CSE310 Lab-2 Final Project Report

Combat Soccer Bot

Created by-

MD Masum Musfique

ID: 1920582

CSE310Lab

Section: 2

Team Name: Null_Function

Group-10

Theory Section: 2(Completed theory part at Autumn 2021)

Submitted to-

Azfar Hossain

Adjunct Faculty, IUB

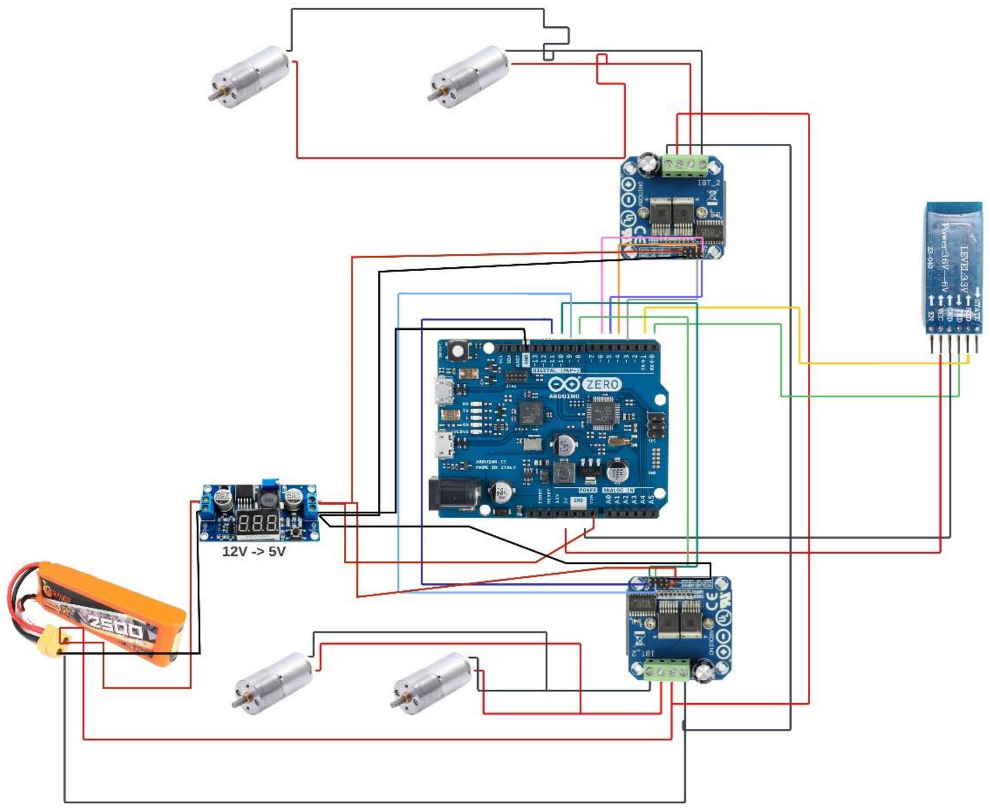
Project Name: Designing a combat soccer bot

Objective: Designing a combat soccer bot with Arduino and motor controller

Apparatus:

- Arduino- Uno (Quantity - 1)
- Buck converter (Quantity - 1)
- BTS 7960 Motor controller (Quantity - 2)
- Shaft coupler short (Quantity - 4)
- Flat wheel (Quantity - 4)
- Bracket 36mm Mount (Quantity - 4)
- 300 RPM DC 37mm motor (Quantity - 4)
- HC-05 Bluetooth Module (Quantity - 1)
- T-connector (Quantity - 1)
- Battery 2200mAh (Quantity - 1)
- Charger B3 (Quantity - 1)

Circuit Diagram:



Procedure:

1. Connect the Bluetooth module with the Arduino to pins according to the diagram. Connect the 5v pin and ground of the Arduino to power up Bluetooth module.
2. Connect the motors parallelly & connect the positive & negative wires according to the BTS motor's to the M+ & M- point. Now connect the battery with the motor controller with B+ & B- point. Then connect another motor controller parallelly.
3. Connect another 2 parallelly connected motors with the motor controller.
4. Connect the motor controllers' pin with the Arduino according to the circuit diagram.
5. Connect the battery with buck converter, adjust the buck converter's voltage to 5v and connect the output wires(Vout+, Vout-) to the Vin & ground pin with Arduino and motor controllers' VCC and ground pins.
6. Upload the code to the Arduino board.
7. Power on the Arduino and pair the HC-05 Bluetooth module with your phone.
8. Install the "Bluetooth RC car" app on your phone.
9. Open the app, connect to the HC-05 module, and use the on-screen buttons to send commands to control the RC car or soccer bot.

Code:

```
// RC and Soccer bot control using 2BTS motor driver
int L_EN_FOR_ONE=3;
int R_EN_FOR_ONE=4;
int L_PWM_FOR_ONE=5;
int R_PWM_FOR_ONE=6;
int L_EN_FOR_TWO=8;
int R_EN_FOR_TWO=9;
int L_PWM_FOR_TWO=10;
int R_PWM_FOR_TWO=11;
char incomingByte; // for incoming serial data
int speed_min = 155; //the minimum "speed" the motors will turn - take it lower and motors don't turn
int speed_max = 255; //

int speed_left = speed_max; // set both motors to maximum speed
int speed_right = speed_max;

void left();
void right();
void forward();
void backward();
void forward_left();
void forward_right();
void back_left();
void back_right();
void setup() {
  Serial.begin(9600);
pinMode(L_EN_FOR_ONE,OUTPUT);
pinMode(R_EN_FOR_ONE,OUTPUT);
pinMode(L_PWM_FOR_ONE,OUTPUT);
pinMode(R_PWM_FOR_ONE,OUTPUT);
pinMode(L_EN_FOR_TWO,OUTPUT);
pinMode(R_EN_FOR_TWO,OUTPUT);
pinMode(L_PWM_FOR_TWO,OUTPUT);
pinMode(R_PWM_FOR_TWO,OUTPUT);

digitalWrite(L_EN_FOR_ONE,HIGH);
digitalWrite(R_EN_FOR_ONE,HIGH);
digitalWrite(L_EN_FOR_TWO,HIGH);
digitalWrite(R_EN_FOR_TWO,HIGH);
}

void loop() {

if (Serial.available() > 0) {
    incomingByte = Serial.read();
    }
  switch(incomingByte)
  {
     case 'S':
      {
        stopo();
       //Serial.println("Stop\n");
       incomingByte='*';}
     break;

     case 'L':

     {  left();
```

```
 // Serial.println("Forward\n");
  incomingByte='*';}
break;

 case 'F':

{   right();
 // Serial.println("Backward\n");
  incomingByte='*';}
break;

case 'R':
// turn right
{
  forward();
 // Serial.println("Rotate Right\n");
  incomingByte='*';}
break;
  case 'B':
 {
  backward();
  //Serial.println("Rotate Left\n");
  incomingByte='*';}
break;
case '1':

 { speed_left = 20;
  speed_right = 20;
  //Serial.println("Speed 1\n");
  incomingByte='*';}
 break;
case '2':
  {
    speed_left = 40;
   speed_right = 40;
   //Serial.println("Speed 2 \n");
   incomingByte='*';}
 break;
case '3':
  {
    speed_left = 60;
   speed_right = 60;
   //Serial.println("Speed 3 \n");
   incomingByte='*';}
 break;
    case '4':
  {
    speed_left = 80;
   speed_right = 80;
   //Serial.println("Speed 4 \n");
   incomingByte='*';}
 break;
    case '5':
  {
    speed_left = 100;
   speed_right = 100;
   //Serial.println("Speed 5 \n");
   incomingByte='*';}
 break;
    case '6':
```

```
 {
   speed_left = 120;
  speed_right = 120;
  //Serial.println("Speed 6 \n");
  incomingByte='*';}
break;
   case '7':
 {
   speed_left = 140;
  speed_right = 140;
 // Serial.println("Speed 7 \n");
  incomingByte='*';}
break;
   case '8':
 {
   speed_left = 160;
  speed_right = 160;
  //Serial.println("Speed 8 \n");
  incomingByte='*';}
break;
   case '9':
 {
   speed_left = 200;
  speed_right = 200;
  //Serial.println("Speed 9 \n");
  incomingByte='*';}
break;
   case 'q':
 {
   speed_left = 255;
```

```
     speed_right = 255;
     Serial.println("Speed full \n");
     incomingByte='*';}
   break;
      case 'H':
    {
     back_right();
     Serial.println("Speed full \n");
     incomingByte='*';}
   break;
    case 'I':
    {
     back_left();
     Serial.println("Speed full \n");
     incomingByte='*';}
   break;
    case 'G':
    {
     forward_right();
     Serial.println("Speed full \n");
     incomingByte='*';}
   break;
    case 'J':
    {
    forward_left();
     Serial.println("Speed full \n");
     incomingByte='*';}
   break;
}
```

```
}
void forward(){

  analogWrite(R_PWM_FOR_ONE,speed_left);
  analogWrite(L_PWM_FOR_ONE,0);
  analogWrite(R_PWM_FOR_TWO,0);
  analogWrite(L_PWM_FOR_TWO,speed_right);
  };
void backward(){
    analogWrite(R_PWM_FOR_ONE,0);
  analogWrite(L_PWM_FOR_ONE,speed_left);
  analogWrite(R_PWM_FOR_TWO,speed_right);
  analogWrite(L_PWM_FOR_TWO,0);
  };
void right(){
   analogWrite(R_PWM_FOR_ONE,0);
  analogWrite(L_PWM_FOR_ONE,speed_left);
  analogWrite(R_PWM_FOR_TWO,0);
  analogWrite(L_PWM_FOR_TWO,speed_right);
  };
void left(){
     analogWrite(R_PWM_FOR_ONE,speed_left);
  analogWrite(L_PWM_FOR_ONE,0);
  analogWrite(R_PWM_FOR_TWO,speed_right);
  analogWrite(L_PWM_FOR_TWO,0);


  };
void stopo(){
    analogWrite(R_PWM_FOR_ONE,0);
  analogWrite(L_PWM_FOR_ONE,0);
  analogWrite(R_PWM_FOR_TWO,0);
  analogWrite(L_PWM_FOR_TWO,0);



  };
void forward_left(){
   analogWrite(R_PWM_FOR_ONE,0);
  analogWrite(L_PWM_FOR_ONE,0);
  analogWrite(R_PWM_FOR_TWO,speed_right);
  analogWrite(L_PWM_FOR_TWO,0);



  };
void forward_right(){
     analogWrite(R_PWM_FOR_ONE,0);
  analogWrite(L_PWM_FOR_ONE,speed_left);
  analogWrite(R_PWM_FOR_TWO,0);
  analogWrite(L_PWM_FOR_TWO,0);
  };
void back_left(){
      analogWrite(R_PWM_FOR_ONE,0);
  analogWrite(L_PWM_FOR_ONE,0);
  analogWrite(R_PWM_FOR_TWO,0);
  analogWrite(L_PWM_FOR_TWO,speed_right);

  };
void back_right(){
    analogWrite(R_PWM_FOR_ONE,speed_left);
  analogWrite(L_PWM_FOR_ONE,0);
  analogWrite(R_PWM_FOR_TWO,0);
  analogWrite(L_PWM_FOR_TWO,0);


  };
```

Result Analysis:

Performance Evaluation:

During the testing and evaluation phase, the soccer bot demonstrated several noteworthy outcomes in terms of its design, functionality, and overall performance.

1. Navigation and Maneuverability: The soccer bot exhibited commendable navigation and maneuverability on the playing field. It was able to move in various directions with precision and respond to commands effectively, showcasing its well-calibrated motor control and motion algorithms.
2. Ball Interaction: The soccer bot demonstrated consistent ball interaction skills. It was able to approach the ball, accurately control its movement, and even make attempts at goal scoring. The integration of sensors and control algorithms allowed the bot to maintain a balanced interaction with the ball.
3. Strategy Implementation: The implemented strategies for offense and defense displayed promising results. The bot was able to execute predefined plays, demonstrating a fundamental understanding of the game's dynamics and the ability to adapt its behavior according to different scenarios.

Challenges and Improvements:

While the soccer bot performed well in many aspects, there were a few challenges and areas for potential improvement:

1. Speed and Agility: The bot's speed and agility could be further enhanced to improve its responsiveness during gameplay. Optimizing the motor control algorithms and exploring the use of more powerful motors could result in quicker movements and more agile gameplay.
2. Precision in Ball Control: While the bot was capable of interacting with the ball, fine-tuning its ball control mechanisms could lead to more accurate ball handling. This would involve refining the coordination between the bot's movements and the manipulation of the ball.
3. Advanced Strategies: Implementing more advanced strategies, such as coordinated team plays or adaptive decision-making based on opponent behavior, could take the bot's gameplay to the next level. This would require a deeper understanding of soccer dynamics and sophisticated programming.

Conclusion:

The development of the soccer bot, based on Arduino technology, was a challenging and rewarding experience. The successful construction and testing of the bot showcased the integration of hardware components, sensors, and programming to create a functional and interactive robotic system.

Through this project, valuable insights were gained into various aspects of robotics, including motor control, sensor integration, path planning, and strategy implementation. Additionally, the project highlighted the significance of iterative design and troubleshooting in the development process.

While the soccer bot's current iteration demonstrated promising performance in navigation, ball interaction, and strategy execution, there is room for further enhancement. By addressing the identified challenges and exploring advanced features, the soccer bot has the potential to evolve into a more sophisticated and competitive player in the realm of robotic soccer.

In conclusion, the successful creation of the soccer bot serves as a testament to the capabilities of Arduino-based systems in realizing complex robotic projects. The project's outcomes contribute to the field of robotics and automation, providing valuable insights for future endeavors in creating intelligent and agile robotic systems.

Precautions:

While developing and operating the soccer bot (RC car) project, it's important to consider several precautions to ensure safety, protect equipment, and prevent any potential hazards. The following precautions should be taken into account:

1. Electrical Safety:

- When working with electronic components and power sources, always ensure that the power is disconnected before making any connections or modifications to the circuitry.
- Avoid handling exposed wires while the system is powered on to prevent electrical shocks or short circuits.

2. Battery Handling:

- If using rechargeable batteries, follow the manufacturer's instructions for proper charging procedures to prevent overcharging, overheating, or damaging the batteries.
- Ensure that batteries are inserted in the correct orientation to avoid polarity-related issues.

3. Motor and Moving Parts:

- Exercise caution when testing or operating the soccer bot, especially in confined spaces. Ensure that the bot has enough clearance to move without causing damage to surrounding objects or itself.

- Keep fingers and clothing away from moving parts, such as wheels and motors, while the bot is in operation to avoid any accidents.

4. Obstacle Avoidance:

- While testing the obstacle avoidance functionality, ensure that the soccer bot is operated in an open area with minimal obstructions to prevent unintended collisions.

5. Heat Dissipation:

- During extended periods of operation, electronic components, especially motors and driver circuits, may generate heat. Allow adequate time for components to cool down to prevent overheating.

6. Fire Safety:

- Be cautious when working with any heat-generating components or batteries. Avoid leaving the soccer bot unattended while powered on to mitigate the risk of fire.

7. Data Backup and Version Control:

- Regularly backup your project code and configurations to prevent data loss in case of unexpected hardware failures or software glitches.

- Use version control systems to track changes in your codebase, facilitating easy recovery if errors are introduced.

8. Secure Workspace:

- Set up your work area in a well-ventilated and well-lit space to ensure comfortable working conditions.

- Keep your workspace organized and free from clutter to avoid accidentally damaging components or causing tripping hazards.

9. Supervision:

If the soccer bot project is being demonstrated or used by others, ensure that it is operated under supervision, especially in public or crowded areas.

10. Documentation:

Keep detailed documentation of your project, including circuit diagrams, code, and assembly instructions. This documentation can be valuable for troubleshooting and future reference.