ESP32

RFID SCANNER

Prepared by

Albert Augustine
Gurusaran AB
Hansika Sayyad
Sri Haran A

RFID522

Problem Statement:

i). Implementation Feasibility:

* Technical:

ESP32 is a versatile microcontroller with built-in Wi-Fi and Bluetooth capabilities. It can easily interface with MFRC522 RFID Modules via SPI

* Library Support:

Libraries such as MFRC522 for RFID. and LiquidCrystal_I2C for LCD.

* Power Supply:

The ESP32 can be powered via USB. or a battery, providing flexibility for various applications.

* Cost Feasibility:

The cost of an ESP32 & MFRC522 RFID Module is relatively low, making it cost-effective

ii) No. of Hardware Components Proposed:

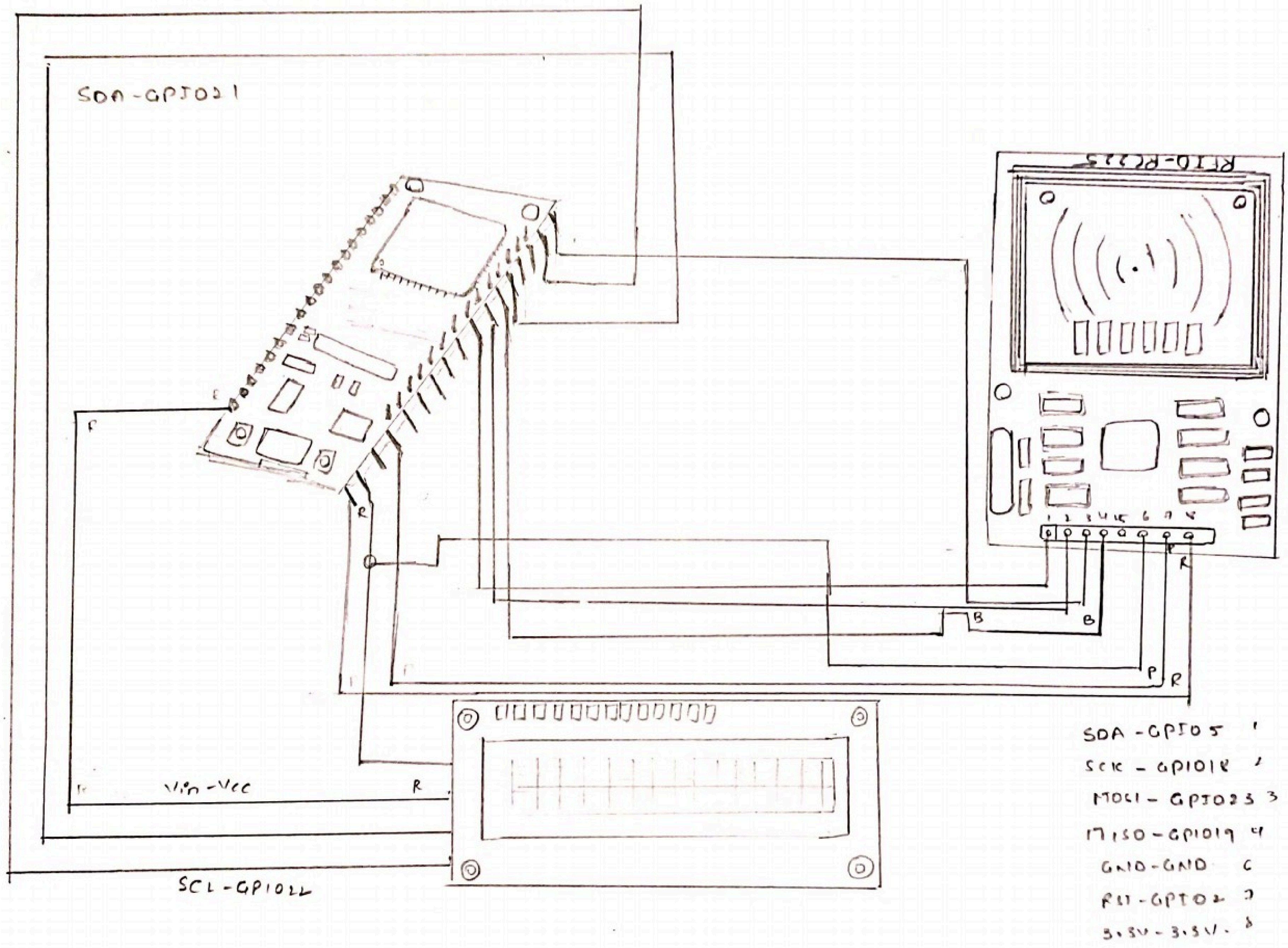
ESP32 Microcontroller	: 1 Unit
MFRC522 RFID	: 1 Unit
LCD Display (I2C)	: 1 Unit
Power Supply	: 1 Unit (USB or battery)
Connecting Wires	: Assorted lengths for connections.
RFID tags/Cards	: 2 Units for testing (Atleast)

iii). Applications Identified:

* Access Control Systems.
* Inventory Management
* Attendance Systems
* Smart Lockers.
* Event Management.

Problem Statement:

This project using an RFID scanner & ESP32 is technically feasible with affordable components and well-documented libraries. The identified applications can significantly streamline various processes such as access control and more. The project is poised for successful implementation within a reasonable timeframe.


SDA-GPIO21
RFID-RC522
Vin-Vcc
SCL-GPIO22
SDA-GPIO5 1
SCK-GPIO18 2
MOSI-GPIO23 3
MISO-GPIO19 4
GND-GND 6
RST-GPIO2 7
3.3V-3.3V. 8

CODE STRUCTURE

```
/*
Wiring:
RFID Reader (SPI):
SDA to Arduino D10
SCK to Arduino D13
MOSI to Arduino D11
MISO to Arduino D12
RST to Arduino D9
GND to Arduino GND
3.3V to Arduino 3.3V
*/

#include <SPI.h>
Communication
#include <MFRC522.h>
#include <Wire.h>
Communication
#include <LiquidCrystal_I2C.h>
LCD

#define SS_PIN 5
#define RST_PIN 2

MFRC522 rfid(SS_PIN, RST_PIN);
LiquidCrystal_I2C lcd(0x27, 16, 2);

void setup() {
    Serial.begin(9600);
    SPI.begin();
    rfid.PCD_Init();
```

```
    lcd.init();
    delay(100);
    lcd.backlight();
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Getting UID Tag");
    delay(2000);
    lcd.clear();
}
void loop() {
  if ( ! rfid.PICC_IsNewCardPresent() ||
        ! rfid.PICC_ReadCardSerial() ) {
      return;
  }
  Serial.print("UID Tag: ");
  String content = "";
  for (byte i = 0; i < rfid.uid.size; i++) {
      Serial.print(rfid.uid.uidByte[i] < 0x10 ? "0 ":"
                                              ");
      Serial.print(rfid.uid.uidByte[i], HEX);
      content.concat(String(rfid.uid.uidByte[i] < 0x10 ?"
                                          0":" "));
      content.concat(String(rfid.uid.uidByte[i], HEX));
  }
  Serial.println();
  Serial.println();
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("RFID UID:");
```

```
    lcd.setCursor(0,1);
    lcd.print(content);
    delay(2000);
    rfid.PICC_HaltA();
}
```

EXCEPTIONS CODE

```
void setup() {
  Serial.begin(9600);
  // Exception 1
  if(!SPI.begin()) {
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("SPI Init Error");
    while(true);
  }
  rfid.PCD_Init();
  // Exception 2
  if(!rfid.PCD_PerformanceSelfTest()) {
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print("RFID Init Error");
      while(true);
  }
  lcd.init();
  delay(100);
  lcd.backlight();
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Getting UID Tag");
```

```
    delay(2000);
    lcd.clear();
}
void loop() {
    // Exception 3
    int attempts = 0;
    while (attempts < 5) {
        if (rfid.PICC_IsNewCardPresent() && rfid.PICC_ReadCardSerial()) {
            break;
        }
        attempts++;
        delay(200);
    }
    if (attempts == 5) {
        lcd.clear();
        lcd.setCursor(0,0);
        lcd.print("No Card Found");
        delay(2000);
        return;
    }
    // Exception 4
    if (!rfid.PICC_ReadCardSerial()) {
        lcd.clear();
        lcd.setCursor(0,0);
        lcd.print("Read UID Error");
        delay(2000);
        return;
    }
```

```
// Exception 5
MFRC 5233 : : PICC_Tyhe . bicc Tyhe = rfid. PICC_ GetTyhe
                                    (rfid.uid. sah );
if ( bicc Tyhe .== MFRC 522 :: PICC_ Tyhe - UNKNOWN ){
    lcd. clear ();
    lcd . set Cursor (0, 0);
    lcd .  hrint ("unknown Card");
    delay (2000);
    return ;
}

Serial . hrint ("UID tag :  ") ;
String Content = " ";
for ( byte i=0 ; i < rfid .uid . Size ; i++ ) {
    Serial . hrint ( rfid .uid . ~~Size ; i++~~)
                                uidByte [i] <0x10 ? "0":
                                " ");
    Serial . hrint (rfid . uid . uidByte [i], HEx);
    Content . concat ( String (rfid . uid . uidByte [i] <0x10 ? "0":
                                " "));
    content . concat ( String ( rfid . uid . uidByte [i], HEx ));
}
Serial . hrintln ();
Serial . hrintln ();
lcd . clear ();
lcd . set Cursor (0. 0);
lcd . hrint ( "Rfid UID : ");
lcd . set Cursor ( 0, 1);
lcd . hrint (content);
delay (2000);
rfid . PICC_ HaltA ( ); }
```

Exception Handling:

Exception 1: SPI Initialization Failure.

The (Serial Peripheral Interface) is used for communication between the Arduino and the RFID reader. If the SPI initialization fails, it could indicate a hardware or wiring issue, preventing any further communication with the RFID module. The LCD displays "SPI Init Error" and the program halts to signal that the RFID setup process cannot continue.

Exception 2: RFID Reader Initialization Failure:

This checks whether the RFID reader initializes correctly by performing a self-test. A failed self-test could mean that the RFID reader is malfunctioning or not connected properly and communication with RFID tags wont be possible.

Exception 3: No Card Detected After Several Tries:

If no RFID card is detected after several attempts, it might be due to a card being out of range or absent, or interference in communication. After five attempts without success, the LCD displays "No Card Found" for a brief period, informing the user that card scan attempt was unsuccessful.

Exception 4: Card UID Read Error:

Once a card is detected, the program attempts to read its UID (Unique Identifier). If the UID read fails, this might indicate that the card is corrupted or that communication was interrupted. When this happen, the LCD displays "Read UID Error" for two seconds before returning to detect a new card.

Exception 5: Unknown Card Type:

Each RFID card has a type that the reader can identify. If the card type cannot be determined this exception occurs. The LCD displays "Unknown Card" to indicate that the RFID card is not recoguized, and the program briefly pauses before scanning for another card.

Challenges faced :-

1. Initial setup and configuration.

Problem :- The initial setup of the hardware components was challenging. There was confusion regarding the correct wiring of the RFID module to the Esp32.

Solution :- A clear wiring diagram was created, detailing the connections of the RFID reader (SPI) to the Esp32. The importance of connecting power and ground correctly was emphasized.

2. LCD Display issues.

Problem :- The LCD display showed a full yellow screen with no text output.

Solution :- After investigating the wiring and I2C address, it was confirmed that the I2C address was incorrectly set. Using an I2C scanner, the correct address was identified, and the initialization code for the LCD was updated accordingly.

3. RFID Tag Reading failures.

Problem :- Initially, the system failed to detect RFID tags, consistently returning 0x00 or 0xFF as output.

Solution :- Several steps were taken to troubleshoot this issue :

1. Wrong inspection :- A thorough check of the wiring was conducted to ensure proper connections, particularly for the SPI pins.

2. Power supply verification :- Ensured that the RFID module received a stable 3.3v supply.

3. SPI configuration :- Explicitly defined SPI settings in the code and reduced the SPI clock speed to enhance communication stability.

4. Minimal test code :- Implemented a simplified version of the code focused solely on RFID functionality to isolate the problem.

4. Communication stability :-

Problem :- Intermittent failures in communication between the Esp32 and the RFID module were observed.

Solution :- Adjustments were made to the SPI settings, including setting SPI mode and clock speed. Adding a delay between scans improved reliability and allowed the system to stabilize before the next read.

5. Debugging output issues :-

Problem :- Lack of clear debugging output made it difficult to identify the root cause of the issues.

Solution :- Implemented comprehensive serial outputs throughout the code to track the program's flow and identify where failures occured.