```
//Ervin Meneses
//Code modifies pwm signal sent to speed controller based on power
measurements using a P.I Control Algorithm


#include <Servo.h> // Includes servo.h file from arduino library
/*****************************/
// Initializes variables

//Define Variables we'll be connecting to PID
int Setpoint=950; //Enter Desired Setpoint
float Pm, Output;
float Error, ITerm,DErr,LastPm,LastAdj;

Servo myservo;// creates a servo pin
int throttle; // creates throttle variable
float signal=0;//Signal sent to  ESC
float stampthro=0; //  Will keep track the the throttle signal the receiver
float deltathro;// change in throttle varaible
//Define Tuning Parameters
int Kp=.7, Ki=0.05, Kd=0;// These Parameters work and have been tested with
setpoint 500W, 600W, and 950W

/* The Current Sensor uses a smoothing algorithm to make the read curent value
more reliable.
Reads repeatedly from an analog input, calculating a running average and printing it
to the computer.
Keeps 10 readings in an array and continually averages them.

Define the number of samples to keep track of.  The higher the number, the more the
readings will be smoothed,
but the slower the output will respond to the input.  Using a constant rather than a
normal variable let
use this value to determine the size of the readings array.
*/

//Current sensor variables
const int numReadings = 10;
float readings[numReadings];      // the readings from the analog input
int index = 0;                  // the index of the current reading
float total = 0;                  // the running total
float average = 0;                // the average
float Im = 0;         //Initial current value measured

//voltage sensor variables
float val = 0; // variable that stores analog value of voltage sensor
```

```
float vout = 0;
float Vm = 0;
float R1 = 56000; //R1 resistor value
float R2 = 10000;//R2 resistor value
float deltaPWM=0;//adjusted pwm signal variable

/*****************************/
//Start setup
void setup(){

// initialize serial communication with computer:
Serial.begin(57600);// Make sure baud is equalk to this number


//initialize the variables we're linked to
 LastPm=0; // Last power measurment used in the derivative term
LastAdj=0;// Last value that we changed the PWM width by

// initialize all the readings to 0:
for (int thisReading = 0; thisReading < numReadings; thisReading++)
   readings[thisReading] = 0;

// set pin 9 as motor pin
  myservo.attach(9);
//sets pin 5 as the signal receiving pin
 pinMode(5, INPUT);

}


/*****************************/
//Start loop
void loop()
{



//voltage variables
val = analogRead(A0); //read voltage from analog signal
vout = (val/1024)*5;// converts signal to value between 0-5V
//Measured voltage
Vm = ((vout*(R1+R2))/R2)*.973; // Calculates input voltage



//Current Variables
```

```
   // subtract the last reading:
   total= total - readings[index];
   // read from the sensor:
   readings[index] = analogRead(A7); //Raw data reading

   readings[index] = (readings[index]-510)*5/1024/0.04-0.04;//Data
processing:510-raw data from analogRead when the input is 0; 5-5v; the first 0.04-
0.04V/A(sensitivity); the second 0.04-offset val;

  // add the reading to the total:
   total= total + readings[index];
  // advance to the next position in the array:
   index = index + 1;
   // if we're at the end of the array...
   if (index >= numReadings)
 // ...wrap around to the beginning:
  {index = 0;}
 // calculate the average:
average = (total/numReadings);   //Smoothing algorithm
(http://www.arduino.cc/en/Tutorial/Smoothing)
Im= average;


Pm=Im*Vm; // Measured power

throttle = pulseIn(5, HIGH,25000); //  ( Reads width of pwm signal going into pin 5
from Receiver)


//Computer error variable for Output Calculation
  Error=Setpoint-Pm;//Error
myservo.writeMicroseconds(throttle);

// Conditional statements to determine whether or not PID controller should
engage

  if (Error>0)// Meaning that the power measurment input is below our setpoint
  // then we just let the PWM signal from the receiver go throguh to the ESC
  { myservo.writeMicroseconds(throttle);

  }


  if(Error<0)  // if the power measurment input is greater than the setpoint then we
want to PID to take action
```

```
 {

   //Once you know the error is negative then we need toadjust the PWM
  while(Error<0)// The while loop will check the Error and reduce the PWM signal
until the error is positive again
  // Within this while loop there is another while loop that is used to maintain/send
the  adjusted PWM signal at the point at which it is right below the setpoint

   {

 // Begin Power calculation within the while loop
      //voltage variables
val = analogRead(A0); //read voltage from analog signal
vout = (val/1024)*5;// converts signal to value between 0-5V
//Measured voltage
Vm = ((vout*(R1+R2))/R2)*.973; // Calculates input voltage

   // subtract the last reading:
   total= total - readings[index];
   // read from the sensor:
   readings[index] = analogRead(A7); //Raw data reading

   readings[index] = (readings[index]-510)*5/1024/0.04-0.04;//Data
processing:510-raw data from analogRead when the input is 0; 5-5v; the first 0.04-
0.04V/A(sensitivity); the second 0.04-offset val;

  // add the reading to the total:
   total= total + readings[index];
  // advance to the next position in the array:
   index = index + 1;

   // if we're at the end of the array...
   if (index >= numReadings)
   // ...wrap around to the beginning:
 {index = 0;}
 // calculate the average:
average = (total/numReadings);   //Smoothing algorithm
(http://www.arduino.cc/en/Tutorial/Smoothing)
Im= average;


Pm=Im*Vm; //measured power
throttle = pulseIn(5, HIGH,25000); //  ( Reads width of pwm signal going into pin 5
from Receiver)
```

```
Error = Setpoint-Pm;// Checks on setpoint everytime there is reiteration



// PID Calculations
ITerm+=(Error*Ki); // Term due to integral
DErr=-(Pm-LastPm);// Term due to derivative


  Output= Kp*Error +ITerm +Kd*DErr;

  if (Output<-90){ Output=-90;} // limits our  ouptut to the maximum amount of error we anticipate before engaging the SAE Power Limiter
  //else if (Output>0){Output=0;}

 // Convert output to a value to a width in PWM which we should reduce the current PWM
 // we know that a 1% increase in throttle corrsponds to a PWM width increase of .004ms
 // From the current, Voltage, and power data collected in the fall term I found that a .5 increase in the throttle position caused an increase of about 160W when near the
 //engaging point of the SAE power limiter
 // .5 increase in throttle position translates to a .08ms (80us) increase in PWM width

 // deltaPWM is a negative number
 deltaPWM= Output*80/160;// converts the PID output to a pwm width by which the current throttle position should be adjusted by

  // Varaibles to remember for each time loop is called
  LastPm=Pm; //Last power measured equals current power measurement
  LastAdj+=deltaPWM;//Sums up the adjusted PWM values



  Kd=Kd;
  Ki=Ki;
  Kd=Kd;



signal=throttle+LastAdj;//difference between actuall throttle signal and value by which it needs to be adjusted
myservo.writeMicroseconds(signal); // Sends adjusted PWM signal to ESC
stampthro=signal-LastAdj;// throttle signal that keeps a steady level
```

```
 }// close first while loop

 //At this point we know that the error>0 , the signal is some pwm signal and that the stampthro= throttle, which is greater than 0



}// end of if statement error<0




// at this point we can keep the power around 500w,but I cant reduce it

 // This while loop keeps keeps you at a level near the Set point until you throttle down below the "signal"
 while ( stampthro>0)
 {

    myservo.writeMicroseconds(signal);
        throttle = pulseIn(5, HIGH,25000);

   if (throttle<=signal)
   {break;}


 }


   // clear variables
stampthro=0;
signal=0;
LastAdj=0;

 }// close  void loop
```