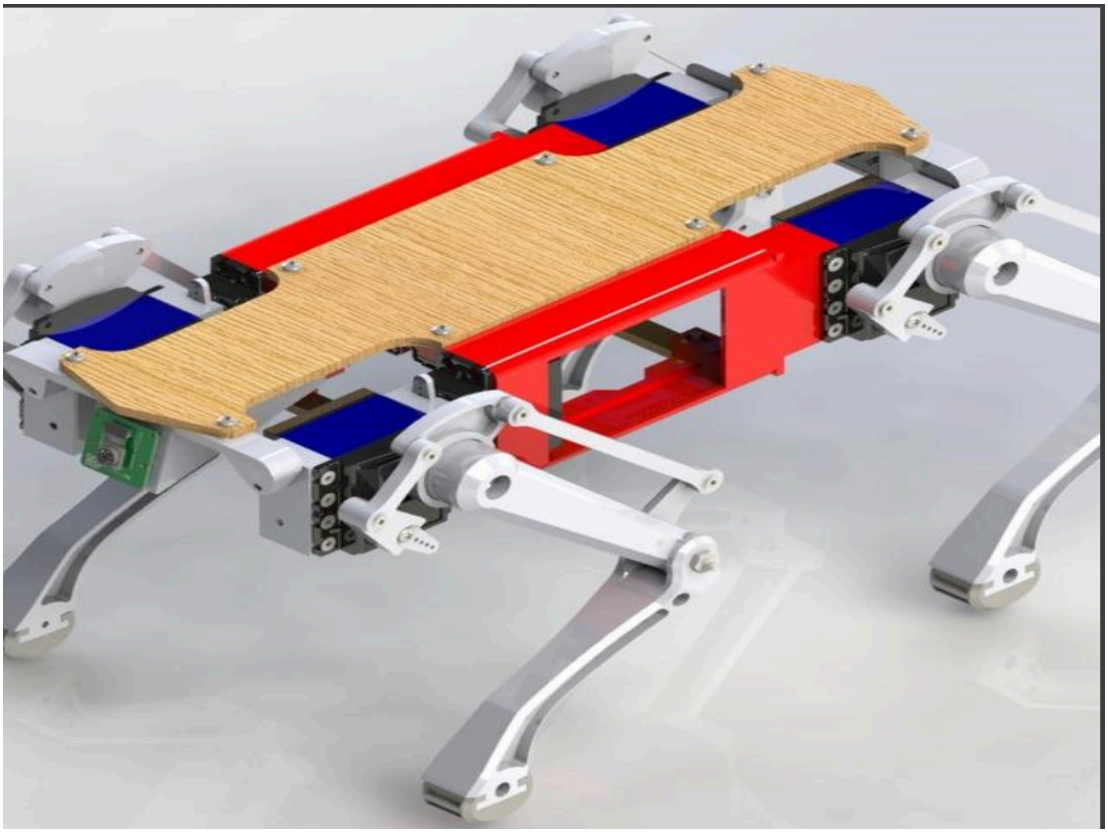
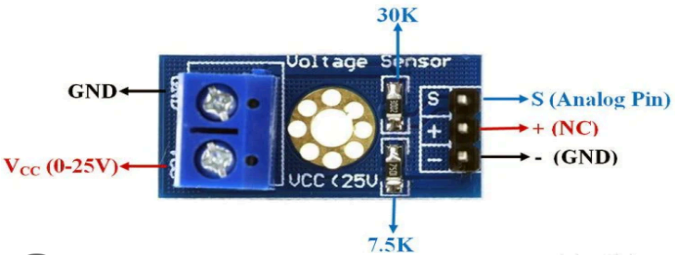
PROJECT QUADRUPED

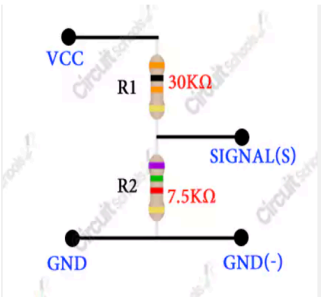


PROGRESS REPORT

DATE: 12/02/2024

VOLTAGE SENSORS:





A voltage sensor is a device that measures voltage. Voltage sensors can measure the voltage in various ways, from measuring high voltages to detecting low current levels. These devices are essential for many applications, including industrial controls and power systems.

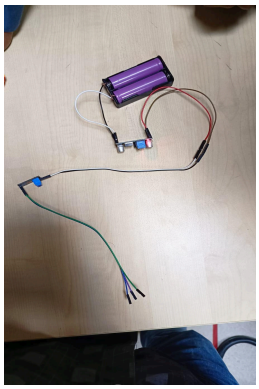
OUTCOME:

COMPONENTS USED:-

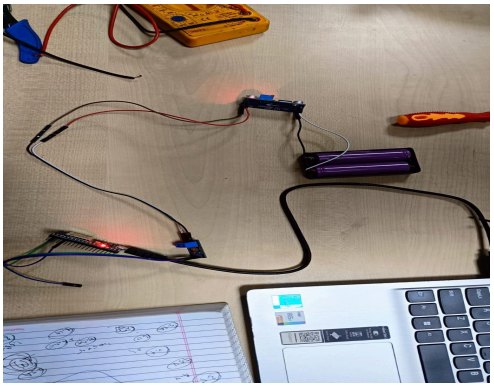
- ESP32
- VOLTAGE SENSOR
- BUCK MODULE(LM2596 DC-DC)

CONNECTIONS:-

- ➢ We connected the battery with a buck module.
- ➢ Output of the buck module is connected to the voltage sensor.
- ➢ Voltage sensor Vcc is connected to ESP32 5V pin,
- ➢ GND is connected to ESP32 GND
- ➢ S(Analog Pin) is connected to ESP32 pin P34



Battery connected with Buck module



Final connection

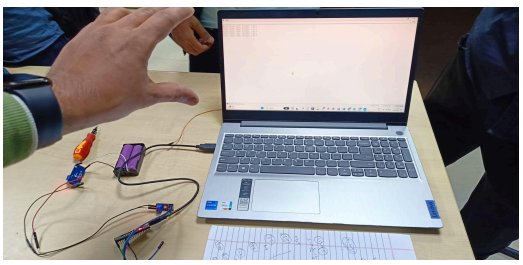
CODE;

```
// Include the necessary libraries
#include <Arduino.h>


// Define the analog pin connected to the voltage sensor
const int voltageSensorPin = 34; // Example pin, you may need to change it according to your wiring


void setup() {
  // Initialize Serial communication
  Serial.begin(115200);
```

```
  // Configure the voltage sensor pin as an input
  pinMode(voltageSensorPin, INPUT);
}

void loop() {
  // Read the analog value from the voltage sensor
  int sensorValue = analogRead(voltageSensorPin);

  // Convert the analog value to voltage (assuming 3.3V reference)
  float voltage = sensorValue * (3.3 / 4095.0); // ESP32 ADC is 12-bit (0-4095)

  // Print the sensor value and voltage to the Serial Monitor
  Serial.print("Sensor Value: ");
  Serial.print(sensorValue);
  Serial.print(", Voltage: ");
  Serial.print(voltage);
  Serial.println(" V");

  // You can add any further processing or actions here

  delay(1000); // Add a small delay to avoid flooding the Serial Monitor
}
```

RESULT :

Successfully working



DATE: 13/02/2024

RELAY

- A relay is an electrically operated switch that controls the flow of current in a circuit. It uses a small electric current to control a larger one, allowing for remote or automated control of electrical systems. Relays are widely used in various applications, including industrial automation, telecommunications, and automotive systems.

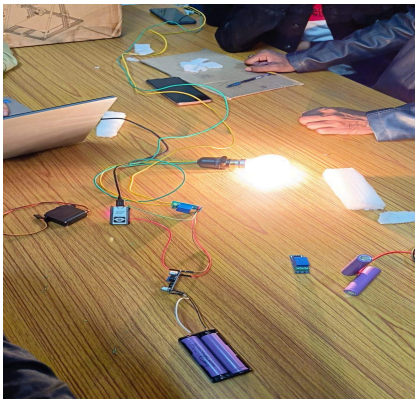
OUTCOME:

COMPONENTS USED:-

- ESP32
- RELAY MODULE
- BUCK MODULE(LM2596 DC-DC)

CONNECTIONS:-

- ➢ We connected the battery with a buck module.
- ➢ Output of the buck module is connected to the relay module.
- ➢ Relay module Vcc is connected to ESP32 5V pin.
- ➢ GND of Relay is connected to ESP32 GND
- ➢ IN pin of Relay is connected to ESP32 pin P2



CODE:

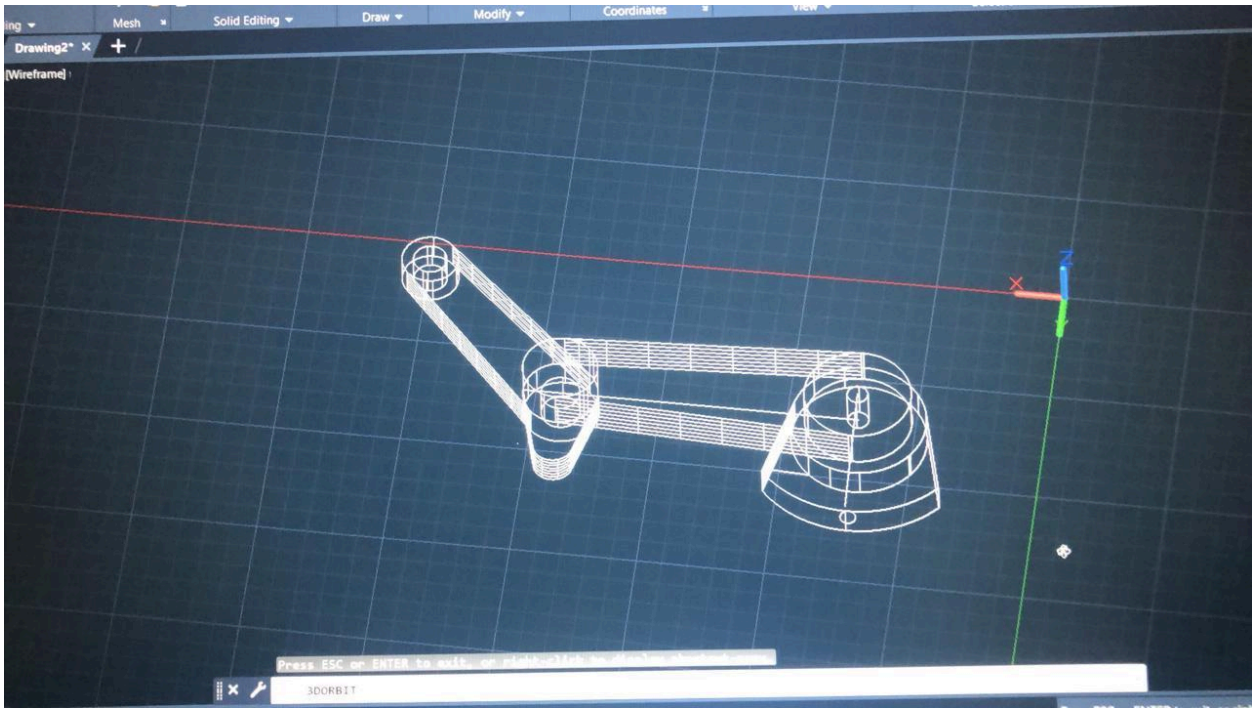
```
//code to control realy with ESP-32 bluetooth,DATE-13 FEB 2024;
#include <BluetoothSerial.h>

char command;
BluetoothSerial SerialBT;  // Bluetooth Serial object
const int relayPin = 2;    // Digital pin for relay module

void setup() {
  pinMode(relayPin, OUTPUT);
  Serial.begin(9600);
   SerialBT.begin(115200);

  // Initialize Bluetooth Serial
  SerialBT.begin("ESP32_BT");  // You can change the device name as needed
}
// ledcSetup(0, 5000, 8);  // Channel 0 for Motor A, 5 kHz frequency, 8-bit
resolution
 // ledcAttachPin(relayPin, R1);
void loop() {
  while (SerialBT.available() == 0) {}

    command = SerialBT.read();
    if (command == 'A') {
      digitalWrite(relayPin, HIGH); // Turn ON the relay
      Serial.println("Relay ON");
       SerialBT.println("Relay ON");
       //delay(100
    }
```

```
  else {
          digitalWrite(relayPin, LOW);
            // Turn ON the relay
      Serial.println("Relay OFF");
       SerialBT.println("Relay OFF");
       //delay(1000);
  }

}
```

RESULT:

Successfully controlled via mobile(bluetooth).

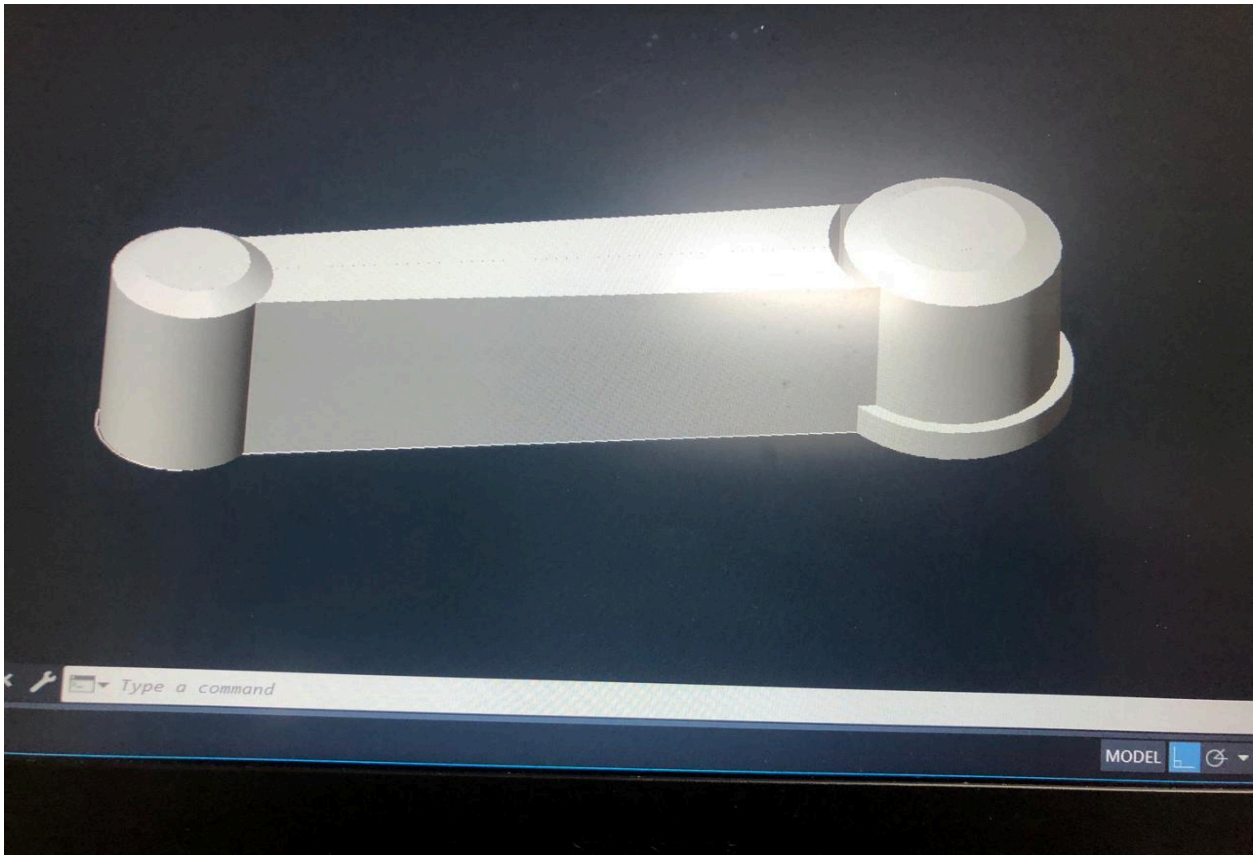
CAD



Only a rough 3d sketch of a limb, needs to be optimized.

DATE: 14/02/2024

CAD:



Arm limb, 100mm length between two centres, 30mm max thickness, bore needed for servo shaft and elbow joint shaft.

ULTRASONIC SENSOR

An ultrasonic sensor is a device that emits ultrasonic sound waves and detects their reflection off objects in its vicinity. These sensors work based on the principle of echolocation, similar to how bats navigate. They typically consist of a transmitter, which emits ultrasonic waves, and a receiver, which detects the waves after they bounce off objects. By measuring the time taken for the waves to return, the sensor can determine the distance to the object. Ultrasonic sensors are commonly used in various applications such as proximity detection, object avoidance in robotics, liquid level measurement, and distance measurement in automotive parking systems. They are favored for their accuracy, reliability, and ability to operate in diverse environmental conditions.

Formula used:- distance = duration * 0.034 / 2

SERVO MOTOR

A servo motor is a rotary actuator that allows for precise control of angular position, velocity, and acceleration. It consists of a motor, a feedback device (such as a potentiometer or an encoder), and a control circuit. Servo motors are widely used in various applications, including robotics, automation, remote control systems, and model aircraft.

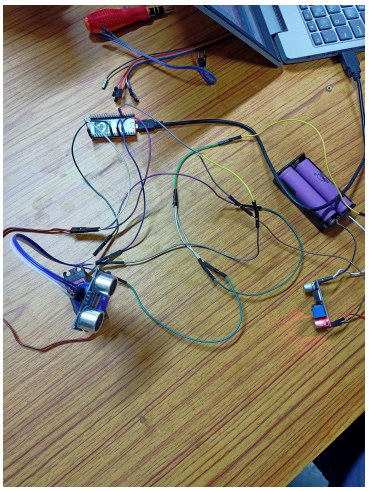
OUTCOME:

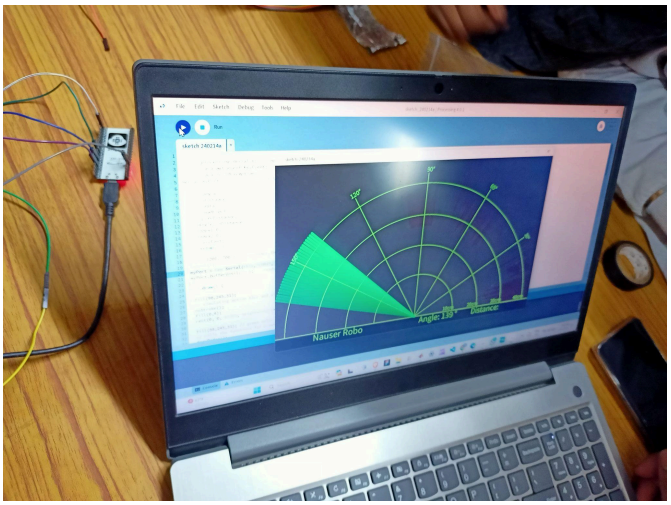
COMPONENTS USED:-

- ESP32
- ULTRASONIC SENSOR
- SERVO MOTOR
- BUCK MODULE(LM2596 DC-DC)

CONNECTIONS:-

- ➢ We connected the battery with a buck module.
- ➢ Output of the buck module is connected to the Ultrasonic Sensor and Servo Motor.
- ➢ Connect all the pin according to define pin in code.





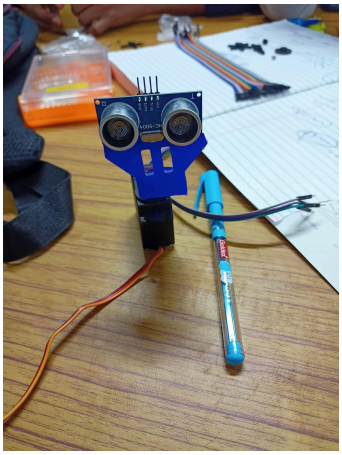
RADAR SYSTEM

CODE

- CHECKING FOR ULTRASONIC SENSOR:

```
//code to measure the distance using ulrasonic sensor, Date-14 Feb. 2024;
const int trigPin = 12;  // Arduino digital pin connected to the sensor's trigger pin
const int echoPin = 13;  // Arduino digital pin connected to the sensor's echo pin

void setup() {
  Serial.begin(9600);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}

void loop() {
  // Trigger pulse to start the measurement
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Measure the duration of the echo pulse
  long duration = pulseIn(echoPin, HIGH);
```

```
  // Calculate distance in centimeters
  float distance_cm = duration * 0.034 / 2;

  // Print the distance to the Serial Monitor
  Serial.print("Distance: ");
  Serial.print(distance_cm);
  Serial.println(" cm");

  // Wait for a moment before taking the next measurement
  delay(500);
}
```



- CHECKING FOR SERVO MOTOR:

```
//code to move the servo motor slowly, Date-14 Feb. 2024;
#include <Servo.h>

Servo myServo;

void setup() {
  myServo.attach(12);
  Serial.begin(9600);
}

void loop() {
  // Move the servo to 180 degrees slowly
  for (int angle = 0; angle <= 180; angle += 1) {
    myServo.write(angle);
```

```
    Serial.print("angle=");
    Serial.println(angle);
    delay(15);  // Adjust the delay for the desired speed
  }

  delay(1000); // Wait for a moment at 180 degrees

  // Move the servo back to 0 degrees slowly
  for (int angle = 180; angle >= 0; angle -= 1) {
    myServo.write(angle);
        Serial.print("angle=");
    Serial.println(angle);
    delay(15);  // Adjust the delay for the desired speed
  }

  delay(1000); // Wait for a moment at 0 degrees
}
```

CODE FOR RADAR SYSTEM

- arduino code.txt

```
// Includes the Servo library
#include <Servo.h>.
// Defines Tirg and Echo pins of the Ultrasonic Sensor
const int trigPin = 10;
const int echoPin = 11;
// Variables for the duration and the distance
long duration;
int distance;
Servo myServo; // Creates a servo object for controlling the servo motor
void setup() {
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin, INPUT); // Sets the echoPin as an Input
  Serial.begin(9600);
  myServo.attach(12); // Defines on which pin is the servo motor attached
}
void loop() {
  // rotates the servo motor from 15 to 165 degrees
  for(int i=15;i<=165;i++){
```

```
  myServo.write(i);
  delay(30);
   distance = calculateDistance();// Calls a function for calculating the distance measured by the Ultrasonic sensor for each degree

  Serial.print(i); // Sends the current degree into the Serial Port
   Serial.print(","); // Sends addition character right next to the previous value needed later in the Processing IDE for indexing
  Serial.print(distance); // Sends the distance value into the Serial Port
   Serial.print("."); // Sends addition character right next to the previous value needed later in the Processing IDE for indexing
  }
  // Repeats the previous lines from 165 to 15 degrees
  for(int i=165;i>15;i--){
  myServo.write(i);
  delay(30);
  distance = calculateDistance();
  Serial.print(i);
  Serial.print(",");
  Serial.print(distance);
  Serial.print(".");
  }
}
// Function for calculating the distance measured by the Ultrasonic sensor
int calculateDistance(){

  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
   duration = pulseIn(echoPin, HIGH); // Reads the echoPin, returns the sound wave travel time in microseconds
  distance= duration*0.034/2;
  return distance;
}
```

- processing code.txt

```
import processing.serial.*; // imports library for serial communication
import java.awt.event.KeyEvent; // imports library for reading the data from the serial port
import java.io.IOException;
Serial myPort; // defines Object Serial
// defubes variables
String angle="";
String distance="";
String data="";
String noObject;
float pixsDistance;
int iAngle, iDistance;
int index1=0;
int index2=0;
PFont orcFont;
void setup() {

 size (1200, 700); // **CHANGE THIS TO YOUR SCREEN RESOLUTION**
 smooth();
 myPort = new Serial(this,"COM4", 9600); // starts the serial communication
  myPort.bufferUntil('.'); // reads the data from the serial port up to the character '.'. So actually it reads this: angle,distance.
}
void draw() {
   fill(98,245,31);
  // simulating motion blur and slow fade of the moving line
  noStroke();
  fill(0,4);
  rect(0, 0, width, height-height*0.065);

  fill(98,245,31); // green color
  // calls the functions for drawing the radar
  drawRadar();
  drawLine();
  drawObject();
  drawText();
}
void serialEvent (Serial myPort) { // starts reading data from the Serial Port
```

```
  // reads the data from the Serial Port up to the character '.' and puts it into the String variable "data".
  data = myPort.readStringUntil('.');
  data = data.substring(0,data.length()-1);

  index1 = data.indexOf(","); // find the character ',' and puts it into the variable "index1"
   angle= data.substring(0, index1); // read the data from position "0" to position of the variable index1 or thats the value of the angle the Arduino Board sent into the Serial Port
    distance= data.substring(index1+1, data.length()); // read the data from position "index1" to the end of the data pr thats the value of the distance

  // converts the String variables into Integer
  iAngle = int(angle);
  iDistance = int(distance);
}
void drawRadar() {
  pushMatrix();
  translate(width/2,height-height*0.074); // moves the starting coordinats to new location
  noFill();
  strokeWeight(2);
  stroke(98,245,31);
  // draws the arc lines
  arc(0,0,(width-width*0.0625),(width-width*0.0625),PI,TWO_PI);
  arc(0,0,(width-width*0.27),(width-width*0.27),PI,TWO_PI);
  arc(0,0,(width-width*0.479),(width-width*0.479),PI,TWO_PI);
  arc(0,0,(width-width*0.687),(width-width*0.687),PI,TWO_PI);
  // draws the angle lines
  line(-width/2,0,width/2,0);
  line(0,0,(-width/2)*cos(radians(30)),(-width/2)*sin(radians(30)));
  line(0,0,(-width/2)*cos(radians(60)),(-width/2)*sin(radians(60)));
  line(0,0,(-width/2)*cos(radians(90)),(-width/2)*sin(radians(90)));
  line(0,0,(-width/2)*cos(radians(120)),(-width/2)*sin(radians(120)));
  line(0,0,(-width/2)*cos(radians(150)),(-width/2)*sin(radians(150)));
  line((-width/2)*cos(radians(30)),0,width/2,0);
  popMatrix();
}
void drawObject() {
  pushMatrix();
  translate(width/2,height-height*0.074); // moves the starting coordinats to new location
```

```
  strokeWeight(9);
  stroke(255,10,10); // red color
   pixsDistance = iDistance*((height-height*0.1666)*0.025); // covers the distance from the sensor from cm to pixels
  // limiting the range to 40 cms
  if(iDistance<40){
    // draws the object according to the angle and the distance
line(pixsDistance*cos(radians(iAngle)),-pixsDistance*sin(radians(iAngle)),(width-width*0.505)*cos(radians(iAngle)),-(width-width*0.505)*sin(radians(iAngle)));
  }
  popMatrix();
}
void drawLine() {
  pushMatrix();
  strokeWeight(9);
  stroke(30,250,60);
  translate(width/2,height-height*0.074); // moves the starting coordinats to new location

line(0,0,(height-height*0.12)*cos(radians(iAngle)),-(height-height*0.12)*sin(radians(iAngle))); // draws the line according to the angle
  popMatrix();
}
void drawText() { // draws the texts on the screen
  pushMatrix();
  if(iDistance>40) {
  noObject = "Out of Range";
  }
  else {
  noObject = "In Range";
  }
  fill(0,0,0);
  noStroke();
  rect(0, height-height*0.0648, width, height);
  fill(98,245,31);
  textSize(25);

  text("10cm",width-width*0.3854,height-height*0.0833);
  text("20cm",width-width*0.281,height-height*0.0833);
  text("30cm",width-width*0.177,height-height*0.0833);
  text("40cm",width-width*0.0729,height-height*0.0833);
```

```
  textSize(40);
  text("Nauser Robotics ", width-width*0.875, height-height*0.0277);
  text("Angle: " + iAngle +" °", width-width*0.48, height-height*0.0277);
  text("Distance: ", width-width*0.26, height-height*0.0277);
  if(iDistance<40) {
  text("        " + iDistance +" cm", width-width*0.225, height-height*0.0277);
  }
  textSize(25);
  fill(98,245,60);

translate((width-width*0.4994)+width/2*cos(radians(30)),(height-height*0.0907)-width/2*
sin(radians(30)));
  rotate(-radians(-60));
  text("30°",0,0);
  resetMatrix();

translate((width-width*0.503)+width/2*cos(radians(60)),(height-height*0.0888)-width/2*si
n(radians(60)));
  rotate(-radians(-30));
  text("60°",0,0);
  resetMatrix();

translate((width-width*0.507)+width/2*cos(radians(90)),(height-height*0.0833)-width/2*si
n(radians(90)));
  rotate(radians(0));
  text("90°",0,0);
  resetMatrix();

translate(width-width*0.513+width/2*cos(radians(120)),(height-height*0.07129)-width/2*
sin(radians(120)));
  rotate(radians(-30));
  text("120°",0,0);
  resetMatrix();

translate((width-width*0.5104)+width/2*cos(radians(150)),(height-height*0.0574)-width/2
*sin(radians(150)));
  rotate(radians(-60));
  text("150°",0,0);
  popMatrix();
}
```

DATE: 15/:02/2024

MPU 6050:

The MPU-6050 is a 12-axis movement sensor chip, commonly used in devices such as smartphones, smartwatches, and motion-controlled remote controls. It measures acceleration, rotation, and orientation in 3D space. The MPU-6050 also features a built-in gyroscope, magnetometer, and temperature sensor. It is widely used in various applications such as robotics, drones, gaming, fitness tracking, and virtual reality. The sensor's small form factor and low power consumption make it a popular choice in portable devices.

OUTCOME:

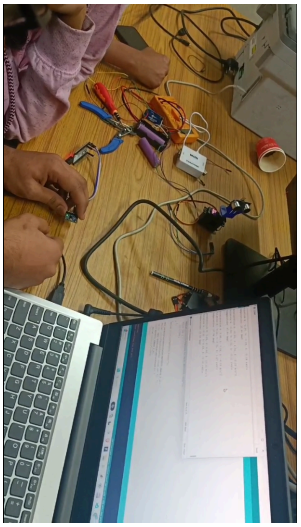
COMPONENTS USED:-

- ESP32
- MPU 6050

CONNECTIONS:-

➢ ESP 32:

- For I2C connection we use pin P21 and P22.
- Connect the SDA pin of MPU 6050 with P21 of ESP 32 and SCL pin with P22 of ESP 32.



CODE:

```
#include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>
#include <Wire.h>

Adafruit_MPU6050 mpu;

void setup(void) {
  Serial.begin(9600);
  while (!Serial)
    delay(10); // will pause Zero, Leonardo, etc until serial console opens

  Serial.println("Adafruit MPU6050 test!");

  // Try to initialize!
  if (!mpu.begin()) {
    Serial.println("Failed to find MPU6050 chip");
    while (1) {
      delay(10);
    }
  }
  Serial.println("MPU6050 Found!");

  mpu.setAccelerometerRange(MPU6050_RANGE_8_G);
  Serial.print("Accelerometer range set to: ");
  switch (mpu.getAccelerometerRange()) {
  case MPU6050_RANGE_2_G:
```

```
  Serial.println("+-2G");
  break;
case MPU6050_RANGE_4_G:
  Serial.println("+-4G");
  break;
case MPU6050_RANGE_8_G:
  Serial.println("+-8G");
  break;
case MPU6050_RANGE_16_G:
  Serial.println("+-16G");
  break;
}
mpu.setGyroRange(MPU6050_RANGE_500_DEG);
Serial.print("Gyro range set to: ");
switch (mpu.getGyroRange()) {
case MPU6050_RANGE_250_DEG:
  Serial.println("+- 250 deg/s");
  break;
case MPU6050_RANGE_500_DEG:
  Serial.println("+- 500 deg/s");
  break;
case MPU6050_RANGE_1000_DEG:
  Serial.println("+- 1000 deg/s");
  break;
case MPU6050_RANGE_2000_DEG:
  Serial.println("+- 2000 deg/s");
  break;
```

```
}

mpu.setFilterBandwidth(MPU6050_BAND_5_HZ);
Serial.print("Filter bandwidth set to: ");
switch (mpu.getFilterBandwidth()) {
case MPU6050_BAND_260_HZ:
  Serial.println("260 Hz");
  break;
case MPU6050_BAND_184_HZ:
  Serial.println("184 Hz");
  break;
case MPU6050_BAND_94_HZ:
  Serial.println("94 Hz");
  break;
case MPU6050_BAND_44_HZ:
  Serial.println("44 Hz");
  break;
case MPU6050_BAND_21_HZ:
  Serial.println("21 Hz");
  break;
case MPU6050_BAND_10_HZ:
  Serial.println("10 Hz");
  break;
case MPU6050_BAND_5_HZ:
  Serial.println("5 Hz");
  break;
}
```

```
  Serial.println("");
  delay(100);
}

void loop() {
  /* Get new sensor events with the readings */
  sensors_event_t a, g, temp;
  mpu.getEvent(&a, &g, &temp);

  /* Print out the values */
  Serial.print("Acceleration X: ");
  Serial.print(a.acceleration.x);
  Serial.print(", Y: ");
  Serial.print(a.acceleration.y);
  Serial.print(", Z: ");
  Serial.print(a.acceleration.z);
  Serial.println(" m/s^2");

  Serial.print("Rotation X: ");
  Serial.print(g.gyro.x);
  Serial.print(", Y: ");
  Serial.print(g.gyro.y);
  Serial.print(", Z: ");
  Serial.print(g.gyro.z);
  Serial.println(" rad/s");
```

```
  Serial.print("Temperature: ");
  Serial.print(temp.temperature);
  Serial.println(" degC");

  Serial.println("");
  delay(1500);
}
```

RESULT:

We have successfully operate the MPU 6050 with ESP 32.

DATE: 16/:02/2024

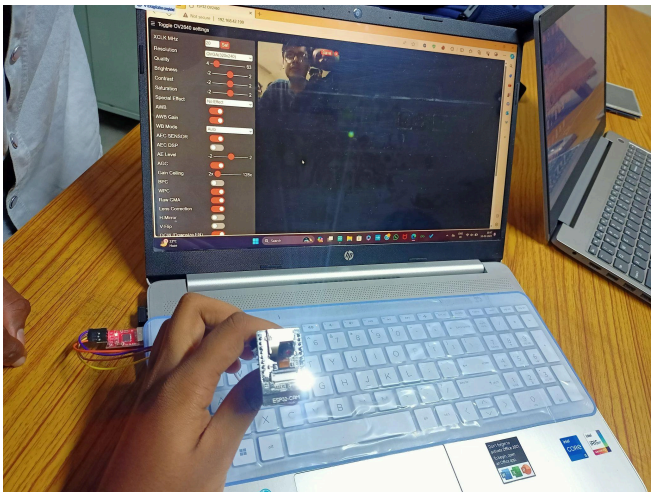
ESP 32 CAM:

The ESP32 is a microcontroller board with integrated Wi-Fi and Bluetooth capabilities, making it perfect for IoT applications. As a camera platform, it pairs well with imaging modules to allow for image capture and processing. There are various camera modules available for the ESP32, including SPI and I2C interfaces, allowing you to choose the one that best fits your project requirements.

OUTCOME:

COMPONENTS USED:-

- ESP32 CAM



CODE:

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


//
// WARNING!!! PSRAM IC required for UXGA resolution and high JPEG quality
//            Ensure ESP32 Wrover Module or other board with PSRAM is selected
```

```
//           Partial images will be transmitted if image exceeds buffer size
//
//           You must select partition scheme from the board menu that has at least 3MB APP space.
//           Face Recognition is DISABLED for ESP32 and ESP32-S2, because it takes up from 15
//           seconds to process single frame. Face Detection is ENABLED if PSRAM is enabled as well

// ===================
// Select camera model
// ===================
//#define CAMERA_MODEL_WROVER_KIT // Has PSRAM
#define CAMERA_MODEL_ESP_EYE // Has PSRAM
//#define CAMERA_MODEL_ESP32S3_EYE // Has PSRAM
//#define CAMERA_MODEL_M5STACK_PSRAM // Has PSRAM
//#define CAMERA_MODEL_M5STACK_V2_PSRAM // M5Camera version B Has PSRAM
//#define CAMERA_MODEL_M5STACK_WIDE // Has PSRAM
//#define CAMERA_MODEL_M5STACK_ESP32CAM // No PSRAM
//#define CAMERA_MODEL_M5STACK_UNITCAM // No PSRAM
//#define CAMERA_MODEL_AI_THINKER // Has PSRAM
//#define CAMERA_MODEL_TTGO_T_JOURNAL // No PSRAM
//#define CAMERA_MODEL_XIAO_ESP32S3 // Has PSRAM
// ** Espressif Internal Boards **
//#define CAMERA_MODEL_ESP32_CAM_BOARD
//#define CAMERA_MODEL_ESP32S2_CAM_BOARD
//#define CAMERA_MODEL_ESP32S3_CAM_LCD
//#define CAMERA_MODEL_DFRobot_FireBeetle2_ESP32S3 // Has PSRAM
```

```
//#define CAMERA_MODEL_DFRobot_Romeo_ESP32S3 // Has PSRAM
#include "camera_pins.h"

// ===========================
// Enter your WiFi credentials
// ===========================
const char* ssid = "Nauser_Robotics";
const char* password = "12345";

void startCameraServer();
void setupLedFlash(int pin);

void setup() {
  Serial.begin(9600);
  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_sccb_sda = SIOD_GPIO_NUM;
config.pin_sccb_scl = SIOC_GPIO_NUM;
config.pin_pwdn = PWDN_GPIO_NUM;
config.pin_reset = RESET_GPIO_NUM;
config.xclk_freq_hz = 20000000;
config.frame_size = FRAMESIZE_UXGA;
config.pixel_format = PIXFORMAT_JPEG; // for streaming
//config.pixel_format = PIXFORMAT_RGB565; // for face detection/recognition
config.grab_mode = CAMERA_GRAB_WHEN_EMPTY;
config.fb_location = CAMERA_FB_IN_PSRAM;
config.jpeg_quality = 12;
config.fb_count = 1;

// if PSRAM IC present, init with UXGA resolution and higher JPEG quality
//                     for larger pre-allocated frame buffer.
if(config.pixel_format == PIXFORMAT_JPEG){
  if(psramFound()){
    config.jpeg_quality = 10;
    config.fb_count = 2;
    config.grab_mode = CAMERA_GRAB_LATEST;
  } else {
    // Limit the frame size when PSRAM is not available
```

```
    config.frame_size = FRAMESIZE_SVGA;
    config.fb_location = CAMERA_FB_IN_DRAM;
   }
  } else {
   // Best option for face detection/recognition
   config.frame_size = FRAMESIZE_240X240;
#if CONFIG_IDF_TARGET_ESP32S3
   config.fb_count = 2;
#endif
  }

#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
```

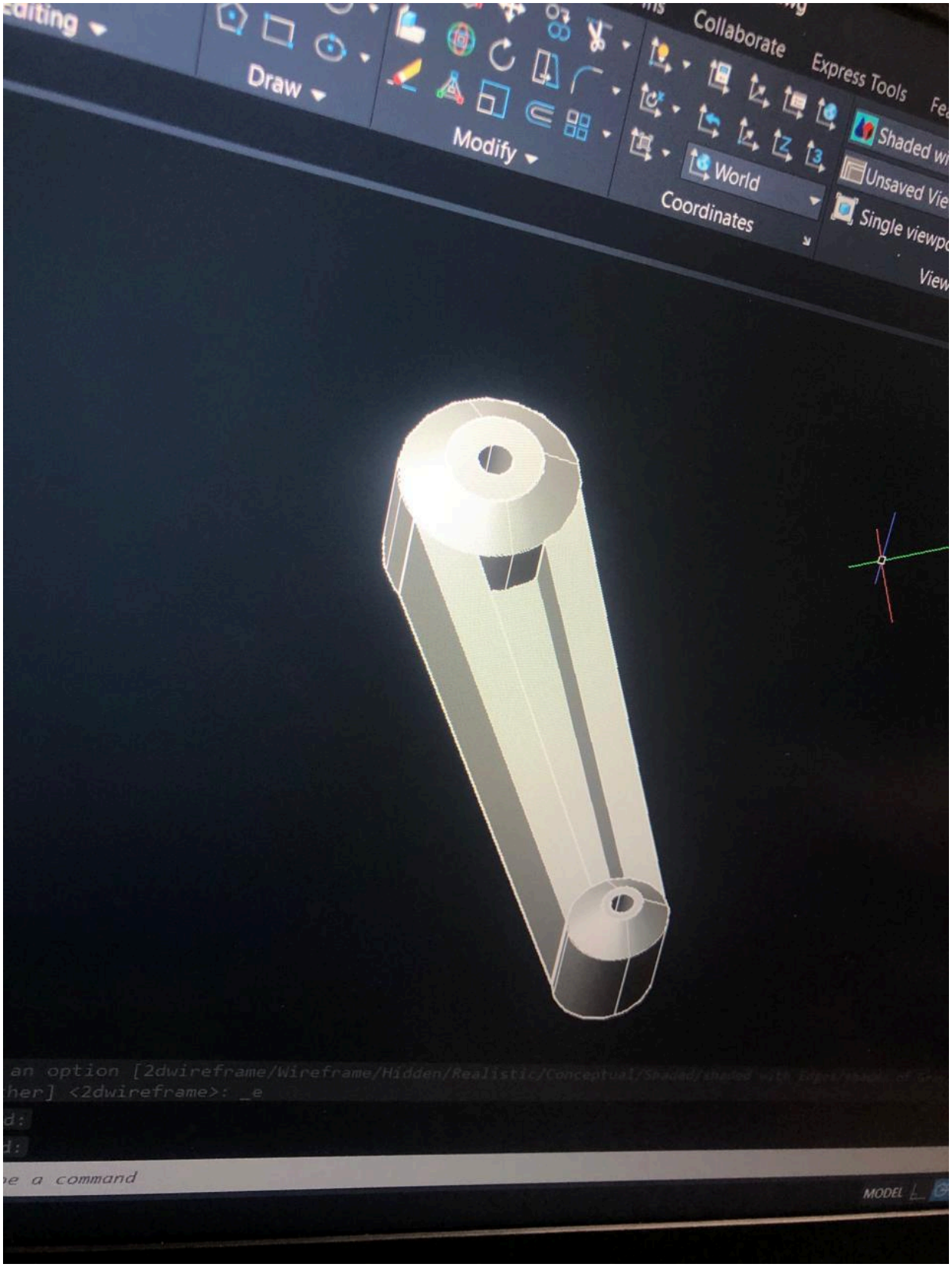
```
    s->set_brightness(s, 1); // up the brightness just a bit
    s->set_saturation(s, -2); // lower the saturation
  }
  // drop down frame size for higher initial frame rate
  if(config.pixel_format == PIXFORMAT_JPEG){
    s->set_framesize(s, FRAMESIZE_QVGA);
  }

#if defined(CAMERA_MODEL_M5STACK_WIDE) ||
defined(CAMERA_MODEL_M5STACK_ESP32CAM)
  s->set_vflip(s, 1);
  s->set_hmirror(s, 1);
#endif

#if defined(CAMERA_MODEL_ESP32S3_EYE)
  s->set_vflip(s, 1);
#endif

// Setup LED FLash if LED pin is defined in camera_pins.h
#if defined(LED_GPIO_NUM)
  setupLedFlash(LED_GPIO_NUM);
#endif

  WiFi.begin(ssid, password);
  WiFi.setSleep(false);

  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());
  Serial.println("' to connect");
}

void loop() {
  // Do nothing. Everything is done in another task by the web server
  delay(10000);
}
```

RESULT:

Successfully Done.

CAD 16/2/24:


Draw
Modify
Collaborate
Express Tools
World
Coordinates
Unsaved Vie
Single viewpo
View
an option [2dwireframe/Wireframe/Hidden/Realistic/Conceptual/Shaded/shaded with Edges/shaded of
her] <2dwireframe>: _e
e a command
MODEL

exactly to scale, the size on the screen will be the size of real object

1cm bore is given for servo shaft connector at head, 4mm bore for the tail part..the rectangular edges will be chamfered for material saving