**Schematic Diagram**

Diagram, schematic

Description automatically generated

**Source Code**

**Arduino Uno**

#include <SPI.h>

#include <RFID.h>

RFID rfid(10, 5);

int DataPinOne = 0;

int DataPinTwo = 0;

int WifiControlState\_OutletA = LOW;

int WifiControlState\_OutletB = LOW;

int OutletState\_A = LOW;

int OutletState\_B = LOW;

bool OutletA\_Selected = true;

bool OutletB\_Selected = false;

int buttonA\_State = LOW;

int buttonB\_State = LOW;

bool RFID\_MODE = true;

const int OutletA\_Pin = 3;

const int OutletB\_Pin = 4;

unsigned char serNum[5];

unsigned char status;

unsigned char str[MAX\_LEN];

unsigned char blockAddr;

String MasterID = "ecb40e49"; // REPLACE this Tag ID with your Tag ID!!!

String OutletA\_ID = "335f7a1a";

String OutletB\_ID = "22910834";

String ChangeModeID = "b01a8032";

String Read\_ID = "";

unsigned char sectorKeyA[16][16] = {

{ 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF } ,

{ 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF } ,

{ 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF } ,

{ 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF } ,

{ 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF } ,

{ 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF } ,

{ 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF } ,

{ 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF } ,

{ 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF } ,

{ 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF } ,

{ 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF } ,

{ 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF } ,

{ 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF } ,

{ 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF } ,

{ 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF } ,

{ 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF } ,

};

void setup() {

// put your setup code here, to run once:

Serial.begin(9600);

SPI.begin();

rfid.init();

pinMode(6,INPUT);

pinMode(7,INPUT);

pinMode(A4, INPUT);

pinMode(A5, INPUT);

pinMode(OutletA\_Pin,OUTPUT);

pinMode(OutletB\_Pin,OUTPUT);

}

void loop() {

// put your main code here, to run repeatedly:

ReadButton();

findCard();

if(!RFID\_MODE)

{

UpdateWifiPinState();

}

UpdateDataPinState();

}

//Read the selector button inputs

void ReadButton()

{

buttonA\_State = digitalRead(A5);

buttonB\_State = digitalRead(A4);

if(buttonA\_State == HIGH)

{

OutletA\_Selected = true;

OutletB\_Selected = false;

Serial.println("Outlet A is selected");

}

else if(buttonB\_State == HIGH)

{

OutletA\_Selected = false;

OutletB\_Selected = true;

Serial.println("Outlet B is selected");

}

}

//Find if card is present

void findCard()

{

rfid.findCard(PICC\_REQIDL, str);

if (rfid.anticoll(str) == MI\_OK) {

Read\_ID = "";

char id[8];

Serial.print("The card's number is : ");

for (int i = 0; i < 4; i++) {

Serial.print(0x0F & (str[i] >> 4), HEX);

Serial.print(0x0F & str[i], HEX);

Read\_ID += String(0x0F & (str[i] >> 4), HEX);

Read\_ID += String(0x0F & (str[i]), HEX);

}

// id[8] = Read\_ID;

// char chr;

//

// int i = 0;

// while(id[i]){

// chr = Read\_ID[i];

// toupper(chr);

// i++;

// }

Serial.println("");

Serial.println(Read\_ID);

memcpy(rfid.serNum, str, 5);

}

rfid.selectTag(rfid.serNum);

readCard(4);

rfid.halt();

}

//Process RFID card. Will only process it if RFID mode is on.

void readCard(int blockAddr) {

if ( rfid.auth(PICC\_AUTHENT1A, blockAddr, sectorKeyA[blockAddr / 4], rfid.serNum) == MI\_OK)

{

if(RFID\_MODE)

RFID\_StateChange();

if(Read\_ID == ChangeModeID)

{

if(RFID\_MODE)

{

RFID\_MODE = false;

Serial.println("RFID OFF");

}

else

{

RFID\_MODE = true;

Serial.println("RFID ON");

}

}

}

}

//Check for WIFI input only in WIFI mode

void UpdateWifiPinState()

{

WifiControlState\_OutletA = digitalRead(6);

WifiControlState\_OutletB = digitalRead(7);

if(WifiControlState\_OutletA == LOW)

{

OutletState\_A = LOW;

//Serial.println("Turning off outlet A");

}

else

{

OutletState\_A = HIGH;

//Serial.println("Turning on outlet A");

}

if(WifiControlState\_OutletB == LOW)

{

OutletState\_B = HIGH;

//Serial.println("Turning off outlet B");

}

else

{

OutletState\_B = LOW;

//Serial.println("Turning on outlet B");

}

}

//Called when altering states via RFID

void RFID\_StateChange(){

//Turn on/off both outlets

//ADD condition if which outlet is selected

if(Read\_ID == OutletA\_ID && OutletA\_Selected || Read\_ID == MasterID && OutletA\_Selected)

{

Serial.println(" Access Granted!");

if(OutletState\_A == HIGH)

{

OutletState\_A = LOW;

Serial.println("Turning off outlet A");

}

else

{

OutletState\_A = HIGH;

Serial.println("Turning on outlet A");

}

}

//Turn on/off OutletB. Considering OutletB as active LOW

else if(Read\_ID == OutletB\_ID && OutletB\_Selected || Read\_ID == MasterID && OutletB\_Selected)

{

Serial.println(" Access Granted!");

if(OutletState\_B == LOW)

{

OutletState\_B = HIGH;

Serial.println("Turning off outlet B");

}

else

{

OutletState\_B = LOW;

Serial.println("Turning on outlet B");

}

}

else

{

Serial.println(" Access Denied!");

}

delay(650);

}

//Called when an outlet's state is altered and is always called last.

void UpdateDataPinState()

{

if(OutletState\_A == HIGH)

{

digitalWrite(OutletA\_Pin, HIGH);

}

else

{

digitalWrite(OutletA\_Pin, LOW);

}

if(OutletState\_B == HIGH)

{

digitalWrite(OutletB\_Pin, LOW);

}

else

{

digitalWrite(OutletB\_Pin, HIGH);

}

}

**ESP32 (WI-FI Module)**

// Template ID, Device Name and Auth Token are provided by the Blynk.Cloud

// See the Device Info tab, or Template settings

#define BLYNK\_TEMPLATE\_ID "TMPL-30TuIjM"

#define BLYNK\_DEVICE\_NAME "Quickstart Device"

#define BLYNK\_AUTH\_TOKEN "sRttWQKSYYIZlejxxwyQ-ryjWE3-HB0n"

// Comment this out to disable prints and save space

#define BLYNK\_PRINT Serial

#include <WiFi.h>

#include <WiFiClient.h>

#include <BlynkSimpleEsp32.h>

char auth[] = BLYNK\_AUTH\_TOKEN;

// Your WiFi credentials.

// Set password to "" for open networks.

char ssid[] = "PLDTHOMEFIBRz5yfG-Orion";

char pass[] = "Confidential";

BlynkTimer timer;

// This function is called every time the Virtual Pin 0 state changes

BLYNK\_WRITE(V0)

{

// Set incoming value from pin V0 to a variable

int value = param.asInt();

// Update state

Blynk.virtualWrite(V1, value);

}

// This function is called every time the device is connected to the Blynk.Cloud

BLYNK\_CONNECTED()

{

// Change Web Link Button message to "Congratulations!"

Blynk.setProperty(V3, "offImageUrl", "https://static-image.nyc3.cdn.digitaloceanspaces.com/general/fte/congratulations.png");

Blynk.setProperty(V3, "onImageUrl", "https://static-image.nyc3.cdn.digitaloceanspaces.com/general/fte/congratulations\_pressed.png");

Blynk.setProperty(V3, "url", "https://docs.blynk.io/en/getting-started/what-do-i-need-to-blynk/how-quickstart-device-was-made");

}

// This function sends Arduino's uptime every second to Virtual Pin 2.

void myTimerEvent()

{

// You can send any value at any time.

// Please don't send more that 10 values per second.

Blynk.virtualWrite(V2, millis() / 1000);

}

// This function will be called every time Slider Widget

// in Blynk app writes values to the Virtual Pin V1

int vPin4 = 0;

int vPin5 = 0;

BLYNK\_WRITE(V4)

{

vPin4 = param.asInt(); // assigning incoming value from pin V4 to a variable

// process received value

}

BLYNK\_WRITE(V5)

{

vPin5 = param.asInt(); // assigning incoming value from pin V5 to a variable

// process received value

}

void setup()

{

// Debug console

Serial.begin(115200);

Blynk.begin(auth, ssid, pass);

pinMode(4, OUTPUT);

pinMode(5, OUTPUT);

// You can also specify server:

//Blynk.begin(auth, ssid, pass, "blynk.cloud", 80);

//Blynk.begin(auth, ssid, pass, IPAddress(192,168,1,100), 8080);

// Setup a function to be called every second

timer.setInterval(1000L, myTimerEvent);

}

void loop()

{

Blynk.run();

timer.run();

if(vPin4 == 1)

{

digitalWrite(4, HIGH);

}

else

{

digitalWrite(4, LOW);

}

if(vPin5 == 1)

{

digitalWrite(5, HIGH);

}

else

{

digitalWrite(5, LOW);

}

// You can inject your own code or combine it with other sketches.

// Check other examples on how to communicate with Blynk. Remember

// to avoid delay() function!

}

**Hardware Requirements**

Arduino UNO: On hand

RFID-RC522: Php 117

2-Gang Outlet w/ Utility Box: Php 200

Open Back Extension Cord Php 63

AC Wire: Php 60

Pluggable Light Bulbs: On hand

(2) Open Back Plug: Php 75

(2) Relay Module: Php 60

(2) Buttons

(3) RFID Tags/RFID Cards: Php 135

ESP32 WIFI IoT Dev Board: Php 349

Soldering Kit: Php 444

Total: Php 1503

**Software Requirements:**

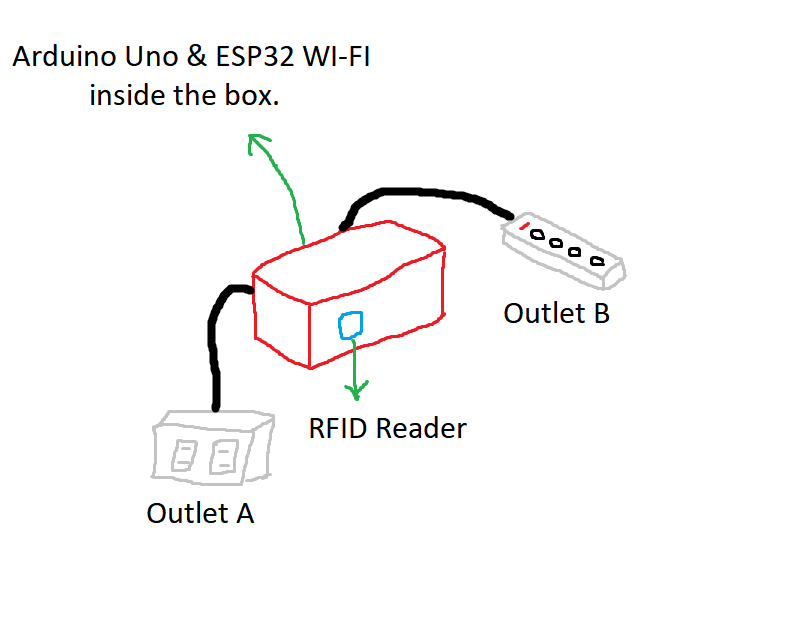
Arduino IDE

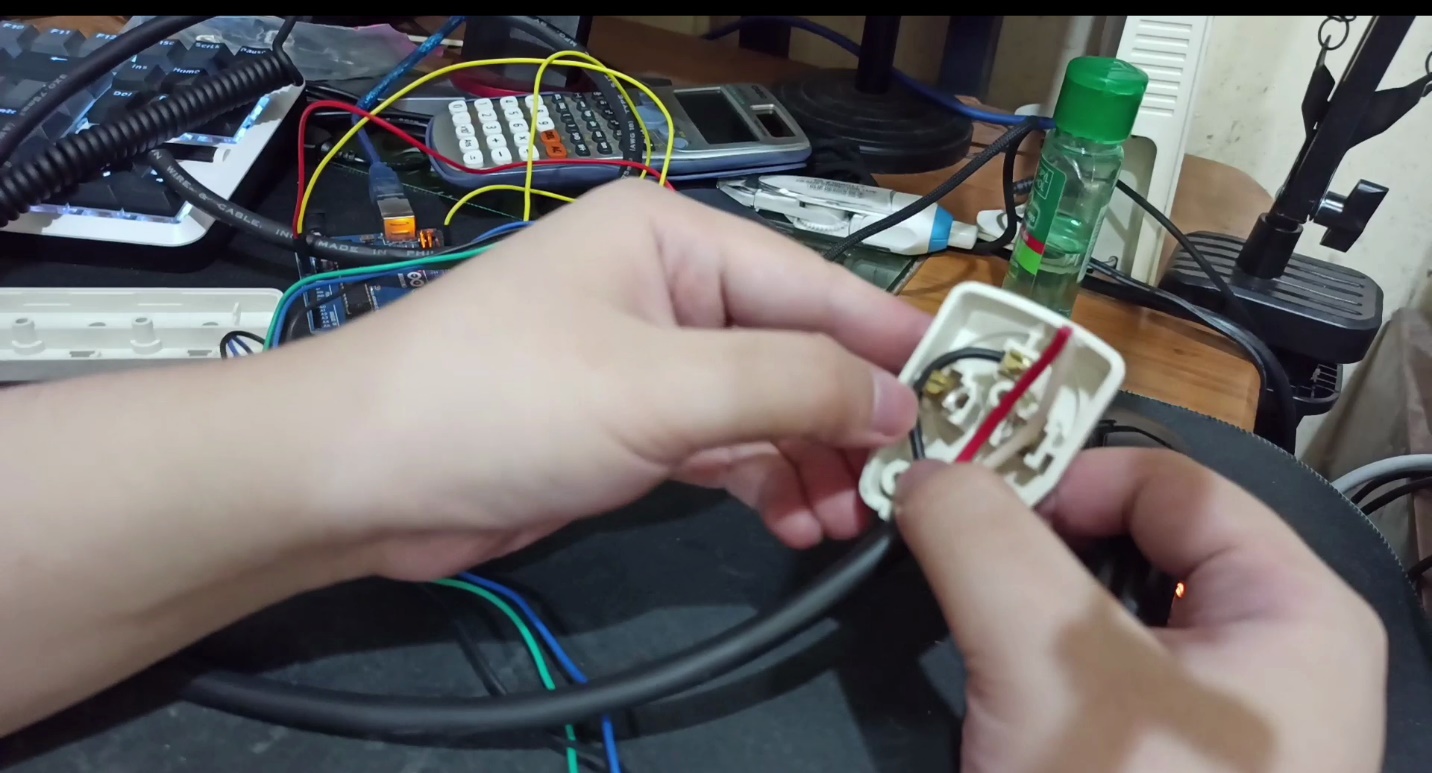
SPI (Library)

MFRC522 (Library)

Blynk.io

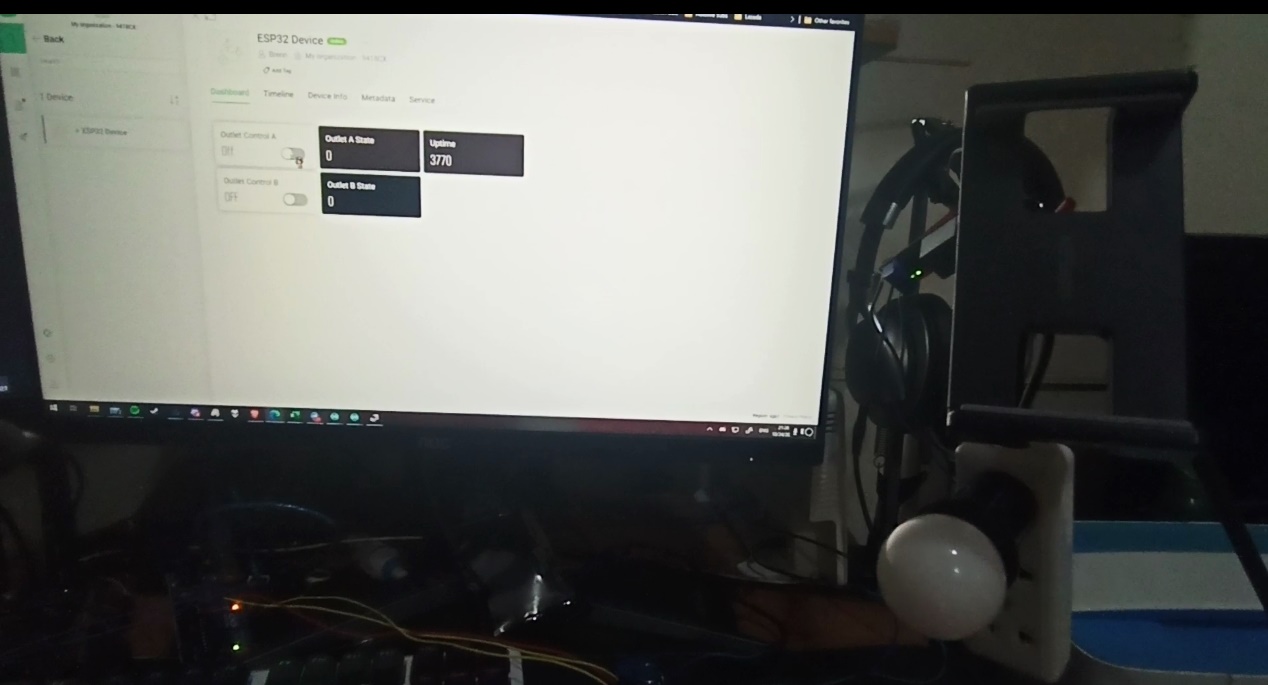
**Pictorials with short label or description during:**

1. Project Design
2. Project Creation



**Assembling the plug**

1. Project Testing & Debugging



**Debugging Web App**

1. Finished Project for Presentation

**Finished Product**