#ifdef ESP32

#include <WiFi.h>

#else

#include <ESP8266WiFi.h>

#endif

#include <WiFiClientSecure.h>

#include <UniversalTelegramBot.h> // Universal Telegram Bot Library written by Brian Lough: https://github.com/witnessmenow/Universal-Arduino-Telegram-Bot

#include <ArduinoJson.h>

// Replace with your network credentials

const char\* ssid = "SRC 24G";

const char\* password = "src@internet";

// Initialize Telegram BOT

#define BOTtoken "6299072347:AAHJBg-pTm348LN7VBvQgDTmo1OCtV3pR8M" // your Bot Token (Get from Botfather)

String strs[8]={"0","0","0","0","0","0","0","0"};

int StringCount = 0;

// Use @myidbot to find out the chat ID of an individual or a group

// Also note that you need to click "start" on a bot before it can

// message you

#define CHAT\_ID "5993400389"

#ifdef ESP8266

X509List cert(TELEGRAM\_CERTIFICATE\_ROOT);

#endif

WiFiClientSecure client;

UniversalTelegramBot bot(BOTtoken, client);

// Checks for new messages every 1 second.

int botRequestDelay = 1000;

unsigned long lastTimeBotRan;

const int ledPin = 2;

bool ledState = LOW;

// Handle what happens when you receive new messages

void handleNewMessages(int numNewMessages) {

Serial.println("handleNewMessages");

Serial.println(String(numNewMessages));

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

// Chat id of the requester

String chat\_id = String(bot.messages[i].chat\_id);

if (chat\_id != CHAT\_ID){

bot.sendMessage(chat\_id, "Unauthorized user", "");

continue;

}

// Print the received message

String text = bot.messages[i].text;

Serial.println(text);

String from\_name = bot.messages[i].from\_name;

if (text == "/start") {

String welcome = "Welcome, " + from\_name + ".\n";

welcome += "Use the following commands to control your outputs.\n\n";

welcome += "/led\_on to turn GPIO ON \n";

welcome += "/led\_off to turn GPIO OFF \n";

welcome += "/state to request current GPIO state \n";

bot.sendMessage(chat\_id, welcome, "");

}

if (text == "/led\_on") {

bot.sendMessage(chat\_id, "LED state set to ON", "");

ledState = HIGH;

digitalWrite(ledPin, ledState);

}

if (text == "/led\_off") {

bot.sendMessage(chat\_id, "LED state set to OFF", "");

ledState = LOW;

digitalWrite(ledPin, ledState);

}

if (text == "/state") {

if (digitalRead(ledPin)){

bot.sendMessage(chat\_id, "LED is ON", "");

}

else{

bot.sendMessage(chat\_id, "LED is OFF", "");

}

}

}

}

void setup() {

Serial.begin(9600);

#ifdef ESP8266

configTime(0, 0, "pool.ntp.org"); // get UTC time via NTP

client.setTrustAnchors(&cert); // Add root certificate for api.telegram.org

#endif

pinMode(ledPin, OUTPUT);

digitalWrite(ledPin, ledState);

// Connect to Wi-Fi

WiFi.mode(WIFI\_STA);

WiFi.begin(ssid, password);

#ifdef ESP32

client.setCACert(TELEGRAM\_CERTIFICATE\_ROOT); // Add root certificate for api.telegram.org

#endif

while (WiFi.status() != WL\_CONNECTED) {

delay(1000);

Serial.println("Connecting to WiFi..");

}

// Print ESP32 Local IP Address

Serial.println(WiFi.localIP());

delay(1000);

bot.sendMessage(CHAT\_ID, "BOT STARTED", "");

}

void loop() {

if (Serial.available())

{

String rcv = Serial.readStringUntil('\n');

StringCount=0;

while (rcv.length() > 0)

{

int index = rcv.indexOf('^');

if (index == -1) // No space found

{

strs[StringCount++] = rcv;

break;

}

else

{

strs[StringCount++] = rcv.substring(0, index);

rcv = rcv.substring(index+1);

}

Serial.println("Sending..");

bot.sendMessage(CHAT\_ID, strs[0], "");

}

}

if (millis() > lastTimeBotRan + botRequestDelay) {

int numNewMessages = bot.getUpdates(bot.last\_message\_received + 1);

while(numNewMessages) {

Serial.println("got response");

handleNewMessages(numNewMessages);

numNewMessages = bot.getUpdates(bot.last\_message\_received + 1);

}

lastTimeBotRan = millis();

}

}