DOKUZ EYLUL UNIVERSITY
ENGINEERING FACULTY

Differential Robot Construction and Position Determination

ETE 3007 FUNDAMENTAL ROBOTICS

Project Final Report

2021-2022 SPRING

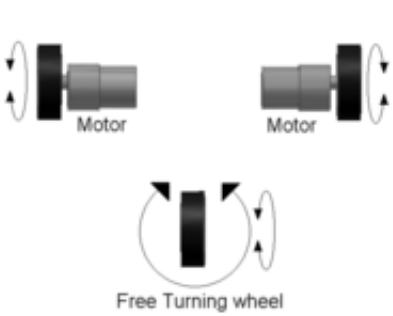
IZMIR

Contents

1. Introduction

The aim of this project is to design, develop and test a hardware system that containing differential robot construction and position determination. There are several main aspects of the project that needs to be explored in this part of the report. First, it is important to be able to use the information set that has systematic access, manageable, updatable, and defined relationships among each other, in the most efficient way. The project should be well designed so that the idea can be developed and modified, if necessary, in the future. This requires a thinking that is ahead of its time. Hence, our developer team started to design and develop differential robot based on these principles.

2. Project Description



A differential wheeled robot is a mobile robot whose movement is based on two separately driven wheels placed on either side of the robot body. It can thus change its direction by varying the relative rate of rotation of its wheels and hence does not require an additional steering motion.

The list of materials to be used in the construction of the project.

1- Arduino UNO

2- Bluetooth module

3- Receiver and Transmitter

4- Gyroscope – MPU6050

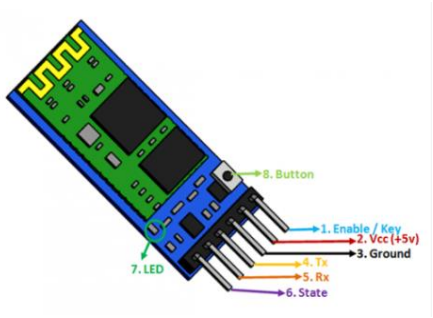
5- Encoder

The Differential robot will be controlled by remote control. While the application is running, it sends the direction information received from the user via the keyboard to the robot via Bluetooth. The sensor and movement information during the movement will be directly transferred to a computer via Bluetooth. Incoming data, which are speed and angle of rotation, will be processed with a python script and the current direction information and movement traces of the vehicle will be output to a screen.

Materials

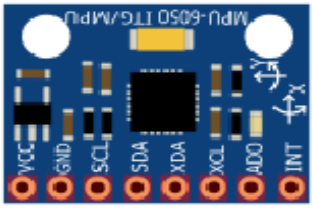
HC05

HC-05 Bluetooth hardware is made for use of Bluetooth SSP (Serial Port Standard) and wireless serial applications. This board allows Bluetooth to be done from 2.4GHz application. It has a device from 10 meters in open space.



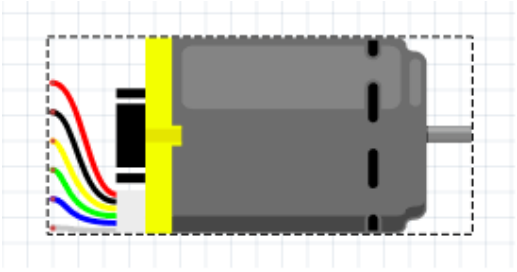
MPU6050

MPU6050 is IMU sensor with 3-axis gyro and 3-axis angular accelerometer. This sensor, which gives a total of 6 outputs, uses the I2C protocol for communication. IMU sensors such as the MPU 6050 are used in most electronic devices. This sensor is in many parts of our lives. It is used in smart watches, Fitbit bands, Robots, Drone, Gimbal, and smart phones.



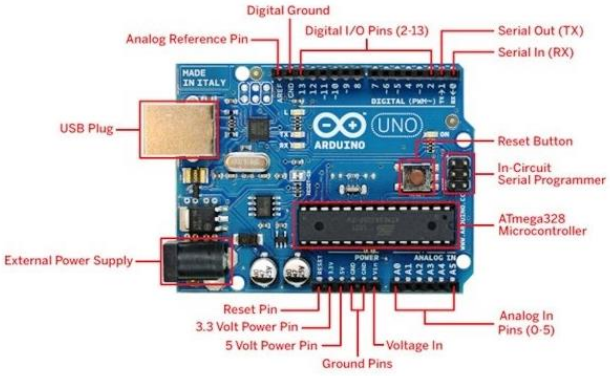
Encoder

The encoder converts the angular position, movement, axis of a shaft into an analog or digital code. It is a sensing device that monitors the current positions of the connected shaft and provides feedback. Generally, there are many usage areas such as servo motor, robot, moving camera, CNC benches, automation.



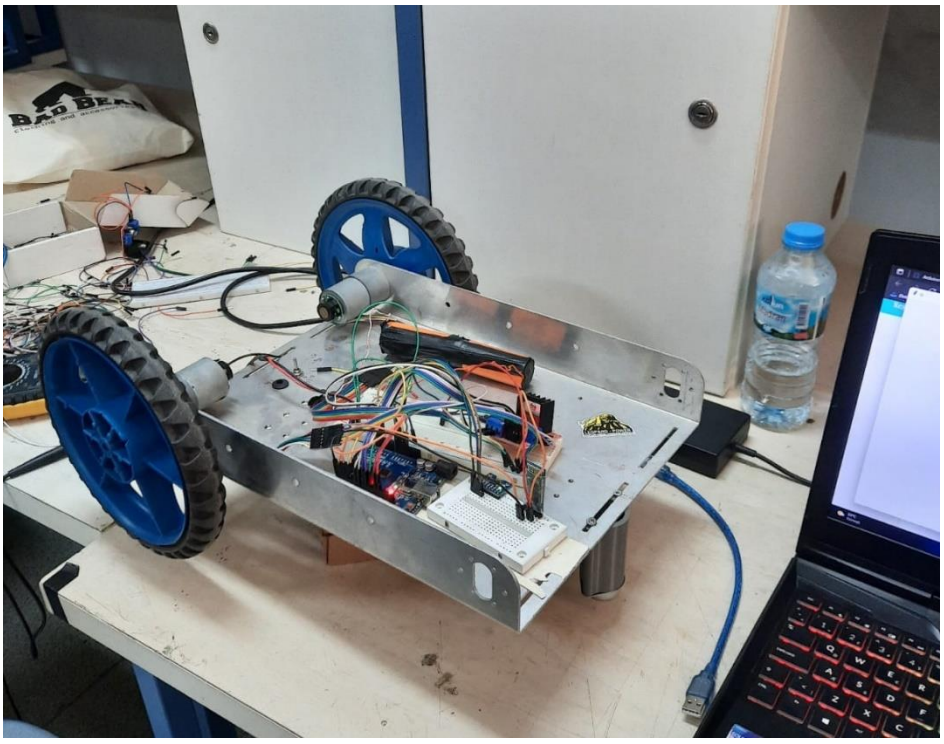
Arduino UNO

Arduino's usage areas can be listed as robot programming, smart home, smart projects, automation, 3D printer. The reason why it is used in these areas is that it is easily integrated and a started project can be revealed quickly.



Differential Drive Robot

General view of our differential robot that we created using all these materials.



3. Progress Summary

3.1 Current Status

The hardware part of the robot has been successfully completed. On the other hand, the necessary software coding was done through Tkinter, a python desktop application library. Thanks to this library, we visualized the route followed by the vehicle after processing the data. While the application was running, the direction information received from the user was transmitted to the vehicle as a command. The speed information of the vehicle was calculated by the encoders on the motors.

3.2 Team Responsibilities

Murat DUMAN deals with the hardware part of the project. After fixing the physical connections of the differential robot, each of the team members must deal with the differential robot kinematics. In the studies to be carried out on the motion algorithm, he will work on the software as well as the hardware part.

Süheyla UYGUR is interested in the desktop app of the project. She is working on Tkinter and python. She is working on active role in the processing of incoming location and motion data.

Osman MUTLU works on the hardware and software parts of the project. He will work on algorithmic solutions for processing motion sensors and creating motion traces accordingly. He also works on the control algorithm of the differential robot.

Beyza GÜR works on the creating motion tracks of the project. She will generate algorithmic solutions to process motion sensors and create motion tracks accordingly. She will work on determining the position of the robot thanks to the gyro and compass sensor.

3.3 Problem Encountered

1- Deviation of angle data calculated using Gyro and Acc.

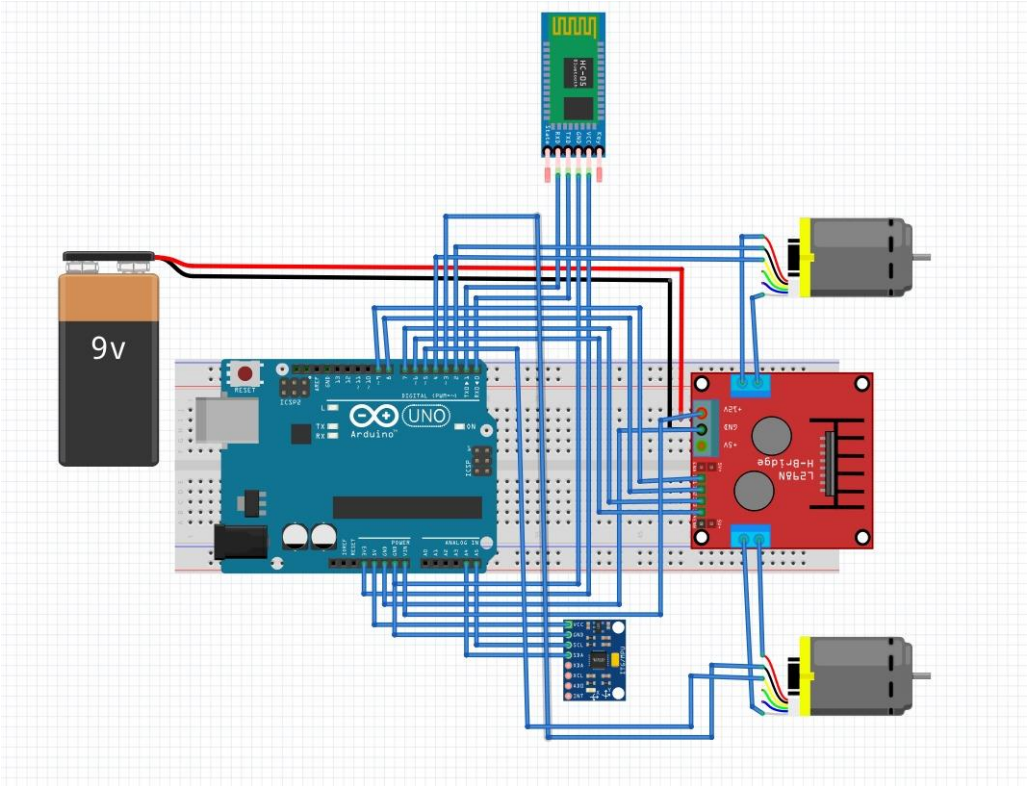
2- Bluetooth connection problems.

3- The intermittent hang problem of the program coded with python.

4- Disconnection of the IMU sensor.

3.4 Algorithm and Problem Solutions

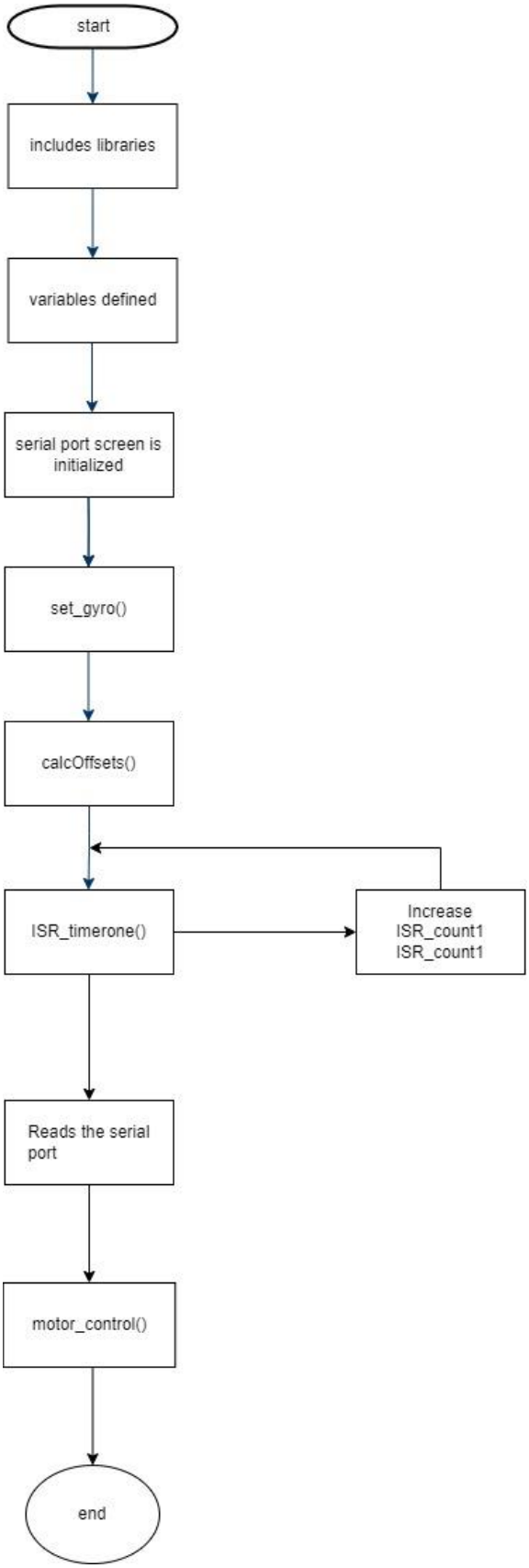
Flowchart and Circuit Design



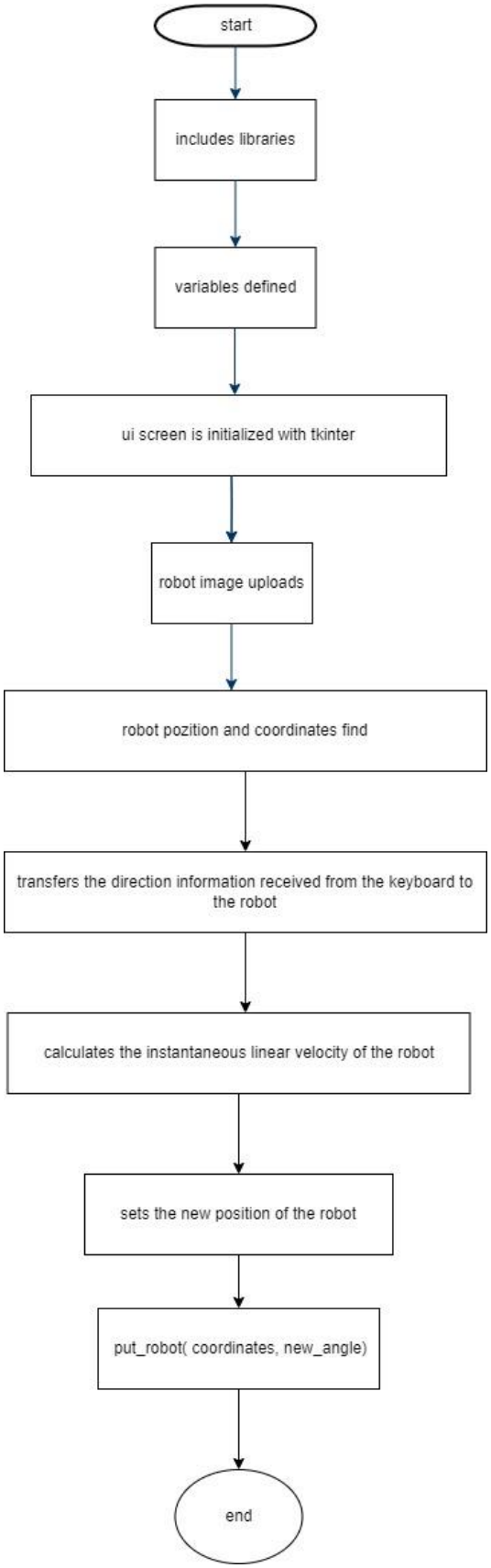
Solution

Arduino and Python were used in accordance with the purpose of the project. A Bluetooth module was purchased to enable the computer and Arduino to communicate. The serial library of python was used for serial port communication. The MPU6050 used in the construction of the robot was used to find the angle of the vehicle. The angle of the vehicle was calculated with gyro and acc using *MPU6050 light* library. With the *read_data()* function, the data from the serial port was transferred into data. The new position of the vehicle is calculated with the *get_new_coordinate()* function. With *set_position()*, the *put_robot()* function is called at certain intervals and the vehicle is placed in its new position.

Arduino Flowchart



Python Flowchart



Python Code

```
import math
import tkinter as tk
from tkinter import *
from PIL import Image, ImageTk
import PIL
import serial
import time

start_angle = 0

# It is the starting coordinate of the robot.
start_coordinates = [600, 300]
# "COM5" is the communication port between the robot and the computer.
arduino = serial.Serial(port='COM5', baudrate=9600, timeout=.1)


img = Image.open("image.png")
root = tk.Tk()
root.geometry("1250x700")
w = 1250
h = 700
canvas = tk.Canvas(master=root, width=w, height=h)
canvas.pack()
canvas.images = []
data = {}
frame_rate = 100
radius = 0.2 # radius of wheel is 20 cm
motion =""

# Calculates the instantaneous linear velocity of the robot.
def get_v():
    global motion
    if motion == "F" or motion == "B":
        v0 = (radius*2.0*math.pi*float(data["L"]))/60.0
        v1 = (radius*2.0*math.pi*float(data["R"]))/60.0
        return (v0+v1)/2
    else:
        return 0

# Calculates the new coordinate with instantaneous velocity and angle.
def get_new_coordinate():
    global canvas,img
    r = 700*get_v()

    y = -math.sin(start_angle*0.017453292519943295)*r
    x = math.cos(start_angle*0.017453292519943295)*r
    if motion == "B":
        x = -x
        y = -y

    canvas.create_line(start_coordinates[0], start_coordinates[1],
                       (start_coordinates[0]+x),
start_coordinates[1]+y)
    start_coordinates[0] += x
    start_coordinates[1] += y
    canvas.after(frame_rate,get_new_coordinate)
```

```
# It sends data to the serial port of the Arduino.
def write_data(x):
    arduino.write(bytes(x, 'utf-8'))

# It organizes the data coming from the communication port and saves it
in the data.
def read_data():
    global canvas,data
    value = arduino.readline()
    print(value)
    value = str(value).replace('b', '')
    value = value.replace("'", '')
    if value != "":
        value = value.strip()

        value = value.split()
        for temp in value:
            t = temp.split(":")
            data[t[0]] = t[1]

    canvas.after(10,read_data)

# Sets the instant position of the robot.
def set_pozition():
    global canvas,img
    put_robot(canvas, img, start_coordinates, data["Z"])
    canvas.after(frame_rate,set_pozition)

# The motion functions of the robot.
def front():
    global motion
    write_data("0")
    motion = "F"

def back():
    global motion
    write_data("1")
    motion = "B"

def left():
    global motion
    write_data("2")
    motion = "L"

def right():
    global motion
    write_data("3")
    motion = "R"

def stop():
    global motion
    write_data("4")
    motion = "S"

# It places the robot in that position according to the calculated
coordinate and angle values.
def put_robot(canvas, img, coordinates, new_angle):
```

```
    global start_angle

    start_angle = int(new_angle)
    if start_angle > 180:
        start_angle -= 360
    elif start_angle < -180:
       start_angle += 360
    canvas.delete("image")
    img = img.rotate(start_angle, PIL.Image.NEAREST, expand=1)
    image = ImageTk.PhotoImage(img)
    canvas.images += [image]
    canvas.create_image(
        coordinates,
        image=image,
        tag="image"
    )


def main():
    global img,root,canvas
    img = img.rotate(-90, PIL.Image.NEAREST, expand=1)
    image = ImageTk.PhotoImage(img)
    canvas.images.append([image])
    canvas.create_image(
        (50, 50),
        image=image,
        tag="image"
    )
    time.sleep(5)
    read_data()
    get_new_coordinate()
    set_pozition()


    # It transfers the direction information received from the keyboard
to the robot.
    root.bind_all("<KeyPress-Left>", lambda event: left())
    root.bind_all("<KeyRelease-Left>", lambda event: stop())

    root.bind_all("<KeyPress-Right>", lambda event: right())
    root.bind_all("<KeyRelease-Right>", lambda event: stop())

    root.bind_all("<KeyPress-Up>", lambda event: front())
    root.bind_all("<KeyRelease-Up>", lambda event: stop())

    root.bind_all("<KeyPress-Down>", lambda event: back())
    root.bind_all("<KeyRelease-Down>", lambda event: stop())
    root.mainloop()


if __name__ == '__main__':
    main()
```

Arduino Code

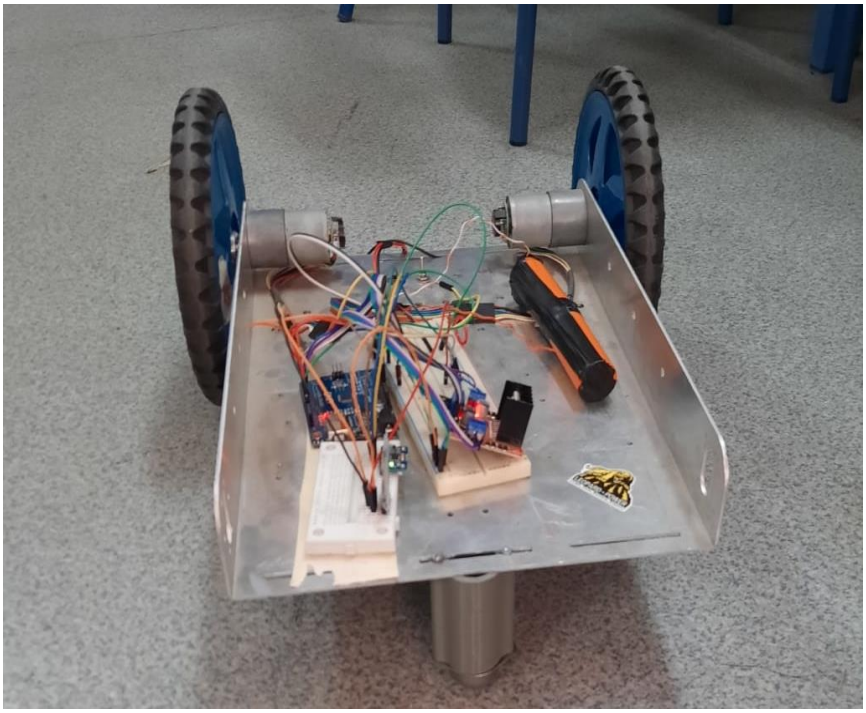
```
// Include the TimerOne Library from Paul Stoffregen
```

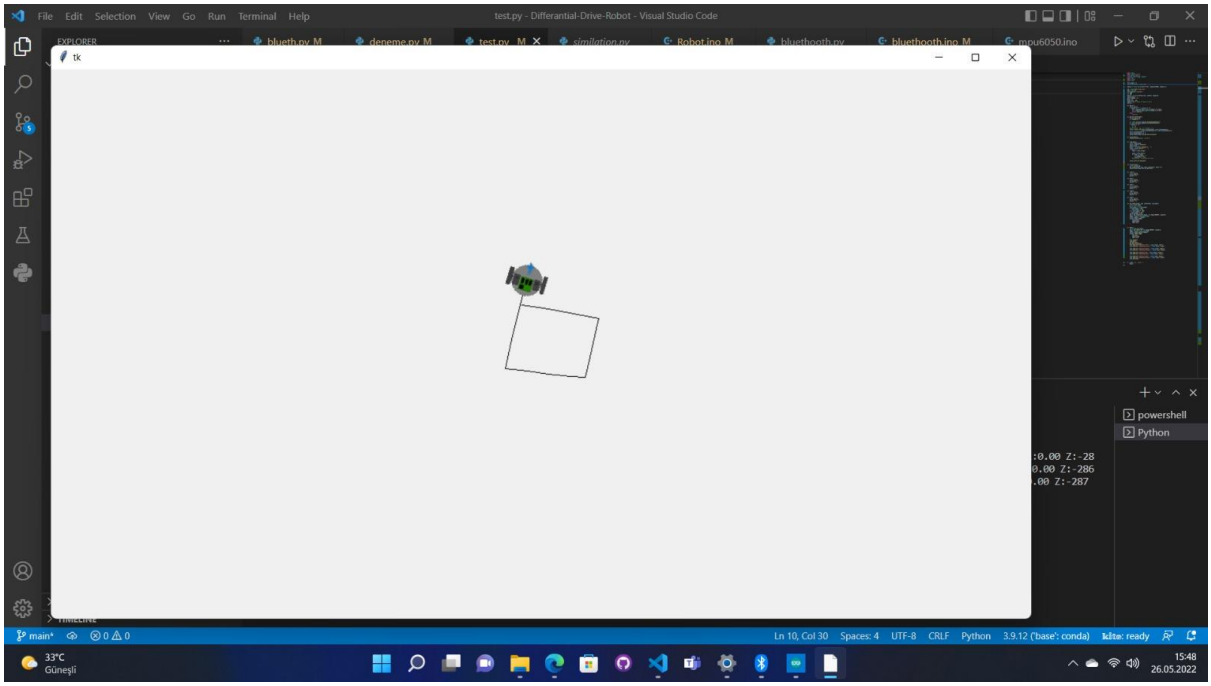
```
#include "TimerOne.h"
#include "Wire.h"
#include <MPU6050_light.h>
// MPU6050 Setup
MPU6050 mpu(Wire);
// Left Motor (A)
int enA = 10;
int in1 = 9;
int in2 = 8;
// Right Motor (B)
int enB = 5;
int in3 = 7;
int in4 = 6;

// Constants for Interrupt Pins
// Change values if not using Arduino Uno

const byte MOTOR1 = 2;  // Motor 1 Interrupt Pin - INT 0 MOTOR RIGHT
const byte MOTOR2 = 3;  // Motor 2 Interrupt Pin - INT 1 MOTOR LEFT

// Integers for pulse counters
unsigned int counter1 = 0;
unsigned int counter2 = 0;
int x = 4;
// Float for number of slots in encoder disk
const float diskslots = 2500.00;  // Change to match value of encoder
disk
const float wheeldiameter = 20.0; // Wheel diameter in cm
float wheelbase = 30.0; // in cm
// Interrupt Service Routines

// Motor 1 pulse count ISR
void ISR_count1()
{
  counter1++;  // increment Motor 1 counter value
}

// Motor 2 pulse count ISR
void ISR_count2()
{
  counter2++;  // increment Motor 2 counter value
}

// TimerOne ISR
void ISR_timerone()
{
  Timer1.detachInterrupt();  // Stop the timer
  //Serial.print("Right Motor Speed: ");
  float rotation1 = (counter1 / diskslots) * 60.00;  // calculate RPM
for Motor 1
  Serial.print(" R:");
  Serial.print(rotation1);

  counter1 = 0;  //  reset counter to zero
  //Serial.print("Left Motor Speed: ");
  float rotation2 = (counter2 / diskslots) * 60.00;  // calculate RPM
for Motor 2
  Serial.print(" L:");
```

```
  Serial.print(rotation2);

  int new_angle = mpu.getAngleZ();
  Serial.print(" Z:");
  Serial.print(new_angle);
  counter2 = 0;  //  reset counter to zero
  Timer1.attachInterrupt( ISR_timerone );  // Enable the timer
}

void setup()
{
  Serial.begin(9600);
  Serial.setTimeout(1);
  Wire.begin();
  set_gyro();
  Timer1.initialize(100000); // set timer for 1sec
  attachInterrupt(digitalPinToInterrupt (MOTOR1), ISR_count1, RISING);
// Increase counter 1 when speed sensor pin goes High
  attachInterrupt(digitalPinToInterrupt (MOTOR2), ISR_count2, RISING);
// Increase counter 2 when speed sensor pin goes High
  Timer1.attachInterrupt( ISR_timerone ); // Enable the timer
}

void loop()
{
  mpu.update();
  if (Serial.available()) {
    x = Serial.readString().toInt();
  }
  motor_control();
}
void motor_control(){
  if (x == 0) {
      front();
    }
    if (x == 1) {
      back();
    }
    if (x == 2) {
      left();
    }
    if (x == 3) {
      right();
    }
    if (x == 4) {
      stop_();
    }
}
void front() {
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  analogWrite(enA, 255);
  // Set Motor B forward
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
  analogWrite(enB, 255);
}
void back() {
```

```
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
  analogWrite(enA, 255);
  // Set Motor B forward
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);
  analogWrite(enB, 255);
}
void stop_() {
  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);
  analogWrite(enA, 255);
  // Set Motor B forward
  digitalWrite(in3, LOW);
  digitalWrite(in4, LOW);
  analogWrite(enB, 255);
}
void right() {
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  analogWrite(enA, 255);
  // Set Motor B forward
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);
  analogWrite(enB, 255);
}
void left() {
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
  analogWrite(enA, 255);
  // Set Motor B forward
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
  analogWrite(enB, 255);
}
void set_gyro(){
  byte status = mpu.begin();
  mpu.calcOffsets(); // gyro and accelero
}
void set_gyro_print() {
  byte status = mpu.begin();
  Serial.print(F("MPU6050 status: "));
  Serial.println(status);
  while (status != 0) { } // stop everything if could not connect to
MPU6050
  Serial.println(F("Calculating offsets, do not move MPU6050"));
  delay(1000);
  mpu.calcOffsets(); // gyro and accelero
  Serial.println("Done!\n");
}
```

Scenario - Have the robot draw a square

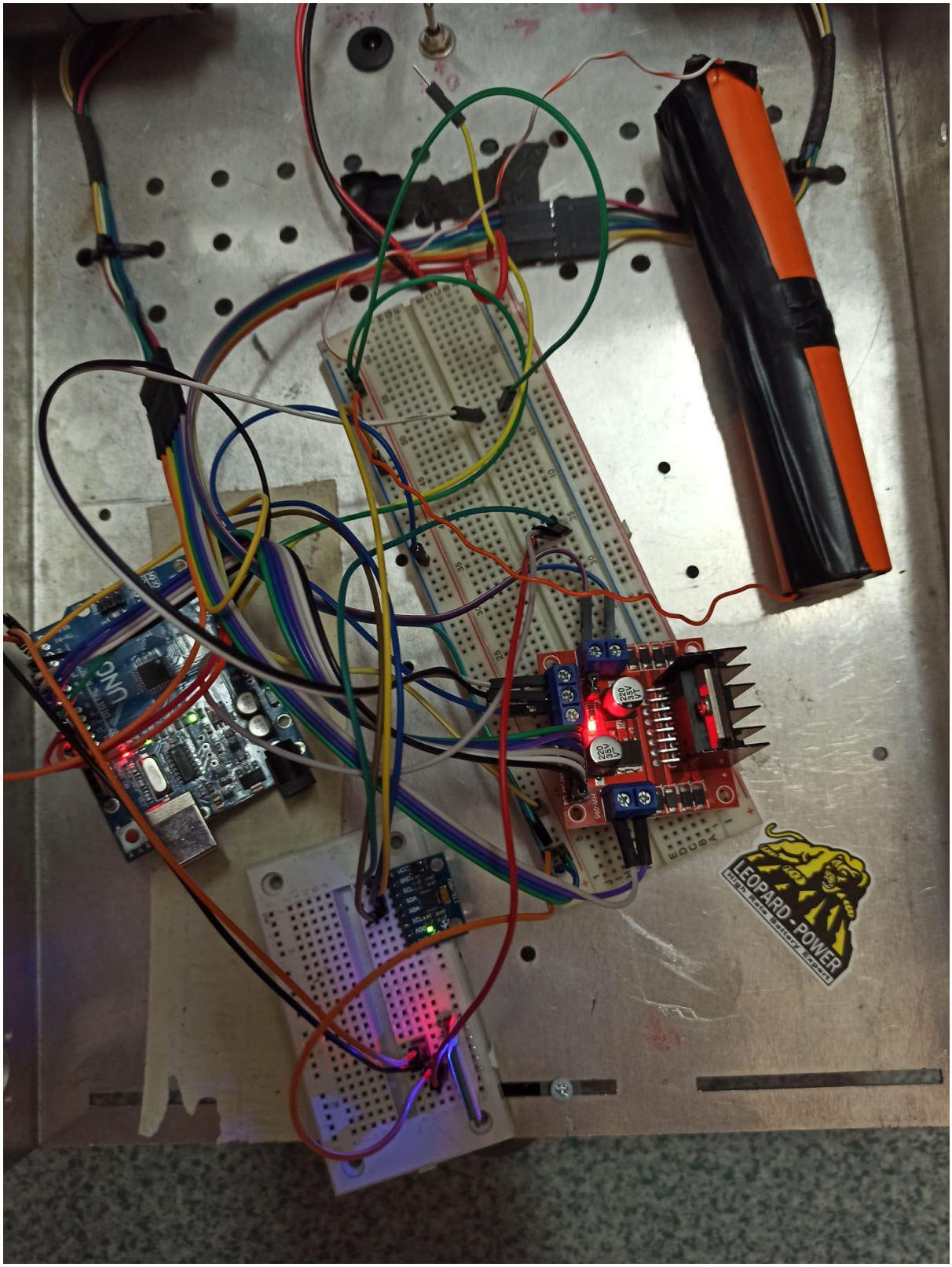


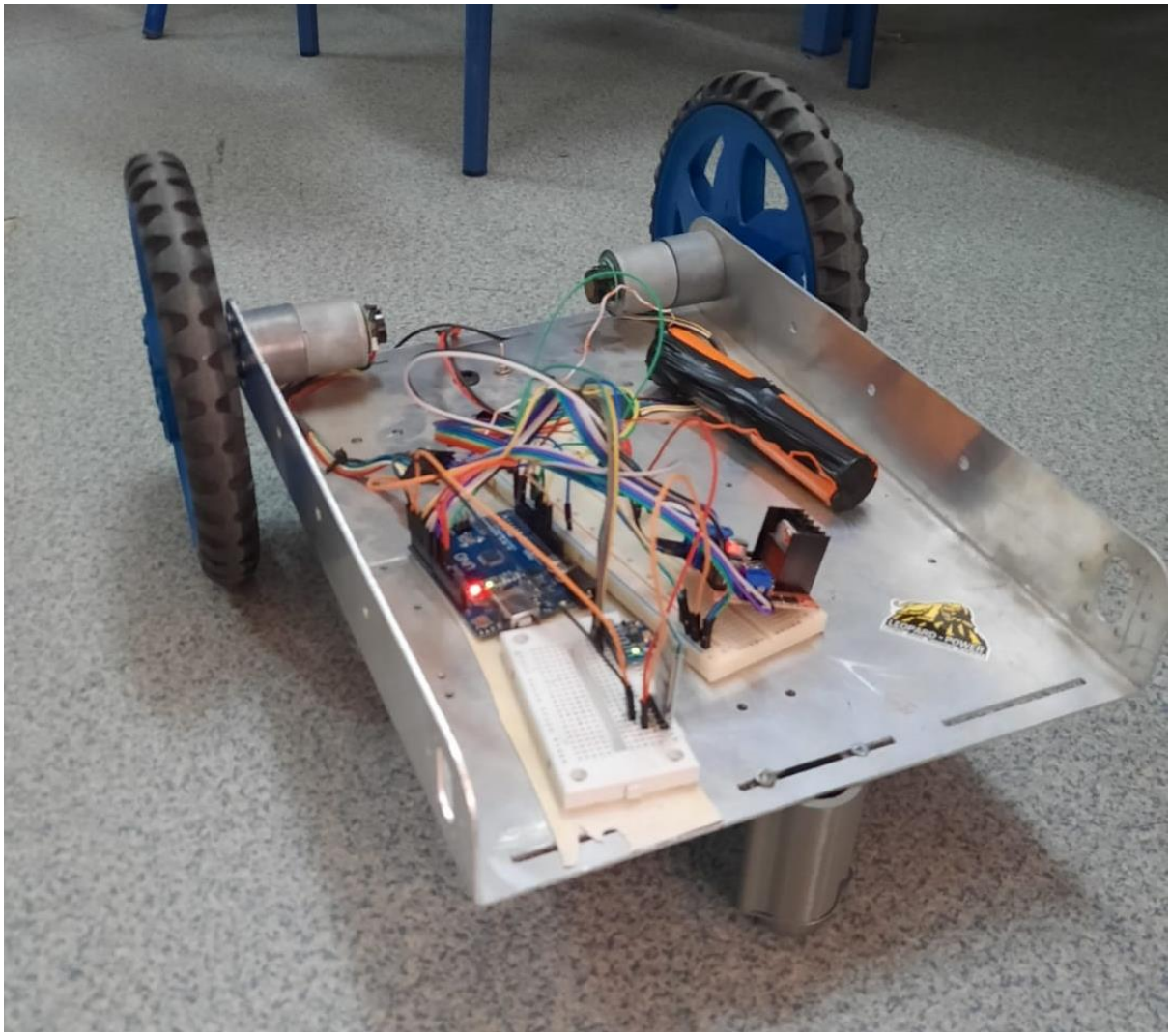


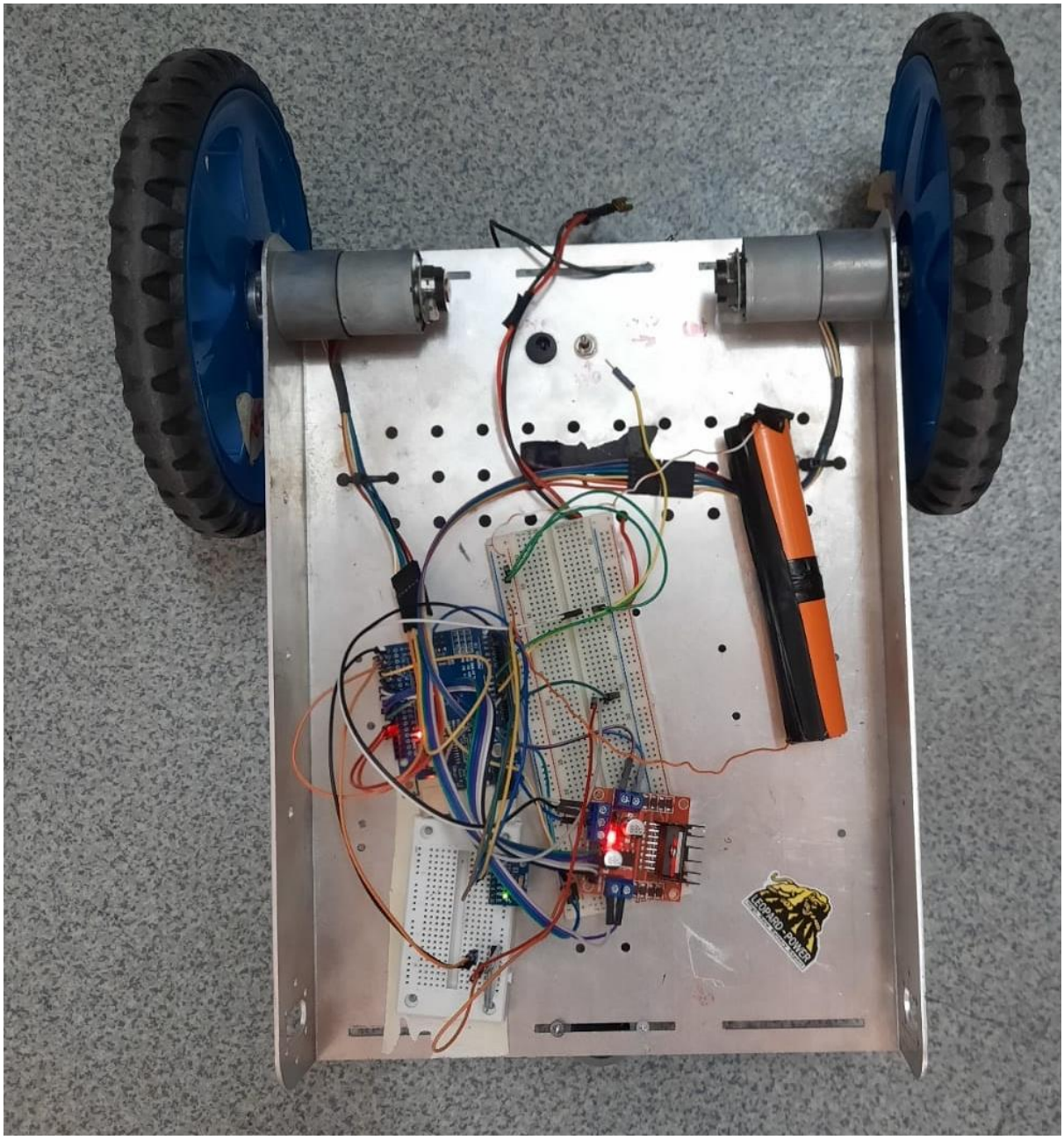
4. Conclusion

We were able to find the instant location and direction information of the vehicle with high accuracy. With the desktop application we designed, we were able to find an algorithm that can visualize in high resolution. We used the resources we had in the most efficient way. Although we used cheap sensors, we were able to solve the problem in the most effective way. At the end of the project, we were able to visualize location information and motion traces with high accuracy.

5. Appendix






LEOPARD - POWER

6. References

[1] www.cs.columbia.edu/~allen/F17/NOTES/icckinematics.pdf

[2] https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf

[3] https://www.botnroll.com/img/cms/HMC5883L-3-Axis-Arduino-test-Example.pdf

[4]https://github.com/e-Gizmo/QMC5883L-GY-271-Compass-module/blob/master/QMC5883L%20Datasheet%201.0%20.pdf

[5] https://docs.arduino.cc/learn/communication/wire

[6]https://github.com/Osman-MUTLU/QuadCopter-Flight-Controller/blob/main/Receiver_Read_PWM.ino

[7]https://github.com/Osman-MUTLU/QuadCopter-Flight-Controller/blob/main/MPU6050_Reading_Gyros.ino

[8] Arduino ile Bluetooth Kontrollü Araba Yapımı (robotistan.com)

[9] https://github.com/LessonStudio/Arduino_Bluetooth/blob/master/Arduino_bluetooth.ino

[10] https://create.arduino.cc/projecthub/ansh2919/serial-communication-between-python-and-arduino-e7cce0