

# Project - PAWTAL

## AMC SS2025

Deiona Abraham (34055) - Florentina Stroissnigg (33724) - Mhwae Mhwae Kyaing Kyaing (33186)

### Introduction

by Mhwae Mhwae Kyaing Kyaing

Cat owners often face the daily challenge of their pets meowing or scratching at the door only to go outside for a few minutes before wanting to come back in. While cats are famously independent and occasionally mischievous, their quirky behavior is part of what makes them so lovable. To improve the coexistence between cats and their owners, Project PAWTAL developed an automatic flap door system designed for quick setup. PAWTAL allows cats to move freely while keeping their owners informed of their whereabouts in real time by notifying via email.

### Materials & Methods

#### Materials

by Florentina Stroissnigg

For this project an ESP32 S3 Dev Module Microcontroller was used. Additionally, a RC522 RFID Sensor with the tag and a PIR motion sensor enabled the detection of the cat, while two Micro Servo Motor SG90 serve as locks to keep the cat door closed. Their functions are broken down below:

- The ESP 32 S3 Dev Module is a microcontroller with Wi-Fi and Bluetooth capabilities with multiple GPIOs (General-Purpose Input/Outputs).
- The RC522 is a RFID (Radio Frequency Identification) Sensor using 3.3V. It is a wireless technology that uses radio waves to identify and track objects, animals, or people. To communicate with the micro controller the RC522 uses the Serial Peripheral Interface (SPI) communication protocol.
- The PIR motion sensor uses 3.3V. It works by detecting a change in infrared levels within its field of view. To communicate with the microcontroller, the sensor uses a single digital pin, that sends a HIGH signal when motion is detected
- The SG90 Servo Motor are small actuators using 5V provided by the ESP32's V output pin. Since the motor only acts as a lock, and there is no significant mechanical load acting on it.
- Jumper wires to connect the sensors and actuators to the microcontroller
- Power source connection to provide the prototype with 5V
- cat door

## Methods

by Deiona Abraham

The software for this project was developed using Arduino IDE to control the hardware components. MQTT, HiveMQ, and Node-RED were used to establish a client-based network to enable email notifications about the cat's whereabouts.

- Arduino IDE is an open-source Integrated Development Environment used to write, develop, compile and upload the code to microcontrollers such as the ESP32-S3 Dev Module. It supports C/C++ based programming and offers a wide range of libraries.

Libraries needed for Pawtal:

- ESP32 Board Package → used to develop codes for ESP32 boards in Arduino IDE
  - SPI.h - used for the SPI communication between ESP32 and the RC522 RFID Sensor
  - MFRC522.h → used for the interface with the RC522 RFID module and to read RFID tags
  - ESP32Servo.h → used to control the SG90 micro servo motors via PWM
  - WiFi.h → Connects the ESP32 to a Wi-Fi network for wireless communication
  - MQTTClient.h → Enables MQTT protocol functions to publish and subscribe to messages between devices
  - WiFiClientSecure.h → Allows secure (TLS/SSL) communication over Wi-Fi, required for connecting to brokers like HiveMQ over HTTPS
- MQTT (Message, Queue, Telemetry, Transport) is a publish-subscribe protocol commonly used in IoT (internet of things) systems. It is a reliable and efficient method to establish communication between machines and devices, eliminating the need for wire to wire connection, while still allowing them to communicate indirectly.
    - Clients, members of a network, can be defined as a data source or sender who publishes data to the network. In our project, our sensors mentioned above (RC522 and PIR) serve as the data publishers HiveMQ, our broker serves as an intermediate who then distributes the data received to other clients. Likewise, clients can subscribe to the network and receive the data. In our project, once the data was sent to HiveMQ, and received by NodeRED.
  - HiveMQ: A broker that establishes a bond between the different clients (subscriber or publisher) as it receives from the publisher, then filters and distributes the message to the subscriber allowing bi-directional data movement.
  - NodeRED: is a visual programming tool that allows the user to see the flow of messages.

Libraries needed for Pawtal:

## Results

### Hardware

by Florentina Stroissnigg

To create the prototype the actuators and sensors were connected to the board via breadboard and jumper wires as described in figure 1. Then the breadboard, sensors and actuators were positioned and glued on a cardboard box to test the functionality of the prototype. This first prototype can be seen in image 1. To test the functionality even further, a second was developed using a cat door, another cardboard box and the components, which can be seen in image 2.

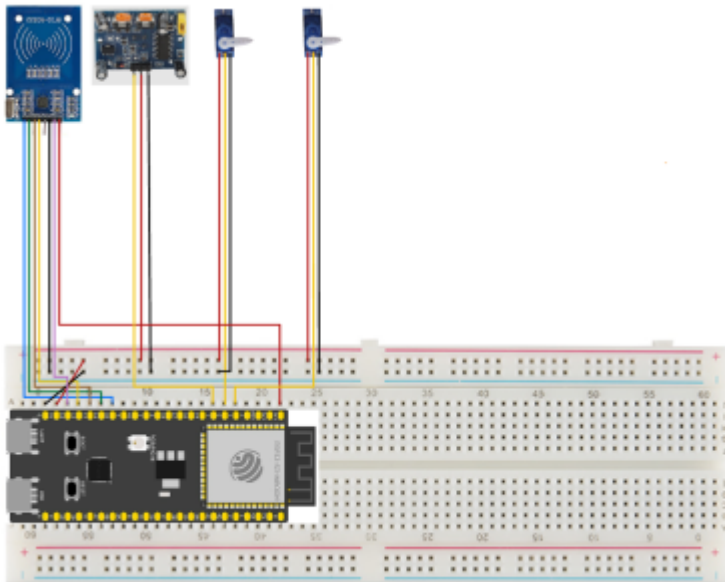


Fig. 1: Wiring model



Fig. 2: Prototype 1.0

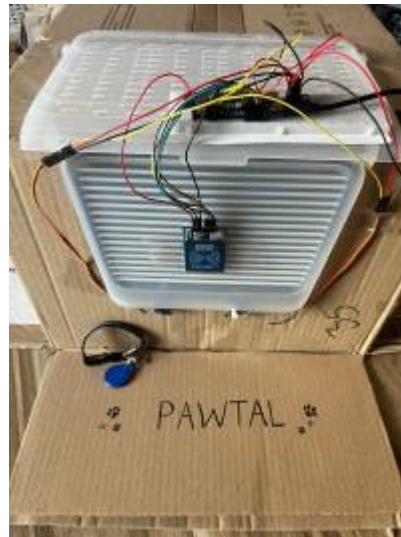


Fig. 3: Prototype 2.0

## Software

by Florentina Stroissnigg and Deiona Abraham

The code was developed using the Arduino IDE. Step one of the coding process involved importing and including all necessary libraries. For better readability and maintainability, all programmable pins connected to actuators and sensors were defined as local constants at the beginning of the code.

```
#include <SPI.h>
#include <MFRC522.h>
#include <ESP32Servo.h>
#include <WiFi.h>
#include <MQTTClient.h>
#include <WiFiClientSecure.h> // For TLS/SSL

// RC522 (RFID) SPI Pins
#define SDA_PIN    10
#define SCK_PIN    11
#define MOSI_PIN   12
```

```
#define MISO_PIN 13
#define RST_PIN 14

// Servo Pin
#define SERVO_PIN 6
#define SERVO_PIN_2 5
// PIR Motion Sensor Pin
#define PIR_PIN 7

// WiFi credentials
const char WIFI_SSID[] = "Mein Hotspot";
const char WIFI_PASSWORD[] = "RaInbowFlo";

// MQTT broker config (HiveMQ Cloud)
const char MQTT_BROKER_ADDRESS[] =
"a970366178a840aa8a2c2230bab330aa.s1.eu.hivemq.cloud";
const int MQTT_PORT = 8883; // TLS port
const char MQTT_CLIENT_ID[] = "";
const char MQTT_USERNAME[] = "hivemq.webclient.1751197217617"; // Optional
const char MQTT_PASSWORD[] = ";?Q<%F986R7ZliScYakj"; // Optional

// MQTT topic
const char PUBLISH_TOPIC[] = "AMC/MQTT/rfid/tag";

bool catInside = false;
bool waitingForMotion = false;
unsigned long motionStartTime = 0;
bool motionDetected = false;

int pirSignal = 0;
int rfidWindow = 0;
int servoState = 0;

MFRC522 rfid(SDA_PIN, RST_PIN);
Servo myServo1;
Servo myServo2;

WiFiClientSecure wifiClient;
MQTTClient MQTTClient;

void connectToWiFi() {
  Serial.print("Connecting to WiFi");
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nWiFi connected.");
  Serial.println(WiFi.localIP());
}
```

```
<codeC++> void connectToMQTT() {
```

```
wifiClient.setInsecure();
```

```
MQTTClient.begin(MQTT_BROKER_ADDRESS, MQTT_PORT, wifiClient);  
MQTTClient.setOptions(60, true, 1500);  
while (!MQTTClient.connect(MQTT_CLIENT_ID, MQTT_USERNAME, MQTT_PASSWORD)) {  
    Serial.print(".");  
    delay(1000);  
}  
Serial.println("\nMQTT connected.");
```

```
} </code>
```

```
void setup() {  
    Serial.begin(115200);  
    delay(500);  
  
    connectToWiFi();  
    connectToMQTT();  
  
    SPI.begin(SCK_PIN, MISO_PIN, MOSI_PIN, SDA_PIN);  
    rfid.PCD_Init();  
  
    myServo1.setPeriodHertz(50);  
    myServo1.attach(SERVO_PIN_1, 500, 2400);  
    myServo1.write(0); // locked  
  
    myServo2.setPeriodHertz(50);  
    myServo2.attach(SERVO_PIN_2, 500, 2400);  
    myServo2.write(0); // locked  
  
    pinMode(PIR_PIN, INPUT_PULLUP);  
  
    Serial.println("System ready. Waiting for tag...");  
}  
>/code>
```

```
<code C++>
```

```
void loop() {  
    MQTTClient.loop(); // Maintain MQTT connection  
    pirSignal = digitalRead(PIR_PIN);  
  
    // Step 1: RFID detected  
    if (!waitingForMotion && rfid.PICC_IsNewCardPresent() &&  
    rfid.PICC_ReadCardSerial()) {  
        Serial.println("RFID tag detected. Unlocking...");  
        myServo1.write(90);  
        myServo2.write(90);  
        servoState = 90;
```

```
motionStartTime = millis();
waitingForMotion = true;
motionDetected = false;

rfid.PICC_HaltA();
rfid.PCD_StopCrypto1();
}

// Step 2: Motion check
if (waitingForMotion) {
  rfidWindow = 1;
  if (pirSignal == HIGH) {
    motionDetected = true;
  }
  if (millis() - motionStartTime >= 5000) {
    myServo1.write(0);
    myServo2.write(0);
    servoState = 0;
    if (motionDetected) {
      catInside = !catInside;
      String message = catInside ? "Cat is now INSIDE ☐☐" : "Cat is now
OUTSIDE ☐☐";
      MQTTClient.publish(PUBLISH_TOPIC, message.c_str());
      delay(1000);
    } else {
      delay(1000);
    }
    waitingForMotion = false;
    rfidWindow = 0;
  }
} else {
  rfidWindow = 0;

  delay(50);
}
```

## Discussion

by Mhwae Mhwae Kyaing Kyaing

One of the main limitations encountered during the development of PAWTAL is the short detection range of the RFID sensor, which is largely due to the small antenna size, 60mm x 40 mm (Osoyoo, 2017). The effective range of an RFID system is generally proportional to the size of its antenna (Rose & Kurtz, 2016). In our prototype, the detection range is less than 2 cm, requiring the cat to be in very close proximity to the door for the tag to be recognized.

This limitation could be addressed in future improvements through two potential approaches. Firstly by using an ultra-high frequency (UHF) RFID system operating at 868 MHz, which offers significantly greater range but comes at a high cost which may not be suitable for a student project. Secondly by increasing the size of the tag's antenna, which could enhance the detection range but requires technical knowledge and resources beyond the current scope of the team.

NodeRED is subscribed to all topic messages including the message that is sent when the door is unlocked but no motion is detected. This message is then sent to the email. While the other messages notify of the cat's whereabouts, this message is functionally useless and clutters the email inbox. To improve efficiency, later updates to the nodeRED flow's function node should disallow the continued flow of unwanted messages.

## Conclusion

by Mhwae Mhwae Kyaing Kyaing

Project Pawtal has successfully integrated sensor technology and MQTT communication into a fully functional prototype with real-time data access. The system demonstrates a practical and accessible approach to automating a pet door. Despite the limited range of RFID antenna and minor issues, this project presents a strong proof of concept as a versatile product with future applications and improvements. Future developments could include the integration of location tracking sensors, AI-based behavior analysis, or facial recognition technology to eliminate the need for RFID tags. Additionally, replacing the tag with a pet microchip would further improve the user experience. Moreover, PAWTAL can be further developed into a more complete solution by adding features like a mobile/web app for real-time monitoring and integration with automated pet feeders.

## Demonstration & Presentation

[youcut\\_20250725\\_093122533.mp4](#)

## Reference

Osoyoo.(2017, September 11).Arduino lesson – RFID RC522. Osoyoo Learning.  
<https://osoyoo.com/2017/09/11/arduino-lesson-rfid-rc522/>

Rose, M., & Kurtz, J.(2016, May 16). NFC – A closer look [Presentation].Future Electronics.  
<http://www.FutureElectronics.com>

From:  
<https://student-wiki.eolab.de/> - **HSRW EOLab Students Wiki**

Permanent link:  
<https://student-wiki.eolab.de/doku.php?id=amc:ss2025:group-d:start&rev=1753526618>

Last update: **2025/07/26 12:43**

