MREN 103 – Mechatronics Design I
Mechatronics and Robotics Engineering Program
Faculty of Engineering and Applied Science, Queen's University, Kingston

Group #2
Stage #2 – Pickup and Drop-Off Cycle

Student Names: Hannah Vibien and Coleman Farvolden
Instructor Name: Prof. Surgenor
Date Report Submitted: April 9th, 2023

Summary:
A bucket system attached to a prewired MiniBot from previous labs was used in Task 2. Activities included: a) using a sharp sensor, b) completing a 180-degree turn, c) line following, d) dropping off wire caps, e) picking up wire caps. The objective was to detect the wall, then turn 180 degrees, pick up the wire caps, then move forward until another wall is detected then dump the pieces. This process must be repeated 3 times in a row. These steps, as shown in Figure 1, should be repeated indefinitely. The testing setup is shown in Figure 2.

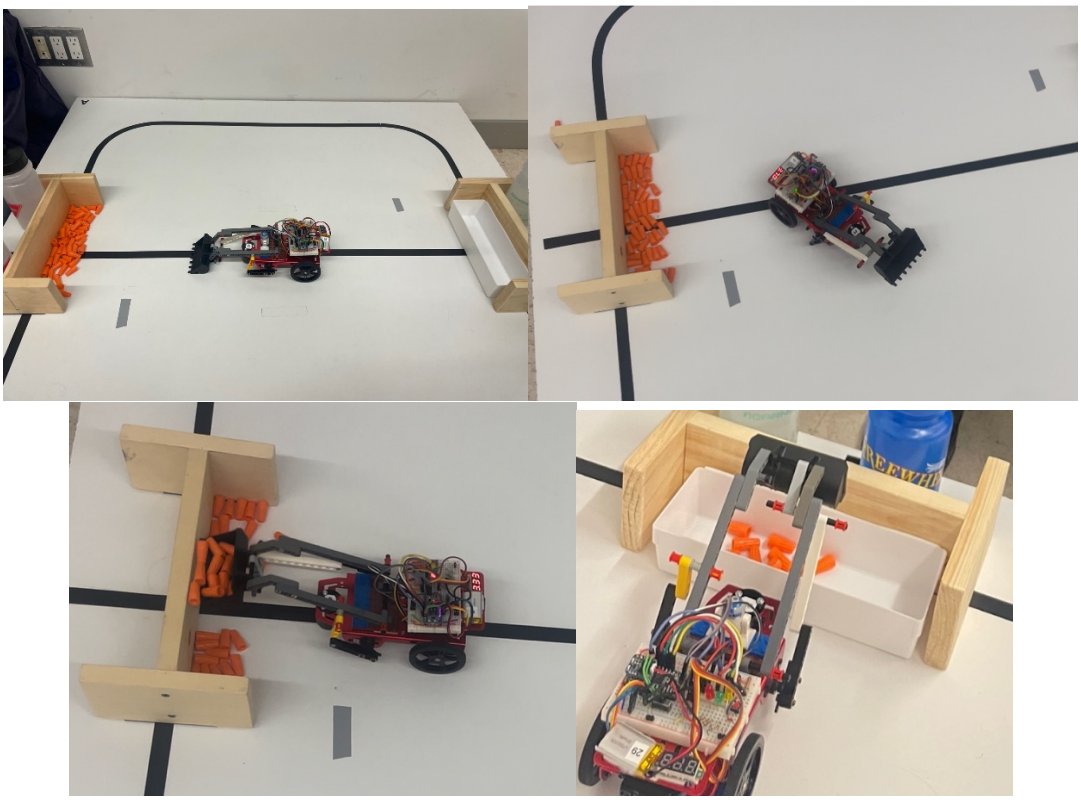

Figure 1: Order of images shows path the robot takes left to right, top to bottom.

From this lab Group 2 learned how important it is to hardcode turns, as settings delays were not reliable due to the changing voltages during runtime. Group 2 also saw how the weight of the wire caps, even though it is minuscule can affect, the MiniBots movements. The bucket pickup method was also essential to getting enough pieces and had to be reprogrammed a few times.

Software:
A listing of the program ***Group2ProjectLoaderStage2*** used in the lab is given in **Appendix A**.

This program continuously follows the line, the miniBot moves forward until its's about 8 cm away from the blockades. Then the miniBot turns 180 degrees, by pivoting for a certain amount of time. Then the miniBot moves towards the detected object and changes between picking up the blocks and dumping them off. This happens continuously and the miniBot moves back and forth repeating the steps above but always rotating between picking up and dropping off blocks using a code flag.

Discussion*:*
The biggest problem Group 2 encountered was when turning 180 degrees that by the last round it would not turn enough because the battery voltage would have decreased meaning the delay was not long enough to complete the turn, as shown in Figure 2. This meant that we had to overcorrect the first 180 turn to turn farther than 180, so that by the time it got to the third pick up the MiniBot would still turn enough. Group 2 also encountered the problem that after it had picked up the pieces, it was heavier and did not turn as far, so we had to increase the delay for turning when dropping off blocks. This meant that our code was not very reliable, which is why for Task 3 Group 2 ended up hardcoding the 180-degree turn, so that it did not rely on the battery voltage or the heaviness of the bucket.

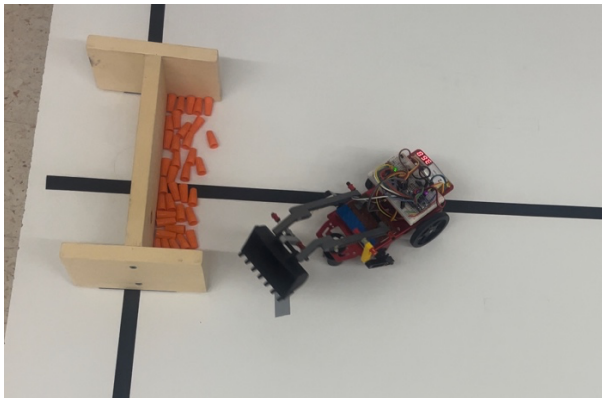

Figure 2: Shows the MiniBot not completing the 180-degree turn

Appendix A - Program Listing:

```
/**********************************************
***
Group2ProjectLoaderStage2

 By Coleman Farvolden and Hannah Vibien 09/04/2023


Lift and then lower the bucket once. Start with
bucket resting (at an angle) touching ground, drive
link parallel to the ground.
**********************************************
****/

#include <Servo.h>  // Includes the library
Servo myServoA;  // Makes a servo object to control servo A
Servo myServoB;// Makes a servo object to control servo B
Servo leftWheel;
Servo rightWheel;
// Pin Assignments
const int LSENSOR = A1; // Left Sensor on Analog Pin 1
const int RSENSOR = A2; // Right Sensor on Analog Pin 2
float Lvoltage = analogRead(A1);
float Rvoltage = analogRead(A2);
const boolean PLOT = false;  //true=plot sensor reading;
false=serial monitor output.

int MOTOR_R = 3;       // right motor signal pin
int MOTOR_L = 4;       // left motor signal pin

const int stopPulse = 148;  // stop speed for motors (default
= 150))
const int delta = 15;      // pulse differential (default = 15)
const int offset = 0;      // offset, slows left wheel (default =
0)

//global variables
int lvalue = 0;  //left sensor value
int rvalue = 0;  //right sensor value

// Pin Assignments
int SHARP = A3;    // Sharp Sensor on Analog Pin 3
int value = 0;
int mv_value = 0;


// Pin Assignments
int GRN = 9;          // Green LED Pin
int YLW = 5;          // Yellow LED Pin
int RED = 10;         // Red LED Pin
int BUTTON = 7;       // Pushbutton Pin

int servoPinA = 11;    // Bucket servomotor #1 pin
int myAngleA1 = 155;   // initial angle, bucket lifts off ground
if too high
int posA = myAngleA1;   // if set to 180, bucket lifts robot off
of ground
int myAngleA2 = 110;    // highest angle (lift), puts almost
straight, set to 110
                //    still bent (i.e. not as high)

// this code is added
int servoPinB= 12;
int myAngleB1 = 100;
int posB = myAngleB1;
int myAngleB2 = 130;
int flag;

// Set-up routine
void setup() {

// Set-up LED pins as outputs
  pinMode(GRN, OUTPUT);
  pinMode(YLW, OUTPUT);
  pinMode(RED, OUTPUT);

// Set-up button pin as input
  pinMode(BUTTON, INPUT);

// Set-up servo motors
  myServoA.write(posA);       // Servo A starting position
  myServoA.attach(servoPinA);   // Attaches the servo to the
servo object

  //this code is new
  myServoB.write(posB);       // Servo B starting position
  myServoB.attach(servoPinB);   // Attaches the servo to the
servo object
  // Initialize line following sensor pins as inputs
  pinMode(LSENSOR, INPUT);
  pinMode(RSENSOR, INPUT);

  //initialize motor control pins as servo pins
  leftWheel.attach(MOTOR_L);
  rightWheel.attach(MOTOR_R);

// Initialize led pins as outputs.
  pinMode(GRN, OUTPUT);
  pinMode(YLW, OUTPUT);
  pinMode(RED, OUTPUT);

// Initialize pins as inputs
  pinMode(BUTTON, INPUT);
  pinMode(SHARP, INPUT);

// Initialize serial printout
  Serial.begin(9600);    // default 9600
  turnOnLED(YLW);
  digitalWrite(YLW, LOW);
  digitalWrite(GRN, HIGH);

   raise();

   flag = 1;

}


// Main routine
void loop() {
       //read the sensor value
     lvalue = analogRead(LSENSOR);
     rvalue = analogRead(RSENSOR);

        //map the values into millivolts (assuming 3000 mV
reference voltage)
     lvalue = map(lvalue,0,1023,0,3000);
     rvalue = map(rvalue,0,1023,0,3000);

//add the reading value
```

```
//Serial.println(mv_value);

value = analogRead(SHARP);

   mv_value = map(value,0,1023,0,3300); //convert AtoD
count to millivolts

   Serial.println(mv_value);


//When MiniBot is close to pickup ore dropoff
if(mv_value>1200){


//Robot is dropping off pieces when flag is one
if(flag == 1){
//Stops and turns 180 degrees
 runMotors(0,0);
  turn180();
    runMotors(0,0);

//line follows until straight on lined
  while(lvalue>600){
lineFollow();
}

//goes in reverse and dumps off pieces
   runMotors(0,0);
      lower();
   delay(1000);
   runMotors(-16,-16);
   delay(500);
flag = 0;
}

//robot is picking up pieces  when flag is zero
else if(flag == 0){
 //stop and turn 18- degrees
      runMotors(0,0);
  turn180();
  runMotors(0,0);

//follow line until straight
   while(lvalue>600 ){
   lineFollow();
  }

//pickup pieces
   runMotors(0,0);
      raise();
   delay(1000);
   runMotors(-16,-16);
   delay(500);
   flag = 1;
}
}
//linefollow until pickup or droppoff
else{
lineFollow();
}
}
//********************* Functions
(subroutines)*****************


//Toggle an LED on/off
void toggleLED(int colour){
 digitalWrite(colour, HIGH);
 delay(250);
 digitalWrite(colour, LOW);
 delay(250);
}

// Turn on a single LED and turn others off
void turnOnLED(int COLOUR)
{
 digitalWrite(GRN, LOW);
 digitalWrite(YLW, LOW);
 digitalWrite(RED, LOW);
 digitalWrite(COLOUR, HIGH);
}

void runMotors(int deltaL, int deltaR)
{
 int pulseL = (stopPulse + deltaL)*10;    //length of pulse in
microseconds
 int pulseR = (stopPulse + deltaR)*10;
 leftWheel.writeMicroseconds(pulseL);
 rightWheel.writeMicroseconds(pulseR);
}

void turn180(){

delay(1000);
    runMotors(7,-13);
    delay(500);

 while(rvalue<600) {
       //read the sensor value
    lvalue = analogRead(LSENSOR);
    rvalue = analogRead(RSENSOR);

      //map the values into millivolts (assuming 3000 mV
reference voltage)
    lvalue = map(lvalue,0,1023,0,3000);
    rvalue = map(rvalue,0,1023,0,3000);

    runMotors(7,-13);
 }
runMotors(0,0);
      runMotors(7,-13);
      delay(200);

 while(rvalue>600){
     //read the sensor value
    lvalue = analogRead(LSENSOR);
    rvalue = analogRead(RSENSOR);

      //map the values into millivolts (assuming 3000 mV
reference voltage)
    lvalue = map(lvalue,0,1023,0,3000);
    rvalue = map(rvalue,0,1023,0,3000);
runMotors(7,-13);
  }

  runMotors(7,-13);
  delay(300);
     runMotors(0,0);
}

void turn90(){
 runMotors(0,0);
```

```
 delay(2000);
 digitalWrite(GRN, LOW);
    digitalWrite(RED, LOW);
    digitalWrite(YLW, LOW);
    runMotors(-10,-10);
    delay(150);
     runMotors(10,-10);
     delay(1000);
     runMotors(0,0);

}

void raise(){
 delay (2000);          // A couple seconds to stand back
 for (posA = myAngleA1; posA >= myAngleA2; posA--) { //
Lift action
   myServoA.write(posA);
   delay(20);
 }
}
void lower(){
 delay(1000);
 for (posA = myAngleA2; posA <= myAngleA1; posA++) {  //
Drop action
   myServoA.write(posA);
   delay(20);
 }
}

void lineFollow(){
   if(lvalue<600){
       digitalWrite(YLW, HIGH);

       runMotors(10, 0);
    }else if(lvalue>600){
     digitalWrite(YLW, LOW);
    };


     if(rvalue<600){

       digitalWrite(RED, HIGH);
        runMotors(0, 10);
    }else if(rvalue>600){
     digitalWrite(RED, LOW);
    }else;

    if(lvalue<600 && rvalue<600){

    digitalWrite(GRN, HIGH);
    digitalWrite(RED, LOW);
    digitalWrite(YLW, LOW);
         runMotors(10, 10);
    }else{
     digitalWrite(GRN, LOW);
    }
}

void dumpbucket(){
 delay (2000);          // A couple seconds to stand back
 for (posB = myAngleB1; posB <= myAngleB2; posB++) { //
Lift action
   myServoB.write(posB);
   delay(40);
 }
}

void returnbucketfromDown(){
 delay (2000);          // A couple seconds to stand back
 for (posB = myAngleB2; posB >= myAngleB1; posB--) { // Lift
action
   myServoB.write(posB);
   delay(20);
 }
}



void liftbucket(){
 delay (2000);          // A couple seconds to stand back
 for (posB = myAngleB1; posB >= 60; posB--) { // Lift action
   myServoB.write(posB);
   delay(50);
 }
}

void returnbucketfromUp(){
delay (2000);          // A couple seconds to stand back
 for (posB = 60; posB <= myAngleB1; posB++) { // Lift action
   myServoB.write(posB);
   delay(40);
 }

}

void pickupMotions(){

 runMotors(0,0);
 delay(500);
 runMotors(-15,-15);
 delay(1200);
 runMotors(0,0);
 turn180();
 delay(500);
 lower();
  runMotors(-16,-16);
  delay(1000);
   runMotors(0,0);
//Serial.println("lifting");
liftbucket();
Serial.println("");


raise();

//Serial.println("dump");
//dumpbucket();
//Serial.println("raise");
//returnbucketfromDown();
//Serial.println("Lower");
//lower();
}


void dropOffMotions(){

  runMotors(0,0);
Serial.println("Ran 2");
raise();

 delay(500);
```

```
  runMotors(-15,-15);
  delay(1200);
  runMotors(0,0);
  turn180();
  delay(500);
   runMotors(-16,-16);
   delay(1000);
    runMotors(0,0);

Serial.println("dump");
dumpbucket();
returnbucketfromDown();

}
```