PID Controlled Ball on Stewart Platform
Max Guerrero & Mustafa Ali
12/17/17

Motivation and Overall Concept:
As physicist's in training, we are naturally attracted to, and seek to understand physical systems. We have been trained to solve complex problems by breaking them down into their most basic and essential ingredients, then building up from there. Though we have been learning mechanics and electromagnetism from first principles, we have yet to apply these, what most consider the "bread and butter" of a classical physics to some real world application. We both expressed an interest in involving as much dynamic physics as possible. We have decided to use our Arduino Uno to sense and respond via a PID controller feedback loop to track and control a free moving ball on a flat actuated surface. The controller shall be tasked with restoring the ball to any user chosen position on the table, even if it were to be perturbed. The flat surface will have a resistive LCD touch screen to track the position of the ball bearing, and will be fastened to what is known as a Stewart Platform. This platform will be supported by six independent connecting rods driven by servo motors which will provide up to six degrees of freedom, X, Y and Z translations, roll, pitch and yaw (rotations about the X, Y, and Z axes respectively). Constructing and programming such a highly mobile platform present's its own challenges, so for this project, we will only call upon the the roll and pitch degrees of freedom, leaving the others as optional upgrades of functionality. Along with the platform moving the ball to any of a set of static user defined positions, we are also considering an add on such as moving the ball dynamically along a circle, or figure eight trajectory.

Functional Requirements:
The design will consist of a hexagonal base platform, which employs six servo motors, two for every other hexagonal face, that are attached to a triangular stewart platform. The arduino will process the position data and control the orientation of the top surface in order to ultimately direct the motion of the ball. Detection of the ball's position is done by a 5 wire resistive touch panel which tracks the position based on a nearly continuous voltage divider array concept. From this data, the velocity can be calculated through the execution of a finite difference derivative, written into our loop. The most critical part of this project will be having an accurate basis function, which when given the position will give you the desired orientation of the platform so that the ball rolls directly towards the goal. For simplicity, we will initially choose the center of the screen as the goal position. Our functional requirement will be for the ball to reach a static equilibrium, at our desired target position, as quickly as possible, no matter the initial condition.So that even if the ball is given an initial velocity towards, away, or if the ball is simply released in say a far corner, then the platform will be tasked to re-equilibrate at the goal position, as quickly as possible.

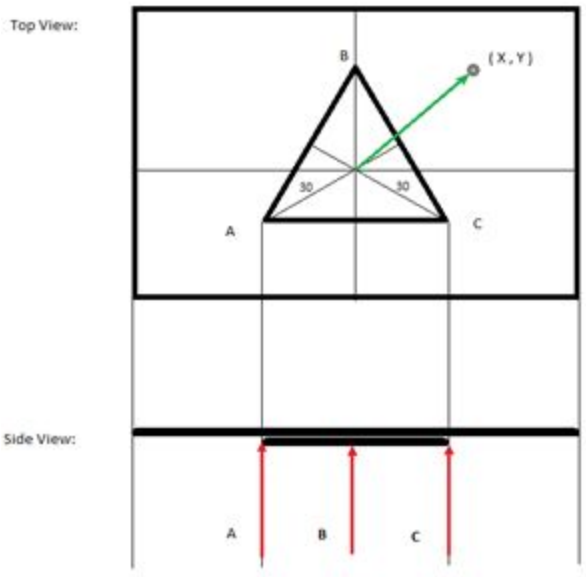
Sensors

- **17" Five Wire Resistive Touch Screen Panel USB kit (sense position of ball bearing)**
- **Keypad (trajectory control, or interface)**

PID Controlled Ball on Stewart Platform
Max Guerrero & Mustafa Ali
12/17/17

Mechanical Considerations

The primary structural material used to construct our apparatus was ¼" acrylic. It is easily machined and fastened, as well as glued. However it is fairly brittle. The actual support that will form the Stewart platform itself could be constructed of something just a bit thicker to accommodate the stresses at the joints of the supporting connecting rods. These connections will be made via ball joint connecting rods, also known as heim joints. These rod end connectors do not sustain a "moment", which makes them great to actuate the platform, since this property will keep them from binding against each other. All they can do is push or pull at a point and the fact that there's 12 of them means that all stability comes from the strict definition of the separation lengths between the servos ends and the platform mounts. This requires some adjustability of the rods so that we can adjust them to give us a level and well "zeroed" platform. From many years of engineering experience, we have learned that everything comes down to what our tolerances are, which for this project are not very tight. We foresee tolerances to be on the order of 1 mm. It is especially important to note the tolerances of the servo's actuation, because the platform relies on precise and accurate actuation, this can be compensated with shorter servo arms, though too short and we will be sacrificing speed and responsiveness. This issue is something best solved by a few experiments to check and see what combination works best. We have also decided to permanently affix the Arduino into the base such that its USB port is available, making it much portable.



Basis:

$$A = -\sqrt{3}/2\, X - 1/2\, Y;$$
$$B = Y;$$
$$C = \sqrt{3}/2\, X - 1/2\, Y.$$

PID Controlled Ball on Stewart Platform

Max Guerrero & Mustafa Ali

12/17/17

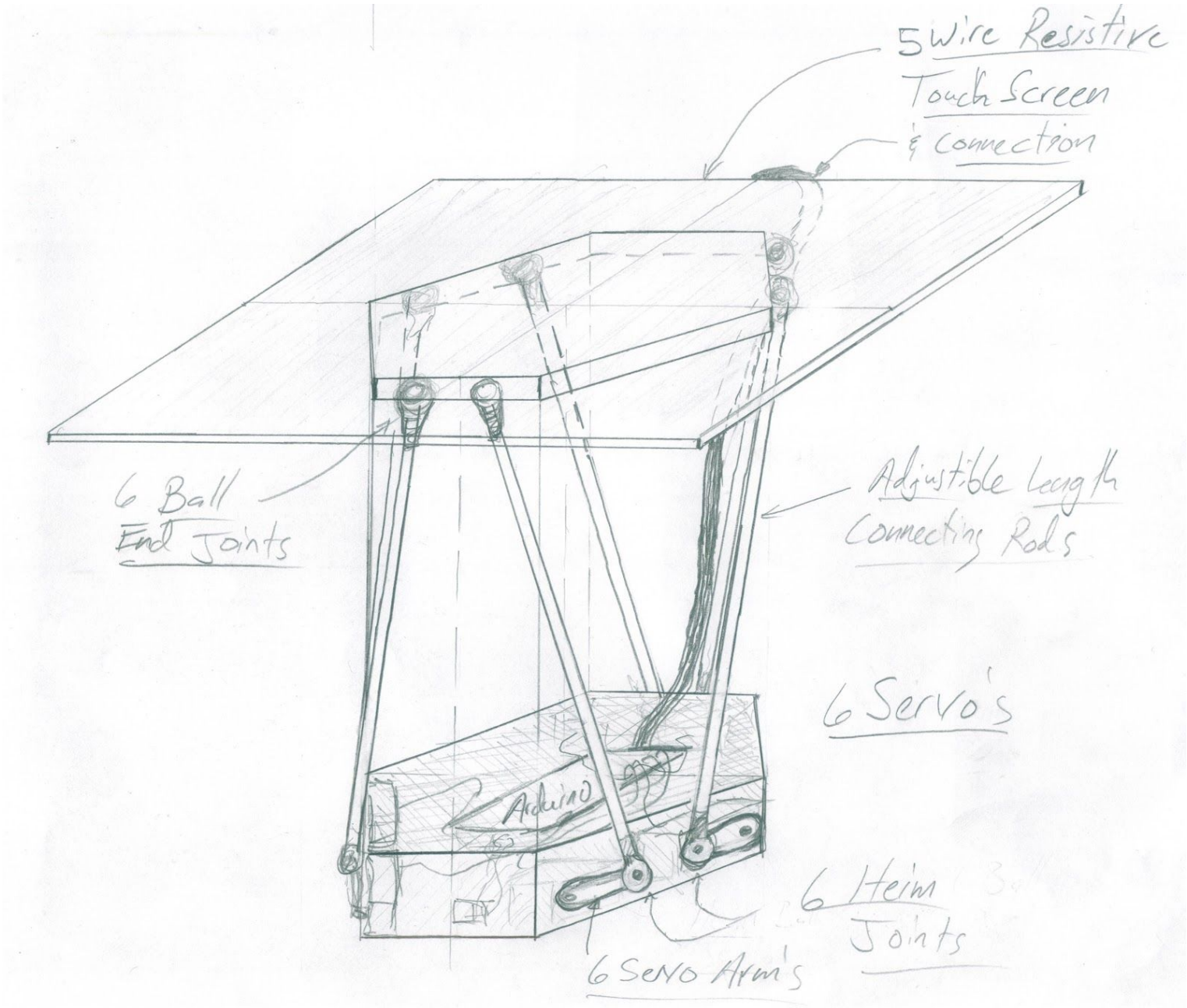
Knowns:

6 - HS5485HB Servo's

6 - CNC Machined Servo Arm's with multiple holes for adjustability

12 - Heim Joint Rod Ends

6 - Rods (Adjustable)



PID Controlled Ball on Stewart Platform

Max Guerrero & Mustafa Ali

12/17/17

Electrical Considerations:

Connecting the touchscreen:
Use ribbon connectors to connect to Analog pins A0 through A4 on Arduino.

Powering the Arduino:
Connect to computer. However, we both expressed interest in powering Arduino with a battery so that we can be more mobile for the final demonstration.

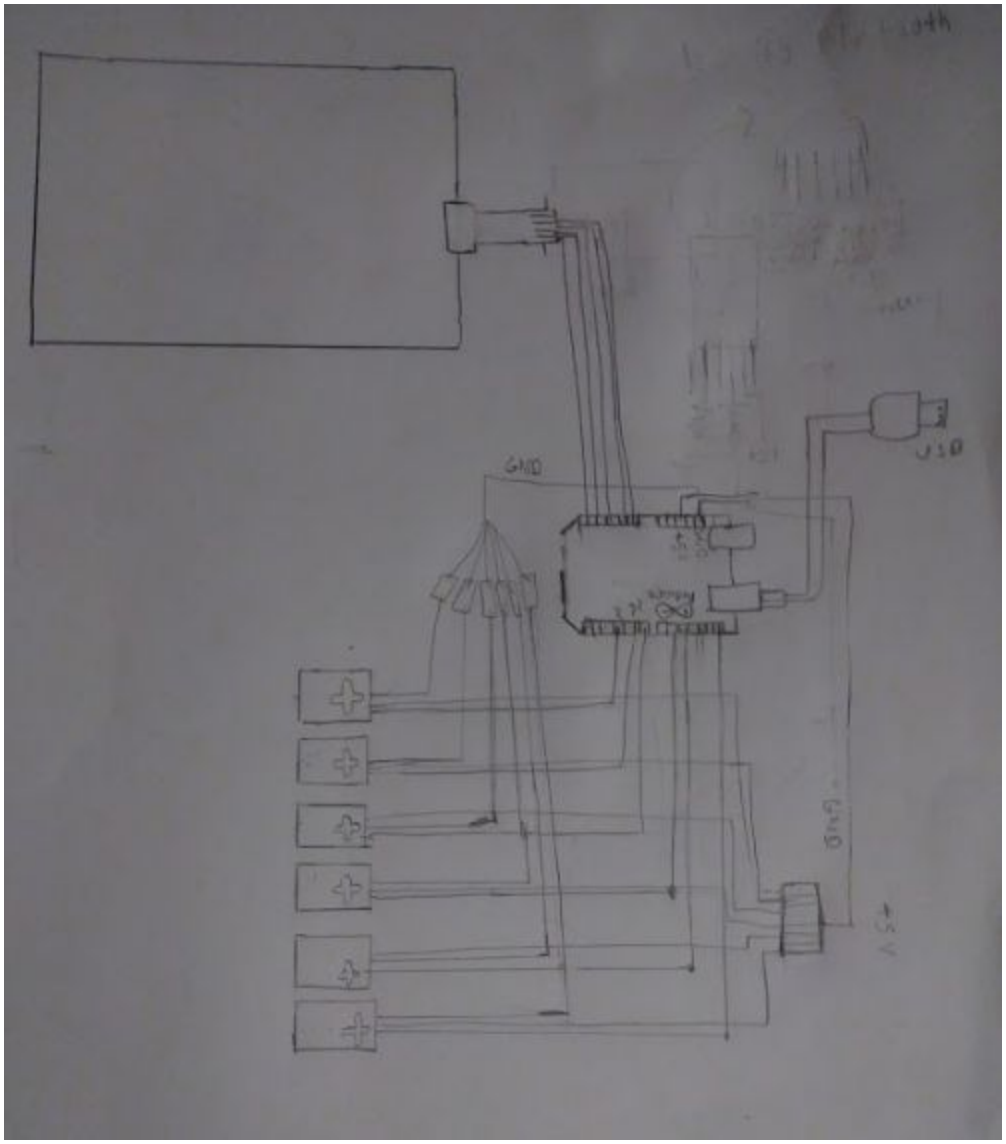
Connecting the Six-Servo's:
The servos will be connected to Digital PWM pins 3,5,6,9,10, and 11. They will be connected to a common +5Volts and common ground..

Electrical Diagrams:

PID Controlled Ball on Stewart Platform

Max Guerrero & Mustafa Ali

12/17/17



Interface

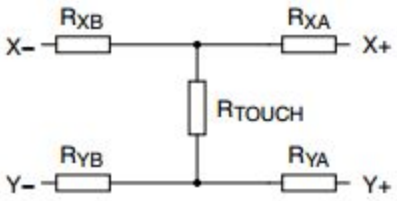
Touchscreen:

When touched with detectable pressure, the top plate makes contact with the bottom plate. When this contact occurs, the bottom layer effectively divides the top layer into two resistors in series. Similarly, the bottom layer is effectively divided into two resistors at the point of contact with top layer. Please see the figure below:

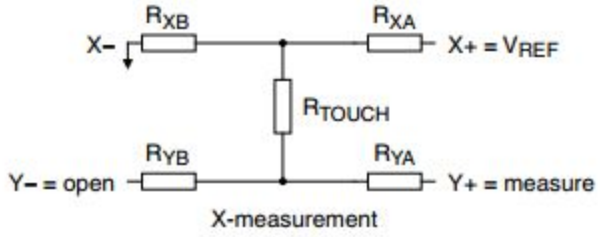
PID Controlled Ball on Stewart Platform

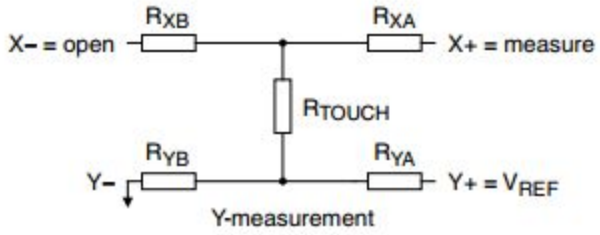
Max Guerrero & Mustafa Ali

12/17/17



By proper biasing, each plate can function as a voltage divider where the output voltage represents the rectangular coordinate of the part of contact.





Biasing the x-axis allows us to use the y-axis to measure the tap on the x-axis. In a similar manner, biasing the y-axis allows us to use the x-axis to measure the tap on the y-axis.

These are the basic principles that tell us how we are going to have to program the analog pins to read and write voltages, however we will have to wait until our touch screen arrives so that we can begin programming it and testing it for ourselves.

Software

- Arduino C, with servo libraries. Program the orientation of the servos according to stability algorithms (PID Control).

PID Controlled Ball on Stewart Platform

Max Guerrero & Mustafa Ali

12/17/17

- **Controls:** The closed loop will be as follows:
  1) The Arduino Uno will measure the voltages from the touchscreen so that it can calculate the ball's current X and Y coordinates on the screen
  2) Using a PID controls algorithm, the ball's current position and the desired position.
  3) we will then send commands to all of the servos in order to move the touch screen to the desired position.
  4) Repeat steps one through three until the ball has executed desired trajectory, or the if the ball is in the desired destination.

- **Touchscreen:** We set the origin at the geometric centroid screen. Slide your finger across the screen perfectly straight, purely horizontal line, as X would go up, Y would also go up. Next, use the Serial Monitor and take 15 data points across (what we defined as the X Axis), plot and find the trend. Repeat the same for Y - Axis. You will observe a linear correlation. Therefore, you will write down how X depends on Y and Y on X. When you compute your X coordinate, compute the Y coordinate and from this Y coordinate, determine how much X has deviated from being linearly independent and subtract that amount This new amount will result in a linearly independent X Coordinate. Repeat procedure for same Y-Coordinate. Repeat the straight line experiment for both X and Y, and you will see X and Y where orthogonal, if not, your data points were not consistent.

Safety

Everything will be powered with Arduino which is connected via USB to the computer. The Servo motors will also be powered by Arduino. Since the maximum voltage provided by the Arduino is 5V, we don't expect any safety hazards as long as we behave properly.

Parts and Reusability

The major components should be reusable. The 5 wire resistive touch screen is meant to be used on a regular day basis as long as it used responsibly. The legs attached for the servo should also be reusable.

Expansion Options

We achieved our goal of getting the ball to be restored to equilibrium in a stable manner. However we didn't achieve our intended expansion option of continuously changing the set point so that the ball can do a circle pattern. This was due to the setbacks we encountered when trying to filter out noise in the input data.

De-scope options

If we wouldn't have gotten the ball to balance into perfect equilibrium than, perhaps we would've tried to get equilibrium only along the horizontal or vertical direction. Or, we could've changed our objective and have said that we're studying oscillations about equilibrium.

Final Thoughts and Reflection of Project:

PID Controlled Ball on Stewart Platform

Max Guerrero & Mustafa Ali

12/17/17

Overall, this was a highly challenging project, however it was equally rewarding. This project required a complex integration of some very interesting mechanics as well as some technically rich software. On the mechanical side, we had to find a basis that not only spanned the two degrees of freedom we wanted, but did it without involving any other degrees of freedom, based on the geometry. This wasn't as challenging, when the small angle approximation was made.

Second on the software end, the initial challenge was figuring out exactly how we were going to program the touch screen. When we initially obtained the touchscreen we read the user manual that came with the disk, none of the parts where relevant for us as the user manual is intended for people who want to use the calibrate the touchscreen for drawing or writing on computer screen. Luckily, there was extensive documentation on how to power each of the screens built-in pins to measure the voltage in the X and Y directions.

However, the next challenge we faced once we managed to actually measure x and y voltages was that x-y and where not linearly independent, with a bit of application of linear algebra, we overcome this obstacle. The next challenge was to gain an intuitive feel for how the servos move, in order to create rotations of the platform about the origin. We realized that we really only need to program three outputs to get the desired rotations, however, due to the orientation of the servo arms, what is considered positive angle for one is actually negative for the other, so we had to be careful in implementing the movement of the servos which led to the rotations of the platform. Once everything was assembled, we proceed to write a rudimentary PID Code, just to test our proof of concept. Initially, we tested only the Proportional Term, and even when tuning the Kp, the ball wouldn't last on the platform for more than 2-3 seconds on its own. The behavior improved when we added the Derivative Term and after tuning, the best behavior we got where oscillations about the origin, but no convergence towards equilibrium. The breakthrough we made was when we decided to apply a Simple Moving Average to the derivative term, keeping 6 terms. This actually led to us achieving equilibrium for the first time. We then tuned to make our system as close to being critically damped as possible without letting the input noise lead to growing oscillations over time. Even though it is recommended in engineering practice to apply filters to the input (xy - coordinates in our case), this actually didn't work nearly as well as applying the Simple Moving Average to the derivative term.

Overall, we would have achieved our expansion option of the circle trajectory if we diagnosed the issue of noise in the earlier stages of our tuning. This problem is not unique however, there exists methods in control theory that deal with filtering out noise and obtaining new inputs which are smoother. However, this would've required extensive study so that we could've implemented it correctly into our PID Calculations. Even though we didn't achieve our expansion option, we're glad that our presentation and video demonstration achieved the goal we intended to and that we figured out how to overcome all of the issues that came into our way.

SOURCE CODE:

```
//Define touchscreen connections
#include <Servo.h>

#define Xres 100
#define Yres 125
float xSum, ySum, xAvg, yAvg;
```

PID Controlled Ball on Stewart Platform

Max Guerrero & Mustafa Ali

12/17/17

```
float lastY = 0;
float lastX = 0;
float noUpdateY = false;
float noUpdateX = false;
float yTol = 20; //Noise tolerance for x and y
float xTol = 20;
boolean redo = false;
/* Iterators used in computing running average */
float x,y; //real-time position of ball
int i,j,k;
int l = 0;
int o = 0;
int m = 0;
int n = 0;
int p = 0;
/*****************************************/

const float sqrt3 = 1.732;

const float xMin = 46.03;
const float yMin = 25.30;

const float yGoal = 0.0;
const float xGoal = 0.0;

float xPrime, xOffset;
float yPrime, yOffset;



/*Servos*/
Servo platformServo[6];
int pins[] = {3,5,6,9,10,11};
float arg[6];
/********/
float error[3];
bool noPrint = false;

/* Time Variables */
float now, deltaT, lastT;
float compT = 10;
/*****************/
```

PID Controlled Ball on Stewart Platform

Max Guerrero & Mustafa Ali

12/17/17

```
/*PID Terms*/
float kp[] = {0.175,0.175,0.175};
float kd[] = {0.10,0.10,0.10};
float ki[] = {0.00,0.00,0.00};
double ka[3] = {0.00,0.00,0.00};

double ITerm[3] = {0,0,0};

double lastError[3] = {0,0,0};
double lastLastError[3] = {0,0,0};
double lastDVal[3] = {0,0,0};
double lastLastDVal[3] = {0,0,0};

double lastPropTerm = 0;
boolean notNormal = true;
boolean firstCalc = true;
/*********/

/*output terms*/
float output[3];
float outputMedianArray[5];
float lastOutput[3] = {0,0,0};
/************/



/********/
double DValArray[3][6];
double DTerm[3];
double DValSum[3];
boolean sample = false;
boolean firstSample = true;
/********/
void setup() {
  pinMode(A2,INPUT_PULLUP);
  Serial.begin(9600);
    Serial.begin(9600);

  for(i = 0; i < 6; ++i){
    platformServo[i].attach(pins[i]);
    platformServo[i].write(90);
  }
```

```
}

void loop() {
  now = millis();
  deltaT = now - lastT;
  if(deltaT >= 2.0){
    //take mean
    controlPID();
    //then compute args
    computeArgs();
    writeServos();

    lastT = now;
  }

}

void controlPID(){
    for(j = 0; j <3; ++j){
    getCoordinates();
    if(!noPrint){
    computeError(j);
    }
  }

  for(k = 0; k < 3; ++k){
    if(!noPrint){
      computePID(k);
    }

  }

  if(!noPrint){
    computeArgs();
  }
  for(j = 0; j > 5; ++j ){
    limitOutput(j);
  }

}

void computePID(int i){
```

PID Controlled Ball on Stewart Platform

Max Guerrero & Mustafa Ali

12/17/17

```
  double propTerm = error[i]*kp[i];
  double DVal = (error[i] - lastError[i]);
  DTerm[i] = kd[i]*(((5*DVal) + (3*lastDVal[i]) + (2*lastLastDVal[i]))/2.0);
  ITerm[i] += (ki[i] * error[i]);
  double ATerm = ka[i] * (((DVal - lastDVal[i]) + (lastDVal[i] - lastLastDVal[i]))/2.0);

 if(firstSample){
   DValSum[i] = DTerm[i] + DValSum[i];
  }

  if(i == 0 && firstSample){
    DValArray[i][m] = DTerm[i];
    ++m;
  }
  else if(i == 1 && firstSample){
    DValArray[i][n] = DTerm[i];
    ++n;
  }
  else if(i == 2 && firstSample){
    DValArray[i][p] = DTerm[i];
    ++p;
  }

  if(I == 17 && firstSample){
    DTerm[i] = DValSum[i]/18;
    sample = true;
    firstSample = false;
  }
  if(sample){
   for(o = 0; o < 5; ++o){
     DValArray[i][o]  =  DValArray[i][o + 1];

   }
     DValArray[i][5] = DTerm[i];
     DTerm[i] = (DValArray[i][0] +   DValArray[i][1] + DValArray[i][2] + DValArray[i][3] +   DValArray[i][4] +
DValArray[i][5] )/6.0;
  }

  output[i] = propTerm + DTerm[i] + ITerm[i] + ATerm;

  lastError[i] = error[i];
```

```
    lastLastDVal[i] = lastDVal[i];
    lastDVal[i] = DVal;
    ++I;

}

void computeError(int i){

    if(i == 0){ //error projected onto lower left basis.
              //if error > in y direcion, then lift platform up.

      error[0] = 4*(0.5*(x - xGoal) + sqrt3*0.5*( yGoal -  y));

    }
    if(i == 1){ //erro onto lower right basis.

     error[1] = 4*(0.5*(x - xGoal) + sqrt3*0.5*( y - yGoal));

    }
    if(i == 2){

      error[2] =  4*(xGoal - x);

    }

}


void computeArgs(){
  arg[5] = 90 + output[0];
  arg[0] = 90 - output[0];

  arg[3] = 90 + output[1];
  arg[4] = 90 - output[1];

  arg[1] = 90 + output[2];
  arg[2] = 90 - output[2];

}

void limitOutput(int i ){
```

```
  if(output[i] > 181){
    output[i] = 180;
    Serial.println("ARGUMENT ON SERVO TOO LARGE! LIMITING IT!");
  }
  else if(output[i] < -1){
    output[i] = 0;
    Serial.println("ARGUMENT ON SERVO TOO SMALL! LIMITING IT!");
  }

  if(ITerm[i] > 181) ITerm[i] = 180;
  else if(ITerm[i] < -1 ) ITerm[i] = 90;
}

void writeServos(){
  /*Basis 1*/
  platformServo[5].write(arg[3]);
  platformServo[0].write(arg[4]);

  /*Basis 2*/
  platformServo[3].write(arg[5]);
  platformServo[4].write(arg[0]);

  /*Basis 3*/
  platformServo[1].write(arg[1]);
  platformServo[2].write(arg[2]);

}
void getCoordinates(){
    //let's meaa\sure x axis touch position

    //Set Left Electrodes to GND
    ySum = 0;
    xSum = 0;

    for(i = 1; i < 15; ++i){
      pinMode(A4,OUTPUT);
      digitalWrite(A4,LOW);

      pinMode(A1, OUTPUT);   //Set Right Electrodes to +5V
      digitalWrite(A1, HIGH);

      xSum = float(analogRead(A2))/(1024/Xres) + xSum;
```

```
  xAvg = float(xSum)/i;
 }                    //Measure SENSE pin (Voltage divider)


  //let's meaasure y axis touch position
for(i = 1; i < 15; ++i){
  pinMode(A1,OUTPUT);
  digitalWrite(A1,LOW);

   //Set Bottom Electrodes to +5V
   pinMode(A4, OUTPUT);
   digitalWrite(A4, HIGH);
                  //Measure SENSE pin (Voltage divider)
   ySum  = float(analogRead(A2))/(1024/Yres) + ySum;
   yAvg = float(ySum)/i;
}

xPrime = 0.2134*yAvg +39.725;
xOffset = xPrime - xMin;
xAvg = xAvg - xOffset;

yPrime = 0.0535*xAvg + 22.908;
yOffset = yPrime - yMin;
yAvg = yAvg - yOffset;

//let's "de-correlate x and y"

 x = xAvg - 43.9415;
 y = yAvg - 62.774;

 y = y - (0.0834)*x;
 x = x + (0.13182)*y;
 if( x < 40 ){

 }
 else{
 //Serial.println("No Ball on Screen");
   noPrint = true;
   firstCalc = true;
 }
```

PID Controlled Ball on Stewart Platform

Max Guerrero & Mustafa Ali

12/17/17

```
/**Check for Noise **/
if((abs(lastY - y )> yTol)&& !firstCalc){
   redo = true;
   noUpdateY = true;

 }

if((abs(lastX - x )> xTol)&& !firstCalc){
   redo = true;
   noUpdateX = true;
 }
 /*If noise, get Coordinates Again */

 if(redo){
   redo = false;
 for(i = 1; i < 15; ++i){
   pinMode(A4,OUTPUT);
   digitalWrite(A4,LOW);

   pinMode(A1, OUTPUT);   //Set Right Electrodes to +5V
   digitalWrite(A1, HIGH);

   xSum = float(analogRead(A2))/(1024/Xres) + xSum;
   xAvg = float(xSum)/i;
 }                    //Measure SENSE pin (Voltage divider)


  //let's meaasure y axis touch position
for(i = 1; i < 15; ++i){
  pinMode(A1,OUTPUT);
  digitalWrite(A1,LOW);

   //Set Bottom Electrodes to +5V
   pinMode(A4, OUTPUT);
   digitalWrite(A4, HIGH);
                    //Measure SENSE pin (Voltage divider)
   ySum  = float(analogRead(A2))/(1024/Yres) + ySum;
   yAvg = float(ySum)/i;

}

xPrime = 0.2134*yAvg +39.725;
```

PID Controlled Ball on Stewart Platform
Max Guerrero & Mustafa Ali
12/17/17

```
 xOffset = xPrime - xMin;
 xAvg = xAvg - xOffset;

 yPrime = 0.0535*xAvg + 22.908;
 yOffset = yPrime - yMin;
 yAvg = yAvg - yOffset;

 //let's "de-correlate x and y"

  x = xAvg - 43.9415;
  y = yAvg - 62.774;

  y = y - (0.0834)*x;
  x = x + (0.13182)*y;

  if((abs(lastY - y )> yTol)&& !firstCalc){
    redo = true;
    noUpdateY = true;

  }

 if((abs(lastX - x )> xTol)&& !firstCalc){
    redo = true;
    noUpdateX = true;
  }


}

 if(redo){
    redo = false;
  for(i = 1; i < 15; ++i){
    pinMode(A4,OUTPUT);
    digitalWrite(A4,LOW);

    pinMode(A1, OUTPUT);   //Set Right Electrodes to +5V
    digitalWrite(A1, HIGH);

    xSum = float(analogRead(A2))/(1024/Xres) + xSum;
    xAvg = float(xSum)/i;
  }                    //Measure SENSE pin (Voltage divider)
```

PID Controlled Ball on Stewart Platform

Max Guerrero & Mustafa Ali

12/17/17

```
  //let's meaasure y axis touch position
 for(i = 1; i < 15; ++i){
   pinMode(A1,OUTPUT);
   digitalWrite(A1,LOW);

    //Set Bottom Electrodes to +5V
    pinMode(A4, OUTPUT);
    digitalWrite(A4, HIGH);
                    //Measure SENSE pin (Voltage divider)
    ySum  = float(analogRead(A2))/(1024/Yres) + ySum;
    yAvg = float(ySum)/i;

 }

 xPrime = 0.2134*yAvg +39.725;
 xOffset = xPrime - xMin;
 xAvg = xAvg - xOffset;

 yPrime = 0.0535*xAvg + 22.908;
 yOffset = yPrime - yMin;
 yAvg = yAvg - yOffset;

 //let's "de-correlate x and y"

  x = xAvg - 43.9415;
  y = yAvg - 62.774;

  y = y - (0.0834)*x;
  x = x + (0.13182)*y;


}

  if(x < 40){//Display X and Y on Serial Monitor
    Serial.print(x);
    Serial.print(" y = ");
    Serial.println(y);
    lastY = y;
    lastX = x;
    noUpdateY = false;
    noUpdateX = false;
```

```
  noPrint = false;


  noUpdateY = false;
  noUpdateX = false;
  noPrint = false;

}
else{
//Serial.println("No Ball on Screen");
  noPrint = true;
  firstCalc = true;
}


if((x > -16 && x < 16)&&(y > -21 && y < 21) && !((x > -10 && x < 10)&&(y > -13 && y < 13))){
  //Serial.println("KP set to Normal");
  kp[0] = 0.2;
  kp[1] = 0.2;
  kp[2] = 0.2;
  kd[0] = 0.4;
  kd[1] = 0.4;
  kd[2] = 0.4;
  xTol = 20.0;
  yTol = 25.0;
}
else if((x > -10 && x < 10)&&(y > -13 && y < 13)){
  //Serial.println("KP set to small");
  kp[0] = 0.15;
  kp[1] = 0.15;
  kp[2] = 0.15;
  kd[0] = 0.4;
  kd[1] = 0.4;
  kd[2] = 0.4;
  xTol = 4.0;
  yTol = 5.0;
  x = round(x);
  y = round(y);

}
else{
  kp[0] = 0.20;
```

```
      kp[1] = 0.20;
      kp[2] = 0.20;
      kd[0] = 0.4;
      kd[1] = 0.4;
      kd[2] = 0.4;
      xTol = 20.0;
      yTol = 25.0;

    }
    firstCalc = false;
}
```