



JAIN
DEEMED-TO-BE UNIVERSITY

FACULTY OF
ENGINEERING
AND TECHNOLOGY

A project synopsis on

“Surveillance System Using ESP-32 Cam”

Submitted in partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY (HONOURS)

IN

COMPUTER SCIENCE AND ENGINEERING

(DATA SCIENCE)

Submitted by

Teella Dheeraj

Naidu

18BTRCY011

Kota Narsimha

Reddy

18BTRCY006

Harshith Guptha

18BTRCY026

Under the guidance of

Dr. Panduranga rao

Asst Professor

Faculty of Engineering & Technology

Jain (Deemed-To-Be University)

Department of Computer Science & Engineering

Jain Global Campus, Kanakapura Taluk - 562112

Ramanagara District, Karnataka, India

2020-2021

INTRODUCTION

Today many homeowners have a home surveillance/monitoring system. Traditionally these systems have been built in an ad hoc fashion with direct wired connections between the control center and all of the sensors. This is changing due to the use of local area network technology for the interconnections (be they wired or wireless) and the fact that the control system is increasingly connected to the Internet. The connection to the Internet enables home owners (and potentially others) to access information collected by the home security and monitoring system from any place in the Internet.

We have designed an interesting and cheap home surveillance system. This Gadget helps you to protect your house from thieves. In this project we are going to use an ESP32-CAM, P.I.R Sensor module, Blynk app and some other components. This Project can either powered with 5V Battery or with U.S.B of your computer. This is a basic motion-sensing alarm that detects when someone enters the area. When an intruder is detected, it activates the camera. Our body generates heat energy in the form of infrared which is invisible to human eyes. But it can be detected by electronic sensor. This type of sensor is made up of crystalline material that is Pyroelectric. In this project, we are using P.I.R. Motion Sensor Module as an infrared sensor that generates electric charge when exposed in heat and sends a signal to ESP32-CAM. We can use this system to remotely monitor our home simply through our smartphone. We can not only get the pictures taken by ESP32-CAM when a motion is detected, but also take multiple pictures through the phone remotely.

Literature Survey

We have found different papers related to the security systems. Based on the purpose of the security camera, various systems are proposed and used. Ms. Sushma .N. Nichal and Prof. J.K. Singhas done abstraction of Smart supervisor system using IOT based on embedded Linux O.S. with ARM11 architecture. In this paper they have implemented real-time image monitoring system and acquired data. They have also used PIR sensor. The system first requires authentication from user to activate the system if the system detect human it will send that data to the server or user smart phone.[1]

Shetel and Agarwal (2016) explains in their paper that IoT enables internet connectivity for all kind of devices and physical objects in real time system. The virtualization of this system enables to perform activities without direct physical synchronization between the devices. The IoT enables to manage multiple jobs without any limitation of distances with the help of intelligent devices and high-speed network.[2]

Gill et al. (2009) explains network enabled digital technology is rapidly introduced in the home automation. For the purpose of home automation this technology introduces new and existing opportunities to increase the connectivity of the devices. The remote-control technology is rapidly synchronizing with the expansion of Internet.[3]

Lee et al. (2017) explains in their paper the web of physical objects is Internet of Thing which contains the embedded technology helping in developing machine to machine or man to machine communication. This paper provides a dynamic data sheet about the city environment parameters taken from the stand-alone system.[3]

Sahadevan et al. (2017) explains in their paper how the Internet of Things is amazingly impacting the attention of consumers and the enterprise electronics market rapidly implementing in home automation, smart cities, automated industries, etc. To build these applications many power efficient and low cost sensors are available in the market for the developers.[4]

Limitation of Existing Systems

The already existing home surveillance systems are outdated, even if some of them used Internet of Things to make an automated system. The hardware and software tools which were used are not that effective and also consumes a lot of power. The earlier system is very expensive and has poor stability. It is not an efficient system. It is a low intelligence system. It provides weak security. These are the limitations of the existing system.

- Being fully wired, the connection can be interrupted with a single loose cable.
- Easy for burglars to chop off the wires.
- Limited access for homeowners.
- Less technological advancements.

- Hassle of manual planning and management.

Problem Definition

Due to the increase of crime rate and fire/gas accidents in households, it became a necessity to use a home surveillance camera. But the surveillance kits available on the market are expensive, complex, hard to install and maintain by ourselves. So there is a need for home surveillance systems, which are affordable and easy to use by anyone.

Today these systems are an intelligent product integrating multiple functions, and future developments will port the user interface to different terminals -- enabling people to better manage their home and do it more easily than they can do at present. The following subsections will describe the structure of the current product whose further development is the focus of this thesis.

Objectives

The objective of this paper is home security using ESP32-CAM through IoT. Surveilling your own house for security purposes shouldn't cost you a fortune. Even if you find a cheaper version, it may not work properly. So the main objective is to make a simple, easy to use, cheaper, efficient home surveilling system.

- To make an efficient home surveillance camera.
- It should be cost effective, to be used by common households.
- It should be easily accessible through smart phones.

Proposed System

In this paper we are proposing a Home Security System where it senses any intruders, takes picture, and sends directly to the smartphone. We are using ESP32-CAM and Internet of Things for this system. If some unauthorized person is coming near the home the camera will capture their photo and sends it to the phone using Wi-Fi and Blynk app. We are using a motion sensor which

detects motion. IoT helps in signaling ESP32-CAM when the motion sensor detects something to take a picture, and then helps in sending the images directly to the smartphone.

In order to solve the limitations of the existing home surveillance systems on the market, this IoT solution uses an Intelligent controller along with a reliable software to access the data. This project has advantages such as higher intelligence, higher stability, and easy installation. The system detects the intruder and immediately and notifies the owner using the Blynk app as a notification alert and also as an email.

Methodology

The methodology of this home surveillance system is very simple. We have to upload the code on to the AI thinker, and then connect it to a motion sensor module. When the sensor senses a motion, the camera will take a picture because of the code uploaded onto it. Then with the help of an app, we can get the pictures took by the camera directly onto our smartphone. We can surveil and take pictures remotely from our smartphones.

- Firstly we programmed the ESP32-CAM board by using FTDI 232 USB to Serial Interface board by connecting these two as in the circuit diagram shown below.
- Then we connected the above board to my laptop to upload the code onto ESP32-CAM.
- After uploading the code, we disconnected ESP32-CAM and then connected it to the PIR motion sensor module as in the circuit diagram shown below.
- When the PIR module senses a motion, the output will become high, and it will give the high pulse to BC547 NPN transistor.
- Then the transistor will turn on and will ground the GPIO 13.

When GPIO 13 is low the ESP32-CAM will start taking the picture and the LED will also glow whenever the transistor is turned on.

- The following flow chart Explains the process of the project:

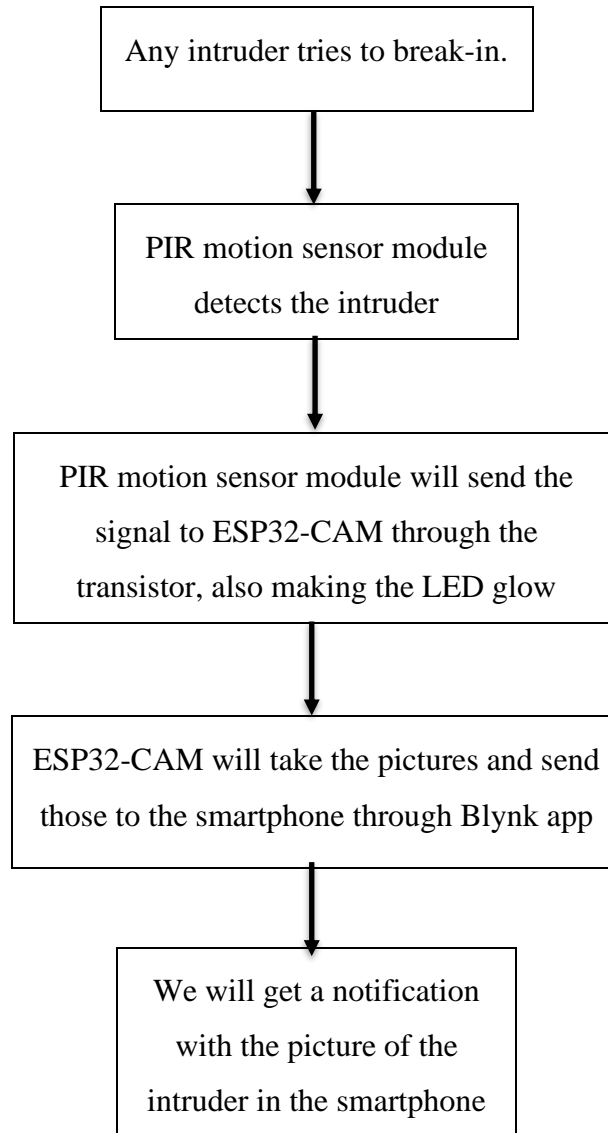


Figure 3.2.2: Flow Chart of the Methodology

We can also take pictures remotely from the smartphone through the Blynk app. We can also surveil the region through the smartphone, if you are connected to the same Wi-fi as that of this home surveilling system. The whole process of the project follows the above flow chart in order to accomplish the goal. It shows how both hardware tools and software tools work to achieve the same.

- The setup of ESP32-CAM with PIR motion sensor is connected to the breadboard as shown in the below figure.

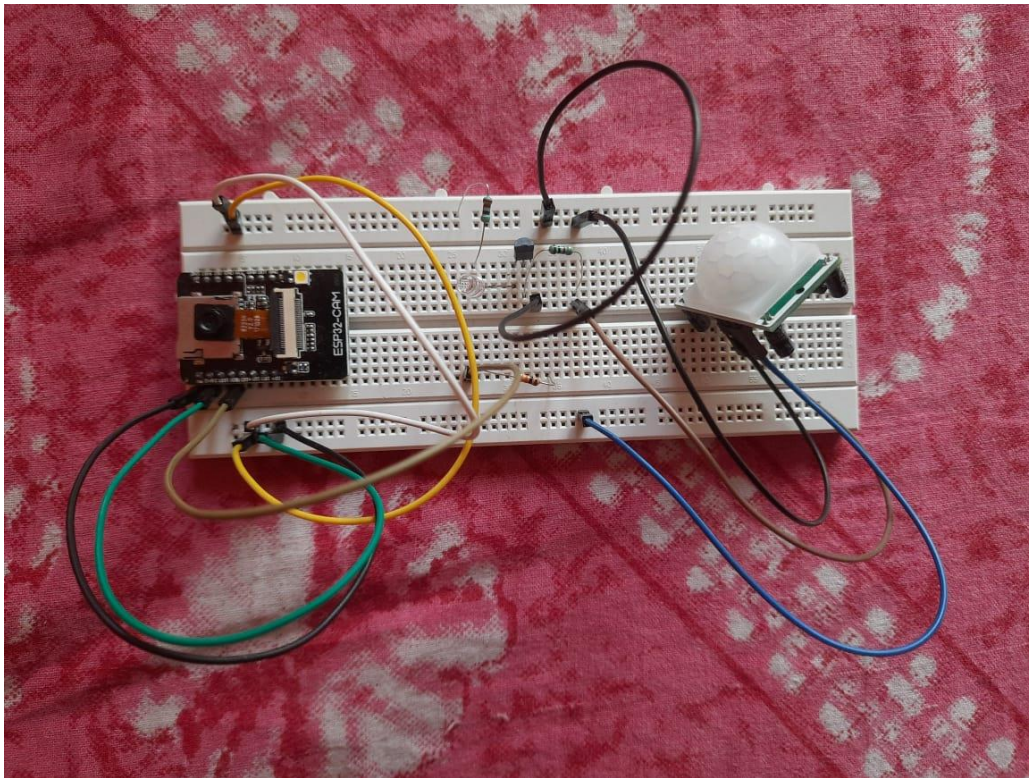


Figure 3.2.1: Circuit Connection

- The Block diagram is very simple. So you can easily make this motion sensor security camera using the ESP32 CAM board, PIR motion sensor.

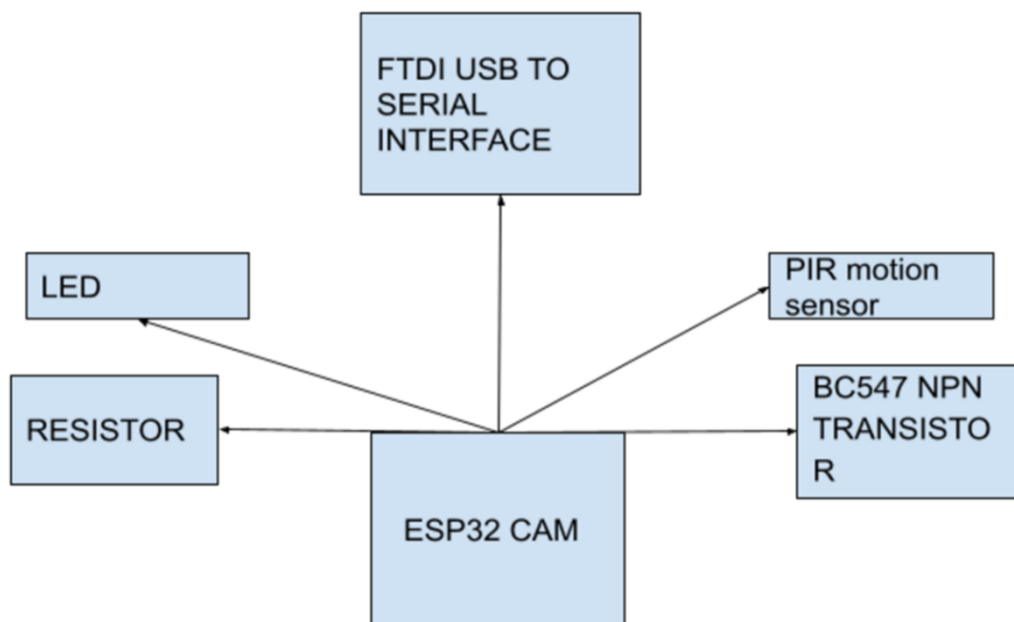


Figure 3.3.1: Sequence Diagram

Implementation

We have to upload the code onto ESP32-CAM for it to take pictures and send it to the smartphone through Blynk app. Before uploading the code to ESP32CAM, please check the following setting:

Update the Preferences-> Additional boards Manager URLs:

https://dl.espressif.com/dl/package_esp32_index.json,

http://arduino.esp8266.com/stable/package_esp8266com_index.json

• **Board Settings:**

Board: "ESP32 Wrover Module"

Upload Speed: "921600"

Flash Frequency: "80MHz"

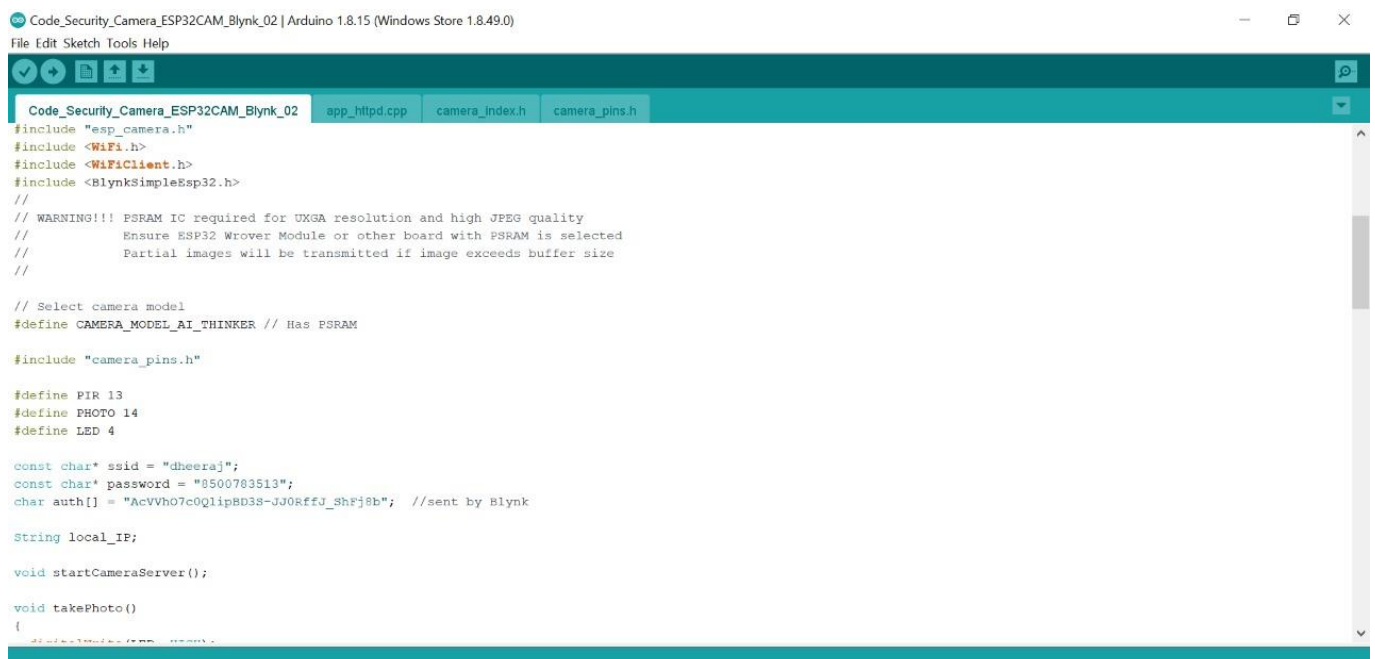
Flash Mode: "QIO"

Partition Scheme: "Hue APP (3MB No OTA/1MB SPIFFS)"

Core Debug Level: "None"

.COM Port: Depends On Your System

- **GPIO 0 must be connected to GND pin while uploading the sketch**
- **After connecting GPIO 0 to GND pin, press the ESP32 CAM on-board RESET button to put the board in flashing mode.**



```
Code_Security_Camera_ESP32CAM_Blynk_02 | Arduino 1.8.15 (Windows Store 1.8.49.0)
File Edit Sketch Tools Help

Code_Security_Camera_ESP32CAM_Blynk_02  app_httpd.cpp  camera_index.h  camera_pins.h

#include "esp_camera.h"
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>

//
// WARNING!!! PSRAM IC required for UXGA resolution and high JPEG quality
//           Ensure ESP32 Wrover Module or other board with PSRAM is selected
//           Partial images will be transmitted if image exceeds buffer size
//
//
// Select camera model
#define CAMERA_MODEL_AI_THINKER // Has PSRAM

#include "camera_pins.h"

#define PIR 13
#define PHOTO 14
#define LED 4

const char* ssid = "dheeraj";
const char* password = "8500783513";
char auth[] = "AcVvho7cQqlip8D3S-JJ0RffJ_ShFj9b"; //sent by Blynk

String local_IP;

void startCameraServer();

void takePhoto()
{
  // ...
}
```

Figure 5.1.1: Main Code


```
Code_Security_Camera_ESP32CAM_Blynk_02 - app_httpd.cpp | Arduino 1.8.15 (Windows Store 1.8.49.0)
File Edit Sketch Tools Help

Code_Security_Camera_ESP32CAM_Blynk_02 app_httpd.cpp camera_index.h camera_pins.h

// limitations under the License.
#include "esp_http_server.h"
#include "esp_timer.h"
#include "esp_camera.h"
#include "img_converters.h"
#include "camera_index.h"
#include "Arduino.h"

#include "fb_gfx.h"
#include "fd_forward.h"
#include "fr_forward.h"

#define ENROLL_CONFIRM_TIMES 5
#define FACE_ID_SAVE_NUMBER 7

#define FACE_COLOR_WHITE 0x00FFFFFF
#define FACE_COLOR_BLACK 0x00000000
#define FACE_COLOR_RED 0x000000FF
#define FACE_COLOR_GREEN 0x0000FF00
#define FACE_COLOR_BLUE 0x00FF0000
#define FACE_COLOR_YELLOW (FACE_COLOR_RED | FACE_COLOR_GREEN)
#define FACE_COLOR_CYAN (FACE_COLOR_BLUE | FACE_COLOR_GREEN)
#define FACE_COLOR_PURPLE (FACE_COLOR_BLUE | FACE_COLOR_RED)

typedef struct {
    size_t size; //number of values used for filtering
    size_t index; //current value index
    size_t count; //value count
    int sum;
}
```

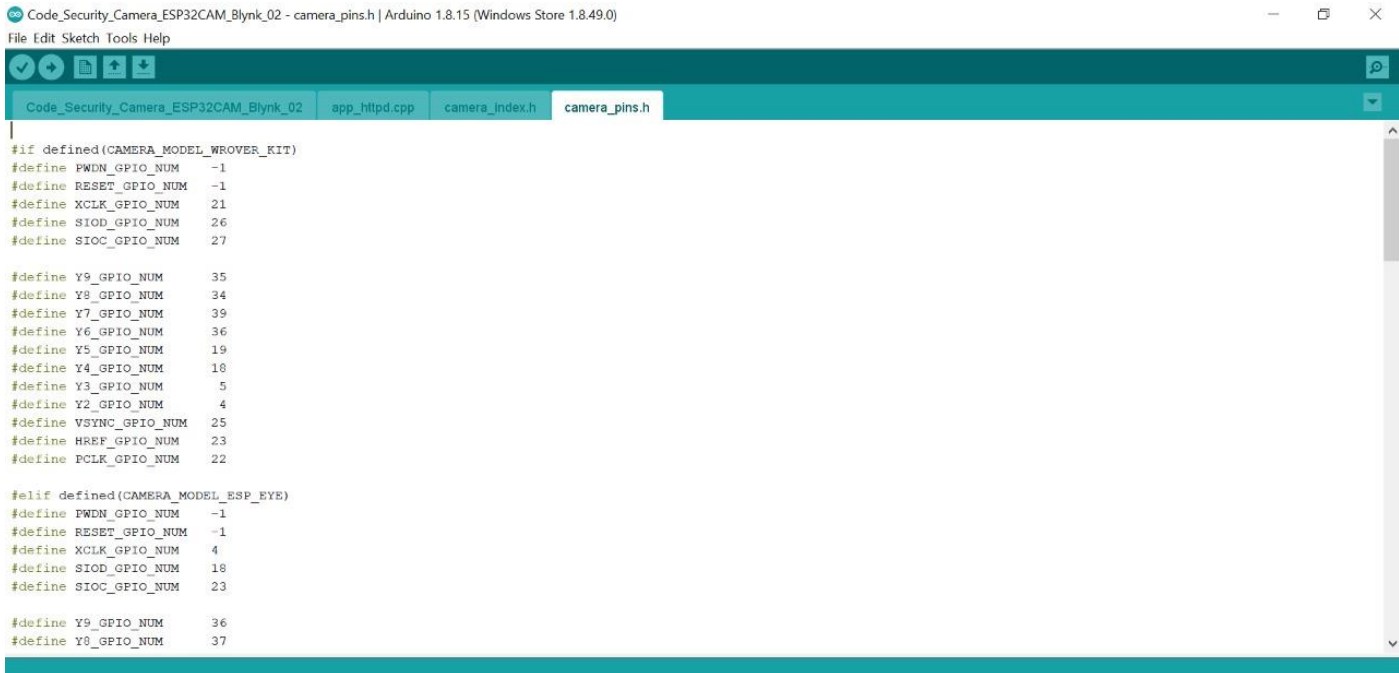
Figure 5.1.2: Code A

```
Code_Security_Camera_ESP32CAM_Blynk_02 - camera_index.h | Arduino 1.8.15 (Windows Store 1.8.49.0)
File Edit Sketch Tools Help

Code_Security_Camera_ESP32CAM_Blynk_02 app_httpd.cpp camera_index.h camera_pins.h

const uint8_t index_ov2640_html_gz[] = {
0x1F, 0x0B, 0x08, 0x08, 0x50, 0x5C, 0xAE, 0x5C, 0x00, 0x03, 0x69, 0x6E, 0x64, 0x65, 0x78, 0x5F,
0x6F, 0x76, 0x32, 0x36, 0x34, 0x30, 0x2E, 0x68, 0x74, 0x6D, 0x6C, 0x00, 0xE5, 0x5D, 0x7B, 0x73,
0xD3, 0xC6, 0x16, 0xFF, 0x9F, 0x4F, 0x21, 0x04, 0x25, 0xF6, 0x34, 0x76, 0x6C, 0xC7, 0x84, 0xE0,
0xDA, 0xE2, 0x42, 0x08, 0xD0, 0x19, 0x5E, 0x25, 0x2D, 0x74, 0xA6, 0xD3, 0x81, 0xB5, 0xB4, 0xB2,
0x55, 0x64, 0xC9, 0x95, 0x56, 0x76, 0x52, 0x26, 0x9F, 0x83, 0x7E, 0xA0, 0xFB, 0xC5, 0xEE, 0xD9,
0x87, 0xA4, 0x95, 0xBC, 0x7A, 0xD8, 0x26, 0x36, 0x97, 0xEB, 0xCC, 0x14, 0xD9, 0xDA, 0x73, 0xF6,
0x9C, 0xF3, 0x3B, 0xAF, 0x5D, 0x3D, 0x3A, 0xBC, 0x6D, 0xF9, 0x26, 0xB9, 0x9A, 0x63, 0x6D, 0x4A,
0x66, 0xAE, 0x71, 0x6B, 0xC8, 0xFF, 0xD1, 0xE0, 0x33, 0x9C, 0x62, 0x64, 0xF1, 0x43, 0xF6, 0x75,
0x86, 0x09, 0xD2, 0xCC, 0x29, 0x0A, 0x42, 0x4C, 0x46, 0x7A, 0x44, 0xEC, 0xD6, 0xA9, 0x9E, 0x3F,
0xED, 0xA1, 0x19, 0x1E, 0xE9, 0x0B, 0x07, 0x2F, 0xE7, 0x7E, 0x40, 0x74, 0xCD, 0xF4, 0x3D, 0x82,
0x3D, 0x18, 0xBE, 0x74, 0x2C, 0x32, 0x1D, 0x59, 0x78, 0xE1, 0x98, 0xB8, 0xC5, 0xBE, 0x1C, 0x3A,
0x9E, 0x43, 0x1C, 0xE4, 0xB6, 0x42, 0x13, 0xB9, 0x78, 0xD4, 0x95, 0x79, 0x11, 0x87, 0xB8, 0xD8,
0x38, 0xBF, 0x78, 0x7B, 0xDC, 0xD3, 0xDE, 0xBC, 0xEF, 0xF5, 0x4F, 0x3A, 0xC3, 0x23, 0xFE, 0x5B,
0x3A, 0x26, 0x24, 0x57, 0xF2, 0x77, 0xFA, 0x19, 0xFB, 0xD6, 0x95, 0xF6, 0x25, 0xF3, 0x13, 0xFD,
0xD8, 0x20, 0x44, 0xCB, 0x46, 0x33, 0xC7, 0xBD, 0x1A, 0x68, 0x8F, 0x03, 0x98, 0xF3, 0xF0, 0x05,
0x76, 0x17, 0x98, 0x38, 0x26, 0x3A, 0x0C, 0x91, 0x17, 0xB6, 0x42, 0x1C, 0x38, 0xF6, 0x4F, 0x2B,
0x84, 0x63, 0x64, 0x7E, 0x9E, 0x04, 0x7E, 0xE4, 0x59, 0x03, 0xED, 0x4E, 0xF7, 0x94, 0xFE, 0xAD,
0x0E, 0x32, 0x7D, 0xD7, 0x0F, 0xE0, 0xFC, 0xF9, 0x33, 0xFA, 0xB7, 0x7A, 0x9E, 0xCD, 0x1E, 0x3A,
0xFF, 0xE0, 0x81, 0xD6, 0x3D, 0x99, 0x5F, 0x66, 0xCE, 0x5F, 0xDF, 0xCA, 0x7C, 0x9D, 0xF6, 0x8A,
0xA4, 0x17, 0xF4, 0xA7, 0xE5, 0xF4, 0x21, 0x36, 0x89, 0x83, 0x7B, 0xED, 0x19, 0x72, 0x3C, 0x05,
0x27, 0xCB, 0x09, 0xE7, 0xE5, 0x02, 0x1B, 0xD8, 0x2E, 0x2E, 0x73, 0x67, 0x86, 0xBD, 0xE8,
0xB0, 0x02, 0x1B, 0x65, 0xD2, 0xB2, 0x9C, 0x00, 0x0F, 0x1A, 0x50, 0x3B, 0x44, 0x33, 0xAF, 0x92,
0x6D, 0x99, 0x5C, 0x9E, 0xEF, 0x61, 0x85, 0x01, 0xE9, 0x44, 0xCB, 0x00, 0xCD, 0xE9, 0x00, 0xFA,
0xEF, 0xEA, 0x90, 0x99, 0xE3, 0x71, 0xA7, 0x1A, 0x68, 0xC7, 0xFD, 0xCE, 0xFC, 0xB2, 0x02, 0xCA,
0xE3, 0x13, 0xFA, 0xB7, 0x3A, 0x68, 0x8E, 0x2C, 0xCB, 0xF1, 0x26, 0x03, 0xED, 0x54, 0xC9, 0xC2,
0x0F, 0x2C, 0x1C, 0xB4, 0x02, 0x64, 0x39, 0x51, 0x38, 0xD0, 0xFA, 0xAA, 0x31, 0x33, 0x14, 0x4C,
0x40, 0x16, 0xE2, 0x83, 0xB0, 0xAD, 0xAE, 0x52, 0x12, 0x31, 0x24, 0x70, 0x26, 0x53, 0x02, 0x90,
0xAE, 0x8C, 0xC9, 0x1B, 0x4D, 0x84, 0x50, 0x15, 0x9E, 0xA5, 0x76, 0x53, 0x5B, 0xD0, 0xB9, 0xCE,
```

Figure 5.1.3: Code B



```
#if defined(CAMERA_MODEL_WROVER_KIT)
#define PWDN_GPIO_NUM    -1
#define RESET_GPIO_NUM   -1
#define XCLK_GPIO_NUM    21
#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       19
#define Y4_GPIO_NUM       18
#define Y3_GPIO_NUM        5
#define Y2_GPIO_NUM        4
#define VSYNC_GPIO_NUM    25
#define HREF_GPIO_NUM     23
#define PCLK_GPIO_NUM     22

#elif defined(CAMERA_MODEL_ESP_EYE)
#define PWDN_GPIO_NUM    -1
#define RESET_GPIO_NUM   -1
#define XCLK_GPIO_NUM     4
#define SIOD_GPIO_NUM    18
#define SIOC_GPIO_NUM    23

#define Y9_GPIO_NUM       36
#define Y8_GPIO_NUM       37
```

Figure 5.1.4: Code C

The above figures shows the code which is uploaded onto ESP32-CAM by Arduino IDE from the computer. The Main code o the project is shown in the figure 5.1.1. Figueres 5.1.2, 5.1.3 and 5.1.4 shows the extended codes of the main code.

Now that we uploaded the code onto ESP32-CAM, we have to setup the Blynk app for it to receive the pictures and also to take pictures through it remotely.

Steps for the Blynk App setup:

- Open the project in the Blynk App Click on the “+” icon on the top.
- Select the Image Gallery Widget from the Widget Box
(Setting: Pin- V1) (Function: show the image)
- Select the Styled Button from the Widget Box
(Setting: Pin- GP14, Mode- PUSH) (Function: capture photo)
- Select the Notification from the Widget Box
(Function: get the notification)

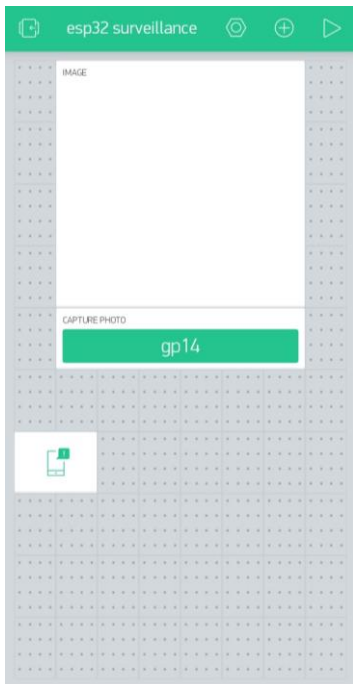


Figure 5.1.5: ESP32 Surveillance Pin

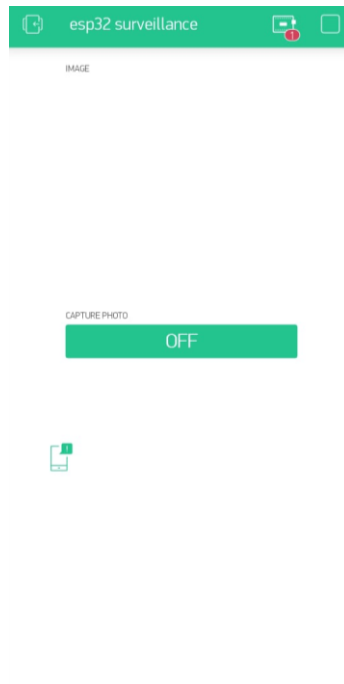


Figure 5.1.6: ESP32 Surveillance Off

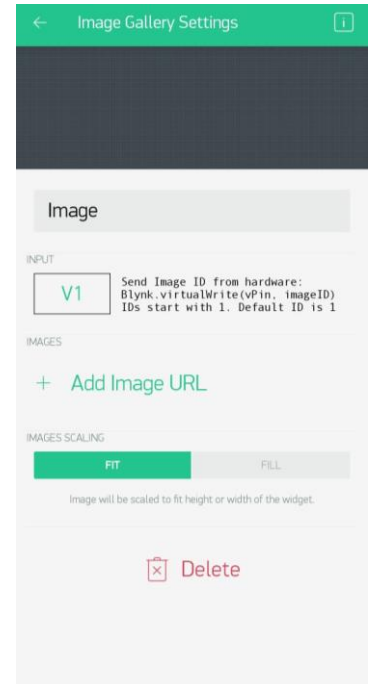


Figure 5.1.7: Image Gallery Settings

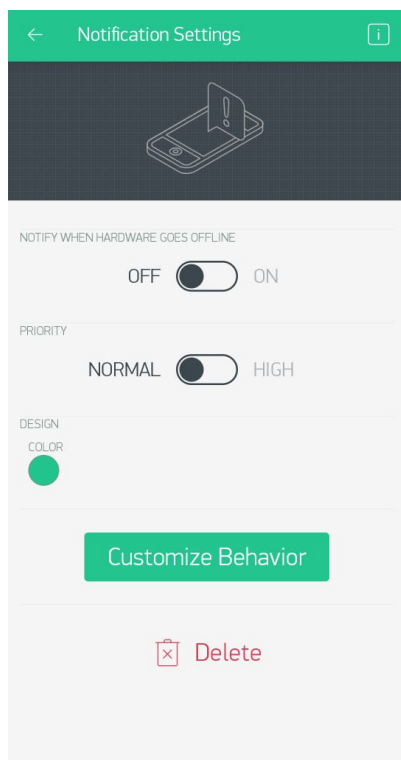


Figure 5.1.8: Notification Settings

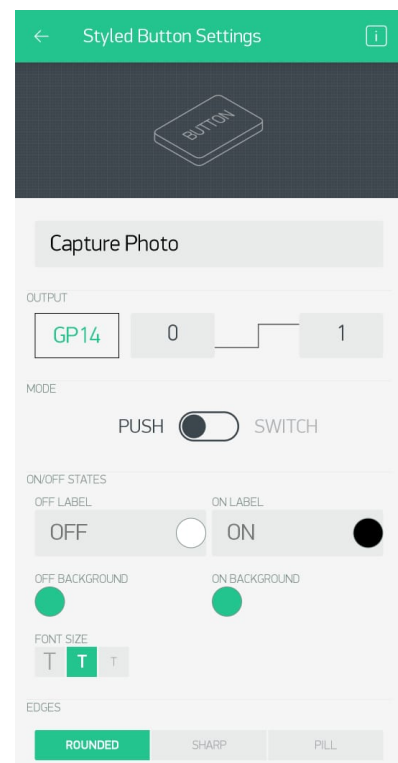


Figure 5.1.9: Styled Button Settings

Figure 5.1.5 shows that the signal is received from gp14 pin, since in the circuit, we connected ESP32-CAM at that place. We have the facility to on and off the surveillance whenever we wish. When the surveillance is off, you can see figure 5.1.6 in the Blynk app. When PIR motion sensor detects no motion, we can see figure 5.1.7 in the Blynk app. If it senses any motion, instead of blank space we will can see the image of the intruder. We can also customize the notification settings as shown in the figure 5.1.8. Since we are able to take pictures through Blynk app remotely, we can customize the settings regarding the same as shown in the figure 5.1.9.

After the Blynk app setup we have tested the circuit. We Supplied 5V DC to this security camera circuit and connect your smartphone with the same Wi-Fi network. Now if the PIR sensor detects any motion, you should get a notification on the mobile phone. After that click on the 'Take Picture' button to get the picture. The camera can also take pictures in the dark as the inbuilt LED on the ESP32-CAM will provide sufficient light.

Project Main Code

```
#include "esp_camera.h"
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>
//
// WARNING!!! PSRAM IC required for UXGA resolution and high JPEG quality y
//      Ensure ESP32 Wrover Module or other board with PSRAM is selected
//      Partial images will be transmitted if image exceeds buffer size
//

// Select camera model
#define CAMERA_MODEL_AI_THINKER // Has PSRAM

#include "camera_pins.h"

#define PIR 13
#define PHOTO 14
#define LED 4

const char* ssid = "dheeraj";
const char* password = "8500783513";
char auth[] = "AcVVhO7c0QlipBD3S-JJ0RffJ_ShFj8b"; //sent by Blynk

String local_IP;
```

```

void startCameraServer();

void takePhoto()
{
    digitalWrite(LED, HIGH);
    delay(200);
    uint32_t randomNum = random(50000);
    Serial.println("http://" + local_IP + "/capture?_cb=" + (String)randomNum);
    Blynk.setProperty(V1, "urls", "http://" + local_IP + "/capture?_cb=" + (String)randomNum);
    digitalWrite(LED, LOW);
    delay(1000);
}

void setup() {
    Serial.begin(115200);
    pinMode(LED, OUTPUT);
    Serial.setDebugOutput(true);
    Serial.println();

    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;

    // if PSRAM IC present, init with UXGA resolution and higher JPEG quality
    //           for larger pre-allocated frame buffer.
    if(psramFound()){
        config.frame_size = FRAMESIZE_UXGA;
        config.jpeg_quality = 10;
        config.fb_count = 2;
    } else {

```

```

        config.frame_size = FRAMESIZE_SVGA;
    config.jpeg_quality = 12;
    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);
    return;
}

sensor_t * s = esp_camera_sensor_get();
// initial sensors are flipped vertically and colors are a bit saturated
if (s->id.PID == OV3660_PID) {
    s->set_vflip(s, 1); // flip it back
    s->set_brightness(s, 1); // up the brightness just a bit
    s->set_saturation(s, -2); // lower the saturation
}
// drop down frame size for higher initial frame rate
s->set_framesize(s, FRAMESIZE_QVGA);

WiFi.begin(ssid, password);

while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
}
Serial.println("");
Serial.println("WiFi connected");

startCameraServer();

Serial.print("Camera Ready! Use 'http://");
Serial.print(WiFi.localIP());
local_IP = WiFi.localIP().toString();
Serial.println("' to connect");
Blynk.begin(auth, ssid, password);
}

void loop() {
    // put your main code here, to run repeatedly:
    Blynk.run();
    if(digitalRead(PIR) == LOW){
        Serial.println("Send Notification");
        Blynk.notify("Intruder Detected...");
        Serial.println("Capture Photo");
        takePhoto();
        delay(3000);
    }
}

```

```
        }  
    if(digitalRead(PHOTO) == HIGH){  
        Serial.println("Capture Photo");  
        takePhoto();  
    }  
}
```

Results

As compared to the previous studies, this system is more efficient, more affordable, and easier to install, operate and maintain. Various hardware and software tools are available on the market to make a home surveillance system. This system consumes very less power than others. By using the best and cheaper versions, this home surveillance system has many advantages as compared to the others. But for this surveillance system, we need constant internet to get the notifications to the smartphone. Some systems in the market may not use internet to notify, but those are very expensive to be afforded by common people. In conclusion, we can say that this home surveillance system has more advantageous as compared to the other systems.

REFERENCES

- [1] Sushma.N.Nichal, Prof.J.K.Singh, “Raspberry pi Based Smart Supervisor using Internet of Things (IoT)”,International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE) Volume 4, Issue 7, July 2015, ISSN: 2278 – 909X
- [2] R.Shete and S. Agrawal. Iot based urban climate monitoring usingraspberry pi. In2016 International Conference on Communication andSignal Processing (ICCSP), pages 2008–2012, April 2016.
- [3] S.Lee, N.Lee, J.Ahn, J.Kim, B.Moon, S. h. Jung, and D. Han.Construction of an indoor positioning system for home iot applications.In2017 IEEE International Conference on Communications (ICC),pages 1–7, May 2017.
- [4] A.Sahadevan, D.Mathew, J.Mookathana, and B. A. Jose. An offlineonline strategy for iot using mqtt. In2017 IEEE 4th InternationalConference on Cyber Security and Cloud Computing (CSCloud), pages369–373, June 2017.
- [5] Ms. RenukaChumurkar, Prof. Vijay Bagdi, “Smart Surveillance Security &Monitoring System Using Raspberry PI and PIR Sensor”, International Journal of Scientific Engineering and Applied Science (IJSEAS) – Volume-2, Issue-1, January 2016 ISSN: 2395-3470
- [6] Sowmiya .U, ShafiqMansoor.J., “Raspberry Pi based home door security through 3g dongle”, International Journal of Engineering Research and General Science Volume 3, Issue 2, March-April, 2015,ISSN 2091-2730
- [7] ShivprasadTavagad, ShivaniBhosale, Ajit Prakash Singh, Deepak Kumar, “ Survey Paper on Smart Surveillance System”, International Research Journal of Engineering and Technology (IRJET), Volume: 03 Issue: 02 | Feb-2016 e-ISSN: 2395 -0056, pISSN: 2395-0072