Robot With Collision Detection

Razvan Popan

November 2024

1 Introduction

Build a robot that has ultrasonic sensors for collision detection and a gyroscope and accelerometer module. The robot will move using two DC motors and, and two Servo motors will be used to adjust the robot's movement. The MPU-6500 will be used to read data about the positon, speed and acceleration of the robot in order to control it's movement. This project will use an Arduino Mega2560 MCU and an ESP32, with the Mega will control the Servo Motors, the DC motors, will process the data from the Ultrasonic Sensors and the data received from the ESP32, via UART, which is connected to the MPU-6500 to get relevant data about the motion and position of the robot.

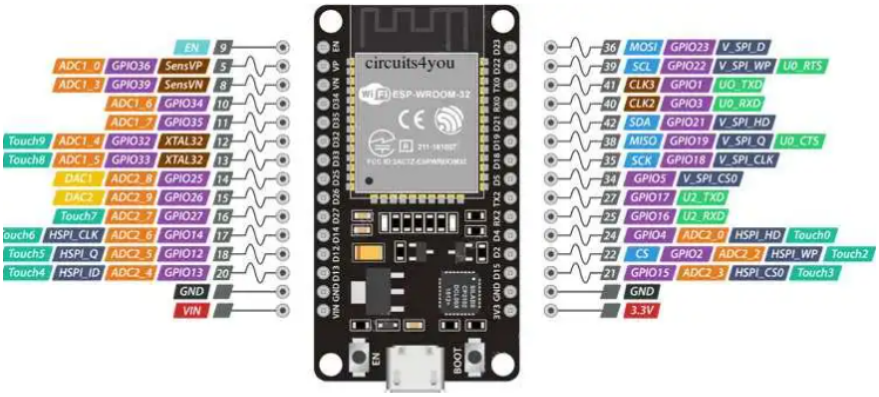

Figure 1: ESP32

The microcontrollers will be connected to eachother on the RX1 and TX1 pins (RX1 from Mega to TX1 from ESP32 and TX1 from Mega to RX1 from ESP32). The ESP32 will use the I^2C communication protocol to receive the data from the MPU-6500. They will be connected on the SDA and SCL pins (SDA from ESP32 to SDA of the MPU and SCL from the ESP32 to the SCL of the MPU).

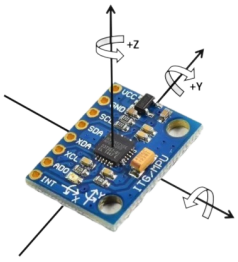

Figure 2: MPU-6500

Since the Mega is on 5V and the ESP32 is on 3.3V logic, we need to use a bi-directional logic level converter to ensure that we don't damage the boards, and the data is sent correctly.

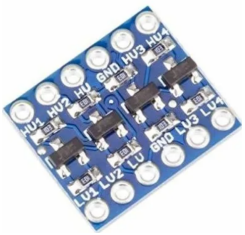

Figure 3: Bi-directional Logic Level Converter

For working with the DC motors an H-bridge is needed. The one we will be using is an L298N which has the capability of controlling two DC motors and has additional logic that prevents the damage H-bridge.

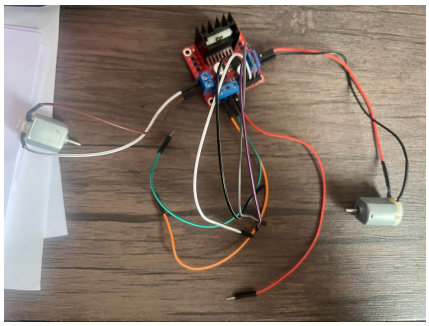

Figure 4: DC Motors and L298N H-Bridge

The following code is for the Arduino Mega 2560, it shows the libraries needed, the pins of the components and the setup

```
#include "Servo.h"
#include <stdio.h>

const int trigPin1 = 13;
const int echoPin1 = 12;
const int trigPin2 = 8;
const int echoPin2 = 7;

const int motor00 = 11;
const int motor01 = 10;
const int motor10 = 6;
const int motor11 = 5;
```

```
boolean stringComplete = false;
String inputString = "";

Servo myServo1, myServo2;

struct accelerometer{
  float x;
  float y;
  float z;
} accl;

struct gyroscope{
  float x;
  float y;
  float z;
} gyro;

void setup() {
  Serial.begin(9600);
  while(!Serial){
    delay(10); //wait for Serial Monitor to open
  }

  Serial1.begin(9600);
  while(!Serial1){
    delay(10);
  }
  //initialize motor pins
  stop();
  //reset Servo  motors to angle 0
  resetServo();


  //pin modes for ultrasonic sensors
  pinMode(trigPin1, OUTPUT);
  pinMode(echoPin1, INPUT);

  pinMode(trigPin2, OUTPUT);
  pinMode(echoPin2, INPUT);

  //pin modes for motors
  pinMode(motor00, OUTPUT);
  pinMode(motor01, OUTPUT);
  pinMode(motor10, OUTPUT);
  pinMode(motor11, OUTPUT);
}
```

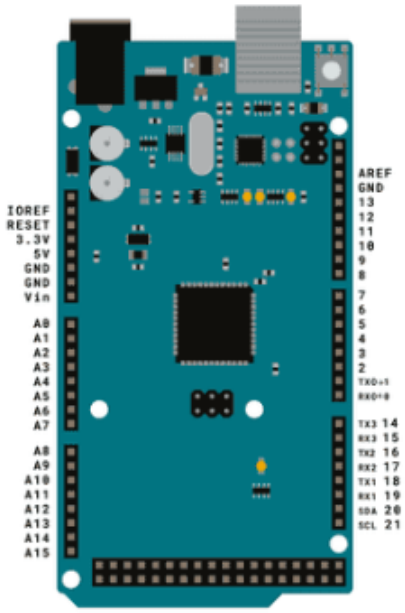

Figure 5: Arduino Mega 2560

2 Ultrasonic Sensors

To measure the distance using the ultrasonic sensors, we first set the trigger pin to '0', then send a pulse for 10 microseconds and then set it back to low. The ultrasonic sensor will detect the soundwave, and set the echo pin to a pulse who's width is proportional with the distance. To calculate the distance measured by the sensor we use the equation $speed = \frac{distance}{duration}$ so the distance will be $distance = speed \times duration$, where the speed is the speed of sound, and the duration is how much the echo pin is in HIGH logic. The speed of sound in $\frac{cm}{\mu s}$ is approximately .0343. This distance is divided by two because it travels from the sensor, to the nearest object and back.

```
bool ultrasonicSensor(const int echoPin, const int trigPin,
    char which) {

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

  float duration = pulseIn(echoPin, HIGH);
  float distance = (duration * .0343) / 2;
```

```
11
12    Serial.print("Distance: ");
13    Serial.println(distance);
```

The distance will be used to stop the motors when an object is too close (10 cm in this case) then the motors are started in reverse to go back for a short distance, and then the DC motors are stopped again.

```
1     if (distance < 10.0f) {
2       stop();
3       delay(1000);
4       start(motor00,motor01,false,30); //go back
5       start(motor10,motor11,false,30); //go back
6       delay(1000);
7       stop();
8     }
9   }
```

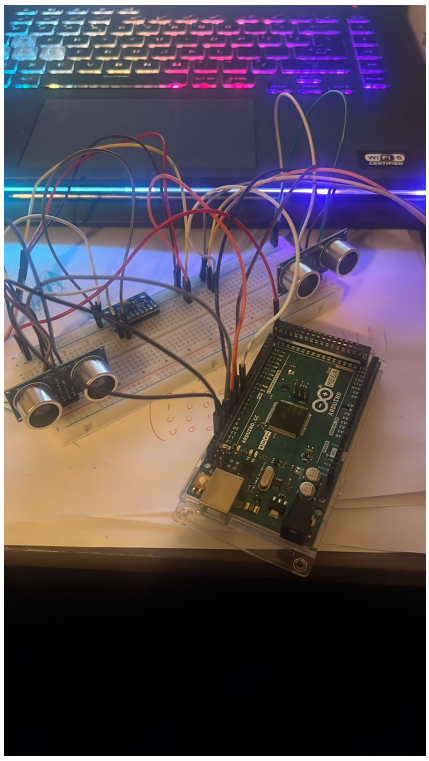

Figure 6: Ultrasonic Sensor with Arduino Mega

3 Servo Motors

```
void controlServo(int angle1, int angle2, int which) {
  switch (which) {
    case 0:
      myServo1.attach(9);
      delay(30);
      myServo1.write(angle1);
      delay(1000);
      myServo1.detach();
      break;
    case 1:
      myServo2.attach(4);
      delay(30);
      myServo2.write(angle2);
      delay(1000);
      myServo2.detach();
      break;
    case 2:
      myServo1.attach(9);
      myServo2.attach(4);

      delay(30);

      myServo1.write(angle1);
      myServo2.write(angle2);

      delay(1000);

      myServo1.detach();
      myServo2.detach();
      break;
    default:
     Serial.println("Wrong command for controlling Servos!")
        ;
  }
}
void resetServo() {
  myServo1.attach(9);
  myServo2.attach(4);
  delay(30);
  myServo1.write(0);
  myServo2.write(0);

  delay(1000);
  myServo1.detach();
  myServo2.detach();
}
```

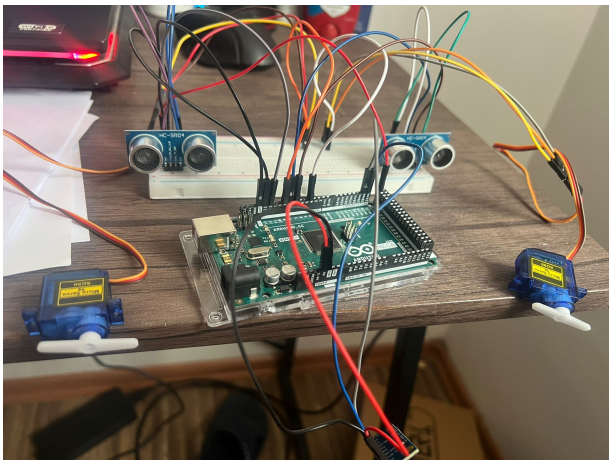

Figure 7: Servo Motors with Arduino Mega

4 DC Motors

```
void stop() {
  digitalWrite(motor00, LOW);
  digitalWrite(motor01, LOW);
  digitalWrite(motor10, LOW);
  digitalWrite(motor11, LOW);
}
void start(const int motorPin0, const int motorPin1, bool
    forward, uint8_t speed) {
  if (forward) {
    analogWrite(motorPin0, speed);
    digitalWrite(motorPin1, LOW);
  } else {
    digitalWrite(motorPin0, LOW);
    analogWrite(motorPin1, speed);
  }
}
```

5 MPU-6500

```
#include "MPU_X.h"

const int IMU_ADDR_ON_BUS = 0x68;

MPUx imu(IMU_ADDR_ON_BUS);
```

```
  Wire.begin();
  delay(200);
```

```
  imu.init();
  delay(200);
```

```
void loop() {
  imu.printMPUData();
  delay(1000);
}
```

```
void MPUx::printMPUData() {
  floatThreeVals accl = getAcclVals();
  floatThreeVals gyro = getGyroVals();
  Serial1.printf("| Accelerometer (g) >>> x: %+07.2f y:
      %+07.2f z: %+07.2f |",
                 accl.x, accl.y, accl.z);
  Serial1.println("| Gyroscope (degrees/sec) >>> x: %+07.2f
      y: %+07.2f z: %+07.2f |",
                 gyro.x, gyro.y, gyro.z);
}
```

6 Arduion Mega - ESP32 Serial Communication

```
  Serial1.begin(9600);
  while(!Serial1){
    delay(10); //wait for Serial Monitor to open
  }
```

```
void serialEvent1(){
    while (Serial.available()) {
    char inChar = (char)Serial.read();

    if (inChar != '\n') {
      inputString += inChar;
    }

    if (inChar == '\n') {
      stringComplete = true;
    }
  }
}
```

```
  Serial1.begin(9600);
  while(!Serial1){
    delay(10);
  }
```

7 Processing Received Data

```
  if(stringComplete){
    sscanf(inputString.c_str(),"| Accelerometer (g) >>> x:
        %+07.2f y: %+07.2f z: %+07.2f || Gyroscope (degrees/
        sec) >>> x: %+07.2f y: %+07.2f z: %+07.2f |",&accl.x
        ,&accl.y,&accl.z,
    &gyro.x,&gyro.y,&gyro.z);

    int angleX = map(accl.x, -1.0, 1.0, 0, 180); // Map
        accelerometer values to angles for servo motor
        movement
    int angleY = map(accl.y, -1.0, 1.0, 0, 180);

    controlServo(angleX, angleY, 2);

    if (abs(accl.x) > 0.5 || abs(accl.y) > 0.5) {
      stop(); // Stop motors on excessive tilt
    } else {
      start(motor00, motor01, true, 100); // Move forward
      start(motor10, motor11, true, 100);
    }

    if (gyro.x > 50.0 || gyro.y > 50.0) {
      controlServo(0, 0, 2);
    }
    stringComplete=false;
  }
```