EE321 TERM PROJECT

FINAL REPORT

MAY 22, 2024

BERKE ÇEÇEN S029287 /GENCO GÜVEN S029392

1. Objectives:

Our main goal is to program our ZUMO robot to count the number of objects in the area that we put it in. The robot will constantly turn right and when an object is sensed, it will count the object. After counting the object, the robot will blink a LED (LED 13) and run towards the sensed object to knock it down. When the round is over, our robot is going to blink its LED as many as the number of objects it sensed.

2. Tools and Libraries:

We do not use any external tools in our project. We just used the infrared sensor (MZ80) to detect the objects and the reflectance sensor array which is located under the ZUMO robot to detect the borders of the area.

For libraries, we only used the libraries that are included in Zumo starter code. These libraries are QTRSensors, ZumoReflectanceSensorArray and ZumoMotors. The purposes of these libraries are as follows:

- QTRSensors: It is used for MZ80 infrared sensor readings.
- ZumoMotors: It is used for the movement of Zumo robot.
- ZumoReflectanceSensorArray: It is used for detecting the borders of the area.

3. Operation Principle:

Part 1 – Calibration

When the robot is first started, it will calibrate for 10 seconds. We wanted to keep this part a little bit longer than usual to eliminate any errors regarding the colors of the area. During these 10 seconds, we let our robot learn the black and white colors using its sensors underneath.

```
// -------------------------- Start Of The Calibration ------------------------
reflectanceSensors.init();
unsigned long startTime = millis();
while (millis() - startTime < 10000)  // make the calibration take 10 seconds
{
  reflectanceSensors.calibrate();
}
// ------------------ End Of The Calibration ----------------------
```

Part 2 – Fight Started

The code runs in to a loop after the calibration and at the start, the robot will go into a “Fight Started” phase. When the fight starts, the robot will do as follows:

Part 2.1 – Filtering

The robot can detect an object when the filter is “off”. When the robot detects an object, the filter turns on. When the filter turns on, the robot cannot detect any objects for 1 second. After that 1 second, the filter turns off automatically and the robot is ready to detect another object.

Part 2.2 – Bottom Sensors

If any one of the bottom sensors receive “1” as input, it means that our robot has arrived to the perimeter of the area. Therefore, the motor speeds will be set as 0 for both wheels. After that, there will be a small delay of 100 milliseconds before turning right for 500 milliseconds.

Part 2.3 – Sensing an Object

If none of the bottom sensors receive “1” as input, it means that our robot is still inside the black area. At the same time, if the robot detects an object (or MZ80 pin senses an object), it goes through that object and knocks it down. Knocking down the object is essential because with this implementation, we prevent counting the same object more than once.

Part 3 – Fight Ended

After 35 seconds, the robot will go into a “Fight Ended” phase. In this phase, the motor speed will be 0 for both wheels. Our robot will wait for 4 seconds and blink its LED as many as the number of objects it sensed. It will blink the LED in 1 second periods.

4. Flaws and Imperfections:

- Knocking down objects may be useful in many situations, but in this case our main objective is to count the number of objects in a specific area. Therefore, knocking down objects can sometimes make the robot’s job harder since those objects can fall into each other. When such a scenario happens, at least one object will go down before the robot can sense it.

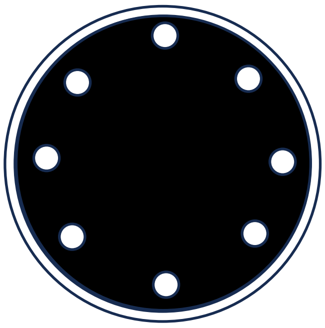
- The robot has some width. This width sometimes helps us in the knocking down stage. When the robot detects an object, it goes through that object for 1 second and knocks it down. However, maybe the object is no longer in the sight of MZ80 at some point inside that 1 second. In this case, with the help of the width of the robot, we could knock down the object and avoid counting that object twice.

- Our robot cannot do any image processing tasks. It means it works in a very simple way: If there is some obstacle in the range of MZ80, it sees it continuously. Therefore, if we put the objects linearly with respect to the sight of MZ80, it could not differ these objects and it may count this set of objects as 1 object.

- We set the filter to be on for 1 second. If the robot cannot knock down the object in 1 second, it turns right and tries to detect another object or the same object for the 2nd time. It causes some mistake in the “printing the results” stage.

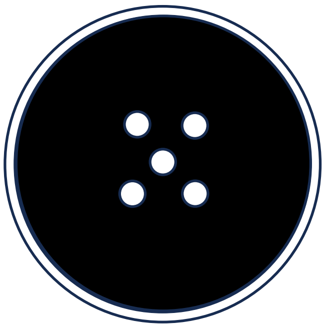
5. Testing:

Case 1 – *"Perimeter Defense"*



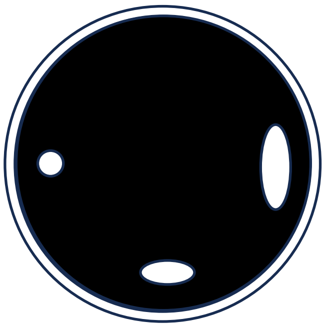
In this case, we put eight objects near the perimeter of our area. Our robot could count all of them properly.

Case 2 – *"X"*



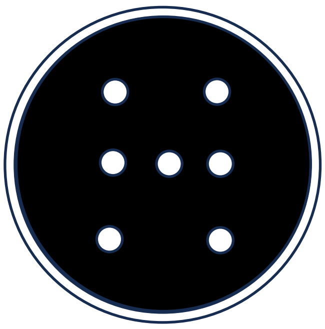
In this case, we put five objects on the center of our area making them form an "X" shape. Our robot could count all of them properly.

Case 3 – *"Small, Medium, Large"*



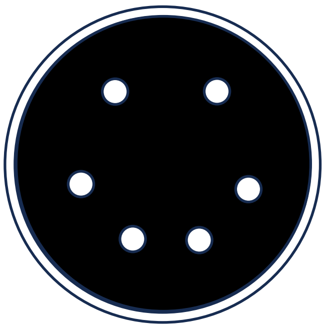
In this case, we put three objects in different sizes. We only had one type of object when working in lab so we put the same type of objects side by side in order to make the surface area bigger. Our robot could count all of them properly.

Case 4 – *"The Helipad"*



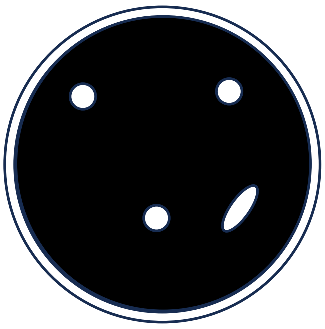
In this case, we put seven objects on the area forming an "H" shape just like a helipad. Our robot could count all of them properly.

Case 5 – *"The Smiley Face"*



In this case, we put six objects on the area forming a smiley face. Our robot could count all of them properly.

Case 6 – *"The Random"*



In this case, we put 4 objects on the area randomly. One of them was bigger than others. Our robot could count all of them properly.

6. Coding:

The architecture of our code is here.

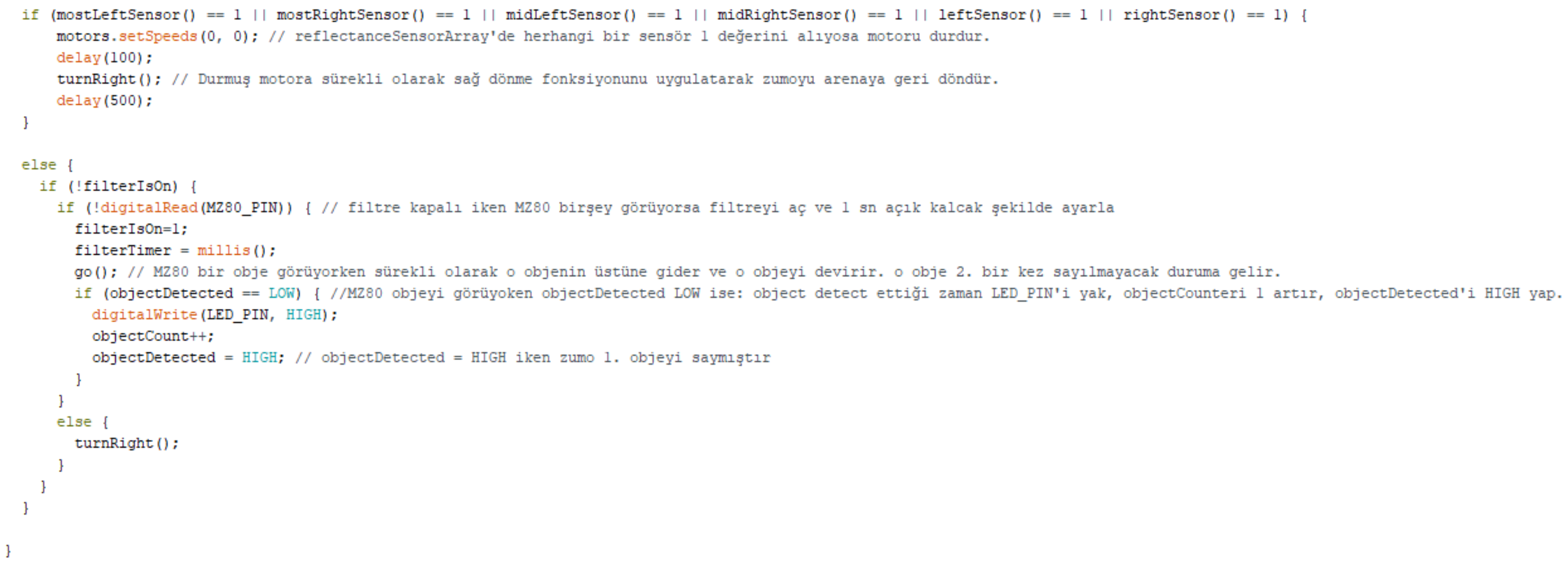
```
#include <QTRSensors.h>
#include <ZumoReflectanceSensorArray.h>
#include <ZumoMotors.h>


#define LED_PIN 13
#define MZ80_PIN 6
#define NUM_SENSORS 6
#define FIGHT_STARTED 1
#define FIGHT_ENDED 0

//Pushbutton button(ZUMO_BUTTON);
ZumoReflectanceSensorArray reflectanceSensors;
ZumoMotors motors;

// Define an array for holding sensor values.
unsigned int sensorValues[NUM_SENSORS];
unsigned int positionVal = 0;
unsigned int objectCount = 0;
bool objectDetected = LOW;
bool fightStatus = FIGHT_STARTED;
bool fightResultsShown = 0;

void setup() {

  pinMode(LED_PIN, OUTPUT);
  digitalWrite(LED_PIN, HIGH);

  // --------------------------- Start Of The Calibration -------------------------
  reflectanceSensors.init();
  unsigned long startTime = millis();
  while (millis() - startTime < 10000)  // make the calibration take 10 seconds
  {
    reflectanceSensors.calibrate();
  }
  // ------------------- End Of The Calibration -----------------------

  digitalWrite(LED_PIN, LOW);
}
```

Here is the setup () part of the code. When you run the code, setup runs for only one time.

```
unsigned int mostLeftSensor() {
  if (sensorValues[0] < 600)
    return 1;
  else
    return 0;
}

unsigned int leftSensor() {
  if (sensorValues[1] < 600)
    return 1;
  else
    return 0;
}

unsigned int midLeftSensor() {
  if (sensorValues[2] < 600)
    return 1;
  else
    return 0;
}

unsigned int midRightSensor() {
  if (sensorValues[3] < 600)
    return 1;
  else
    return 0;
}

unsigned int rightSensor() {
  if (sensorValues[4] < 600)
    return 1;
  else
    return 0;
}

unsigned int mostRightSensor() {
  if (sensorValues[5] < 600)
    return 1;
  else
    return 0;
}
```

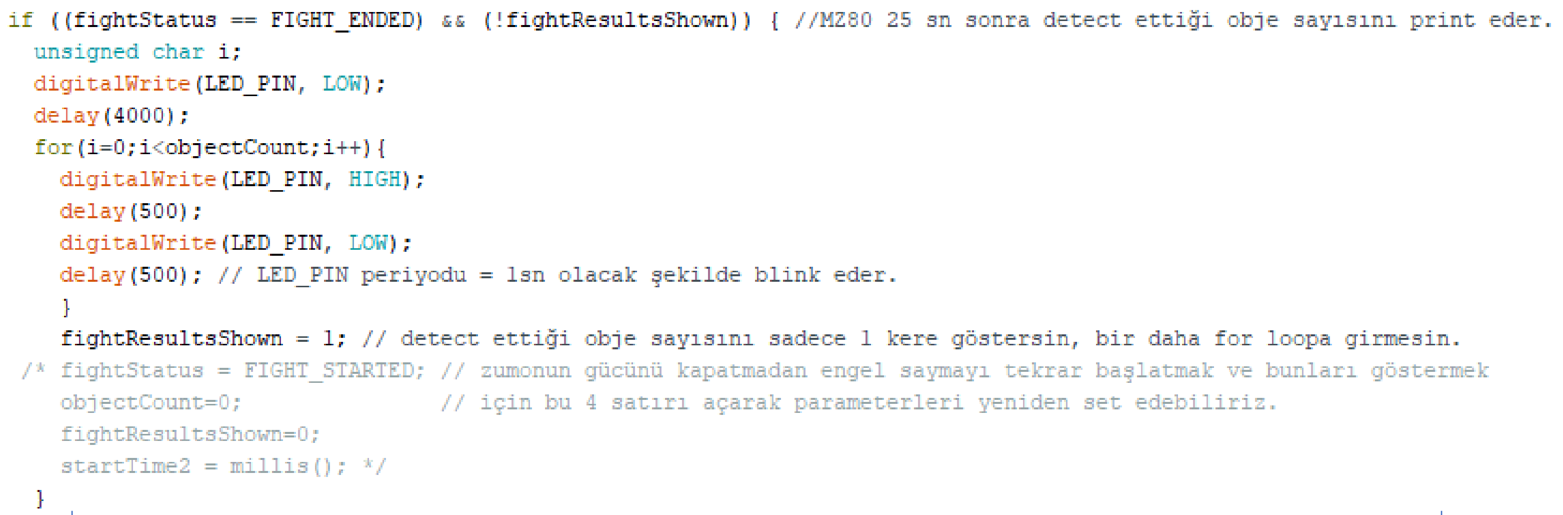
Setup of the sensors on the ReflectanceSensorArray.

```
void turnRight() {
  motors.setSpeeds(200, -200);
}

void go() {
  motors.setSpeeds(400, 400);
}

unsigned long startTime2 = millis();
bool filterIsOn=0;
unsigned long filterTimer;

void loop() {

if (fightStatus == FIGHT_STARTED){

  positionVal = reflectanceSensors.readLine(sensorValues);

  if (filterIsOn) { // filtre açık: false positive'leri filtreler.
    if ((millis()-filterTimer) > 1000){ //1sn boynca filtre açıktır. 1sn sonunda kapanacaktır
      filterIsOn=0; // filtre kapalı
      if (!digitalRead(MZ80_PIN)) { // hala bir object görüyorsam bir şey yapma
      }
      else { // görmüyorum
        digitalWrite(LED_PIN, LOW); //MZ80 objeyi görmezken LED_PIN'i söndür, objectDetected'i LOW yap.
        objectDetected = LOW; // objectDetected = LOW iken zumo 2. objeyi detect etmeye hazırdır.
        turnRight(); // Zumo sürekli sağ dönerek yeni bir obje bulmaya çalışır.
      }
    }
  }
```



```
 if (mostLeftSensor() == 1 || mostRightSensor() == 1 || midLeftSensor() == 1 || midRightSensor() == 1 || leftSensor() == 1 || rightSensor() == 1) {
    motors.setSpeeds(0, 0); // reflectanceSensorArray'de herhangi bir sensör 1 değerini alıyosa motoru durdur.
    delay(100);
    turnRight(); // Durmuş motora sürekli olarak sağ dönme fonksiyonunu uygulatarak zumoyu arenaya geri döndür.
    delay(500);
 }

 else {
   if (!filterIsOn) {
     if (!digitalRead(MZ80_PIN)) { // filtre kapalı iken MZ80 birşey görüyorsa filtreyi aç ve 1 sn açık kalcak şekilde ayarla
       filterIsOn=1;
       filterTimer = millis();
       go(); // MZ80 bir obje görüyorken sürekli olarak o objenin üstüne gider ve o objeyi devirir. o obje 2. bir kez sayılmayacak duruma gelir.
       if (objectDetected == LOW) { //MZ80 objeyi görüyoken objectDetected LOW ise: object detect ettiği zaman LED_PIN'i yak, objectCounteri 1 artır, objectDetected'i HIGH yap.
         digitalWrite(LED_PIN, HIGH);
         objectCount++;
         objectDetected = HIGH; // objectDetected = HIGH iken zumo 1. objeyi saymıştır
       }
     }
     else {
       turnRight();
     }
   }
 }

}
```

Everything about the task is happening inside of the void loop() function.



```
if ((fightStatus == FIGHT_ENDED) && (!fightResultsShown)) { //MZ80 25 sn sonra detect ettiği obje sayısını print eder.
  unsigned char i;
  digitalWrite(LED_PIN, LOW);
  delay(4000);
  for(i=0;i<objectCount;i++){
    digitalWrite(LED_PIN, HIGH);
    delay(500);
    digitalWrite(LED_PIN, LOW);
    delay(500); // LED_PIN periyodu = 1sn olacak şekilde blink eder.
    }
    fightResultsShown = 1; // detect ettiği obje sayısını sadece 1 kere göstersin, bir daha for loopa girmesin.
 /* fightStatus = FIGHT_STARTED; // zumonun gücünü kapatmadan engel saymayı tekrar başlatmak ve bunları göstermek
    objectCount=0;              // için bu 4 satırı açarak parametreleri yeniden set edebiliriz.
    fightResultsShown=0;
    startTime2 = millis(); */
  }
```

This part of code is responsible for the printing the number of objects in the area.

7. YouTube Video:

Here is the YouTube video link of our term project. The video includes our presentation, project demo and test cases.

https://youtu.be/   rN17pKsAk