Sensors and Motor Lab

Cole Gulino

Team C / Column Robotics

Teammates: Job Bedford, Erik Sjoberg, Rohan Thakker

ILR # 1

October 16, 2015

Individual Progress

For the Sensors and Motor Lab, I implemented the debounced interrupt button, the potentiometer, the servo motor, and communication with the GUI from the Arduino.

The button (shown in Figure 1 with circuit diagram) is used as a physical way to change the states of the motor control logic. I used the button to implement four different states: potentiometer controlling the servo motor, the IR sensor used to control DC motor velocity control, the IR sensor used to control DC motor position control, and the flex sensor controlling the direction of the stepper motor.

Figure 1) Electronic Button used in Lab [1]

The first thing that I did was to set up the circuit for the button. I used a 10k ohm resistor to pull down the voltage to ground where I measured if the button was pressed (circuit shown in Figure 2).

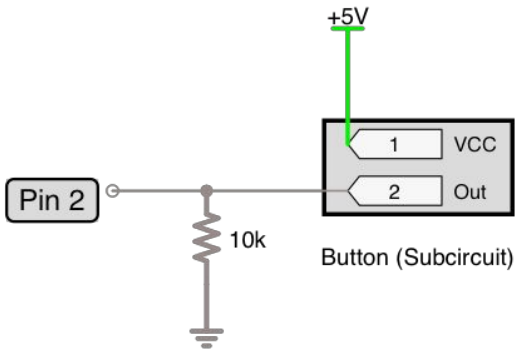

Figure 2) Button Subcircuit

Then I set up the button interrupt system. I set the button up on pin 2, which corresponds to interrupt 0 in the arduino. Figure 3 shows the code that I used to setup the button interrupt system.

```
#define stateButton 2 // define the interrupt pin the button is on
void setup(){ // small part of the setup dealing with the button
  pinMode(stateButton, INPUT); // set the button up as an input
  digitalWrite(stateButton, HIGH); // set the internal pull up resistor in the arduino high
  attachInterrupt(0, stateChange, RISING); // interrupt zero (pin 2) on rising edge
  …
  …
}
```

Figure 3) Button Interrupt Setup

The next thing that I did was to write the stateChange function that would be run whenever the button was pressed. This is also where I debounced the button. I debounced the button in the main loop and in the interrupt function. The code is shown in Figure 4.

```
volatile int state; // the integer that names the state, volatile because button changes it
long prevTime; // A long which keeps track of previous time for debouncing
long debounceDelay = 200; // Debounce delay

void loop(){ // Main loop
    if( digitalRead(stateButton) == LOW && stillPressed &&
        (millis() - prevTime) >= debounceDelay ){
    // If the button has been pressed for longer than the debounce delay and if the button
    // is read to be low, assume button is no longer being pushed
    stillPressed = 0;
  }
  …
  ...
}
void stateChange(){
  if( ( millis() - prevTime ) >= debounceDelay && !stillPressed ){
    // If the debounce delay has been reached and the button is no longer pressed
    state = ( state + 1 ) % 4; // Switch between the four states
  }
  guiCntrl = 0; // Give control back to the sensors (more later on this)
  prevTime = millis(); // Update pref time
}
```

Figure 4) Button Debounce and State Change Code

If the button is pushed, it will be read HIGH in the main loop. Whenever the button is seen to be LOW, you update a boolean that lets the state change interrupt function know that the button is no longer pressed. It will not change the state unless the button is not being pressed. This fixes a problem where you could hold the button longer than the debounce delay and the button would change the state twice. The guiCntrl variable is a boolean that will determine whether the GUI or the sensors are controlling the motors. This will be discussed later

Figure 5 shows the code in the main loop where I set up the state machine.

```
void loop(){
    switch( state ){
        case 0: // Potentiometer servo control
          ...
          break;
        case 1:   // DC Motor Velocity Control
          ...
          break;
        case 2:   // DC Motor Position Control
          ...
          break;
        case 3:{ //Stepper control
          ...
          break;
    }
    ....
}
```

Figure 5) State machine set up

The next part of the project that I completed was to implement servo motor control with a 10k ohm potentiometer (images shown in Figure 6 and 7).

Figure 6) CA9 Trimmer Potentiometer [2]

Figure 7) Simple Servo Motor [3]

The circuit for the servo motor and potentiometer circuit are shown in Figure 8.

In order to filter out noise from the power supply, I used 100mF capacitors in order to filter out some noise from the power supply to the servo motor. Figure 8 shows the circuit diagram for the servo and potentiometer.

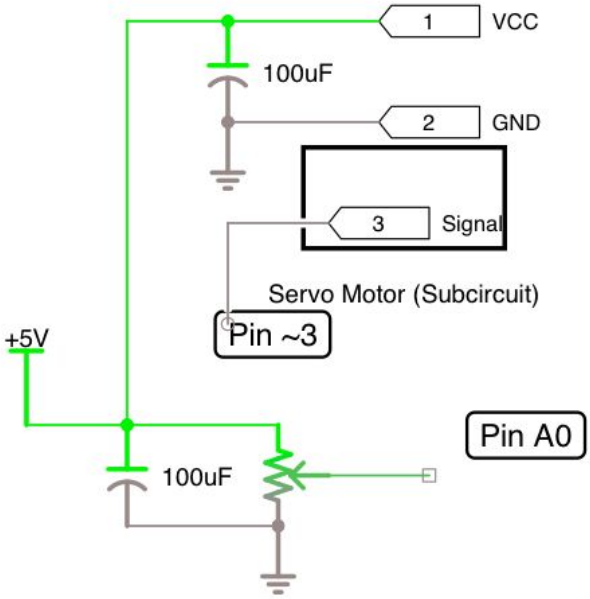

Figure 8) Servo and Potentiometer Circuit Diagram

I used the Arduino servo motor library in order to control the servo motor. The code I used is shown below in Figure 9.

```
#include <Servo.h> // Include servo library
#define potPin A0 // Define the poteniometer input pin
#define servoPin 3 // Define the servo pin

int prevPotVal; // Define a previous potentiometer variable to do software filtering
int servoVal; // Value read from potentiometer
int servoValMap; // Define a value from potentiometer mapped to servo value variable
Servo serv; // Set up a servo variable

void setup(){
    serv.attach(servoPin); // attach servo motor to servo pin
    ...
}

void loop(){
    switch( state ) {
        case 0: // Potentiometer and servo mode
            servoVal = analogRead(potPin); // Read potentiometer value
            if( abs( prevPotVal - servoVal ) > 15 && !guiCntrl ){ // Some software filtering
                // If the difference between the new potentiometer value is > 15 from last
                // And the GUI does not have control
                servoValMap = map(servoVal, 0, 1023, 0, 180); // Map value to 180 for servo
                serv.write(servoValMap); // Write the mapped value to servo
            }
            if( guiCntrl ){ // If GUI has control
                serv.write(servoValMap); // use the value given by the GUI
            }
            delay(10); // Some small delay to reduce noise
            prevPotVal = servoVal; // save the old value of pot reading for later comparison
            break;
        ...
    }
    ...
    ...
}
```

Figure 9) Servo and Potentiometer Code

The first thing that the code does is read an analog value from the potentiometer. I first just connected the potentiometer to the arduino and printed out the analog values in order to ensure that it was given me values from 0 to 1023. Then I attached the servo to a PWM pin on the arduino. I then used the Servo.h library to attach the servo motor to the pin I connected the servo signal wire to. So once I put them together, I used the map() function to map the analog value from (0 to 1023) to (0 to 180), which are the values that the Servo library can read to choose the angle it moves to.

That was my initial start. Then I added a few other features. There is a check in the first if statement, which checks to see if the new potentiometer value is different enough from the previous value to warrant a change. It also checks to see if the GUI has control by checking to see if guiCntrl variable is 0. If it does this, it knows it should change its value based on the reading from the potentiometer. If guiCntrl is 1, it will write the servo from a value that the GUI specifies. It then saves the previous potentiometer value so the filter in the first if statement will have something to compare it to.

The next thing that I did for the project was to handle the communication from the Arduino to the GUI created by Job in Processing. We needed to pass along real time information about the sensors and motors information. The communication to the GUI is shown in Figure 10.

```
void sendData(){ // Send bytes of data to the GUI through the serial port
  Serial.write("A"); // Start character
  delay(10);
  Serial.write(state); // State the motors are in
  Serial.write(abs(encoderPos%360)); // Encoder Angle
  Serial.write(dcDirect); // Direction of DC motor
  Serial.write(map(measureOpticalSensorVoltage(), 0, 550, 0, 255)); // Get Updated IR sensor
voltage
  Serial.write(abs(stepperCurrentAngle)); // Angle of stepper motor
  Serial.write(map(analogRead(flexPin), 0, 1023, 0, 255)); // Mapped value from flex reading
  Serial.write(map(analogRead(potPin), 0, 1023, 0, 180)); // Mapped value from pot reading
for servo angle
  Serial.write(map(analogRead(potPin), 0, 1023, 0, 255)); // Mapped value from pot reading
}
```

Figure 10) Communication from Arduino to GUI in Processing

The first character to be written along the serial port from the Arduino to Processing is the letter ‘A’, which is used to denote the start of a new batch of data. We then sent over the state in order to let the GUI know which state the state machine is currently in. Then we send the data. We needed to map all of the data to values from 0-255 in order to send them one byte at a time. We ran this function at the end of every loop(). In this function, we updatensor values in order to get accurate real time readings even if we are not in the state that required to gather it.

Once we had communication to the GUI working, I wrote the code to send information from the GUI to the Arduino. Figure 11 shows the code for communication to the GUI.

```
void loop(){
  if(Serial.available() > 0){ // If we are receiving data from serial
    state = Serial.parseInt(); // Get the state
    delay(5); // Delay
    guiCntrl = 1; // If we get serial information, set control to GUI
    if( state == 0 ){ // If state is potentiometer and servo, get data from GUI
      servoValMap = Serial.parseInt();
    }
    if( state == 2 ){ // If state is DC motor get angle
      dcMotorAngle = Serial.parseInt();
    }
    if( state == 3 ){ // If state is stepper motor get angle
      stepperRelativeRefAngle = Serial.parseInt();
    }
  }
  ...
  ...
}
```

Figure 11) Communication from GUI to Arduino

This code runs at the beginning of every loop(). It sets the guiCntrl bit which lets the state machine know to take these values instead of reading the sensor data values. It also gathers the sensor data values from the GUI.

Challenges

The first challenge that I faced was with the servo motor and potentiometer. In order to reduce noise from the signal, I needed to add 100uF capacitors in order to smooth out the signal of the potentiometer and servo motor. I was still seeing some twitching, so I added the if statement shown in Figure 9 in order to only move the servo if the potentiometer had been moved enough to warrant a change. The readings were flipping one or two values from the potentiometer, so the if statement stops the twitching from receiving the small variations in the signal.

The next challenge that we faced as a team was electrical hardware difficulties. We had trouble with the DC motor driver and the stepper motor driver. We were receiving bad signals from the DC motor driver at first, but we added pull up resistors in order to fix this. We also needed to isolate the power supply from the motors and the arduino.The stepper motor driver was more difficult. We needed to debug the circuit many times in order to find the problem. During this debug, we realized that the stepper motor driver was faulty. We had to replace three different stepper motor drivers before we found one that was working correctly.

The last major challenge that we faced was with the communication between the Arduino and the GUI. The first problem was with a large lag from the arduino to the GUI. The first thing that we did was to change the timeout of the serial port. The next thing that we did was to reduce the delay on parts of the code. This made some of the motors slightly less smooth, but it greatly increased the speed of communication between the Arduino and the GUI. The next challenge that we faced was the differing communication methods between Processing and Arduino. Processing was sending us bytes of data, but we were expecting ASCII characters. So whenever we were using the parseInt() Arduino function, it was returning NULL which translates to a zero, so every value that we were receiving was 0, which was constantly changing the state to zero no matter what state we meant it to be in. To fix this, we cast the integer values to a string in order for parseInt() to work.

Teamwork

We split up the work as evenly as we could. I took the servo motor, potentiometer, button, and Arduino side of GUI-Arduino communication. Rohan took the stepper motor and flex sensor. Erik took the DC motor and the IR sensor. And Job took the GUI and GUI side of GUI-Arduino communication.

Even with the breakdown of the work, some tasks took longer than others like the PID controls and debugging the stepper motor. In order to remedy this, we worked together in debugging each other's circuits and software. We also worked together in the end for the integration step.

A major challenge was integrating all of our code together. This was accomplished by working together during the coding integration process. We want to continue to improve on this in later parts of the project.

Figure 12 shows the diagram of the total project.

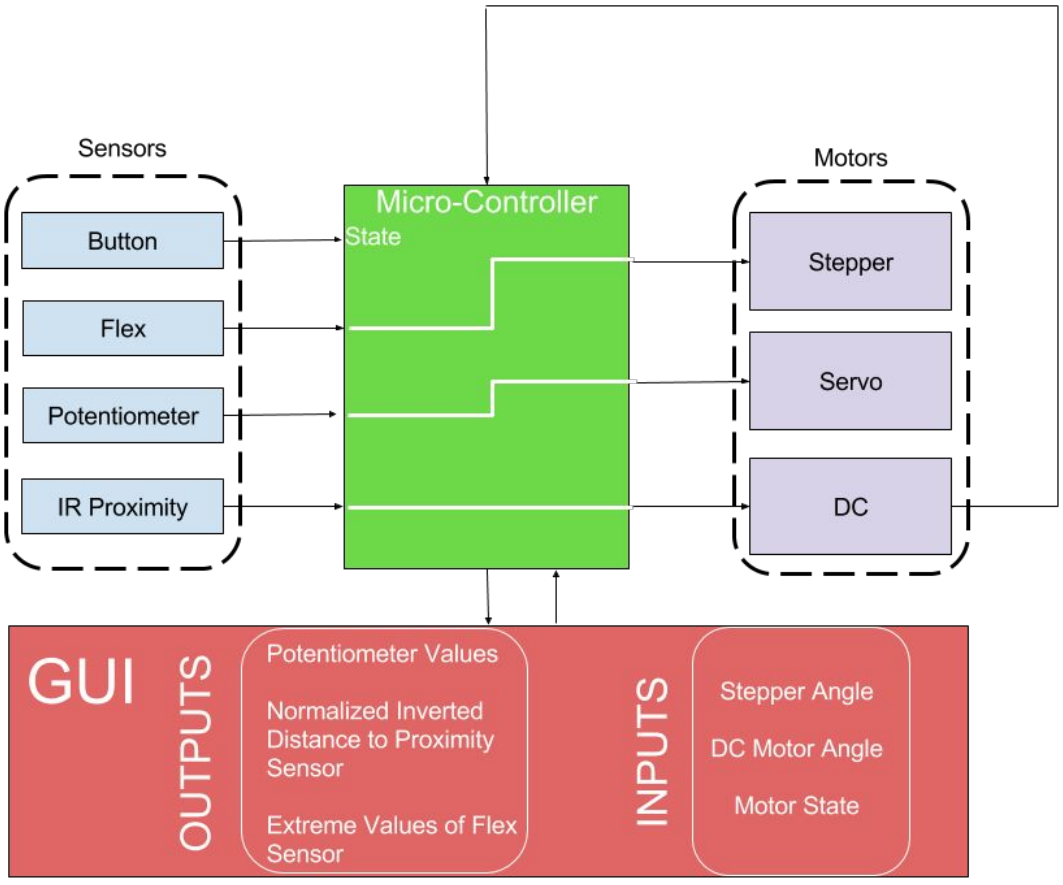

Figure 12) Flow Chart of Motor Sensor Lab

Future Plans

During the week before the next performance review, we will be working on various tasks in order to push our project further focussing on risk management and software systems.

Erik will be working on displaying the 2D x,y map display in ROS and researching the localization ROS package. This is for the part of the December functionality that will be used to demonstrate that we will be able to localize ourselves in our search area.

Rohan will be working on the node that is receiving data from the AR.Drone 2.0. This data will be used in our system for our low-level control system

Job will be working on designing our mover node to issue commands for the low level control.

I will be also doing some research on the localization ROS package, designing the risk management strategies and revising our work breakdown structure tasks.

Full Code

Arduino Code

```
#include <Servo.h>
#include <pins_arduino.h>
#include <PID_v1.h>
Servo serv;

// Define pin numbers
#define potPin A0
#define servoPin 3
#define stateButton 2
#define dcMotorEnablePin 6
#define dcMotorPinA 11
#define dcMotorPinB 8
#define encoderPinA 4
#define encoderPinB 5
#define opticalPin A1
#define stepPin 13
#define dirPin 12
#define flexPin A2
#define stepperEnable 10
#define stepAngle 1.8
#define flexThreshold 760

// Variables
long servoVal;
volatile int state;
long prevTime;
long debounceDelay;
long opticalSensorVoltage;
long opticalSensorVoltageSmooth;
int stepCount;
int currState;
int prevState;
bool stillPressed = 0;
int prevPotVal;
int servoValMap;
int dcDirect;
int counter;
int guiCntrl;
int dcMotorAngle;
int stepperRelativeRefAngle;
int stepperCurrentAngle;
bool stepperDir;

//DC encoder
volatile int encoderPos = 0;
int lastEncoderPinA = LOW;
int lastEncoderPinB = LOW;
int lastEncoderPos = 0;
double lastSpeedTime = 0;
```

```
int epA = LOW;
int epB = LOW;
double dcSpeedMeasured; //Degrees per second
double dcSpeedSmoothed;
double dcTargetSpeed; //Degrees per second

//PID: Reference
http://brettbeauregard.com/blog/2011/04/improving-the-beginners-pid-introduction/
//Define Variables we'll be connecting to
/*working variables*/
unsigned long lastTime;
double Input, Output, Setpoint;
double errSum, lastErr;
double kp, ki, kd;
double avgSpeed = 0;
double avgOutput = 0;

// pins_arduino Reference http://playground.arduino.cc/Main/PinChangeInterrupt
void pciSetup(byte pin){
    *digitalPinToPCMSK(pin) |= bit (digitalPinToPCMSKbit(pin));  // enable pin
    PCIFR  |= bit (digitalPinToPCICRbit(pin)); // clear any outstanding interrupt
    PCICR  |= bit (digitalPinToPCICRbit(pin)); // enable interrupt for the group
}

ISR (PCINT2_vect){  // handle pin change interrupt for D0 to D7 here
   epA = digitalRead(encoderPinA);
   if (epA != lastEncoderPinA) //Only trigger if pin A has changed = 360 counts per
revolution
   {
     if (epA == digitalRead(encoderPinB)) {
        encoderPos++; //360 counts per revcolution clockwise
      } else {
        encoderPos--;
      }
      lastEncoderPinA = epA;
   }
}

void setup(){
  // Setup pins and servo
  serv.attach(servoPin);
  pinMode(stateButton, INPUT);
  digitalWrite(stateButton, HIGH);
  digitalWrite(dcMotorEnablePin, HIGH);
  attachInterrupt(0, stateChange, RISING);

  //DC motor control
  pinMode(encoderPinA, INPUT);
  pinMode(encoderPinB, INPUT);
  digitalWrite(dcMotorPinA, LOW);
  digitalWrite(dcMotorPinB, LOW);
  pciSetup(encoderPinA);
  //PID
 Input, Output, Setpoint = 0;
  lastTime = 0;
  SetTunings(2, 0, 0);
  // Initializations for Stepper Motor
  pinMode(dirPin, OUTPUT);
  pinMode(stepPin, OUTPUT);
  pinMode(stepperEnable, OUTPUT);
```

```
  digitalWrite(stepperEnable, HIGH);
   stepperCurrentAngle=0;
  // Initialize Flex Sensor
  pinMode(flexPin, INPUT);
  stepperDir = 1;

  // Initialize variables
  servoVal = 0;
  counter = 0;
  guiCntrl;
  state = 1;
  prevTime = 0;
  debounceDelay = 200;
  opticalSensorVoltage = 0;
  prevState = LOW;
  Serial.begin( 9600 );
  Serial.setTimeout(5);
}

void loop(){
  if( digitalRead(stateButton) == LOW && stillPressed &&
      (millis() - prevTime) >= debounceDelay ) {
    stillPressed = 0;
  }
  if(Serial.available() > 0){
    state = Serial.parseInt();
    delay(5);
    guiCntrl = 1;
    if( state == 0 ){
      servoValMap = Serial.parseInt();
    }
    if( state == 2 ){
      dcMotorAngle = Serial.parseInt();
    }
    if( state == 3 ){
      stepperRelativeRefAngle = Serial.parseInt();
    }

  }
  switch( state ){ // Potentiometer servo control
    case 0:{
      servoVal = analogRead(potPin);
      if( abs( prevPotVal - servoVal ) > 15 && !guiCntrl ){
        servoValMap = map(servoVal, 0, 1023, 0, 180);
        serv.write(servoValMap);
      }
      if( guiCntrl ){
        serv.write(servoValMap);
      }
      delay(10);
      prevPotVal = servoVal;
      break;
    }
    case 1: {  // DC Motor Velocity Control
      // Sensing range ~ 50 (far) to 550 (close)
      opticalSensorVoltage = measureOpticalSensorVoltage();
      if(opticalSensorVoltage > 50){
        dcTargetSpeed = 2 * (300 - opticalSensorVoltage);
        dcSpeedMeasured = measureDCSpeed();
        dcDirect = (dcSpeedMeasured < 0)? 0 : 1;
```

```
        //Update PID values
        Input = dcSpeedMeasured;
        Setpoint = dcTargetSpeed;
        Compute(); //Modifies Output variable
        moveDCMotor();
      }
      else{
        digitalWrite(dcMotorPinA, LOW);
        digitalWrite(dcMotorPinB, LOW);
        lastTime = millis();
      }
      break;
    }
    case 2: {  // DC Motor Position Control
      // Read sensor voltage
      opticalSensorVoltage = measureOpticalSensorVoltage(); // int( 50-550 )
      // Set PID input and setpoint
      // Setpoint is the desired state
      Setpoint = (guiCntrl) ? dcMotorAngle : opticalSensorVoltage; // Target state in
degrees
      Input = encoderPos; // Actual encoder degree value
      // Compute direction we need to go
      Compute(); // Modifies Output Global variable
      moveDCMotor(); // Run DC motor control
      break;
    }
    case 3:{ //Stepper control
      if(analogRead(flexPin)<flexThreshold)
        stepperDir = 0;
      else
        stepperDir = 1;
      moveStepper(stepperRelativeRefAngle, stepperDir);
      stepperRelativeRefAngle = 0;
      break;
     }
  if (state != 1 || state != 2){
      lastTime = millis(); //Prevent PID windup
      digitalWrite(dcMotorPinA, LOW); // Switch off the DC Motor
      digitalWrite(dcMotorPinB, LOW);
    }
  }
  delay(15);
  // send data every 10th count
  counter = (counter + 1)%6;
  if( counter == 5 ){
    sendData();
  }
}
void stateChange(){
  if( ( millis() - prevTime ) >= debounceDelay && !stillPressed ){
    state = ( state + 1 ) % 4;
    //Serial.println( state );
  }
  guiCntrl = 0;
  prevTime = millis();
}

void Compute(){
   /*How long since we last calculated*/
   unsigned long now = millis();
```

```
   double timeChange = (double)(now - lastTime);

   /*Compute all the working error variables*/
   double error = Setpoint - Input;
   errSum += (error * timeChange / 1000);
   double dErr = (error - lastErr) / timeChange;
   /*Compute PID Output*/
   Output = kp * error + ki * errSum + kd * dErr;

   /*Remember some variables for next time*/
   lastErr = error;
   lastTime = now;
}

void SetTunings(double Kp, double Ki, double Kd){
   kp = Kp;
   ki = Ki;
   kd = Kd;
}

double measureDCSpeed(){
   double speedNow;
   double speedNowSmooth;
   long now = millis();
   if(now - lastSpeedTime > 0){
     speedNow = 1000 *  //Units in seconds
                (encoderPos - lastEncoderPos) /  // 360 encoders per rev
                (now - lastSpeedTime);
     lastEncoderPos = encoderPos;
     lastSpeedTime = now;

     dcSpeedSmoothed = (2 * dcSpeedSmoothed + speedNow) / 3;
   }
   return dcSpeedSmoothed;
}

long measureOpticalSensorVoltage(){
  long osv;
  osv = analogRead(opticalPin);
  //Smoothing
  opticalSensorVoltageSmooth = (3 * opticalSensorVoltageSmooth + osv) / 4;

  return opticalSensorVoltageSmooth;
}

void moveDCMotor(){ //Limit output
  if (Output > 255){
    Output = 255;
  }
  if (Output < -255){
    Output = -255;
  }
  if (Output >= 0){  //counter clockwise
    digitalWrite(dcMotorPinB, HIGH);
    //analogWrite(dcMotorPinA, Output);
    digitalWrite(dcMotorPinA, LOW);
    analogWrite(dcMotorEnablePin, Output);
  }
  else{ //clockwise
    double reverse = -Output;
```

```
    //Serial.print("\t rev: ");Serial.println(reverse);
    digitalWrite(dcMotorPinB, LOW);
    //analogWrite(dcMotorPinA, reverse);
    digitalWrite(dcMotorPinA, HIGH);
    analogWrite(dcMotorEnablePin, reverse);
  }
}

void moveStepper(int angle, int dir){ // angle: in degrees; dir: boolean
  digitalWrite(stepperEnable, LOW);
  // put your main code here, to run repeatedly:
  digitalWrite(dirPin,(dir)?HIGH:LOW);
  int stepCount;
  for(stepCount=0; stepCount < angle/stepAngle; stepCount++) {
    digitalWrite(stepPin, HIGH);
    delayMicroseconds(900);
    digitalWrite(stepPin, LOW);
    delayMicroseconds(900);
  }
  stepperCurrentAngle += stepCount*stepAngle*(dir?1:-1);
  digitalWrite(stepperEnable, HIGH);
}

void sendData(){ // Send bytes of data to the GUI through the serial port
  Serial.write("A"); // Start character
  delay(10);
  Serial.write(state); // State the motors are in
  Serial.write(abs(encoderPos%360)); // Encoder Angle
  Serial.write(dcDirect); // Direction of DC motor
  Serial.write(map(measureOpticalSensorVoltage(), 0, 550, 0, 255)); // Get Updated IR sensor
voltage
  Serial.write(abs(stepperCurrentAngle)); // Angle of stepper motor
  Serial.write(map(analogRead(flexPin), 0, 1023, 0, 255)); // Mapped value from flex reading
  Serial.write(map(analogRead(potPin), 0, 1023, 0, 180)); // Mapped value from pot reading
for servo angle
  Serial.write(map(analogRead(potPin), 0, 1023, 0, 255)); // Mapped value from pot reading
}
```

GUI Processing Code

```
//Task 7 using ControlP5

//toggle, bang, slider, text label, knobs

//add: Button, Knobs, Toggles, Charts
//states of 3 motors, of 3 sensors
//control motors
//turn on, turn off

import controlP5.*;
import processing.serial.*;

Serial myPort;
int[] serialInArray = new int[9];//9
int serialCount = 0;
```

```
ControlP5 cp5;

int current_state=0;
Chart Sen1; //IRsensor
int IRdist;
Chart Sen2; //Flex Sensor
int Flex_val;
Chart Sen3; //Potentometer
int pot_val;
//encoder text
int encoder_val = 0;
Chart Mot1; //Speed
int mot_dir = 0;
//stepper value text
int stepper_val = 0;
int step_dir = 1;
Chart Mot3; //Servo
int servo_val;

Textlabel title;
Textlabel encoder1;

int color1 = color(230);
int c1, c2, c3, n, n1;

int g1, g2, g3, g4, g5;
int StpAngle = 100;
int MotAngle = 100;
int SrvAngle = 100;

void setup(){
  size(800, 400);
  noStroke();
  printArray(Serial.list());
  String portName = Serial.list()[4];
  myPort = new Serial(this, portName, 9600);
  myPort.clear();
  cp5 = new ControlP5(this);
  title = cp5.addTextlabel("ttitle")
             .setText("Team C: Column Robotics Task 7")
             .setPosition(170,350)//160
             .setColorValue(0xC0C00000)
             .setFont(createFont("Arial", 32));
  cp5.addButton("Stepper_Mot")//SendSteps")
    .setValue(0)
    .setPosition(370, 20)
    .setSize(100,49);//200 19
  cp5.addButton("SendSteps2")
    .setValue(0)
```

```
  .setPosition(370, 120)
  .setSize(100,49);//200 19

cp5.addButton("State2")
  .setValue(0)
  .setPosition(370, 69)
  .setSize(100,49);//200 19
Sen1 = cp5.addChart("Sensor1") //IRSensor
          .setPosition(40,20)
          .setSize(40,300)
          .setRange(-10, 255) //555
          .setView(Chart.BAR);
Sen2 = cp5.addChart("Sensor2") //Flex Sensor
          .setPosition(160, 20)
          .setSize(200,140)
          .setRange(160, 210)
          .setView(Chart.LINE) ;
Sen3 = cp5.addChart("Sensor3") //Potentiometer
          .setPosition(100,20)//50 500
          .setSize(40,300)
          .setRange(0, 255)
          .setView(Chart.BAR);
Sen1.addDataSet("IR");
Sen1.setData("IR", new float[1]);
Sen2.addDataSet("Flex");
Sen2.setData("Flex", new float[50]);
Sen3.addDataSet("POT");
Sen3.setData("POT", new float[1]);
Mot1 = cp5.addChart("motor1") //Servo
          .setPosition(700,20)
          .setSize(40,300)
          .setRange(-10, 255)
          .setView(Chart.BAR)
          //.setStrokeWeight(1.5)
          //.setColorCaptionLabel(color(40)) ;
Mot3 = cp5.addChart("motor3") //Speed NOT USED
          .setPosition(480, 20)
          .setSize(200,140)
          .setRange(-1, 255)
          .setView(Chart.LINE)
          //.setStrokeWeight(1.5)
          //.setColorCaptionLabel(color(40));
Mot1.addDataSet("Speed");
Mot1.setData("Speed", new float[1]);
Mot3.addDataSet("Servo");
Mot3.setData("Servo", new float[50]);
cp5.addSlider("StpAngle")
   .setPosition(160, 200)
   .setSize(80,120)
```

```
      .setRange(0,255);
  cp5.addSlider("MotAngle")
      .setPosition(260, 200)
      .setSize(80,120)
      .setRange(0,255);
  cp5.addSlider("SrvAngle")
      .setPosition(360, 200)
      .setSize(80,120)
      .setRange(0,255);
}
void draw(){
  switch(current_state){
    case 0:

      background(0);
      break;
    case 1:
      background(0);
      //background(40 , 102, 153);
      break;
    case 2:
      //background(0 , 132, 193);
      background(0);
      break;}
  //Graphs
  g1 = IRdist;//int(sin(frameCount*0.1)*20+20);
  g2 = Flex_val;//int(sin(frameCount*0.02)*50+140);
  g3 = pot_val; //int(cos(frameCount*0.01)*100+128);
  g4 = servo_val;//int(cos(frameCount*0.05)*20+20);
  g5 = encoder_val%255; // = int(sin(frameCount*0.2)*50+140);

  push_graphs(g1, g2, g3, g4, g5);
  print_values(g1, g2, g3, g4, g5);
  Encoder_update();
  Stepper_update();
  LED_update(mot_dir, step_dir);
}
void serialEvent(Serial myPort) {
  int inByte = myPort.read();
  ////////////////print(inByte);//////////////////
  if(inByte == 'A'){ //startChar cleans buffer
    serialCount = 0;}
  else{
    serialInArray[serialCount] = inByte;
    serialCount++;

    //////////VARIABLES////////////
    if(serialCount > 7){
      current_state = serialInArray[0]; //state
```

```
      encoder_val = serialInArray[1];
      mot_dir = serialInArray[2]; //direction
      ///////mot_spd = serialInArray[3]; //speed
      IRdist = serialInArray[3];
      stepper_val = serialInArray[4];
      Flex_val = serialInArray[5];
      servo_val =  serialInArray[6];
      pot_val =  serialInArray[6];
      serialCount = 0;
      myPort.clear();
    }
  }
}
public void send_cont(int states, int stp, int MotA, int Srv){
  //send control.
  println("Sending data");
  switch(states){
    case 0:
      //myPort.write(char(states));
      myPort.write(str(2));
      myPort.write(str(' '));
      myPort.write(str(stp));
      myPort.write('\n');
      print(states);
      print(' ');
      print(stp);
      print('\n');
      break;
    case 1:
      myPort.write(str(states));
      myPort.write(str(' '));
      myPort.write(str(stp));
      myPort.write('\n');
      print(states);
      print(' ');
      print(stp);
      print('\n');
      break;
    case 2:
      myPort.write(str(states));
      myPort.write(str(' '));
      myPort.write(str(MotA));
      myPort.write('\n');
      print(states);
      print(' ');
      print(stp);
      print('\n');
      break;
    case 3:
```

```
      myPort.write(str(states));
      myPort.write(str(' '));
      myPort.write(str(Srv));
      myPort.write('\n');
      print(states);
      print(' ');
      print(stp);
      print('\n');
      break;
  }
  println("State is: "+ current_state);
  //myPort.write(states);
  //myPort.
}
public void Encoder_update(){
  //encoder_val = int((sin(frameCount*0.02)*50+140));
  /*textSize(24);
  fill(0 , 102, 153);
  strokeWeight(1);
  text("Encoder: "+(encoder_val%255), 480, 210);//160 220*/
}
public void Stepper_update(){
  //stepper_val = int((cos(frameCount*0.04)*70+100));
  textSize(24);
  fill(0 , 102, 153);
  strokeWeight(1);
  text("Stepper: "+(stepper_val%255), 480, 240);//160 260
}
public void LED_update(int l1, int l2){
  //directions of motor and stepper
  int lxpos = 480;
  int lypos = 280;
  int bl = 90;
  int bw = 40;
  if(l1==0){fill(0 , 102, 153);
  rect(lxpos, lypos, bl, bw);
  textSize(16); fill(0);
  text("FORW", lxpos+bl/4, lypos+bw/2+7);}
  else{fill(0 , 45, 85);
  rect(lxpos, lypos, bl, bw);
  textSize(16); fill(0);
  text("REVS", lxpos+bl/4, lypos+bw/2+7);}
  if(l2==0){fill(0 , 102, 153);
  rect(lxpos+110, lypos, bl, bw);
  textSize(16); fill(0);
  text("FORW", lxpos+110+bl/4, lypos+bw/2+7);}
  else{fill(0 , 45, 85);
  rect(lxpos+110, lypos, bl, bw);
  textSize(16); fill(0);
```

```
  text("REVS", lxpos+110+bl/4, lypos+bw/2+7);}
}
public void print_values(int gg1, int gg2, int gg3, int gg4, int gg5){
  textSize(18); fill(0 , 102, 153);
  text("IR "+gg1, 40, 350); //40
  text("Flex "+gg2, 210, 180); //240
  text("POT "+gg3, 90, 350); //100
  text("Servo "+gg4, 690, 350); //700
  //text("Encoder "+gg5, 530, 180); //560
  text("Motor", 500, 275);
  text("Stepper", 600,275);
  text("State"+current_state, 400,150);
}
public void push_graphs(int gg1, int gg2, int gg3, int gg4, int gg5){
  Sen1.push("IR", gg1);
  Sen2.push("Flex", gg2);
  Sen3.push("POT", gg3);
  Mot1.push("Speed", gg4);
  Mot3.push("Servo", gg5);
}
public void Stepper_Mot(int theValue) {//SendSteps
  //println("Sending step numbers: "+theValue);
  //c1=c2;
  c2 = color(0, 160,100);
  c1=c2;
  Steps();
}
public void SendSteps2(int theValue) {
  Steps2();
}
public void Steps(){
  //myPort.write(desire_step_value);
  //current_state = 2;
  myPort.write(str(2));
  myPort.write(' ');
  myPort.write(SrvAngle);//srvAngle
  myPort.write('\n');
  println(current_state);
}
public void Steps2(){
  //myPort.write(desire_step_value);
  //current_state = 2;
  myPort.write(str(3));
   myPort.write(' ');
  myPort.write(MotAngle);
  myPort.write('\n');
  println(current_state);

}
```

```
public void State2(int theValue) {
  current_state += 1;
  current_state = current_state %4;
  println("State is: "+ current_state);

  send_cont(current_state, StpAngle ,MotAngle,SrvAngle);
}
```

Resources

[1] aliexpress.com
[2] CA9 Trimmer Poteniometer
[3] Servo Motor Image