BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY

Department of Electrical and Electronic Engineering

Course: Control System Laboratory (EEE 318)

Project Report

Cable Suspended Spidercam with Three-Dimensional Movement

Submitted By: Touhidul Islam

Section: C2

Group: 01

1806170-Saheeb Ahmad

1806177-Touhidul Islam

1806184-Arnob Ghosh

1806185-Adib Md. Tawsif

Summary:

The Spidercam is a cable-suspended camera system which enables film and television cameras to move both vertically and horizontally over a predetermined area, typically the playing field of a sporting event such as a cricket pitch, football field or a tennis court. The Spidercam operates with four motorized winches positioned at each corner at the base of the covered area, each of which controls a cable connected to a camera-carrier.

Objective:

The basic objective of this project is to make a camera suspended by 4 wires which will be controlled by servo motors that will allow the camera to have three-dimensional movement. Three potentiometers were used to give x,y,z co-ordinate updates and four potentiometers were also used for the individual control of each servo motor for the benefit of manual adjustments. Primarily, the three- dimensional movement of the camera was obtained by adjusting the lengths of the wires. ESP32 camera module was used as the camera. Using ESP32 captured video can also be streamed to destination i.e. we can get livestream feedback.

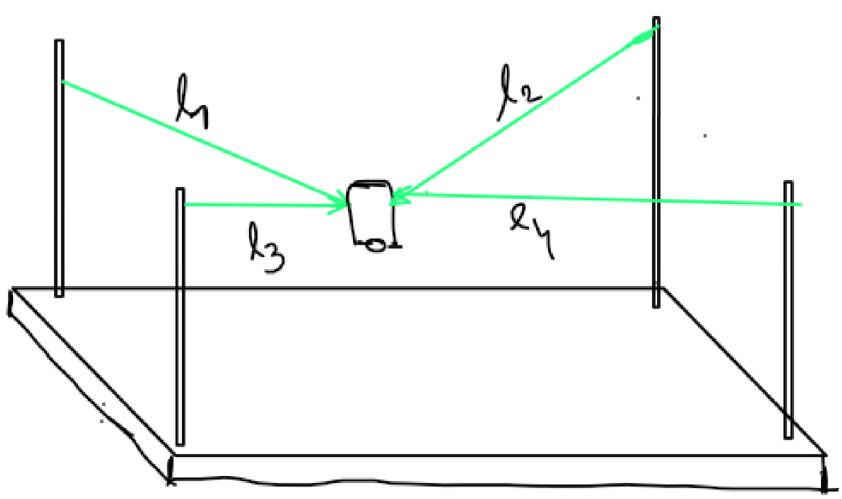


Figure 1: Basic Design

Methodology:

Four servo motors are the center piece of the whole setup. Servo motors allow precise control of the motor shaft rotation angle. 3 potentiometers were used to take input of the current desired coordinate. The 3 potentiometers' value correspond to the x, y, z coordinate.

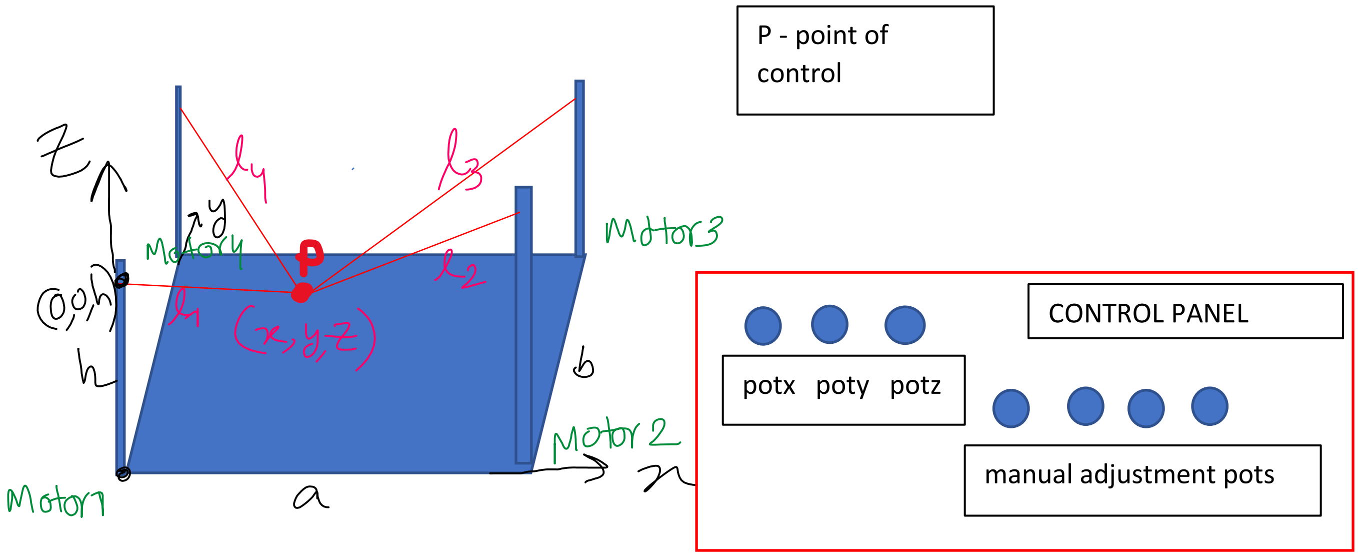


Figure 2: The setup

A coordinate system was placed as shown in figure 2. Here,
a = length of the field
b = width of the field
h = height of the field

Initial setup:

Initially the point of control was positioned at coordinate (0,0,h). Then input of required coordinate would be taken from the three potentiometers.

Taking input and reacting to input:

If the required coordinate of P is (x,y,z) then, the required length of the 4 wires are calculated as:

$$l1 = \sqrt{x^2 + y^2 + (h-z)^2}$$
$$l2 = \sqrt{(a-x)^2 + y^2 + (h-z)^2}$$

$$l_3 = \sqrt{(a-x)^2 + (b-y)^2 + (h-z)^2}$$
$$l_4 = \sqrt{x^2 + (b-y)^2 + (h-z)^2}$$

After getting an input of a new required coordinate, the required length of the wires would be subtracted from the current length of the wire and 4 values would be calculated:

$$\Delta l_1 = l_{1required} - l_{1current}$$
$$\Delta l_2 = l_{2required} - l_{2current}$$
$$\Delta l_3 = l_{3required} - l_{3current}$$
$$\Delta l_4 = l_{4required} - l_{4current}$$

These are the required change in wire length that we have to create to move point P at required coordinate. To do that, we would have to move the stepper motors with number of steps as per the following formulae:

$$stepper1Steps = \frac{1024 \cdot \Delta l_1}{\pi \cdot r}$$

$$stepper2Steps = \frac{1024 \cdot \Delta l_2}{\pi \cdot r}$$

$$stepper3Steps = \frac{1024 \cdot \Delta l_3}{\pi \cdot r}$$

$$stepper4Steps = \frac{1024 \cdot \Delta l_4}{\pi \cdot r}$$

Here, $r = radius\ of\ the\ shaft\ of\ the\ motor$

The steppers should be rotated accordingly and the P would reach the required coordinate.

A very important detail here is that, the wires should be wound on the motor shaft so that clockwise rotation of the motor shaft expands the wire and anticlockwise rotation cause contraction of wire.

One problem is that Arduino cannot run 4 stepper motors concurrently. For concurrent movement of the 4 steppers, we periodically rotated each motor 5 steps at a time (stepper 1 rotates 5 steps, then stepper 2 rotates 5 steps, then stepper 3, 4, 1 and so on), so that it practically acts as concurrent movement.

Accounting for error:

Potentiometers often give random noise output even when there is no real input. To account for that, an error margin was set, so that the input mechanism is unaffected by noise.

The output coordinate might not be at precisely at required coordinate for various reasons like environmental disturbance, unprecise motor movement etc. To account for that 4 manual adjustment potentiometers were mounted on the side panel of the setup. The manual adjustment potentiometers can be used to rotate the motor clockwise or anticlockwise to manually adjust the 4 wires' length. When manual adjustment of any motor is active, the system would not be taking coordinate input from x, y, z potentiometers.

Camera:

ESP32 camera module was used as camera and streaming module.

A asynchronous server was created in the ESP32 module. ESP32 also creates a local Wi-Fi network. The server is hosted as a local server accessible from the created Wi-Fi network. An asynchronous server was required as we needed a continuous camera output stream.

A webpage hosted in the IP address of the local server showed the live output stream of the camera, streaming the field output wirelessly to phone, tablet or PC.

Code:

Here are the snippets of the code used in controlling the servo motors:

For servo motor 1 & 2:

```
#include <Stepper.h>

//Setting up the stepper
Stepper stepper1 = Stepper(2048, 8, 10, 9, 11);
Stepper stepper2 = Stepper(2048, 2, 4, 3, 5);

//int step_count = 0;
int flag = 1;
int e = 1; //error threshold for pot inputs
int step_size = 5;

int potx; //pot1
int poty; //pot2
int potz; //pot3

int adjpot1;
int adjpot2;

int potx_prev = 0; int poty_prev = 0; int potz_prev = 0;

int del_potx, del_poty, del_potz; //////////////// may be needed in future

bool potx_changed = 0;
bool poty_changed = 0;
bool potz_changed = 0;

// field parameters
float r = 1 ; // r = radius of stepper head in cm
float a = 40; // cm
float b = 30.4; // cm
float h = 34; //cm

float l1_prev = 0;
float l2_prev = a;
float l3_prev = sqrt(a*a + b*b);
float l4_prev = b;

float x_prev = 0; float y_prev = 0; float z_prev = h;
```

```
float x_curr, y_curr, z_curr;
float del_x, del_y, del_z;
float l1_curr, l2_curr, l3_curr, l4_curr;
float del_l1, del_l2, del_l3, del_l4;
int st1_steps, st2_steps, st3_steps, st4_steps; // only stepper 1 and 2 in this arduino

void setup() {

  //Set the RPM of the stepper motor
  stepper1.setSpeed(5);
  stepper2.setSpeed(5);

  Serial.begin(9600);
}

void loop() {
    potx = analogRead(0);
    poty = analogRead(1);
    potz = analogRead(2);

    //adjustment pot
    adjpot1 = analogRead(4);
    adjpot2 = analogRead(5);

    //adjpot1
    if(((adjpot1 >= 0) && (adjpot1 <=100) )|| ((adjpot1 <=1023) && (adjpot1 >= 923)) ){
      // ignore adjpot input
      Serial.print("no adjustment 1\n");
    }
    else{
      if(adjpot1 < 511){
        stepper1.step(10);
        return;
      }
      else if (adjpot1 >= 511){
        stepper1.step(-10);
        return;
      }
```

```
    }

    //adjpot2
    if(((adjpot2 >= 0) && (adjpot2 <=100) )|| ((adjpot2 <=1023) && (adjpot2 >= 923)) ){
      // ignore adjpot input
      Serial.print("no adjustment 2\n");
    }
    else{
      if(adjpot2 < 511){
        stepper2.step(10);
        return;
      }
      else if (adjpot2 >= 511){
        stepper2.step(-10);
        return;
      }
    }
    // end of adjustment



    potx = map(potx, 0, 1023, 0, a);
    poty = map(poty, 0, 1023, 0, b);
    potz = map(potz, 0, 1023, 0, h);


    if ((potx <= potx_prev + e) && (potx >= potx_prev - e)) potx_changed = 0;
    else potx_changed = 1;

    if ((poty <= poty_prev + e) && (poty >= poty_prev - e)) poty_changed = 0;
    else poty_changed = 1;

    if ((potz <= potz_prev + e) && (potz >= potz_prev - e)) potz_changed = 0;
    else potz_changed = 1;


    if ((potx_changed == 0) && (poty_changed == 0) && (potz_changed == 0) ){
      flag = 0;
    }else{
```

```
      flag = 1;
    }

    x_curr = potx;
    y_curr = poty;
    z_curr = potz;

    l1_curr = sqrt(x_curr*x_curr + y_curr*y_curr + (h - z_curr)*(h - z_curr));
    l2_curr = sqrt( (a - x_curr)*(a - x_curr) + y_curr*y_curr + (h - z_curr)*(h - z_curr));
    l3_curr = sqrt( (a - x_curr)*(a - x_curr) + (b - y_curr)*(b - y_curr) + (h - z_curr)*(h - z_curr) );
    l4_curr = sqrt( x_curr*x_curr + (b - y_curr)*(b - y_curr) + (h - z_curr)*(h - z_curr) );

    del_l1 = l1_curr - l1_prev;
    del_l2 = l2_curr - l2_prev;
    del_l3 = l3_curr - l3_prev;
    del_l4 = l4_curr - l4_prev;

    Serial.print("pot val\n");

    Serial.print(potx); Serial.print('\n');
    Serial.print(poty); Serial.print('\n');
    Serial.print(potz); Serial.print('\n');
    Serial.print('\n');

    Serial.print(del_l1); Serial.print('\n');
    Serial.print(del_l2); Serial.print('\n');
    Serial.print(del_l3); Serial.print('\n');
    Serial.print(del_l4); Serial.print('\n');
     Serial.print('\n');

    st1_steps = (1024*del_l1)/(3.1416*r);
    st2_steps = (1024*del_l2)/(3.1416*r);
    st3_steps = (1024*del_l3)/(3.1416*r);
    st4_steps = (1024*del_l4)/(3.1416*r);

    Serial.println("steps \n");
    Serial.print(st1_steps); Serial.print('\n');
    Serial.print(st2_steps); Serial.print('\n');
    Serial.print(st3_steps); Serial.print('\n');
```

```
  Serial.print(st4_steps); Serial.print('\n');
  Serial.print('\n');

  if(flag == 1){

    //moving stepper1, stepper2 concurrently
    int st1_rotdir = abs(st1_steps)/st1_steps;
    int st2_rotdir = abs(st2_steps)/st2_steps;

    st1_steps = abs(st1_steps);
    st2_steps = abs(st2_steps);

    int st1_rem = st1_steps % step_size; // steps that cannot be included using the specified step size
    int st2_rem = st2_steps % step_size;

    for(int i=step_size; i<max(st1_steps, st2_steps); i = i+step_size){
      if(st1_steps >= step_size){
        stepper1.step(st1_rotdir*step_size);
        st1_steps = st1_steps - step_size;
      }

      if(st2_steps >= step_size){
        stepper2.step(st2_rotdir*step_size);
        st2_steps = st2_steps - step_size;
      }
    }
    //for end

    //performing the remaining steps
    stepper1.step(st1_rotdir*st1_rem);
    stepper2.step(st2_rotdir*st2_rem);


  } // if end

  delay(3000);

  potx_prev = potx;
  poty_prev = poty;
  potz_prev = potz;

  l1_prev = l1_curr;
  l2_prev = l2_curr;
  l3_prev = l3_curr;
  l4_prev = l4_curr;


}
```

Similar code was used for servo motors 3 and 4 which was uploaded using arduino 2.

Code for ESP32 camera module:

```
#include "esp_camera.h"
#include <Arduino.h>
#include <WiFi.h>
```

```
#include <AsyncTCP.h>
#include <ESPAsyncWebServer.h>
#include <sstream>

//Camera related constants
#define PWDN_GPIO_NUM     32
#define RESET_GPIO_NUM    -1
#define XCLK_GPIO_NUM      0
#define SIOD_GPIO_NUM     26
#define SIOC_GPIO_NUM     27
#define Y9_GPIO_NUM       35
#define Y8_GPIO_NUM       34
#define Y7_GPIO_NUM       39
#define Y6_GPIO_NUM       36
#define Y5_GPIO_NUM       21
#define Y4_GPIO_NUM       19
#define Y3_GPIO_NUM       18
#define Y2_GPIO_NUM        5
#define VSYNC_GPIO_NUM    25
#define HREF_GPIO_NUM     23
#define PCLK_GPIO_NUM     22

const char* ssid     = "SpiderCam Live Feed";
const char* password = "12345678";
AsyncWebServer server(80);
AsyncWebSocket wsCamera("/Camera");
uint32_t cameraClientId = 0;


const char* htmlHomePage PROGMEM = R"HTMLHOMEPAGE(

// webpage code here
<!DOCTYPE html>
<html>

<head>
  <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1, user-
scalable=no">
  <style>
    body {
      margin: 0px;
      padding: 0px;
      border: 0px;
    }
    .top-bar {
      background-color: rgba(171, 30, 38, 255);
      padding: 5px;
      margin: 0px;
      color: white;
    }
    #small-info {
      font-size: 12px;
      color: #ececec;
```

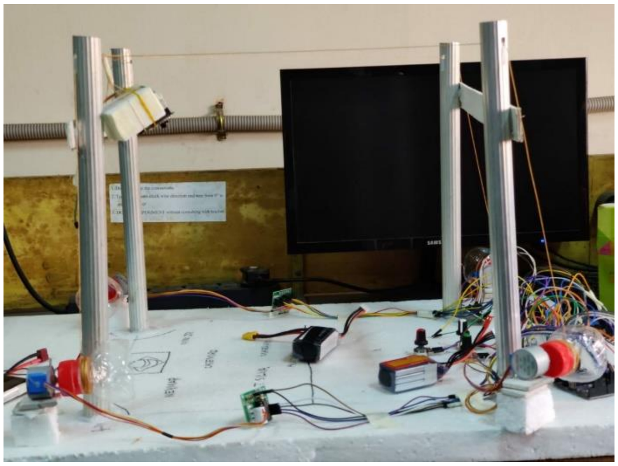
```
    }
    .arrows {
      font-size: 20px;
      color: white;
    }
    td.button {
      background-color: black;
      border-radius: 10%;
      box-shadow: 5px 5px 8px #888888;
      padding: 15px;
    }
    td.button:active {
      transform: translate(5px, 5px);
      box-shadow: none;
    }
    .noselect {
      -webkit-touch-callout: none;
      /* iOS Safari */
      -webkit-user-select: none;
      /* Safari */
      -khtml-user-select: none;
      /* Konqueror HTML */
      -moz-user-select: none;
      /* Firefox */
      -ms-user-select: none;
      /* Internet Explorer/Edge */
      user-select: none;
      /* Non-prefixed version, currently
                                supported by Chrome and Opera */
    }
    #container {
      width: 2;
      overflow: hidden;
    }
  </style>
</head>
<body class="noselect" align="center"
  style="background-color:rgb(255, 223, 223); font-family: Arial, Helvetica, sans-serif;">
  <!--h2 style="color: teal;text-align:center;">Wi-Fi Camera &#128663; Control</h2-->
  <h2 class="top-bar">
    SpiderCam Live View Panel
    <br>
    <span id="small-info" class="output">
      developed by: touhid, tawsif, saaheb, arnob
    </span>
 </h2>
  <table id="mainTable" style="width:400px;margin:auto;table-layout:fixed" CELLSPACING=10>
    <tr>
      <img id="cameraImage" src="" style="width:400px;height:300px"></td>
    </tr>
  </table>
  <script>
    var webSocketCameraUrl = "ws:\/\/" + window.location.hostname + "/Camera";
```

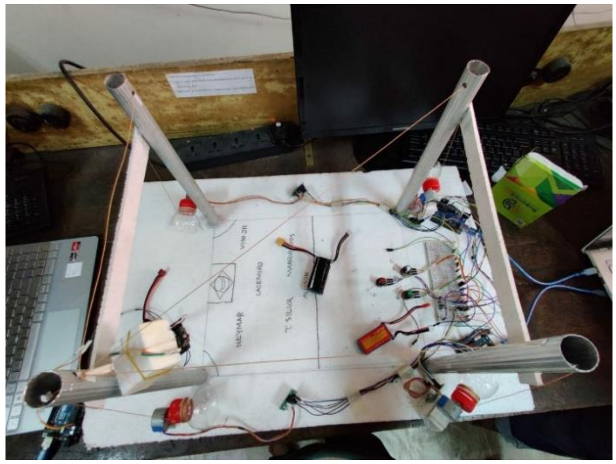
```
    var websocketCamera;

    function initCameraWebSocket() {
      websocketCamera = new WebSocket(webSocketCameraUrl);
      websocketCamera.binaryType = 'blob';
      websocketCamera.onopen = function (event) { };
      websocketCamera.onclose = function (event) { setTimeout(initCameraWebSocket, 2000); };
      websocketCamera.onmessage = function (event) {
        var imageId = document.getElementById("cameraImage");
        imageId.src = URL.createObjectURL(event.data);
      };
    }
    function initWebSocket() {
      initCameraWebSocket();
      }
    window.onload = initWebSocket;
    document.getElementById("mainTable").addEventListener("touchend", function (event) {
      event.preventDefault()
    });
  </script>
</body>
</html>)HTMLHOMEPAGE";

void handleRoot(AsyncWebServerRequest *request)
{
  request->send_P(200, "text/html", htmlHomePage);
}
void handleNotFound(AsyncWebServerRequest *request)
{
    request->send(404, "text/plain", "File Not Found");
}

void onCameraWebSocketEvent(AsyncWebSocket *server,
                      AsyncWebSocketClient *client,
                      AwsEventType type,
                      void *arg,
                      uint8_t *data,
                      size_t len)
{
  switch (type)
  {
    case WS_EVT_CONNECT:
      cameraClientId = client->id();
      break;
    case WS_EVT_DISCONNECT:
      cameraClientId = 0;
      break;
    case WS_EVT_DATA:
      break;
    case WS_EVT_PONG:
    case WS_EVT_ERROR:
      break;
    default:
```

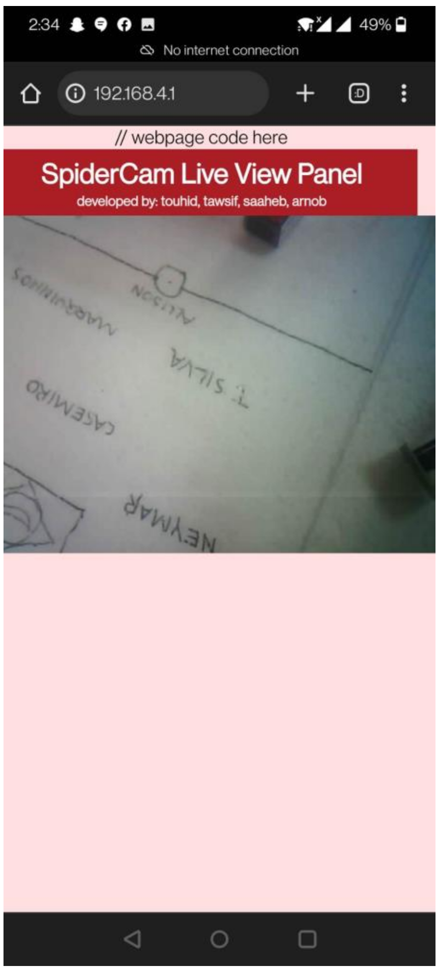
```
      break;
    }
}

void setupCamera()
{
  camera_config_t config;
  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;

  config.frame_size = FRAMESIZE_VGA;
  config.jpeg_quality = 10;
  config.fb_count = 1;

  // camera init
  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK)
  {
    return;
  }

  if (psramFound())
  {
    heap_caps_malloc_extmem_enable(20000);
  }
}

void sendCameraPicture()
{
  if (cameraClientId == 0)
  {
    return;
  }
  unsigned long  startTime1 = millis();
  //capture a frame
  camera_fb_t * fb = esp_camera_fb_get();
  if (!fb)
```

```
  {
      return;
  }
  unsigned long  startTime2 = millis();
  wsCamera.binary(cameraClientId, fb->buf, fb->len);
  esp_camera_fb_return(fb);

  //Wait for message to be delivered
  while (true)
  {
    AsyncWebSocketClient * clientPointer = wsCamera.client(cameraClientId);
    if (!clientPointer || !(clientPointer->queueIsFull()))
    {
      break;
    }
    delay(1);
  }

  unsigned long  startTime3 = millis();
}

void setup(void)
{
  Serial.begin(115200);
  WiFi.softAP(ssid, password);
  IPAddress IP = WiFi.softAPIP();
  server.on("/", HTTP_GET, handleRoot);
  server.onNotFound(handleNotFound);
  wsCamera.onEvent(onCameraWebSocketEvent);
  server.addHandler(&wsCamera);
  server.begin();
  Serial.println("HTTP server started");
  setupCamera();
}

void loop()
{
  wsCamera.cleanupClients();
  sendCameraPicture();
}
```

Final Product:







Here, 4 potentiometers are used as a kind of control panel to individually adjust the wire lengths and 3 other potentiometers are used to give x,y,z coordinate values which the system than takes as input, compares it against previously stored coordinates and moves the camera accordingly. Lastly, the Arduinos were mainly used to power the system and upload code to the servo motors and the ESP32 module.

EQUIPMENTS:

- Servo motors – 4pc
- Arduino Uno – 3pc
- ESP32 module – 1pc
- Wires
- Power supply – 2pc
- Potentiometer – 7pc

Results:

Our test results have mostly shown positive results. The project was almost fully successful with the system being able to move the camera according to the inputs within an acceptable range of error. Also, the ESP32 module was successful in livestreaming its video with which we were able to get a good view of the whole field by moving the camera. All in all, in the primary stage, our project of making a cable suspended spider-cam with three dimensional was a resounding success.

Limitations:

- Because of low-grade servo motors, the system is quite slow and it takes a moderate amount of time for the camera to move.
- The ESP32 model's video quality was also not quite up to the mark.
- Also, because of low grade wires, there was a tendency of the wires to loosen up and decrease the overall efficiency of the system.

- Due to limited budget the model was very miniature.

Further Modifications:

Although the system works pretty fine, some further modifications can be done to improve the performance.

A new "mircrostep algorithm" can be used to move the point of control. In this algorithm to get to a desired coordinate, the system would move a small length at a time. For example, if step size is set at 1cm, then the system would move 1 cm at a time, in the required direction. As a result, the overall movement would be much smoother and a particular wire would not expand or contract too much at a time.

Also, using image processing technology, the system can be given the capability of following a particular moving object or person on the field. The camera stream would be input to an image processing system which would track the desired object and calculate the required direction of movement.