Security Surveillance System using IOT

Dissertation
(MCA 302)

Submitted in partial fulfillment of requirement for the award of the degree of

Master of Computer Application

Submitted by

Aanandita (Enrollment No: 02314504419)

Under the supervision of

Dr. Rekha Gupta
Professor

Lal Bahadur Shastri Institute of Management, Delhi
Affiliated to
Guru Gobind Singh Indraprastha University, Delhi
(2022)

TITLE OF THE PROJECT

Security Surveillance System using IOT

LAL BAHADUR SHASTRI INSTITUTE OF MANAGEMENT
DWARKA, DELHI

CERTIFICATE

It is certified that the dissertation entitled "Security Surveillance system using IOT" submitted by AANANDITA, Enrolment No. 02314504419 for the award of the degree of **Master of Computer Applications** to the Lal Bahadur Shastri Institute of Management, Dwarka, Delhi affiliated to Guru Gobind Singh Indraprastha University, Delhi is a bonafide work carried out by her under my supervision and guidance. It is further certified that she is a bonafide student at this institute.

Date:
Place:

Dr. Rekha Gupta
(Project Guide)

DECLARATION

I, AANANDITA, hereby declare that the work done in this dissertation, has been carried out by me under the supervision of Dr. Rekha Gupta, Professor, Lal Bahadur Shastri Institute of Management, Dwarka, Delhi.
Further, I declare that no part of this dissertation has been submitted either in part or full for the award of any degree to this university or any other university.

Date: Aanandita
Place: 02314504419

ACKNOWLEDGEMENT

Apart from the efforts of myself, the success of any project depends largely on the encouragement and guidelines of many others. I would like to take this opportunity to express my gratitude to the people who have been instrumental in the successful completion of this project.

The completion of any inter-disciplinary project depends upon cooperation, coordination and combined efforts of several sources of knowledge.

I am eternally grateful to my project supervisor Dr. Rekha Gupta for her ever willingness to provide me with valuable advice and direction; under whom I have executed this project. Her constant guidance and disposition in sharing her vast knowledge made me understand this project and its manifestations in great depths. Her invaluable guidance has been monumental for this project.
She has proved to be a guiding light in my entire journey and her valuable insight has helped me improve and make this project a success.

Aanandita Diwan

02314504419

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Abstract

In today's tech era, IOT (Internet of things) has become very popular around the world. Almost all the devices, which are known as smart devices, can connect to the internet and access data from any corner of the world. There was a time when people used to waste their valuable time just to get a piece of information. Now the technology is more advanced compared to previous times. One of the blessings of technology is web application. It allows users to interact with the system from anywhere as long as they are connected to the internet.

This report presents the work done on the project "Security surveillance System" which is a hardware application. As we already discussed above, the internet has made the user's interaction through the system easier, so this system uses the web to connect to respective services,i.e. email, telegram, for sending of data which will surely help users to get information in form of live alerts about their home/office security. Moreover, in recent decades people live life full of stress and would like to get some relief from their hectic daily lives. So, they want to live their lives without constant fear for their safety . In this modern tech era they want a system that will enhance the portability, security as well as is easy to use and hassle free. So here, we are going to implement a system, which we already stated above, having all the features that will make it more user friendly and accessible to all at a low cost.

The aim of this project was to develop a System that would help the public live a secure, safe and tension free life with help of technology that is within their budget.

Security is an important issue and we can only be at peace once we know that we and our loved ones are safe. The purpose of this project is to provide surveillance through smartphones, so that all users of the application can secure their premises whether they are at their workplace or at their home.

1. Introduction along with Literature Survey and objectives

In 2020, the capital union territory of Delhi had the highest rate of robbery, with almost 9.7 reported cases per 100,000 people. There were over 24 thousand robbery cases in the country that year.
Delhi recorded over 15 per cent rise in crime in 2021 as compared to the previous year. Citing data, police said 3,06,389 cases were reported in 2021 as compared to 2,66,070 cases the previous year. Last year, 2,87,563 cases were registered under 'other IPC sections' (theft, robbery, burglary). Nearly 70 per cent of the crimes reported in 2021 were burglary, robbery and theft, police said.
This goes to show the importance of security and surveillance systems and how much they are needed.

Literature survey of past projects :

Monitoring projects are popular in the IOT domain. Some projects that use the ESP32 module are :
1) Smart Aquarium - This project is built with an Espressif ESP32S board which monitors the aquarium. Specifically, it allows users to schedule when the fish will be fed automatically, or manually with a touch of a button in an Android app. Users can even specify the amount of food to be fed.
2) Face recognition - The camera in ESp32 can capture the images and these images can be run through Python libraries that help recognise and categorize faces. This project can be further bifurcated into an attendance system (marking attendance of students by mapping their faces) or home unlocking system (which verifies the homeowner's face and only then unlocks the door).

2. *Body of Proposal*

To combat the problem of theft it's better to take precautions beforehand and install surveillance systems. A study of 422 burglars found that security cameras were the most effective deterrent to burglars. 60% of the burglars said that they would pick a different target if they knew that a building or site had security cameras, and if they didn't know ahead of time then 50% said that discovering surveillance cameras would cause them to abort the burglary. Just by installing a security camera system, you're reducing your odds of being burglarized by half.

2.0 Methodology & Modules

In our system, A motion sensor - PIR sensor - will continuously detect movement. Once a movement is triggered , the sensor will issue an interrupt to the microcontroller module, Asking it to send an alert. This alert will be sent over the internet to the owner's phone.

The PIR sensor can also trigger an alarm that can be placed inside the house to alert the residents or can be placed outside to spook the burglar.

The people within the same wifi network can then log into the local network to see a live stream of who has entered the premises through the camera module and microcontroller which will be installed beside the PIR sensor.

The communication between PIR and the microcontroller module is done through wiring. The camera and microcontroller will be configured to the same IP address as home wifi, this way the live video stream can be viewed using the local area network. For people outside the local network an email or telegram alert will be sent through the microcontroller through the wifi using SMTP protocol. The PIR can trigger the buzzer depending on the placement of the buzzer. If the distance is less they can be connected through wiring, otherwise a receiver and transmitter need to be connected with the buzzer and PIR respectively.

MODULE 1 : MOTION DETECTION

PIR sensors allow you to sense motion, almost always used to detect whether a human has moved in or out of the sensor's range. They are small, inexpensive, low-power, easy to use and don't wear out. For that reason they are commonly found in appliances and gadgets used in homes or businesses. They are often referred to as PIR, "Passive Infrared", "Pyroelectric", or "IR motion" sensors.
Whenever a person or intruder will walk in the PIR sensor's range the PIR will experience a rising value and send a trigger to the ESP32 board.

MODULE 2 : ALERT THROUGH INTERNET

ESP32 is a low-cost System on Chip (SoC) Microcontroller from Espressif Systems.ESP32 Development board is based on the ESP WROOM32 WIFI + BLE Module.It's a low-footprint, minimal system development board powered by the latest ESP-WROOM-32 module. It contains the entire basic support circuitry, including the USB-UART bridge, reset- and boot-mode buttons, LDO regulator and a micro-USB connector. Every important GPIO is available to the developer.
The WIFI enables theESP to send and receive serial communication over the internet through the local area network. When the ESP board receives the trigger through PIR it will send an alert through SMTP protocol over email to the registered email id. ESP itself needs an email id to send these alerts.
If you don't want to configure an email id for ESP to use, we can use a telegram app to receive these alerts or install an application called Blynk that supports features of ESP.

MODULE 3 : LIVE SECURITY VIDEO STREAM

The OV7670/OV7171 CAMERACHIPTM is a low voltage CMOS image sensor that provides the full functionality of a single-chip VGA camera and image processor in a small footprint package. The OV7670 provides full-frame, sub-sampled or windowed 8-bit images in a wide range of formats, controlled through the Serial Camera Control Bus (SCCB) interface. We can use localhost to view live stream video of the surveillance we want to monitor.

2.1 Technology Used

HARDWARE REQUIREMENTS :

1. ESP32 microcontroller module
2. PIR motion sensor (AM312) or (HC-SR501)
3. FTDI programming board
4. Buzzer
5. Jumper wires
6. Breadboard
7. 433Mhz RF Transmitter and Receiver Module
8. OV7670 Camera module
9. A screen that is used for viewing live stream surveillance footage such as: LCD module, monitor, laptop, desktop

SOFTWARE REQUIREMENTS :

1. Arduino IDE
2. App used to send alerts on such as - telegram, gmail, blinkit
3. An email account that ESP will use to send alerts from
4. An email account that will act as a receiver.
5. Supporting libraries - ESP-Mail-Client library , WiFi.h
6. Wifi credentials
7. Working internet connection

2.2 Project Planning

TASK NAME	10 APR 2022	15 APR 2022	20 APR 2022	25 APR 2022	15 MAY 2022	20 MAY 2022	25 MAY 2022
Planning and Research	🟨	🟨					
Finalization of project		🟨	🟨				
Synopsis			🟨	🟨			
Implementation				🟨	🟨		
Testing					🟨	🟨	
Documentation						🟨	🟨

Table 1 : Gantt Chart

2.3 Feasibility Study

1) TECHNICAL FEASIBILITY :

This involves financial considerations to accommodate technical enhancements. If the budget is a serious constraint, then the project is judged not feasible. In the proposed system only the cost of developing and installing the system is involved. If the system is placed outside of serious environmental calamities then no harm will come to the system and there is no need for any special maintenance or other special software entities.

2) ECONOMICAL FEASIBILITY :

With the help of this application, it will lead to a decrease in cost of employing security guards or installing high end security systems, which will be more than the cost of developing and maintaining this project. No manpower is needed for the maintenance after installation which will again lead to decrease in cost.

3) OPERATIONAL FEASIBILITY :

This Application is very easy to operate as it is made user-friendly. Main consideration is the user's easy access to all the functionality of the system.

4) SECURITY MECHANISM :

The alerts will only be sent to the appropriate mail/chat id that makes use of an already tested and secure mailing/messaging system i.e. either through gmail or through the telegram app.

The live stream of footage outside the house can only be seen with person having access to the local network that the ESP is using and thus encapsulates security to only the people who know the credentials and are within the same IP address.

2.4 Design of the system

1. Use Case Diagram

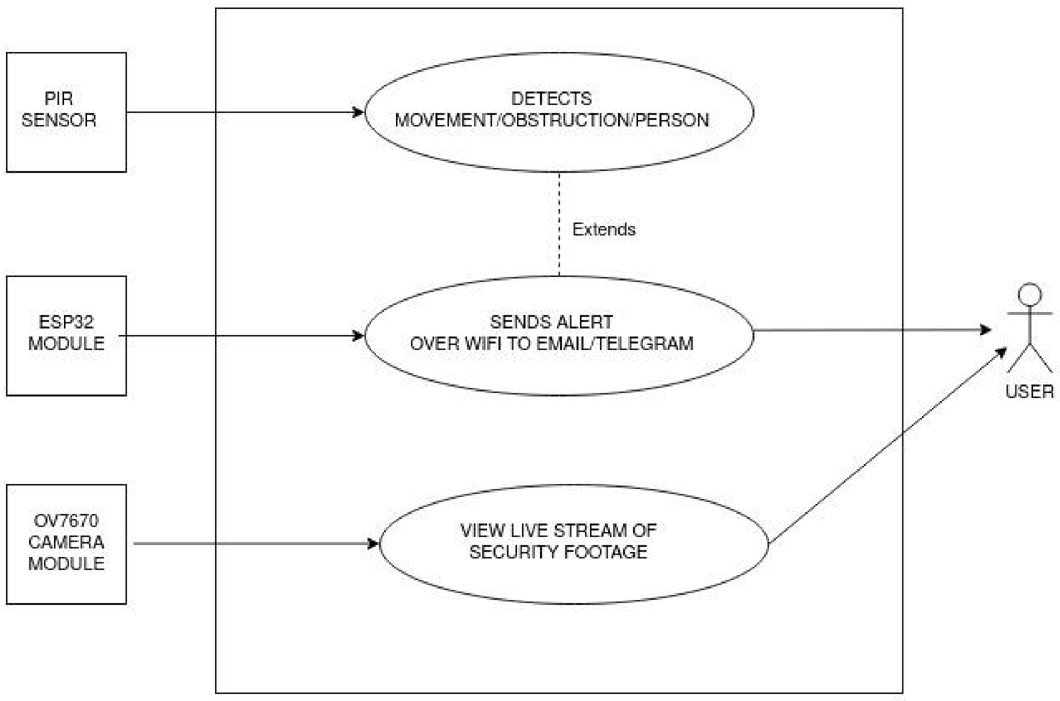

PIR SENSOR
ESP32 MODULE
OV7670 CAMERA MODULE
DETECTS MOVEMENT/OBSTRUCTION/PERSON
Extends
SENDS ALERT OVER WIFI TO EMAIL/TELEGRAM
VIEW LIVE STREAM OF SECURITY FOOTAGE
USER

FIGURE 1: USE CASE DIAGRAM

2. Use Case Description

The Pir sensor is a hardware module that detects motion. Once any object passes through the range of PIR's detection, the PIR will trigger an electronic signal to wake ESP32.

The ESP32 will use SMTP protocol to send an alert over the wifi though the application selected by the user, i.e. email or telegram.

The user can then log into the local network and see a live stream of the footage where the camera is installed to determine if the alert is urgent and a trespasser has breached onto the property.

3. Entity Relationship Diagram

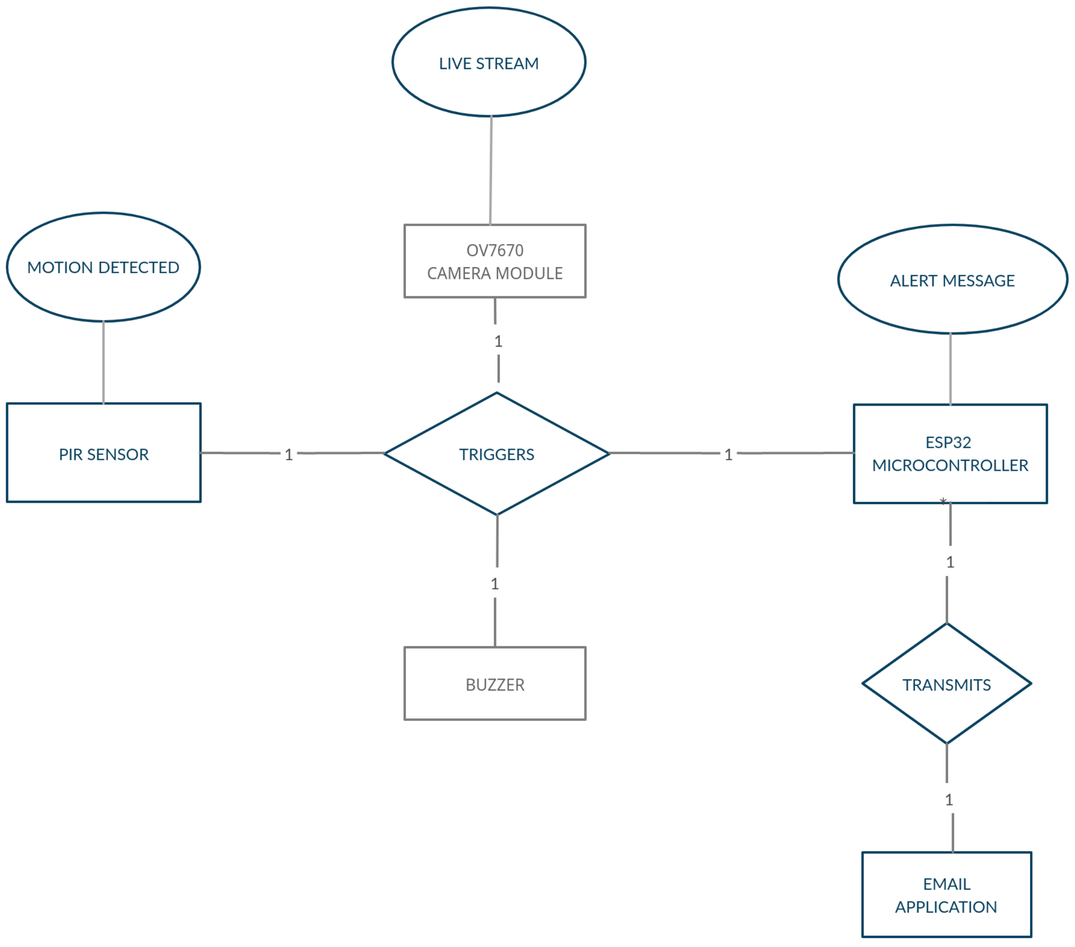


FIGURE 2 : ER-DIAGRAM

4. Data Flow Diagrams (DFDs)

4.1. Context Diagram or Level-0 DFD

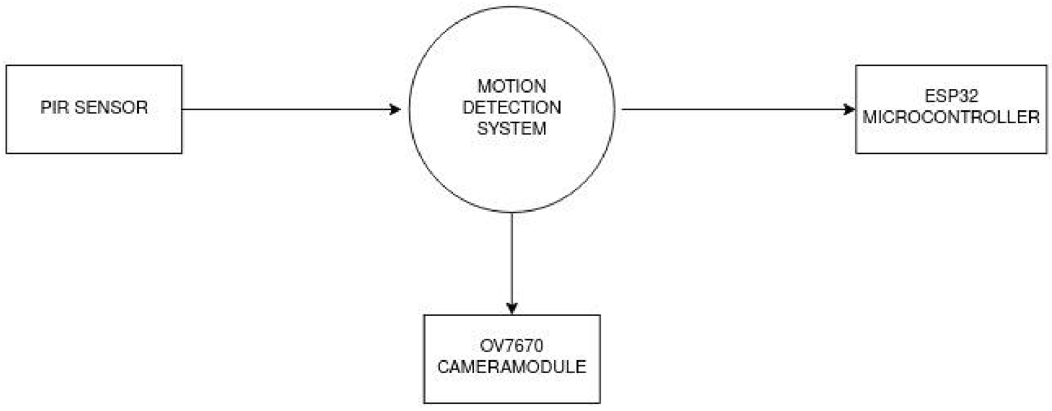


FIGURE 3 : DFD LEVEL 0

4.2. Level-1 DFD

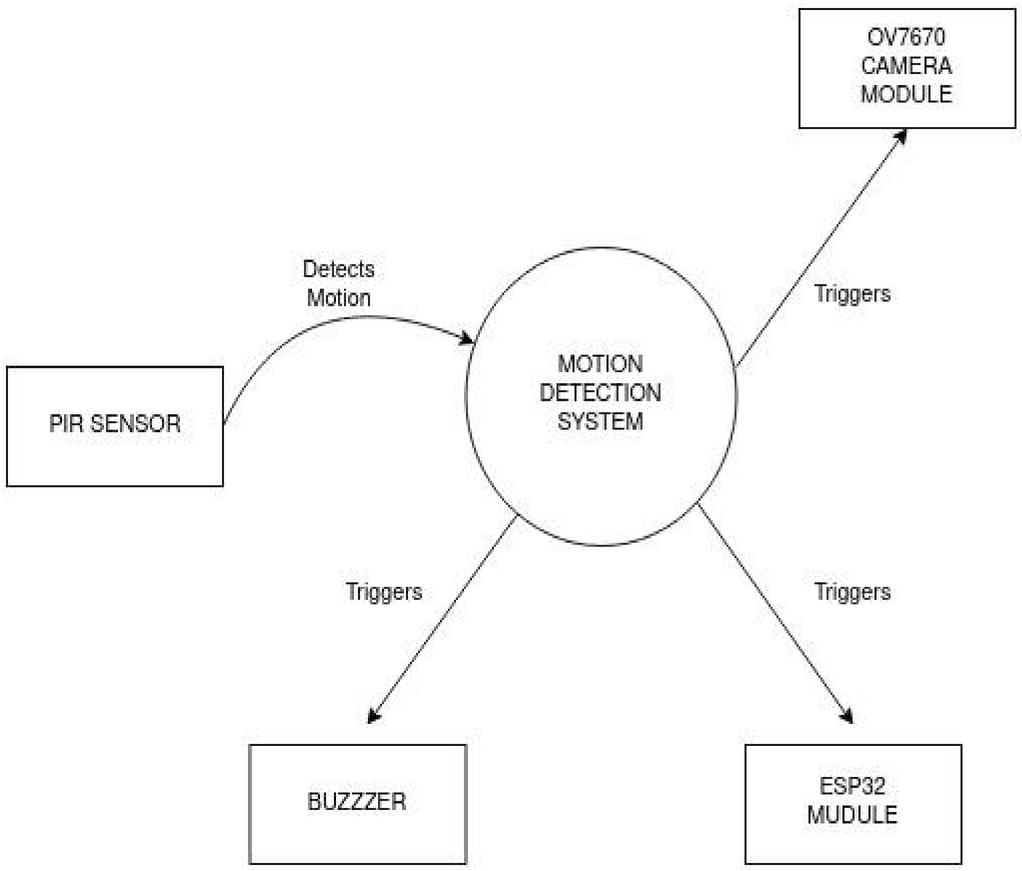


FIGURE 4 : DFD LEVEL 1

4.3. Level-2 DFD

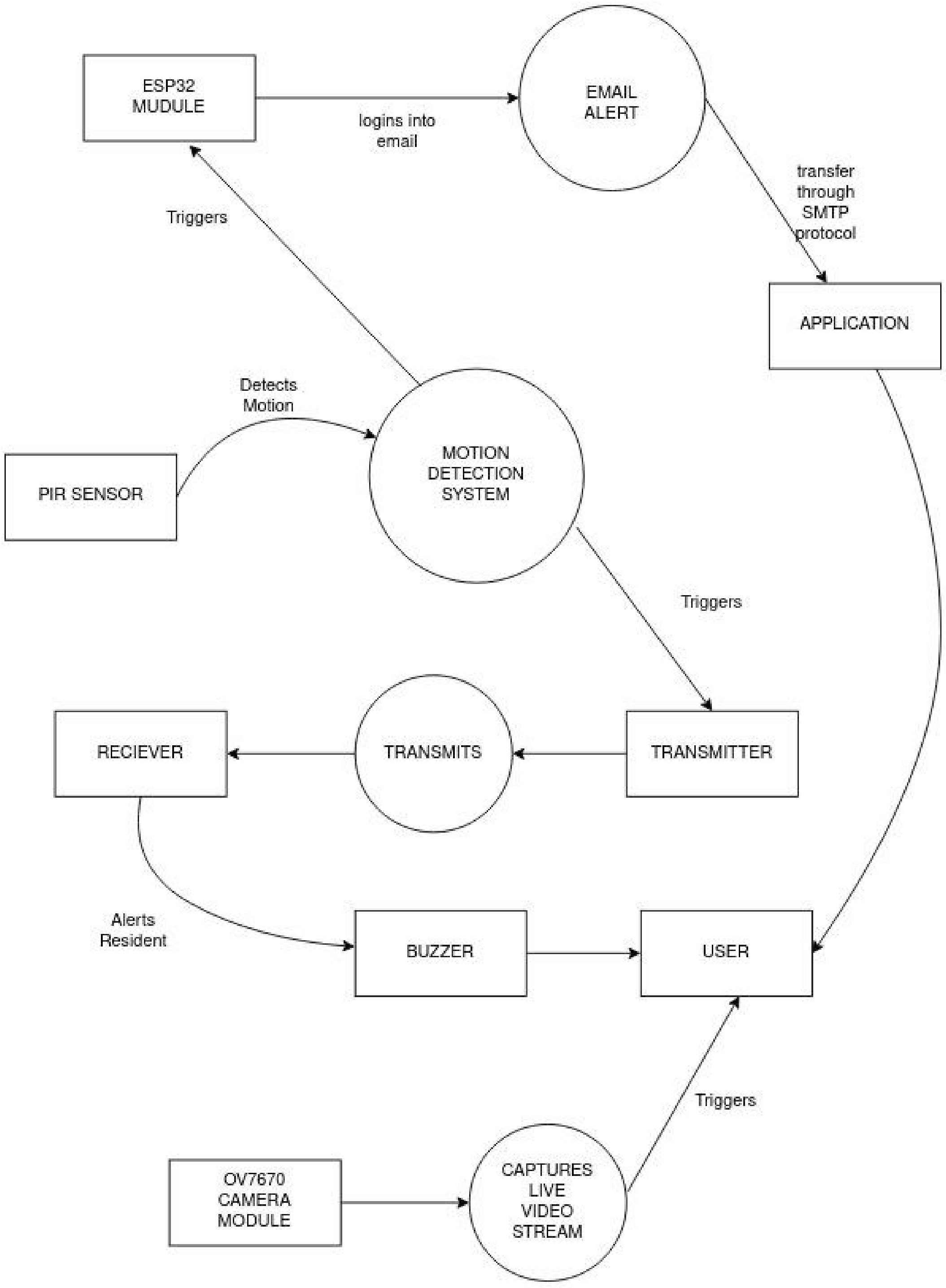


FIGURE 5 : DFD LEVEL 2

5. Class Diagram

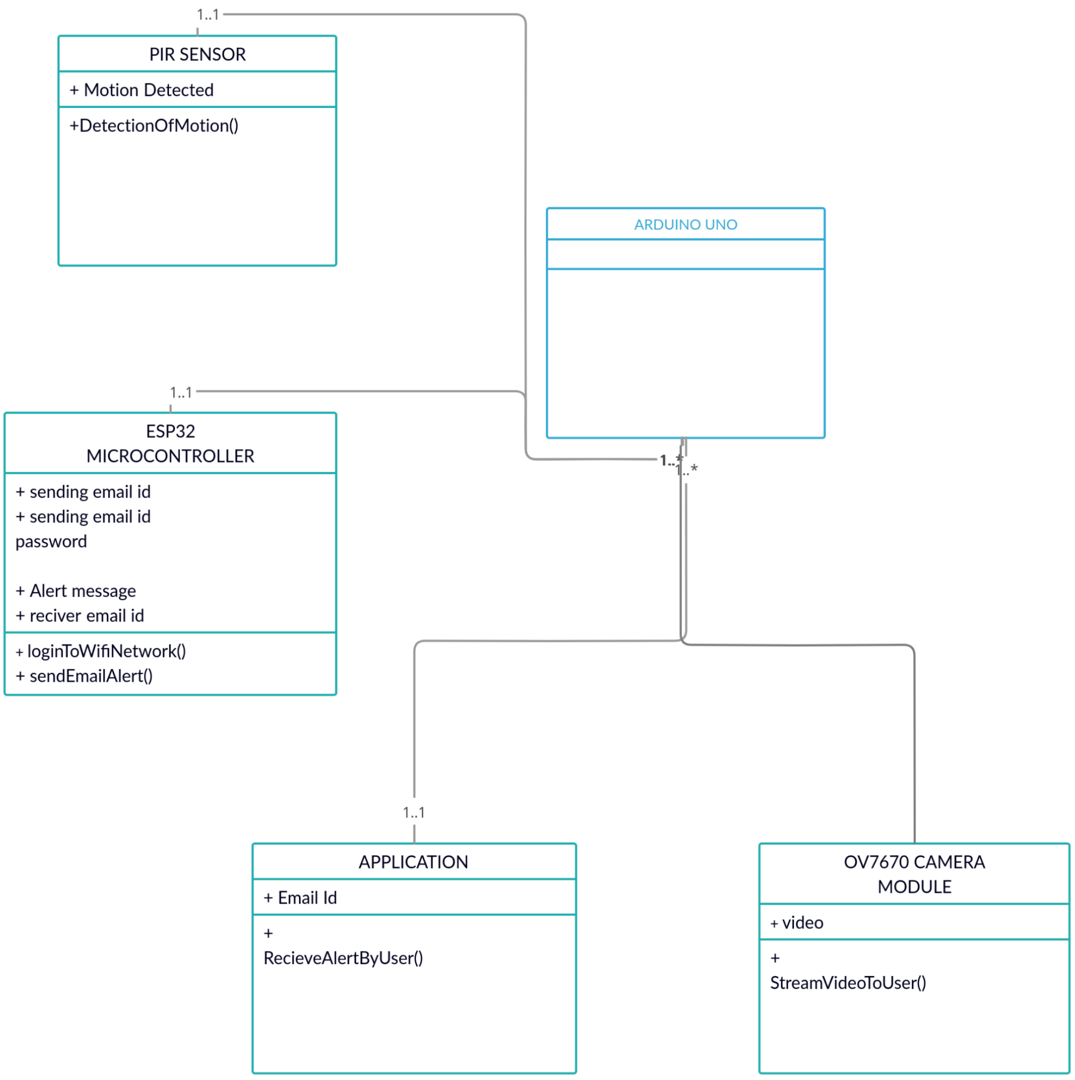


FIGURE 6 : CLASS DIAGRAM

6. Sequence Diagram

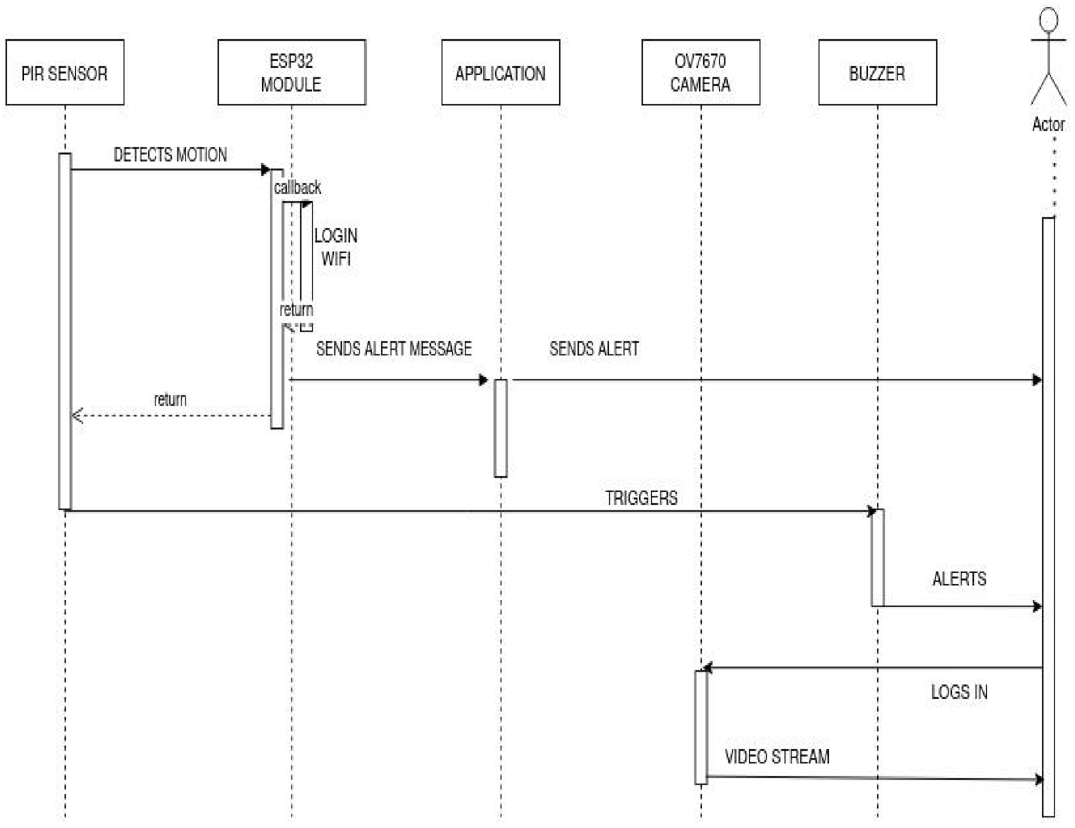


FIGURE 7 : SEQUENCE DIAGRAM

7. Activity Diagram

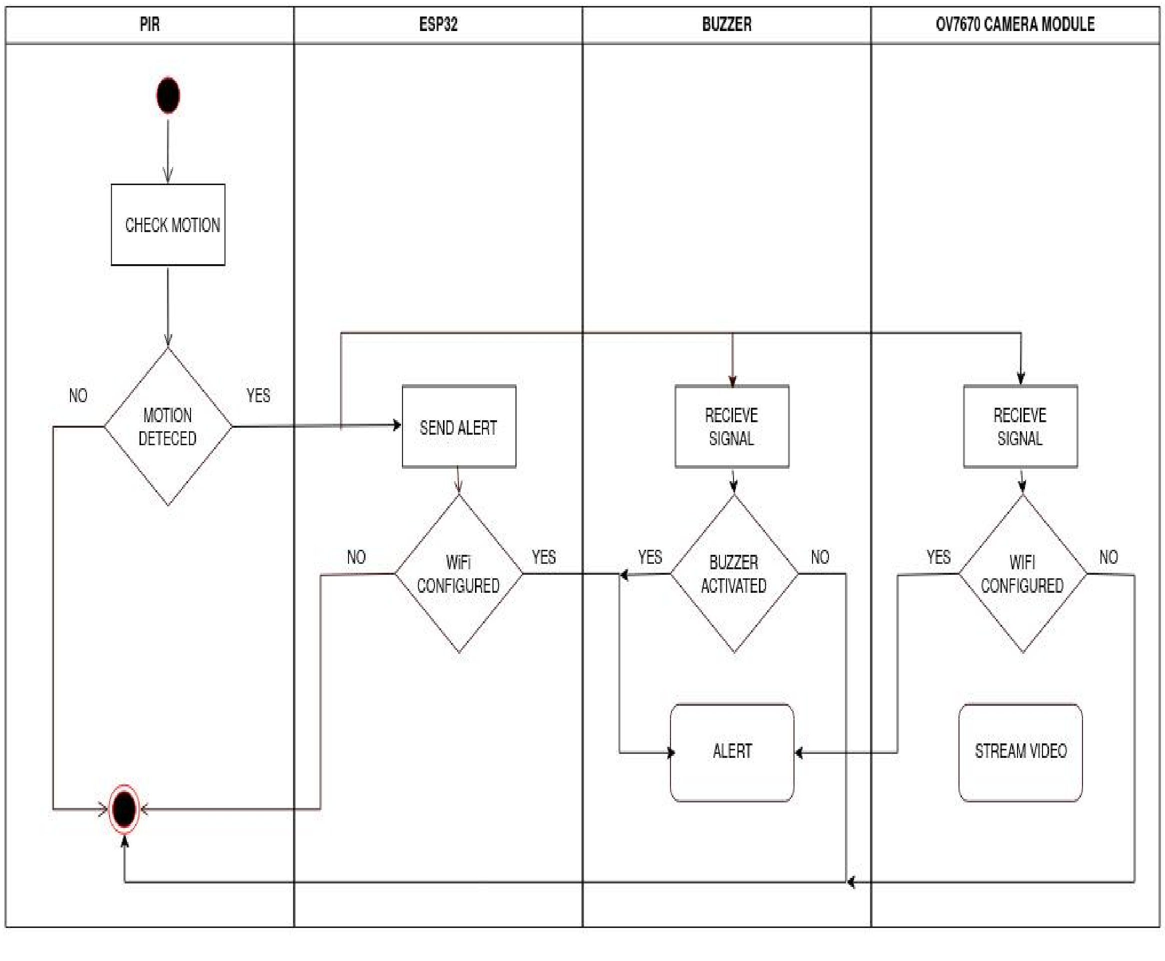


FIGURE 8 : ACTIVITY DIAGRAM

8. Circuit Diagram

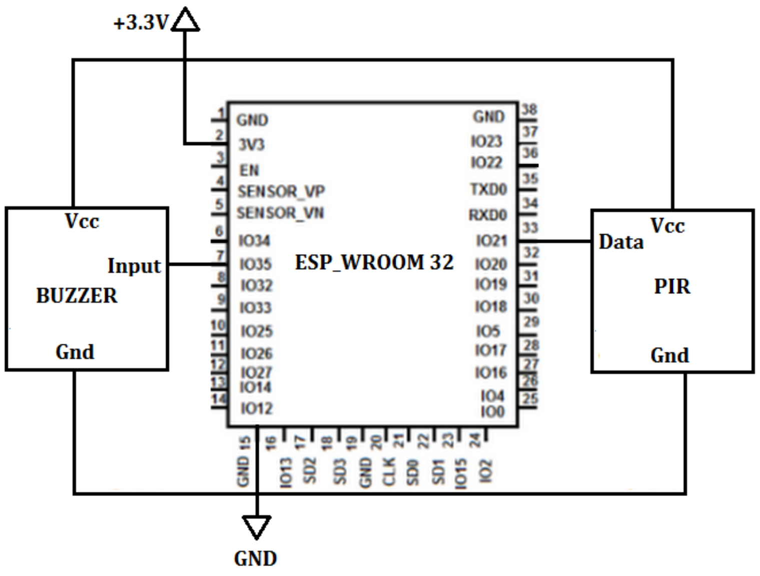


FIGURE 9 : CIRCUIT DIAGRAM

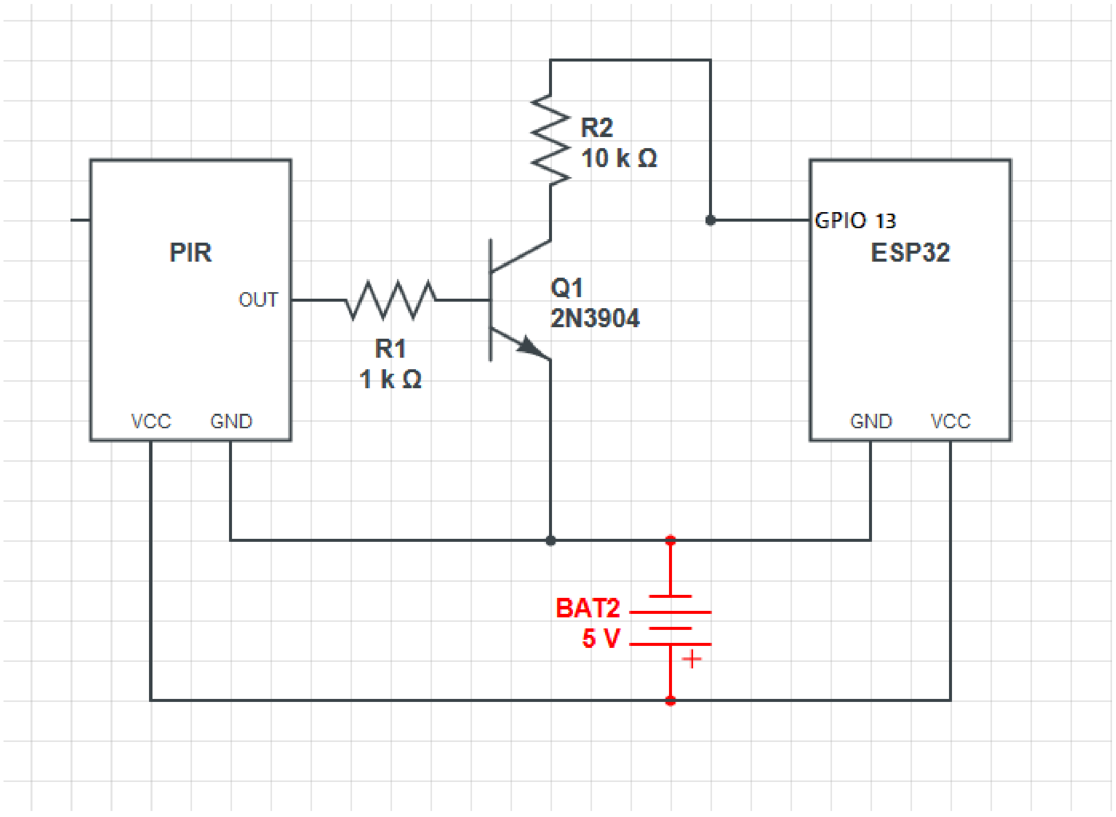


FIGURE 10 : CIRCUIT DIAGRAM

9. Connection Diagrams

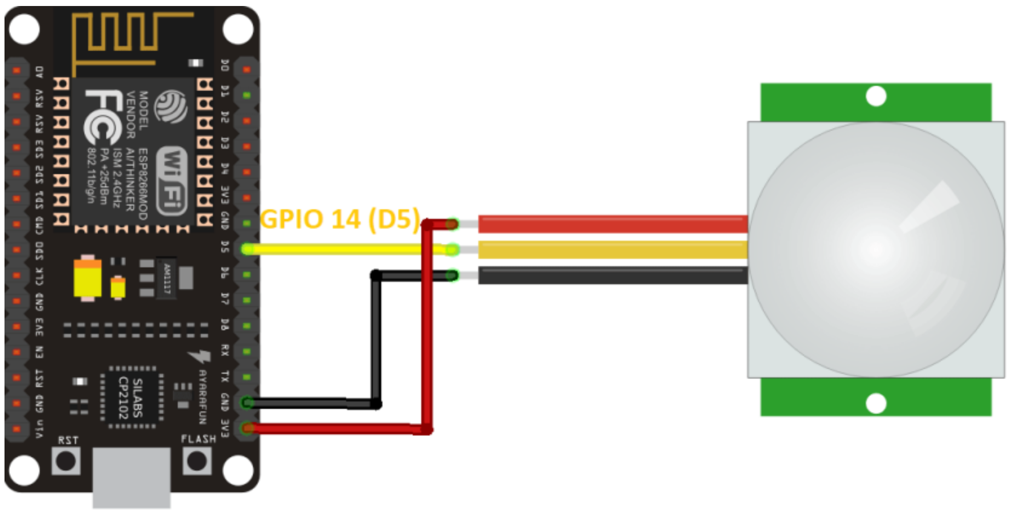


FIGURE 11 : CONNECTION DIAGRAM

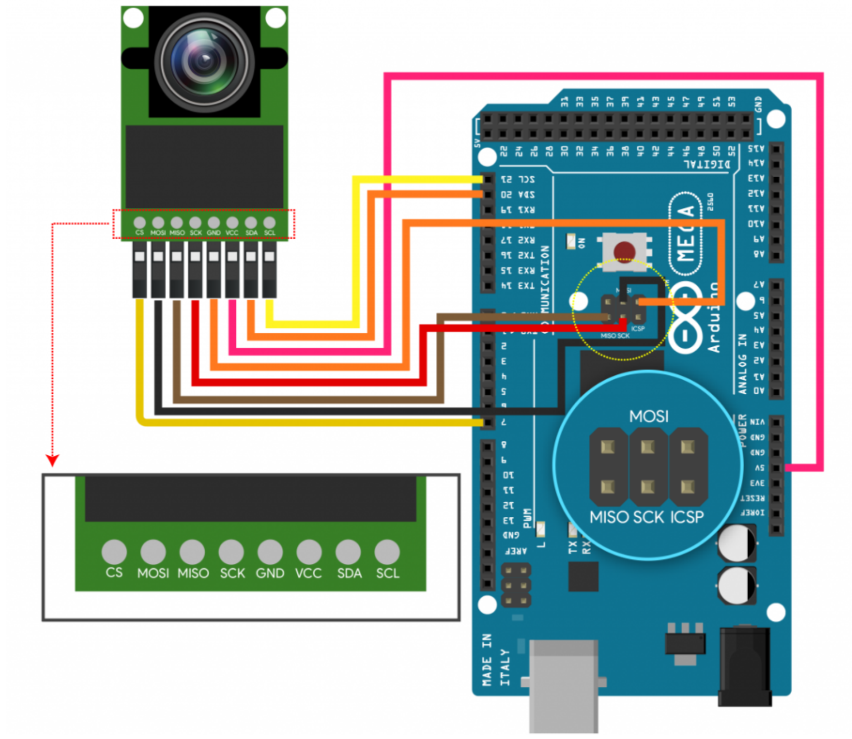
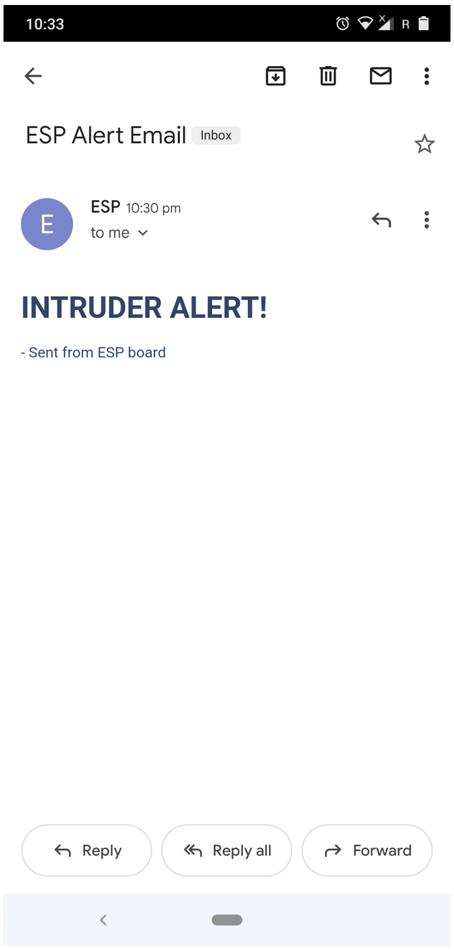


FIGURE 12 : CONNECTION DIAGRAM

3. *Implementation*

3.1 Screenshots

1. Graphical User Interface

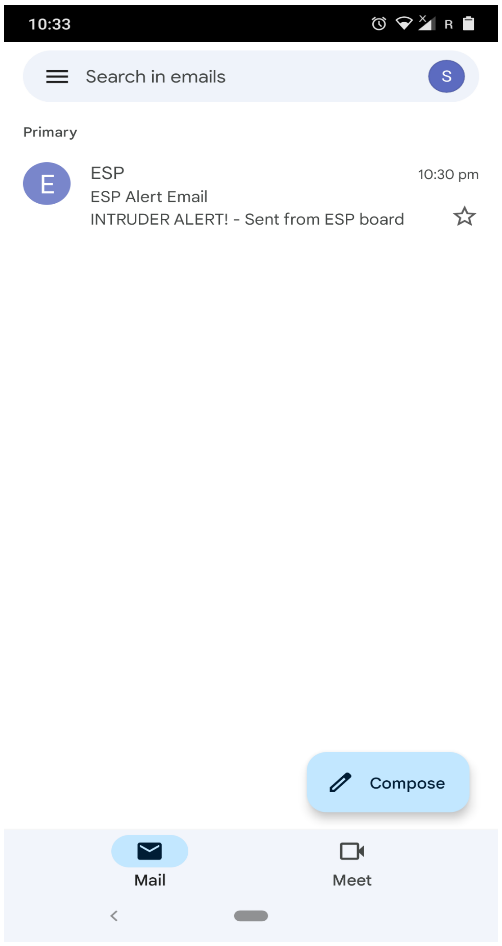


ESP Alert Email Inbox

ESP 10:30 pm
to me

INTRUDER ALERT!

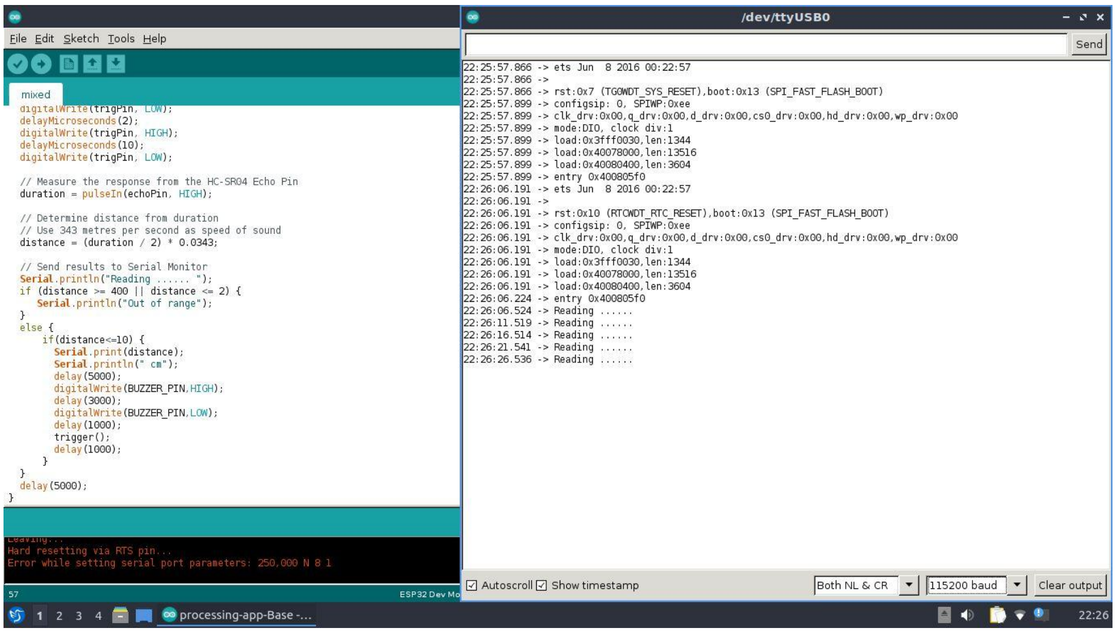
- Sent from ESP board



Primary

ESP 10:30 pm
ESP Alert Email
INTRUDER ALERT! - Sent from ESP board

When the PIR is triggered this kind of alert will be sent through the ESP32 module using SMTP Protocol.
The user can view this alert and take actions accordingly.



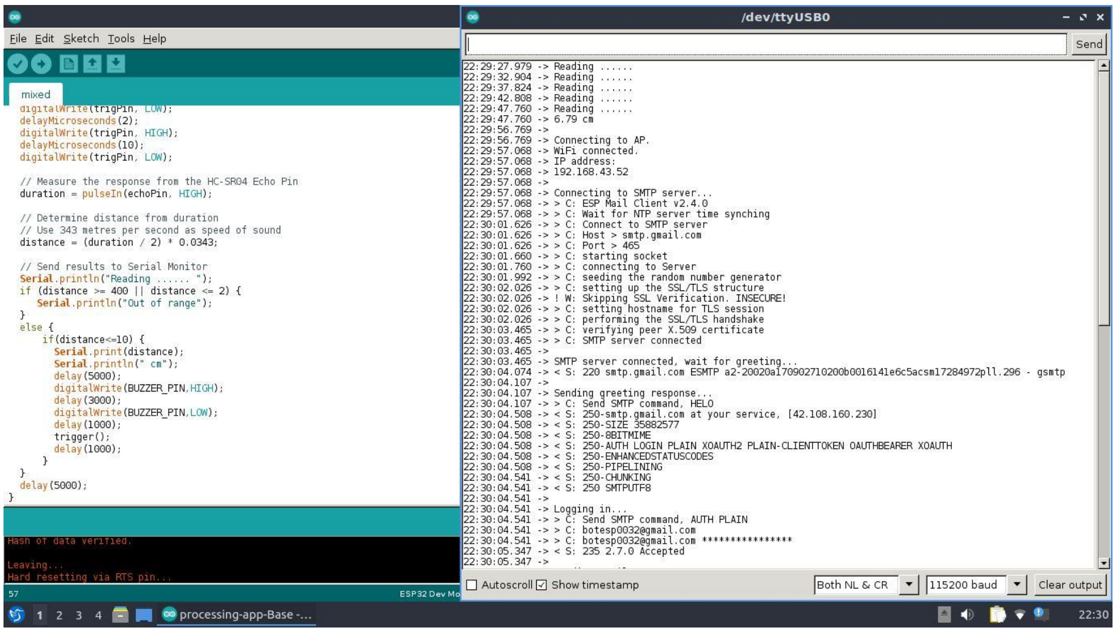
```
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);

// Measure the response from the HC-SR04 Echo Pin
duration = pulseIn(echoPin, HIGH);

// Determine distance from duration
// Use 343 metres per second as speed of sound
distance = (duration / 2) * 0.0343;

// Send results to Serial Monitor
Serial.println("Reading ...... ");
if (distance >= 400 || distance <= 2) {
  Serial.println("Out of range");
}
else {
    if(distance<=10) {
      Serial.print(distance);
      Serial.println(" cm");
      delay(5000);
      digitalWrite(BUZZER_PIN,HIGH);
      delay(3000);
      digitalWrite(BUZZER_PIN,LOW);
      delay(1000);
      trigger();
      delay(1000);
    }
}
delay(5000);
}
```

```
Leaving...
Hard resetting via RTS pin...
Error while setting serial port parameters: 250,000 N 8 1
```

```
22:25:57.866 -> ets Jun  8 2016 00:22:57
22:25:57.866 ->
22:25:57.866 -> rst:0x7 (TG0WDT_SYS_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
22:25:57.899 -> configsip: 0, SPIWP:0xee
22:25:57.899 -> clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
22:25:57.899 -> mode:DIO, clock div:1
22:25:57.899 -> load:0x3fff0030,len:1344
22:25:57.899 -> load:0x40078000,len:13516
22:25:57.899 -> load:0x40080400,len:3604
22:25:57.899 -> entry 0x400805f0
22:26:06.191 -> ets Jun  8 2016 00:22:57
22:26:06.191 ->
22:26:06.191 -> rst:0x10 (RTCWDT_RTC_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
22:26:06.191 -> configsip: 0, SPIWP:0xee
22:26:06.191 -> clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
22:26:06.191 -> mode:DIO, clock div:1
22:26:06.191 -> load:0x3fff0030,len:1344
22:26:06.191 -> load:0x40078000,len:13516
22:26:06.191 -> load:0x40080400,len:3604
22:26:06.224 -> entry 0x400805f0
22:26:06.524 -> Reading ......
22:26:11.519 -> Reading ......
22:26:16.514 -> Reading ......
22:26:21.541 -> Reading ......
22:26:26.536 -> Reading ......
```

☑ Autoscroll ☑ Show timestamp — Both NL & CR — 115200 baud — Clear output

This image shows the PIR sensor reading whether there is an intruder or not.



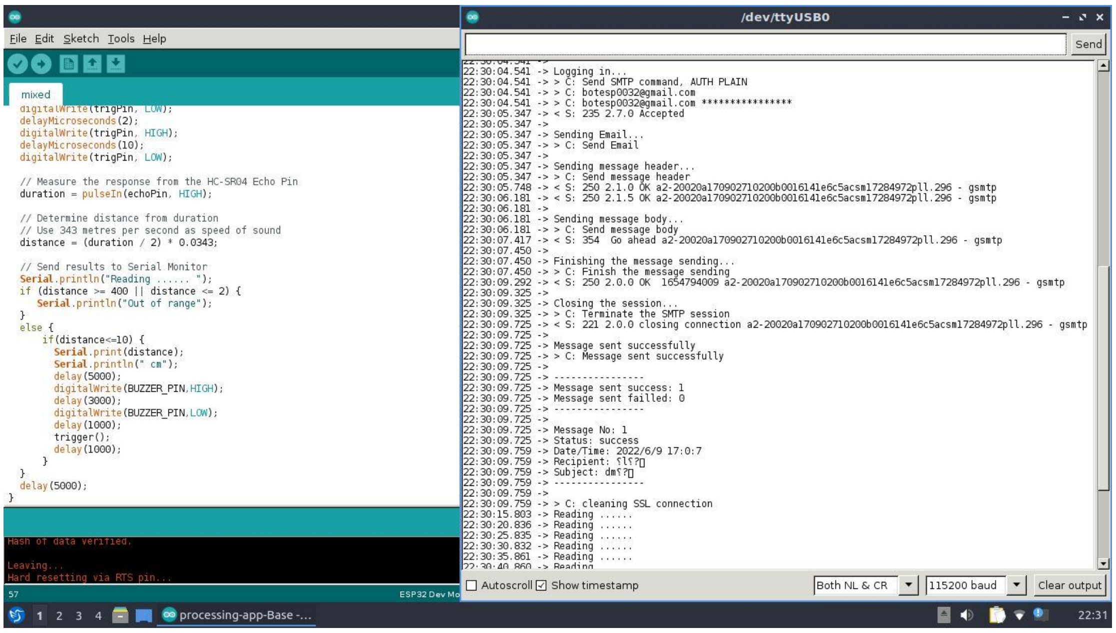
```
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);

// Measure the response from the HC-SR04 Echo Pin
duration = pulseIn(echoPin, HIGH);

// Determine distance from duration
// Use 343 metres per second as speed of sound
distance = (duration / 2) * 0.0343;

// Send results to Serial Monitor
Serial.println("Reading ...... ");
if (distance >= 400 || distance <= 2) {
  Serial.println("Out of range");
}
else {
    if(distance<=10) {
      Serial.print(distance);
      Serial.println(" cm");
      delay(5000);
      digitalWrite(BUZZER_PIN,HIGH);
      delay(3000);
      digitalWrite(BUZZER_PIN,LOW);
      delay(1000);
      trigger();
      delay(1000);
    }
}
delay(5000);
}
```

```
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

```
22:29:27.979 -> Reading ......
22:29:32.904 -> Reading ......
22:29:37.824 -> Reading ......
22:29:42.808 -> Reading ......
22:29:47.760 -> Reading ......
22:29:47.760 -> 6.79 cm
22:29:56.769 ->
22:29:56.769 -> Connecting to AP.
22:29:57.068 -> WiFi connected.
22:29:57.068 -> IP address:
22:29:57.068 -> 192.168.43.52
22:29:57.068 ->
22:29:57.068 -> Connecting to SMTP server...
22:29:57.068 -> > C: ESP Mail Client v2.4.0
22:29:57.068 -> > C: Wait for NTP server time synching
22:30:01.626 -> > C: Connect to SMTP server
22:30:01.626 -> > C: Host > smtp.gmail.com
22:30:01.626 -> > C: Port > 465
22:30:01.660 -> > C: starting socket
22:30:01.760 -> > C: connecting to Server
22:30:01.992 -> > C: seeding the random number generator
22:30:02.026 -> > C: setting up the SSL/TLS structure
22:30:02.026 -> ! W: Skipping SSL Verification. INSECURE!
22:30:02.026 -> > C: setting hostname for TLS session
22:30:02.026 -> > C: performing the SSL/TLS handshake
22:30:03.465 -> > C: verifying peer X.509 certificate
22:30:03.465 -> > C: SMTP server connected
22:30:03.465 ->
22:30:03.465 -> SMTP server connected, wait for greeting...
22:30:04.074 -> < S: 220 smtp.gmail.com ESMTP a2-20020a170902710200b0016141e6c5acsm17284972pll.296 - gsmtp
22:30:04.107 ->
22:30:04.107 -> Sending greeting response...
22:30:04.107 -> > C: Send SMTP command, HELO
22:30:04.508 -> < S: 250-smtp.gmail.com at your service, [42.108.160.230]
22:30:04.508 -> < S: 250-SIZE 35882577
22:30:04.508 -> < S: 250-8BITMIME
22:30:04.508 -> < S: 250-AUTH LOGIN PLAIN XOAUTH2 PLAIN-CLIENTTOKEN OAUTHBEARER XOAUTH
22:30:04.508 -> < S: 250-ENHANCEDSTATUSCODES
22:30:04.508 -> < S: 250-PIPELINING
22:30:04.541 -> < S: 250-CHUNKING
22:30:04.541 -> < S: 250 SMTPUTF8
22:30:04.541 ->
22:30:04.541 -> Logging in...
22:30:04.541 -> > C: Send SMTP command, AUTH PLAIN
22:30:04.541 -> > C: botesp0032@gmail.com
22:30:04.541 -> > C: botesp0032@gmail.com ****************
22:30:05.347 -> < S: 235 2.7.0 Accepted
22:30:05.347 ->
```

☐ Autoscroll ☑ Show timestamp — Both NL & CR — 115200 baud — Clear output

Once a movement is detected PIR will send signal to ESP32. This images shows ESP32 connecting to the Wifi to send alert



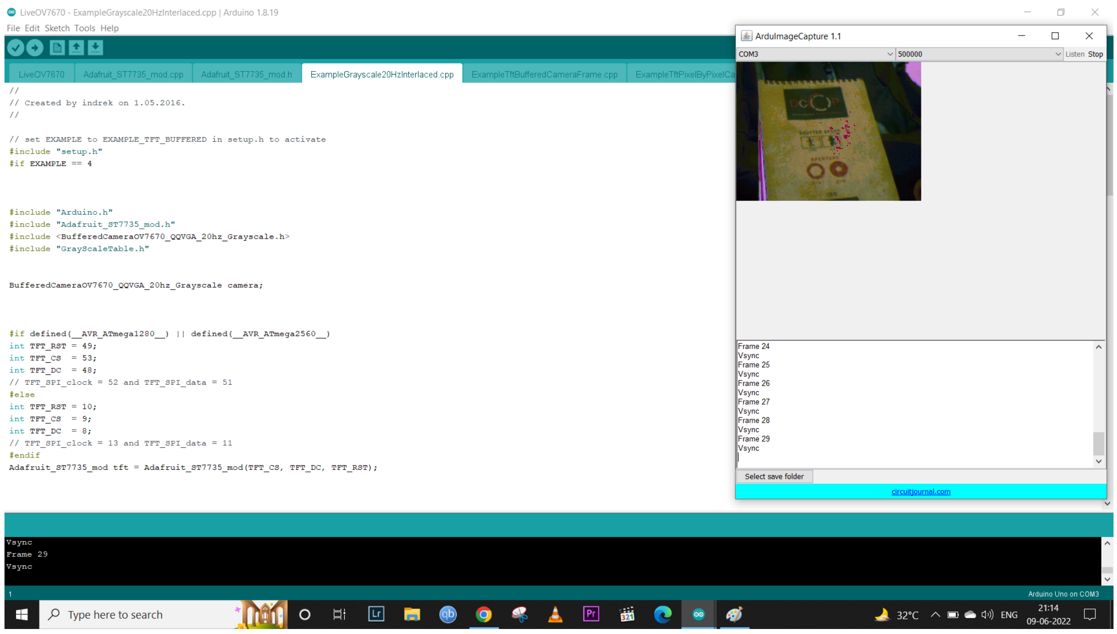
```
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Measure the response from the HC-SR04 Echo Pin
  duration = pulseIn(echoPin, HIGH);

  // Determine distance from duration
  // Use 343 metres per second as speed of sound
  distance = (duration / 2) * 0.0343;

  // Send results to Serial Monitor
  Serial.println("Reading ...... ");
  if (distance >= 400 || distance <= 2) {
     Serial.println("Out of range");
  }
  else {
      if(distance<=10) {
        Serial.print(distance);
        Serial.println(" cm");
        delay(5000);
        digitalWrite(BUZZER_PIN,HIGH);
        delay(3000);
        digitalWrite(BUZZER_PIN,LOW);
        delay(1000);
        trigger();
        delay(1000);
      }
  }
  delay(5000);
}
```

```
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

```
22:30:04.541 ->
22:30:04.541 -> Logging in...
22:30:04.541 -> > C: Send SMTP command, AUTH PLAIN
22:30:04.541 -> > C: botesp0032@gmail.com
22:30:04.541 -> > C: botesp0032@gmail.com ****************
22:30:05.347 -> < S: 235 2.7.0 Accepted
22:30:05.347 ->
22:30:05.347 -> Sending Email...
22:30:05.347 -> > C: Send Email
22:30:05.347 ->
22:30:05.347 -> Sending message header...
22:30:05.347 -> > C: Send message header
22:30:05.748 -> < S: 250 2.1.0 OK a2-20020a170902710200b0016141e6c5acsm17284972pll.296 - gsmtp
22:30:06.181 -> < S: 250 2.1.5 OK a2-20020a170902710200b0016141e6c5acsm17284972pll.296 - gsmtp
22:30:06.181 ->
22:30:06.181 -> Sending message body...
22:30:06.181 -> > C: Send message body
22:30:07.417 -> < S: 354  Go ahead a2-20020a170902710200b0016141e6c5acsm17284972pll.296 - gsmtp
22:30:07.450 ->
22:30:07.450 -> Finishing the message sending...
22:30:07.450 -> > C: Finish the message sending
22:30:09.292 -> < S: 250 2.0.0 OK  1654794009 a2-20020a170902710200b0016141e6c5acsm17284972pll.296 - gsmtp
22:30:09.325 ->
22:30:09.325 -> Closing the session...
22:30:09.325 -> > C: Terminate the SMTP session
22:30:09.725 -> < S: 221 2.0.0 closing connection a2-20020a170902710200b0016141e6c5acsm17284972pll.296 - gsmtp
22:30:09.725 ->
22:30:09.725 -> Message sent successfully
22:30:09.725 -> > C: Message sent successfully
22:30:09.725 ->
22:30:09.725 -> ----------------
22:30:09.725 -> Message sent success: 1
22:30:09.725 -> Message sent failled: 0
22:30:09.725 -> ----------------
22:30:09.725 ->
22:30:09.725 -> Message No: 1
22:30:09.725 -> Status: success
22:30:09.759 -> Date/Time: 2022/6/9 17:0:7
22:30:09.759 -> Recipient: ʕlʕ?▯
22:30:09.759 -> Subject: dmʕ?▯
22:30:09.759 -> ----------------
22:30:09.759 ->
22:30:09.759 -> > C: cleaning SSL connection
22:30:15.803 -> Reading ......
22:30:20.836 -> Reading ......
22:30:25.835 -> Reading ......
22:30:30.832 -> Reading ......
22:30:35.861 -> Reading ......
22:30:40.860 -> Reading
```

☐ Autoscroll ☑ Show timestamp Both NL & CR 115200 baud Clear output

This image shows successful connection of ESP32 to wifi and sending the alert. AFter alert is sent the ESP goes back to sleep and PIR starts detecting motion again



```
//
// Created by indrek on 1.05.2016.
//

// set EXAMPLE to EXAMPLE_TFT_BUFFERED in setup.h to activate
#include "setup.h"
#if EXAMPLE == 4



#include "Arduino.h"
#include "Adafruit_ST7735_mod.h"
#include <BufferedCameraOV7670_QQVGA_20hz_Grayscale.h>
#include "GrayScaleTable.h"


BufferedCameraOV7670_QQVGA_20hz_Grayscale camera;



#if defined(__AVR_ATmega1280__) || defined(__AVR_ATmega2560__)
int TFT_RST = 49;
int TFT_CS  = 53;
int TFT_DC  = 48;
// TFT_SPI_clock = 52 and TFT_SPI_data = 51
#else
int TFT_RST = 10;
int TFT_CS  = 9;
int TFT_DC  = 8;
// TFT_SPI_clock = 13 and TFT_SPI_data = 11
#endif
Adafruit_ST7735_mod tft = Adafruit_ST7735_mod(TFT_CS, TFT_DC, TFT_RST);
```

```
Frame 24
Vsync
Frame 25
Vsync
Frame 26
Vsync
Frame 27
Vsync
Frame 28
Vsync
Frame 29
Vsync
```

```
Vsync
Frame 29
Vsync
```

If the alert seems fishy the user can sue the localhost to view a livestream video of the door or where the camera is placed.

These two images screenshots are taken 2 frames apart to show functionality of the live stream instead of just capturing images. A video output provides more security and clear idea then just what an image could have captured.

3.2 Code

1. Code for PIR sensor triggering ESP32 , buzzer and ESP sending email over the wifi.

```
#include <Arduino.h>
#if defined(ESP32)
  #include <WiFi.h>
#elif defined(ESP8266)
  #include <ESP8266WiFi.h>
#endif
#include <ESP_Mail_Client.h>

#define WIFI_SSID "Nokia"
#define WIFI_PASSWORD "foobar123"

#define SMTP_HOST "smtp.gmail.com"
#define SMTP_PORT 465

/* The sign in credentials */
#define AUTHOR_EMAIL "botesp0032@gmail.com"
#define AUTHOR_PASSWORD "tzsernorvhblqrao"

/* Recipient's email*/
#define RECIPIENT_EMAIL "snowballrohini@gmail.com"

/* The SMTP Session object used for Email sending */
SMTPSession smtp;

/* Callback function to get the Email sending status */
void smtpCallback(SMTP_Status status);
```

```
#define trigPin 23
#define echoPin 22
#define BUZZER_PIN 17


float duration, distance;
void setup() {
  // put your setup code here, to run once:
  Serial.begin (115200);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(BUZZER_PIN, OUTPUT);
}


void trigger() {
  Serial.begin(115200);
  Serial.println();
  Serial.print("Connecting to AP");
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  while (WiFi.status() != WL_CONNECTED){
    Serial.print(".");
    delay(200);
  }
  Serial.println("");
  Serial.println("WiFi connected.");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  Serial.println();

  /** Enable the debug via Serial port
   * none debug or 0
   * basic debug or 1
```

```
 */
 smtp.debug(1);

 /* Set the callback function to get the sending results */
 smtp.callback(smtpCallback);

 /* Declare the session config data */
 ESP_Mail_Session session;

 /* Set the session config */
 session.server.host_name = SMTP_HOST;
 session.server.port = SMTP_PORT;
 session.login.email = AUTHOR_EMAIL;
 session.login.password = AUTHOR_PASSWORD;
 session.login.user_domain = "";

 /* Declare the message class */
 SMTP_Message message;

 /* Set the message headers */
 message.sender.name = "ESP";
 message.sender.email = AUTHOR_EMAIL;
 message.subject = "ESP Alert Email";
 message.addRecipient("Sara", RECIPIENT_EMAIL);

 /*Send HTML message*/
    String htmlMsg = "<div style=\"color:#2f4468;\"><h1>INTRUDER
ALERT!</h1><p>- Sent from ESP board</p></div>";
 message.html.content = htmlMsg.c_str();
 message.html.content = htmlMsg.c_str();
 message.text.charSet = "us-ascii";
 message.html.transfer_encoding = Content_Transfer_Encoding::enc_7bit;

 /*
```

```
  //Send raw text message
  String textMsg = "INTRUDER ALERT! - Sent from ESP board";
  message.text.content = textMsg.c_str();
  message.text.charSet = "us-ascii";
  message.text.transfer_encoding = Content_Transfer_Encoding::enc_7bit;

  message.priority = esp_mail_smtp_priority::esp_mail_smtp_priority_low;
          message.response.notify     =     esp_mail_smtp_notify_success     |
esp_mail_smtp_notify_failure | esp_mail_smtp_notify_delay;*/

  /* Set the custom message header */
  //message.addHeader("Message-ID: <abcde.fghij@gmail.com>");

  /* Connect to server with the session config */
  if (!smtp.connect(&session))
    return;

  /* Start sending Email and close the session */
  if (!MailClient.sendMail(&smtp, &message))
    Serial.println("Error sending Email, " + smtp.errorReason());
}

/* Callback function to get the Email sending status */
void smtpCallback(SMTP_Status status){
  /* Print the current status */
  Serial.println(status.info());

  /* Print the sending result */
  if (status.success()){
    Serial.println("----------------");
    ESP_MAIL_PRINTF("Message sent success: %d\n", status.completedCount());
    ESP_MAIL_PRINTF("Message sent failled: %d\n", status.failedCount());
    Serial.println("----------------\n");
    struct tm dt;
    for (size_t i = 0; i < smtp.sendingResult.size(); i++){
      /* Get the result item */
      SMTP_Result result = smtp.sendingResult.getItem(i);
      time_t ts = (time_t)result.timestamp;
      localtime_r(&ts, &dt);

      ESP_MAIL_PRINTF("Message No: %d\n", i + 1);
      ESP_MAIL_PRINTF("Status: %s\n", result.completed ? "success" : "failed");
       ESP_MAIL_PRINTF("Date/Time: %d/%d/%d %d:%d:%d\n", dt.tm_year + 1900,
dt.tm_mon + 1, dt.tm_mday, dt.tm_hour, dt.tm_min, dt.tm_sec);
      ESP_MAIL_PRINTF("Recipient: %s\n", result.recipients);
      ESP_MAIL_PRINTF("Subject: %s\n", result.subject);
    }
    Serial.println("----------------\n");
  }

}
void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Measure the response from the HC-SR04 Echo Pin
  duration = pulseIn(echoPin, HIGH);

  // Determine distance from duration
  // Use 343 metres per second as speed of sound
  distance = (duration / 2) * 0.0343;

  // Send results to Serial Monitor
```

```
 Serial.println("Reading ...... ");
 if (distance >= 400 || distance <= 2) {
   Serial.println("Out of range");
 }
 else {
    if(distance<=10) {
      Serial.print(distance);
      Serial.println(" cm");
      delay(5000);
      digitalWrite(BUZZER_PIN,HIGH);
      delay(3000);
      digitalWrite(BUZZER_PIN,LOW);
      delay(1000);
      trigger();
      delay(1000);
    }
 }
 delay(5000);
}
```

2) Code for functioning of OV7670 camera module

LIVEOV7670.ino

```
// change setup.h to switch between buffered and pixel-by-pixel processing
#include "setup.h"

void setup() {
  // This is not necessary and has no effect for ATMEGA based Arduinos.
  // WAVGAT Nano has slower clock rate by default. We want to reset it to maximum
speed
  CLKPR = 0x80; // enter clock rate change mode
  CLKPR = 0; // set prescaler to 0. WAVGAT MCU has it 3 by default.

  initializeScreenAndCamera();
}

void loop() {
  processFrame();
}
```

SETUP.h

```
ifndef LIVEOV7670_SETUP_H
#define LIVEOV7670_SETUP_H

#define EXAMPLE 1

void initializeScreenAndCamera();
void processFrame();

#endif //LIVEOV7670_SETUP_H
```

ADAFRUIT_ST7735_mod.h

```
#ifndef _ADAFRUIT_ST7735_MOD_H_
#define _ADAFRUIT_ST7735_MOD_H_

#if ARDUINO >= 100
 #include "Arduino.h"
 #include "Print.h"
#else
 #include "WProgram.h"
#endif

#include <Adafruit_GFX.h>

#if defined(__SAM3X8E__)
  #include <include/pio.h>
  #define PROGMEM
  #define pgm_read_byte(addr) (*(const unsigned char *)(addr))
  #define pgm_read_word(addr) (*(const unsigned short *)(addr))
  typedef unsigned char prog_uchar;
#elif defined(__AVR__)
  #include <avr/pgmspace.h>
#elif defined(ESP8266)
  #include <pgmspace.h>
#endif

#if defined(__SAM3X8E__)
    #undef __FlashStringHelper::F(string_literal)
    #define F(string_literal) string_literal
#endif

// some flags for initR() :(
#define INITR_GREENTAB 0x0
```

```
#define INITR_REDTAB   0x1
#define INITR_BLACKTAB   0x2

#define INITR_18GREENTAB         INITR_GREENTAB
#define INITR_18REDTAB   INITR_REDTAB
#define INITR_18BLACKTAB         INITR_BLACKTAB
#define INITR_144GREENTAB   0x1

#define ST7735_TFTWIDTH  128
// for 1.44" display
#define ST7735_TFTHEIGHT_144 128
// for 1.8" display
#define ST7735_TFTHEIGHT_18  160

#define ST7735_NOP            0x00
#define ST7735_SWRESET 0x01
#define ST7735_RDDID   0x04
#define ST7735_RDDST   0x09

#define ST7735_SLPIN   0x10
#define ST7735_SLPOUT  0x11
#define ST7735_PTLON   0x12
#define ST7735_NORON   0x13

#define ST7735_INVOFF  0x20
#define ST7735_INVON   0x21
#define ST7735_DISPOFF 0x28
#define ST7735_DISPON  0x29
#define ST7735_CASET   0x2A
#define ST7735_RASET   0x2B
#define ST7735_RAMWR   0x2C
#define ST7735_RAMRD   0x2E

#define ST7735_PTLAR   0x30
```

```
#define ST7735_COLMOD  0x3A
#define ST7735_MADCTL  0x36

#define ST7735_FRMCTR1 0xB1
#define ST7735_FRMCTR2 0xB2
#define ST7735_FRMCTR3 0xB3
#define ST7735_INVCTR  0xB4
#define ST7735_DISSET5 0xB6

#define ST7735_PWCTR1  0xC0
#define ST7735_PWCTR2  0xC1
#define ST7735_PWCTR3  0xC2
#define ST7735_PWCTR4  0xC3
#define ST7735_PWCTR5  0xC4
#define ST7735_VMCTR1  0xC5

#define ST7735_RDID1   0xDA
#define ST7735_RDID2   0xDB
#define ST7735_RDID3   0xDC
#define ST7735_RDID4   0xDD

#define ST7735_PWCTR6  0xFC

#define ST7735_GMCTRP1 0xE0
#define ST7735_GMCTRN1 0xE1

// Color definitions   R5:G6:B5
#define    ST7735_BLACK   0x0000
#define    ST7735_BLUE     0x001F
#define    ST7735_RED      0xF800
#define    ST7735_GREEN   0x07E0
#define ST7735_CYAN      0x07FF
#define ST7735_MAGENTA 0xF81F
#define ST7735_YELLOW  0xFFE0
```

```
#define ST7735_WHITE   0xFFFF


class Adafruit_ST7735_mod : public Adafruit_GFX {

 public:

  Adafruit_ST7735_mod(int8_t CS, int8_t RS, int8_t SID, int8_t SCLK, int8_t RST =
-1);
  Adafruit_ST7735_mod(int8_t CS, int8_t RS, int8_t RST = -1);

  void   initB(void),                             // for ST7735B displays
         initR(uint8_t options = INITR_GREENTAB), // for ST7735R
         setAddrWindow(uint8_t x0, uint8_t y0, uint8_t x1, uint8_t y1),
         startAddrWindow(uint8_t x0, uint8_t y0, uint8_t x1, uint8_t y1),
         endAddrWindow(),
         pushColor(uint16_t color),
         fillScreen(uint16_t color),
         drawPixel(int16_t x, int16_t y, uint16_t color),
         drawFastVLine(int16_t x, int16_t y, int16_t h, uint16_t color),
         drawFastHLine(int16_t x, int16_t y, int16_t w, uint16_t color),
         fillRect(int16_t x, int16_t y, int16_t w, int16_t h, uint16_t color),
         setRotation(uint8_t r),
         invertDisplay(boolean i);

  uint16_t Color565(uint8_t r, uint8_t g, uint8_t b);

  /* These are not for current use, 8-bit protocol only!
  uint8_t  readdata(void),
           readcommand8(uint8_t);
  uint16_t readcommand16(uint8_t);
  uint32_t readcommand32(uint8_t);
  void   dummyclock(void);
  */
 private:
  uint8_t  tabcolor;

  void   spiwrite(uint8_t),
           writecommand(uint8_t c),
           writedata(uint8_t d),
           commandList(const uint8_t *addr),
           commonInit(const uint8_t *cmdList);
//uint8_t  spiread(void);

  boolean  hwSPI;

#if defined(__AVR__) || defined(CORE_TEENSY)
  volatile uint8_t *dataport, *clkport, *csport, *rsport;
  uint8_t  _cs, _rs, _rst, _sid, _sclk,
           datapinmask, clkpinmask, cspinmask, rspinmask,
           colstart, rowstart; // some displays need this changed
#elif defined(__arm__)
  volatile RwReg  *dataport, *clkport, *csport, *rsport;
  uint32_t  _cs, _rs, _sid, _sclk,
           datapinmask, clkpinmask, cspinmask, rspinmask,
           colstart, rowstart; // some displays need this changed
  int32_t   _rst;  // Must use signed type since a -1 sentinel is assigned.
#endif

};

#endif
```

ExampleTftPixelByPixelCameraFrame.cpp

```
#include "setup.h"
#if EXAMPLE == 2


#include "Arduino.h"
#include "Adafruit_ST7735_mod.h"
#include "CameraOV7670.h"


// scaler values for specific refresh rates
static const uint8_t FPS_1_Hz = 9;
static const uint8_t FPS_0p5_Hz = 19;
static const uint8_t FPS_0p33_Hz = 29;


static const uint16_t lineLength = 640;
static const uint16_t lineCount = 480;



// Since the 1.8" TFT screen is only 160x128 only top right corner of the
VGA picture is visible.
CameraOV7670
camera(CameraOV7670::RESOLUTION_VGA_640x480,
CameraOV7670::PIXEL_RGB565, FPS_1_Hz);



#if defined(__AVR_ATmega1280__) || defined(__AVR_ATmega2560__)
int TFT_RST = 49;
int TFT_CS  = 53;
int TFT_DC  = 48;
// TFT_SPI_clock = 52 and TFT_SPI_data = 51
#else
int TFT_RST = 10;
int TFT_CS  = 9;
int TFT_DC  = 8;
// TFT_SPI_clock = 13 and TFT_SPI_data = 11
#endif
Adafruit_ST7735_mod tft = Adafruit_ST7735_mod(TFT_CS, TFT_DC,
TFT_RST);
```

```
// this is called in Arduino setup() function
void initializeScreenAndCamera() {
  bool cameraInitialized = camera.init();
  tft.initR(INITR_BLACKTAB);
  if (cameraInitialized) {
    tft.fillScreen(ST7735_BLACK);
  } else {
    tft.fillScreen(ST7735_RED);
    delay(3000);
  }

  TIMSK0 = 0; // disable "millis" timer interrupt
}



inline void screenLineStart(void) __attribute__((always_inline));
inline void screenLineEnd(void) __attribute__((always_inline));
inline void sendPixelByte(uint8_t byte) __attribute__((always_inline));
inline void pixelSendingDelay() __attribute__((always_inline));



// Normally it is a portrait screen. Use it as landscape
uint8_t screen_w = ST7735_TFTHEIGHT_18;
uint8_t screen_h = ST7735_TFTWIDTH;
uint8_t screenLineIndex;




// this is called in Arduino loop() function
void processFrame() {
  uint8_t pixelByte;
  screenLineIndex = screen_h;

  camera.waitForVsync();
  camera.ignoreVerticalPadding();

  for (uint16_t y = 0; y < lineCount; y++) {
    screenLineStart();
    camera.ignoreHorizontalPaddingLeft();

    for (uint16_t x = 0; x < lineLength; x++) {

    camera.waitForPixelClockRisingEdge();
    camera.readPixelByte(pixelByte);
    sendPixelByte(pixelByte);

    camera.waitForPixelClockRisingEdge();
    camera.readPixelByte(pixelByte);
    sendPixelByte(pixelByte);
    }

    camera.ignoreHorizontalPaddingRight();
    pixelSendingDelay(); // prevent sending collision
    screenLineEnd();
  }
}




void screenLineStart()   {
  if (screenLineIndex > 0) screenLineIndex--;
          tft.startAddrWindow(screenLineIndex,      0,      screenLineIndex,
screen_w-1);
}

void screenLineEnd() {
  tft.endAddrWindow();
}


void sendPixelByte(uint8_t byte) {
  SPDR = byte;

  // this must be adjusted if sending loop has more/less instructions
  /*
  asm volatile("nop");
  asm volatile("nop");
  asm volatile("nop");
  asm volatile("nop");
  asm volatile("nop");
  asm volatile("nop");
  asm volatile("nop");
  */
}



void pixelSendingDelay() {
  asm volatile("nop");
  asm volatile("nop");
  asm volatile("nop");
  asm volatile("nop");
  asm volatile("nop");
  asm volatile("nop");
  asm volatile("nop");
  asm volatile("nop");
  asm volatile("nop");
  asm volatile("nop");
  asm volatile("nop");
  asm volatile("nop");
  asm volatile("nop");
  asm volatile("nop");
  asm volatile("nop");
  asm volatile("nop");
  asm volatile("nop");
  asm volatile("nop");

}


#endif
```

ExampleTftBufferedCameraFrame.cpp

```
#include "setup.h"
#if EXAMPLE == 1



#include "Arduino.h"
#include "Adafruit_ST7735_mod.h"
#include <BufferedCameraOV7670_QQVGA_10hz.h>
#include <BufferedCameraOV7670_QQVGA.h>
#include <BufferedCameraOV7670_QVGA.h>
#include <BufferedCameraOV7670_QQVGA_10hz_Grayscale.h>
#include "GrayScaleTable.h"



#define GRAYSCALE_PIXELS 0

#if GRAYSCALE_PIXELS == 1
BufferedCameraOV7670_QQVGA_10hz_Grayscale camera;
#else
BufferedCameraOV7670_QQVGA_10hz
camera(CameraOV7670::PIXEL_RGB565);
//BufferedCameraOV7670_QQVGA camera(CameraOV7670::PIXEL_RGB565,
BufferedCameraOV7670_QQVGA::FPS_5_Hz);
//BufferedCameraOV7670_QQVGA camera(CameraOV7670::PIXEL_RGB565,
BufferedCameraOV7670_QQVGA::FPS_2_Hz);
//BufferedCameraOV7670_QVGA camera(CameraOV7670::PIXEL_RGB565,
BufferedCameraOV7670_QVGA::FPS_2p5_Hz);
#endif




#if defined(__AVR_ATmega1280__) || defined(__AVR_ATmega2560__)
int TFT_RST = 49;
int TFT_CS  = 53;
int TFT_DC  = 48;
// TFT_SPI_clock = 52 and TFT_SPI_data = 51
#else
int TFT_RST = 10;
int TFT_CS  = 9;
int TFT_DC  = 8;
// TFT_SPI_clock = 13 and TFT_SPI_data = 11
#endif
Adafruit_ST7735_mod tft = Adafruit_ST7735_mod(TFT_CS, TFT_DC, TFT_RST);
```

```
// this is called in Arduino setup() function
void initializeScreenAndCamera() {
  bool cameraInitialized = camera.init();
  tft.initR(INITR_BLACKTAB);
  if (cameraInitialized) {
     // flash green screen if camera setup was successful
     tft.fillScreen(ST7735_GREEN);
     delay(1000);
     tft.fillScreen(ST7735_BLACK);
  } else {
     // red screen if failed to connect to camera
     tft.fillScreen(ST7735_RED);
     delay(3000);
  }
}




inline void sendLineToDisplay() __attribute__((always_inline));
inline void screenLineStart(void) __attribute__((always_inline));
inline void screenLineEnd(void) __attribute__((always_inline));
inline void sendPixelByte(uint8_t byte) __attribute__((always_inline));


// Normally it is a portrait screen. Use it as landscape
uint8_t screen_w = ST7735_TFTHEIGHT_18;
uint8_t screen_h = ST7735_TFTWIDTH;
uint8_t screenLineIndex;




// this is called in Arduino loop() function
void processFrame() {
  screenLineIndex = screen_h;

  noInterrupts();
  camera.waitForVsync();
  camera.ignoreVerticalPadding();

  for (uint8_t i = 0; i < camera.getLineCount(); i++) {
     camera.readLine();
     sendLineToDisplay();
  }
  interrupts();
}
```

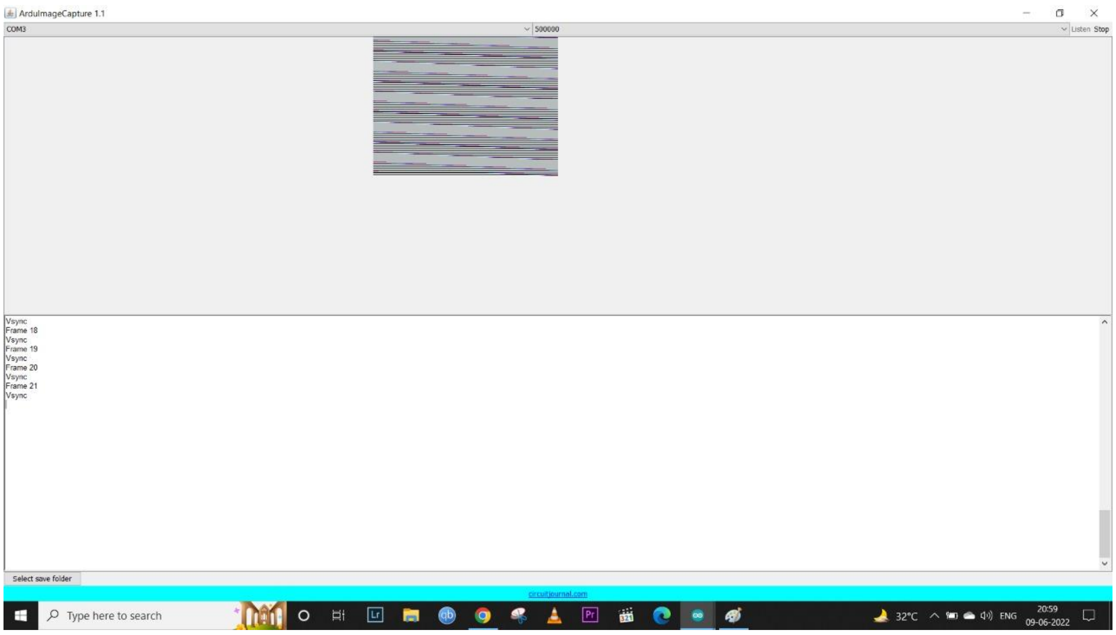
```
static const uint16_t byteCountForDisplay = camera.getPixelBufferLength() <
screen_w*2 ?
                                camera.getPixelBufferLength() : screen_w*2;


void sendLineToDisplay() {
  if (screenLineIndex > 0) {

      screenLineStart();
#if GRAYSCALE_PIXELS == 1
      for (uint16_t i=0; i<camera.getLineLength(); i++) {
      sendPixelByte(graysScaleTableHigh[camera.getPixelByte(i)]);
      sendPixelByte(graysScaleTableLow[camera.getPixelByte(i)]);
      }
#else
      for (uint16_t i=0; i<byteCountForDisplay; i++) {
      sendPixelByte(camera.getPixelByte(i));
      }
#endif
      screenLineEnd();
  }
}


void screenLineStart()   {
  if (screenLineIndex > 0) screenLineIndex--;
  tft.startAddrWindow(screenLineIndex, 0, screenLineIndex, screen_w-1);
}

void screenLineEnd() {
  tft.endAddrWindow();
}


void sendPixelByte(uint8_t byte) {
  SPDR = byte;

  // this must be adjusted if sending loop has more/less instructions

  asm volatile("nop");
  asm volatile("nop");
  asm volatile("nop");
  asm volatile("nop");
  asm volatile("nop");

#if GRAYSCALE_PIXELS == 1
```

```
  asm volatile("nop");
  asm volatile("nop");
  asm volatile("nop");
  asm volatile("nop");
  asm volatile("nop");
  asm volatile("nop");
#endif

}
```

4. Testing

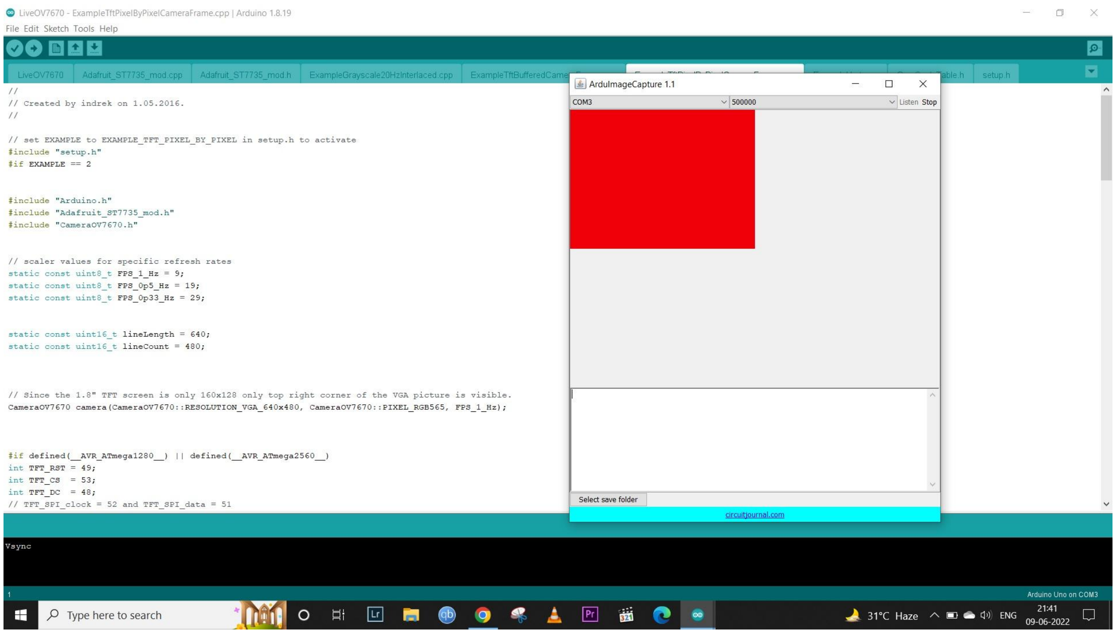
There are various checkpoints we can check while making connections in OV7670.

CheckPoint 1 :
Before doing any connections itself, after we add all the dependent libraries and plugin for display the window should start listening to the serial port automatically. If not, select the correct COM port and click "Listen."



```
Vsync
Frame 18
Vsync
Frame 19
Vsync
Frame 20
Vsync
Frame 21
Vsync
```

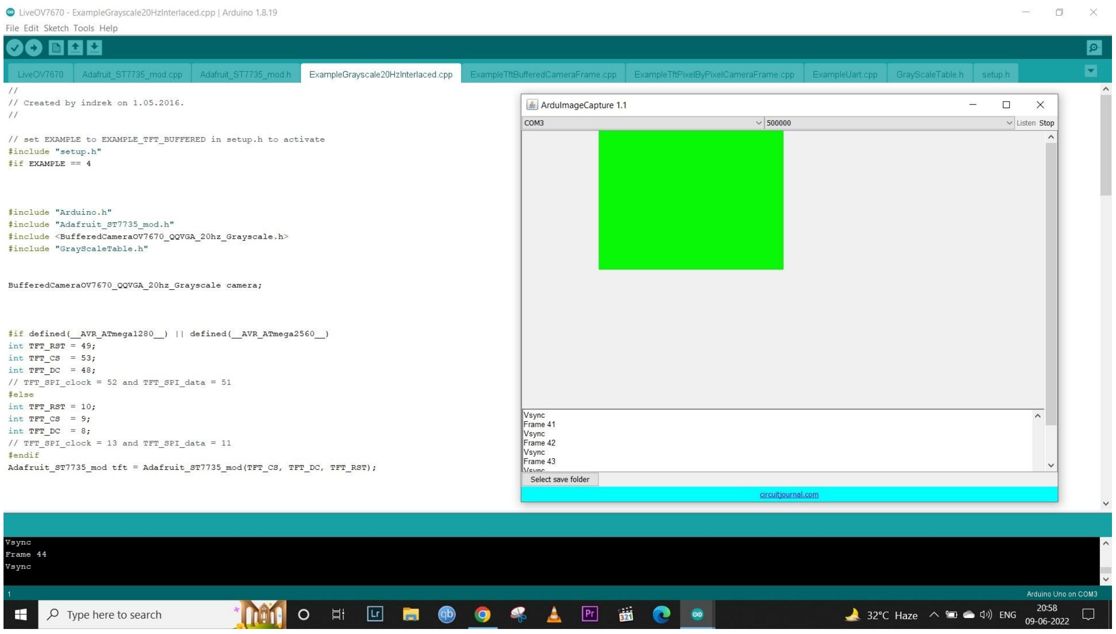
This white noise should turn to Red after a few minutes. This means that you have the correct code in your Arduino, but it couldn't detect the camera module. This is because we have not made the connections yet.



```
//
// Created by indrek on 1.05.2016.
//

// set EXAMPLE to EXAMPLE_TFT_PIXEL_BY_PIXEL in setup.h to activate
#include "setup.h"
#if EXAMPLE == 2


#include "Arduino.h"
#include "Adafruit_ST7735_mod.h"
#include "CameraOV7670.h"


// scaler values for specific refresh rates
static const uint8_t FPS_1_Hz = 9;
static const uint8_t FPS_0p5_Hz = 19;
static const uint8_t FPS_0p33_Hz = 29;


static const uint16_t lineLength = 640;
static const uint16_t lineCount = 480;



// Since the 1.8" TFT screen is only 160x128 only top right corner of the VGA picture is visible.
CameraOV7670 camera(CameraOV7670::RESOLUTION_VGA_640x480, CameraOV7670::PIXEL_RGB565, FPS_1_Hz);



#if defined(__AVR_ATmega1280__) || defined(__AVR_ATmega2560__)
int TFT_RST = 49;
int TFT_CS  = 53;
int TFT_DC  = 48;
// TFT_SPI_clock = 52 and TFT_SPI_data = 51
```

```
Vsync
```

checkPoint 2 :
After making all power and pin connections except the pixel data input pins, the display should now flash a green image. This means that the LiveOV7670 library was able to detect and configure the camera successfully.


LiveOV7670 - ExampleGrayscale20HzInterlaced.cpp | Arduino 1.8.19

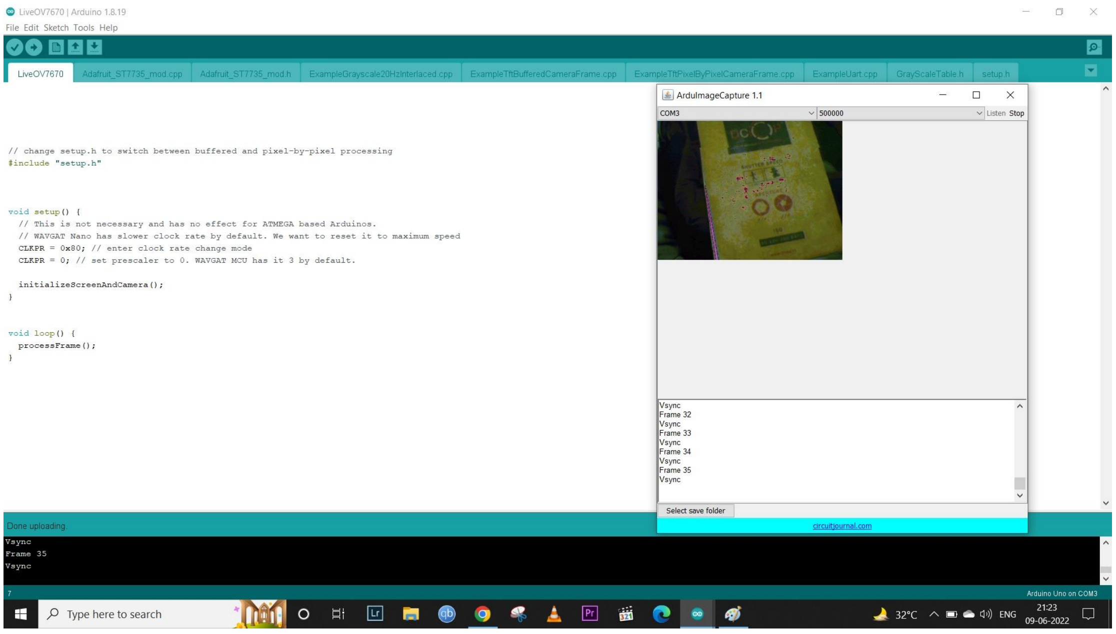
```
//
// Created by indrek on 1.05.2016.
//

// set EXAMPLE to EXAMPLE_TFT_BUFFERED in setup.h to activate
#include "setup.h"
#if EXAMPLE == 4



#include "Arduino.h"
#include "Adafruit_ST7735_mod.h"
#include <BufferedCameraOV7670_QQVGA_20hz_Grayscale.h>
#include "GrayScaleTable.h"


BufferedCameraOV7670_QQVGA_20hz_Grayscale camera;



#if defined(__AVR_ATmega1280__) || defined(__AVR_ATmega2560__)
int TFT_RST = 49;
int TFT_CS  = 53;
int TFT_DC  = 48;
// TFT_SPI_clock = 52 and TFT_SPI_data = 51
#else
int TFT_RST = 10;
int TFT_CS  = 9;
int TFT_DC  = 8;
// TFT_SPI_clock = 13 and TFT_SPI_data = 11
#endif
Adafruit_ST7735_mod tft = Adafruit_ST7735_mod(TFT_CS, TFT_DC, TFT_RST);
```

ArduImageCapture 1.1 — COM3 — 500000

```
Vsync
Frame 41
Vsync
Frame 42
Vsync
Frame 43
```

```
Vsync
Frame 44
Vsync
```

checkPoint 3 :
Now we can connect all the pixel data input pins. This will show the live capture of the image in real time.


LiveOV7670 | Arduino 1.8.19

```
// change setup.h to switch between buffered and pixel-by-pixel processing
#include "setup.h"



void setup() {
  // This is not necessary and has no effect for ATMEGA based Arduinos.
  // WAVGAT Nano has slower clock rate by default. We want to reset it to maximum speed
  CLKPR = 0x80; // enter clock rate change mode
  CLKPR = 0; // set prescaler to 0. WAVGAT MCU has it 3 by default.

  initializeScreenAndCamera();
}


void loop() {
  processFrame();
}
```

ArduImageCapture 1.1 — COM3 — 500000

```
Vsync
Frame 32
Vsync
Frame 33
Vsync
Frame 34
Vsync
Frame 35
Vsync
```

Done uploading.

```
Vsync
Frame 35
Vsync
```

5. Conclusion/ Results & Discussion

The entire project has been designed and developed as per the requirements of the user. It is found to be bug free as per the testing standards that are implemented.
The whole system's activities are divided into two major parts. For implementing the system, an FTDI module is used but it can also be implemented by Arduino. However, the IDE used is the Arduino IDE.
The system comprises following features:

- Motion Detection
- Live stream of security surveillance
- Alerting through internet
- Alerting through buzzer

There are also few features which can be integrated with this system to make it more flexible. The list below shows the future points to be considered:

- Making the camera move on its hinges/mount so it can cover a larger area of surveillance.
- We can also include facial recognition, so the door unlocks automatically for people with authorization.
- This can further be used in developing a home automation system that will perform certain tasks when the home owner returns home. Such as turning on the lights or opening the garage door.
- The project is not just limited to security. We can use this project to monitor anything. For example this can also be used as a baby monitor.

Finally, I would like to conclude that I have put all my efforts throughout the development of this project to the best of my ability and tried to fulfill most of the requirements of the user.

6. Advantages & Applications

Advantages :

This project will lower the number of theft cases, also possibly contributing to lowering the crime rate and towards a better and safer society. This project will also allow you to monitor your home with the ease of a smartphone, no matter where in the world you are.

Major advantage of this project is its costing. compared to all the security devices that are available in the market this project is being implemented at a cost that is much lower than those manufactured by large scale industries.

Any one can install this with ease while those available in the market need to be installed by a professional. The person using this project just needs to assemble and connect all the modules together . Then they can clone the project and burn it into the microcontroller through the Arduino IDE that's available for free.

Limitations :

The project is implemented on a smaller scale with a small range of surveillance. The camera is stationary and hence can’t use range of motion to monitor a wider area. The PIR sensor also only works when a person passes through within its range of surveillance, hence the installation of the sensor should be done with precision and care. The sensors, camera and wiring are not waterproof, As all the hardware components will be installed outside the house, the installation should be done keeping the weather conditions in mind like rain and winds.

Future Scope :

In future we can add range of motion to the camera to be able to cover a larger area under inspection. We can use servo motors and attach to it the OV7670 camera module that will be enclosed in a 3D printed casing. The casing containing the camera can then move in a left to right motion, resulting in a wider coverage of surveillance.

This project can be further expanded into a fully fledged home automation system. As the ESP board has wifi facility we can send alerts back to it over the internet. A few examples of home automation could be sending alerts to turn on the light. When we come back home we can simply send back an alert to ESP board through the internet to turn on the light when it detects our arrival through PIR. Or we can set up ESP to turn on the light everyday at a specific time. Further we can control many other things such as the house thermostat , fans , electrical appliances.

Another application could be we can use this as a low cost baby monitor. A sound sensor can keep monitoring the baby and when the baby cries or when it starts crawling the sensors will trigger the alert and you can monitor the baby through OV7670 and see the live stream of whether the baby is safe or not and take action accordingly.

References

[1] Doe, John. "Recommended practice for software requirements specifications (ieee)." IEEE, New York (2011).

[2] Pressman, Roger S. *Software engineering: a practitioner's approach*. Palgrave macmillan, 2005.

[3] Jalote, Pankaj. *A concise introduction to software engineering*. Springer Science & Business Media, 2008.

[4] Statista Database company, "India: number of reported robbery rate by state 2020", article by Sanyukta Kanwal, 2021.

[5] ThePrint, online newspaper, Delhi witnessed 15% rise in crime in 2021 as compared to 2020: Data

[6] Soni, Gaurav, et al. "Design and Implementation of Object Motion Detection Using Telegram." *2021 International Conference on Technological Advancements and Innovations (ICTAI)*. IEEE, 2021.