EE321 Microelectronics Project

Final Report

Erdem Sezen (S028443)
erdem.sezen@ozu.edu.tr

Yigit Emir Soyhan (S028281)
yigit.soyhan@ozu.edu.tr

Department of Electrical and Electronics Engineering

May 22, 2024

Contents

1 Introduction **2**

2 How the System Works **2**
2.1 Defining pins and variables 2
2.2 void setup() 2
2.3 void loop() 3

3 Tests **4**
3.1 Case 1 5
3.2 Case 2 5
3.3 Case 3 5
3.4 Case 4 5
3.5 Case 5 5

4 Conclusion **6**

5 Presentation Video **6**

6 User Manual **7**
6.1 How to use 7
6.2 Charging 7
6.3 Warnings 7

1 Introduction

The aim of this project is to create a robot that can scan its surroundings. To achieve this, we will use an Arduino UNO microcontroller board, a Zumo Shield, and a MZ80 infrared sensor. For coding we have used Arduino IDE version 2, and the Zumo Shield Library by Pololu to control our robot. In our report we will talk about how our robot works, tests we have conducted, and we will include a small user manual at the end.

2 How the System Works

Arduino by design has two methods that run codes inside the machine. These are `void setup()`, and `void loop()`. As the name suggests, `loop()` runs the code infinitely inside a loop, and `setup()` only runs the code once before `loop()`. In this section we will explain what codes the robot executes in these methods using the help of state machine diagrams.

2.1 Defining pins and variables

Before the code starts we have assigned some pins as input for sensors, and some pins for output. Pin 13 was assigned to LED as output to control the built-in LED in Zumo Shield. Pin 6 was assigned to MZ80 sensor as input. We have also created these variables to use later on in the code: `unsigned int count = 0`, `unsigned int state = 0`

2.2 void setup()

We have used this part to run our calibration for reflectance sensor array. This sensor is used to make sure that our robot stays inside the arena bounds. To make this work we have to calibrate the sensor and make sure that it understands the bounds of the arena. Which are white. We use the built-in `calibrate()` function inside Zumo Shield library for calibration. We run the `calibrate()` function for 10 seconds inside the `setup()` method. This is enough for the robot to understand the borders of the test arena. Later on we will use this reflectance sensor to alert the robot when it has reached a border. Code snippet for `setup()` is added below:

```
void setup() {
  pinMode(LED_PIN, OUTPUT);
  digitalWrite(LED_PIN, HIGH);

  // Start Of The Calibration
  reflectanceSensors.init();
  unsigned long startTime = millis();
  while (millis() - startTime < 5000)  // make the calibration take 10 seconds
  {
    reflectanceSensors.calibrate();
  }
  // End Of The Calibration
}
```

2.3 void loop()

This is where we have implemented most of the logic for the robot. Firstly, to understand when the calibration sequence ends we made the buzzer play a simple sound. Then it will start a loop that lasts for 13.4 seconds. This value is not a made up value. We have calculated how long it takes for the robot to spin around itself and multiplied that value by 2. We have done this to minimize any possible error that could occur when scanning the area. During this 13.4 seconds it does a number of things. First, using the reflectance sensor it checks whether it is inside the arena or not. If it is not it corrects itself, if it is inside the bounds it will just turn right. While turning right, it will also count the number of objects in the arena. For this we will use the variables we have created in Section 2.1. We will increment the count variable every time an object is detected. However, we want to be able to count objects even if they are of different shapes and sizes. To achieve this we have used a state system.

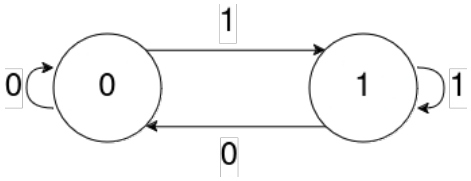


Figure 1: State Diagram

A state diagram explaining this system is shown in Figure 1. Circles represent the state of the system which is saved in the state integer. Arrows represent readings of the MZ80 infrared sensor. Where 1 means it has read something and 0 means it hasn't. When the state is zero and the reading is zero it means that it is in the empty space, so it does nothing. In this state if the sensor reads something, the state will change to 1. Meaning that it has started to detect and object. This State will not change until the sensor stops reading a value. When the sensor stops reading we will understand that we have stopped scanning that object. Therefore, our state will return to zero. This will suggest that we have detected an object. So we will increment the counter by 1. This state algorithm allows our robot to detect objects regardless of their shape and size.

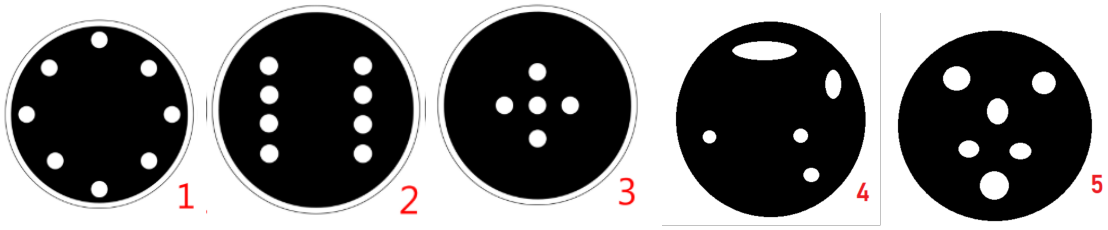
All the stuff until this point happens in the 13.4 seconds loop. When that happens we will set the motor speeds to zero to make the robot stop. Since it has spun around itself two times, it will detect everything twice. So we divide this number by a factor of 2. If it is a fractional number we will round it to the next integer. We have decided to round up, because we have noticed that errors have a tendency to occur from not reading an object, rather than reading it more than once. Then, with a simple while loop it will blink the LED defined in Section 2.1 the number of times it has counted. For example if 5 objects were counted, it will blink 5 times for 1 second, with intervals of 0.5 seconds. In the end, to stop the robot and finish the sequence we have put an infinite while loop. In this loop, we have also added a code to stop the motors for good measure. Code snippet for `loop()` is added below:

```
  void loop() {

  buzzer.playFrequency(1000, 500, 15);
  delay(3000);
  unsigned long startTime = millis();
  while (millis() - startTime < 13400){
    positionVal = reflectanceSensors.readLine(sensorValues);
    if (mostLeftSensor() == 1) {
      motors.setSpeeds(-200,-200);
      delay(200);
      motors.setSpeeds(-400, 400);
      delay(300);
    } else {
      turnRight();
    }
    if(state == 0){
      if (!digitalRead(MZ80_PIN)) {
        state = 1 ;
      }
    }else{
      if (!digitalRead(MZ80_PIN)) {
        state = 1 ;
      }else{
        state = 0;
        count++;
      }
    }
  }
  motors.setSpeeds(0,0);
  count = ceil((count/2));
  digitalWrite(LED_PIN, LOW);
  delay(2000);
  while(count>0){
    motors.setSpeeds(0,0);
    delay(1000);
    digitalWrite(LED_PIN, HIGH);
    delay(500);
    digitalWrite(LED_PIN, LOW);
    delay(500);
    count--;
  }
  while(1){
    motors.setSpeeds(0,0);
  }
}
```

3 Tests

To ensure that our robot is working correctly, we have conducted 5 test cases. This cases are explained shortly below. Representation of the cases mentioned below can be found in Figure 2.



(a) Case 1 (b) Case 2 (c) Case 3 (d) Case 4 (e) Case 5

Figure 2: Test cases

3.1 Case 1

For the first case, we placed objects in circular position and put our robot in the center of the platform to test its 360-degree turn ability. We concluded that it can do 360 turn and detect objects without error. The columns made out of papers had near-exact shape. Our robot spun 2 times around its axis and counted the objects on the platform with exact precision.

3.2 Case 2

In the second case, we placed objects in parallel lines like columns to test robots ability to differentiate the objects near each other in a line. These objects were also had near-exact shape. We concluded that it's successful at differentiate objects that are in line position.

3.3 Case 3

In the third case, we placed objects of near same size on a cross-like shape(X shape) to check that can our robot detect objects in the middle while middle object has other objects near at it in all directions and our robot once again proved it can do it without error.

3.4 Case 4

For the fourth case, we placed three single circular columns, one column with 2 column-width and one column with 3 column-width to check can our differentiate between various sized objects and count them as one rather than 2 or 3 or more. We concluded that our robot understands that various sized objects are also one objects regardless of their size.

3.5 Case 5

For the last case, we placed object in randomized manner to detect what our robot counts in chaotic order. We saw that our robot can handle the randomly placed object counting regardless of distance and angle between them.

4 Conclusion

In conclusion our robot satisfies all the expected results. It can count the number of objects in an area, even if they are varying in size. It does not fall from the edges, and it is easy to use. This project has helped us understand how functional programming in integrated microprocessors are done. It has also been a great real-life example for us. Thank you for reading to the end of this document. We hope our goals and progress has been clearly expressed. Next Page includes a user Manual on how to use this Zumo Robot.

5 Presentation Video

Here is a link for a video Presentation about our project:
`https://www.youtube.com/watch?v=0N69X0A24nA`

6 User Manual

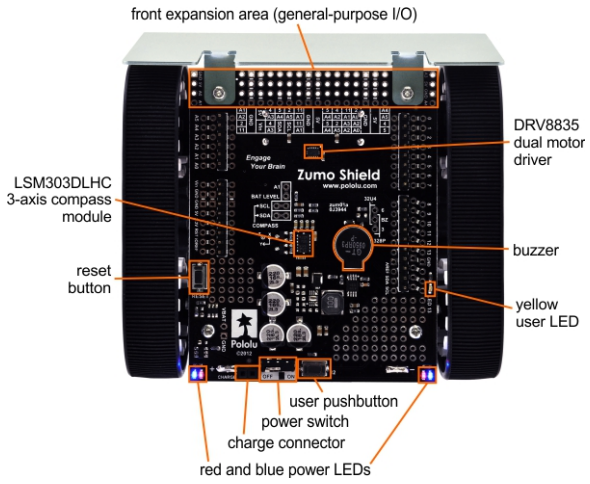


Figure 3: Zumo Robot diagram

6.1 How to use

To use our robot simply turn on the power switch shown in Figure 3. You will understand that it has turned on from the power LEDs. When the robot starts it will be in calibration phase for 5 seconds. During this phase simply show the front-end (the face with metal plate) of the robot to the edges of the arena and inside the arena, repeatedly. When this sequence ends you will hear a beep noise. When you hear this noise quickly put the robot to your desired point in the arena. Be fast you have 3 seconds to do this!

After the scanning is complete your robot will stop and starting showing the number of objects in the arena by blinking the yellow user LED shown in Figure 3. It will blink as many times as the objects in the arena. For example, if there are 5 pillars in the arena, it will blink 5 times. To run it again simply press the reset button on the robot, or shutdown and start the robot using the power switch.

6.2 Charging

Robot is charged using 4 AA batteries. You can find the compartment for batteries at the bottom of the robot.

6.3 Warnings

- Do not let the robot get in contact with water! If so turn it off, take out the batteries and dry it off.
- When placing the robot in the area make sure that it is looking at an empty space or results may vary.
- Don't hold the robot from motor, or you might damage them.