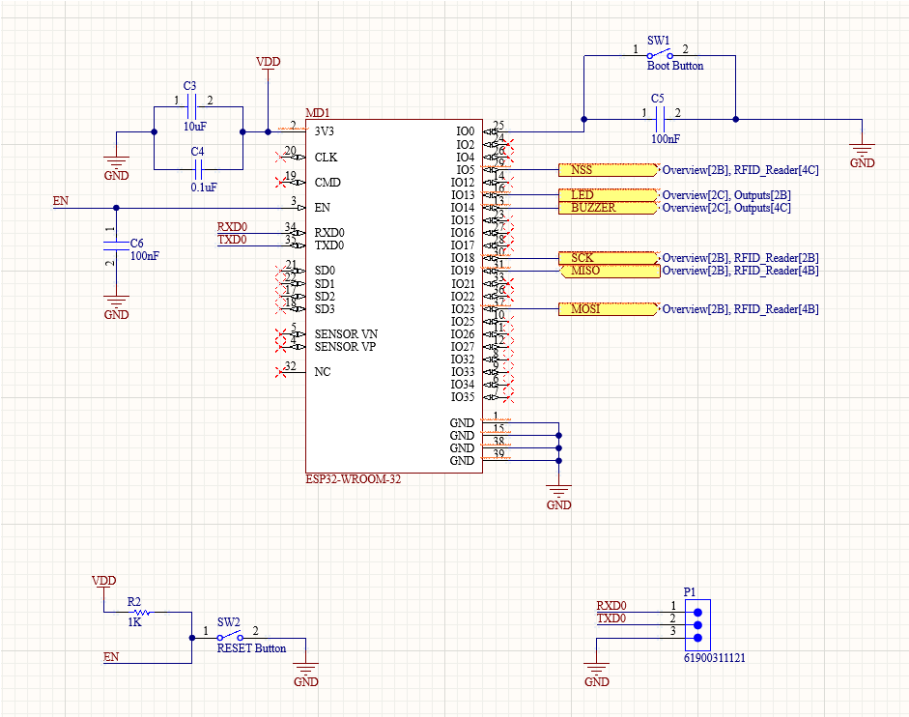


Figure 10: ESP32 WROOM 32 - Circuit Diagram

4 Power Circuit Design (02/04/2024)

In the power circuit design, the goal is to efficiently convert the 12V battery power from the forklift into a stable 3.3V output to power the components of the system. This section will detail the components and their connections required to achieve this voltage conversion, ensuring that the RFID reader and other electronics operate reliably. The design will focus on efficiency and reliability, considering factors such as voltage regulation, and current capacity.

4.1 Voltage Regulator - LM2596S-3.3

Selecting the right voltage regulator is crucial for ensuring stable and reliable power supply in electronic projects. For our project, which involves the PN532 and ESP32 WROOM 32 chips operating on 3.3V DC voltage, choosing the appropriate regulator was a critical decision. Factors such as efficiency, cost, thermal performance, and compatibility with the chips' voltage requirements were considered. After careful

evaluation, the LM2596S-3.3 regulator was selected due to its efficiency, cost-effectiveness, and compatibility with our project's needs. This selection process highlights the importance of choosing the right voltage regulator to meet the power supply requirements of electronic devices.

- **Link -**LM2596S-3.3 datasheet

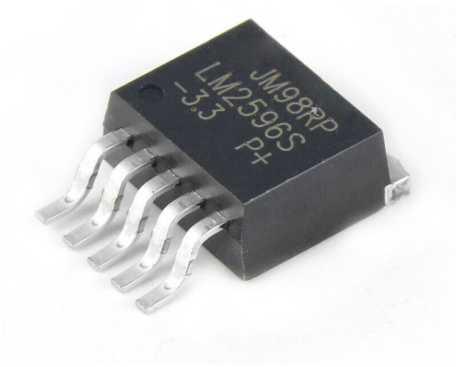


Figure 11: LM2596S-3.3 Regulator

4.2 Power Circuit Diagram

Here is the final power circuit diagram, showcasing the efficient conversion of the 12V battery power from the forklift into a stable 3.3V output.

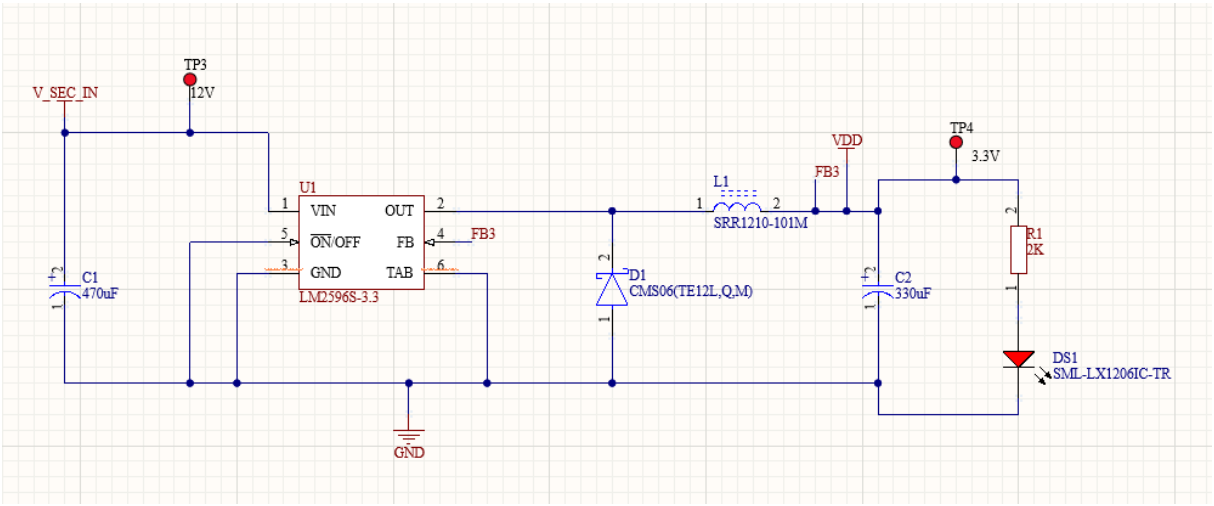


Figure 12: Power Circuit Diagram

- **D1 - CMS06 Schottky Diode:** The CMS06 is a compact and versatile power management module designed to provide efficient power conversion for various electronic devices. The CMS06 is selected for its compact size, high efficiency, and ability to handle the required voltage and current levels. Its inclusion ensures reliable and efficient power delivery, critical for the overall performance and functionality of the system.
- **DS1 - RED Led:** This indicator signifies the operational status of the device, providing a visual cue when power is actively supplied.

5 Output Circuit Design (04/04/2024)

The device integrates two indicators, a buzzer, and a green LED, to provide informative feedback to the user. These indicators play crucial roles in enhancing user interaction and awareness regarding asset detection:

- **Buzzer**: The buzzer serves as an auditory indicator, emitting a distinct beep sound when an asset is detected. This audible signal is useful in environments where visual cues may not be immediately noticeable, ensuring that users are promptly informed of detected assets.
- **Green LED**: The green LED functions as a visual indicator, illuminating to signal the detection of an asset. This visual feedback is valuable in situations where users need to quickly identify the presence of detected assets, providing a clear and intuitive indication.

The combination of these indicators offers users dual modes of feedback, accommodating different preferences and environmental conditions.

5.1 Buzzer Circuit

Presented below is the circuit diagram for connecting the buzzer, illustrating the necessary connections and components required for integrating the buzzer into the system.

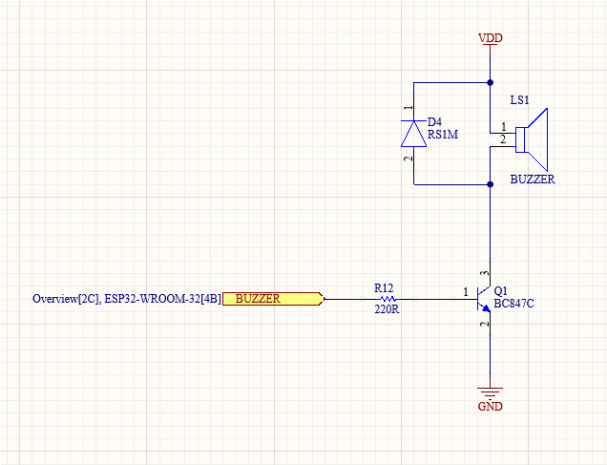


Figure 13: Circuit Diagram - Buzzer

5.2 LED Indicator Circuit

Presented below is the circuit diagram for connecting the LED, illustrating the necessary connections and components required for integrating the LED into the system.

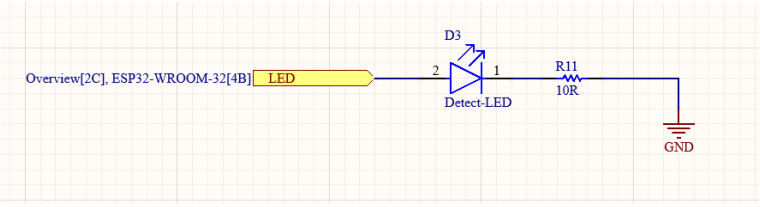


Figure 14: Circuit Diagram - LED Indicator

6 Interconnection Between Schematics

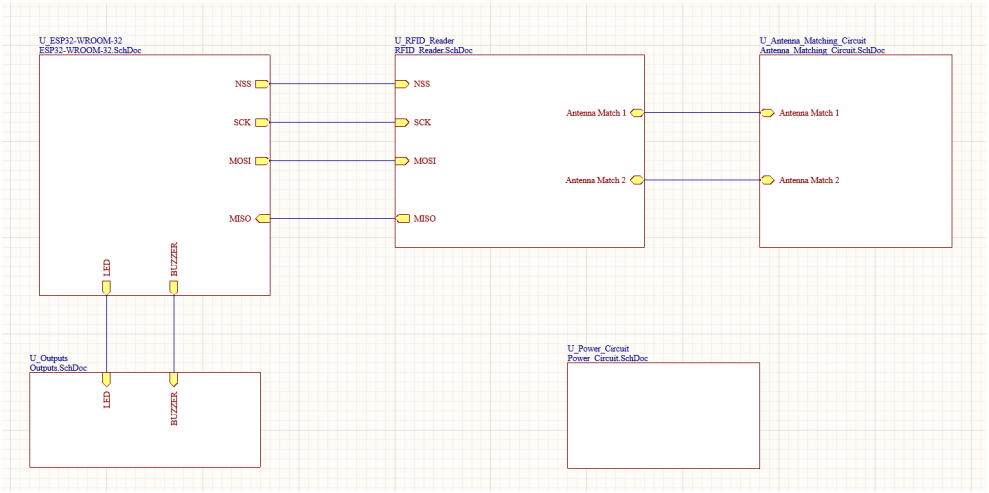


Figure 15: Overview

7 PCB Design (05/04/2024)

Within this subsection, detailed images of the PCB design are presented, showcasing the layout, component placement, and routing strategies employed in the development of the RFID-based asset tracking device. These images provide a visual representation of the circuit design and serve to illustrate the meticulous planning and implementation of the project.

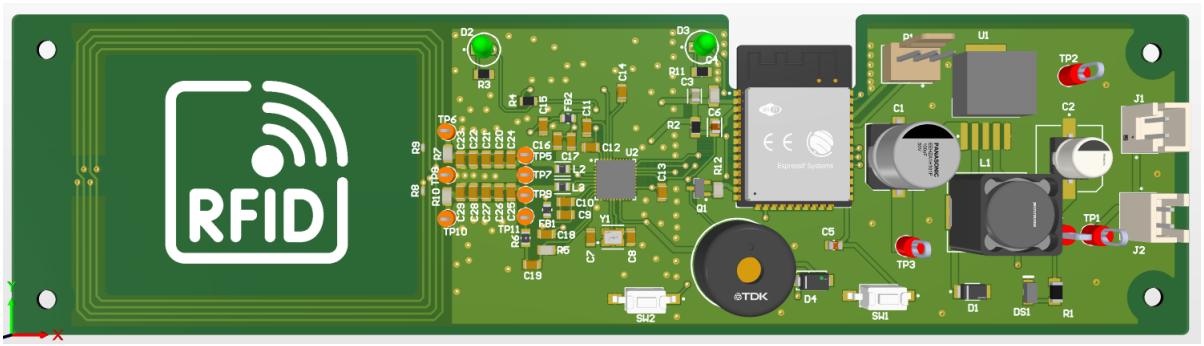
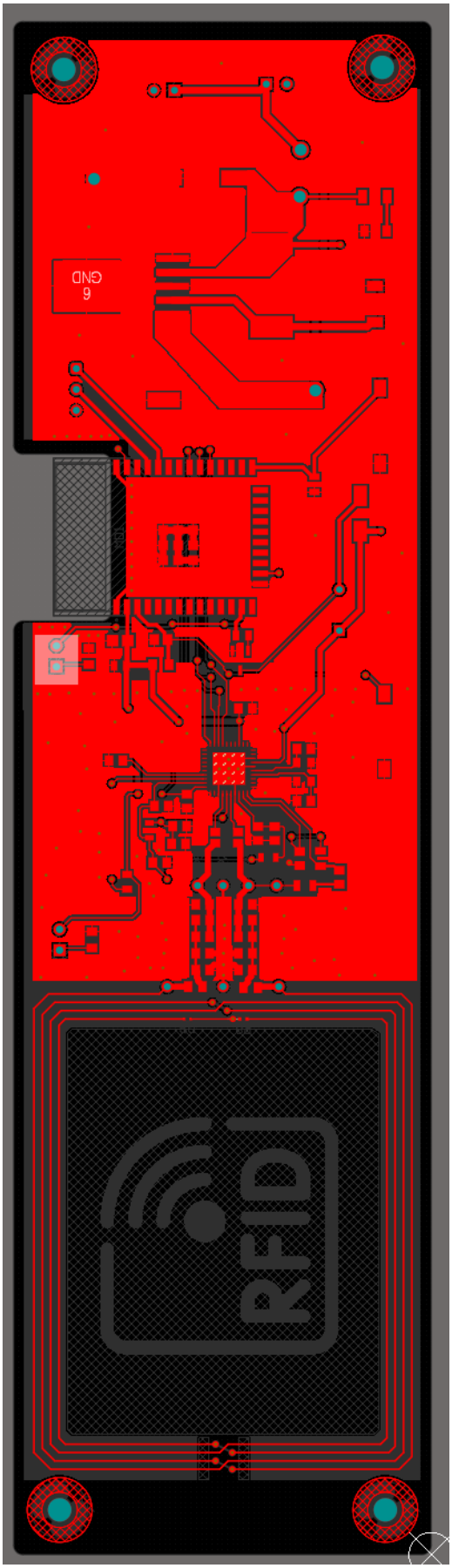
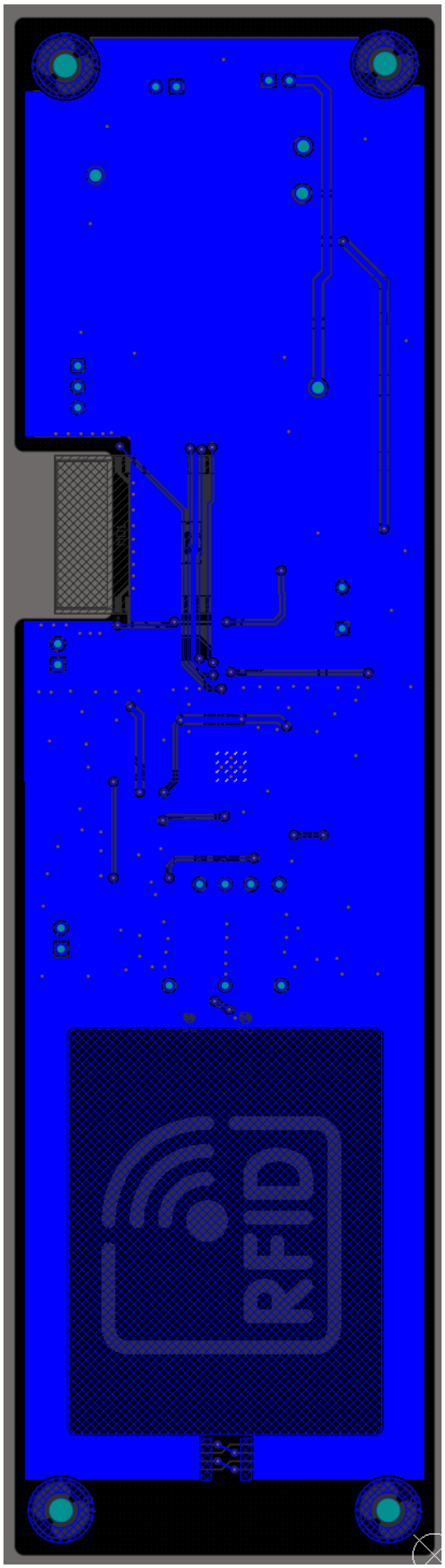


Figure 16: Top View

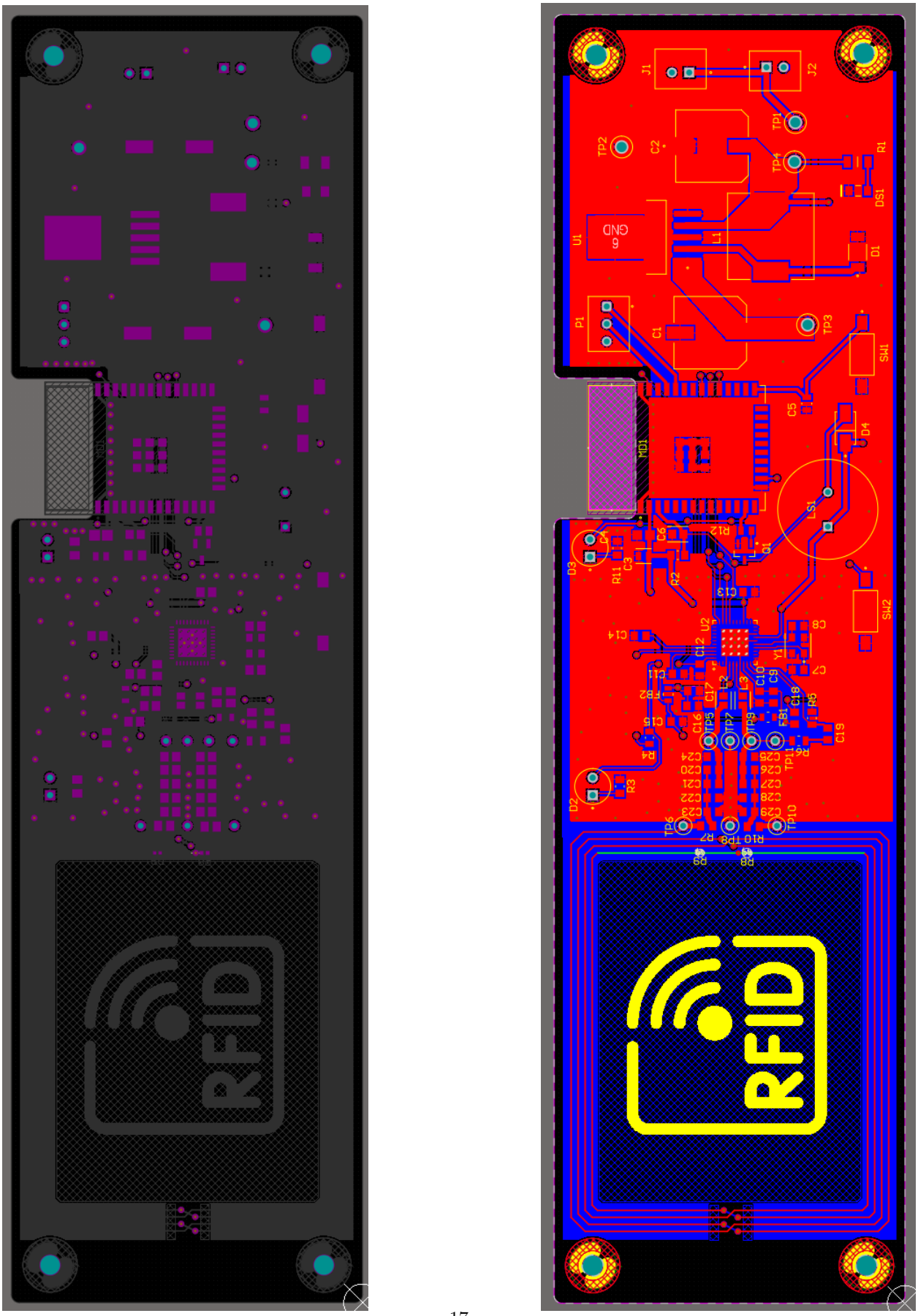


((a)). Top Layer



((b)). Bottom Layer

Figure 17: PCB Design (i)



((a)). Soldermask

((b)). PCB View

Figure 18: PCB Design (ii)

8 Bill Of Materials

The Bill of Materials (BOM) is a detailed inventory of all the components, parts, and materials needed for the construction and assembly of the RFID-based asset tracking device for forklifts. Each item listed in the BOM includes a unique designator, a description of the component, and the required quantity. The BOM ensures that all necessary components are accounted for, facilitating efficient procurement and assembly processes. The following table outlines the BOM for this project:

Designator	Description	Quantity
C1	0805 SMD 470uF Capacitor	1
C2	0805 SMD 330uF Capacitor	1
C3, C10, C14	0805 SMD 10uF Capacitor	3
C4, C5, C6, C9, C11, C12, C13	0805 SMD 0.1uF Capacitor	7
C7, C8	0805 SMD 33pF Capacitor	2
C19	0805 SMD 1nF Capacitor	1
C22, C28	0805 SMD 100pF Capacitor	2
C24, C25	0805 SMD 220pF Capacitor	2
C21, C27	0805 SMD 22pF Capacitor	2
R1, R2, R5	0805 SMD 1K Resistor	3
R3, R11	0805 SMD 10R Resistor	2
R4	0805 SMD 10K Resistor	1
R6	0805 SMD 1.69K Resistor	1
R12	0805 SMD 220R Resistor	1
R7, R10	0805 SMD 1.5R Resistor	2
D1	CMS06 - DIODE SCHOTTKY 30V 2A SMD	1
D2	SMD 0805 Blue LED	1
D3	Green through hole LED	1
D4	RS1M Zener Diode SMD	1
DS1	Red through hole LED	1
U1	LM2596S-3.3 SMD	1
U2	PM5321A3HW SMD	1
MD1	ESP32 - WROOM 32	1
Y1	27.12MHz Crystal Oscillator SMD	1
Q1	BC847C Shockley Diode SMD	1
LS1	Buzzer	1
L1	SMD 0805 100uH Inductor	1
L2, L3	SMD 0805 560nH Inductor	2
FB1, FB2	Ferrite Bead 2.7K	2
SW1, SW2	SMD Push Buttons	2
-	2 pin JST	2
-	Female Headers	3

9 Firmware Implementation (12/04/2024)

Firmware implementation is a crucial aspect of embedded systems development, involving the creation and deployment of software that operates directly on hardware devices like microcontrollers or system-on-chips (SoCs). In the context of this introduction, we will focus on the implementation of firmware using the ESP-IDF framework for the ESP32-WROOM32 IC, leveraging UART communication for uploading firmware.

The ESP32-WROOM32 IC is based on the Xtensa LX6 architecture developed by Tensilica (now owned by Cadence Design Systems). The Xtensa architecture is known for its flexibility and configurability, allowing for efficient customization of the processor's instruction set and hardware features to meet specific application requirements.

ESP-IDF (Espressif IoT Development Framework) is a comprehensive platform for developing applications for ESP32 and ESP32-S series SoCs. It provides a rich set of libraries, tools, and APIs tailored for efficient and scalable firmware development.

9.1 Importance of Register Programming

In firmware development, register programming plays a vital role in configuring and controlling hardware peripherals directly at the register level. This approach offers several key advantages:

- **Performance Optimization:** Register programming allows developers to fine-tune hardware settings for optimal performance, often achieving faster data processing and reduced latency compared to higher-level APIs.
- **Resource Efficiency:** By accessing hardware registers directly, firmware can conserve system resources such as memory and processing power, crucial in embedded systems where resource constraints are common.
- **Low-Level Access:** It facilitates precise control over timing, interrupts, and low-level hardware features, essential for tasks like real-time data acquisition, signal processing, and device synchronization.

```
#include <stdio.h>
#include <string.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/event_groups.h"
#include "nvs_flash.h"
#include "esp_wifi.h"
#include "esp_event.h"
#include "mqtt_client.h"
#include "driver/spi_master.h"
#include "pn532.h"

#define PN532_SCK 5
#define PN532_MISO 19
#define PN532_MOSI 23
#define PN532_SS 18

#define BUZZER_PIN 14
#define INDICATOR_PIN 13

#define WIFI_SSID "NTWS-4G"
#define WIFI_PASS "XXXXXXXXXXXX"
```

```
#define MQTT_BROKER "mqtt://test.mosquitto.org"

static const char *TAG = "RFID_TRACKER";

static EventGroupHandle_t wifi_event_group;
const int WIFI_CONNECTED_BIT = BIT0;

esp_mqtt_client_handle_t mqtt_client;
pn532_t pn532;

char details[50];
char details_long[200];
char idcard[50];
char idcard_old[50];

// Function prototypes
static void wifi_event_handler(void *arg, esp_event_base_t event_base,
                               int32_t event_id, void *event_data);
static void wifi_init(void);
static void mqtt_event_handler(void *handler_args, esp_event_base_t base,
                               int32_t event_id, void *event_data);
static void mqtt_app_start(void);
static void setup_pn532(void);
static void read_pn532(void);
static void indicate(void);

// Macros for direct register access
#define REG_WRITE(addr, val) (*((volatile uint32_t *)(addr)) = (val))
#define REG_SET_BIT(addr, bit) REG_WRITE((addr), (REG_READ(addr) | (1ULL
    << (bit))))
#define REG_CLEAR_BIT(addr, bit) REG_WRITE((addr), (REG_READ(addr) & ~(1
    ULL << (bit))))
#define REG_READ(addr) (*((volatile uint32_t *)(addr)))

// GPIO register addresses and bit masks for ESP32
#define GPIO_OUT_REG(pin)   (GPIO_REG_BASE + 0x0C + (4 * ((pin) % 32)))
#define GPIO_ENABLE_W1TC_REG(pin) (GPIO_REG_BASE + 0x88 + (4 * ((pin) /
    32)))
#define GPIO_ENABLE_W1TS_REG(pin) (GPIO_REG_BASE + 0x78 + (4 * ((pin) /
    32)))

// Function to set GPIO pin level
static inline void gpio_set_level(uint32_t pin, uint32_t level) {
    if (level)
        REG_SET_BIT(GPIO_OUT_REG(pin), pin % 32);
    else
        REG_CLEAR_BIT(GPIO_OUT_REG(pin), pin % 32);
}

// Function to set GPIO pin direction
static inline void gpio_set_direction(uint32_t pin, uint32_t mode) {
    if (mode == GPIO_MODE_OUTPUT) {
        REG_SET_BIT(GPIO_ENABLE_W1TS_REG(pin), pin % 32);
    } else {
```

```
        REG_CLEAR_BIT(GPIO_ENABLE_W1TC_REG(pin), pin % 32);
    }
}

// WiFi event handler
static void wifi_event_handler(void *arg, esp_event_base_t event_base,
                               int32_t event_id, void *event_data) {
    if (event_base == WIFI_EVENT && event_id == WIFI_EVENT_STA_START) {
        esp_wifi_connect();
    } else if (event_base == WIFI_EVENT && event_id ==
       WIFI_EVENT_STA_DISCONNECTED) {
        esp_wifi_connect();
        xEventGroupClearBits(wifi_event_group, WIFI_CONNECTED_BIT);
    } else if (event_base == IP_EVENT && event_id == IP_EVENT_STA_GOT_IP)
       {
        xEventGroupSetBits(wifi_event_group, WIFI_CONNECTED_BIT);
    }
}

// Initialize WiFi
static void wifi_init(void) {
    esp_err_t ret = nvs_flash_init();
    if (ret == ESP_ERR_NVS_NO_FREE_PAGES || ret ==
       ESP_ERR_NVS_NEW_VERSION_FOUND) {
        ESP_ERROR_CHECK(nvs_flash_erase());
        ret = nvs_flash_init();
    }
    ESP_ERROR_CHECK(ret);

    wifi_event_group = xEventGroupCreate();

    ESP_ERROR_CHECK(esp_netif_init());
    ESP_ERROR_CHECK(esp_event_loop_create_default());
    esp_netif_create_default_wifi_sta();

    wifi_init_config_t cfg = WIFI_INIT_CONFIG_DEFAULT();
    ESP_ERROR_CHECK(esp_wifi_init(&cfg));

    esp_event_handler_instance_t instance_any_id;
    esp_event_handler_instance_t instance_got_ip;
    ESP_ERROR_CHECK(esp_event_handler_instance_register(WIFI_EVENT,
       ESP_EVENT_ANY_ID,
                                                        &
                                                            wifi_event_handler
                                                            , NULL, &
                                                            instance_any_id
                                                            ));
    ESP_ERROR_CHECK(esp_event_handler_instance_register(IP_EVENT,
       IP_EVENT_STA_GOT_IP,
                                                        &
                                                            wifi_event_handler
                                                            , NULL, &
                                                            instance_got_ip
                                                            ));
```

```
114
115     wifi_config_t wifi_config = {
116         .sta = {
117             .ssid = WIFI_SSID,
118             .password = WIFI_PASS,
119         },
120     };
121     ESP_ERROR_CHECK(esp_wifi_set_mode(WIFI_MODE_STA));
122     ESP_ERROR_CHECK(esp_wifi_set_config(ESP_IF_WIFI_STA, &wifi_config));
123     ESP_ERROR_CHECK(esp_wifi_start());
124
125     xEventGroupWaitBits(wifi_event_group, WIFI_CONNECTED_BIT, pdFALSE,
        pdTRUE, portMAX_DELAY);
126 }
127
128 // MQTT event handler
129 static void mqtt_event_handler(void *handler_args, esp_event_base_t base,
130                                int32_t event_id, void *event_data) {
131     esp_mqtt_event_handle_t event = event_data;
132     switch (event->event_id) {
133         case MQTT_EVENT_CONNECTED:
134             esp_mqtt_client_subscribe(mqtt_client, "210610H-CardDetails",
                0);
135             break;
136         case MQTT_EVENT_DISCONNECTED:
137             break;
138         default:
139             break;
140     }
141 }
142
143 // Start MQTT client
144 static void mqtt_app_start(void) {
145     esp_mqtt_client_config_t mqtt_cfg = {
146         .uri = MQTT_BROKER,
147     };
148     mqtt_client = esp_mqtt_client_init(&mqtt_cfg);
149     esp_mqtt_client_register_event(mqtt_client, ESP_EVENT_ANY_ID,
        mqtt_event_handler, NULL);
150     esp_mqtt_client_start(mqtt_client);
151 }
152
153 // Setup PN532 module
154 static void setup_pn532(void) {
155     spi_bus_config_t buscfg = {
156         .miso_io_num = PN532_MISO,
157         .mosi_io_num = PN532_MOSI,
158         .sclk_io_num = PN532_SCK,
159         .quadwp_io_num = -1,
160         .quadhd_io_num = -1,
161         .max_transfer_sz = 0,
162     };
163     spi_device_interface_config_t devcfg = {
164         .clock_speed_hz = 1 * 1000 * 1000, // 1 MHz
```

```
        .mode = 0,
        .spics_io_num = PN532_SS,
        .queue_size = 7,
    };

    ESP_ERROR_CHECK(spi_bus_initialize(HSPI_HOST, &buscfg, 1));
    ESP_ERROR_CHECK(spi_bus_add_device(HSPI_HOST, &devcfg, &pn532.
       spi_handle));

    pn532_init(&pn532);
    uint32_t versiondata = pn532_get_firmware_version(&pn532);
    if (!versiondata) {
        ESP_LOGE(TAG, "Didn't find PN53x board");
        while (1);
    }

    pn532_set_passive_activation_retries(&pn532, 0xFF);
    ESP_LOGI(TAG, "PN532 setup complete");
}

// Read data from PN532
static void read_pn532(void) {
    uint8_t success;
    uint8_t uid[7]; // Buffer to store the returned UID
    uint8_t uidLength;
    success = pn532_read_passive_target(&pn532, PN532_MIFARE_ISO14443A,
       uid, &uidLength, 0);
    if (success) {
        idcard_old[0] = '\0';
        strcpy(idcard_old, idcard);

        idcard[0] = '\0';
        for (uint8_t i = 0; i < uidLength; i++) {
            char hex[3];
            sprintf(hex, "%02X", uid[i]);
            strcat(idcard, hex);
        }

        if (strcmp(idcard, idcard_old) != 0) {
            snprintf(details, sizeof(details), "%s", idcard);
            esp_mqtt_client_publish(mqtt_client, "210610H-CardDetails",
               details, 0, 1, 0);
            indicate();
        }
    }
}

// Indicate action with LEDs and buzzer
static void indicate(void) {
    gpio_set_level(BUZZER_PIN, 1);
    gpio_set_level(INDICATOR_PIN, 1);
    vTaskDelay(400 / portTICK_PERIOD_MS);
    gpio_set_level(BUZZER_PIN, 0);
    gpio_set_level(INDICATOR_PIN, 0);
```

```
}

// Main application entry point
void app_main(void) {
    esp_log_level_set("*", ESP_LOG_INFO);

    // Initialize GPIO pins using direct register access
    gpio_set_direction(BUZZER_PIN, GPIO_MODE_OUTPUT);
    gpio_set_direction(INDICATOR_PIN, GPIO_MODE_OUTPUT);

    // Initialize WiFi, MQTT, and PN532 module
    wifi_init();
    mqtt_app_start();
    setup_pn532();

    // Main loop to read from PN532 and handle MQTT communication
    while (1) {
        read_pn532();
        vTaskDelay(1000 / portTICK_PERIOD_MS);
    }
}
```

Listing 1: C Code for programming ESP32 IC - main.c

The CMakeLists.txt file is a configuration file used by CMake, a cross-platform build system generator. CMake automates the process of configuring build environments, making it easier to manage large projects and their dependencies. When working with the ESP-IDF (Espressif IoT Development Framework) for ESP32, CMakeLists.txt is used to specify how the project should be built.

```
# In main/CMakeLists.txt

idf_component_register(SRCS "main.c" "pn532.c"
                    INCLUDE_DIRS "."
                    REQUIRES esp_netif esp_wifi mqtt spi_master)
```

```
                   /your_project
                             CMakeLists.txt
                             sdkconfig
                             main
                                 CMakeLists.txt
                                 main.c
                                 pn532.c
                                 pn532.h
                             components
                                 Adafruit-PN532
                                     Adafruit_PN532.cpp
                                     Adafruit_PN532.h
                                     ...
                                     library.properties
```

Listing 2: Project Directory Structure in ESP-IDF

1. `setupWifi()`: This function is responsible for setting up the WiFi connection. It attempts to connect to a WiFi network with the SSID "NTWS-4G" and the password "XXXXXXXXXXXX". It waits in a loop until the connection is established.

2. `setupMqtt()`: Sets up the MQTT client with the MQTT broker's address and port. It initializes the `PubSubClient` object `mqttClient` to communicate with the MQTT broker at "test.mosquitto.org" on port 1883.

3. `setupPn532()`: Initializes the PN532 NFC module. It starts the PN532 module and checks the firmware version. If the firmware version cannot be retrieved, the function enters an infinite loop, halting further execution.

4. `readPn532()`: Reads the NFC tag using the PN532 module. It checks if a card is present and reads its unique identifier (UID). If a new card is detected (UID is different from the previous one), it converts the UID to a string and publishes it to the MQTT topic "210610H-CardDetails". It also calls the `Indicate()` function to indicate a successful read.

5. `connectToBroker()`: Attempts to connect to the MQTT broker. If the connection is successful, it publishes a message to the MQTT topic "210610H-Details" indicating that the RFID-based asset tracking device is connected. If the connection fails, it retries after a delay of 5 seconds.

6. `Indicate()`: This function indicates a successful NFC tag read. It activates a buzzer and LED for a short duration to provide visual and auditory feedback.

9.2 Compatibility Issue Between the Adafruit PN532 library and ESP32 IC

When implementing the software component, I encountered an issue where the PN532 board was not being properly detected by the ESP32 board, despite correct wiring. Upon investigation, I identified this as a problem stemming from the Adafruit PN532 library.

The specific issue with the Adafruit PN532 library pertains to the communication bitrate configuration when used with an ESP32 board. By default, the library sets the bitrate to 1000000 (1 MHz), which typically functions adequately. However, when the ESP32 is operating at a clock speed of 240 MHz, communication becomes erratic and prone to glitches.

To mitigate this issue, it is necessary to adjust the bitrate in the Adafruit PN532 library to 100000 (100 kHz) when the ESP32 is running at 240 MHz. This lower bitrate stabilizes communication and prevents glitches from occurring. This indicates a potential need for optimization or adjustment in the library to ensure compatibility with higher clock speeds.

So you need to add the following code to the `Adafruit_PN532` library to resolve that problem.

```
Adafruit_PN532::Adafruit_PN532(uint8_t ss) {

    #ifdef ESP32
        uint32_t frequency = 100000;
    #else
        uint32_t frequency = 1000000;
    #endif

    spi_dev = new Adafruit_SPIDevice(ss, frequency, SPI_BITORDER_LSBFIRST,
        SPI_MODE0);
}
```

Listing 3: Additional code for Resolve the issue

10 Enclosure Design (08/03/2024)

This section showcases the SolidWorks designs of the enclosure for the RFID-based asset tracking device, presenting detailed images of the top, bottom, and assembly views. These images highlight the design's aesthetics, functionality, and compatibility with the forklift, providing insights into the enclosure's robustness and suitability for industrial use.

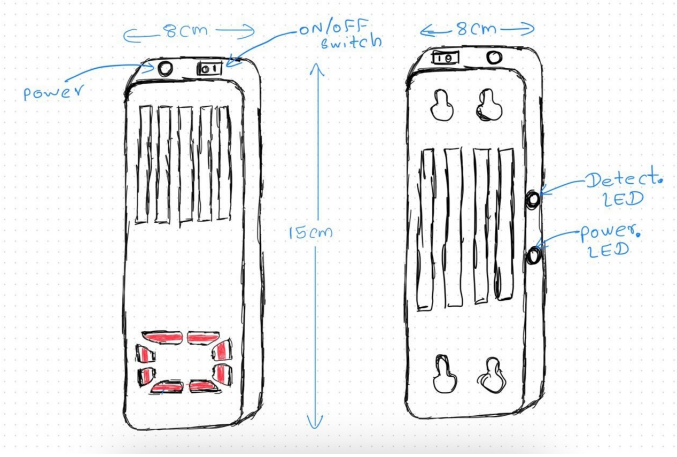
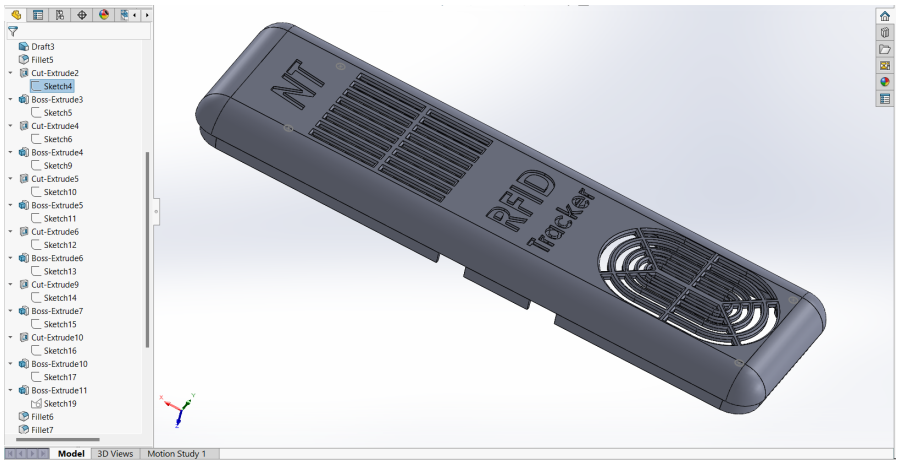
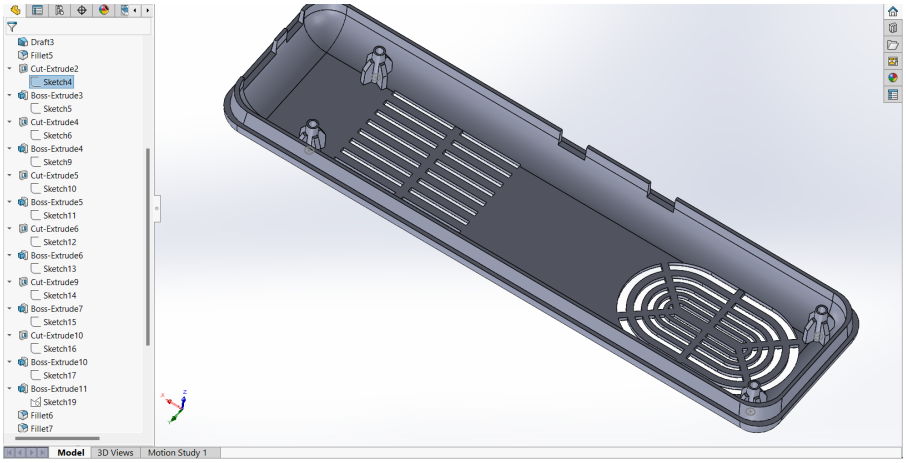


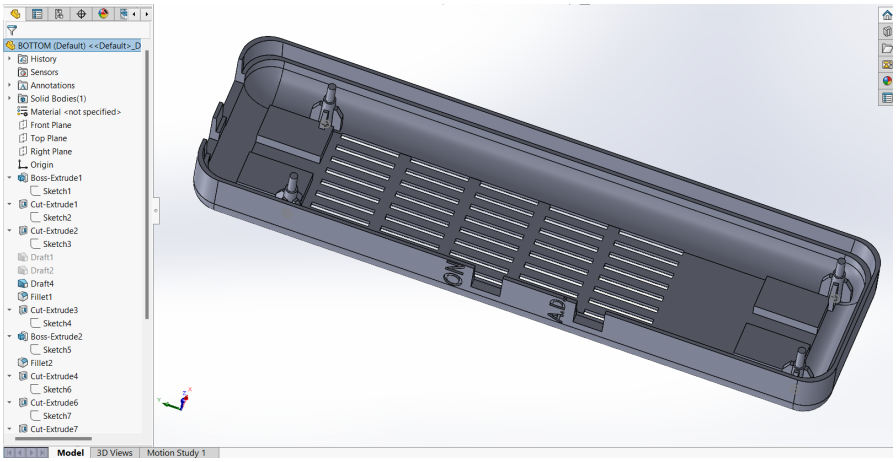
Figure 19: Handsketch

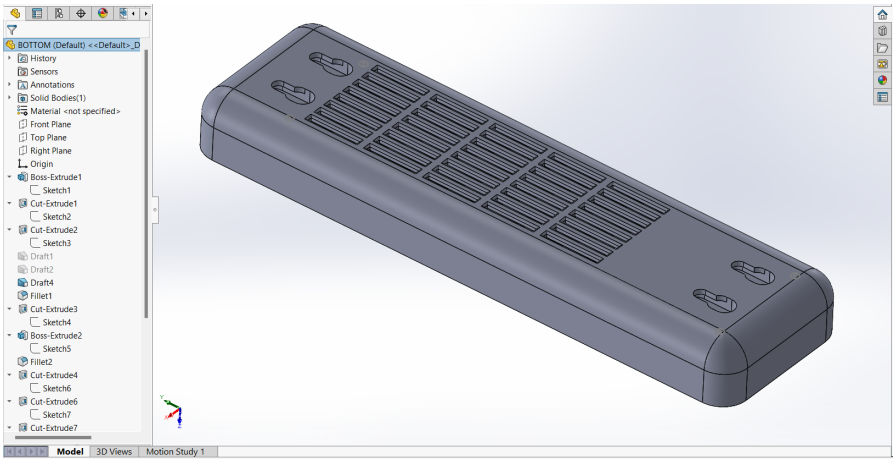
10.1 Upper Housing Component



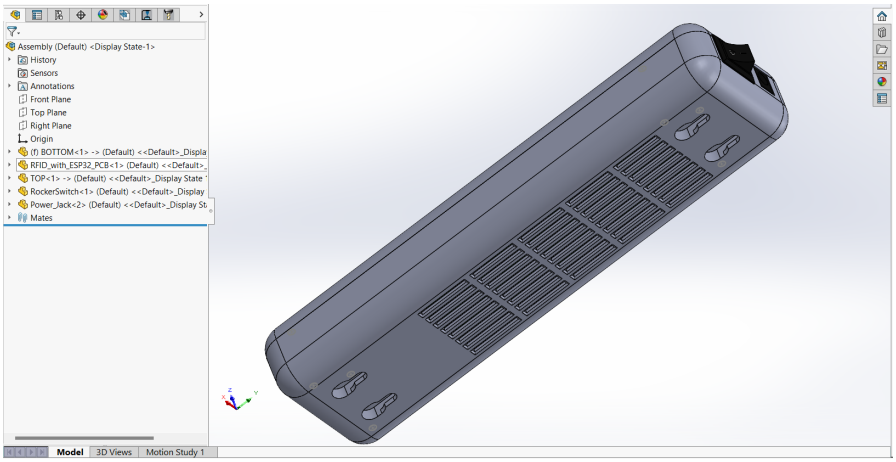


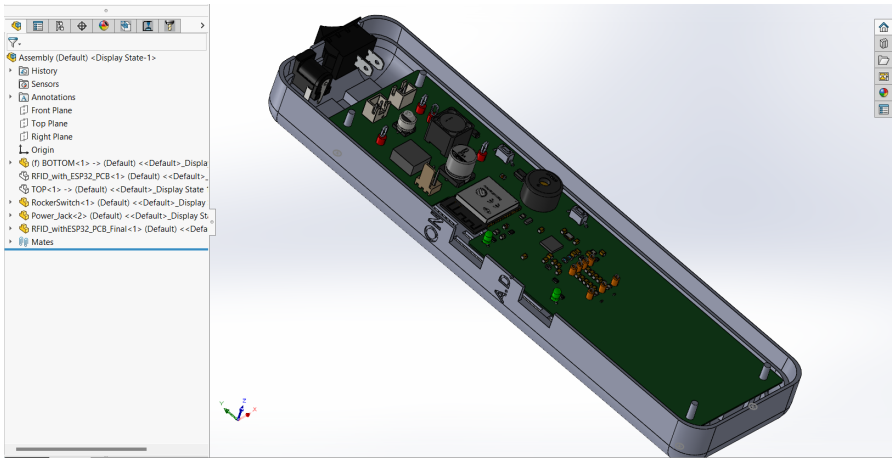
10.2 Lower Chassis Component





10.3 Enclosure Integration System





11 Other Components

11.1 ON OFF Switch

To enhance the functionality and user experience of the device, an on/off switch has been integrated into the design. This switch allows users to easily power the device on or off.

Figure 20: ON OFF Switch

11.2 Power Input

A DC barrel female socket has been integrated into the device as a power input enhancement. This feature enables direct input of 12V from a forklift battery, ensuring a dependable and uninterrupted power supply. Users are required to utilize a compatible barrel connector for this purpose.

Figure 21: Base Barrel Connector

12 PCB Manufacturing (03/05/2024)

PCB was manufactured by JLC PCB.

12.1 PCB Specifications

- **Layers**: 2
- **Dimension**: 186.8 mm x 50.8 mm
- **Product Type**: Industrial/Consumer Electronics
- **PCB Thickness**: 0.6 mm
- **Material Type**: FR4-Standard TG 135-140
- **Via Covering**: Tented
- **Surface Finish**: HASL (with lead)
- **Outer Copper Weight**: 1 oz

13 Bare PCB

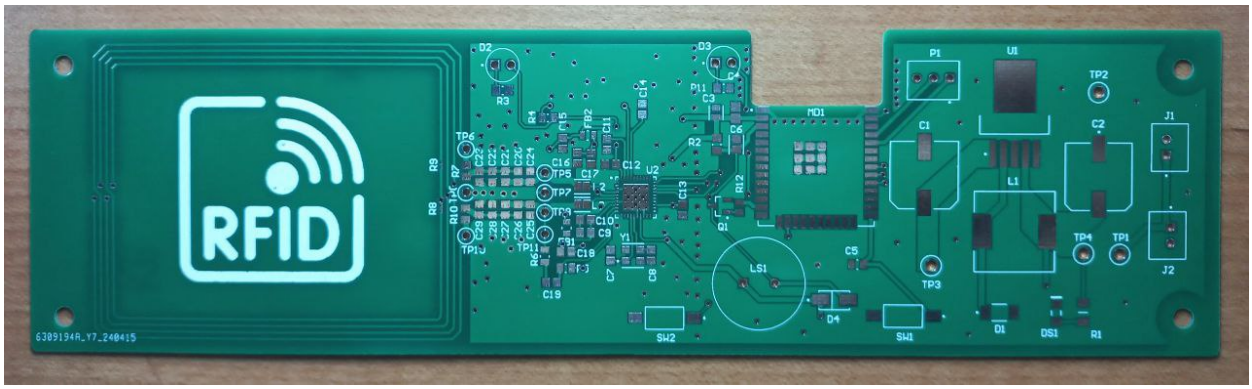


Figure 22: Upside View

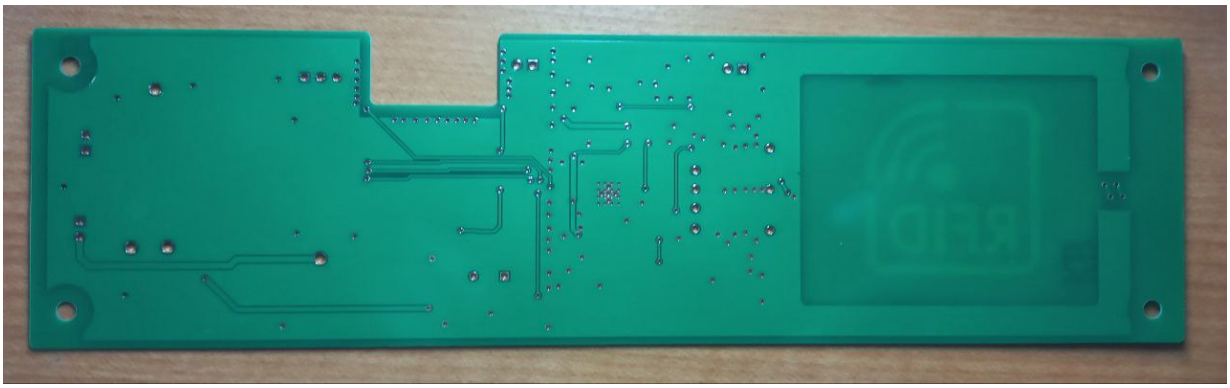
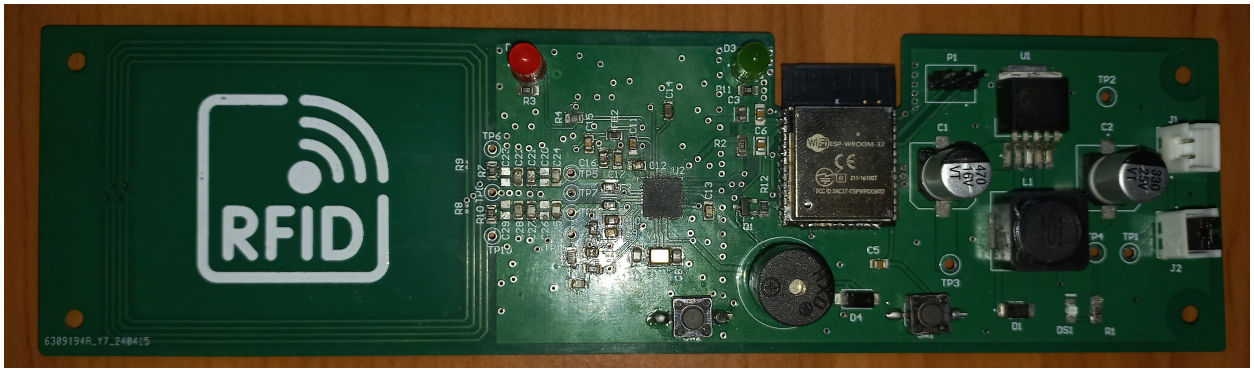


Figure 23: Downside View

13.1 Soldered PCB



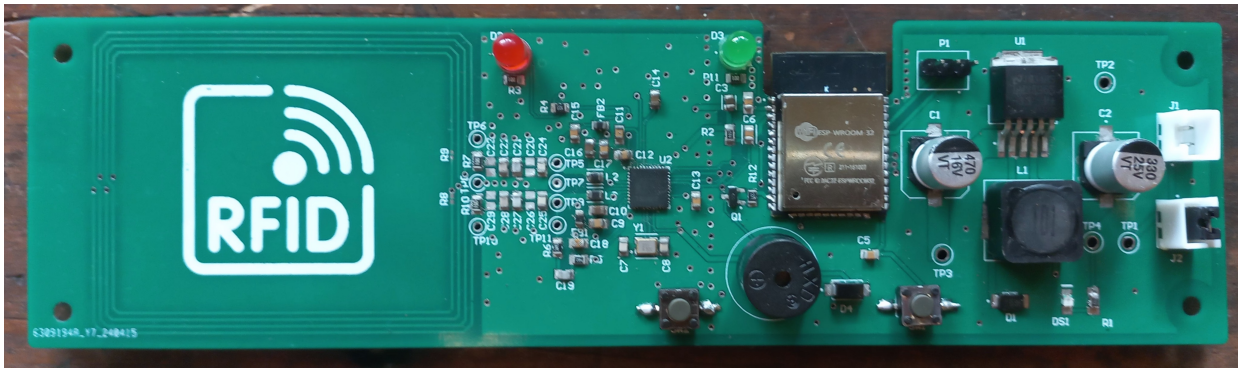


Figure 24: Upside View

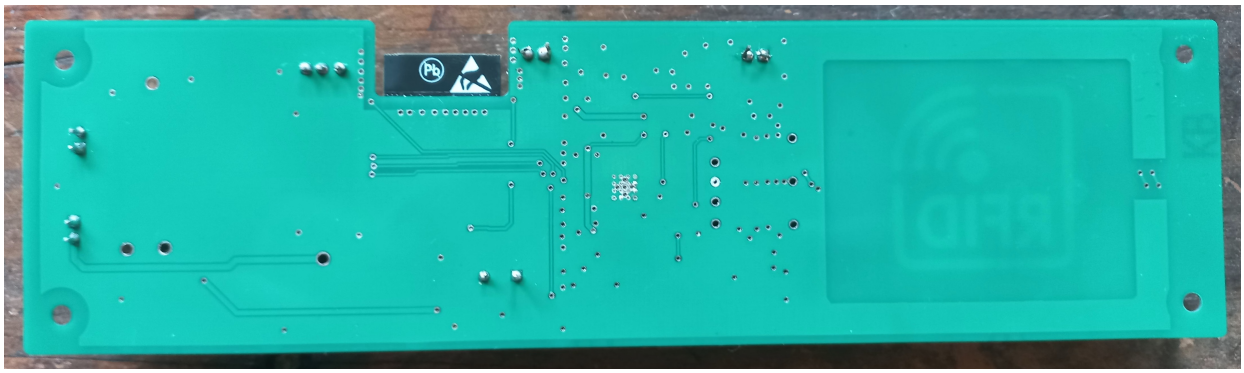


Figure 25: Downside View

14 PCB Testing

This task has not yet been completed due to the inability to use the workshop since it is closed due to the non-academic staff strike. This will be completed, and the relevant documentation will be updated once it is done.

15 Physically Build Enclosure

Figure 28: Printed Enclosure

16 System Integration

To be done.

17 Product Specifications

- **Input Voltage**: 12 V (From Forklift Battery)
- **Operating Voltage**: 3.3 V
- **Reading Distance (max)**: (To be tested)
- **Maximum Power**: (To be tested)
- **RFID Operating Frequency**: 13.56MHz
- **Minimum Buzzer Sound Pressure Level**: (To be tested)
- **Operating Ambient Temperature**: (To be tested)
- **Wi-Fi Connectivity**: Wi-Fi 2.4GHz
- **Product Dimensions**: $210 \times 52 \times 48$ mm

18 Future Improvements

In future iterations of my RFID-based asset tracking device, I plan to integrate more advanced RFID readers with enhanced capabilities. These readers will support higher read ranges and faster processing speeds, enabling more efficient asset tracking even in larger and more complex warehouse environments. Additionally, I aim to implement multi-frequency RFID readers that can simultaneously operate in different RFID frequency bands, providing greater flexibility and compatibility with a wider range of RFID tags. This enhancement will significantly improve the overall performance and reliability of the asset tracking system.

To further leverage the data collected by my RFID system, I plan to develop advanced data analytics and reporting features. By incorporating machine learning algorithms and data visualization tools, my system will be able to provide deeper insights into asset usage patterns, inventory levels, and operational efficiency. Users will benefit from customizable dashboards and real-time alerts, allowing them to make informed decisions and optimize their warehouse operations. These features will not only enhance the utility of the asset tracking device but also provide actionable intelligence for better resource management and cost reduction.

As part of my vision for future developments, I aim to integrate my RFID-based asset tracking device with broader IoT and Industry 4.0 ecosystems. By connecting my device to other smart sensors and automation systems within the warehouse, I can create a more cohesive and intelligent asset management solution. This integration will enable seamless communication and interoperability between different systems, leading to improved automation, predictive maintenance, and enhanced operational efficiency. Ultimately, my goal is to transform the asset tracking device into a key component of a fully integrated, smart warehouse environment.

19 Acknowledgement

I deeply appreciate and am grateful to everyone who has played a significant role in the development of this exceptional product. The successful completion of this design document would not have been possible without the collective efforts and support from all contributors.

Initially, I would like to extend my sincere thanks to my lecturer for his unwavering encouragement and support throughout this project. His dedication to innovation and strategic guidance has been instrumental in shaping my ideas and driving me toward success.

We also thank the members of our main project group for their support. Their continuous encouragement, rigorous testing, and meticulous attention to detail have ensured that our product meets the highest standards of reliability and performance.

20 Data sheets and References

1. **Case study of smart forklift integration with RFID -** (https://www.turck.nl/static/media/downloads/more22254e.pdf).
2. **An RFID data-based forklift traffic management system by Research gate -** (https://www.researchgate.net/figure/An-RFID-data-based-forklift-traffic-management-system_fig6_228657522).
3. **PN532/C1 datasheet -** (https://www.nxp.com/docs/en/nxp/data-sheets/PN532_C1.pdf).
4. **PN532 and MFRC522 Antenna Design Guideline -** (https://community.nxp.com/pwmxy87654/attachments/pwmxy87654/nfc/3055/1/AN1445-2010%20-%20Antenna%20design%20guide%20for%20MFRC52x,%20PN51x,%20PN53x.pdf).
5. **TRF79xxA datasheet -** (https://pdf1.alldatasheet.com/datasheet-pdf/download/1244246/TI1/TRF79XXA.html).
6. **Antenna Design Guide for the TRF79xxA -** (https://www.ti.com/lit/pdf/sloa241).
7. **ESP32 WROOM 32 datasheet -** (https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf).
8. **LM2596S-3.3 datasheet -** (https://www.ti.com/lit/ds/symlink/lm2596.pdf).

21 Appendix - Daily Log Entries

- **29 January - 4 February** At the outset of the Electronic Design Realization module, my first task was to select a project. After thorough research into various industrial project scopes, I finalized three potential projects. Following a presentation to our lecturer, I were pleased to be accepted for my project based on RFID technology. This marked the beginning of my journey into RFID technology, where I sought to understand its basic principles and real-world applications. I explored RFID's role in warehouse management, gaining valuable insights into its potential uses and benefits.

- **5 February - 9 February** This week, I focused on deepening our understanding of RFID technology, particularly its applications in warehouse management. Our lecturer provided us with a unique opportunity to visit operational warehouses that use RFID devices for managing goods and activities. This firsthand experience inspired us to delve deeper into the technology, studying existing RFID devices and systems used in warehouses. Despite the widespread use of RFID in foreign warehouses, we observed a scarcity of RFID technology in Sri Lanka. This realization motivated us to explore the potential for introducing RFID technology in local warehouse management systems.

- **10 February - 23 February** As part of a main group tasked with developing a comprehensive RFID solution for warehouse management, we divided ourselves into three specialized teams: forklift reader, portal reader, and handheld reader. I were assigned to develop the RFID based forklift reader. My focus shifted towards researching existing forklift RFID readers, analyzing their features, strengths, and weaknesses. This analysis provided valuable insights into the improvements and innovations needed in forklift RFID reader technology.

- **24 February - 1 April** During this period, I focused on several key aspects of my RFID Reader Design 3 project.

  - **RFID Chip Selection (24/02/2024):**
    I researched various RFID chips, including the NXP MFRC522, NXP PN532, and Texas TRF7970A, evaluating their features and specifications. After careful consideration, I selected the NXP PN532 as the most suitable chip for my project due to its compatibility with different RFID frequencies and its robust performance.
  - **RFID Reader Circuit Diagram (26/02/2024):**
    I created a detailed circuit diagram for the RFID reader, outlining the connections and components required for the project. This diagram served as a crucial reference point for the assembly and testing phases of the project.
  - **Enclosure Design (08/03/2024):**
    I worked on the design of the enclosure for the RFID reader, focusing on the upper housing component and the lower chassis component. Additionally, I developed an enclosure integration system to ensure the components fit together securely and efficiently.
  - **Antenna and Matching Circuit Design (11/03/2024):**
    I designed the antenna and matching circuit for the RFID reader, ensuring optimal performance and compatibility with the chosen RFID chip. This involved determining the equivalent circuit and creating a circuit diagram that integrated the antenna, matching circuit, and EMC filter.
  - **How to Access WiFi Connection, ESP32 WROOM 32 (25/03/2024):**
    I researched and implemented methods to access the WiFi connection using the ESP32 WROOM 32. This involved establishing SPI communication with the ESP32 and programming the IC to enable WiFi connectivity. I also finalized the circuit diagram for the ESP32 WROOM 32, ensuring seamless integration with the RFID reader circuit.

  These details are included on the following pages:

  - RFID Chip Selection: Page 3
  - RFID Reader Circuit Diagram: Page 6

  - Antenna and Matching Circuit Design: Page 7
  - How to Access WiFi Connection, ESP32 WROOM 32: Page 9
  - Enclosure Design: Pages 19-20

- **2 April - 26 April** During this period, I focused on the power circuit design and output circuit design for my RFID Reader Design for forklift project.

  - **Power Circuit Design (02/04/2024):**
    I designed the power circuit for the RFID reader, incorporating a voltage regulator LM2596S-3.3 to ensure stable power supply to the components. This design was crucial for maintaining the proper functioning of the RFID reader.
  - **Output Circuit Design (04/04/2024):**
    I designed the output circuit for the RFID reader, including circuits for a buzzer and LED indicator. These circuits were essential for providing feedback to the user regarding the RFID reading process, ensuring smooth operation of the device.
  - **Software Implementation (12/04/2024):**
    I encountered a compatibility issue between the Adafruit PN532 library and the ESP32 IC during the software implementation phase. This required me to troubleshoot the issue and find a workaround to ensure proper communication between the RFID reader and the ESP32 IC.

  The details for each section are included on the following pages:

  - Power Circuit Design: Page 11
  - Output Circuit Design: Page 13
  - Software Implementation: Page 16

- **3 July - 7 July** Initially, I developed the firmware for our project using the Arduino IDE. The Arduino IDE is known for its user-friendly interface, making it popular among hobbyists and for rapid prototyping. However, after consulting with our professor, we realized that while the Arduino IDE simplifies coding, it is not considered a professional tool for engineering-grade projects. It abstracts many lower-level operations, which can limit the control and optimization necessary for more complex and high-performance applications.

  ESP-IDF provides a more robust development environment, allowing for finer control over the hardware through register programming. This approach not only enhances performance but also aligns with best practices in embedded systems engineering.

  Previously, my programming approach did not involve direct register manipulation, relying instead on high-level functions provided by the Arduino framework. With the switch to ESP-IDF, I have learned to engage in register-level programming, which provides greater flexibility and efficiency, crucial for our project's requirements.

  For reference, I have included the initial code implemented using the Arduino IDE in the appendix below.

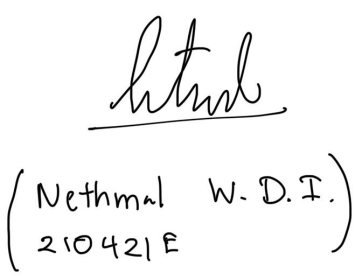
```
#include <SPI.h>
#include <Adafruit_PN532.h>
#include <WiFi.h>
#include <PubSubClient.h>

#define PN532_SCK  (5) // SCK (Serial Clock)
#define PN532_MISO (19) // MISO (Master In Slave Out)
```

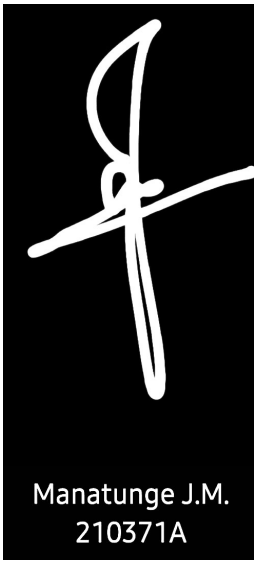
```
#define PN532_MOSI (23) // MOSI (Master Out Slave In)
#define PN532_SS   (18) // SS (Slave Select)

#define Buzzer 14
#define Indicator 13

int size1 = 50;
int size2 = 200;

char details[40];
char detailsLong[250];

WiFiClient espClient;
PubSubClient mqttClient(espClient);

Adafruit_PN532 nfc(PN532_SCK, PN532_MISO, PN532_MOSI, PN532_SS);
String idcard;
String idcardOld;

void setup() {
  Serial.begin(115200);

  setupWifi();

  setupMqtt();

  setupPn532();

  pinMode(Buzzer, OUTPUT);
  digitalWrite(Buzzer, LOW);

  pinMode(Indicator, OUTPUT);
  digitalWrite(Indicator, LOW);
}

void loop() {

  if (!mqttClient.connected()){
    connectToBroker();
  }

  mqttClient.loop();

  readPn532();
}

void setupWifi() {

  WiFi.begin("NTWS-4G", "XXXXXXXXXXX");

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
  }
}
```

```
62
63 void setupMqtt() {
64   mqttClient.setServer("test.mosquitto.org", 1883);
65 }
66
67 void setupPn532(){
68   nfc.begin();
69   uint32_t versiondata = nfc.getFirmwareVersion();
70   if (! versiondata) {
71     while (1); // halt
72   }
73
74   nfc.setPassiveActivationRetries(0xFF);
75 }
76
77 void readPn532(){
78   uint8_t success;
79   uint8_t uid[] = { 0, 0, 0, 0, 0, 0, 0 };  // Buffer to store the
       returned UID
80   uint8_t uidLength;
81   success = nfc.readPassiveTargetID(PN532_MIFARE_ISO14443A, uid, &
       uidLength);
82   if (success) {
83     // Display some basic information about the card
84
85     idcardOld = idcard;
86
87     idcard = "";
88     for (byte i = 0; i <= uidLength - 1; i++) {
89       idcard += (uid[i] < 0x10 ? "0" : "") + String(uid[i], HEX);
90     }
91
92     if (idcard != idcardOld){
93       String(idcard).toCharArray(details, size1);
94       mqttClient.publish("210610H-CardDetails", details);
95       Indicate();
96     }
97   }
98 }
99
100 void connectToBroker(){
101   while (!mqttClient.connected()){
102
103     if (mqttClient.connect("210610H-forklift")){
104       String("RFID based asset tracking device CONNECTED").toCharArray
          (details, size2);
105       mqttClient.publish("210610H-Details", details);
106     }
107     else{
108       delay(5000);
109     }
110   }
111 }
112
```

```
113 void Indicate(){
114   tone(Buzzer, 400);
115   digitalWrite(Indicator, HIGH);
116   delay(400);
117 
118   noTone(Buzzer);
119   digitalWrite(Indicator, LOW);
120 }
```

Listing 4: Arduino Code for pragram ESP32 IC





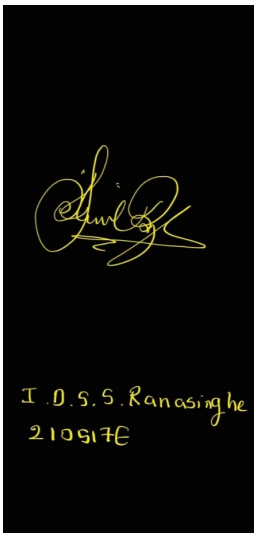


Figure 30: Signs Of Other Members Of Main Group