ENGINEERING CLINICS

GROUP NUMBER – 230577

REVIEW – III

Faculty Coordinator

S Bhaskar Nikhil (20BCI7175)

Asutosh Doppalapudi (20BCI7178)

Venkatesh Panyam (20BCI7062)

Tiruveedhula Chandra Mouli (20BEV7029)

Karnati Naga Prashanth Reddy (20BES7056)

Niranjan S Nair (20BCI7037)

Web/App Controlled Surveillance Robot

Problem Statement

Traditional surveillance systems are limited by their fixed camera positions and its inability to move, which can result in blind spots. One of their main drawbacks is that they are unable to effectively monitor changing situations. To address these limitations, there is a need for a web or app-controlled surveillance robot that can autonomously navigate and monitor a given area, while providing real- time video feedback to the user. Such a robot would allow for greater flexibility and coverage, as well as increased efficiency and accuracy in monitoring tasks. However, there are several key technical and practical challenges that must be addressed in developing such a system. These include ensuring the robot's mobility and navigational abilities in a variety of indoor and outdoor environments, developing robust and reliable communication between the robot and the user, ensuring the security and privacy of the data collected and transmitted, and providing an intuitive and user- friendly interface for controlling and interacting with the robot. Therefore, the problem statement for this project is to design and develop a web or app-controlled surveillance robot that can autonomously navigate and monitor a given area, while addressing the technical and practical challenges associated with such a system.

Required Components

Components name	Quantity
Raspberry Pi Pico	1
Motor Driver, L298	1
DC Motor, 60 rpm	2
Plane Holder	1
18650 Rechargable Li-ion batteries	2
ESP 32 Camera	1
Battery Holder	1
Power Switch	1
Jumper Wires	15 - 20

About Components

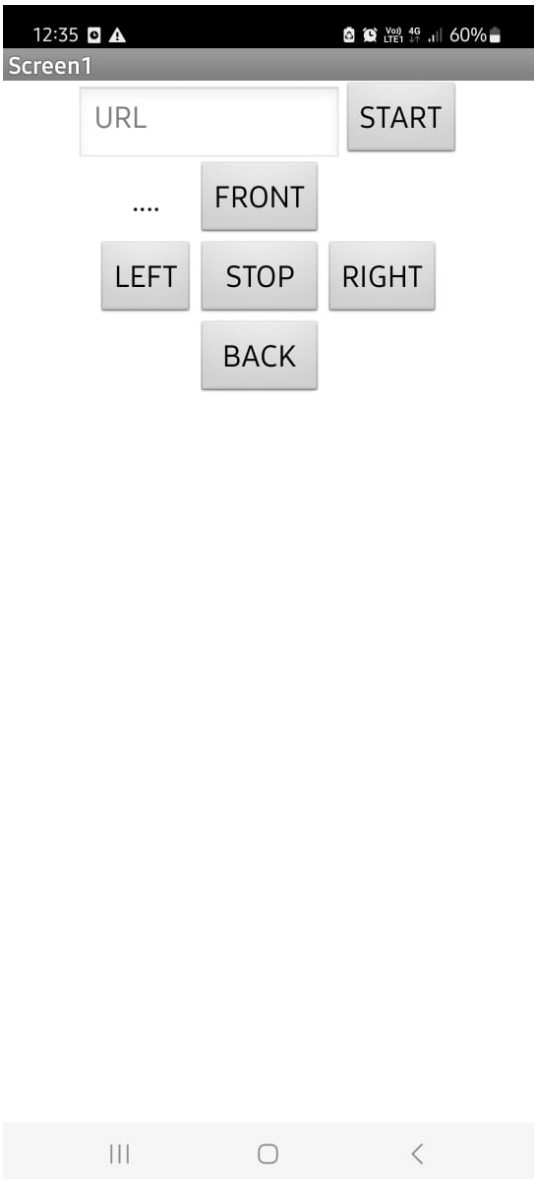
1) **Raspberry Pi Pico**
   Raspberry Pi Pico is a microcontroller board based on the RP2040 microcontroller chip designed by the Raspberry Pi Foundation. It can be programmed using C, C++ or MicroPython. The microcontroller has 264KB of SRAM, 2MB of flash memory, 26 multi-functional GPIO pins, Hardware PWM, SPI, I2C, UART and PIO support. It has an operating voltage of 3.3V with dual core ARM Cortex – M0+ processors.

2) **Motor Driver, L298**
   The L298 is a popular motor driver IC that is commonly used in robotics and other projects that involve controlling DC motors or stepper motors. It is a dual H-bridge driver that can drive two DC motors or one stepper motor, and it is capable of handling currents up to 2A per channel.

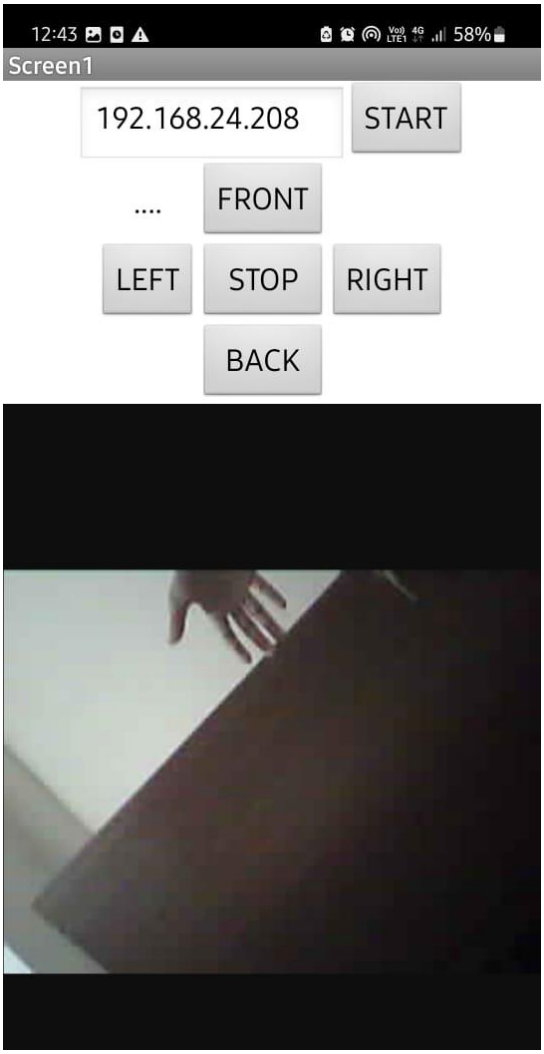
3) **ESP 32 Camera**
ESP32 Camera is a compact and low-cost camera module designed for IoT applications. It is based on the ESP32-S2 system-on-chip, which includes a powerful dual-core processor and built-in Wi-Fi and Bluetooth connectivity. The camera module features a OV2640 camera sensor with a resolution of 2 megapixels, and it can capture JPEG images and H.264-encoded video streams.

APP Interface

1) Before Connecting to ESP32 Camera



2) After Connection is established



Code

1) *Code for ESP32 Camera in Arduino IDE*

```
#include <WebServer.h>
#include <WiFi.h>
#include <esp32cam.h>
#include "DHT.h"
#define DHTPIN 14
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

const char* WIFI_SSID = "RAAS IPHONE";
const char* WIFI_PASS = "09346575127";

WebServer server(80);

int buz=12;
```

```
int bt=13;
int rcv;
static auto loRes = esp32cam::Resolution::find(320, 240);
static auto midRes = esp32cam::Resolution::find(350, 530);
static auto hiRes = esp32cam::Resolution::find(800, 600);

int tmp=30;
float get_distance_cm()
{
 // Connect the arduino to the SR04 pins as defined here
 //  (you can change A4 and A5 to any other unused pins for GPIO mode)
 const int SR04_Trig_RX_SCL = 15;
 const int SR04_Echo_TX_SDA = 14;

 // Ensure the pins are in the correct mode
 pinMode(SR04_Echo_TX_SDA,INPUT);
 pinMode(SR04_Trig_RX_SCL,OUTPUT);

 // We need to delay by at least 30ms between readings
 //  so we don't get a false reading
 static unsigned long lastread = 0;
 if(millis() - lastread < 30)
 {
  delay(millis()-lastread);
 }
 lastread = millis();

 // Send a trigger pulse
 digitalWrite(SR04_Trig_RX_SCL,HIGH);
 delayMicroseconds(500);
 digitalWrite(SR04_Trig_RX_SCL,LOW);

 // Read the distance pulse,
 // the number of microseconds for a return trip at the speed of sound
 unsigned long microseconds  = pulseIn(SR04_Echo_TX_SDA,HIGH);

 // Sound travels in air at about 343 m/s at 20degrees C
 //  that is 343 / 1000000 m per microsecond
 //  that is ( 343 / 1000000  * 100) cm per microsecond
 // Note thoat we need to "cast" to a float here to calculate correctly
 float distance  = microseconds * ( (float) 343 / 1000000 * 100);
 distance  = distance/2; // divide by 2 since that was a return trip

 return distance;
}
void serveJpg()
{
 auto frame = esp32cam::capture();
 if (frame == nullptr) {
  //Serial.println("CAPTURE FAIL");
  server.send(503, "", "");
```

```
    return;
  }
//Serial.printf("CAPTURE OK %dx%d %db\n", frame->getWidth(), frame->getHeight(),
//               static_cast<int>(frame->size()));

  server.setContentLength(frame->size());
  server.send(200, "image/jpeg");
  WiFiClient client = server.client();
  frame->writeTo(client);
}

void handleRoot() {
  if(Serial.available())
  {
    rcv=Serial.read();

  }

  String json = "{\"C\":" +String(rcv)+ "}";
  rcv=0;

  server.send(200, "application/json", json);

}

void handleJpgLo()
{
  if (!esp32cam::Camera.changeResolution(loRes)) {
   // Serial.println("SET-LO-RES FAIL");
  }
  serveJpg();
}

void handle_1()
{
   Serial.print("1");
}

void handle_2()
{

   Serial.print("2");

}

void handle_3()
{
   Serial.print("3");
}
```

```
void handle_4()
{

   Serial.print("4");

}

void handle_5()
{
   Serial.print("5");
}


void handle_6()
{

   Serial.print("6");

}
void handleJpgHi()
{
 if (!esp32cam::Camera.changeResolution(hiRes)) {
   //Serial.println("SET-HI-RES FAIL");
 }
 serveJpg();
}

void handleJpgMid()
{
 if (!esp32cam::Camera.changeResolution(midRes)) {
  // Serial.println("SET-MID-RES FAIL");
 }
 serveJpg();
}

void  setup(){
 Serial.begin(9600);
 Serial.println();
 {
   using namespace esp32cam;
   Config cfg;
   cfg.setPins(pins::AiThinker);
   cfg.setResolution(hiRes);
   cfg.setBufferCount(2);
   cfg.setJpeg(80);
dht.begin();
   bool ok = Camera.begin(cfg);
 //  Serial.println(ok ? "CAMERA OK" : "CAMERA FAIL");
 }
 WiFi.persistent(false);
 WiFi.mode(WIFI_STA);
```

```
 WiFi.begin(WIFI_SSID, WIFI_PASS);
 while (WiFi.status() != WL_CONNECTED) {
   delay(500);
 }
 Serial.print("http://");
 Serial.println(WiFi.localIP());
 //Serial.println("  /cam-lo.jpg");
 //Serial.println("  /cam-hi.jpg");
 //Serial.println("  /cam-mid.jpg");
 server.on("/1", handle_1);
 server.on("/2", handle_2);
 server.on("/3", handle_3);
 server.on("/4", handle_4);
 server.on("/5", handle_5);
 server.on("/6", handle_6);
 server.on("/cam-lo.jpg", handleJpgLo);
 server.on("/cam-hi.jpg", handleJpgHi);
 server.on("/cam-mid.jpg", handleJpgMid);
 server.on("/", handleRoot);

 pinMode(bt,INPUT);
 pinMode(buz,OUTPUT);

  server.begin(); }

void loop()
{
 server.handleClient();
}
```

2) *Code for DC*

```
# Example showing how functions, that accept tuples of RGB values,
# Simplify working with gradients
from machine import Pin, UART, PWM, ADC
import time,utime

rcv = UART(1, baudrate=9600, tx=Pin(4), rx=Pin(5))
buff = bytearray(255)
TIMEOUT = False

m1=Pin(6,Pin.OUT)
m2=Pin(7,Pin.OUT)
m3=Pin(8,Pin.OUT)
m4=Pin(9,Pin.OUT)

while(1):
        time.sleep(0.1)
        x=rcv.read()

        if x is not None:
           x=(x.decode()).strip()
```

```
#x=input('ENTER:')
print(x)
if(x=='1'):
    print('Front')
    m1.value(1)
    m2.value(0)
    m3.value(1)
    m4.value(0)

if(x=='2'):
    print('BACK')
    m1.value(0)
    m2.value(1)
    m3.value(0)
    m4.value(1)

if(x=='3'):
    print('LEFT')
    m1.value(1)
    m2.value(0)
    m3.value(0)
    m4.value(1)

if(x=='4'):
    print('RIGHT')
    m1.value(0)
    m2.value(1)
    m3.value(1)
    m4.value(0)

if(x=='5'):
    print('STOP')
    m1.value(0)
    m2.value(0)
    m3.value(0)
    m4.value(0)
```

Image of the Prototype