BRAC UNIVERSITY

Inspiring Excellence

CSE 461 : Introduction to Robotics

Hardware Project Report

Project Title : Motion-Based Mapping Robot

Group Information

Lab Section: 2

Group Number: 7

Group Members:

Name	Id
Zarin Tasnim Haider	21301380
Mezbha Ul Haq Fahim	21301243
Humayera Tabassum Tunan	22299539
Faraz Sikder Shafin	21101171
Zubaida Hossain Ramisha	23141028

Introduction

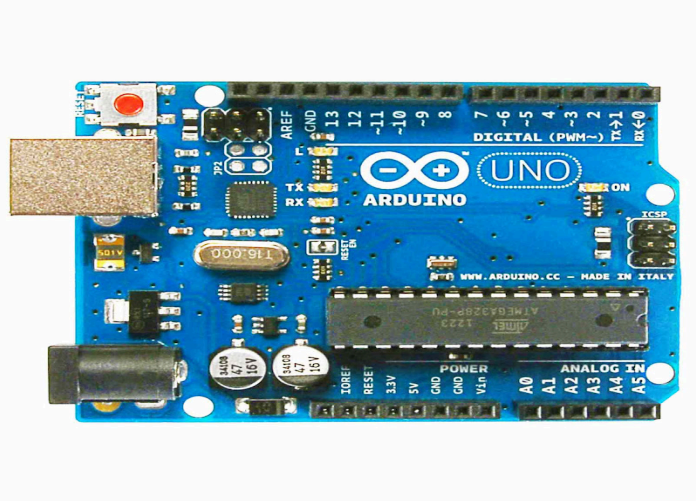
When there is a discussion about robotics and autonomous systems, the quest for enhancing capabilities and pushing the boundaries of innovation continues. The mentioned project aims to develop an Arduino-powered Bot car with the ability to autonomously map its surrounding environment.

The primary goal of this project is to construct a robot chassis equipped with fundamental components such as Arduino, motors, wheels, and a strategically positioned camera. The esp32 cam integrated with the Arduino system is used to capture real time footage and tries to map according to the motion.

This project tries to achieve the function of autonomous exploration along with self-navigation and mapping. Even though the challenges like integrating the camera module with Arduino and developing efficient navigation algorithms are present, the project aims to solve these using diligent planning and innovative problem-solving.

Equipments:

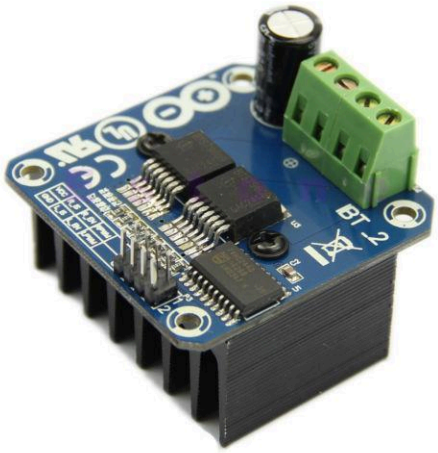
1. Arduino Uno



Arduino Uno is a microcontroller board based on the ATmega328. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It features the Atmega16U2

programmed as a USB-to-serial converter. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

2. IBT2 Motor Driver *2



This driver uses two high current half bridge Infineon BTS 7960 chip for motor drive applications. Interfacing to a microcontroller is made easy using this driver which features current sensing, slew rate adjustment and protection against high temperature, overvoltage, under voltage, overcurrent and short circuit. This small size driver provides a cost optimized solution for protected high current PWM motor drives.

3. Chassis and Frame:

For the chassis and frame, we have decided to choose an eco-friendly option as we are recycling and kitchen chopping board as the primary base for this robot. This will help us eliminate the use of plastic based products and help us recycle the things we already have at our disposal. Also, for the stands of the camera, we have also recycled a used bottle for efficiency.

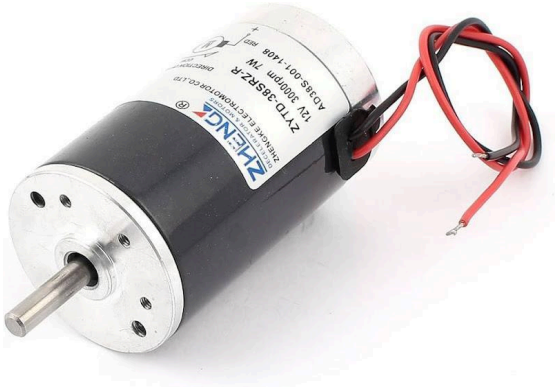
4. ESP32 Cam:

The ESP32 CAM WiFi Module Bluetooth with OV2640 Camera Module 2MP For Environment Recognition has a very competitive small-size camera module that can operate independently as a minimum system with a footprint of only 40 x 27 mm; a deep sleep current of up to 6mA and is widely used in various IoT applications. This module adopts a DIP package and can be directly inserted into the backplane to realize rapid production of products, providing customers with high-reliability connection mode, which is convenient for application in various IoT hardware terminals.

5. SG90 Servo Motor:

The Mini Servo SG90 is a small, lightweight, and low-cost servo motor used in hobbyist and educational projects. Servo motors are a type of motor that can be controlled with a precise amount of rotation, and the SG90 servo is specifically designed for small applications, such as RC models, toys, or robotic projects. The SG90 servo features a compact size, high-speed response, and easy interface with microcontroller boards, such as the Arduino or Raspberry Pi. It is often used for controlling simple motions, such as positioning a servo horn, moving a robotic arm, or rotating a wheel.

6. 3000 rpm high torque motor *4

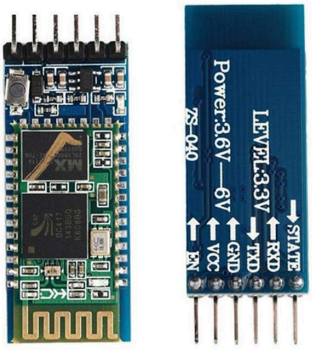


A 3000 RPM high-torque motor is the powerhouse for our Arduino-powered bot car. This motor offers exceptional torque for efficient movement. When paired with Arduino's control capabilities, it provides precise control and responsiveness to navigate the bot car with ease. This motor can deliver the power and torque needed for optimal performance for our project.

7. Jumper wires

Jumper wires are used for connecting components in Arduino projects to build circuits.

8. HC-05 Bluetooth Module

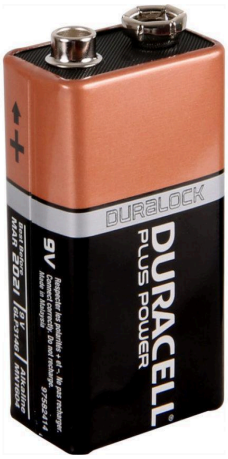


The HC-05 Bluetooth module is a versatile wireless communication component compatible with Arduino projects. It has an easy-to-use interface. Also, it helps to provide seamless integration of Bluetooth capabilities into various applications. It is easy to establish wireless serial communication between Arduino and other devices or enable remote control via smartphone apps using this HC-05 module.

9. 4200 mah lipo 11.5v

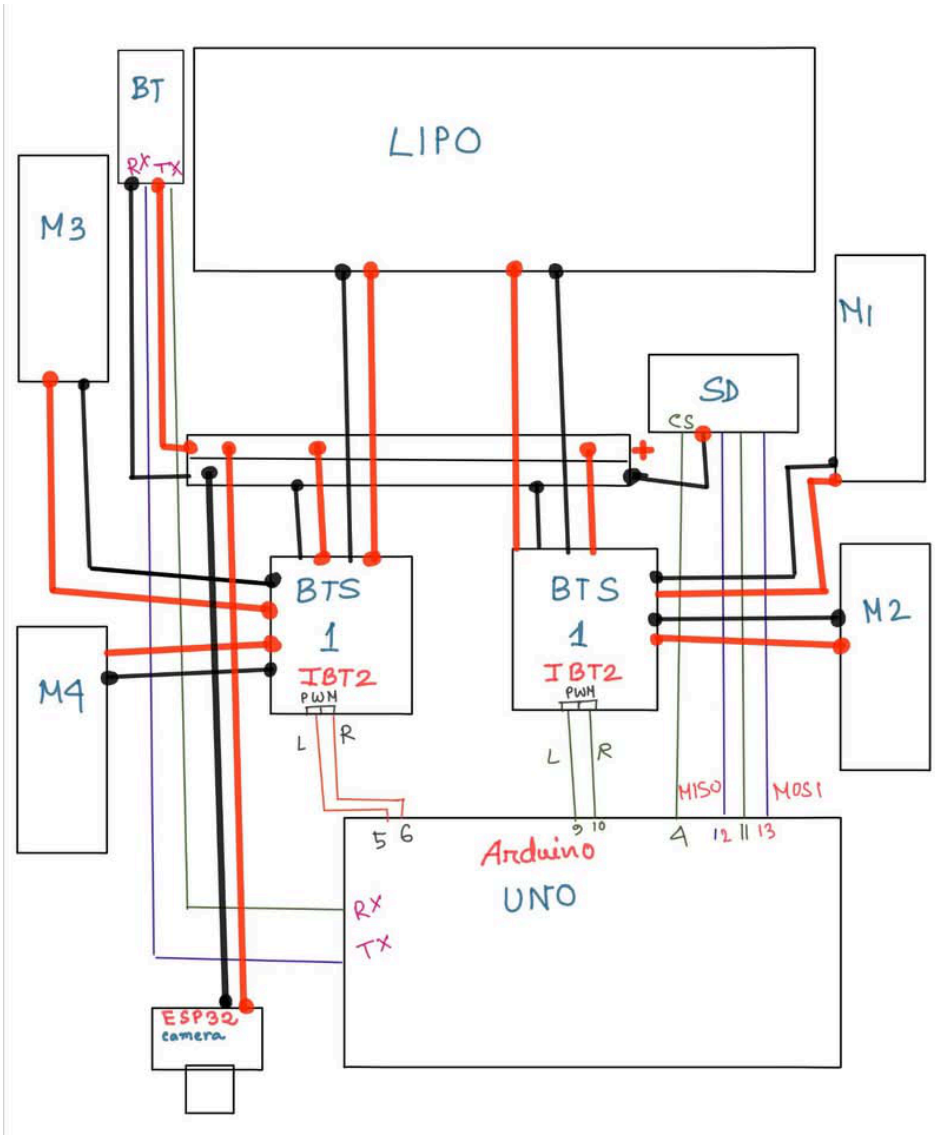
The 4200mAh LiPo battery provides ample power with an output of 11.5V, ideal for sustaining extended operation in high-demand applications. Its high capacity ensures prolonged usage, making it suitable for powering Arduino projects, RC bots, drones, and other electronic devices.

10. 9V battery

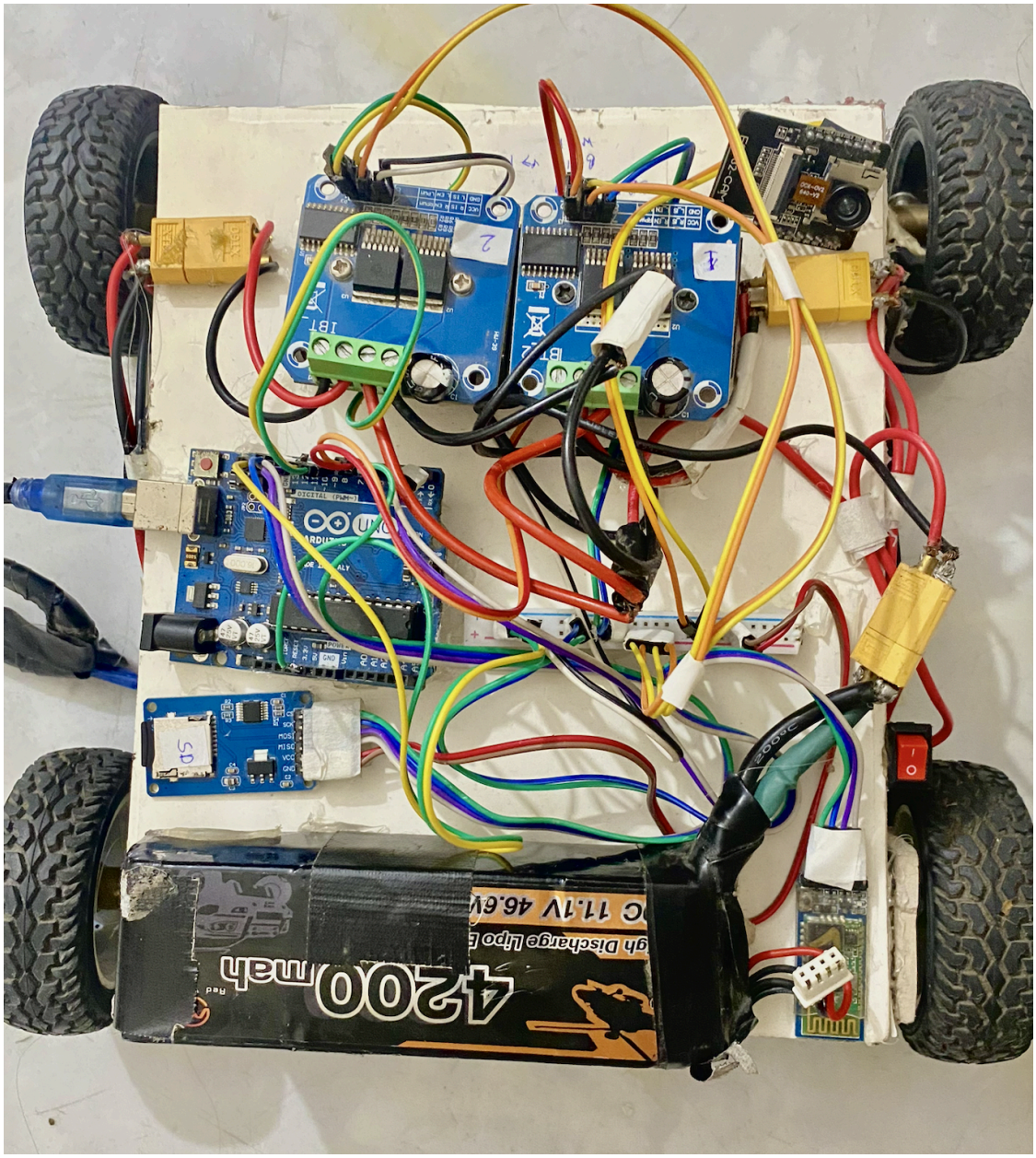


The 9V battery is a compact and portable power source. It is commonly used in small electronics and Arduino projects. It provides convenient and reliable energy for various applications.

Circuit Diagram:



Experiment Setup:



Working Details:

In the above picture,the BTS IBT2 motor driver is used because it's not possible to drive a 12V motor with a 5V input, so we utilize the 12V output to drive the motor. The Arduino sends commands to the BTS, which functions as an output to control the motor. The internal switch (transistor) in the BTS turns the motor on and off as needed.

For this project, we have used an ESP32 with a universal code library installed, specifically designed for gameservers. The ready code is uploaded to the ESP32 using an FTDI component. The setup also includes an Arduino UNO, an SD card reader for output 2, a LIPO battery, and an HC05 module for input.

Commands are sent from a mobile device to the Arduino using the HC05 Bluetooth module. The mobile software, "BT RC Controller," facilitates this communication. For example, the "F" command is sent for forward movement, mapped as approximately 20 inches forward, while the "B" command is sent for backward movement. These commands run in a loop where functions are performed sequentially, and each function is called serially.

The SD card plays a crucial role in storing new commands received from Bluetooth. The commands are copied and pasted into a file named "Path.txt" on the SD card, stored serially. The mapping of commands includes "H" for a 90° left turn and "I" for a 90° right turn, ensuring precise control over the robot's movements and same goes for other commands.

Code 1: t_car.ino

```
#include <SD.h>
#include <SPI.h>

const int chipSelect = 4;
const int speed = 255;
const int curving_speed = 20;
String bt_sig;
const int B1_R_PWM = 5;
const int B1_L_PWM = 6;
const int B2_R_PWM = 9;
const int B2_L_PWM = 10;

void setup() {
   Serial.begin(9600);
   Serial.print("Initializing SD card...");
   SPI.begin();
   SPI.setClockDivider(SPI_CLOCK_DIV8);
   pinMode(SS, OUTPUT);
   if (!SD.begin(chipSelect)) {
      Serial.println("SD card initialization failed!");
      Serial.println("Things to check:");
      Serial.println("* is a card inserted?");
      Serial.println("* is your wiring correct?");
      Serial.println("* did you change the chipSelect pin to match your shield or module?");
      Serial.println("Note: press reset or reopen this serial monitor after fixing your issue!");
      while (1);
   }
   Serial.println("SD card initialization done.");
   if (SD.exists("path.txt")) {
      SD.remove("path.txt");
      Serial.println("Existing path.txt removed");
   }
   File dataFile = SD.open("path.txt", FILE_WRITE);
   if (dataFile) {
      dataFile.println("Movement Log Start");
      dataFile.close();
      Serial.println("New path.txt created");
   }
   pinMode(B1_R_PWM, OUTPUT);
   pinMode(B1_L_PWM, OUTPUT);
   pinMode(B2_R_PWM, OUTPUT);
   pinMode(B2_L_PWM, OUTPUT);
}

void logMovement(const char* movement) {
```

```
    File dataFile = SD.open("path.txt", FILE_WRITE);
    if (dataFile) {
        dataFile.println(movement);
        dataFile.close();
    } else {
        Serial.println("Error opening path.txt");
    }
}

void loop() {
    bt_sig = "";
    while(Serial.available()) {
        bt_sig = (char)Serial.read();
    }

    if(bt_sig == "F") {
        forward();
        Serial.println("move forward");
        logMovement("forward");
    }
    else if(bt_sig == "L") {
        turn_left();
        Serial.println("move left");
        logMovement("left");
    }
    else if(bt_sig == "R") {
        turn_right();
        Serial.println("move right");
        logMovement("right");
    }
    else if(bt_sig == "B") {
        backward();
        Serial.println("move back");
        logMovement("backward");
    }
    else if (bt_sig == "S") {
        stop_bot();
        Serial.println("stop");
    }
    else if (bt_sig == "G") {
        Left_forward();
        Serial.println("Fwd left");
    }
    else if (bt_sig == "I") {
        right_forward();
        Serial.println("Fwd Right");
    }
    else if (bt_sig == "H") {
        left_backward();
        Serial.println("Back left");
    }
```

```
    else if (bt_sig == "J") {
      right_backward();
      Serial.println("Back right");
    }
    else if(bt_sig == "X") {
      kachra_dance();
      delay(600);
      stop_bot();
      kachra_dance_2();
      Serial.println("Kachra dance");
    }
}

void stop_bot() {
    analogWrite(B1_R_PWM, 0);
    analogWrite(B1_L_PWM, 0);
    analogWrite(B2_R_PWM, 0);
    analogWrite(B2_L_PWM, 0);
}

void forward() {
    analogWrite(B1_R_PWM, speed);
    analogWrite(B1_L_PWM, 0);
    analogWrite(B2_R_PWM, speed);
    analogWrite(B2_L_PWM, 0);
}

void backward() {
    analogWrite(B1_R_PWM, 0);
    analogWrite(B1_L_PWM, speed);
    analogWrite(B2_R_PWM, 0);
    analogWrite(B2_L_PWM, speed);
}

void turn_left() {
    analogWrite(B1_R_PWM, 0);
    analogWrite(B1_L_PWM, speed);
    analogWrite(B2_R_PWM, speed);
    analogWrite(B2_L_PWM, 0);
}

void turn_right() {
    analogWrite(B1_R_PWM, speed);
    analogWrite(B1_L_PWM, 0);
    analogWrite(B2_R_PWM, 0);
    analogWrite(B2_L_PWM, speed);
}

void Left_forward() {
    analogWrite(B1_R_PWM, curving_speed);
    analogWrite(B1_L_PWM, 0);
```

```
  analogWrite(B2_R_PWM, speed);
  analogWrite(B2_L_PWM, 0);
}

void right_forward() {
  analogWrite(B1_R_PWM, speed);
  analogWrite(B1_L_PWM, 0);
  analogWrite(B2_R_PWM, curving_speed);
  analogWrite(B2_L_PWM, 0);
}

void left_backward() {
  analogWrite(B1_R_PWM, 0);
  analogWrite(B1_L_PWM, curving_speed);
  analogWrite(B2_R_PWM, 0);
  analogWrite(B2_L_PWM, speed);
}

void right_backward() {
  analogWrite(B1_R_PWM, 0);
  analogWrite(B1_L_PWM, speed);
  analogWrite(B2_R_PWM, 0);
  analogWrite(B2_L_PWM, curving_speed);
}

void kachra_dance() {
  analogWrite(B1_R_PWM, 0);
  analogWrite(B1_L_PWM, speed);
  analogWrite(B2_R_PWM, 0);
  analogWrite(B2_L_PWM, 75);
}

void kachra_dance_2() {
  analogWrite(B1_R_PWM, speed);
  analogWrite(B1_L_PWM, 0);
  analogWrite(B2_R_PWM, 0);
  analogWrite(B2_L_PWM, 65);
}
```

Code 2: t_plot.py

```
import matplotlib.pyplot as plt
import numpy as np


def read_log_file(filename='path.txt'):
```

```
    with open(filename, 'r') as file:
        lines = file.readlines()
        if 'Movement Log Start' in lines[0]:
            movements = [line.strip() for line in lines[1:]]
        else:
            movements = [line.strip() for line in lines]
    return movements


def calculate_trajectory(movements):
    x = 10
    y = 10
    angle = 0
    step = 1
    RIGHT_ANGLE_PER_COMMAND = 90/8
    LEFT_ANGLE_PER_COMMAND = 90/8
    x_coords = [x]
    y_coords = [y]

    for move in movements:
        if move == 'forward':
            x += step * np.cos(np.radians(angle))
            y += step * np.sin(np.radians(angle))
        elif move == 'backward':
            x -= step * np.cos(np.radians(angle))
            y -= step * np.sin(np.radians(angle))
        elif move == 'left':
            angle += LEFT_ANGLE_PER_COMMAND
        elif move == 'right':
            angle -= RIGHT_ANGLE_PER_COMMAND

        x_coords.append(x)
        y_coords.append(y)

    return x_coords, y_coords


def plot_trajectory(x_coords, y_coords):
    plt.figure(figsize=(10, 10))
    plt.plot(x_coords, y_coords, 'b-', linewidth=2, label='Path')
    plt.plot(x_coords[0], y_coords[0], 'go', markersize=15, label='Start')
    plt.plot(x_coords[-1], y_coords[-1], 'ro', markersize=15, label='End')

    for i in range(0, len(x_coords)-1, 5):
        dx = x_coords[i+1] - x_coords[i]
        dy = y_coords[i+1] - y_coords[i]
        if dx != 0 or dy != 0:
            plt.arrow(x_coords[i], y_coords[i], dx*0.2, dy*0.2,
                      head_width=0.1, head_length=0.2, fc='blue', ec='blue')

    plt.grid(True)
```

```
    plt.axis('equal')
    plt.title('Robot Trajectory')
    plt.xlabel('X Position')
    plt.ylabel('Y Position')
    plt.legend()
    plt.text(0.02, 0.98, f'Total Movements: {len(x_coords)-1}',
             transform=plt.gca().transAxes, fontsize=10,
             verticalalignment='top')
    plt.show()


def main():
    try:
        movements = read_log_file()
        x_coords, y_coords = calculate_trajectory(movements)
        plot_trajectory(x_coords, y_coords)
        print(f"Total movements: {len(movements)}")
        print(f"Starting position: ({x_coords[0]:.2f}, {y_coords[0]:.2f})")
        print(f"Ending position: ({x_coords[-1]:.2f}, {y_coords[-1]:.2f})")
    except FileNotFoundError:
        print("Error: path.txt file not found!")
    except Exception as e:
        print(f"An error occurred: {str(e)}")


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