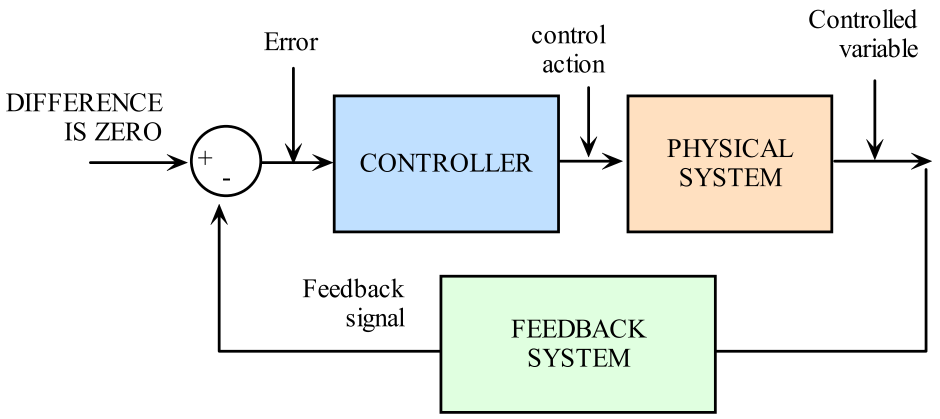
LAB 7 - SERVO PID (H)



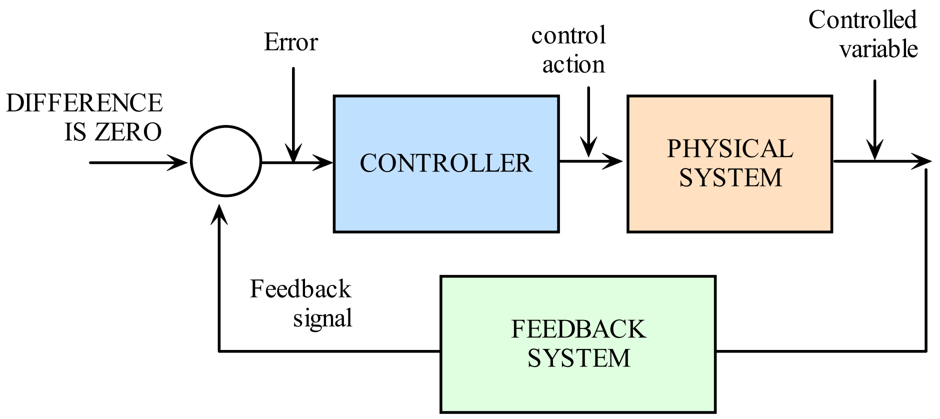
PARTS LIST

(1) Computer, Arduino & USB cable
(1) Servo motor (e.g., Hitec HS-422).
(1) Breadboard
(2) Resistors (2kΩ)
(2) Phototransistors (or photo-resistors)
(n) Hook up or jumper wire
(1) Partition

INTRO

Refer to the CT GUIDE on PID CONTROL

7.1 PID MOTOR CONTROL



INTRO

Now let's implement PID servo motor control based on the smoothed difference between 2 light detectors. The controller will "drive" the motor so that the difference in light received is zero. Reminder - since the motor is un-mounted it can't really control anything yet. We just want to see that the motor responds as expected as we manually move the light over the 2 detectors.

This exercise represents one "axis" of our solar tracker (without mounting the motor). A solar tracker could implement 2 axes to point in any direction.

HARDWARE, CIRCUIT, & CONNECTIONS

Connections and circuitry from the last exercise are unchanged. For now, we will keep the motor un-mounted. We just want to see if the motor turns as we expect as the light source is moved back and forth across the partition. This is an important step in completing the solar tracker project.

CODE

Start with the code from the last lab. Do a Save As. Then add the PID controller code to it. To start, we will only use the proportional (P) component (no D or I). Later, we may add the D component. For now you don't have to worry about it, but eventually you must keep track of sign and direction if you want the motor to turn in the correct direction. The direction of the motor depends on how it was wired, which pins are selected for sensor1 and sensor2, and how you define error (s1 - s2 or s2 - s1).

Adding the delay( ) function and/or a filtering function can smooth out any jittery motion. Adding the D component can also remove oscillations.

```
// solar_tracker_PD_volts.ino --------------------
// sense light (2 analog voltage light detectors)
// ctl variable is DIFF in output of 2 light circuits
// filter the control u (exponFilter)
// include deadband

#include <Servo.h>
Servo myservo;

int servoPin = 10;
int sensorPin1 = A1;
int sensorPin2 = A3;

void setup() // ------------------------------------------
{
  Serial.begin(9600);
  myservo.attach(servoPin);
}

void loop() // --------------------------------------------
{
  //int servoAngle;

  int sensor1 = analogRead(A1);                      //0 - 1023
  int sensor2 = analogRead(A3);
  int x = sensor2 - sensor1;                     // x = diff
  //int xexpon = exponFilterCT(x);
  float u;
  static float servoAngle = 0;
  float servoAngleFiltered;
  u = computePID(x);

  servoAngle += u;                                  // add control to servoAngle
  servoAngleFiltered = exponFilterCT(servoAngle);   // filter servoAngle

  servoAngleFiltered = constrain(servoAngleFiltered, 70, 110);// constrain value
  servoAngle = constrain(servoAngle, 0, 180);                 // constrain value

  myservo.write((int)servoAngleFiltered);          // command motor
  //myservo.write((int)servoAngle);                //option: command motor unfiltered

  float x1 = x;                                    // output to SM to debug
  float x2 = u;
  Serial.print(x1);   Serial.print("  ");
  Serial.println(x2);

  delay(50);
}

float computePID (int x) //---------------------------------
{
  float Kp = .01;             //P gain
  float Kd = 0.02;            //D gain
  float v, u;
  float xd = 0;
  float vd = 0;
  unsigned long t;
  static unsigned long to = 0;
  static float xo = 0;
  int minerror = 50;        // min error for deadband

  t = millis();
  //v = (float) (x - xo) / (t - to);      // compute veloc
  v = 0.0;
  //Serial.println(v);
  xo = x;               //reset old val for next loop
  to = t;
  u = Kp * (xd - x) + Kd * (vd - v);

  //add dead band code here if nec
  if (abs(x - xd) < minerror)
  {
    u = 0.0;
  }

  return u;
}


int exponFilterCT(int rawData) //------------------------------------
{
  static int lastFilteredData;
  float w = .05;
  int y;
  y = w * rawData + (1 - w) * lastFilteredData;
  lastFilteredData = y;
  return y;
}


//end ------------------------------------------------------------------
```

OUTPUT

If the light is shined on one sensor more than the other, the motor should turn one way. If it is shined on the other sensor more, the motor should reverse. If the light is shined on both sensors equally, the motor should stop. It may twitch or oscillate, but hopefully just a little bit. The system does not have to respond quickly. Solar trackers do not have to respond that fast.

NEXT STEPS (not this lab)

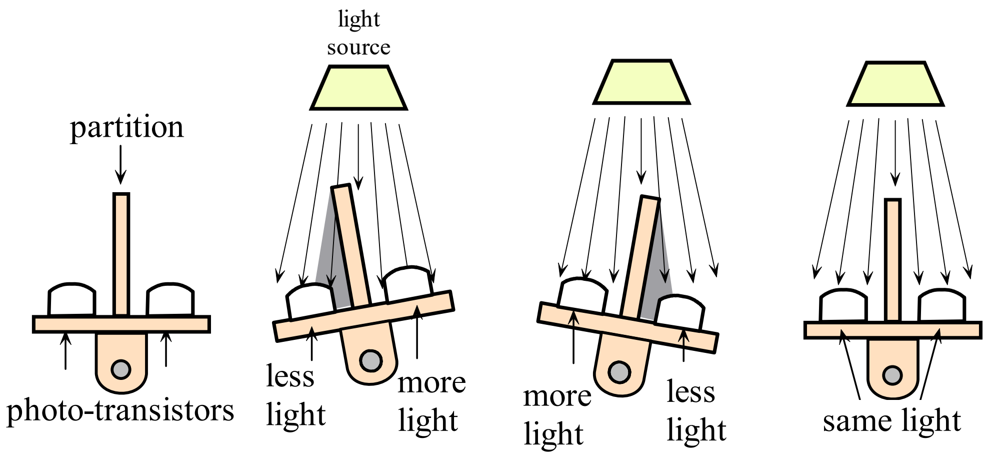
If you get this exercise to work, you are very close to completing the solar tracker project. The code in this exercise is the exact code you will use for the solar tracker project. All that is left is to design and fabricate the mechanical hardware and put it all together.

For instance, you will need to mount your sensors (phototransistor or photoresistor) and a partition to a tilting panel (note you do not actually need a solar panel!). It's probably best NOT to mount the breadboard on the tilting panel. Instead, mount the breadboard in a fixed position below and run long flexible wires to the phototransistors mounted on the tilting panel.

You must then do some mechanical design. Mount the panel to a hinge joint. Hinges are usually mounted in pairs, and there is usually a shaft. You must then mount the motor's shaft to the tilting panel somehow. This way when the motor turns, the panel is rotated. Everything must be oriented correctly. The motor shaft axis (line) should run parallel to the partition so when the motor turns the partition tilts. It is important to mount the motor shaft so that its axis is "co-linear" (aligned) with the hinge pin's axis. Otherwise the motor will bind or otherwise be loaded.

Parts can be made out of wood and cut on the laser. Build parts and assemblies in Solidworks.

If desired, you can also create a second "axis" so the panel can orient in any direction in 3D space (optional).



LAB SIGN-OFF FORM
(for students to print out)

Student Name: ___________________________

Student Name: ___________________________

1. __________ P motor control

SUBMISSION:

Students will demonstrate the working exercises to the instructor in class.

You should move the light source back and forth across the partition. Your top panel should track the light motion. Fast responsiveness is not super important. The movement should be smooth and should settle with little or no oscillation.

A possible option is uploading videos to Canvas.
The instructor will let you know which is acceptable.
See the rules for file uploads to Canvas.