Lab 4: Braitenberg Vehicles, Meta-Sensing and Randomness

ECE 564: Fundamentals of Autonomous Robots Lab

Team 1: Jacob Cassady, Chase Crutcher, and Olalekan Olakitan Olowo

September 11, 2019

The group members have worked together and face-to-face at all stages of this project work.
The contributions of members to the report and to the codes are equal.
(Initials of group members)

Introduction

The goal of this lab is to make the Demobot be attracted to light with the use of two light sensors on either side of the robot. Local variables are used to keep track of internal state and functions are used to prevent the demobot from getting stuck in an emergent loop. The functionality of these sensors should measure the amount of light hitting the sensors alongside the robot's ability to avoid obstacles as demonstrated in the previous lab project.

Part 1: Light Sensors

The light sensors were placed on the left and right side of the Demobot and connected to the analog input 0 and 1 to act as an eye so as to detect infrared light. This will allow the Demobot to model several different behaviors like hiding in the dark, staying in the light, and following a flashlight.

Part 1.1) Code

```
void displayAnalogReadings(int lightSensor1, int lightSensor2) {
    int lightSensor1Reading = 0;
    int lightSensor2Reading = 0;

    while(1) {
      lightSensor1Reading = analog(lightSensor1);
      lightSensor2Reading = analog(lightSensor2);
      printf("Sensor 1 [%d]: %d | Sensor 2 [%d]: %d\n", lightSensor1, lightSensor1Reading, lightSensor2, lightSensor2Reading);
      wait_for_milliseconds(1000);
    }
}
```

Part 1.2) The sensor readings get larger with less light and gets smaller in magnitude with less light.

Part 1.3) The sensors do not often given the same reading. Each sensor's reading depends on the angle of the nearest light source.

Part 1.4) For a tested light sensor the minimum analog value was 157 and the max was 4058. Typical readings for this light sensor at ambient light levels are between 1800 and 2400.

Part 1.5) The sensors will provide similar readings although the offset of each might be slightly different. This can be mitigated by normalizing the data.

Part 2: Shielding Light Sensors

Part 2.1) When a flashlight is pointed directly at each sensor. The left sensor produces an analog value of 188 and right sensor produces an analog value of 216.

Part 2.2)

Angle	Left Sensor	Right Sensor
10	1422	190
20	1919	207
45	2250	230
90	3350	275

As you can see from the table above, as the light rotates 90 degrees towards the right light sensor, you can see the left light sensor is receiving less light. As a consequence, the left sensor value raises with angle while remaining relatively constant with respect to the right sensor.

Part 3: Normalizing Light Readings to Motor Commands

```
int NormalizeLightReadings(int lightValue, int minLightValue, int maxLightValue) {
    // Apply transform
    int output = 100 - ((lightValue - maxLightValue) * 100) / (minLightValue - maxLightValue);

    // Return value based on range
    if (output < 0) {
      return 0;
    } else if (output > 100) {
      return 100;
    } else {
      return output;
    }
}
```

Part 3.1) The min and max values used when normalizing light readings were gathered experimentally. A table of the min and max values used when normalizing light readings can be found in the table below.

	Left Light Sensor	Middle (Ambient) Light Sensor	Right Light Sensor
min	260	236	220
max	4050	4080	4036

Part 3.2) The function given in the lab does work; the range of values that was produced was 0 to 100.

Part 3.3) A value outside the min and max experimental values is received.

Part 4: Light - Seeking

Part 4.1) Code - Seek Light

A program was written to seek the light, and the actions of the robot were recorded on video. Light values are taken by the two “eyes” (light sensors), normalized, and fed to the motors. This is the point at which the note mentioned in Part 3, question 1 is required. Higher physical light levels result in lower light values obtained from the sensors meaning that after normalization, a sensor in direct light will output a low power value and a sensor in the shade outputs a large power value. Sending these powers directly to the motors makes the robot turn toward the light. Additionally, a default speed was chosen for when the equation output drops below a certain value. This ensures that the robot will move toward the light source when in direct light on both sensors. The code segment describing this behavior for the robot is provided below.

```
void seekLight(int leftMotor, int rightMotor, int leftLightSensor, int rightLightSensor, int ambientLightSensor) {
  // Initialize variables
  int leftSensorReading = 0;
  int rightSensorReading = 0;

  while(1) {
    // get reading from sensors
    leftSensorReading = NormalizeLightReadings(analog(leftLightSensor), MINLEFTLIGHTREADING, MAXLEFTLIGHTREADING);
    rightSensorReading = NormalizeLightReadings(analog(rightLightSensor), MINRIGHTLIGHTREADING, MAXRIGHTLIGHTREADING);

    // Move demobot in response to the sensor values
    goDemobotMav(leftMotor, rightMotor, 10, leftSensorReading*SPEEDFACTOR, rightSensorReading*SPEEDFACTOR);
  }
}
```

Part 4.2) Code - Avoid Light

The program used to achieve a light seeking behavior was adapted to make the robot avoid light. This was achieved by subtracting the normalized power level from 100 effectively mirroring the power level on the 0-100 scale. Again, the code segment written for this behavior is provided below.

```
void seekDarkness(int leftMotor, int rightMotor, int leftLightSensor, int rightLightSensor, int ambientLightSensor) {
  int leftSensorReading = 0;
  int rightSensorReading = 0;

  while(1) {
    // get reading from sensors
    leftSensorReading = 100 - NormalizeLightReadings(analog(leftLightSensor), MINLEFTLIGHTREADING, MAXLEFTLIGHTREADING);
    leftSensorReading -= NormalizeLightReadings(analog(ambientLightSensor), MINAMBIENTLIGHTREADING, MAXAMBIENTLIGHTREADING);
    rightSensorReading = 100 - NormalizeLightReadings(analog(rightLightSensor), MINRIGHTLIGHTREADING, MAXRIGHTLIGHTREADING);
    rightSensorReading -= NormalizeLightReadings(analog(ambientLightSensor), MINAMBIENTLIGHTREADING, MAXAMBIENTLIGHTREADING);
    printf("Left Light Sensor : %d | Right Light Sensor 2 : %d\n", leftSensorReading, rightSensorReading);

    goDemobotMav(leftMotor, rightMotor, 10, leftSensorReading*SPEEDFACTOR, rightSensorReading*SPEEDFACTOR);
  }
}
```

Part 4.3)

Rather than directly feeding the normalized light values into the motor() command, a third light sensor pointed directly upward with no shielding was added to the robot. This sensor was used to take an average ambient light value. Readings from each “eye” were then compared to this value to determine a magnitude difference between the “eye” reading and the ambient light. In this way, the robot was able to determine more accurately if a sensor was obtaining light from a separate light source. The robot’s behavior was observed and demonstrated. The code is provided below.

```
void seekLight(int leftMotor, int rightMotor, int leftLightSensor, int rightLightSensor, int ambientLightSensor) {
  // Initialize variables
  int leftSensorReading = 0;
  int rightSensorReading = 0;

  while(1) {
    // get reading from sensors
    leftSensorReading = NormalizeLightReadings(analog(leftLightSensor), MINLEFTLIGHTREADING, MAXLEFTLIGHTREADING);
    rightSensorReading = NormalizeLightReadings(analog(rightLightSensor), MINRIGHTLIGHTREADING, MAXRIGHTLIGHTREADING);
    // Subtract ambient light sensor value
    leftSensorReading -= NormalizeLightReadings(analog(ambientLightSensor), MINAMBIENTLIGHTREADING, MAXAMBIENTLIGHTREADING);
    rightSensorReading -= NormalizeLightReadings(analog(ambientLightSensor), MINAMBIENTLIGHTREADING, MAXAMBIENTLIGHTREADING);
    printf("Left Light Sensor : %d | Right Light Sensor 2 : %d\n", leftSensorReading, rightSensorReading);

    // Move demobot in response to the sensor values
    goDemobotMav(leftMotor, rightMotor, 10, leftSensorReading*SPEEDFACTOR, rightSensorReading*SPEEDFACTOR);
  }
}
```

```
void seekDarkness(int leftMotor, int rightMotor, int leftLightSensor, int rightLightSensor, int ambientLightSensor) {
  int leftSensorReading = 0;
  int rightSensorReading = 0;

  while(1) {
    // get reading from sensors
    leftSensorReading = 100 - NormalizeLightReadings(analog(leftLightSensor), MINLEFTLIGHTREADING, MAXLEFTLIGHTREADING);
    rightSensorReading = 100 - NormalizeLightReadings(analog(rightLightSensor), MINRIGHTLIGHTREADING, MAXRIGHTLIGHTREADING);
    // Subtract ambient light sensor value
    leftSensorReading -= NormalizeLightReadings(analog(ambientLightSensor), MINAMBIENTLIGHTREADING, MAXAMBIENTLIGHTREADING);
    rightSensorReading -= NormalizeLightReadings(analog(ambientLightSensor), MINAMBIENTLIGHTREADING, MAXAMBIENTLIGHTREADING);

    goDemobotMav(leftMotor, rightMotor, 10, leftSensorReading*SPEEDFACTOR, rightSensorReading*SPEEDFACTOR);
  }
}
```

Part 5: Light and Touch Sensitivity

The code written and pasted below allows the robot to avoid obstacles in addition with the light seeking program. In other words this allows the robot to both seek light and avoid obstacle at the same time.

```
void seekLightAndAvoidObstacles(int leftMotor, int rightMotor, int leftLightSensor, int rightLightSensor, int ambientLightSensor,
                                int leftTouchSensor, int rightTouchSensor) {
  int leftSensorReading = 0;
  int rightSensorReading = 0;
  int leftTouchSensorReading = 0;
  int rightTouchSensorReading = 0;

  while(1) {
    // get reading from sensors
    leftTouchSensorReading = digital(leftTouchSensor);
    rightTouchSensorReading = digital(rightTouchSensor);
    leftSensorReading = NormalizeLightReadings(analog(leftLightSensor), MINLEFTLIGHTREADING, MAXLEFTLIGHTREADING);
    rightSensorReading = NormalizeLightReadings(analog(rightLightSensor), MINRIGHTLIGHTREADING, MAXRIGHTLIGHTREADING);
    // Subtract ambient light sensor value
    leftSensorReading -= NormalizeLightReadings(analog(ambientLightSensor), MINAMBIENTLIGHTREADING, MAXAMBIENTLIGHTREADING);
    rightSensorReading -= NormalizeLightReadings(analog(ambientLightSensor), MINAMBIENTLIGHTREADING, MAXAMBIENTLIGHTREADING);

    if(leftTouchSensorReading) {
      // Back up
      goStraightMav(leftMotor, rightMotor, 1000, -500);
      // Turn Right 90
      turnRight(leftMotor, rightMotor);
      // Go straight
      goStraightMav(leftMotor, rightMotor, 500, 500);
      // Turn left
      turnLeft(leftMotor, rightMotor);
```

```
    } else if (rightTouchSensorReading) {
      // Back up
      goStraightMav(leftMotor, rightMotor, 1000, -500);
      // Turn left 90
      turnLeft(leftMotor, rightMotor);
      // Go straight
      goStraightMav(leftMotor, rightMotor, 500, 500);
      // Back up
      turnRight(leftMotor, rightMotor);
    } else {
      // Go straight
      goDemobotMav(leftMotor, rightMotor, 50, leftSensorReading*SPEEDFACTOR, rightSensorReading*SPEEDFACTOR);
    }
  }
}
```

Part 6: Randomness

Adding the “get out of the jar” procedure, random turn routine was created, allowing the robot to get out of unproductive loop randomly selecting to turn right or left, then execute the turn. The procedure enabled the robot to get out of any situation where it was stuck by randomly stopping, backup and taking a left or right turn without any intervention.
The code for this behaviour is provided below

```
void getOutOfJar(int leftMotor, int rightMotor) {
  goStraightMav(leftMotor, rightMotor, 0, random(1000));
}
```

```
void randomSeekLight(int leftMotor, int rightMotor, int leftLightSensor, int rightLightSensor, int ambientLightSensor,
                     int leftTouchSensor, int rightTouchSensor) {
  int leftSensorReading = 0;
  int rightSensorReading = 0;
  int leftTouchSensorReading = 0;
  int rightTouchSensorReading = 0;
  int leftTouchPrevious = 0;
  int rightTouchPrevious = 0;

  while(1) {
    // get reading from sensors
    leftTouchSensorReading = digital(leftTouchSensor);
    rightTouchSensorReading = digital(rightTouchSensor);
    leftSensorReading = NormalizeLightReadings(analog(leftLightSensor), MINLEFTLIGHTREADING, MAXLEFTLIGHTREADING);
    rightSensorReading = NormalizeLightReadings(analog(rightLightSensor), MINRIGHTLIGHTREADING, MAXRIGHTLIGHTREADING);
    // Subtract ambient light sensor value
    leftSensorReading -= NormalizeLightReadings(analog(ambientLightSensor), MINAMBIENTLIGHTREADING, MAXAMBIENTLIGHTREADING);
    rightSensorReading -= NormalizeLightReadings(analog(ambientLightSensor), MINAMBIENTLIGHTREADING, MAXAMBIENTLIGHTREADING);

    if(leftTouchSensorReading) {
      // Back up
      goStraightMav(leftMotor, rightMotor, 1000, -500);
      // Turn Right 90
      turnRight(leftMotor, rightMotor);
      // Go straight
      goStraightMav(leftMotor, rightMotor, 500, 500);
      // Turn left
      turnLeft(leftMotor, rightMotor);
```

```
    // If leftTouchSensor has not recently been touched.
    if (!leftTouchPrevious) {
      // Left touch is now denoted as previous.  Right is enforced not to be previous.
      leftTouchPrevious = 1;
      rightTouchPrevious = 0;
    } else {
      // If the leftTouchSensor has previously been touched, get out of jar.
      getOutOfJar(leftMotor, rightMotor);
      leftTouchPrevious = 0;
    }
  } else if (rightTouchSensorReading) {
    // Back up
    goStraightMav(leftMotor, rightMotor, 1000, -500);
    // Turn left 90
    turnLeft(leftMotor, rightMotor);
    // Go straight
    goStraightMav(leftMotor, rightMotor, 500, 500);
    // Back up
    turnRight(leftMotor, rightMotor);

    // If rightTouchSensor has not recently been touched.
    if (!rightTouchPrevious) {
      // Right touch is now denoted as previous.  Left is enforced not to be previous.
      rightTouchPrevious = 1;
      leftTouchPrevious = 0;
    } else {
      // If the rightTouchSensor has previously been touched, get out of jar.
      getOutOfJar(leftMotor, rightMotor);
      rightTouchPrevious = 0;
    }
  } else {
    // Move towards light
    goDemobotMav(leftMotor, rightMotor, 50, leftSensorReading*SPEEDFACTOR, rightSensorReading*SPEEDFACTOR);
  }
}
```

Part 7: Meta-Sensing

The thread command was introduced here. First, the thread was created for the light seeking capability and then for obstacle avoidance behavior and initiated for a particular duration after which it was then terminated.

```
void metaSensing() {
    // create threads
    lightThread = thread_create(senseLight);
    touchThread = thread_create(feelObjects);

    // start threads
    thread_start(lightThread);
    thread_start(touchThread);

    // Run for 5 minutes
    wait_for_milliseconds(60000 * 5);

    // Destroy threads
    thread_destroy(lightThread);
    thread_destroy(touchThread);
}
```

```
void senseLight() {
    int leftLightSensor = 2;
    int ambientLightSensor = 3;
    int rightLightSensor = 4;
    int leftSensorReading = 0;
    int rightSensorReading = 0;

    while(1) {
        // WHile not dealing with any touch sensors. dealingWithTouchSensor is a global variable.
        while(!dealingWithTouchSensor) {
            // get sensor readings
            leftSensorReading = NormalizeLightReadings(analog(leftLightSensor), MINLEFTLIGHTREADING, MAXLEFTLIGHTREADING);
            rightSensorReading = NormalizeLightReadings(analog(rightLightSensor), MINRIGHTLIGHTREADING, MAXRIGHTLIGHTREADING);
            // Subtract ambient light sensor value
            leftSensorReading -= NormalizeLightReadings(analog(ambientLightSensor), MINAMBIENTLIGHTREADING, MAXAMBIENTLIGHTREADING);
            rightSensorReading -= NormalizeLightReadings(analog(ambientLightSensor), MINAMBIENTLIGHTREADING, MAXAMBIENTLIGHTREADING);

            // Move towards light
            goDemobotMav(leftMotor, rightMotor, 10, leftSensorReading*SPEEDFACTOR, rightSensorReading*SPEEDFACTOR);
        }
    }
}
```

```
void feelObjects() {
    int leftTouchSensor = 0;
    int rightTouchSensor = 1;
    int leftTouchSensorReading = 0;
    int rightTouchSensorReading = 0;

    while(1) {
        // Get sensor readings
        leftTouchSensorReading = digital(leftTouchSensor);
        rightTouchSensorReading = digital(rightTouchSensor);

        if(leftTouchSensorReading) {
          dealingWithTouchSensor = 1;
          // Back up
          goStraightMav(leftMotor, rightMotor, 1000, -500);
          // Turn Right 90
          turnRight(leftMotor, rightMotor);
          // Go straight
          goStraightMav(leftMotor, rightMotor, 500, 500);
          // Turn left
          turnLeft(leftMotor, rightMotor);
          dealingWithTouchSensor = 0;
        } else if (rightTouchSensorReading) {
          dealingWithTouchSensor = 1;
          // Back up
          goStraightMav(leftMotor, rightMotor, 1000, -500);
          // Turn left 90
          turnLeft(leftMotor, rightMotor);
          // Go straight
          goStraightMav(leftMotor, rightMotor, 500, 500);
          // Back up
          turnRight(leftMotor, rightMotor);
          dealingWithTouchSensor = 0;
        }
    }
```

Conclusion

During this lab, the team added functionality to the DemoBot by adding 3 light sensors that give the robot the ability to follow or flee from a light source. The concept of threads was also introduced to seemingly run segments of the program concurrently. While this portion of the lab proved to be problematic, the idea and implementation based on examples provided is clear.

The main issues encountered in this lab were mounting and shielding of the light sensors. A good shield for the light sensors that would nearly completely separate the two “eyes” and the ambient light sensor was difficult to devise. The team achieved this using standard Lego blocks available. A more prevalent issue was experienced when trying to implement the thread commands to perform the task in the last part of the lab as described in Part 7.

Through this lab, the team has gained further knowledge on including light sensors in a system, using an analog sensor to drive motor controls by normalizing the readings, and the team has discovered that more research into the thread implementation on the wallaby is needed for it to perform properly.

Suggestions

It is suggested that a directed light source be provided for the lab. Not many students have a nonLED flashlight, and it was more difficult to test the robot with a large dish shaped lamp that was in the lab.