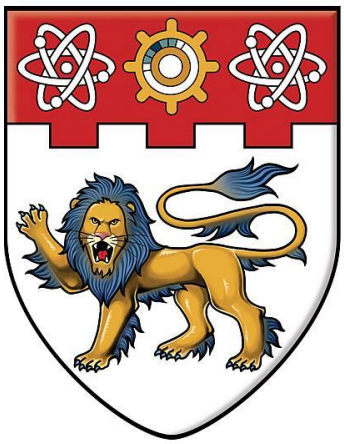


NANYANG TECHNOLOGICAL UNIVERSITY

FYP Interim Report

Submitted by

Ooi Kok Yih

U1921262E

School of Computer Science and Engineering

Nanyang Technological university

Introduction

Artificial Intelligence, monitoring, automation, geotargeting and other forms of advancement in information technology has led to a technological evolution in today. Houseware appliances such as TVs, printers and refrigerators etc. can be accessible wirelessly via the internet. This has led to an interconnection and intercommunication of things we operate within our daily lives. As such, it also raises the question of how we can utilize these advancements to innovate and enhance our home and quality of life.

Objective of Project

This project aims to develop smart home automation systems using ESP32 and Blynk, an online IOT platform service. ESP32 will serve as a communication point between user and the desired object to operate while Blynk act as a controller for user input and output.

The implementation of these systems ultimately aims to innovate and improve the way people live by upgrading from traditional physical switches and keys to more efficient and intelligent mode of interacting with household appliances.

Project TimeLine

Final Year Project - Smart Home Automation

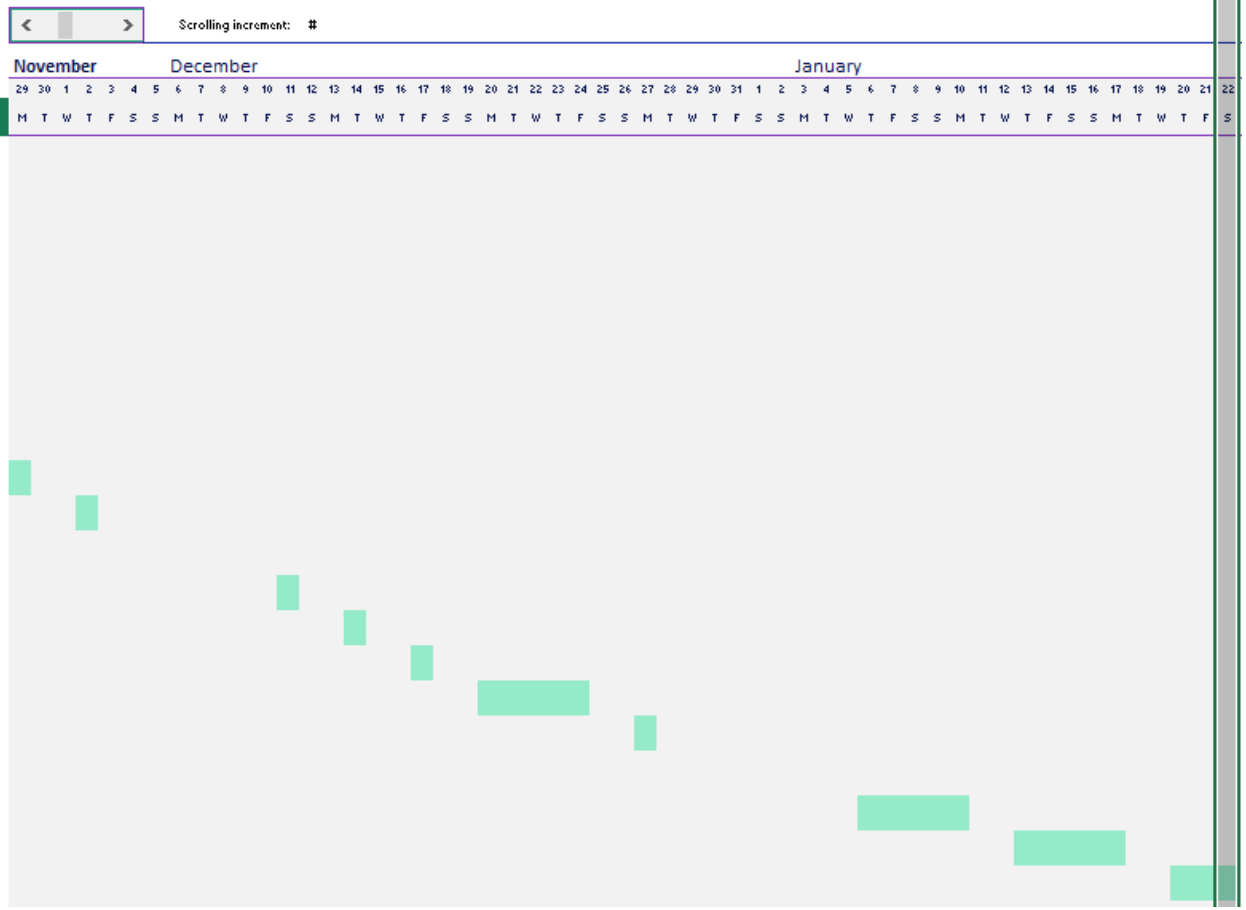
Ooi Kok Yih (U1921262E)

Project start date: 8/8/2021

Milestone marker: 0

Scrolling increment: #

Milestone description	Assigned to	Progress	Start	Days
Proposal		100%	8/8/2021	3
Discussion on objectives		100%	8/11/2021	3
Research on Ideas		100%	8/14/2021	60
Sourcing for suitable components		100%	10/13/2021	14
Purchasing of components		100%	10/27/2021	20
Implementation of Idea 1 (Light and Fan)				
Motion Sensing on LED		100%	11/16/2021	5
Motion Sensing on mini motor Fan		100%	11/21/2021	3
Setting up Blynk for control		100%	11/24/2021	7
Redesigning circuit to swap LED to lightbulb		100%	12/1/2021	3
Implementation of Idea 2 (Smart Door)				
Wiring of Circuits and Soldering		100%	12/10/2021	3
RFID for Door Control		100%	12/13/2021	3
Setting up Blynk for control		100%	12/16/2021	3
Testing of ESP32 Cam Facial Recognition		100%	12/19/2021	7
Incoporating Facial Recognition to circuit		100%	12/26/2021	3
Additional Ideas (Object Detection)				
Research on Additional Ideas		70%	1/5/2022	7
Purchasing of components		80%	1/12/2022	7
Implementation of Object Detection on ESP32		30%	1/19/2022	5



Work Progress

Idea 1 – Light and Fan Control

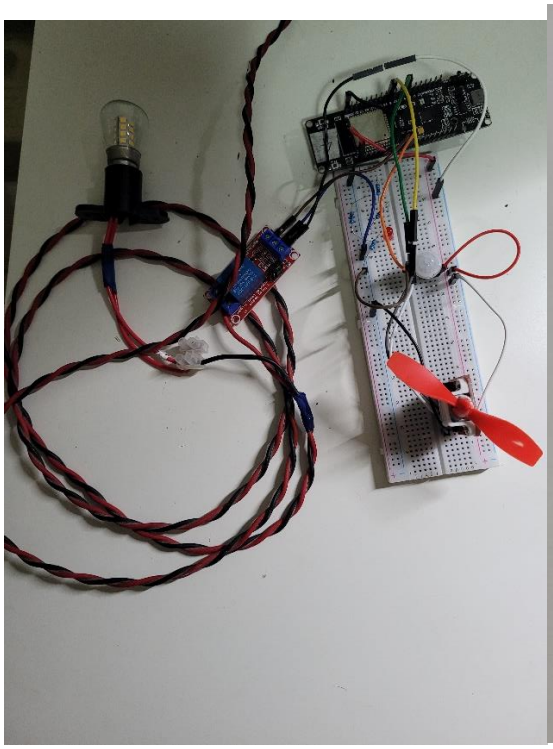
Hardware used

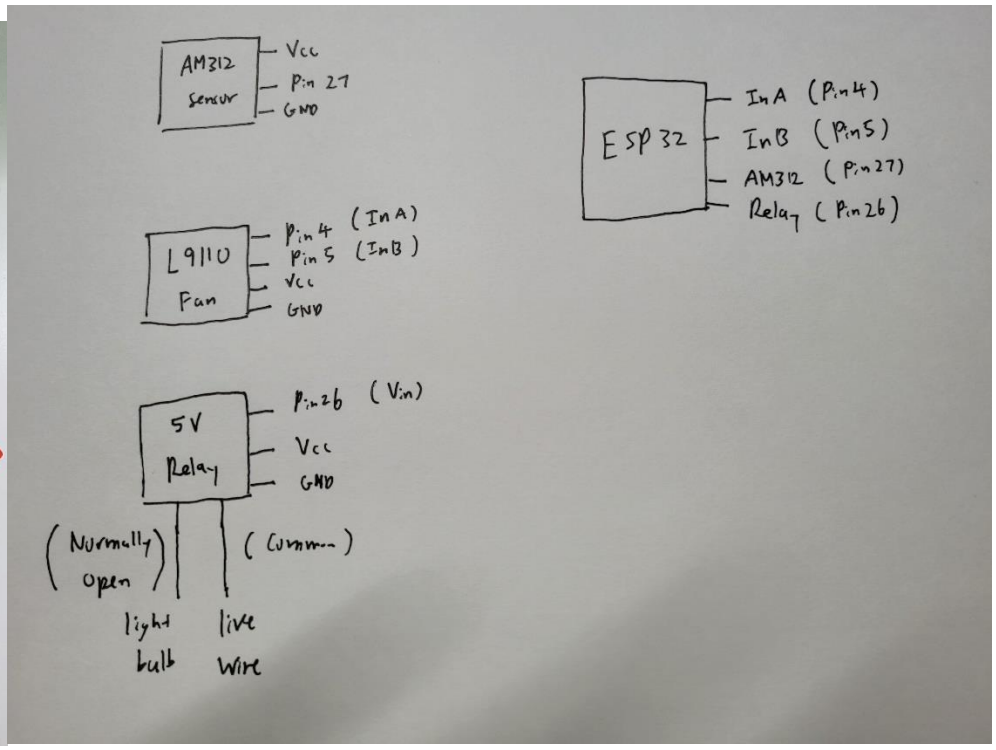
Components	Quantity
ESP32	1
PIR Sensor	1
3 pin plug	1
L9110 Fan Motor Module	1
Breadboard	1
LED Light Bulb	1
5V Relay	1

Software used

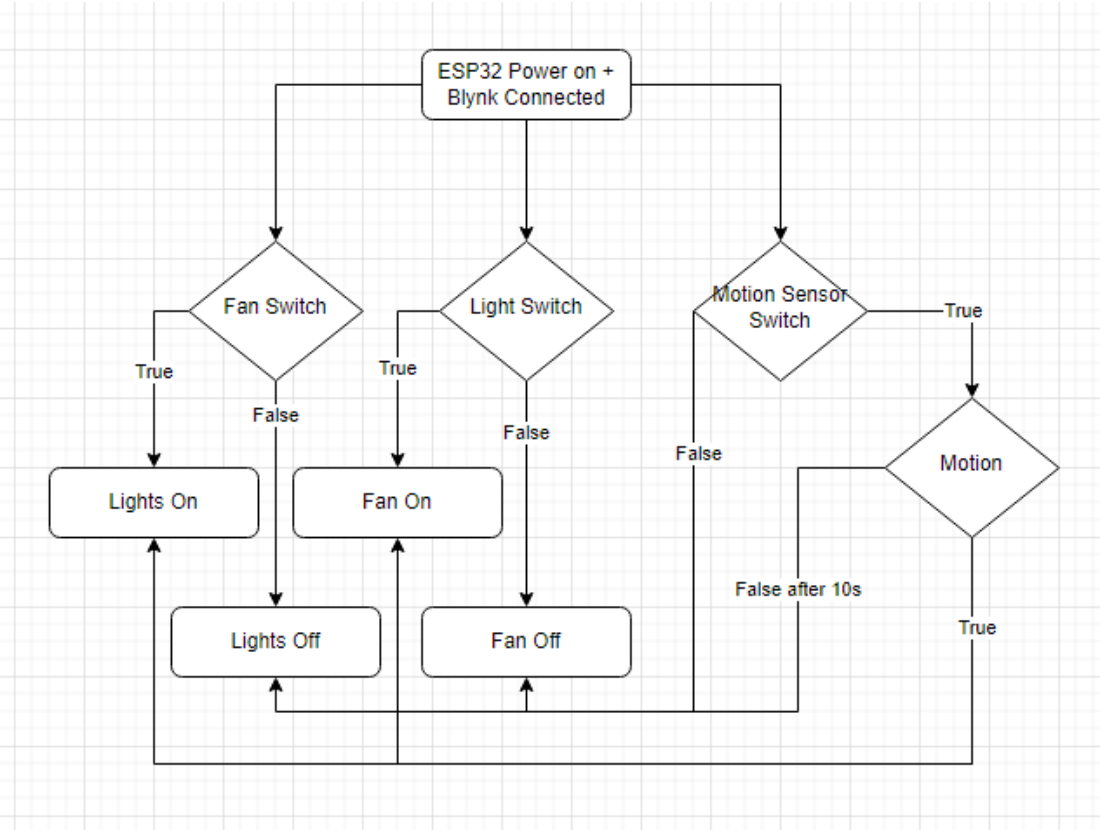
1. Arduino
2. Blynk

Circuit Wiring



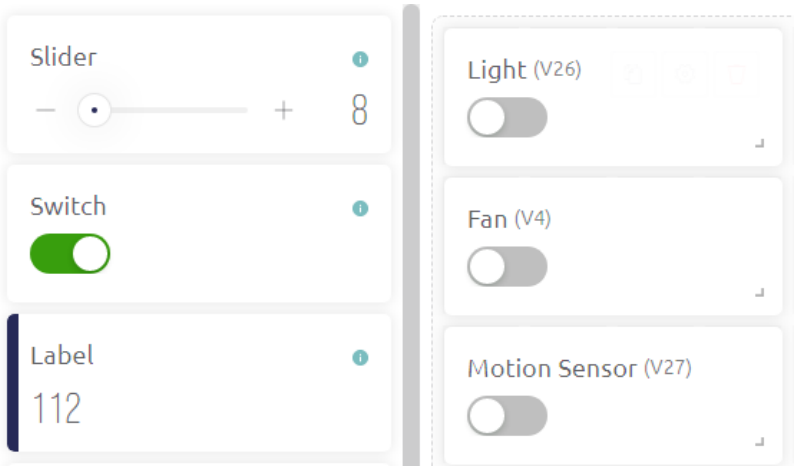


System Logic



Blynk was chosen as the mode of control for this idea as the tool is readily available online and its libraries were very well documented. It was relatively simple to setup and operate. The server connection was very reliable as well with minimal errors encountered.

The application layout can be designed on the website itself simply by dragging the icons, then assigning a virtual pin number to it.



The numbers at the end of the name of the switch is the assigned virtual pin number.

The Blynk library is then imported into the Arduino sketch, followed by declaring the auth token of the layout that was created.

```
#define BLYNK_TEMPLATE_ID "TMPLpiM5SPEw" //Auth Token
#define BLYNK_DEVICE_NAME "LED and Fan 1"

#define BLYNK_FIRMWARE_VERSION        "0.1.0"

#define BLYNK_PRINT Serial
//#define BLYNK_DEBUG

#define APP_DEBUG

// Uncomment your board, or configure a custom board in Settings.h
//#define USE_WROVER_BOARD
//#define USE_TTGO_T7

#include "BlynkEdgent.h"
```

The respective virtual pin number's functions (BLYNK_WRITE) are called whenever an event is triggered on the mobile application or web server.

```
BLYNK_WRITE(V27)
{
  motion = param.asInt();
}

BLYNK_WRITE(V26)//LED
{
  light  = param.asInt();
}

BLYNK_WRITE(V4)//FAN
{
  fan = param.asInt();
}
```

```
void Fan(int value)
{
  if(value ==1)
  {
    digitalWrite(INA,LOW);
    digitalWrite(INB,HIGH);
  }
  else
  {
    digitalWrite(INA,LOW);
    digitalWrite(INB,LOW);
  }

}

void Light(int value)
{
  digitalWrite(led, value);
}
```

BLYNK_WRITE function can be configured to assign a value to our desired device.

We then use this "assigned value" to change the state of our device

```
void getPirValue(void)
{
  pirValue = digitalRead(motionSensor);
  //Serial.println(pirValue);
  if (pirValue == HIGH)
  {
    Light(1);
    Fan(1);
    startTimer = true;
    lastTrigger = millis();
    if(pirState == LOW)//check whether we turn off
    {
      pirState = HIGH; //to indicate we just turn on
      Serial.println("Motion detected!");
    }
  }
  else
  {
    if(pirState == HIGH && (startTimer &&(now-lastTrigger > (timeSeconds*1000))))
    {
      Light(0);
      Fan(0);
      pirState = LOW; //to indicate we just turn off
      startTimer = false;
      Serial.println("Motion ended!");
    }
  }

  if(now-lastTrigger < (timeSeconds*1000))
  {
    Serial.println(now-lastTrigger);
    Light(1);
    Fan(1);
  }

}
```

As for the motion sensing part, the sensor will continuously output values to the GPIO pin. Thus, it is difficult to control the state of our device, due to the rapid change in values. Hence, a “state” value is needed to indicate what state our lights and fans are at.

Whenever the sensor detects motion, it will turn on our devices, refresh the last triggered time, change device state to HIGH if LOW and start a timer. Devices will keep running if (current time – last triggered time) is less than timer value.

If device state is HIGH and timer times out, device state transitions to LOW, timer stops, and everything turns off.

In conclusion, both concepts show that any household appliances (not just limited to fans and lights in this simulation) can be remotely controlled over internet or automated by motion sensing via rewiring of the power cables to a relay and assigning a GPIO pin of ESP32 to control the state. Thus, avoiding the need to press physical switches.

Idea 2 – Door Control

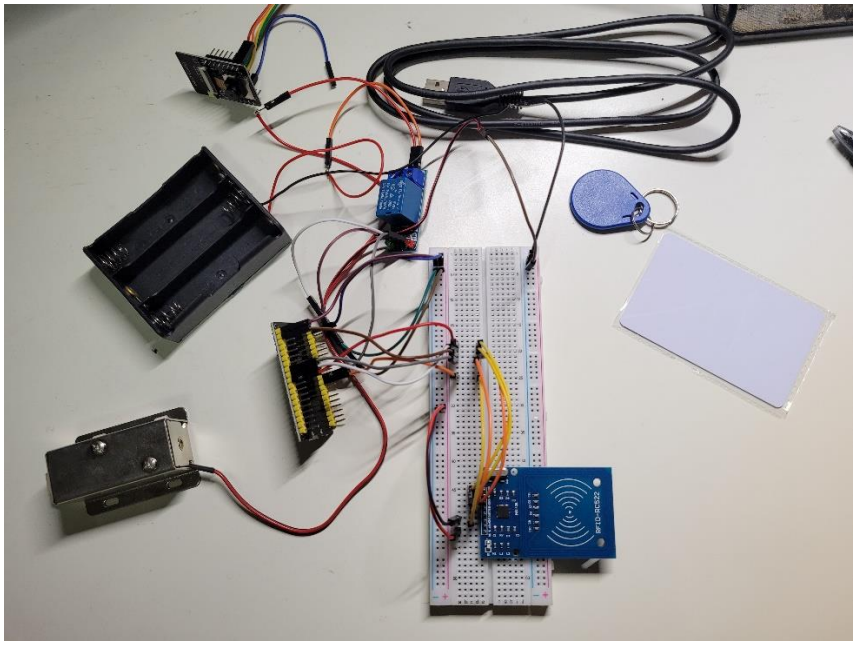
Hardware Used

Components	Quantity
ESP32	1
Solenoid Lock 12V	1
RFID RC522	1
18650 batteries	3
3x18650 battery holder	1
ESP32 Cam	1
5V Relay	1
USB-TTL FT232RL Converter	1

Software Used

1. Arduino
2. Blynk

Circuit Wiring

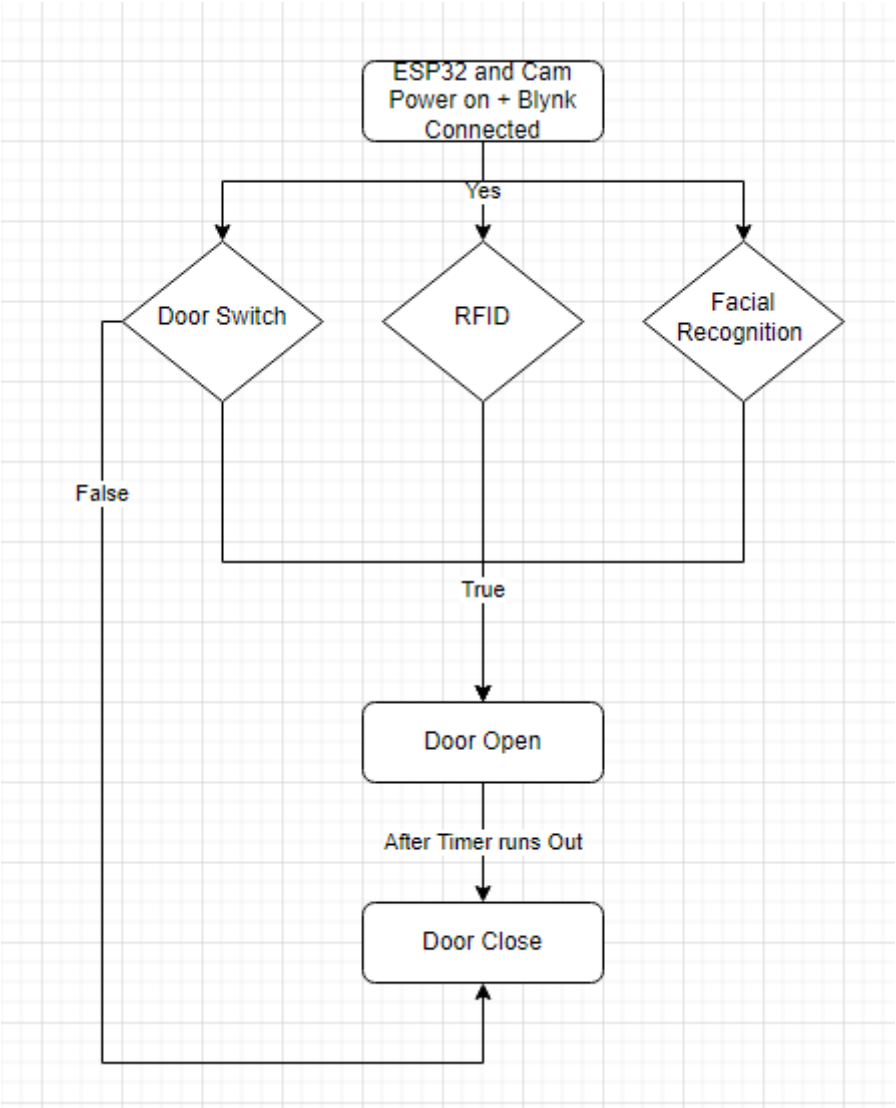


5V Relay	ESP32
Vcc	Vcc
Gnd	Gnd
Vin	Pin 26

ESP32	RFID Reader
3.3V	3.3V
22	RST
GND	GND
Not required	IRQ
19	MISO
23	MOSI
18	SCK
5	SDA

ESP32-CAM	FTDI Programmer
GND	GND
5V	VCC (5V)
U0R	TX
U0T	RX
GPIO 0	GND

System Logic



Similar to the first idea, the Blynk app was chosen as the mode for remotely controlling the door.

```
BLYNK_WRITE(V5)
{
  door = param.asInt();
}

 if (door == 1){
   Serial.println("Opening door");
   digitalWrite(RELAY_PIN, LOW);
   delay(5000);
   digitalWrite(RELAY_PIN, HIGH);
 }
```

As for physical method, RFID was selected as it is much faster to scan than to unlock with traditional bulky keys or inputting pin numbers on a keypad.

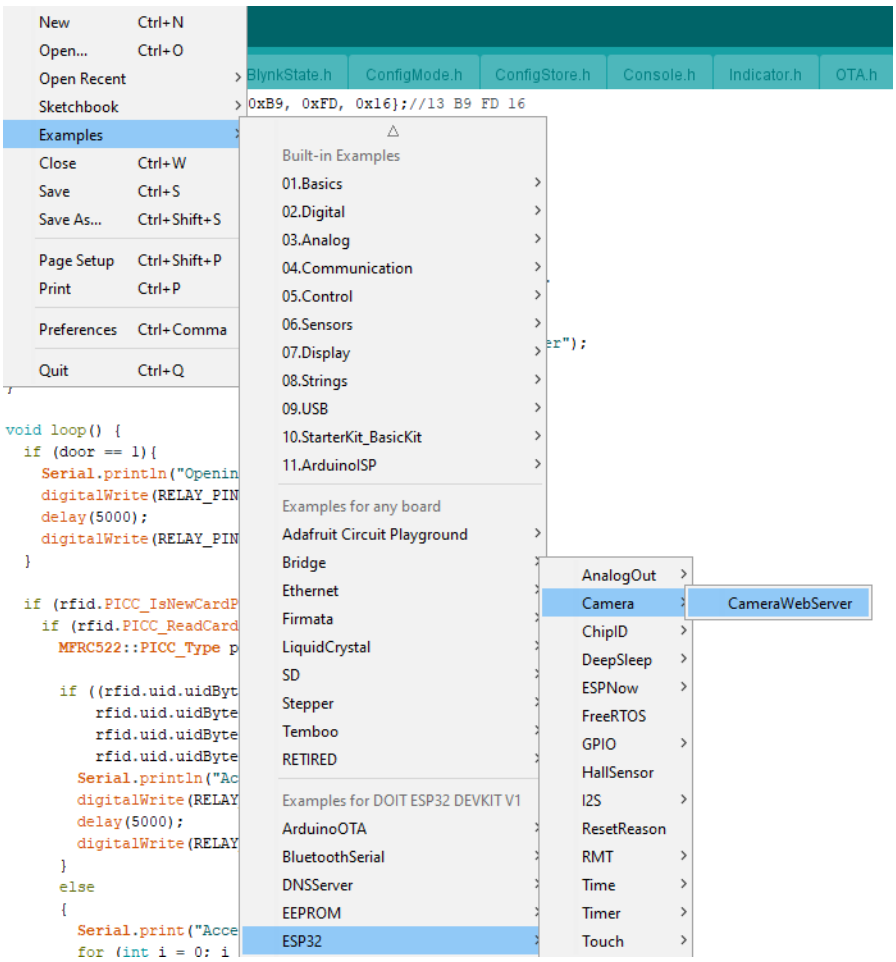
```
if (rfid.PICC_IsNewCardPresent()) { // new tag is available
  if (rfid.PICC_ReadCardSerial()) { // NUID has been readed
    MFRC522::PICC_Type piccType = rfid.PICC_GetType(rfid.uid.sak);

    if ((rfid.uid.uidByte[0] == keyTagUID[0] &&
        rfid.uid.uidByte[1] == keyTagUID[1] &&
        rfid.uid.uidByte[2] == keyTagUID[2] &&
        rfid.uid.uidByte[3] == keyTagUID[3])) {
      Serial.println("Access is granted");
      digitalWrite(RELAY_PIN, LOW);  // unlock the door
      delay(5000);
      digitalWrite(RELAY_PIN, HIGH); // lock the door
    }
    else
    {
      Serial.print("Access denied, UID:");
      for (int i = 0; i < rfid.uid.size; i++) {
        Serial.print(rfid.uid.uidByte[i] < 0x10 ? " 0" : " ");
        Serial.print(rfid.uid.uidByte[i], HEX);
      }
      Serial.println();
    }

    rfid.PICC_HaltA(); // halt PICC
    rfid.PCD_StopCrypto1(); // stop encryption on PCD
  }
}
```

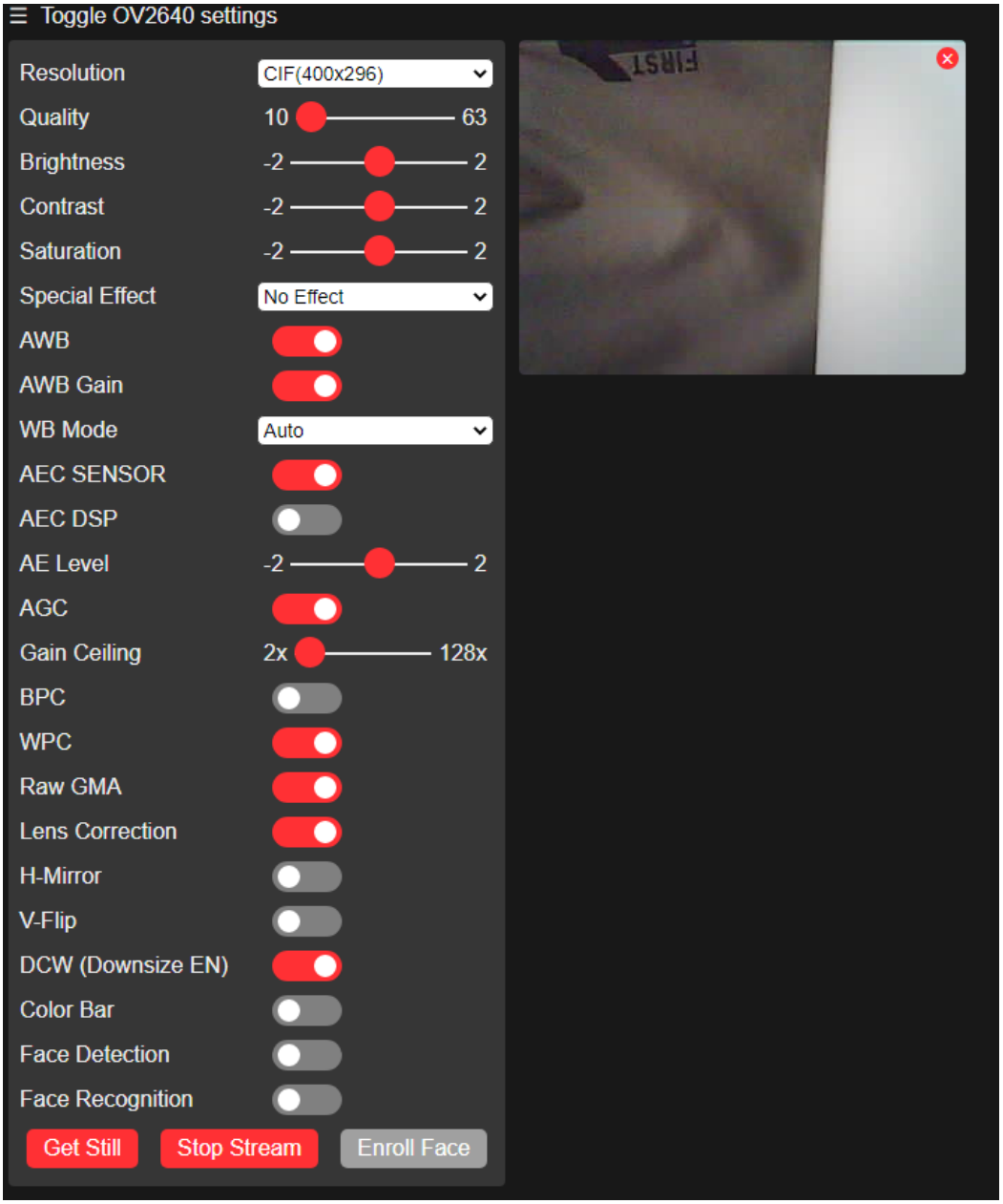
When scanning a RFID card, the program will check and compare the UID to an array. If the UID matches, the 12V solenoid is unlocked for 5 seconds before going back to lock state.

For facial recognition implementation, the CameraWebServer example available on Arduino was used.




```
WiFi connected
Starting web server on port: '80'
Starting stream server on port: '81'
Camera Ready! Use 'http://192.168.1.131' to connect
```

Upon starting the ESP32 Cam, the web server can be accessed with the IP Address given.



The camera web server has an option below to enable face detection as well as face recognition. The output of face recognition was then used to control the state of the 12V solenoid lock.

Future Work

I have two additional ideas before the completion of this project.

For the first idea, I am researching on how to utilise object detection as part of a home monitoring system. This monitoring system can also be equipped with various sensors such as rain sensor. For example, if the ESP32 Cam detects laundry and its sensor detects rain, it will prompt user to keep their laundry. The system can also be used to alert users whenever there is a parcel in front of their doorstep.

For the second idea, I am planning to develop a portable touchscreen remote control with a ILI9341 so that users will not need to reach for their phones and launch the Blynk App whenever they wish to control a device. This remote control can also be scaled down to a wearable device similar to LilyGo TTGO ESP32 Smartwatch.

Conclusion

In conclusion, I have learnt the versatility and usefulness of ESP32 in implementing smart home automation systems. ESP32 has fast computing speed, better Wi-Fi and more GPIO ports. These features make the chip a very reliable hardware in operating numerous household appliances and sensors in real time. There were also many well documented tutorials online to guide me along the way on how to develop my systems. Overall, it was a very interesting project and I am looking forward to complete my next two ideas.