Lab Report for EE321 Microprocessors

Final Project Report

Çağatay Duru
`cagatay.duru@ozu.edu.tr`

Emirhan Benderli
`emirhan.benderli@ozu.edu.tr`

Department of Electrical and Electronics Engineering

May 22, 2024

1 Project Overview

The objective of this project is to program a Zumo robot to scan its surroundings and report the number of objects around it in an arena with objects and white borders. The robot should move around the arena, detect objects using the MZ80 infrared sensor, and indicate the number of objects detected by blinking its LED a corresponding number of times. The robot should stay within the arena's borders, which will be detected using the Zumo Reflectance Sensor Array.

2 Team Members

- Çağatay Duru-S029021
- Emirhan Benderli-S025235

3 Components and Libraries

1. Components
    - MZ80 Infrared Sensor
    - 6 X Reflectance Sensors
    - Buzzer
2. Libraries
    - ZumoShield.h which includes:
        - QTRSensors.h
        - ZumoMotors.h
        - ZumoBuzzer.h
        - ZumoReflectanceSensorArray.h

4 User Manual

4.1 Power

- First, insert 6 x AA battery to battery compartment

4.2 Calibrate

- For starting the zumo robot, press the on button. Then, zumo will process calibration for 5 seconds. During this time, show both white and black ground to your zumo robot's reflectance sensors which are placed at front bottom part of the zumo.

4.3 Placement

- After calibration part, place your zumo to somewhere in the arena. Usually center of the arena is a good placement point however, in some experiments there might be a obstacle in the center. Therefore, feel comfortable to place the zumo anywhere you like, as long as you place the Zumo completely inside the arena, it will work well and won't fall down in most cases.

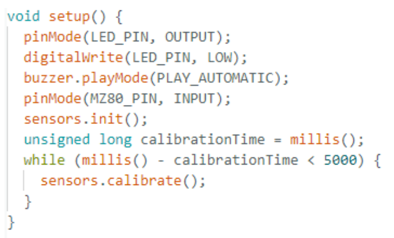
5 Challenges

1. Accurate object counting: The MZ80 infrared sensor have to count each object only once even if it's observed more than once or if it's a wider object.
2. Different start locations: Effectively using the Zumo Reflectance Sensor Array to detect and stay within the arena's borders may require careful calibration and programming.
3. Different sized objects: Object size shouldn't bother our algorithm, MZ80 infrared sensor have to count each object only once even if it's a wider object.
4. LED blinking timing: Program shall stop operating when there are no objects left that zumo didn't count, not when there's still uncounted objects.

6 Method

6.1 Calibration

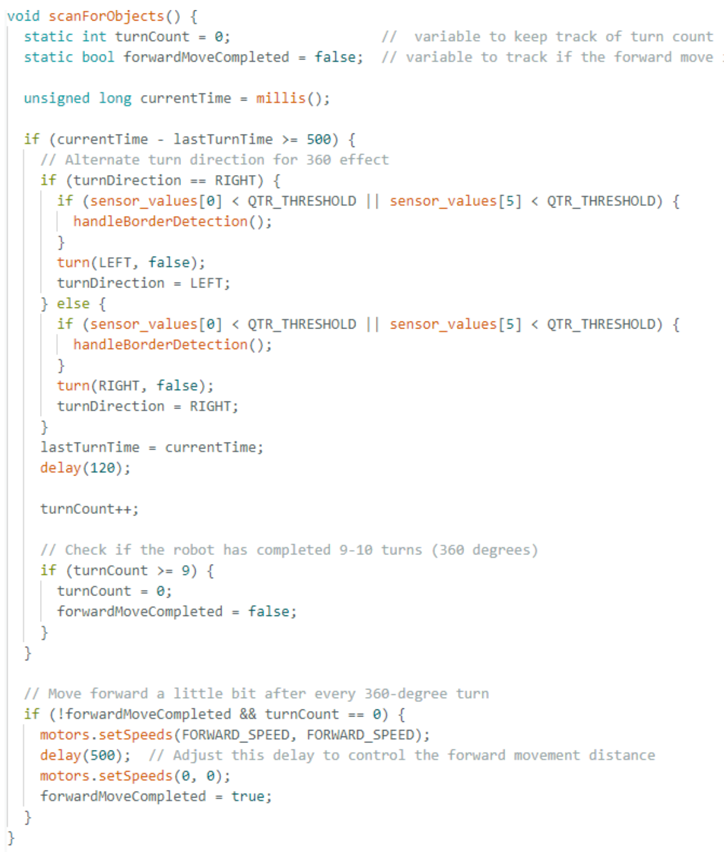
- Calibration process is essential for zumo not to fall down from the arena. After calibrating for 5 seconds, reflectance sensors starts to differentiate between the white borders and black ground of the arena which will be used later to keep the zumo within the arena.



```
void setup() {
  pinMode(LED_PIN, OUTPUT);
  digitalWrite(LED_PIN, LOW);
  buzzer.playMode(PLAY_AUTOMATIC);
  pinMode(MZ80_PIN, INPUT);
  sensors.init();
  unsigned long calibrationTime = millis();
  while (millis() - calibrationTime < 5000) {
    sensors.calibrate();
  }
}
```

6.2 Search Algorithm

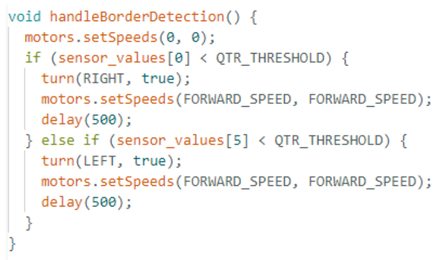
- The search algorithm is designed to scan the arena for objects while avoiding the borders. It works by continuously turning the robot and periodically moving it forward a small distance.
- The robot alternates between turning left and right, keeping track of the number of turns made using the turnCount variable.
- After every 9-10 turns (approximately 360 degrees), the robot moves forward a small distance to scan a new area of the arena.
- Before each turn, the algorithm checks if the robot is near the border using the reflectance sensors. If it detects the border, it calls the handleBorderDetection() function to prevent the robot from falling off the arena.
- The algorithm continues this process of turning, checking for borders, and periodically moving forward until all objects in the arena have been detected and handled.



```
void scanForObjects() {
  static int turnCount = 0;                     //  variable to keep track of turn count
  static bool forwardMoveCompleted = false;  // variable to track if the forward move

  unsigned long currentTime = millis();

  if (currentTime - lastTurnTime >= 500) {
    // Alternate turn direction for 360 effect
    if (turnDirection == RIGHT) {
      if (sensor_values[0] < QTR_THRESHOLD || sensor_values[5] < QTR_THRESHOLD) {
        handleBorderDetection();
      }
      turn(LEFT, false);
      turnDirection = LEFT;
    } else {
      if (sensor_values[0] < QTR_THRESHOLD || sensor_values[5] < QTR_THRESHOLD) {
        handleBorderDetection();
      }
      turn(RIGHT, false);
      turnDirection = RIGHT;
    }
    lastTurnTime = currentTime;
    delay(120);

    turnCount++;

    // Check if the robot has completed 9-10 turns (360 degrees)
    if (turnCount >= 9) {
      turnCount = 0;
      forwardMoveCompleted = false;
    }
  }

  // Move forward a little bit after every 360-degree turn
  if (!forwardMoveCompleted && turnCount == 0) {
    motors.setSpeeds(FORWARD_SPEED, FORWARD_SPEED);
    delay(500);  // Adjust this delay to control the forward movement distance
    motors.setSpeeds(0, 0);
    forwardMoveCompleted = true;
  }
}
```

6.3 Border Detection

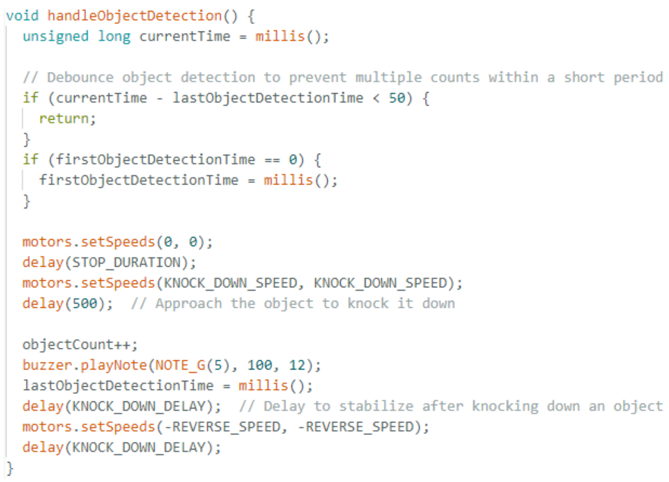
- Using reflectance sensors, the code checks if the zumo is about to fall down. If white ground is observed, zumo turns slightly to it's right or left (based on which sensor detected the white ground) and moves forward for 0.5 seconds.



```
void handleBorderDetection() {
  motors.setSpeeds(0, 0);
  if (sensor_values[0] < QTR_THRESHOLD) {
    turn(RIGHT, true);
    motors.setSpeeds(FORWARD_SPEED, FORWARD_SPEED);
    delay(500);
  } else if (sensor_values[5] < QTR_THRESHOLD) {
    turn(LEFT, true);
    motors.setSpeeds(FORWARD_SPEED, FORWARD_SPEED);
    delay(500);
  }
}
```

6.4 Obstacle Detection and Counting

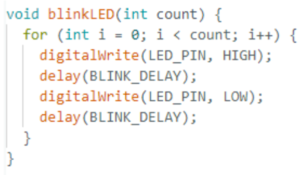
- If reflectance sensors doesn't observe white ground our zumo slightly moves forward and turns left continuously. When MZ80 infrared sensor detects an obstacle, zumo moves forward, knocks it down, counts the object (also makes a sound using buzzer) and moves backwards.



```
void handleObjectDetection() {
  unsigned long currentTime = millis();

  // Debounce object detection to prevent multiple counts within a short period
  if (currentTime - lastObjectDetectionTime < 50) {
    return;
  }
  if (firstObjectDetectionTime == 0) {
    firstObjectDetectionTime = millis();
  }

  motors.setSpeeds(0, 0);
  delay(STOP_DURATION);
  motors.setSpeeds(KNOCK_DOWN_SPEED, KNOCK_DOWN_SPEED);
  delay(500);  // Approach the object to knock it down

  objectCount++;
  buzzer.playNote(NOTE_G(5), 100, 12);
  lastObjectDetectionTime = millis();
  delay(KNOCK_DOWN_DELAY);  // Delay to stabilize after knocking down an object
  motors.setSpeeds(-REVERSE_SPEED, -REVERSE_SPEED);
  delay(KNOCK_DOWN_DELAY);
}
```

6.5 LED blinking

- If MZ80 infrared sensor doesn't detect any object for 45 seconds zumo stops moving and LED starts blinking equals to object counted.



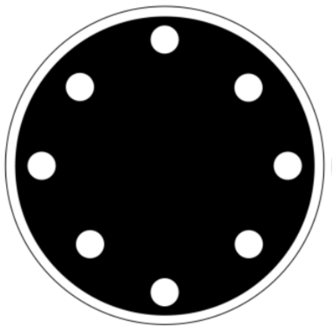
```
void blinkLED(int count) {
  for (int i = 0; i < count; i++) {
    digitalWrite(LED_PIN, HIGH);
    delay(BLINK_DELAY);
    digitalWrite(LED_PIN, LOW);
    delay(BLINK_DELAY);
  }
}
```

7 Flaws

1. Knocking down an object may cause another object to fall down resulting in not counting that object.
2. If current possition of the zumo is really far from other objects by any chance, MZ80 may not detect the other objects. In this case we would have to increase the range of the MZ80.

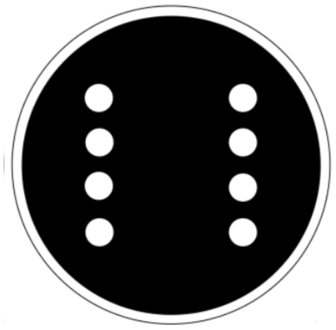
8 Experiments

1. First experiment have some objects close to the edges which could cause zumo to fall down.

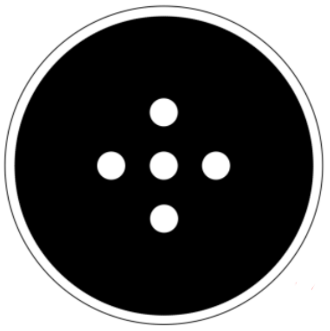
   Result: Our zumo counted every object without any problem

2. Experiment 2 places objects close to each other which could cause zumo to count them as same object.

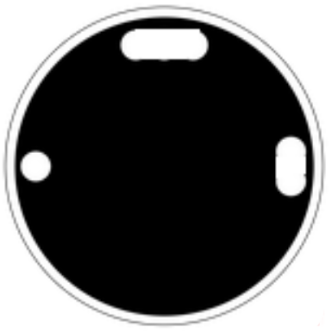
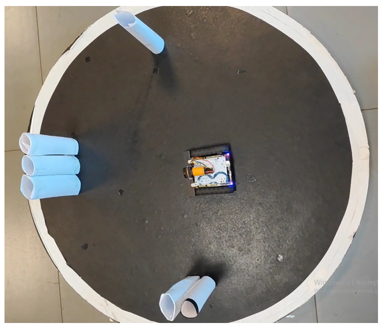
   Result: Our zumo counted every object without any problem.

3. Experiment 3 has an object at the center and 4 more objects around it. This makes it impossible to place zumo at the center which means an algorithm that makes zumo turn and count the objects it detects won't work. In order to, knock the center object out of the arena zumo shall knock some of the other objects too. This means an algorithm which knock objects and then counts them one by one won't work properly.

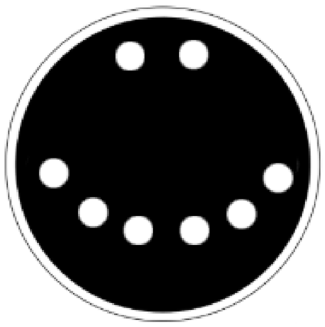
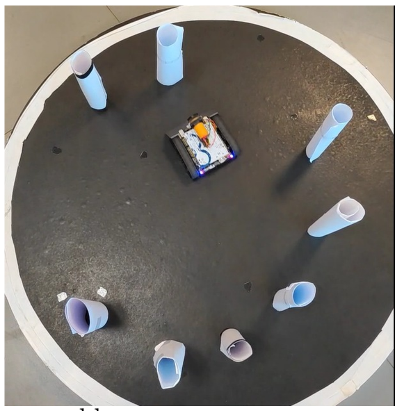
   Result: Our zumo couldn't count every object because When it knocks down the first object, another object also gets knocked down.

4. Experiment 4 has different sized object near to the edges which could cause counting the same object twice or more.

Result: Our zumo counted every object without any problem.

5. Experiment 5 places obstacles in a shape of smile.

Result: Our zumo counted every object without any problem.

9 Conclusion

1. In this project, we successfully programmed a Zumo Shield robot to navigate an arena, detect objects using the MZ80 infrared sensor, and count them using LED blinks. Through this process, we gained practical insights into robotics and embedded systems, particularly in the areas of sensor integration and algorithm development.

2. Our implementation met all the project requirements and passed the defined test cases. The robot demonstrated reliable performance in scanning its surroundings, accurately detecting and counting objects, and remaining within the arena's boundaries through the use of the Zumo Reflectance Sensor Array. We encountered challenges such as ensuring accurate object counting and effective border detection, which we addressed through careful calibration and robust algorithm design.

3. The experiments conducted highlighted the robustness of our solution in various scenarios, including handling objects near the edges, closely placed objects, and different sized objects. Although we identified some limitations, such as potential miscounts due to object displacement, the overall functionality of the Zumo robot was validated by its consistent performance across multiple tests.

4. In conclusion, this project not only achieved its technical goals but also provided valuable hands-on experience with the complexities of robotics programming. The successful completion of this project reflects our understanding and capability in working with embedded systems and paves the way for more advanced explorations in the field of robotics.