Jack Dibachi
ME 179L
Fall 2019

Final Report

Introduction

Competition rules

The competition board is peppered with various “cheeses” made of foam. Once the start light is engaged, the two robots begin collecting cheeses for the 90 second span of each round. The robots may not be touched until the end, when the robots are lifted straight up and the collected cheeses are counted. Depending on their initial position on the board, and their final resting place with respect to the robot, cheeses are assigned point values as follows:

	Home	Neutral/Random	Opponent
Ground Cheese	2	3	4
Sky Cheese	4	5	6

If these cheeses are deposited on the wall, their point values are quadrupled, such that one opponent sky cheese is worth 24 points rather than 6 points. Additionally, there is a golden cheese suspended in the middle row of sky cheeses worth 80 points when deposited on the wall, but only if it is the only sky cheese placed on the wall.

Design constraints

In addition to the parameters of the competition, the following constraints also inform the design of the robot:

- The robot must have a kill-switch in the event of a catastrophic collision
- Detachable baskets are permitted
- All structural components must be made of Legos
- Zip ties are only permitted for interfacing sensors and motors with the structure
- The robot must fit in a one foot cube before the competition begins

Available components

Arduino Board

The board provided to us is an Arduino Uno with a motor controller shield attached, and an additional breadboard shield attached above it. The whole stack is rigidly fixed to a Lego plate for interfacing with the rest of the robot structure. The Arduino is powered by a 9 Volt battery connected to a barrel jack adapter, while the motor driver board is powered by a 7.5 Volt battery pack with an integrated power switch.

Sensors

To navigate its surroundings, the robot can integrate many types of sensors.

Microswitches

Microswitches rely on a small lever to close a circuit when depressed.

Beam Sensors

Beam sensors are U-shaped sensors which have an infrared beam shining between the two prongs. When the beam is interrupted, the sensor reflects as such and breaks the circuit.

IR Range Sensors

The IR range sensors rely on the reflected angle of an infrared beam from a wall to determine the wall distance away from the sensor. This is an analog sensor, so the signal varies continuously as a function of distance from the wall.

IR Reflectivity Sensor

Reflectivity sensors shine infrared radiation at a close-range target, and the analog signal varies based on the reflectivity of the target. If the target changes from black to white, there is a dramatic change in the voltage output from the sensor.

Flex Sensor

The flex sensor works on the same principle as strain gauges, which change electrical resistance when undergoing deflection. This sensor can be read with an analog input.

Photoresistor

The photoresistor is a resistor which decreases in resistance when exposed to light. This can be read through an analog pin on the Arduino.

Motors

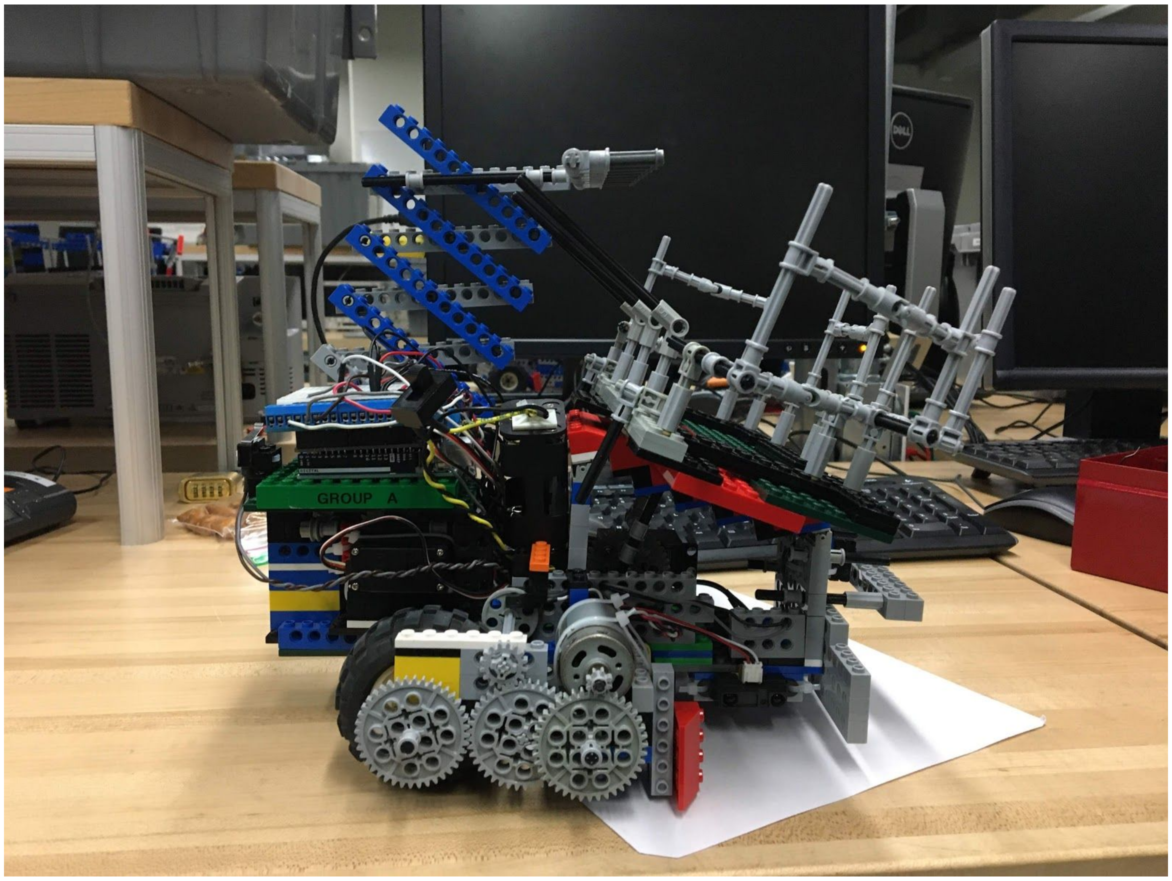
Brushed DC motors take a PWM signal from the Arduino motor driver board to modulate the speed. In conjunction with Lego gears, the output driving speed and torque can be modified. Servo motors have a range of half a rotation, and can be set to precise locations, whereas the DC motors cannot be set to certain locations without assisting hardware and sensors.

Lego hardware

There were a variety of Lego pieces available for use, including, but not limited to:

- Gears
- Solid bricks
- Technic bricks
- Axles and axle fittings
- Wheels

Design Description



The robot utilizes a rear-wheel drive assembly with a front sled to navigate the board. The arm meant to knock down sky cheese is actuated with a servo motor and linkage system. The basket to receive sky cheese is held in place by a second servo on a ramp, which will allow the basket to slide onto the wall when engaged. The robot raises the arm before making its round about the home side of the board, collecting the three sky cheeses and deploying the basket onto the wall.

Pros and Cons

Pros	Cons
Lightweight	Cannot collect ground cheese
High point value	Reliant on cheese depositing in basket
Avoid other robots	

Mechanisms

Arm Linkage

The arm linkage consists of a servo rotating an arm attached to a scissor mechanism, such that the linkage will fully extend upward when in one servo position, and retract in another servo position. The arm will extend upward before moving around the competition board, then retract at the end of its run.

Basket Deploy

To deploy the collection basket onto the wall, a servo connected to a small arm will hold the basket in place until the wall is detected at the end of the cheese collection run. At this point, the servo will rotate to its descended position, swinging the arm down and allowing the basket to slide down the ramp and onto the wall.

Gear ratio

Each wheel is attached to a DC motor with a 1:25 gear ratio, such that the wheel turns one revolution for every 25 revolutions of the DC motor. This gear ratio was chosen to move the robot quickly around the board, allowing us to collect all of our sky cheese and deposit before encountering the opposing robot.

Sensor Integration

A photoresistor is placed on the bottom of the robot configured in an analog input pullup configuration, such that turning on the start light will cause the read pin to read below a certain value, starting the robot. A beam break sensor is configured with a Lego wheel with six holes to form an encoder, which is placed on the drivetrain to track the linear movements of the robot. The encoder is plugged into a digital interrupt pin on the Arduino, and a function in the software counts the number of interruptions in the digital signal. An IR Range sensor maintains the parallel distance from the wall by plugging into an analog read pin, and adjusting each motor PWM based on the variation from the target signal, and thus the target distance. One microswitch is used as a front wall sensor, such that it deploys the basket when the circuit is closed, altering the state of the digital read pin. The same configuration is used for the kill switch, which is used to stop motor function and idle the robot.

Dynamics and stability

The wall follower uses an IR Range sensor in conjunction with a PI controller within the Arduino control software. The preferred distance away from the wall is hard-coded based on calibrated data for the range sensor. When the reading from the sensor is not at the desired value, the PI controller changes the speeds of the motors, causing the robot to rotate closer or further from the wall until at the desired distance from the wall. This iterative process helps keep the robot moving straight as it follows the wall.

Software description

Turning the robot on runs through the Arduino setup portion of the script, which attaches the relevant digital and analog pins, servos, and other sensors. After setup completion, the script enters the loop, at which point it awaits the appropriate analog reading from the photoresistor. When the competition light is turned on, the analog signal from the photoresistor drops dramatically, exiting the waiting phase and turning the servo to raise the sky cheese arm. After a two second delay allowing the arm to rise, the robot enters state 1 and the motors drive forward for a specified distance, counted by the integrated encoder. After this distance goal is reached, the robot turns 90 degrees and enters state 2, employing wall following to correct for any undesired movement, then turns again to begin the cheese collection run. The cheese collection run, also known as state 3, uses encoders to travel for a predictable distance in a straight line. When the distance goal for state 3 is reached, the robot turns to drive back to the wall and enters state 4. Lastly, the robot uses the front microswitch to detect the final wall, entering state 5 and turning the servo to deposit the basket onto the wall. The robot idles for the rest of the competition to prevent sabotage of the cheese on the wall.

Competition performance

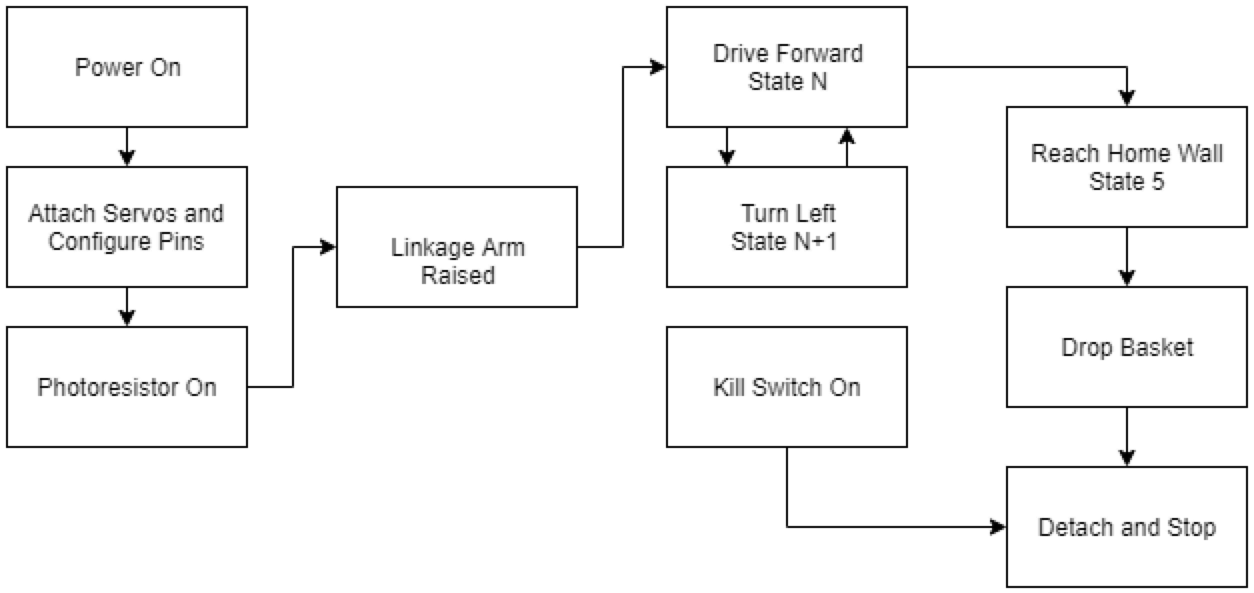
Our highest score in any of the competition rounds was 12 points. In the first round, the robot had collected two of the sky cheeses and landed them in the basket, but a ground cheese was trapped between the wall and the front microswitch, preventing the earning of at least 32 points. During the competition, the robot behaved sporadically. The speed of the robot during the cheese collection run prevented repeatable deposition of sky cheese into the basket, and slowing down caused cheese to fall short of the basket. As such, we made it through the first three rounds without losing a round, but lost the third round and again lost the fourth round. We beat approximately half the competing teams, but lost as we encountered robots with more reliable performance than ours.

Reflections

If time permitted, I would have designed a robot capable of sweeping the neutral cheese on the middle ramp into a basket capable of deposition onto the wall. In this case, eight neutral cheeses at three points per cheese amounts to 32 points, and on the wall amounts to 128 points. Given these movements are fairly simple, they can be dialed in to perfection, winning the competition with no contest. My advice to teams in the future is to pick a strategy that is foolproof, and to employ wall depositing for whichever design they choose. While I would not change the way we approach our design process, I regret the shortage of time I had to complete the project due to unforeseeable obstacles, both inside and outside the scope of the class.

Appendix

Flowchart



Annotated script

```
//Include the needed libraries
#include <Servo.h>
#include <Adafruit_MotorShield.h>
#include <SoftwareSerial.h>
const int onswitch = A0; //photoresistor pin
const int sensor = A1; //wall follow sensor
const int kill = 5; //kill switch pin
const int wall = 4; //front wall switch pin
const int LCDTxPin = 13; // LCD connected to this pin (pin 13)
const int DummyRxPin = 13; // Not used by LCD Board, can be any unused pin
const int EncoderPin = 2; // break-beam sensor connected to this pin (pin 2, must be pin 2 or 3
since they have hardware interrupts 0 and 1, respectively)
const int spf = 1.5; //speed factor applied to motors to control the speed
const int SPEEDL = 163; //PWM for left motor
const int SPEEDR = 200; //PWM for right motor (speeds are different as a result of unresolved
motor faults
int turntime = 500*spf; //time assigned to 90 degree turn
const int raised = 62; //set raised position for linkage arm
const int lowered = 172; //set lowered position for linkage arm
const int bup = 175; //set raised position for basket holder
const int bdown = 130; //set lowered position for basket holder
int sensorvalue = 0; //parameter for wall following controller
int kp = 6; //proportional coefficient for wall follower controller

Servo servoA;
Servo servoB;
SoftwareSerial mySerial =  SoftwareSerial(DummyRxPin, LCDTxPin);  //Change Tx and Rx Pins
to pins of our choosing
Adafruit_MotorShield AFMS = Adafruit_MotorShield(); // create Adafruit_MotorShield object
Adafruit_DCMotor *Left_Motor = AFMS.getMotor(1); // Places the left motor on port 1
Adafruit_DCMotor *Right_Motor = AFMS.getMotor(2); // Places the right motor on port 2

volatile int ticks = 0; // Count of encoder ticks, volatile is used for speed since it will be in
interrupt subroutine
int oldticks = 0; // Last count of ticks
int tickGoal = 100;
int tickGoalA = 28; // Number of ticks desired for first leg
int tickGoalB = 40; // Number of ticks desired for second leg
int tickGoalC = 55; // Number of ticks desired for third leg
int tickGoalD = 55; // Number of ticks desired for final leg
int waiting = 1; // Waiting state; 1 if waiting, 0 if running
int state = 1; //first of five states influencing robot procedure

// countTick runs whenver interrupt is triggered, increments ticks each time it is called
void countTick() {
  ++ticks;
}

void setup(){
  servoA.attach(10); //servo for linkage
  servoB.attach(9); //servo for basket
  pinMode(onswitch, INPUT_PULLUP); //initializes the photoresistor
  pinMode(kill, INPUT); //initializes the kill switch
  digitalWrite(kill, HIGH);
  pinMode(wall, INPUT); //initializes the front wall switch

  pinMode(EncoderPin, INPUT); // Makes Encoder Pin an input
  digitalWrite(EncoderPin, HIGH); // Enables the pull-up resistor on Encoder Pin (Pin 2)
  attachInterrupt(digitalPinToInterrupt(EncoderPin), countTick, FALLING); // Set routine to run
(countTick) and interrupt trigger: FALLING (when pin 2 voltage goes from high to low, countTick
will run)
  AFMS.begin();     // setup the motor controller connection
  servoA.write(lowered); //starts linkage in lowered position to fit in box
  delay(1000);
  servoB.write(bup); //starts basket in raised position
  delay(1000);
  servoB.detach(); //powers off servo until basket must be lowered
}

//drives both motors forward
void DriveForward(){
  Right_Motor->run(FORWARD);
  Left_Motor->run(FORWARD);
}

//drives both motors backward
void DriveBackward(){
  Right_Motor->run(BACKWARD);
  Left_Motor->run(BACKWARD);
}

//turns robot left
void TurnLeft(){
  Right_Motor->run(FORWARD);
  Left_Motor->run(BACKWARD);
}

//stops both motors
void Stop(){
  Right_Motor->run(RELEASE);
  Left_Motor->run(RELEASE);
}
```

```
void loop(){

  //awaiting light to be pressed
  if (waiting){ // If in waiting state
    mySerial.print("?x00?y0"); // move cursor to beginning of screen
    mySerial.print("Waiting to press");

  //detects light and begins operation
  while(analogRead(onswitch)<80)  {
    Right_Motor->setSpeed(SPEEDR/spf); //assigns speed to each motor, adjusting by the speed
factor
    Left_Motor->setSpeed(SPEEDL/spf);
    mySerial.print("?f");
    mySerial.print("?x00?y0");
    mySerial.print("Got it!");
    waiting = 0; // stop waiting
    ticks = 0; // reset ticks and old ticks
    oldticks = -1;

    mySerial.print("?f"); // clear the screen and display counts
    mySerial.print("Counts:");
    servoA.write(raised); //raise linkage arm before starting forward
    delay(2000);
    DriveForward();
  }
  }

  else { // if not in waiting state
    sensorvalue = analogRead(sensor);
    if (state == 1){
      Right_Motor->setSpeed(SPEEDR/spf);
      Left_Motor->setSpeed(SPEEDL/spf);
      tickGoal = tickGoalA; //set tick goal for first leg
    }
    if (state == 2){
      Right_Motor->setSpeed(constrain(SPEEDR-kp*(230-sensorvalue),120,230)/spf); //wall
following scheme for adjusting motor speed
      Left_Motor->setSpeed(constrain(SPEEDL+kp*(230-sensorvalue),120,230)/spf);
      tickGoal = tickGoalB; //tick goal for second leg
      turntime = 600*spf; //adjusted turntime for second leg
    }
    if (state == 3){
      Right_Motor->setSpeed(SPEEDR/spf);
      Left_Motor->setSpeed(SPEEDL/spf);
      tickGoal = tickGoalC; //tick goal for cheese collecting run
    }
    if (state == 4){
      Right_Motor->setSpeed(constrain(SPEEDR-kp*(230-sensorvalue),120,230));
      Left_Motor->setSpeed(constrain(SPEEDL+kp*(230-sensorvalue),120,230));
      tickGoal = tickGoalD;
      turntime = 500*spf;
    }
    if (ticks != oldticks){
      // reprint count if it has changed
      mySerial.print("?x00?y1");
      mySerial.print(ticks);
      oldticks = ticks; //keep going towards tick goal
    }
    if (!digitalRead(kill)){
    Stop(); //stop motors if kill switch is depressed
    delay(90000); //cease programming function until reset button can be pressed
    waiting = 1;
    }
    if (!digitalRead(wall) && state >= 4){
      Stop(); //procedure for hitting the home wall and depositing the basket
        servoB.attach(9); //reattach basket servo
        delay(500);
        servoB.write(bdown); //lower basket
        delay(1000);
        DriveBackward(); //back away from wall to enable vertical pickup
        delay(300);
        Stop();
        servoB.detach(); //power off basket servo
        delay(500);
        servoA.detach(); //power off linkage servo
        delay(90000);
    }
    if (ticks >= tickGoal){ // If tick goal has been reached
      if (state == 3){
        DriveBackward(); //brief backup to prevent cheese blocking the path back to the final home
wall
        delay(300);
      }
      state = state + 1; //turn left and continue to next leg
      TurnLeft();
      delay(turntime);
```

```
      DriveForward();
      ticks = 0;
      oldticks = -1;
      turntime = 600*spf;
    }
  }
}
```