Task 7: Sensors and Motor Control Lab

Individual Lab Report #1

Job Bedford

Team C: Column Robotics
Eric Sjoberg, Rohan Thakker, Cole Gulino

ILR1
10/16/15

Individual Progress:

My main responsibility for Task 7 was the design and development of the Graphical User Interface. I also spearheaded the Communication protocol, orchestrated the initial delegation of tasks, and was the designated presenter for this Demo.

Graphic User interface:

I utilized Processing for my GUI construction and framework. Processing is an coding IDE used mainly by the artists and graphics community. It interfaces wonderfully with Arduino and has a near identical framework. The user community is also very large. Given these traits it was the ideal selection for designing a GUI interface with arduino.

The major goals of the GUI was to read all sensor values and motor states, and control at least one actuation of the system. The breakdown for the task:

Erik: DC Motor control coupled with IR Range Finder. Also implement P-control using encoder values for both position and velocity.

Rohan: Stepper Motor coupled with flex sensor for directional input. Recieves step commands from GUI.

Cole: Servo Motor coupled with Potentiometer. Also physical button for manually controlling state.

Based on this breakdown I designed the GUI starting with the passages of information. As seen on the diagram below:

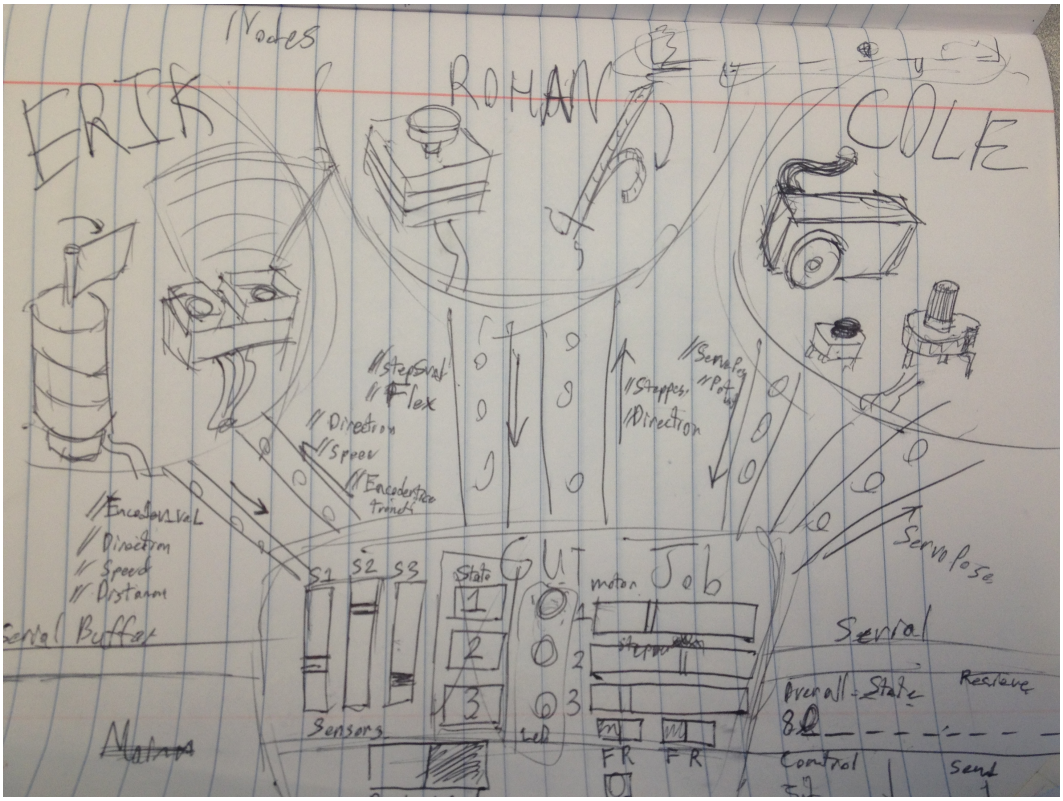

*Figure 1: Initial Breakdown of Tasks and GUI communicatio*n

Given this information, 8 inputs and 2 Outputs were required:
Input:

//State //Encoder Value // DCMotor Direction //IR Distance //Stepper Steps //Flex Sensor Value //Servo Position //Potentiometer Reading

And 2 outputs:

//Changed State //Desired Steps

Below is the GUI face:

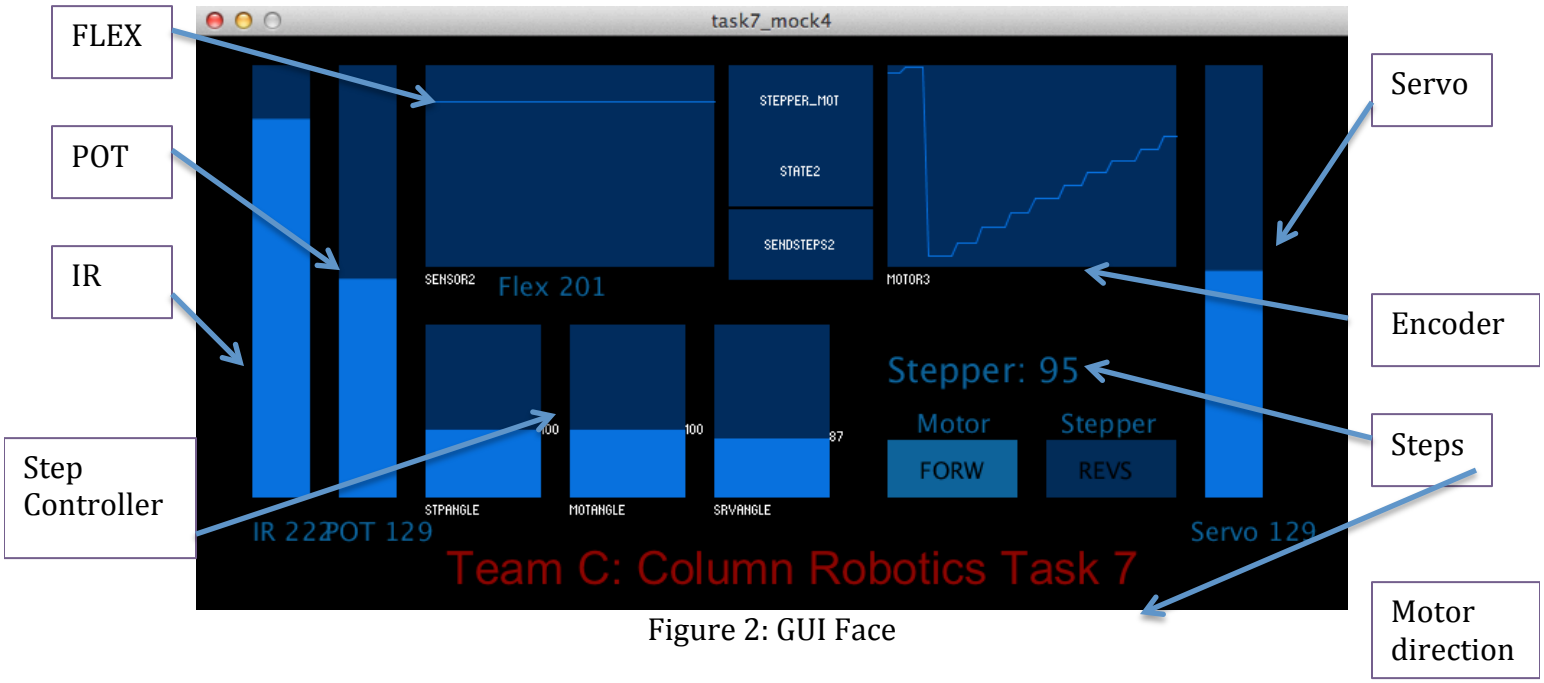

Figure 2: GUI Face

This GUI was made with the help of controlP5 library from processing.

The IR, POT, Servo values are plotted by slider indicators. The Flex and Encoder values are graphed on a time receding line graph. It was advantageous to see the direction and instantaneous change in the values this way. There are indicators for motor direction for the Stepper Motor and DC motor in the bottom left hand corner. The center buttons send the desired steps to the Arduino, The Step controller allows you to establish the desired step angle to be changed.

Communication

The communication was a serial protocol. The Arduino sends a update of the sensor values. Send a start character, and sends then the subsequent bytes are the 9 sensor values. The Processing continually polls the serial line for the start character. When one is found the program then splits the subsequent bytes into the sensor values. If it is not the start character, the program flushes the serial line.

In outputting data to the arduino system to control the stepper motor, the GUI saves the desired step angle from the slider then sends an indicator byte followed by step angle.

Challenges:

I specifically chose to pursue the GUI design because it element of the project I am least experienced in. Initially I attempted to write my own processing GUI libraries with slider bars and buttons as seen below:

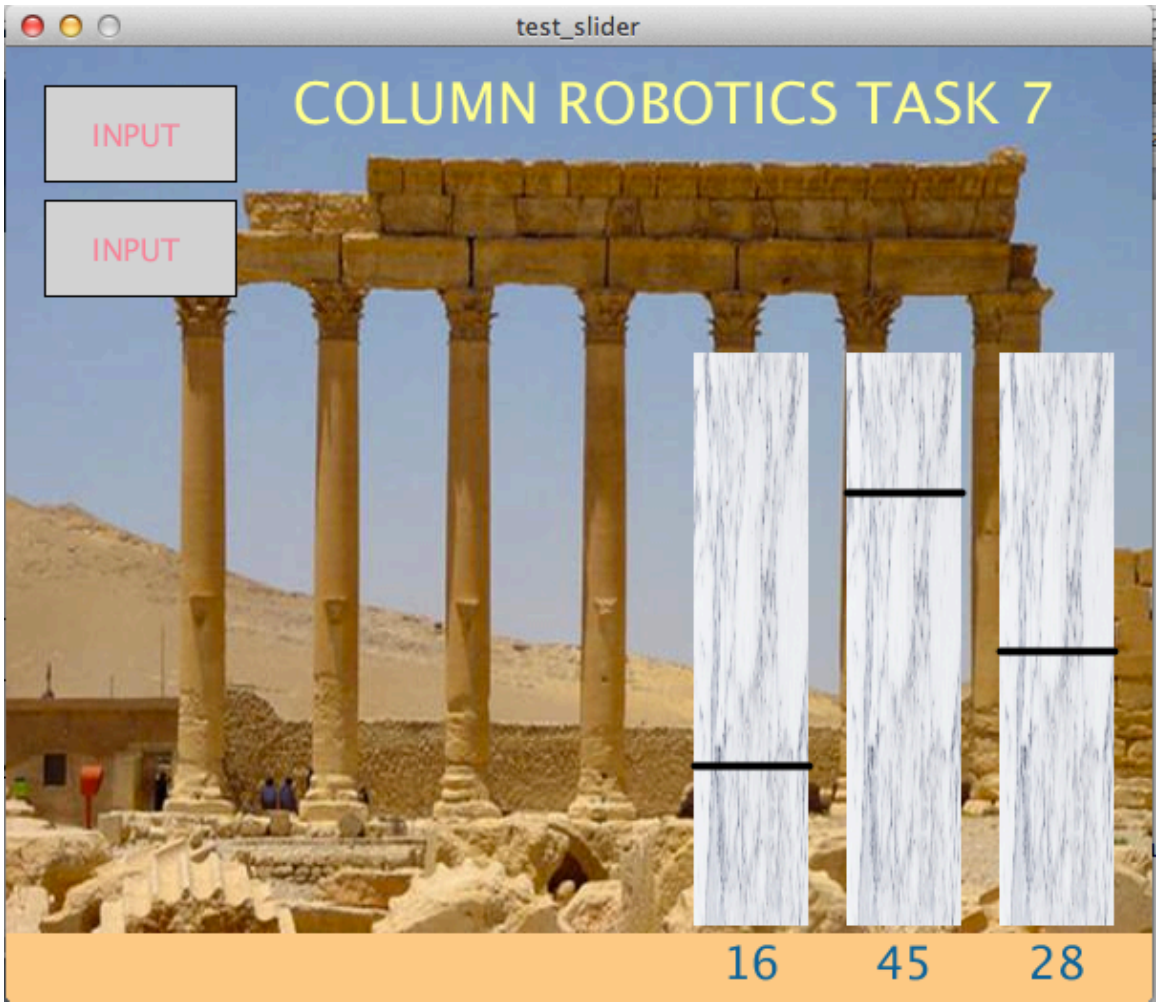

Figure 3: Unused Initial GUI Version

This approach proved too time consuming and unclean.

The other major challenge was establishing the proper output serial communication to the arduino system. The GUI could receive sensor readings fine, but in passing messages from the processing to arduino using the ParseInt() function there exist a problem with String ASCII conversion. We were able to solve the problem by converting every character to a string before writing to the serial line, but debugging this was difficult.

Teamwork

We broke down Task 7 to provide each of us with one sensor and one actuation, save the person writing the GUI. This enabled us to work as independent as possible until the integration phase.

Erik was responsible for the DC Motor control coupled with IR Range Finder. He also implement P-control using encoder values for both position and velocity.

Rohan was responsible for the Stepper Motor coupled with flex sensor for directional input.

Cole was responsible for the Servo Motor coupled with Potentiometer. He took on physical button for manually controlling state.

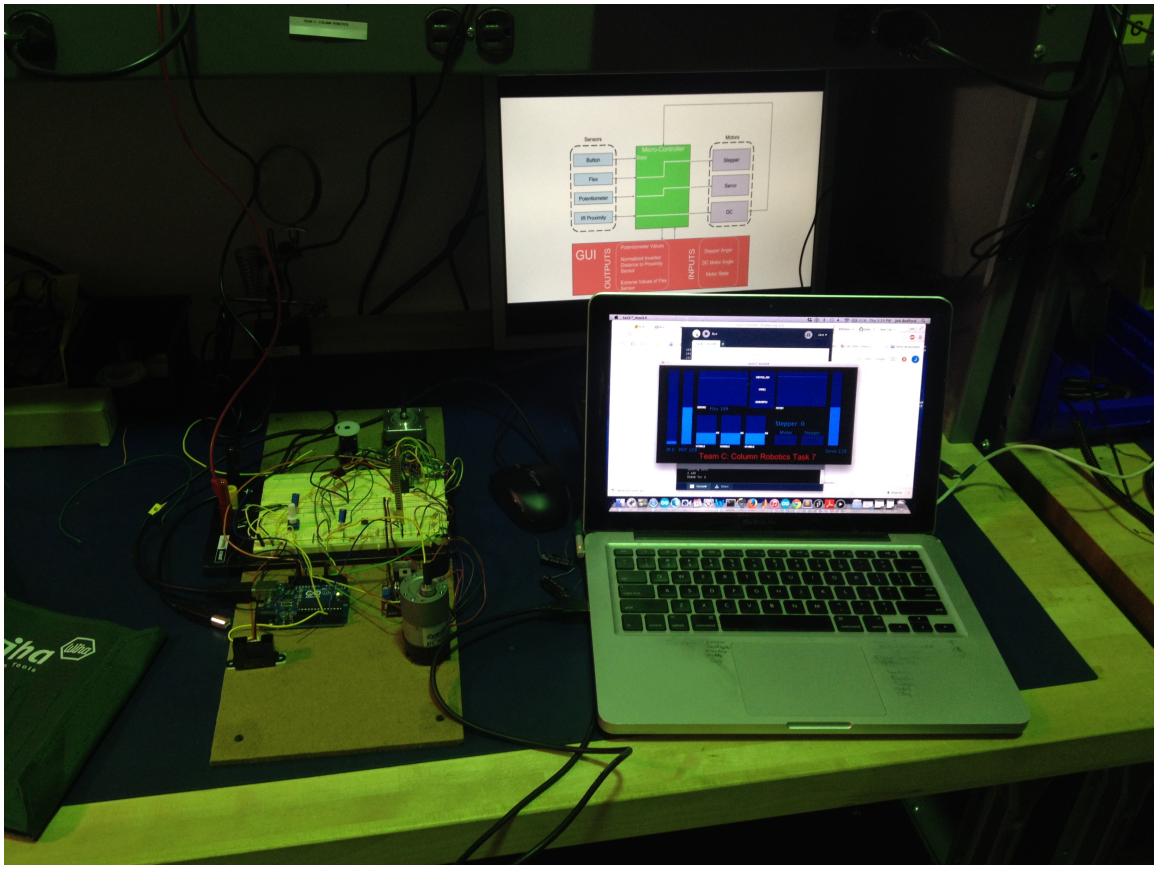

Figure 4: Overall systems view

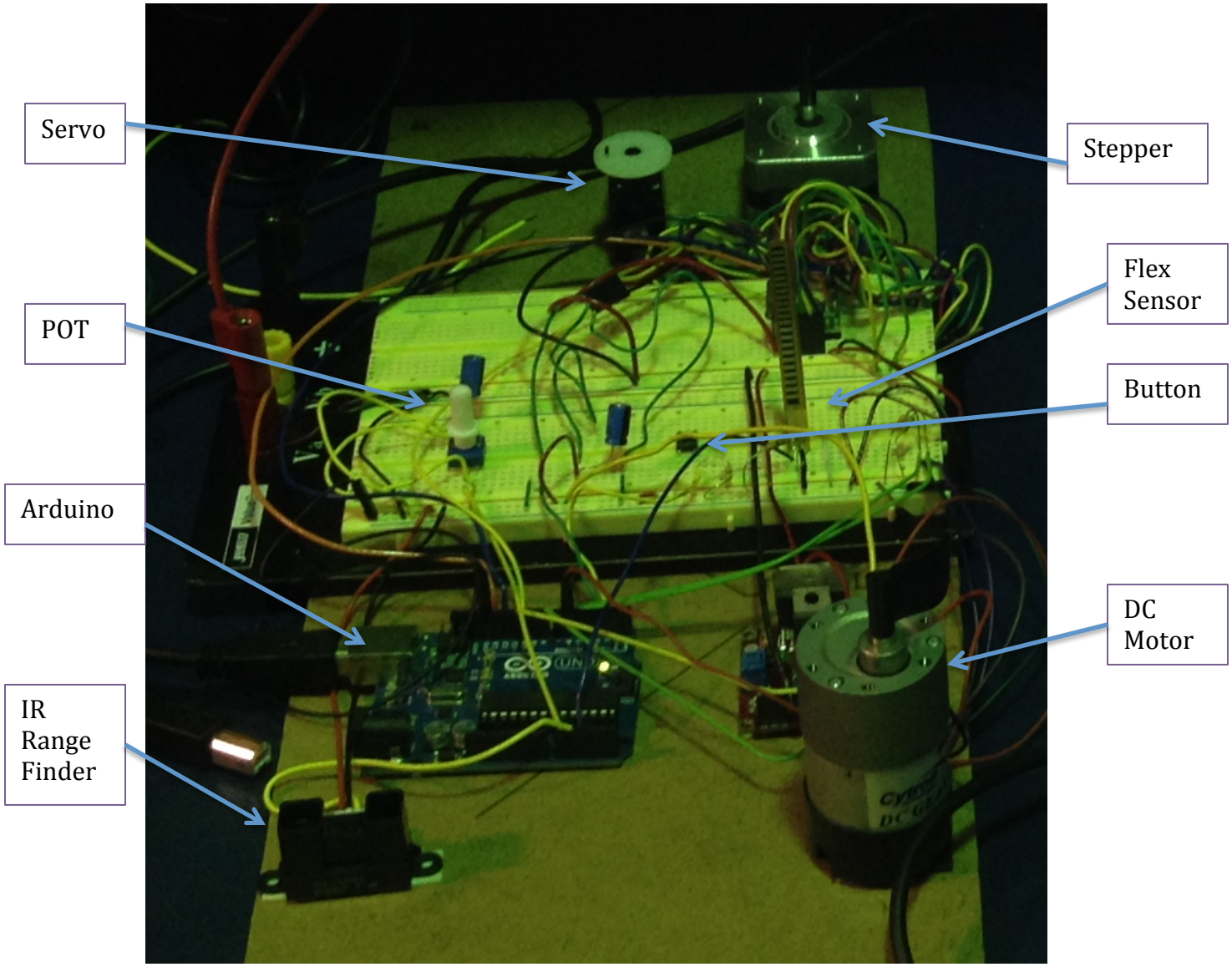

Figure 5: System - Sensors and Motors Up-close view.

Upcoming Week

This upcoming week, I will establish a ROS node to communicate with the AR Drone. The node will command low-level planar x-y movement. This process will have me first implement the ROS AR Drones API, which will take some time to understand and operate. Once the ROS API is establish, I will orchestrating messages to move the drone under tele-operated commands.

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)
          //.setStrokeWeight(1.5)
```

```
        //.setColorCaptionLabel(color(40))
        ;

Sen2 = cp5.addChart("Sensor2") //Flex Sensor
        .setPosition(160, 20)
        .setSize(200,140)
        .setRange(160, 210)
        .setView(Chart.LINE)
        //.setStrokeWeight(1.5)
        //.setColorCaptionLabel(color(40))
        ;
Sen3 = cp5.addChart("Sensor3") //Potentiometer
        .setPosition(100,20)//50 500
        .setSize(40,300)
        .setRange(0, 255)
        .setView(Chart.BAR)
        //.setStrokeWeight(1.5)
        //.setColorCaptionLabel(color(40))
        ;

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;}

  //background(color1);
  //color1 = lerpColor(c1,c2,n);
  //n+= (1-n)*0.1;
  //send_cont(current_state, StpAngle ,MotAngle,SrvAngle);
  //send stpAngle, MotAngle, srvAngle,
  //fill(StpAngle);
```

```
  //rect(0,360,width,40);

  //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);
  //println("State main is: "+ current_state);

}

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];

      //println(xpos + "\t" + ypos + "\t" + fgcolor);
      //println("next");

      serialCount = 0;
      myPort.clear();
      //println("State serial is: "+ current_state);
```

```
    //State2(1);
   }
  }
}

/*
private void controlEvent(ControlEvent theEvent){
 println(theEvent.getController().getName());
 n=0;
}
*/
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(char(states));

   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);
  //c1=c3;
  //c3 = color(160, 100, 0);
  //c1=c3;
}
```