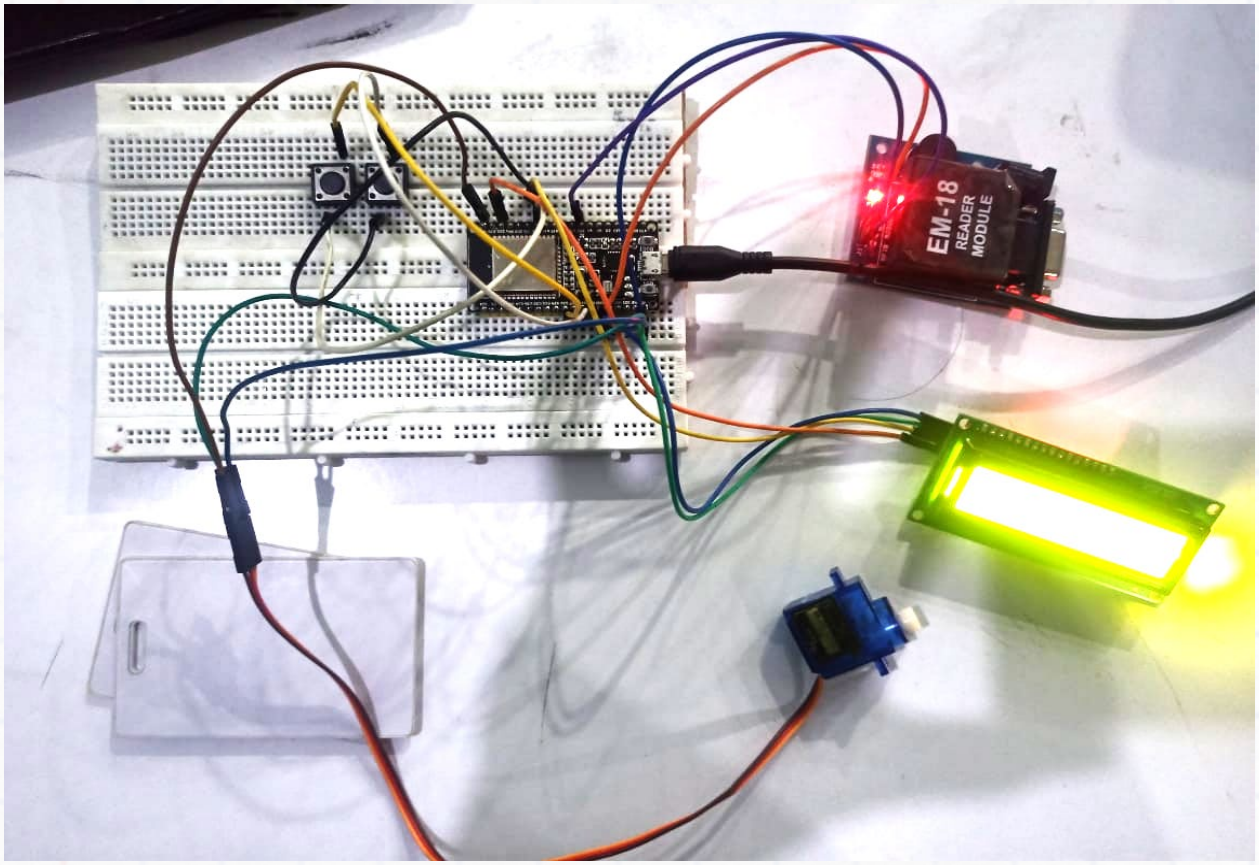
Build a RFID Fast Tag Reader



Index

Aim

Build an RFID Fast Tag Reader that registers RFID card data, processes it, creates actions, and publishes the data to an online dashboard.

Concept

In this project, we mimic the working of a fast tag reader that reads RFID card data and operates the boom barrier in toll gates. Here we read RFID cards, verify them with the registry in the memory, open/close the gate, and finally send the data to the online dashboard.

Components

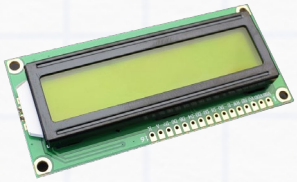
1. ESP 32 Module
2. RFID Card Reader
3. GSM Module
4. Servo Motor
5. 16*2 LCD Display
6. Jumper wires
7. USB cables
8. USB to TTL converter (Component to execute extra activities in the project. Procure externally if not available in the kit, CP2102(6-pin) USB 2.0 to TTL UART serial converter)
9. Push button
10. 12V Power Adaptor

Esp 32 Module

RFID Reader

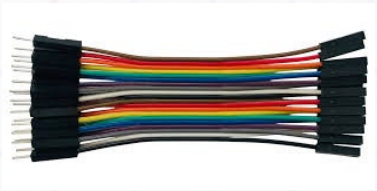
12V Power adapter

Servo Motor



16*2 LCD Display

Push button



Jumper Wires

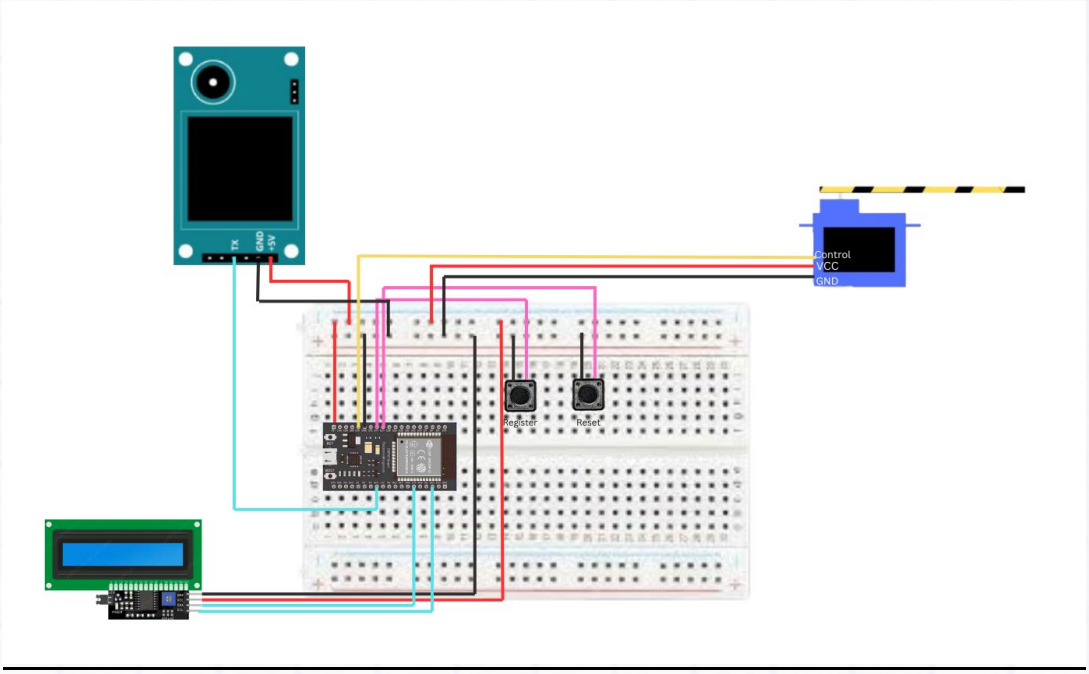
USB

Breadboard

Connections

Circuit Diagram

NOTE: Before starting the connections, verify that all the jumper wires are working using a multimeter. Also, ensure that the connections are strong, else the setup may not work.



Detailed Connection Steps

LCD with I2c Module:

- I2c GND to ESP32 GND
- I2c VCC to ESP32 V5
- I2c SDA to ESP32 G21
- I2c SCL to ESP32 G22

Servo

- Servo Out () to ESP32 G13
- Servo 5V to ESP32 V5
- Servo GND to ESP32 GND

Push button

Button1 (Register button):

- Out to ESP32 G14
- GND to ESP32 GND

Button2 (Reset button):

- Out to ESP32 G27
- GND to ESP32 GND

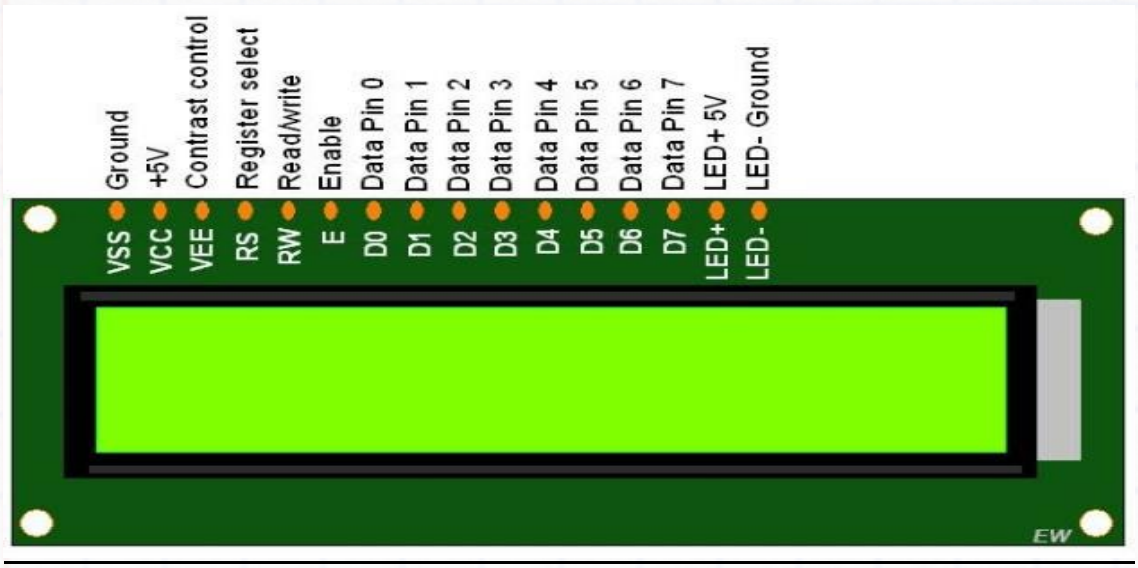
EM-18 Reader Module

- TX to ESP32 G16
- 5v to ESP32 V5

- GND to ESP32 GND

Part 1: Experimenting with the components

1. 16*2 LCD Display



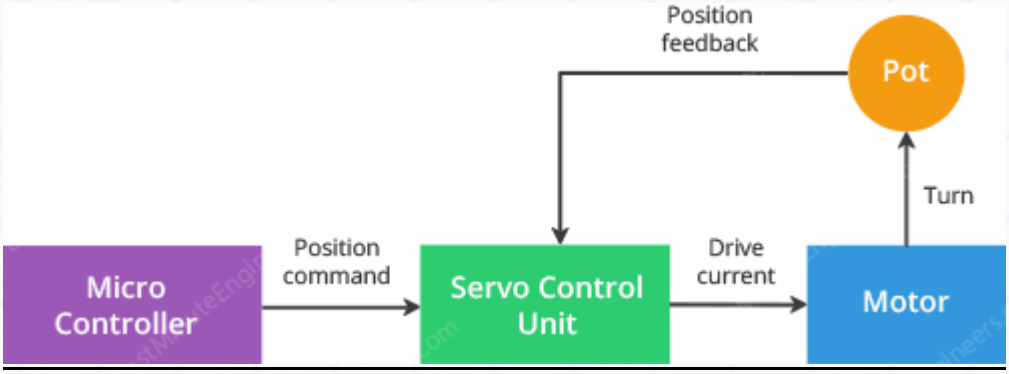
LCD (Liquid Crystal Display) is a flat panel display that uses liquid crystals to display information. It consists of an array of small segments called pixels, which can be manipulated to display status or parameters.

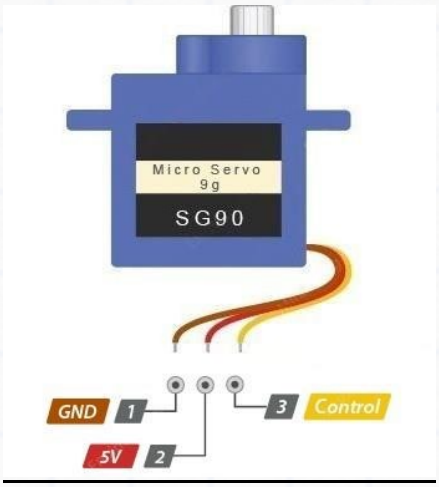
A 16*2 LCD can display 16 characters or numbers column-wise and 2 in row-wise.

2. DC Servo motor

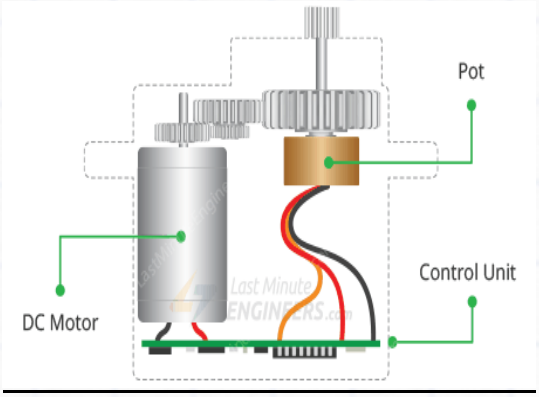
A servo system primarily consists of three basic components: a controlled device, an output sensor, and a feedback system. This is an automatic closed loop control system. Here instead of controlling a device by applying the variable input signal, the device is controlled by a feedback signal generated by comparing output signal and reference input signal. Any electrical motor can be utilised as a servo motor if it is controlled by servomechanism.

In the DC servo motor, a small DC motor connected to the output shaft through the gears. The output shaft drives a servo arm and is also connected to a potentiometer (pot). The potentiometer provides position feedback to the servo control unit where the current position of the motor is compared to the target position. According to the error, the control unit corrects the actual position of the motor so that it matches the target position





DC Servo motor wiring connection



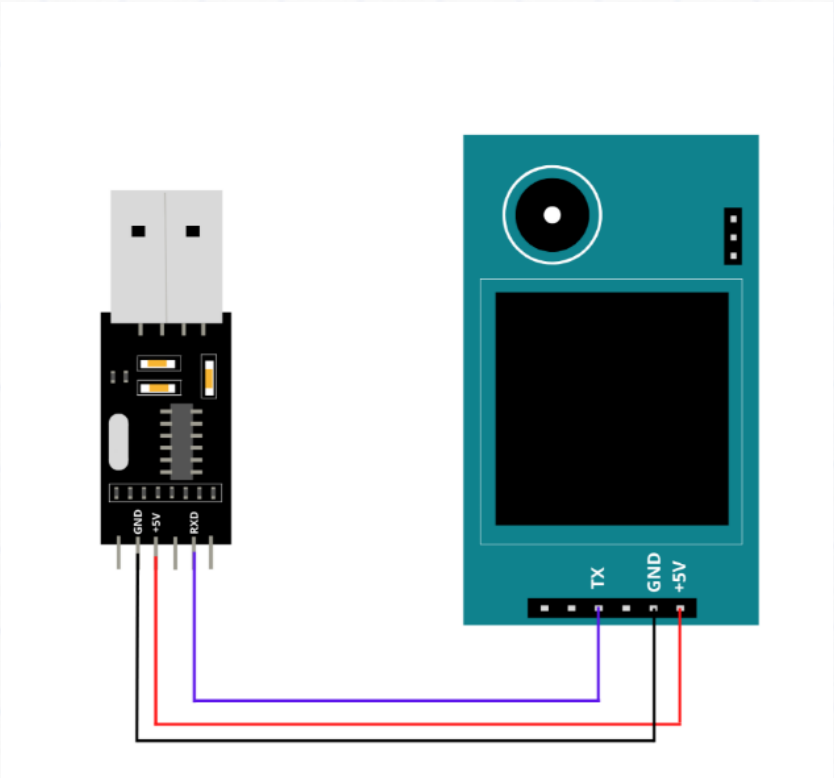
DC Servo motor Internal Illustration

3) RFID Card reader Module

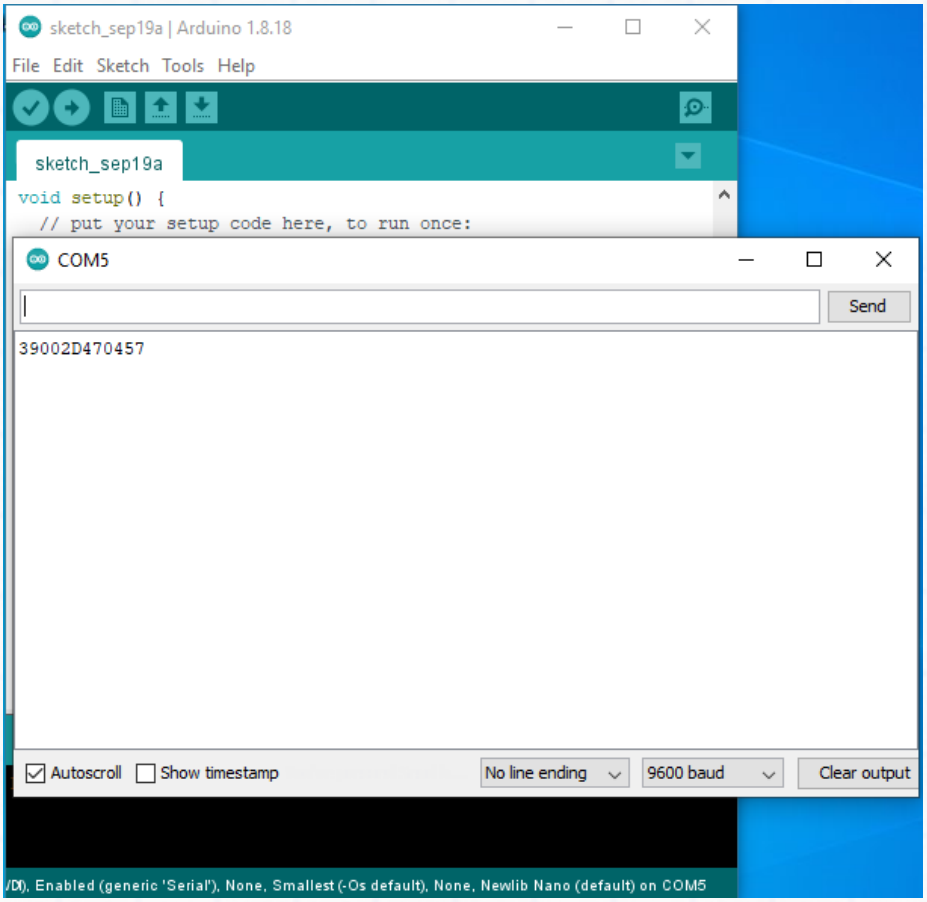
A radio frequency identification reader (RFID reader) module is a device used to gather information from an RFID tag, which is used to track individual tags/cards. Radio waves are used to transfer data from the tag to a reader. RFID is a technology similar to bar codes.

Extra Activity– Reading RFID values

1. Connect a USB to TTL converter to RFID card reader as Below:



2. Show a card to the reader and check the 12 digit RFID codeappearing on Serial Monitor



Note: Each RFID cards will have unique values

Part 2: Implementing the Fast tag reader project

Code:

“Open the .ino file which was attached in the folder and write a code by the below instruction.”

Explanation:

Block 1: Including Libraries and Defining Constants

- **Include Libraries**: Includes necessary libraries for LCD, Servo, EEPROM, Wi-Fi, and HTTP functionality.
- **Define Pins**: Defines the pins for RFID, Servo, and buttons.

- **LCD and Servo Initialization**: Initializes the LCD and Servo objects.
- **Card Management**: Sets up constants and variables for managing registered cards and tracking entries.
- **Wi-Fi and ThingSpeak Settings**: Defines Wi-Fi credentials and ThingSpeak API key and host.

```
#include <Wire.h>                  // Include Wire library for I2C communication
#include <LiquidCrystal_I2C.h>        // Include library for I2C LCD
#include <ESP32Servo.h>              // Include library for controlling servos
#include <EEPROM.h>                 // Include library for reading and writing to EEPROM
#include <WiFi.h>                  // Include library for WiFi functionality
#include <HTTPClient.h>              // Include library for making HTTP requests

#define RFID_RX_PIN 16  // Define pin for RFID RX2 pin on ESP32
#define SERVO_PIN 13    // Define pin for servo motor
#define REGISTER_BUTTON_PIN 14 // Define pin for the register button
#define ERASE_BUTTON_PIN 27    // Define pin for the erase button

LiquidCrystal_I2C lcd(0x27, 16, 2);  // Initialize LCD with address 0x27 and 16x2 size
Servo doorServo;                // Create a Servo object to control the door

const int MAX_CARDS = 10;         // Maximum number of cards that can be registered
String registeredCards[MAX_CARDS];  // Array to store registered card IDs
int cardCount = 0;               // Counter for the number of registered cards
int totalEntries = 0;            // Counter for total entries
int accessGranted = 0;           // Counter for granted accesses
int accessDenied = 0;            // Counter for denied accesses

// WiFi settings
const char* ssid = "YOUR_WIFI_SSID";      // WiFi SSID
const char* password = "YOUR_WIFI_PASSWORD"; // WiFi password

// ThingSpeak settings
const char* thingSpeakApiKey = "YOUR_THINKSPEAK_WRITE_API_KEY"; // ThingSpeak
API key
const char* thingSpeakHost = "api.thingspeak.com";           // ThingSpeak host
```

Explanation:

- **Serial Communication**: Initializes serial communication for debugging and RFID reader.
- **LCD Initialization**: Sets up the LCD, displays initialization message.
- **Servo Initialization**: Attaches and sets the initial position of the servo.
- **Pin Mode Setup**: Configures the button pins as inputs with pull-up resistors.
- **EEPROM Initialization**: Initializes EEPROM and loads stored cards.
- **Wi-Fi Connection**: Connects to Wi-Fi.
- **System Ready**: Displays "System Ready" on LCD and updates ThingSpeak.

Copy the below code that is inside in the void setup and paste it on your code.

```
void setup() {
  Serial.begin(9600);               // Initialize serial communication at 9600 baud rate
  Serial2.begin(9600, SERIAL_8N1, RFID_RX_PIN, -1);  // Initialize Serial2 for RFID communication

  lcd.init();                       // Initialize the LCD
  lcd.backlight();                  // Turn on the LCD backlight
  lcd.clear();                      // Clear the LCD display
  lcd.print("RFID System");         // Print "RFID System" on the LCD
  lcd.setCursor(0, 1);              // Move cursor to the second line
  lcd.print("Initializing...");     // Print "Initializing..." on the LCD

  doorServo.attach(SERVO_PIN);      // Attach the servo motor to the defined pin
  doorServo.write(0);               // Set the servo motor to 0 degrees (closed position)

  pinMode(REGISTER_BUTTON_PIN, INPUT_PULLUP); // Set register button pin as input with pull-up resistor
  pinMode(ERASE_BUTTON_PIN, INPUT_PULLUP);    // Set erase button pin as input with pull-up resistor

  EEPROM.begin(512);                // Initialize EEPROM with 512 bytes of storage
  loadCardsFromEEPROM();            // Load registered cards from EEPROM

  connectWiFi();                    // Connect to WiFi network

  delay(2000);                      // Wait for 2 seconds
  lcd.clear();                      // Clear the LCD display
```

```
 lcd.print("System Ready");          // Print "System Ready" on the LCD

 updateThingSpeak();                // Update ThingSpeak with initial data
}
```

Block 3: Main Loop

Explanation

- **Button Checks:** Checks if the register or erase button is pressed and calls the corresponding function.
- **RFID Reading**: Reads the card ID from the RFID reader and trims any whitespace.
- **Entry Tracking**: Increments the total entry count and checks if the card is registered.
- **Access Control**: Grants or denies access based on card registration and updates ThingSpeak with the new data.

Copy the below code which is inside in the void loop and paste it on your code.

```
void loop() {
 if (digitalRead(REGISTER_BUTTON_PIN) == LOW) { // Check if register button is
pressed
  registerCard();               // Call function to register a new card
 }

 if (digitalRead(ERASE_BUTTON_PIN) == LOW) {   // Check if erase button is pressed
  eraseAllCards();              // Call function to erase all registered cards
 }

 if (Serial2.available()) {                 // Check if RFID data is available
  String cardID = Serial2.readStringUntil('\n'); // Read the card ID from the RFID reader
  cardID.trim();                          // Remove any whitespace from the card ID

  totalEntries++;                         // Increment the total entries counter

  if (isCardRegistered(cardID)) {          // Check if the card is registered
```

```
    grantAccess();              // Call function to grant access
    accessGranted++;            // Increment the access granted counter
   } else {
    denyAccess();               // Call function to deny access
    accessDenied++;             // Increment the access denied counter
   }

   updateThingSpeak();                  // Update ThingSpeak with the latest data
  }
}
```

Block 4: Card Registration and Erasure

Explanation

- **Register Card**: Handles the process of registering a new card, including displaying messages on the LCD and updating EEPROM and ThingSpeak.
- **Erase All Cards**: Clears all registered cards, updates EEPROM, and displays messages on the LCD.
- **Check Card Registration**: Checks if a card is already registered.
- **Grant/Deny Access**: Controls the servo to grant or deny access and updates the LCD with the appropriate message.

Copy the blow code which is inside in the registerCard(),eraseAllCards(),void grantAccess(),denyAccess() functions and paste it on your functions.

```
void registerCard() {
 lcd.clear();                   // Clear the LCD display
 lcd.print("Scan to register");       // Print "Scan to register" on the LCD

 while (!Serial2.available()) {       // Wait for RFID data to be available
  delay(100);                   // Delay for 100 milliseconds
 }

 String cardID = Serial2.readStringUntil('\n'); // Read the card ID from the RFID reader
 cardID.trim();                 // Remove any whitespace from the card ID
```

```
  if (cardCount < MAX_CARDS) {          // Check if the card count is less than the
maximum allowed
    registeredCards[cardCount++] = cardID; // Add the card ID to the registered cards
array
    saveCardsToEEPROM();              // Save the registered cards to EEPROM

    lcd.clear();                      // Clear the LCD display
    lcd.print("Card registered");       // Print "Card registered" on the LCD
    lcd.setCursor(0, 1);              // Move cursor to the second line
    lcd.print(cardID);                // Print the card ID on the LCD
    updateThingSpeak();               // Update ThingSpeak with the latest data
  } else {
    lcd.clear();                      // Clear the LCD display
    lcd.print("Max cards reached");      // Print "Max cards reached" on the LCD
  }

  delay(2000);                        // Wait for 2 seconds
  lcd.clear();                        // Clear the LCD display
  lcd.print("System Ready");          // Print "System Ready" on the LCD
}

void eraseAllCards() {
  cardCount = 0;                      // Reset the card count to 0
  saveCardsToEEPROM();                // Save the empty card list to EEPROM

  lcd.clear();                        // Clear the LCD display
  lcd.print("All cards erased");       // Print "All cards erased" on the LCD
  updateThingSpeak();                 // Update ThingSpeak with the latest data
  delay(2000);                        // Wait for 2 seconds
  lcd.clear();                        // Clear the LCD display
  lcd.print("System Ready");          // Print "System Ready" on the LCD
}

bool isCardRegistered(String cardID) {
  for (int i = 0; i < cardCount; i++) {  // Loop through all registered cards
    if (registeredCards[i] == cardID) {  // Check if the card ID matches any registered card
      return true;                    // Return true if card is found
    }
  }
  return false;                       // Return false if card is not found
```

```
}

void grantAccess() {
  lcd.clear();                    // Clear the LCD display
  lcd.print("Access Granted");          // Print "Access Granted" on the LCD
  doorServo.write(90);                // Open the door by setting the servo to 90 degrees
  delay(3000);                    // Wait for 3 seconds
  doorServo.write(0);                 // Close the door by setting the servo to 0 degrees
  lcd.clear();                    // Clear the LCD display
  lcd.print("System Ready");           // Print "System Ready" on the LCD
}

void denyAccess() {
  lcd.clear();                    // Clear the LCD display
  lcd.print("Access Denied");          // Print "Access Denied" on the LCD
  delay(2000);                    // Wait for 2 seconds
  lcd.clear();                    // Clear the LCD display
  lcd.print("System Ready");           // Print "System Ready" on the LCD
}
```

Block 5: EEPROM and Wi-Fi Functions

Explanation

- **EEPROM Functions**: Handles reading and writing card data to and from EEPROM.
- **Wi-Fi Connection**: Manages connecting to Wi-Fi and displaying the status on the LCD.
- **Update ThingSpeak**: Sends the latest data to ThingSpeak and handles the response, including error handling and updating the LCD.

Copy the below code that is inside in the saveCardsToEEPROM(),loadCardsFromEEPROM(),writeStringToEEPROM(),readStringFromEEPROM(),connectWiFi(),updateThingSpeak() function and paste it on your respective function.

```
void saveCardsToEEPROM() {
  for (int i = 0; i < MAX_CARDS; i++) {   // Loop through all possible card slots
    if (i < cardCount) {                  // Check if the slot is within the card count
      writeStringToEEPROM(i * 50, registeredCards[i]); // Write the card ID to EEPROM
    } else {
      writeStringToEEPROM(i * 50, "");    // Write an empty string to EEPROM
    }
  }
  EEPROM.commit();                        // Commit the changes to EEPROM
}

void loadCardsFromEEPROM() {
  cardCount = 0;                          // Reset the card count to 0
  for (int i = 0; i < MAX_CARDS; i++) {   // Loop through all possible card slots
    String card = readStringFromEEPROM(i * 50); // Read the card ID from EEPROM
    if (card.length() > 0) {              // Check if the card ID is not empty
      registeredCards[cardCount++] = card; // Add the card ID to the registered cards
array
    }
  }
}

void writeStringToEEPROM(int addr, String data) {
  int len = data.length();                // Get the length of the data string
  EEPROM.write(addr, len);                // Write the length of the data string to EEPROM
  for (int i = 0; i < len; i++) {         // Loop through each character in the data string
    EEPROM.write(addr + 1 + i, data[i]);  // Write each character to EEPROM
  }
}

String readStringFromEEPROM(int addr) {
  int len = EEPROM.read(addr);            // Read the length of the data string from
EEPROM
  String data = "";                       // Initialize an empty string to store the data
  for (int i = 0; i < len; i++) {         // Loop through each character in the data string
    data += char(EEPROM.read(addr + 1 + i)); // Read each character from EEPROM and
add to string
  }
  return data;                            // Return the data string
}
```

```
void connectWiFi() {
  WiFi.begin(ssid, password);           // Start WiFi connection with SSID and password
  lcd.clear();                          // Clear the LCD display
  lcd.print("Connecting WiFi");         // Print "Connecting WiFi" on the LCD

  while (WiFi.status() != WL_CONNECTED) { // Wait until WiFi is connected
    delay(1000);                        // Delay for 1 second
    lcd.print(".");                     // Print a dot on the LCD for progress indication
  }

  lcd.clear();                          // Clear the LCD display
  lcd.print("WiFi Connected");          // Print "WiFi Connected" on the LCD
  delay(1000);                          // Wait for 1 second
}

void updateThingSpeak() {
  if (WiFi.status() != WL_CONNECTED) {    // Check if WiFi is disconnected
    Serial.println("WiFi disconnected. Attempting to reconnect..."); // Print message to
serial
    connectWiFi();                      // Attempt to reconnect to WiFi
    return;                             // Exit the function
  }

  HTTPClient http;                      // Create an HTTPClient object
  String url = "http://api.thingspeak.com/update?api_key=" + String(thingSpeakApiKey);
// Build ThingSpeak URL
  url += "&field1=" + String(cardCount);  // Add card count to URL
  url += "&field2=" + String(totalEntries); // Add total entries to URL
  url += "&field3=" + String(accessGranted); // Add access granted count to URL
  url += "&field4=" + String(accessDenied);  // Add access denied count to URL

  http.begin(url);                      // Initialize HTTP request with URL
  int httpResponseCode = http.GET();     // Send HTTP GET request and get response
code

  if (httpResponseCode > 0) {           // Check if the response code is positive
    String response = http.getString();   // Get the response string from ThingSpeak
    Serial.println("HTTP Response code: " + String(httpResponseCode)); // Print response
code
    Serial.println("ThingSpeak response: " + response); // Print ThingSpeak response
    lcd.clear();                        // Clear the LCD display
```

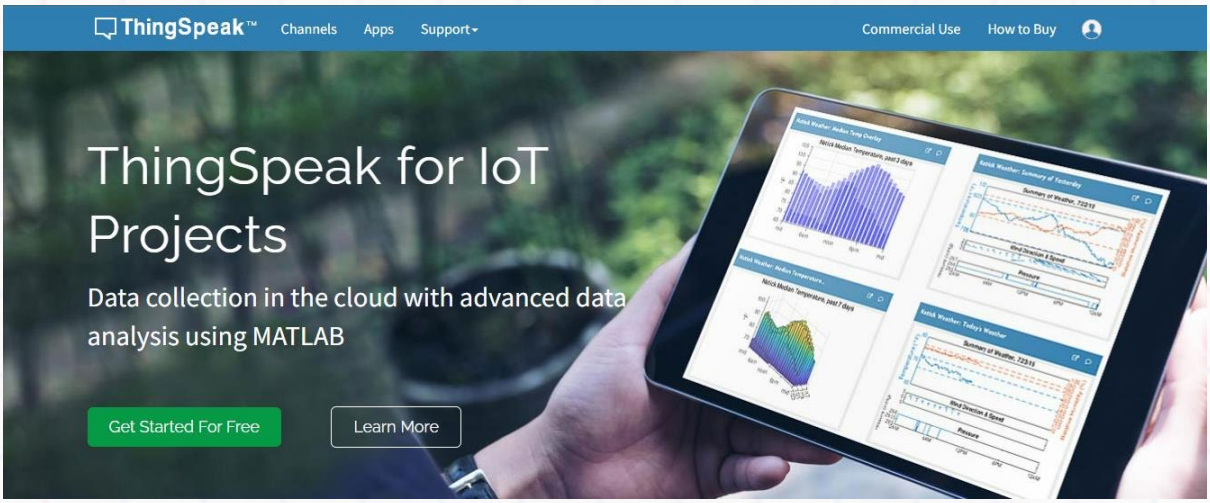
```
  lcd.print("Update success");        // Print "Update success" on the LCD
 } else {
  Serial.print("Error on sending GET: "); // Print error message to serial
  Serial.println(httpResponseCode);     // Print response code to serial
  lcd.clear();                    // Clear the LCD display
  lcd.print("Update failed");         // Print "Update failed" on the LCD
 }

 http.end();                      // End the HTTP request
 delay(2000);                      // Wait for 2 seconds
 lcd.clear();                    // Clear the LCD display
 lcd.print("System Ready");           // Print "System Ready" on the LCD
}
```

After completing these all the steps verify and upload a code to the esp32.

Setting up web dashboard – thingspeak

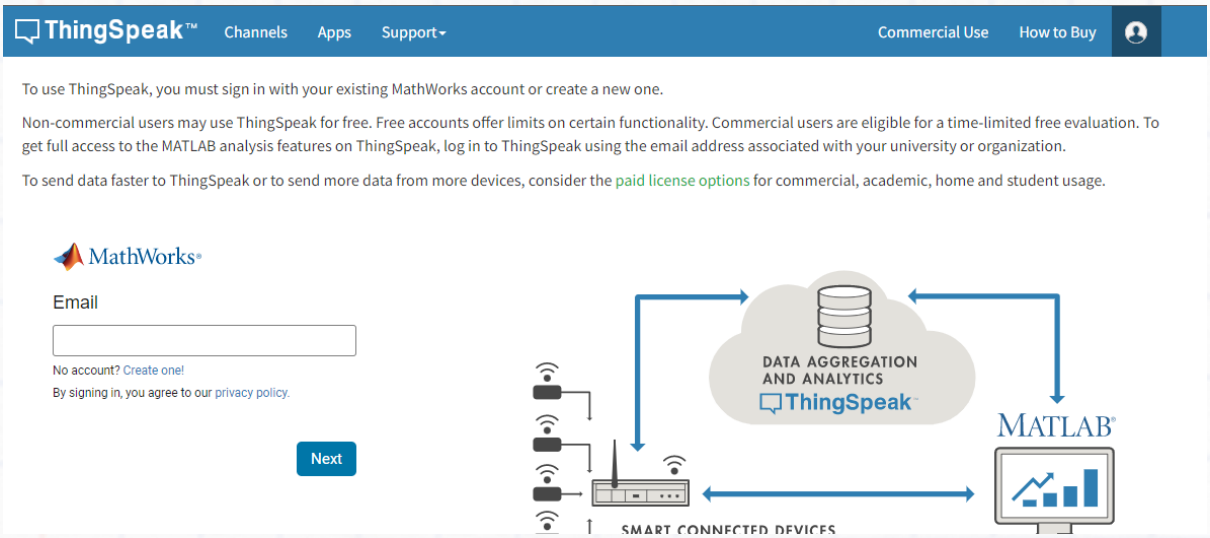
In previous projects we had used ubidots web platform, this time we will try ThingSpeak. ThingSpeak is an IoT analytics platform service that allows you to aggregate, visualize, and analyse live
data streams in the cloud.



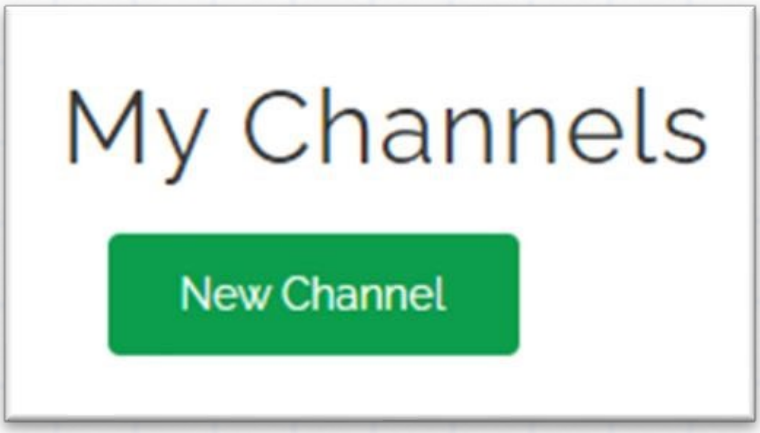
Configuration steps

1. Visit https://thingspeak.com/ website.

2. Create an account



3. Create a new channel in 'My channels'



4. Name the channel as you may prefer like RFID. Tick the fields Field1,Field2,Field3,Field4 since we intend to fill the fields with following data:

- Field1=No of ID's registered
- Field2= Total no. of times card entries
- Field3 = Total no. of access granted

- Field4 = Total no. of access denied

5. Now we have to link this ThingSpeak dashboard with our ESP32 code.

6. Go to '**API keys**' section in RFID channel

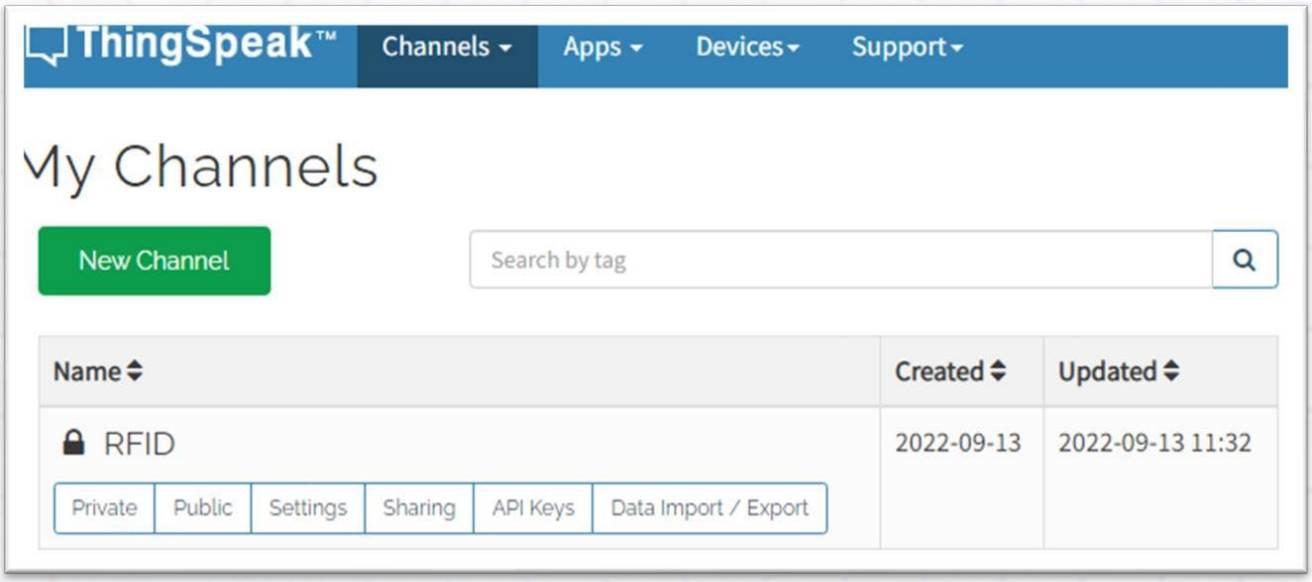
7.Copy the API key and paste it on your code.

```
const char* thingSpeakApiKey = "YOUR_THINKSPEAK_WRITE_API_KEY";
```

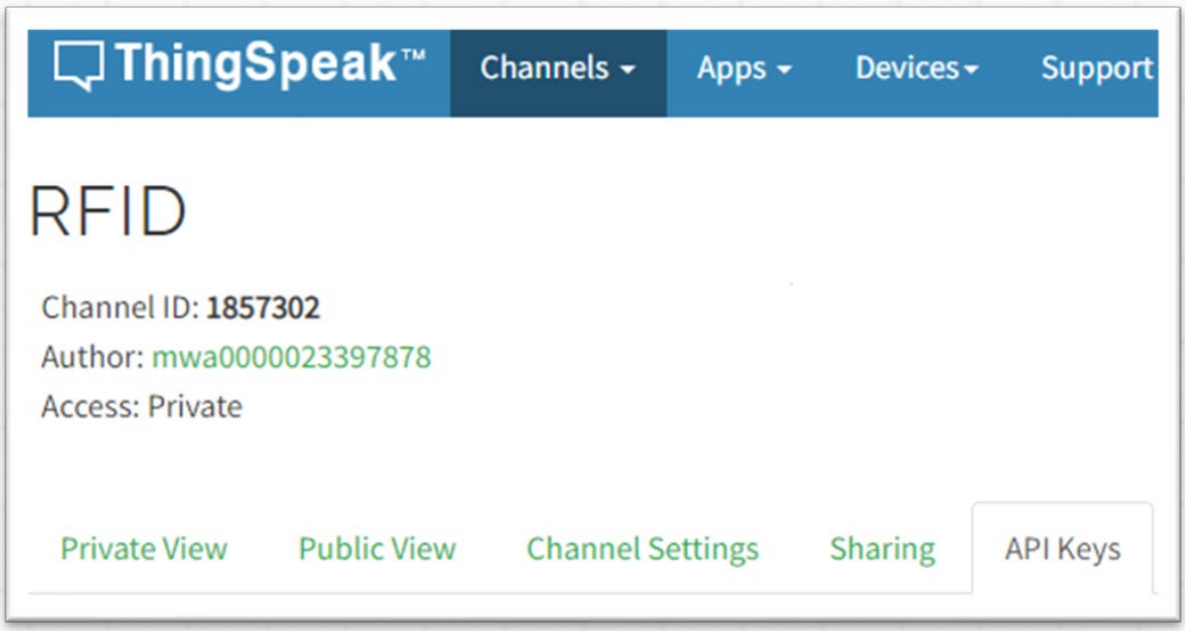
8. Run the code in ESP32 Module

Note: The code run on ESP32 will automatically fill the 4 fields with data

9. Now go to your 'RFID' channel dashboard



10.Click RFID channel and navigate to 'private view'

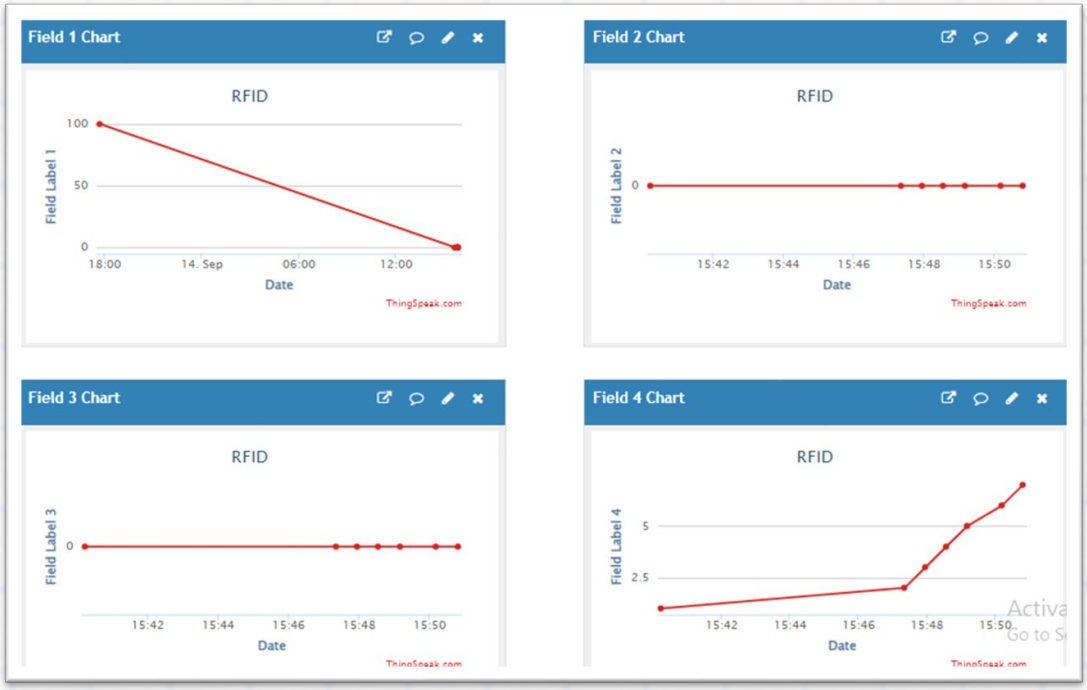


11.Now it's time to receive data to the web dashboard.

12. Check whether all other components like RFID reader, LCD display servo motor, ESP32 Module are powered up properly.

13.Try using different RFID cards, register a card. Check whether the servomotor is working, and entry is permitted. Try with non-registered cards, check whether entry is denied.

14.Now visit the ThingSpeak dashboard. You will see the fields are getting populated.



Task

1. After erasing the current memory. Take 3 new RFID cards, add 2 cards. Keep one card unregistered. Check how the device is behaving with the 3 cards. Check whether the data is received on the Web dashboard.

2. Make a video presenting the build of RFID reader, how you completed the activities and challenges. Share it with the Build Club Community on the Discord Server!