Project Battle Bot

Documentation

Version 1.0

Author:

Nathan P. Pais D'costa

27.05.2024
Marcel Baron, Jaqueline Berghout
NHL Stenden University of Applied Sciences

Table of Contents:

Requirements Analysis

The requirements of Project Battle Bot are as follows:

1. The robot must independently navigate a black track on a white background.
2. The robot must grab an object placed along the track and take that object to the destination.
3. The robot must finish its tracking at a solid black box where the robot my release the object previously grabbed.

Technical Design

Hardware:

- Arduino Nano Microcontroller
- Printed Circuit Board (PCB)
- 10,000 mAh battery pack
- Gripper with a servo motor
- Two Electomotors
- Ultrasonic Distance Sensor
- 8 Analogue line tracking sensors

I/O List:

- leftWheelBack = 8; OUTPUT
- leftWheelFront = 4; OUTPUT
- rightWheelBack = 5; OUTPUT
- rightWheelFront = 6; OUTPUT
- leftWheelSpeedPin = 9; OUTPUT
- rightWheelSpeedPin = 10; OUTPUT
- sensorPins[8] = {A0, A1, A2, A3, A4, A5, A6, A7}; INPUT
- trigPin = 7; INPUT
- echoPin = 13; OUTPUT
- gripperPin = 12; OUTPUT

Testing Environments and Equipment Used:

Multiple testing environments were set up and used. Such environments are listed below:

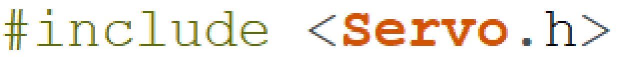
- NHL Stenden Standard Project BattleBot Race Day Track.
- Home made track on clear white background.
    - Includes 2 loops with 2 intersections, and one end-point.

In addition to the testing environments, multiple objects were used to test the functionality of the grippers;

- NHL Stenden Standard Project BattleBot Race Day Object
- Paper Cup
- Black Electrical Tape to mark the track.

Code Explanation:

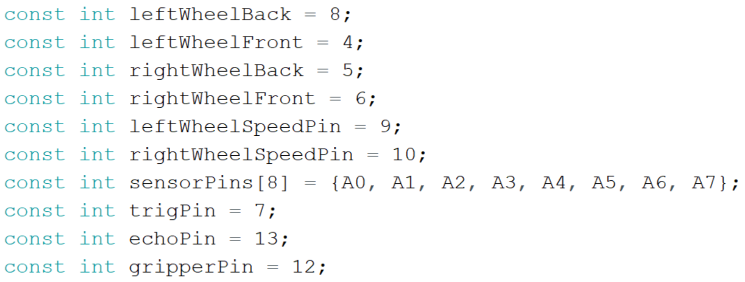
- Libraries



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

- This library enables control of the servo Motors and its manipulation to certain angles and speeds.

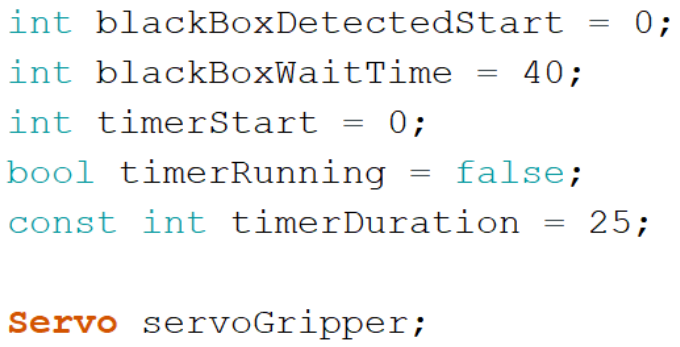
- Pin Definitions



```
const int leftWheelBack = 8;
const int leftWheelFront = 4;
const int rightWheelBack = 5;
const int rightWheelFront = 6;
const int leftWheelSpeedPin = 9;
const int rightWheelSpeedPin = 10;
const int sensorPins[8] = {A0, A1, A2, A3, A4, A5, A6, A7};
const int trigPin = 7;
const int echoPin = 13;
const int gripperPin = 12;
```

- The first four decelerations focus on the movements of the left and right motors. Forward and backward rotation for both the left and the right wheels.
- Sensor Pins: Defines analog pins connected to an array of sensors for line tracking.
- Trig and Echo: These pins define the functionality of the Ultrasonic sensor used to measure distances between objects.
- Gripper Pin: Specifies the pin connected to the servo controlling the gripper.

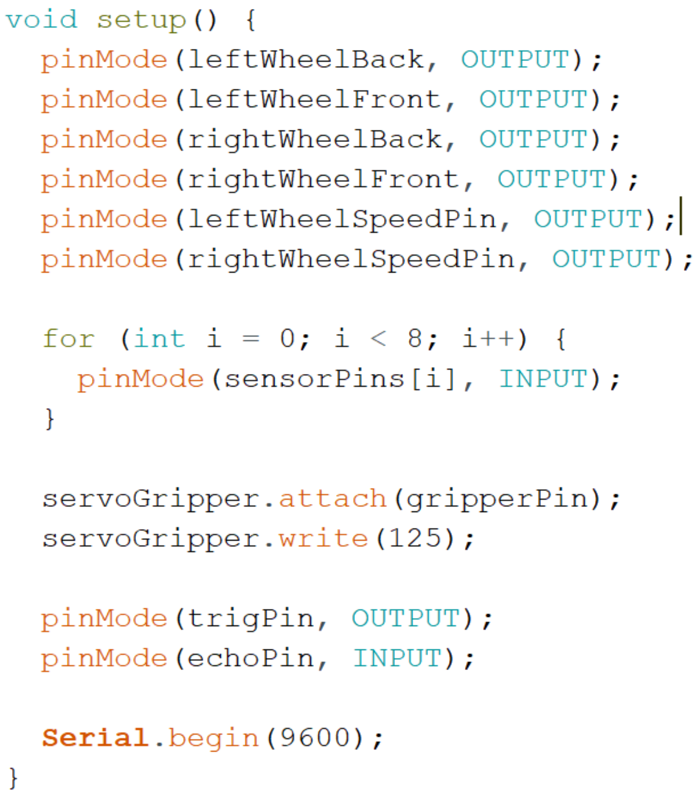
- Timers



```
int blackBoxDetectedStart = 0;
int blackBoxWaitTime = 40;
int timerStart = 0;
bool timerRunning = false;
const int timerDuration = 25;

Servo servoGripper;
```

  - Black box Detection: Keeps track of when a black surface is detected on all the sensors and sets a delay before stopping the motors.
  - Times: Manages timing for motor operations to stop after a set duration.
  - Servo Object: Creates an instance of a servo motor for gripper control.

- Setup



```
void setup() {
  pinMode(leftWheelBack, OUTPUT);
  pinMode(leftWheelFront, OUTPUT);
  pinMode(rightWheelBack, OUTPUT);
  pinMode(rightWheelFront, OUTPUT);
  pinMode(leftWheelSpeedPin, OUTPUT);
  pinMode(rightWheelSpeedPin, OUTPUT);

  for (int i = 0; i < 8; i++) {
    pinMode(sensorPins[i], INPUT);
  }

  servoGripper.attach(gripperPin);
  servoGripper.write(125);

  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);

  Serial.begin(9600);
}
```

  - Pin Mode: Configures the motor control pins for output to drive the motors.
  - Sensor Pins: Sets the sensor pins to input mode to read values.
  - Servo: Sets the initial position for the servo controlling the gripper.
  - Ultrasonic Sensor Initialization: Sets the pins for the ultrasonic sensor.
  - Serial Communication: Initializes serial communication for debugging.

- Main Loop

```
void loop() {
  int sensorValues[8];

  for (int i = 0; i < 8; i++) {
    sensorValues[i] = analogRead(sensorPins[i]);
  }

  if (sensorValues[3] > 500 && sensorValues[4] > 500) {
    moveForward();
    startTimer();
  } else if (sensorValues[2] > 500) {
    turnRight();
  } else if (sensorValues[5] > 500) {
    turnLeft();
  } else {
    blackBoxDetectedStart = millis();
    if (millis() - blackBoxDetectedStart > blackBoxWaitTime) {
      controlGripper(125);
      stopMotors();
    }
  }
  checkTimer();

  if (isObjectDetected()) {
    controlGripper(45);
  } else {
    controlGripper(125);
  }
}
```

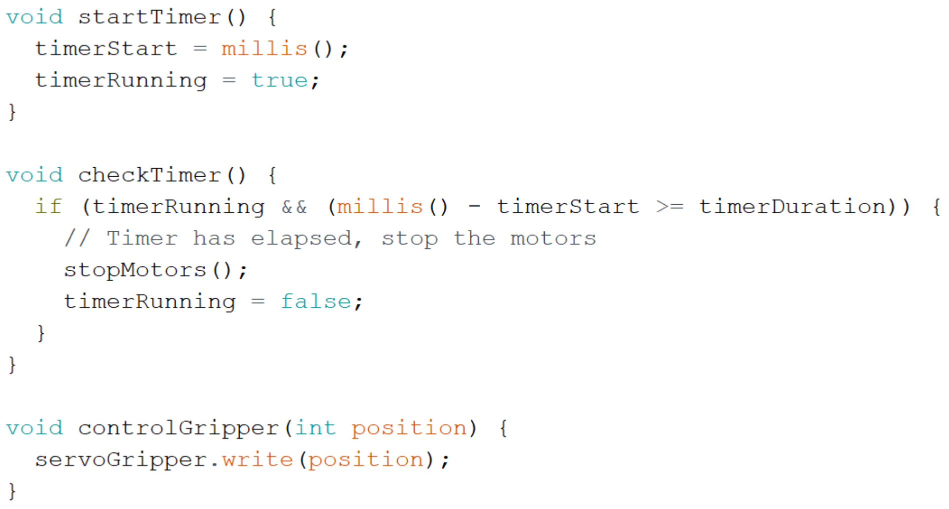
- Sensor Values: Array to hold the readings from the line-tracking sensors.
- Read Sensor Values: Continuously reads values from the sensors.
- Directions:
  - Moves forward if both center sensors detect the line.
  - Turns right if the left center sensor detects the line.
  - Turns left if the right center sensor detects the line.
  - Stops the motors if all the sensors detect a black line (black box)
- Timer Check: Verifies if the timer has expired and stops the motors if it has.
- Gripper Control: Checks the distance using the ultrasonic sensor and controls the gripper based on object detection.

- Functions

```
void moveForward() {
  digitalWrite(leftWheelFront, HIGH);
  digitalWrite(rightWheelFront, HIGH);
  digitalWrite(leftWheelBack, LOW);
  digitalWrite(rightWheelBack, LOW);
}

void turnRight() {
  digitalWrite(leftWheelFront, HIGH);
  digitalWrite(rightWheelFront, LOW);
  digitalWrite(leftWheelBack, LOW);
  digitalWrite(rightWheelBack, LOW);
}

void turnLeft() {
  digitalWrite(leftWheelFront, LOW);
  digitalWrite(rightWheelFront, HIGH);
  digitalWrite(leftWheelBack, LOW);
  digitalWrite(rightWheelBack, LOW);
}

void stopMotors() {
  digitalWrite(leftWheelFront, LOW);
  digitalWrite(rightWheelFront, LOW);
  digitalWrite(leftWheelBack, LOW);
  digitalWrite(rightWheelBack, LOW);
}
```

    - Move Forward: Sets the respective motors to move the robot forward.
    - Move Left: Sets the respective motors to move the robot left.
    - Move right: Sets the respective motors to move the robot right.
    - Stop Motors: Sets all motors off so that the robot stops.

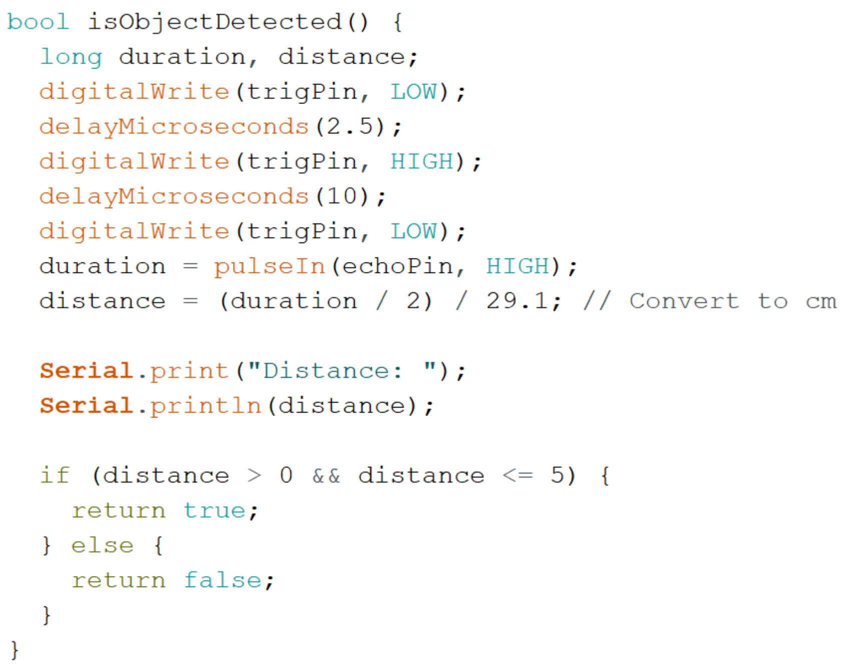
- ## Functions Part 2



```
void startTimer() {
  timerStart = millis();
  timerRunning = true;
}

void checkTimer() {
  if (timerRunning && (millis() - timerStart >= timerDuration)) {
    // Timer has elapsed, stop the motors
    stopMotors();
    timerRunning = false;
  }
}

void controlGripper(int position) {
  servoGripper.write(position);
}
```

  - Start Timer: Initializes the timer by recording the current time.
  - Check Timer: Checks if the timer has expired and stops the motors if the specified duration has passed.
  - Control Gripper: Sets the position of the servo gripper based on the preset angles.

- ## Object Detection



```
bool isObjectDetected() {
  long duration, distance;
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2.5);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distance = (duration / 2) / 29.1; // Convert to cm

  Serial.print("Distance: ");
  Serial.println(distance);

  if (distance > 0 && distance <= 5) {
    return true;
  } else {
    return false;
  }
}
```

  - Measures distance using the ultrasonic sensor by sending a pulse and timing the echo.
  - Converts the duration to distance and checks if the distance is within a specified threshold to detect objects.