Answering Door Bell

Aditya Khemka
DBMS English School
[Date]

Contents:

1. Introduction and brainstorming:

1.1 Introduction and Background:

Calling Bells are common and used in almost every household in our society. As soon as the bell rings, we come to know that someone has arrived but very often it happens that we are busy with our house hold work or something else and due to some reason, cannot open the door immediately. This answering bell has a solution to this problem.

1.2 The problem

This problem came up when the respected Chairperson of our school Mrs. Bhanumati Neelkanthan spoke to us. She says that due to her age, she is unable to attend her guests at the door immediately.

1.3 Problem Statement

Difficulties in answering to guests at the door immediately

1.4 Ideation (after brain storming session)

Make a device to display a message to the guest at the door with the help of a remote

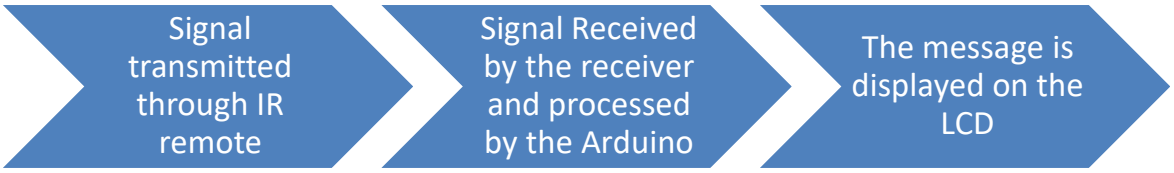
2. The First Prototype

2.1 Introduction:

A simple device with an LCD display (outdoor unit). This will contain some preloaded messages which may be displayed to the guest.
This display may be controlled with a remote.

2.2 Working :

Mostly all remotes work on IR technology. Whenever we press any button on the remote, it emits a IR with a particular frequency. This frequency is unique to all buttons on the remote and can be captured. We propose to use this feature to display different messages on different buttons.



(flowchart representing working of device)

2.3 Pre-requisite :

- Basic knowledge of electronic circuits
- Basic knowledge of Arduino programming Language
- Basic knowledge of working with Arduino

2.4 Hardware required :

- 3 pin IR Receiver (TSOP-848)
- Any IR remote (like the one's used in TVs, here used Sony RMSC1)
- LCD Display (16x2)
- Arduino Uno
- Jumper wires
- potetiometer
- RGB Leds or buzzer

2.5 Circuit / Connections :

IR Receiver

- Vcc to 5V pin of Arduino
- GND to Ground pin of Arduino
- OUT to the pin 6 of Arduino.

LED / Buzzer Connection

- LED Positive to Pin 7
- LED Ground to GND

LCD Connection

- Pin 1: GND to GND of Arduino
- Pin 2: 5V to 5V of Arduino
- Pin4: RS to Pin12 of Arduino
- Pin5: RW to GND of Arduino
- Pin6: EN to Pin11 of Arduino
- Pin7:
- Pin8:
- Pin9:
- Pin10:
- Pin11: D4 to Pin5 of Arduino
- Pin12: D5 to Pin4 of Arduino
- Pin13: D6 to pin3 of Arduino
- Pin14: D7 to Pin2 of Arduino
- Pin15: VCC to 10 of Arduino
- Pin16: GND to GND of Arduino

2.6 Installing required Softwares :

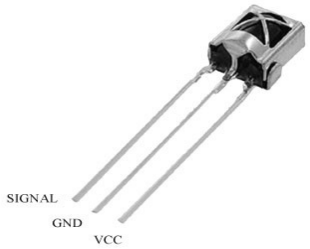
Before working, we must install some additional libraries into our Arduino IDE . To do so :

1. First of all the Arduino IDE – it will enable us to write and edit the code . Download the IDE from here and install it on your computer .
2. Now we need additional libraries to process IR codes. Download the 'IR Library' zip file and 'IR Library Master' zip file from this link . (Check guide)
3. On top the IDE , go to sketch > include library > Add .zip library and select the downloaded zip files.

2.7 Working Logic :

Each button on a remote has its own unique hex value. Whenever we press a button , the remote emits hex values linked that particular button. We will capture these hex values and then use it as per our needs. To do so:

1. On the Top left, go to files > examples . Scroll down to find IR remote and select Remote decode from the drop down menu
2. Take a IR sensor and hook it with the Arduino . In this case we're using the TSOP-1838 sensor
    - left leg (signal) : pin 12 of Arduino
    - Middle leg (ground) Ground of Arduino
    - Right leg (VCC) 5V of Arduino



3. Upload the code and open Serial Monitor. Whenever you press any button on the remote , a code (or number) would show up on screen . That's the code for that button.

2.8 Source Code :

```
////(install additional libraries)
#include <IRremote.h>
#include <LiquidCrystal.h>
LiquidCrystalLCD(12,11,5,4,3,2);
int RECV_PIN = 6;
IRrecvirrecv(RECV_PIN);
decode_results results;
int LED=7;
#define BUTTON_1 0xC41
#define BUTTON_2 0xA41
#define BUTTON_3 0xF01
#define BUTTON_4 0xA81
#define LCD_LIGHT_PIN 10
```

```
void setup() {
pinMode (LED, OUTPUT);
irrecv.enableIRIn();
pinMode(4,OUTPUT);
pinMode(LCD_LIGHT_PIN, OUTPUT);
digitalWrite(LCD_LIGHT_PIN, LOW);

 }

void loop() {
if (irrecv.decode(&results))
 {
if (results.value == BUTTON_1)
 {
digitalWrite(LCD_LIGHT_PIN, HIGH);
digitalWrite(LED,HIGH);
LCD.begin(16,2);
LCD.clear();
LCD.setCursor(5,0);
LCD.print("PLEASE");
LCD.setCursor(1,1);
LCD.print("WAIT FOR 5 MIN");
   }
irrecv.resume();
if (results.value == BUTTON_2)
  {
digitalWrite(LCD_LIGHT_PIN, HIGH);
digitalWrite(LED,HIGH);
LCD.begin(16,2);
LCD.clear();
LCD.setCursor(2,0);
LCD.print("WE WILL MEET");
LCD.setCursor(5,5);
LCD.print("LATER");
  }
irrecv.resume();
if (results.value == BUTTON_3)
  {
```

```
digitalWrite(LCD_LIGHT_PIN, HIGH);
digitalWrite(LED,HIGH);
LCD.begin(16,2);
LCD.clear();
LCD.setCursor(1,0);
LCD.print("COME TOMORROW");
   }
irrecv.resume();
if (results.value == BUTTON_4)
  {
digitalWrite(LCD_LIGHT_PIN, LOW);
digitalWrite(LED,LOW);
   }
}

}
```

2.9 Prototype and working :

(prototype V1.1)

Video: https://www.youtube.com/watch?v=erJRWuSbBtY

2.10 Testing and usage :

The device was tested and was installed (after fubrishing) at Mrs Neelkanthan's house. The device worked perfectly but lacked some features which were rectified in the upcoming prototypes.

2.11 Improvments :

After usage, mam pointed out the following possible improvements.

- Output was limited to only three messages
- Output was only in English, which may not be readable to some users
- Was a wired device , so would not work if electricity was cut-off
- Unless knowing who's the guest, it is difficult to display an appropriate message

3. The Second Prototype

3.1 Introduction:

The second prototype is very similar to the first prototype in terms of working. It mainly eliminated the first three problems faced.

3.2 Difference from the first prototype:

The second prototype is slightly different from the first prototype in terms of hardware and software but showcases the following features absent from the first prototype:

- Can display name and address of host
- Can work on a battery
- Can display a dozen different messages
- Can display messages in both Hindi and English

3.3 Hardware Required:

- IR Receiver (here used 1838T)
- An old Remote (Here using Sony RMT-TX111P)
- Arduino UNO
- LCD display(20 x 2)
- A few Jumper Wires
- An RGB LED or a buzzer
- Potentiometer

3.4 Connections / Circuit:

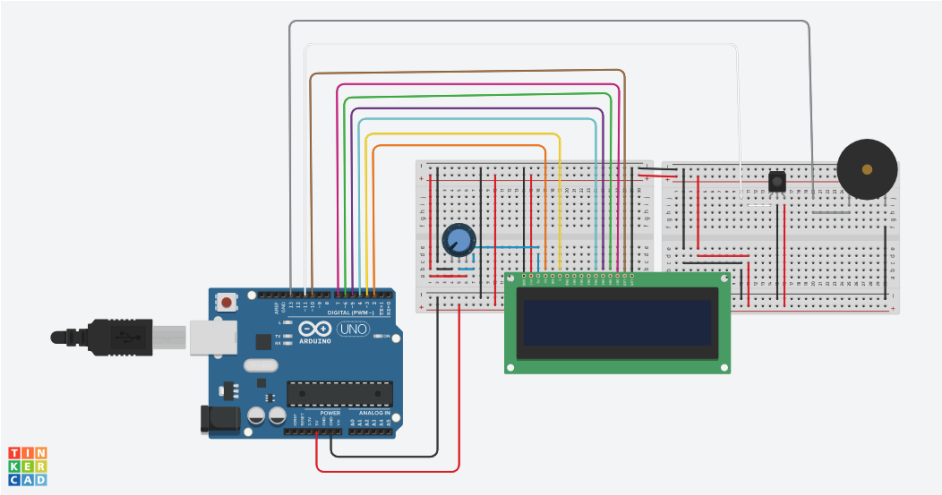
<u>IR Receiver</u>

- Vcc to 5V pin of Arduino
- GND to Ground pin of Arduino
- OUT to the pin 11 of Arduino.

<u>LCD Connection</u>

- Pin 1: VSS to GND of Arduino
- Pin 2: VDD to 5V of Arduino
- Pin 3: VO to potentiometer (centre pin)
- Pin4: RS to Pin2 of Arduino

- Pin5: RW to GND of Arduino
- Pin6: EN to Pin3 of Arduino
- Pin7: [Not used]
- Pin8: [Not used]
- Pin9: [Not used]
- Pin10: [Not used]
- Pin11: D4 to Pin4 of Arduino
- Pin12: D5 to Pin5 of Arduino
- Pin13: D6 to pin6 of Arduino
- Pin14: D7 to Pin7 of Arduino
- Pin15: A to 10 of Arduino
- Pin16: K to GND of Arduino



Visual circuit of the first prototype

3.5 Source Code :

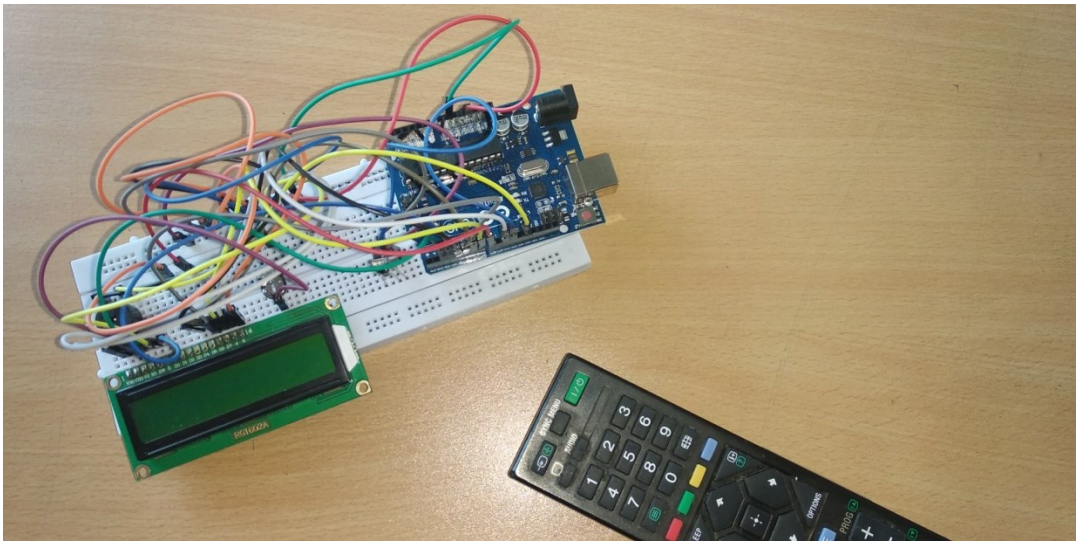
```
///(this is the source code of device displayed in the following video.
///Not the latest one …)

//initialization
#include <IRremote.h>
#include <IRremoteInt.h>
#include <LiquidCrystal.h>
int val;
const int bl=10 , IRpin=12 ;
boolean ir_val ;
IRrecv irrecv(IRpin);
decode_results results;
const int rs=2, e=3, d4=4, d5=5, d6=6, d7=7;
LiquidCrystal lcd(rs,e,d4,d5,d6,d7);

//setup
void setup() {
 lcd.begin(16,2);
 lcd.clear();
 Serial.begin(9600);
 pinMode(bl,OUTPUT);
 irrecv.enableIRIn(); // Start the receiver
 lcd.clear();
 message("D.B.M.S. English","    School    ");
}

//working
void loop() {
 delay(250);
 ir_val = irrecv.decode(&results);
  if (ir_val == true){   //checks if signal is transmitted
    delay(250);
    val =results.value ;
    Serial.println(val);
     // prints the value of the hex code on the serial monitor
```

```
switch (val)
{

  case 2704: //power
  lcd.clear();
  digitalWrite(bl, LOW);
  lcd.setCursor(0,0);
  lcd.print("              ");
  lcd.setCursor(0,1);
  lcd.print("              ");
  break;

  case 112: // home
  message ("   B.H. Area    " , "   Road No. 7   " );
  break;

  case 2672: // ok
  message (" Aditya  Khemka " , " Aniket  Sarkar " );
  break;

  case 2320: //0
  message ("D.B.M.S. English" , "     School     ");
  break;

  case 16: //1
  message ("Please wait for " , "    a minute    ");
  break;

  case 2064: //2
  message ("     Please     " , "    Come  in    ");
  break;

  case 1040: //3
  message ("     Just     " , "     coming     ");
  break;

  case 528: //4
  message ("Please wait for " , "    5 minutes   ");
```

```
      break;

      case 3088: //5
      message (" Please drop it " , " in the mailbox ");
      break;

      case 2576: //6
      message ("   Can we meet  " , "     Later?     ");
      break;

      case 1552: //7
      message ("     Please     " , "    Come  in    ");
      break;

      case 3600: //8
      message (" Please drop it " , " in the mailbox ");
      break;

      case 272: //9
      message ("     Just      " , "     coming      ");
      break;

    }
    delay(250);
    irrecv.resume();
    delay(250);

     //print message on L.C.D. according to the value received
   }

}//void loop

void message (String one , String two )
{
  lcd.clear();
  delay(250);
  digitalWrite(bl,HIGH);
  lcd.setCursor(0,0);
```

```
  lcd.print(one);
  lcd.setCursor(0,1);
  lcd.print(two);
  delay(250);

}
```

3.6 Prototype and Working :



(The second prototype)

The second prototype worked perfectly and was able to eliminate the first two objectives, but to add a camera was still a problem, which was fixed in the third prototype.

Video link: https://www.youtube.com/watch?v=Oe4iaxCR6WY

4. The Third Prototype

4.1 Introduction:

The third prototype adds a camera to the second prototype and an app to stream the camera. The third prototype also enables to unlock the door with the app .

4.2 Difference from the previous prototypes:

Prototype 1 and Prototype 2 were only directive but not interactive. Also, it only had a remote but this may be controlled with an app…

4.3 Hardware required :

In addition to the second prototype, the following hardware may be required

- ESP32-CAM board
- Jumper Wires
- FTDI programmer to upload code (here used Arduino Uno instead)
- Solenoid door lock

4.4 Additional software required :

To prepare the app for the third prototype, we have used the Blynk IOT platform. Blynk is a platform which helps us to interact with specially enabled hardware. In this project, we'll be using esp-32 board which is wifi enabled. So, using blynk, we'll be able to communicate with the app and hardware through wifi.

(Esp32 cam pinout diagram)

4.5 Circuit :

- Esp-32 cam GND to Arduino GND
- Esp-32 cam 5v to Arduino 5v
- Esp-32 cam U0R to Arduino TR
- Esp-32 cam U0T to Arduino TX
- Esp-32 cam GPIO-0 to Esp-32 cam GND
- Arduino reset to Arduino GND
- Push Button

4.6 Preparing the app :

1. Install the Blynk (legacy) software from Google Play Store / Apple App Store and sign up
2. Click on the new project button and name the project. Save the Auth Token (also sent to the registered email ID)
3. Add the image gallery widget and configure to work with virtual pin V1
4. Add notifications widget and configure

4.7 Source Code :

First, we must install the ESP-32 libraries and boards. To do so,

1. go to File > Prefrences.
2. In the bottom corner of the screen, there's a tab for "additional boards manager URLs". In the textbox, paste the following code: https://dl.espressif.com/dl/package_esp32_index.json, http://arduino.esp8266.com/stable/package_esp8266com_index.json
3. click OK
4. Now go to tools (in the upper bar menu) > Board > Boards manager
5. Search esp32. Select and install the board esp32 by espressif systems.
6. Go to sketch > include library > manage libraries . Search for Blynk and install the latest library .

Go to files > examples > esp32 > esp32cam > esp32cam webserver. Edit the code with the below code . Do not make any changes in the other three tabs .

```
#include "esp_camera.h"
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>
//
// WARNING!!! Make sure that you have either selected ESP32 Wrover
Module,
//          or another board which has PSRAM enabled
//

// Select camera model
//#define CAMERA_MODEL_WROVER_KIT
//#define CAMERA_MODEL_ESP_EYE
//#define CAMERA_MODEL_M5STACK_PSRAM
//#define CAMERA_MODEL_M5STACK_WIDE
#define CAMERA_MODEL_AI_THINKER

#include "camera_pins.h"
#define LED 21
#define BUTTON 15

const char* ssid = "SSID";
const char* password = "PASS";
char auth[] = "AUTH_TOKEN";

String my_Local_IP;

void startCameraServer();


void capture()
{
  digitalWrite(LED,HIGH);
  uint32_t number = random(40000000);
  Blynk.notify("Someone is at the door..");
  Serial.println("http://"+my_Local_IP+"/capture?_cb="+
(String)number);
```

```
  Blynk.setProperty(V1, "urls",
"http://"+my_Local_IP+"/capture?_cb="+(String)number);
  delay(1000);
  digitalWrite(LED,LOW);

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

#if defined(CAMERA_MODEL_ESP_EYE)
  pinMode(13, INPUT_PULLUP);
  pinMode(14, INPUT_PULLUP);
#endif

  // 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 blightness 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);

#if defined(CAMERA_MODEL_M5STACK_WIDE)
  s->set_vflip(s, 1);
  s->set_hmirror(s, 1);
#endif

  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());
 my_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(BUTTON) == LOW)
 capture();
}
```

Replace ‘SSID’ with wifi name, ‘PASS’ with wifi password and ‘AUTH_TOKEN’ with the auth key sent by blynk to the registered ID

4.8 Uploading Procedure :

Unlike the previous protypes, the code cannot be directly uploaded onto the board.

1. Connect the Arduino according to the above circuit and to the computer
2. Upload code by changing the following settings:
    - Install esp32 cam libraries and board managers
    - Select Board as ESP wrover module
    - QIO as falsh mode
    - Partition scheme: huge app
    - Flash frequency: 40MHz
    - Upload speed: 115200

3. After the code is successfully uploaded, disconnect the following pins (that were used for shorting)
    - Esp32 cam GPIO-0 and esp32 cam GND
    - Arduino reset and Arduino GND

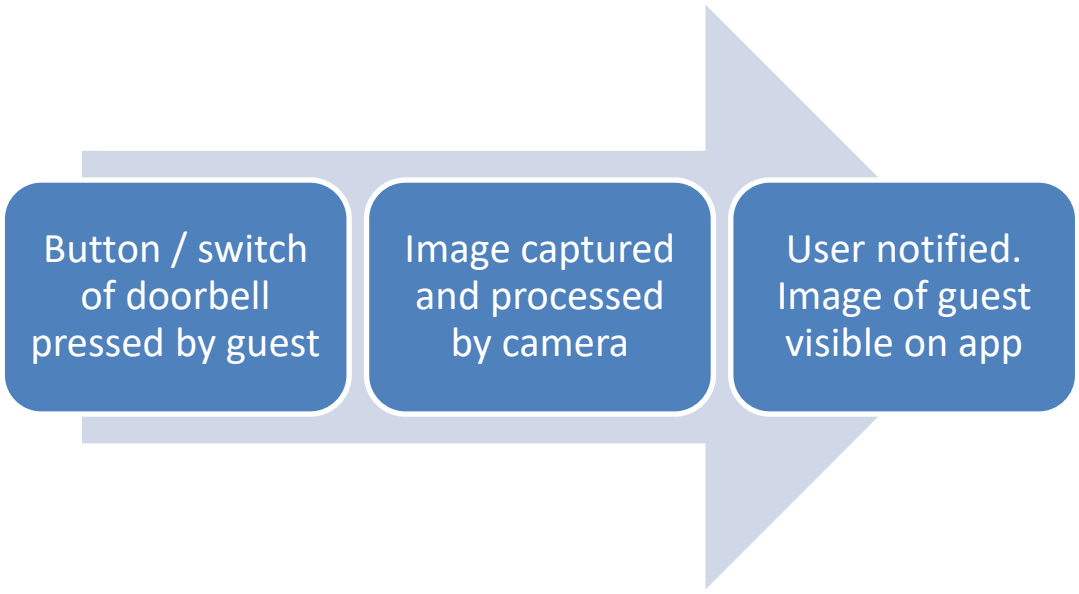
Also reset the esp32cam board by pressing the reset button for 2-5 seconds.

Troubleshooting and upload Guide (references):

- https://randomnerdtutorials.com/program-upload-code-esp32-cam/
- https://randomnerdtutorials.com/esp32-cam-troubleshootingguide/#:~:text=Important%3A%20if%20you%20can't,jumper%20cap%20set%20to%205V.
- https://create.arduino.cc/projecthub/noah_arduino/using-esp32-cam-with-arduino-b4f12c
- https://www.youtube.com/watch?v=U7qbehy9aDo

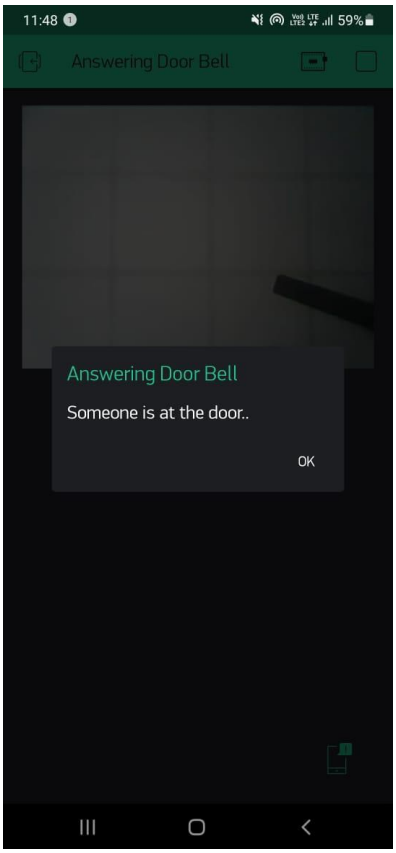
4.9 Working

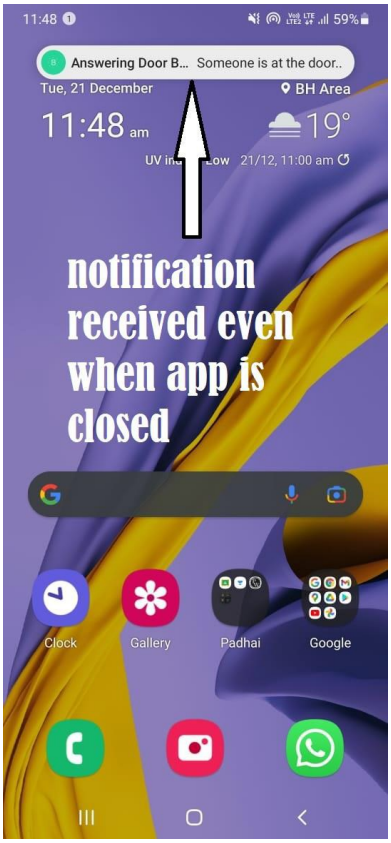
Whenever someone presses the doorbell, an image of the guest is captured This image is sent over to the app and the user is notified as well

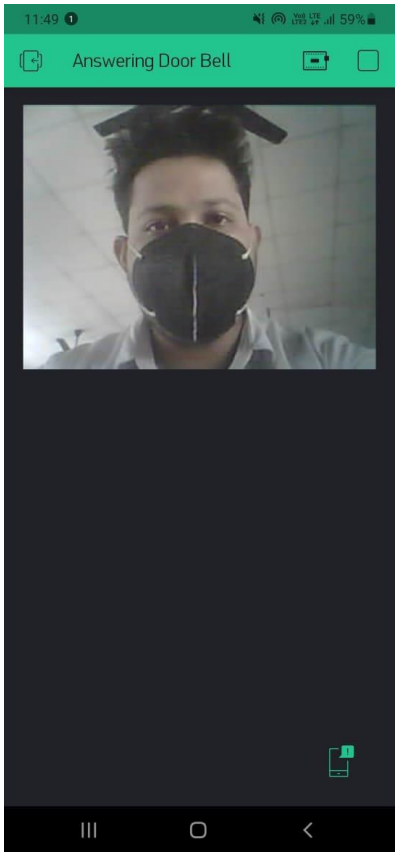


(Block diagram representation of the working of prototype)

4.10 Screenshot:







1. Notification received when user presses the doorbell
2. App works even when it is closed
3. Image captured by camera

4.11 Demonstration

A short demonstration of prototype 3.1 can be found here :

Some important links:

Google Drive (all resources available here): https://bit.ly/3zrMzzf

Powerpoint presentation: https://bit.ly/334oJNK

Detailed documentation: https://bit.ly/3ERiR7S

Video(s): youtu.be/y8ASDURxpCY , youtu.be/Oe4iaxCR6WY , https://youtu.be/aAY93_xtBoM , https://youtu.be/Xu5evn3pz6I