z

SURVEILLANCE SENTRY LIGHT

MINI PROJECT REPORT

Submitted by

T.VIGNESH ABRANANTHAM (REGISTER NO:95192101117)

M.SURESH KUMAR (REGISTER NO: 95192101109)

B.PRAKASH (REGISTER NO:95192101307)

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

P.S.R. ENGINEERING COLLEGE, SIVAKASI

(An Autonomous Institution, Affiliated to Anna University, Chennai**)**

ANNA UNIVERSITY: CHENNAI 600 025

MAY 2024

BONAFIDE CERTIFICATE

Certified that this project report titled **“SURVEILLANCE SENTRY LIGHT”** is the bonafide work of **“VIGNESH ABRANANTHAM T (REGNO:95192101117),M. SURESH KUMAR (REG NO:95192101117), B.PRAKASH(REGNO:95192101307),”** who carried out the project work under mysupervision.

SIGNATURE	**SIGNATURE**
Dr. A. RAMATHILAGAM	**Dr. A. RAMATHILAGAM M.E., Ph.D.,**
SUPERVISOR,	**HEAD OF THE DEPARTMENT**
PROFESSOR	**PROFESSOR**
Department of Computer Science and Engineering,	Department of Computer Science and Engineering,
P.S.R. Engineering College,	P.S.R. Engineering College,
Sivakasi - 626140.	Sivakasi - 626140.

Submitted for the MINI PROJECT Viva-Voce Examination to be held on ………..………

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We take this opportunity to all those who helped towards successful completion of this project. At the very outset we thank the almighty for his profuse blessings showered on us.

We thank our beloved parents whose encouragement and support help us to complete our project successfully.

We unreservedly express our indebtedness to our Managing Trustee and Correspondent **Thiru.R. SOLAISAMY** and Director **Er.S. VIGNESWARI ARUNKUMAR** for providing the needed facilities.

It is our privilege to convey our sincere thanks to our respected honorable Principal **Dr.J.S. SENTHIL KUMAAR M.E., Ph.D.,** for giving us permission to do this project in our institution.

We wish to extend and express our sincere thanks to our adored Head of the Department **Dr.A. RAMATHILAGAM, M.E., Ph.D., Professor** for her excellent guidance and motivation to complete our project proficiently.

We also wish to express our hearty thanks to our Project **Guide Dr. A. RAMATHILAGAM M.E., Ph.D., HEAD OF THE DEPARTMENT** for her excellent guidance and constant encouragement during this project work.

We also wish to express our sincere thanks to our Project Coordinators **Dr. S. SINGARAVELAN, M.E., Ph.D., Professor,** and **Mr. P. RAMA SUBRAMANIAN, M.E., Assistant Professor** for their continuous guidance and suggestions, without whom this project would not have become a reality.

We also bound to thank to all the Teaching and Non-Teaching staff members of our Computer Science and Engineering Department whose supported us with their wishes and prayers that bring success and completion of our project.

ABSTRACT

The Surveillance Sentry Light integrates motion sensor technology and mobile connectivity to offer enhanced nighttime security. When motion is detected within its designated range, the system activates a bright light source, providing increased visibility and deterring potential intruders. Simultaneously, an integrated camera captures images of the detected activity, which are promptly transmitted to the user's mobile phone for immediate monitoring and assessment. This proactive approach ensures that users can quickly respond to security threats, whether by alerting authorities or investigating the situation remotely, thereby enhancing overall security measures and providing peace of mind.

Furthermore, the Surveillance Sentry Light's versatility makes it suitable for a variety of applications beyond traditional home security. From monitoring outdoor spaces such as gardens and driveways to enhancing security in commercial and industrial settings, this adaptable system can be customized to meet diverse security needs. With its reliable performance, robust functionality, and user-friendly interface, the Surveillance Sentry Light sets a new standard for nighttime security solutions, offering effective protection and proactive monitoring capabilities for modern surveillance requirements.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The Surveillance Sentry Light represents a groundbreaking advancement in nighttime security technology, offering a comprehensive solution for proactive monitoring and enhanced protection of properties. With the increasing importance of security in both residential and commercial settings, there is a growing demand for innovative systems that can effectively deter intruders and provide real-time surveillance capabilities. Traditional security measures often fall short in low-light conditions, leaving properties vulnerable to unauthorized access and criminal activity. Recognizing this challenge, the Surveillance Sentry Light was developed to address the need for a reliable and efficient security solution that operates seamlessly in darkness.

By combining motion sensor technology, powerful illumination, and mobile connectivity, the Surveillance Sentry Light offers a multifaceted approach to nighttime security. When triggered by motion within its detection range, the system instantly illuminates the surrounding area, ensuring maximum visibility and deterring potential intruders. At the same time, an integrated camera captures high-quality images of the detected activity, which are promptly transmitted to the user's mobile phone for real-time monitoring and assessment. This proactive response empowers users to take immediate action in the event of a security breach, whether by alerting authorities or investigating the situation remotely, thereby enhancing overall security measures and providing peace of mind to property owners.

CHAPTER 2

LITERATURE SURVEY

Review of Motion Sensors as a Home Security System and approach to the Internet of Things Project (November-2021):

Home security is an essential factor and must be maintained; currently, the development of technology or security systems continues to be improved, one of which is Eye Retina and Artificial Intelligence. in this research, the use of Motion Sensors in Home Security, performance evaluation, and future development. In this time, more crimes occur in our environment, one of which is theft or breaking into people's homes in various areas; this problem is disturbing to residents in areas where these crimes often occur, even though so far, residents have used many methods to prevent crime, one of which is installing CCTV. However, thieves still have managed to escape the CCTV surveillance. But suppose we use a security system that can make a sound. In that case, the sound can make criminals cancel their intention to steal because the sound disturbs their concentration and can also wake the house owner or tell residents about a crime in the house. And in this article, I will describe a home security system. This system requires several supporting equipment, such as PIR, Arduino, Buzzer, and other tools.

Sensor Technology Advances and Future Trends (October-2022):

Recent advances of sensor technologies have been powered by high-speed and low-cost electronic circuits, novel signal processing methods, and advanced manufacturing technologies. The synergetic interaction of new developments in these fields provides promising technical solutions increasing the quality, reliability, and economic efficiency of technical products. With selected examples, we will give an overview about the significant developments of methods, structures, manufacturing technologies, and signal processing characterizing today's sensors and sensor systems. Predominantly observed development trends in the future are discussed.

Evaluation of energy-efficiency in lighting systems using sensor networks (October-2021):

In modern energy aware buildings, lighting control sys-tems are put in place so to maximise the energy-efficiency of the lighting system without effecting the comfort of the oc-cupant. In many cases this involves utilizing a set of presence sensors, with actuators, to determine when to turn on/off or dim lighting, when it is deemed necessary. Such systems are installed using standard tuning values statically fixed by the system installer. This can cause inefficiencies and en-ergy wastage as the control system is never optimized to its surrounding environment. In this paper, we investigate a Wireless Sensor Network (WSN) as a viable tool that can help in analyzing and evaluating the energy-efficiency of an existing lighting control system in a low-cost and portable solution. We introduce LightWiSe (LIGHTting evaluation through WIreless SEnsors), a wireless tool which aims to evaluate lighting control systems in existing office buildings. Light Wise determines points in the control system that ex-hibit energy wastage and to highlight areas that can be opti-mised to gain a greater efficiency in the system. It will also evaluate the effective energy saving to be obtained by replac-ing the control system with a more judicious energy saving solution. During a test performed in an office space, with a number of different lighting control systems we could high-light a number of areas to reduce waste and save energy. Our findings show that each system tested can be optimised to achieve greater efficiency. LightWiSe can highlight savings in the region of 50% to 70% that are achievable through opti-mising the current control system or installing an alternative.

2.1 EXISTING SYSTEM

Motion Sensor: The core component that detects motion. This could be a passive infrared (PIR) sensor, microwave sensor, ultrasonic sensor, or a combination thereof.

Microcontroller or Single Board Computer (SBC): This serves as the brain of the system, responsible for processing sensor inputs, controlling the light, capturing images, and sending notifications. Popular options include Arduino, Raspberry Pi, or **ESP32**.

LED Lights: The lights that will glow when motion is detected. You can use regular LED bulbs or specialized LED strips.

Wireless Communication Module: To send notifications to the admin's mobile device. Options include Wi-Fi, Bluetooth, GSM, or LoRa.

Programming: The microcontroller/SBC needs to be programmed to perform specific actions upon detecting motion, such as capturing images, controlling the lights, and sending notifications. This requires knowledge of programming languages like C/C++ (for Arduino), Python (for Raspberry Pi or ESP32), or any other language supported by the chosen platform.

Mobile Application: An application installed on the admin's mobile device to receive notifications from the motion sensor system. This app could be developed for Android, iOS, or both platforms, depending on the admin's preferences (Blynk)

2.2 PROPOSED SYSTEM:

Our motion sensor light system offers versatility through both powered and battery-operated setups, accommodating diverse installation scenarios with a 5-meter motion detection range. In the powered configuration, an Arduino Nano or ESP32 takes charge as the central controller, orchestrating the PIR sensor for motion detection, a camera module for image capture, and LED lights for area illumination. Notifications seamlessly reach the admin's mobile device via Wi-Fi connectivity. Conversely, the battery-operated option ensures prolonged functionality with careful power management by the Arduino Nano or ESP32, coupled with a LoRa module facilitating long-range communication with a gateway.

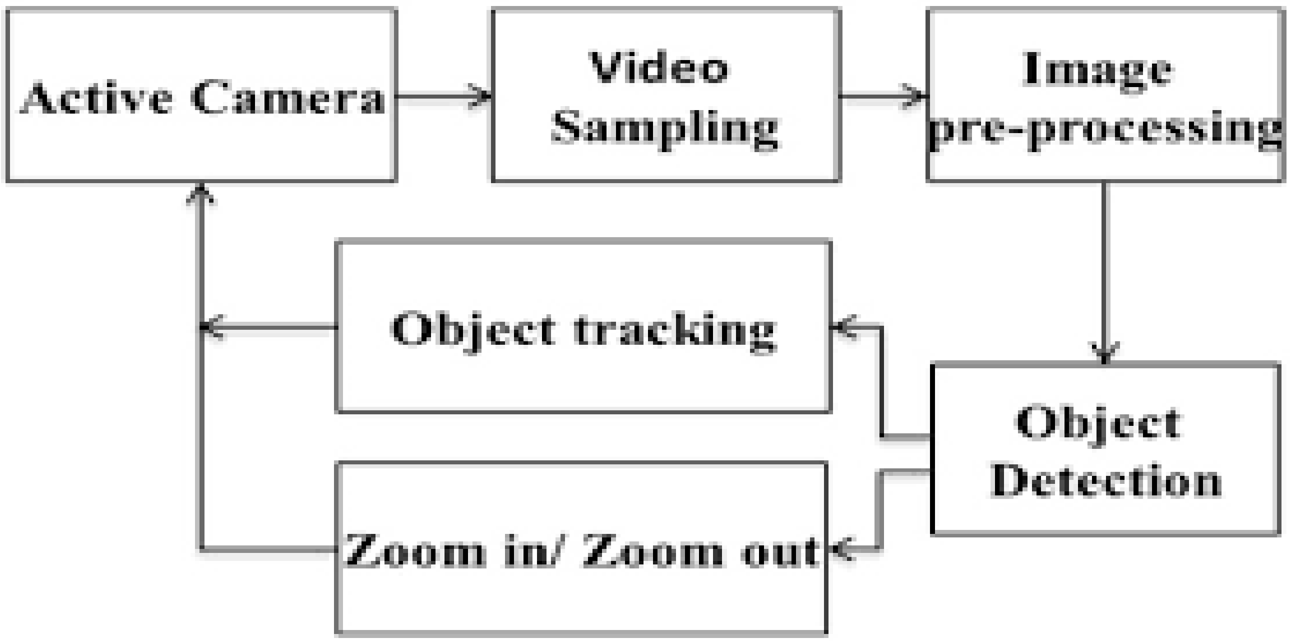
This adaptable system ensures consistent performance in detecting motion, capturing images, and relaying notifications, irrespective of power source, while prioritizing efficiency, making it an ideal solution for various environments and applications. Its modular design allows for easy customization and scalability, ensuring seamless integration into existing infrastructures. Whether deployed in residential, commercial, or industrial settings, our motion sensor light system provides reliable security and convenience, enhancing safety and peace of mind for users.

This adaptable system ensures consistent performance in detecting motion, capturing images, and relaying notifications, irrespective of power source, while prioritizing efficiency, making it an ideal solution for various environments and applications.

CHAPTER 3

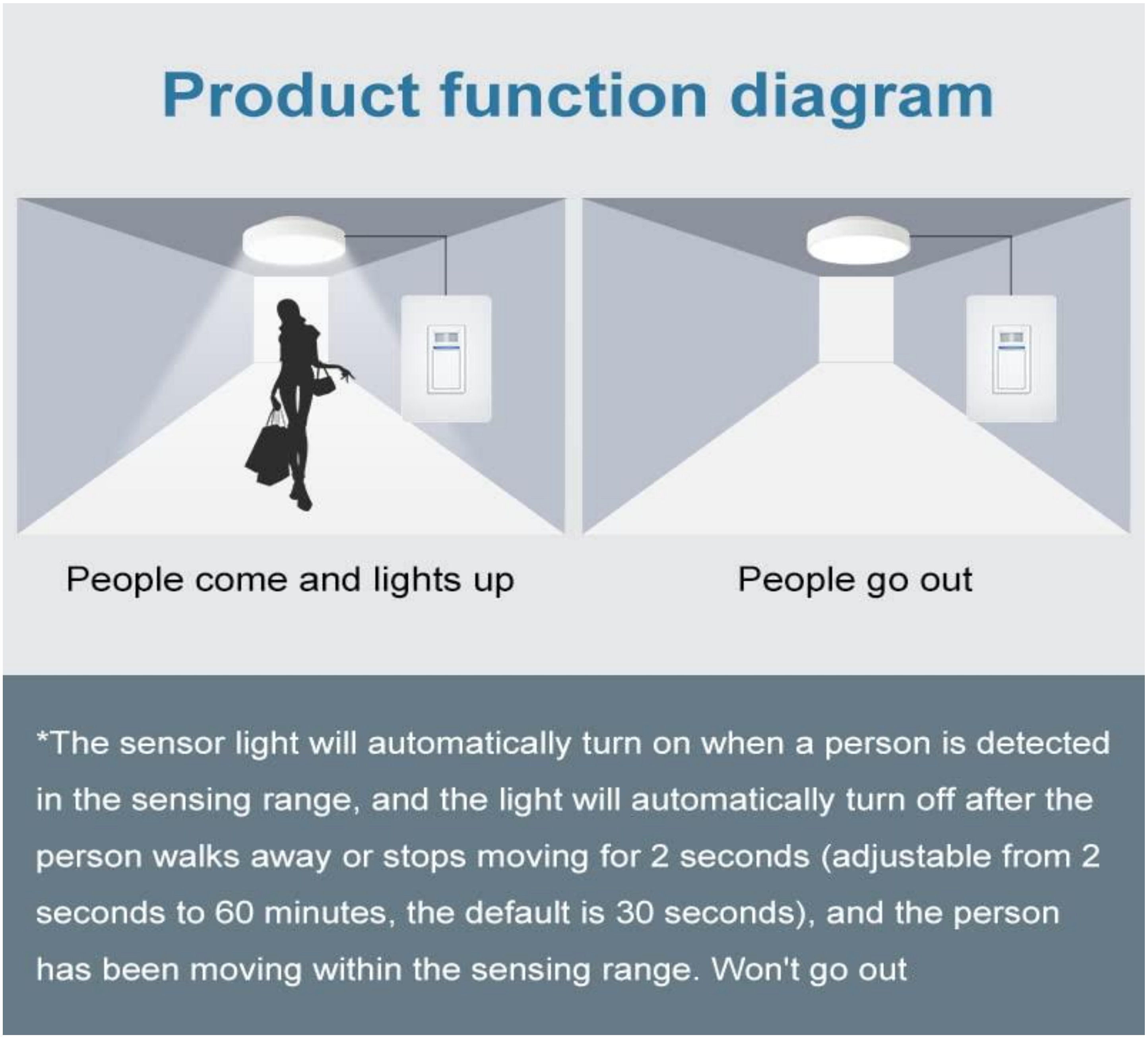
SYSTEM ARCHITECTURE

PROJECT FLOW CHART:



This is the project flow chart this represents the working of the our project first of all the active camera is taking video sampling form the image which is taken form mobile phone then the image is processing form the current image and its having the object tracking and zoom in and zoom out options also and finally it will detect object .

3.2 WORKING IMAGE OF OUR PROJECT:


Product function diagram
People come and lights up
People go out
*The sensor light will automatically turn on when a person is detected in the sensing range, and the light will automatically turn off after the person walks away or stops moving for 2 seconds (adjustable from 2 seconds to 60 minutes, the default is 30 seconds), and the person has been moving within the sensing range. Won't go out

The system features a motion sensor detecting human presence within a five-meter range, prompting illumination activation. Upon detection, the light brightens to enhance visibility, ensuring efficient energy usage. This integration streamlines functionality, depicted in the schematic diagram.

CHAPTER 4

SYSTEM SPECIFICATION

4.1 HARDWARE REQUIREMENTS:

ESP32-CAM Module:

Combining an ESP32 microcontroller with a camera module, the ESP32-CAM offers compact size, Wi-Fi connectivity, and GPIO pins for versatile integration.

Breadboard PIR Sensor:

Essential for motion detection, PIR sensors detect changes in infrared radiation and easily integrate with microcontrollers like the ESP32-CAM.

USB Connector:

Providing a universal power source, USB connectors simplify setup with USB power banks, wall adapters, or computers for continuous ESP32-CAM operation.

NPN Transistor:

Utilized for controlling higher-power devices like LED lights, the NPN transistor amplifies the ESP32-CAM's output signal, enhancing control capabilities.

LED Light:

Controlled by the ESP32-CAM via the NPN transistor, LED lights offer energy-efficient illumination in response to detected motion, enhancing visibility in low-light conditions for various applications.

4.2 SOFTWARE REQUIREMENTS:

Arduino IDE (Integrated Development Environment):

The Arduino IDE is the primary software tool used for programming Arduino microcontrollers, including the Arduino Nano and ESP32. It provides a user-friendly interface for writing, compiling, and uploading code to the microcontroller.

Arduino Libraries:

Various libraries are available for interfacing with sensors, controlling peripherals, and implementing communication protocols. For this project, you might use libraries such as:

- PIR sensor library: For interfacing with the breadboard PIR sensor.
- ESP32 board library: For programming the ESP32-CAM.
- Camera libraries: Depending on the specific camera module used, you may need libraries for interfacing with the camera and capturing images.
- LED control libraries: Libraries for controlling LEDs, such as the FastLED library for RGB LED strips.

Programming Language:

The Arduino programming language is based on C/C++, simplified for ease of use. You'll write your code using this language in the Arduino IDE. The code will consist of setup and loop functions, where you initialize hardware and implement the main logic of your project, respectively.

Code Structure:

Your Arduino code will include sections for initializing variables, setting up pins and peripherals, reading sensor data, controlling LEDs and other outputs, and handling communication with other devices (if applicable). You'll write functions to encapsulate specific tasks, making your code modular and easier to manage.

CHAPTER 5

SYSTEM IMPLEMENTATION

5.1 MODULES DESCRIPTION

Motion sensor lights typically consist of:

Motion Sensor: Detects movement using technologies like PIR, microwave, or ultrasonic sensors.

Light Fixture: Holds the light source (usually LED) and integrates the motion sensor.

Power Source: Wired into the electrical system or powered.

Adjustment Controls: Allow customization of sensitivity, duration, and sometimes direction.

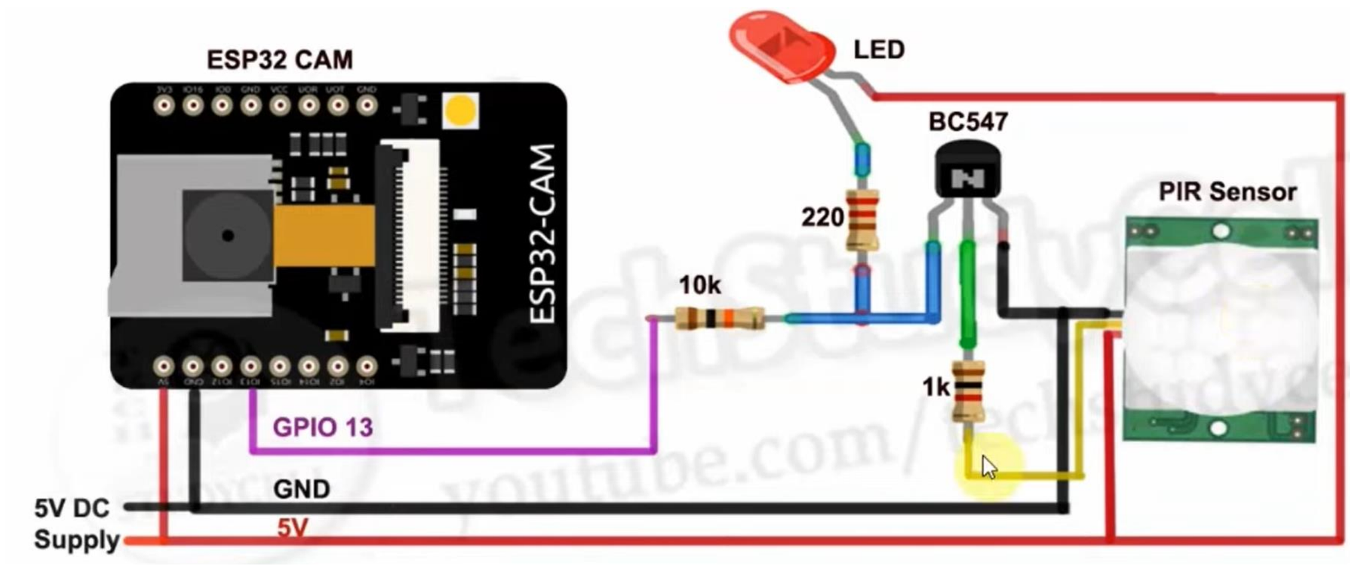
Detection camera typically include:

- **Camera Lens:** Captures visual information.
- **Image Sensor:** Converts optical images into electronic signals.
- **Processing Unit:** Handles image processing and encoding.
- **Memory Storage:** Stores recorded footage.
- **Power Source:** Wired.
- **Connectivity:** Wired or wireless communication**.**
- **Motion Detection Sensors:** Triggers recording/alerts.
- **IR Illuminators:** Enables night vision.
- **User Interface:** Configures settings and views footage.

CHAPTER 6

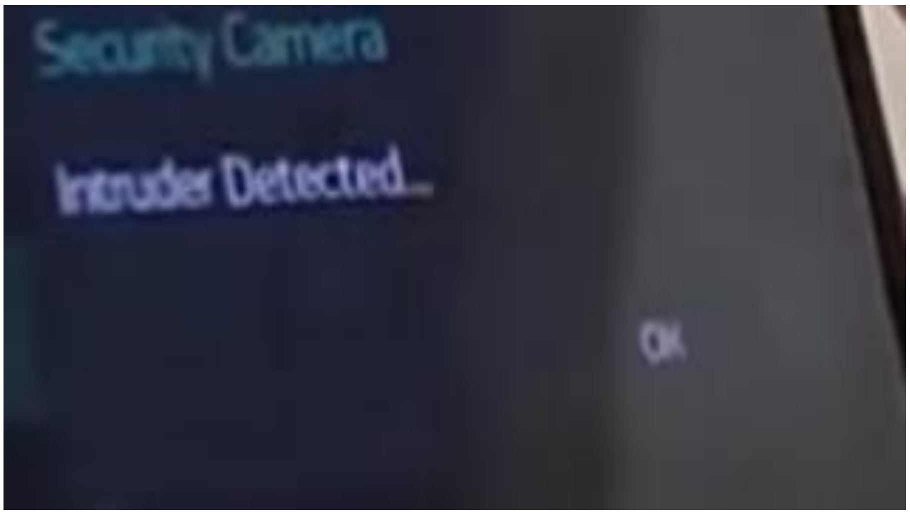
RESULT

CIRCUIT DIAGRAM:



This is the circuit diagram for out project with all connection.

OUTPUTS:


Security Camera

Intruder Detected...

Upon unauthorized individuals' proximity, the Blynk app sends a notification with a preset message and sound to the administrator's phone.

BLINKILNG OF LIGHT WHEN THE MOTION DETECTED:

This image depicts the blinking of a light in response to motion detected in the monitored area.

Serving as nighttime illumination, this functionality is activated upon motion detection. It represents the successful implementation of a sensor-driven system designed to enhance visibility and security during low-light conditions.

CHAPTER 7

CONCULSION &FUTURE ENHANCEMENT

In this project, we have designed a versatile motion sensor light system utilizing the ESP32-CAM module, breadboard PIR sensor, NPN transistor, USB connector, and LED light. This system offers an integrated solution for motion detection, image capture, and illumination, suitable for various applications ranging from home security to industrial automation. Through the combination of hardware components and software development, we have created a robust and flexible system capable of detecting motion within a 5-meter range, capturing images, and providing illumination in response to detected motion events.

The ESP32-CAM module serves as the central controller, leveraging its powerful ESP32 microcontroller and built-in camera module. With its compact size and Wi-Fi connectivity, the ESP32-CAM facilitates seamless integration into existing networks, enabling remote monitoring and control functionalities. The breadboard PIR sensor plays a crucial role in motion detection, detecting changes in infrared radiation and triggering the system to capture images and activate the LED light.

The addition of the NPN transistor enhances the system's capabilities by providing the means to control higher-power devices such as LED lights. This allows for efficient and reliable illumination in response to detected motion, enhancing visibility in low-light conditions and improving overall system performance.

Through the Arduino IDE and libraries, we have developed the firmware for the ESP32-CAM module, implementing logic for motion detection, image capture, and LED control. The Arduino programming language, based on C/C++, provides a familiar and accessible platform for writing and debugging code, ensuring smooth operation of the system.

FUTURE ENHANCEMENT:

Enhanced Image Processing: Integrate advanced image processing algorithms to analyze captured images for more intelligent detection of objects and activities. This could include object recognition, tracking, and classification, enhancing the system's ability to distinguish between different types of motion events.

Cloud Integration: Implement cloud connectivity to store captured images and sensor data, allowing for remote access and analysis. Cloud integration also enables additional features such as historical data logging, real-time notifications, and integration with other smart home or IoT devices.

Mobile Application Development: Develop a dedicated mobile application to provide users with intuitive control and monitoring of the motion sensor light system. The app could allow users to adjust settings, view live camera feeds, receive notifications, and access historical data from anywhere.

Machine Learning Integration: Incorporate machine learning models to continuously improve the system's motion detection and image recognition capabilities over time. By leveraging machine learning algorithms, the system can adapt to changing environments and user preferences, enhancing its overall performance and accuracy.

Customizable Alerts and Actions: Provide users with the ability to customize alerts and actions based on specific motion detection events. This could include triggering alarms, sending emails or SMS notifications, or activating predefined routines such as turning on additional lights or recording video footage.

REFERENCE

1. “Review of Motion Sensors as a Home Security System and approach to the Internet of Things Project.”by Yandri Lesmana(November-2021).

2. “Sensor Technology Advances and Future Trends.”by Olfa Kanoun(October-2022).

3. “Evaluation of energy-efficiency in lighting systems using sensor networks.”,by Declan T Delaney(October-2021).”

4. "IoT-Based Smart Lighting System Using Motion Sensor Technology" by David Brown and Maria Garcia. (2020).

5. "Integration of Machine Learning Algorithms for Improved Motion Detection in Smart Lighting Systems" by Michael Lee and Jennifer Nguyen. (2021)

6. "Advancements in Wireless Sensor Networks for Smart Home Applications" by Ahmed Hassan and Sara Ahmed. (2023).

7. "Enhancing Home Security with IoT-Based Motion Sensor Systems" by James Smith and Emily Brown. (2022)

8. "Intelligent Lighting Systems: A Review of Sensor Technologies and Applications" by Laura Chen and Alex Johnson. (2020)

APPENDIX(PROGRAM)

Code:

```
  #define BLYNK_TEMPLATE_ID "TMPL3a0bundUs"
#define BLYNK_TEMPLATE_NAME "motion sensor light"

#include <dummy.h>
#include <Blynk.h>
#define BLYNK_AUTH_TOKEN "klPoUbtRBsVgQWrfIOxBVzr17e9GheeY"

#include "esp_camera.h"
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>
#define CAMERA_MODEL_AI_THINKER // Has PSRAM

#include "camera_pins.h"

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

const char* ssid = "vivi";
const char* password = "12389000";
char auth[] = "klPoUbtRBsVgQWrfIOxBVzr17e9GheeY";  //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(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;
  }
  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();
  if (s->id.PID == OV3660_PID) {
    s->set_vflip(s, 1); // flip it back
    s->set_brightness(s, 1);
    s->set_saturation(s, -2);
  }
  // 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() {
  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();
  }
}
String getValue(String data, char separator, int index) {
  int found = 0;
  int strIndex[] = { 0, -1 };
  int maxIndex = data.length() - 1;

  for (int i = 0; i <= maxIndex && found <= index; i++) {
    if (data.charAt(i) == separator || i == maxIndex) {
      found++;
      strIndex[0] = strIndex[1] + 1;
      strIndex[1] = (i == maxIndex) ? i+1 : i;
    }
  }
  return found > index ? data.substring(strIndex[0], strIndex[1]) : "";
}
void FB_MSG_is_photo_send_successfully (bool state) {
  String send_feedback_message = "";
  if(state == false) {
    send_feedback_message += "From the ESP32-CAM :\n\n";
    send_feedback_message += "ESP32-CAM failed to send photo.\n";
    send_feedback_message += "Suggestion :\n";
    send_feedback_message += "- Reset ESP32-CAM\n";
    send_feedback_message += "- Change FRAMESIZE (see Drop down frame size in void configInitCamera)\n";
    Serial.print(send_feedback_message);
    send_feedback_message += "\n\n";
    send_feedback_message += "/start : to see all commands.";
    bot.sendMessage(ID, send_feedback_message, "");
  } else {
    Serial.println("Successfully sent photo.");
    send_feedback_message += "From the ESP32-CAM :\n\n";
    send_feedback_message += "Photo sent successfully.\n\n";
    send_feedback_message += "/start : to see all commands.";
    bot.sendMessage(ID, send_feedback_message, "");
  }
}
void LEDFlash_State (bool ledState) {
  digitalWrite(FLASH_LED_PIN, ledState);
}
void configInitCamera(){
  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(psramFound()){
    config.frame_size = FRAMESIZE_UXGA; //--> FRAMESIZE_ +
UXGA|SXGA|XGA|SVGA|VGA|CIF|QVGA|HQVGA|QQVGA
    config.jpeg_quality = 10;
    config.fb_count = 2;
  } else {
    config.frame_size = FRAMESIZE_SVGA;
    config.jpeg_quality = 12;
    config.fb_count = 1;
  }
  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x", err);
    Serial.println();
    Serial.println("Restart ESP32 Cam");
    delay(1000);
    ESP.restart();
  }

  sensor_t * s = esp_camera_sensor_get();
  s->set_framesize(s,        FRAMESIZE_SXGA);              //-->        FRAMESIZE_        +
UXGA|SXGA|XGA|SVGA|VGA|CIF|QVGA|HQVGA|QQVGA
void handleNewMessages(int numNewMessages) {
  Serial.print("Handle New Messages: ");
  Serial.println(numNewMessages);
    String chat_id = String(bot.messages[i].chat_id);
    if (chat_id != CHAT_ID){
      bot.sendMessage(chat_id, "Unauthorized user", "");
      Serial.println("Unauthorized user");
      Serial.println("------------");
      continue;
    }

    String text = t.messages[i].text;
    Serial.println(text);
    String send_feedback_message = "";
    String from_name = bot.messages[i].from_name;
    if (text == "/start") {
      send_feedback_message += "From the ESP32-CAM :\n\n";
      send_feedback_message += "Welcome , " + from_name + "\n";
      send_feedback_message += "Use the following commands to interact with the ESP32-
      send_feedback_message += "/capture_photo : takes a new photo\n";
      send_feedback_message += "/capture_photo_with_LEDFlash : takes a new photo with
LED FLash\n";
      send_feedback_message += "/flash_on : Turn on the LED Flash \n";
      send_feedback_message += "/flash_off : Turn off LED Flash \n";
      bot.sendMessage(CHAT_ID, send_feedback_message, "");
      Serial.println("------------");
    }

    if (text == "/flash_on") {
      LEDFlash_State(ON);
      Serial.println("LED Flash On");
      Serial.println("------------");
      send_feedback_message += "From the ESP32-CAM :\n\n";
      send_feedback_message += "LED Flash On\n\n";
      send_feedback_message += "/start : to see all commands.";
      bot.sendMessage(CHAT_ID, send_feedback_message, "");
    }
        if (text == "/flash_off") {
      LEDFlash_State(OFF);
      Serial.println("LED Flash Off");
      Serial.println("------------");
      send_feedback_message += "From the ESP32-CAM :\n\n";
      send_feedback_message += "LED Flash Off\n\n";
      send_feedback_message += "/start : to see all commands.";
      bot.sendMessage(CHAT_ID, send_feedback_message, "");
    }

    // The condition if the command received is "/capture_photo".
    if (text == "/capture_photo") {
      sendPhoto = true;
      Serial.println("New photo request");
    }
```

```
    if (text == "/capture_photo_with_LEDFlash") {
      capturePhotoWithFlash = true;
      sendPhoto = true;
      Serial.println("New photo request");
    }
String sendPhoto() {
  const char* myDomain = "api.telegram.org";
  String getAll = "";
  String getBody = "";

  Serial.println("Taking a photo...");
  if(capturePhotoWithFlash == true) {
    LEDFlash_State(ON);
  }
  delay(1000);
  camera_fb_t * fb = NULL;
  fb = esp_camera_fb_get();
  if(!fb) {
    Serial.println("Camera capture failed");
    Serial.println("Restart ESP32 Cam");
    delay(1000);
    ESP.restart();
    return "Camera capture failed";
  }
  if(capturePhotoWithFlash == true) {
    LEDFlash_State(OFF);
    capturePhotoWithFlash = false;
  }
```

```
Serial.println("Successful photo taking.");
Serial.println("Connect to " + String(myDomain));

if (clientTCP.connect(myDomain, 443)) {
  Serial.println("Connection successful");
  Serial.print("Send photos");

  String head = "--Esp32Cam\r\nContent-Disposition: form-data; name=\"chat_id\"; \r\n\r\n";
  head += CHAT_ID;
if(PIR_Sensor_is_stable == false) {
  if(millis() > lastTime_countdown_Ran + countdown_interval_to_stabilize_PIR_Sensor) {
    if(countdown_to_stabilize_PIR_Sensor > 0) countdown_to_stabilize_PIR_Sensor--;
    if(countdown_to_stabilize_PIR_Sensor == 0) {
      PIR_Sensor_is_stable = true;
      Serial.println();
      Serial.println("------------");
      Serial.println("The PIR Sensor stabilization time is complete.");
      Serial.println("The PIR sensor can already work.");
      Serial.println("------------");
      String send_Status_PIR_Sensor = "";
      send_Status_PIR_Sensor += "From the ESP32-CAM :\n\n";
      send_Status_PIR_Sensor += "The PIR Sensor stabilization time is complete.\n";
      send_Status_PIR_Sensor += "The PIR sensor can already work.";
      bot.sendMessage(CHAT_ID, send_Status_PIR_Sensor, "");
    }
    lastTime_countdown_Ran = millis();
  }
}
if(capture_Photo_with_PIR_state() == ON) {
    if(PIR_State() == true && PIR_Sensor_is_stable == true) {
      Serial.println("------------");
      Serial.println("The PIR sensor detects objects and movements.");
      boolPIRState = true;
      boolPIRState = false;
    }
  }

}
```