

*School Of Computer Science and Engineering VIT-AP UNIVERSITY, INAVOLU, AMARAVATI*

**HOME AUTOMATION AND SECURITY**

*Use of automated technologies to simplify the action of movement in homes.*

**Guided By:**

Prof: Dr. S.M.A.K.AZAD

**Team Members:**

20BCD7094 - Rampam Greeshma Geethika

20BCD7164 - Momin Mohammad Muneeb

20BCD7179 - E Varun Sai

20BCE7023 - Kinjal

20BEC7035 - Koppala Vianney Xavier

20BEC7037 - Viha Jaiswal

**ABSTRACT**

Generally, we have security in the houses as locks or human security. But we are not updated at every moment if there is something happening in the home. We cannot make any changes in the home also.

A home automation system typically connects controlled devices to a central [smart home hub](https://en.wikipedia.org/wiki/Smart_home_hub) (sometimes called a "[gateway](https://en.wikipedia.org/wiki/Residential_gateway)"). The [user interface](https://en.wikipedia.org/wiki/User_interface) for control of the system uses either wall-mounted terminals, tablet or desktop computers, a mobile phone application, or a Web interface that may also be accessible off-site through the Internet.

While there are many competing vendors, there are increasing efforts toward open-source systems. However, there are issues with the current state of home automation including a lack of standardized security measures and deprecation of older devices without backward compatibility.

Home automation has high a potential for sharing data between family members or trusted individuals for personal security and could lead to energy-saving measures with a positive environmental impact in the future.

Smart home security utilizes a collection of IoT-enabled products in order to allow users to remotely monitor and manage the security of their homes. These systems can manage the surveillance in and around the home as well as who has access to the doors if they are equipped with [smart locks](https://www.iotforall.com/use-case/smart-locks/).

**INDEX**

Introduction ………………………………………………… Page 4

Components Used ………………………………………………… Page 5-6

Circuit Diagram ………………………………………………… Page 6

Problem Definition ………………………………………………… Page 7

Objective ………………………………………………… Page 8

Methodology ………………………………………………… Page 9

Results and Discussion .……………………………………………….. Page 10

Conclusion and Future Scope ………………………………………………… Page 11

References ………………………………………………… Page 11

Appendix ………………………………………………… Page 12-21

**INTRODUCTION**

Nowadays, we have remote controls for our television sets and other electronic systems, which have made our lives easy. Have you ever wondered about home automation which would give the facility of controlling tube lights, fans, and other electrical appliances at home using a remote control? Off-course, yes! But, are the available options cost-effective? If the answer is No, we have found a solution to it.

We have come up with a system called Arduino-based home automation using Bluetooth. This system is super-cost effective and can give the user, the ability to control any electronic device without even spending on the remote control. This project helps the user to control all the electronic devices using his/her smartphone. Time is a very valuable thing. Everybody wants to save time as much as they can. New technologies are being introduced to save time. To save people’s time we are introducing a Home Automation system using Bluetooth. With the help of this system, you can control your home appliances from your mobile phone. You can turn on/off your home appliances within the range of Bluetooth.

**COMPONENTS USED**

**Arduino:**

****

This is the microcontroller for which the rfid

sensor is attached and the vcc and gnd pins

are connected to the nodemcu so that it

powers the nodemcu also.

**ESP8266 – NodeMCU:**

****

NodeMCU is an open-source platform based on ESP8266 which can connect objects and let data transfer using the Wi-Fi protocol. In addition, by providing some of the most important features of microcontrollers such as GPIO, PWM, ADC, etc., it can solve many of the project's needs alone.

**ESP32-CAM module:**

****

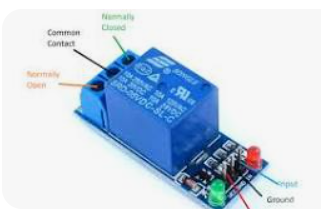
ESP32-CAM is a low-cost ESP32-based development board with an onboard camera, small. It is an ideal solution for IoT application, prototypes constructions, and DIY projects. The board integrates Wi-Fi, traditional Bluetooth, and low-power BLE, with 2 high-performance 32-bit LX6 CPUs.

**RFID sensor:**

****

Sensors and RFID System. RFID is short for “radio-frequency identification” and points to a technology whereby a reader catches digital information encoded in RFID tags or smart labels via radio waves.

**Relay module:**



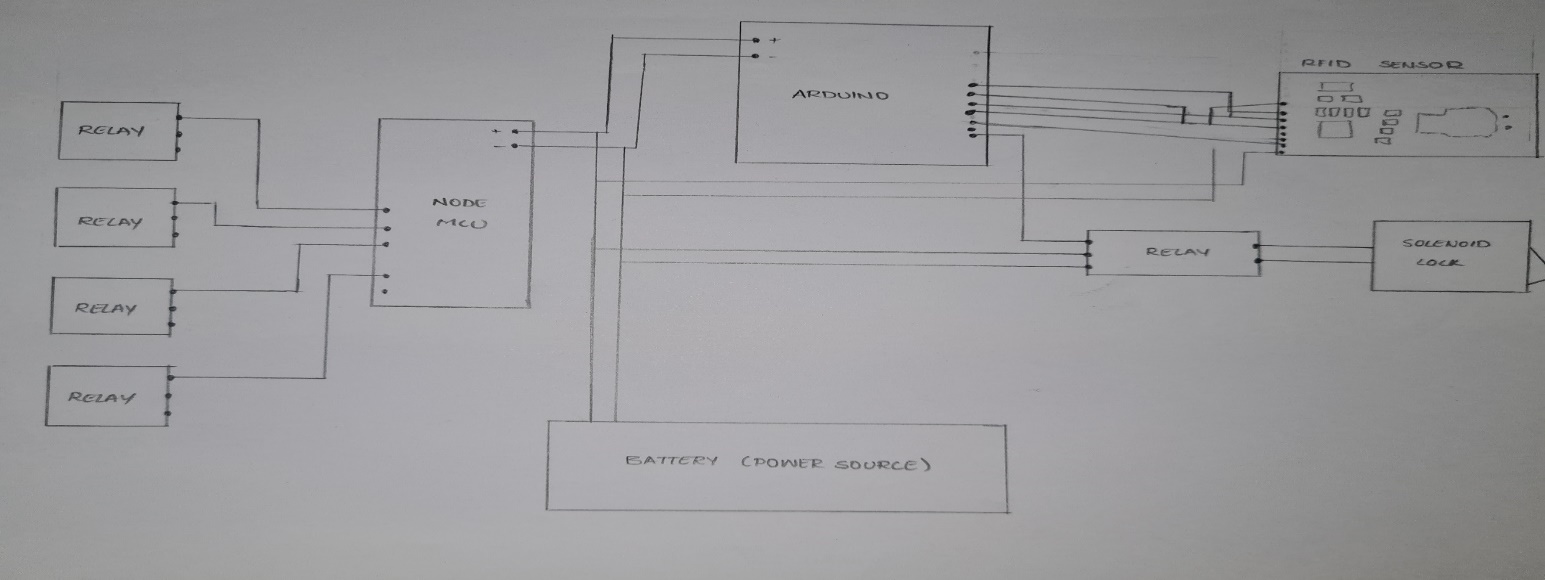
A power relay module is an electrical switch that is operated by an electromagnet. The electromagnet is activated by a separate low-power signal from a microcontroller. When activated, the electromagnet pulls to either open or close an electrical circuit.

**Solenoid lock:**



The solenoid lock denotes a latch for electrical locking and unlocking. It is available in unlocking in the power-on mode type and locking and keeping in the power-on mode type, which can be used selectively for situations. The power-on unlocking type enables unlocking only while the solenoid is powered on.

**CIRCUIT DIAGRAM:**

****

**PROBLEM DEFINITION:**

Household activities are automated by the development of special appliances such as water heaters to reduce the time taken to boil water for bathing and automatic washing machines to reduce the manual labor of washing clothes. In developed countries, homes are wired for electrical power, doorbell, TV outlets, and telephones. The different application includes when a person enters the room, the light turns on. With advanced technology, the room can sense the presence of the person and who the person is.

Taking into account the day of the week, time of the day, and other such factors it can also set apt lighting, temperature levels, television channels, or music levels. In the case of a smoke detector when fire or smoke is detected, the lights in the entire house begin to blink to alert the resident to the probable fire. In the case of a home theatre, the home automation system can avoid distraction and lock the audio and video components, and can also make an announcement. The home automation system can also dial up the house owner on their mobile phone to alert them or call any alarm monitoring company.

**OBJECTIVE:**

The main objective of this project is to develop a home automation system using an Arduino board with Bluetooth to be remotely controlled by any Android OS smartphone. As technology is advancing so houses are also getting smarter. Modern houses are gradually shifting from conventional switches to a centralized control system, involving remote-controlled switches. Presently, conventional wall switches located in different parts of the house make it difficult for the user to go near them to operate. Even more, it becomes more difficult for the elderly or physically handicapped people to do so.

The remote-controlled home automation system provides a most modern solution with smartphones. In order to achieve this, a Bluetooth module is interfaced with the Arduino board at the receiver end while on the transmitter end, a GUI application on the cell phone sends ON/OFF commands to the receiver where loads are connected. By touching the specified location on the GUI, the loads can be turned ON/OFF remotely through this technology.

**Methodology/Procedure:**

The proposed methodology for developing home automation and security.

* An Arduino, RFID, Relays, Solenoid Lock, ESP32 Cam module, and a NodeMCU are used in this project.
* A 5 V battery power source is connected to the NodeMCU. This 5V battery also supplies power to the Arduino. Relays are connected to the NodeMCU.
* The working of the circuit goes like when the NodeMCU has a value of one (1), the circuit is closed in the relay and the connection is made with loads like bulbs, lights, and fans. If the circuit is closed the light glows and if the NodeMCU has a value of zero (0) then the connection is an open circuit which indicates the light does not glow.
* On the other hand, Arduino is also connected to an RFID sensor (which is used for security purposes).
* The NFC card is for the main authentication.
* The NFC card is also connected to a relay and the relay is connected to a solenoid lock.
* The NFC card is the primary card, it helps in opening the solenoid lock which opens for 3 seconds and opens the door, and then closes automatically. If any other NFC card is placed instead of the key NFC card the LED glows red and a buzzer sound is generated which is an alarm that warns the members of the home that, there is someone else trying to intrude in the home
* An esp32 module is also connected to this home automation which is used for motion detection and image capturing.
* The ESP32 has a camera module and inbuilt Wi-Fi, whenever there is some movement, it captures the photo and sends it to the telegram app.
* The NodeMCU is connected to Sinric Pro which allows the connection of three devices we have taken that is a fan and two lights.
* A token is generated and the related code was dumped.
* So, when we on/off the switch from Alexa or google, NodeMCU gets the information through Wi-Fi and it triggers the relay (which acts like a switch) which when turned on passes the current and doesn’t pass any current when it is turned off.
* So, whenever an external Trigger is given to NodeMCU (0 or 1) accordingly the relays are in open or closed condition.
* Hence the design of home automation and security is in such a way that it captures the motion and provides security using the RFID sensor

**Results and Discussion**

****

* If the wrong key is used, it shows a red light, which signals us that the wrong key is used**.**

****

* If the right key is used the lock opens.
* In this way, it secures the house.

**Conclusion and Future Scope:**

This project presented is a low-cost and flexible home control and monitoring system using Node MCU Board with internet and various sensors remotely controlled by Android OS smartphone. In this, the Node MCU microcontroller is used as an interface between the user and hardware components. It is programmed and connected to several components according to the requirements. A microweb server is used as an application layer for communication between remote users, home devices, and security systems. This entire system of communication is enabled through the internet. Notifications are sent to users through the app ALEXA installed on smartphones. Users can operate wirelessly or home appliances. All these together form a completely capable, flexible smart home control and monitoring system.

the RFID Door Lock system in hotels, offices, and many other places where you just have to place the card near the RFID reader for a second, and the door will be opened.

**References:**

[1] Vishwajeet Hari Bhide, Dr.Sanjeev Wagh,” i-Learning loT: An Intelligent Self Learning System for Home Automation Using loT”, International Conference on Communication and Signal Processing, pp. 1763-1767, 2015.

[2] Ian G smith, “The Internet of things” NewHorizons, IERC-Internet of things European Research cluster, 2012.

[3] Vikram. N, Harish.k, Nihaal.M, Raksha Umesh, Shetty Aashik Ashok Kumar, “A Low-Cost Home Automation System Using Wi-Fi Based Wireless Sensor Network Incorporating Internet of Things (IoT)”, IEEE 7th International Advance Computing Conference, pp. 174-178, 2017. [4] Zeinab Kamal Aldein Mohammed, Elmustafa Sayed Ali Ahmed,” Internet of Things Applications, Challenges and Related Future Technologies”, pp. 126- 148, 2017.

[5] Kumar Mandula, Ramu Parupalli, CH.A.S.Murty, E.Magesh, Rutul Lunagariya, ” Mobile based Horne Automation using Internet of Things(IoT)”, International Conference on Control,instrumentation, Communication and Computational Technologies (lCCICCT) ,pp. 340-343, 2016.

[6] Himanshu Singh, Vishal Pallagani†, Vedant Khandelwal, Venkanna U. “IoT based Smart Home Automation System using Sensor Node”, 2018.

[7] Ravi Kishore Kodali, Vishal Jain, Suvadeep Bose and Lakshmi Boppana. “IoT Based Smart Security and Home Automation System”, pp. 1286-1289, 2016

[8] Homera Durani, Mitul Sheth, Madhuri Vaghasia, Shyam Kotech, “ Smart Automated Home Application using IoT with Blynk App”, 2nd International Conference on Inventive Communication and Computational Technologies (ICICCT 2018), pp. 393-397, September 2018.

[9] Yong Tae Park, Pranesh Sthapit, Jae-Young Pyun, “Smart Digital Door Lock for the Home Automation”, IEEE, pp. 1-6, TENCON 2009.

[10] F. M. G. K. D. Sukmana, Husni Teja, “Wireless and mobile (apwimob), IEEE asia pacific conference on,” pp. 183 – 187, 2015.

[11] G. Mahalakshmi, M.Vigneshwaran, “IOT Based Home Automation Using Arduino”, International Journal of Engineering and Advanced Research Technology (IJEART), pp. 7-11, 2017.

**APPENDIX:**

**CODES:**

**CODE1:**

//#define ENABLE\_DEBUG

#ifdef ENABLE\_DEBUG

#define DEBUG\_ESP\_PORT Serial

#define NODEBUG\_WEBSOCKETS

#define NDEBUG

#endif

#include <Arduino.h>

#include <ESP8266WiFi.h>

#include "SinricPro.h"

#include "SinricProSwitch.h"

#include <map>

#define WIFI\_SSID "YOUR-WIFI-NAME"

#define WIFI\_PASS "YOUR-WIFI-PASSWORD"

#define APP\_KEY "YOUR-APP-KEY" // Should look like "de0bxxxx-1x3x-4x3x-ax2x-5dabxxxxxxxx"

#define APP\_SECRET "YOUR-APP-SECRET" // Should look like "5f36xxxx-x3x7-4x3x-xexe-e86724a9xxxx-4c4axxxx-3x3x-x5xe-x9x3-333d65xxxxxx"

//Enter the device IDs here

#define device\_ID\_1 "xxxxxxxxxxxxxxxxxxxxxxxx"

#define device\_ID\_2 "60764ab82fb4f14a3bedebfc"

#define device\_ID\_3 "60764ac948ccc14a4674c049"

#define device\_ID\_4 "60764aa148ccc14a4674c047"

// define the GPIO connected with Relays and switches

#define RelayPin1 5 //D1

#define RelayPin2 4 //D2

#define RelayPin3 14 //D5

#define RelayPin4 12 //D6

#define SwitchPin1 10 //SD3

#define SwitchPin2 0 //D3

#define SwitchPin3 13 //D7

#define SwitchPin4 3 //RX

#define wifiLed 16 //D0

// comment the following line if you use a toggle switches instead of tactile buttons

//#define TACTILE\_BUTTON 1

#define BAUD\_RATE 9600

#define DEBOUNCE\_TIME 250

typedef struct { // struct for the std::map below

int relayPIN;

int flipSwitchPIN;

} deviceConfig\_t;

// this is the main configuration

// please put in your deviceId, the PIN for Relay and PIN for flipSwitch

// this can be up to N devices...depending on how much pin's available on your device ;)

// right now we have 4 devicesIds going to 4 relays and 4 flip switches to switch the relay manually

std::map<String, deviceConfig\_t> devices = {

//{deviceId, {relayPIN, flipSwitchPIN}}

{device\_ID\_1, { RelayPin1, SwitchPin1 }},

{device\_ID\_2, { RelayPin2, SwitchPin2 }},

{device\_ID\_3, { RelayPin3, SwitchPin3 }},

{device\_ID\_4, { RelayPin4, SwitchPin4 }}

};

typedef struct { // struct for the std::map below

String deviceId;

bool lastFlipSwitchState;

unsigned long lastFlipSwitchChange;

} flipSwitchConfig\_t;

std::map<int, flipSwitchConfig\_t> flipSwitches; // this map is used to map flipSwitch PINs to deviceId and handling debounce and last flipSwitch state checks

// it will be setup in "setupFlipSwitches" function, using informations from devices map

void setupRelays() {

for (auto &device : devices) { // for each device (relay, flipSwitch combination)

int relayPIN = device.second.relayPIN; // get the relay pin

pinMode(relayPIN, OUTPUT); // set relay pin to OUTPUT

digitalWrite(relayPIN, HIGH);

}

}

void setupFlipSwitches() {

for (auto &device : devices) { // for each device (relay / flipSwitch combination)

flipSwitchConfig\_t flipSwitchConfig; // create a new flipSwitch configuration

flipSwitchConfig.deviceId = device.first; // set the deviceId

flipSwitchConfig.lastFlipSwitchChange = 0; // set debounce time

flipSwitchConfig.lastFlipSwitchState = true; // set lastFlipSwitchState to false (LOW)--

int flipSwitchPIN = device.second.flipSwitchPIN; // get the flipSwitchPIN

flipSwitches[flipSwitchPIN] = flipSwitchConfig; // save the flipSwitch config to flipSwitches map

pinMode(flipSwitchPIN, INPUT\_PULLUP); // set the flipSwitch pin to INPUT

}

}

bool onPowerState(String deviceId, bool &state)

{

Serial.printf("%s: %s\r\n", deviceId.c\_str(), state ? "on" : "off");

int relayPIN = devices[deviceId].relayPIN; // get the relay pin for corresponding device

digitalWrite(relayPIN, !state); // set the new relay state

return true;

}

void handleFlipSwitches() {

unsigned long actualMillis = millis(); // get actual millis

for (auto &flipSwitch : flipSwitches) { // for each flipSwitch in flipSwitches map

unsigned long lastFlipSwitchChange = flipSwitch.second.lastFlipSwitchChange; // get the timestamp when flipSwitch was pressed last time (used to debounce / limit events)

if (actualMillis - lastFlipSwitchChange > DEBOUNCE\_TIME) { // if time is > debounce time...

int flipSwitchPIN = flipSwitch.first; // get the flipSwitch pin from configuration

bool lastFlipSwitchState = flipSwitch.second.lastFlipSwitchState; // get the lastFlipSwitchState

bool flipSwitchState = digitalRead(flipSwitchPIN); // read the current flipSwitch state

if (flipSwitchState != lastFlipSwitchState) { // if the flipSwitchState has changed...

#ifdef TACTILE\_BUTTON

if (flipSwitchState) { // if the tactile button is pressed

#endif

flipSwitch.second.lastFlipSwitchChange = actualMillis; // update lastFlipSwitchChange time

String deviceId = flipSwitch.second.deviceId; // get the deviceId from config

int relayPIN = devices[deviceId].relayPIN; // get the relayPIN from config

bool newRelayState = !digitalRead(relayPIN); // set the new relay State

digitalWrite(relayPIN, newRelayState); // set the trelay to the new state

SinricProSwitch &mySwitch = SinricPro[deviceId]; // get Switch device from SinricPro

mySwitch.sendPowerStateEvent(!newRelayState); // send the event

#ifdef TACTILE\_BUTTON

}

#endif

flipSwitch.second.lastFlipSwitchState = flipSwitchState; // update lastFlipSwitchState

}

}

}

}

void setupWiFi()

{

Serial.printf("\r\n[Wifi]: Connecting");

WiFi.begin(WIFI\_SSID, WIFI\_PASS);

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

{

Serial.printf(".");

delay(250);

}

digitalWrite(wifiLed, LOW);

Serial.printf("connected!\r\n[WiFi]: IP-Address is %s\r\n", WiFi.localIP().toString().c\_str());

}

void setupSinricPro()

{

for (auto &device : devices)

{

const char \*deviceId = device.first.c\_str();

SinricProSwitch &mySwitch = SinricPro[deviceId];

mySwitch.onPowerState(onPowerState);

}

SinricPro.begin(APP\_KEY, APP\_SECRET);

SinricPro.restoreDeviceStates(true);

}

void setup()

{

Serial.begin(BAUD\_RATE);

pinMode(wifiLed, OUTPUT);

digitalWrite(wifiLed, HIGH);

setupRelays();

setupFlipSwitches();

setupWiFi();

setupSinricPro();

}

void loop()

{

SinricPro.handle();

handleFlipSwitches();

}

**CODE2:**

#include <WiFi.h>

#include <WiFiClientSecure.h>

#include "soc/soc.h"

#include "soc/rtc\_cntl\_reg.h"

#include "esp\_camera.h"

#include <UniversalTelegramBot.h>

#include <ArduinoJson.h>

#include <Wire.h>

#include "SparkFunBME280.h"

// Replace with your network credentials

const char\* ssid = "Xavier";

const char\* password = "xavier123";

// 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

String chatId = "1088289091";

// Initialize Telegram BOT

String BOTtoken = "5645030447:AAHaptar4iG-d1VWihierzmnyq-ljv602YU";

bool sendPhoto = false;

WiFiClientSecure clientTCP;

UniversalTelegramBot bot(BOTtoken, clientTCP);

//CAMERA\_MODEL\_AI\_THINKER

#define PWDN\_GPIO\_NUM 32

#define RESET\_GPIO\_NUM -1

#define XCLK\_GPIO\_NUM 0

#define SIOD\_GPIO\_NUM 26

#define SIOC\_GPIO\_NUM 27

#define Y9\_GPIO\_NUM 35

#define Y8\_GPIO\_NUM 34

#define Y7\_GPIO\_NUM 39

#define Y6\_GPIO\_NUM 36

#define Y5\_GPIO\_NUM 21

#define Y4\_GPIO\_NUM 19

#define Y3\_GPIO\_NUM 18

#define Y2\_GPIO\_NUM 5

#define VSYNC\_GPIO\_NUM 25

#define HREF\_GPIO\_NUM 23

#define PCLK\_GPIO\_NUM 22

#define FLASH\_LED\_PIN 4

bool flashState = LOW;

// Motion Sensor

bool motionDetected = false;

// Define I2C Pins for BME280

#define I2C\_SDA 14

#define I2C\_SCL 15

BME280 bme;

int botRequestDelay = 1000; // mean time between scan messages

long lastTimeBotRan; // last time messages' scan has been done

void handleNewMessages(int numNewMessages);

String sendPhotoTelegram();

// Get BME280 sensor readings and return them as a String variable

String getReadings(){

float temperature, humidity;

temperature = bme.readTempC();

//temperature = bme.readTempF();

humidity = bme.readFloatHumidity();

String message = "Temperature: " + String(temperature) + " ºC \n";

message += "Humidity: " + String (humidity) + " % \n";

return message;

}

// Indicates when motion is detected

static void IRAM\_ATTR detectsMovement(void \* arg){

//Serial.println("MOTION DETECTED!!!");

motionDetected = true;

}

void setup(){

WRITE\_PERI\_REG(RTC\_CNTL\_BROWN\_OUT\_REG, 0);

Serial.begin(115200);

pinMode(FLASH\_LED\_PIN, OUTPUT);

digitalWrite(FLASH\_LED\_PIN, flashState);

// Init BME280 sensor

Wire.begin(I2C\_SDA, I2C\_SCL);

bme.settings.commInterface = I2C\_MODE;

bme.settings.I2CAddress = 0x76;

bme.settings.runMode = 3;

bme.settings.tStandby = 0;

bme.settings.filter = 0;

bme.settings.tempOverSample = 1;

bme.settings.pressOverSample = 1;

bme.settings.humidOverSample = 1;

bme.begin();

WiFi.mode(WIFI\_STA);

Serial.println();

Serial.print("Connecting to ");

Serial.println(ssid);

WiFi.begin(ssid, password);

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

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

Serial.print(".");

delay(500);

}

Serial.println();

Serial.print("ESP32-CAM IP Address: ");

Serial.println(WiFi.localIP());

camera\_config\_t config;

config.ledc\_channel = LEDC\_CHANNEL\_0;

config.ledc\_timer = LEDC\_TIMER\_0;

config.pin\_d0 = Y2\_GPIO\_NUM;

config.pin\_d1 = Y3\_GPIO\_NUM;

config.pin\_d2 = Y4\_GPIO\_NUM;

config.pin\_d3 = Y5\_GPIO\_NUM;

config.pin\_d4 = Y6\_GPIO\_NUM;

config.pin\_d5 = Y7\_GPIO\_NUM;

config.pin\_d6 = Y8\_GPIO\_NUM;

config.pin\_d7 = Y9\_GPIO\_NUM;

config.pin\_xclk = XCLK\_GPIO\_NUM;

config.pin\_pclk = PCLK\_GPIO\_NUM;

config.pin\_vsync = VSYNC\_GPIO\_NUM;

config.pin\_href = HREF\_GPIO\_NUM;

config.pin\_sscb\_sda = SIOD\_GPIO\_NUM;

config.pin\_sscb\_scl = SIOC\_GPIO\_NUM;

config.pin\_pwdn = PWDN\_GPIO\_NUM;

config.pin\_reset = RESET\_GPIO\_NUM;

config.xclk\_freq\_hz = 20000000;

config.pixel\_format = PIXFORMAT\_JPEG;

//init with high specs to pre-allocate larger buffers

if(psramFound()){

config.frame\_size = FRAMESIZE\_UXGA;

config.jpeg\_quality = 10; //0-63 lower number means higher quality

config.fb\_count = 2;

} else {

config.frame\_size = FRAMESIZE\_SVGA;

config.jpeg\_quality = 12; //0-63 lower number means higher quality

config.fb\_count = 1;

}

// camera init

esp\_err\_t err = esp\_camera\_init(&config);

if (err != ESP\_OK) {

Serial.printf("Camera init failed with error 0x%x", err);

delay(1000);

ESP.restart();

}

// Drop down frame size for higher initial frame rate

sensor\_t \* s = esp\_camera\_sensor\_get();

s->set\_framesize(s, FRAMESIZE\_CIF); // UXGA|SXGA|XGA|SVGA|VGA|CIF|QVGA|HQVGA|QQVGA

// PIR Motion Sensor mode INPUT\_PULLUP

//err = gpio\_install\_isr\_service(0);

err = gpio\_isr\_handler\_add(GPIO\_NUM\_13, &detectsMovement, (void \*) 13);

if (err != ESP\_OK){

Serial.printf("handler add failed with error 0x%x \r\n", err);

}

err = gpio\_set\_intr\_type(GPIO\_NUM\_13, GPIO\_INTR\_POSEDGE);

if (err != ESP\_OK){

Serial.printf("set intr type failed with error 0x%x \r\n", err);

}

}

void loop(){

if (sendPhoto){

Serial.println("Preparing photo");

sendPhotoTelegram();

sendPhoto = false;

}

if(motionDetected){

bot.sendMessage(chatId, "Motion detected!!", "");

Serial.println("Motion Detected");

sendPhotoTelegram();

motionDetected = false;

}

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();

}

}

String sendPhotoTelegram(){

const char\* myDomain = "api.telegram.org";

String getAll = "";

String getBody = "";

camera\_fb\_t \* fb = NULL;

fb = esp\_camera\_fb\_get();

if(!fb) {

Serial.println("Camera capture failed");

delay(1000);

ESP.restart();

return "Camera capture failed";

}

Serial.println("Connect to " + String(myDomain));

if (clientTCP.connect(myDomain, 443)) {

Serial.println("Connection successful");

String head = "--RandomNerdTutorials\r\nContent-Disposition: form-data; name=\"chat\_id\"; \r\n\r\n" + chatId + "\r\n--RandomNerdTutorials\r\nContent-Disposition: form-data; name=\"photo\"; filename=\"esp32-cam.jpg\"\r\nContent-Type: image/jpeg\r\n\r\n";

String tail = "\r\n--RandomNerdTutorials--\r\n";

uint16\_t imageLen = fb->len;

uint16\_t extraLen = head.length() + tail.length();

uint16\_t totalLen = imageLen + extraLen;

clientTCP.println("POST /bot"+BOTtoken+"/sendPhoto HTTP/1.1");

clientTCP.println("Host: " + String(myDomain));

clientTCP.println("Content-Length: " + String(totalLen));

clientTCP.println("Content-Type: multipart/form-data; boundary=RandomNerdTutorials");

clientTCP.println();

clientTCP.print(head);

uint8\_t \*fbBuf = fb->buf;

size\_t fbLen = fb->len;

for (size\_t n=0;n<fbLen;n=n+1024) {

if (n+1024<fbLen) {

clientTCP.write(fbBuf, 1024);

fbBuf += 1024;

}

else if (fbLen%1024>0) {

size\_t remainder = fbLen%1024;

clientTCP.write(fbBuf, remainder);

}

}

clientTCP.print(tail);

esp\_camera\_fb\_return(fb);

int waitTime = 10000; // timeout 10 seconds

long startTimer = millis();

boolean state = false;

while ((startTimer + waitTime) > millis()){

Serial.print(".");

delay(100);

while (clientTCP.available()) {

char c = clientTCP.read();

if (state==true) getBody += String(c);

if (c == '\n') {

if (getAll.length()==0) state=true;

getAll = "";

}

else if (c != '\r')

getAll += String(c);

startTimer = millis();

}

if (getBody.length()>0) break;

}

clientTCP.stop();

Serial.println(getBody);

}

else {

getBody="Connected to api.telegram.org failed.";

Serial.println("Connected to api.telegram.org failed.");

}

return getBody;

}

void handleNewMessages(int numNewMessages){

Serial.print("Handle New Messages: ");

Serial.println(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 != chatId){

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

continue;

}

// Print the received message

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

Serial.println(text);

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

if (text == "/flash") {

flashState = !flashState;

digitalWrite(FLASH\_LED\_PIN, flashState);

}

if (text == "/photo") {

sendPhoto = true;

Serial.println("New photo request");

}

if (text == "/readings"){

String readings = getReadings();

bot.sendMessage(chatId, readings, "");

}

if (text == "/start"){

String welcome = "Welcome to the ESP32-CAM Telegram bot.\n";

welcome += "/photo : takes a new photo\n";

welcome += "/flash : toggle flash LED\n";

welcome += "/readings : request sensor readings\n\n";

welcome += "You'll receive a photo whenever motion is detected.\n";

bot.sendMessage(chatId, welcome, "Markdown");

}

}

}