```
#define NINT 100
#include <Wire.h>
#include <SPI.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BMP280.h>


// setting up the barometric pressure sensor
#define BMP_SCK 13
#define BMP_MISO 12
#define BMP_MOSI 11
#define BMP_CS 10
Adafruit_BMP280 bme(BMP_CS);

// defining variables we will need to convert the output of the thermistor (see datasheet)
#define SERIESRESISTOR 10000
#define THERMISTORNOMINAL 10000
#define TEMPERATURENOMINAL 25
#define BCOEFFICIENT 3950

int pressureInput = A0;
int motor = 9;
int offpin = 10;
int thermisterPin = A5;
int mafPin = A3;
int newOutput;
int n;

// setting the desired test pressure
float setPressure = 2.4884;

int currOutput;
int iloop; int ierr;
float Error_History[NINT];
int Time_old = 0; int dataPoint = 0;
unsigned long testPressure = 0;
boolean maxpressure;
boolean Stop;
boolean stay;
unsigned long takeDataTime = 0;
float mafData[50];
float tempData[50];
float pressureData[50];
```

```
// you will need to change these values appropriately to tune your controller
float proportionalGain = 140;
float integralGain = 60;

void setup() {
  Serial.begin(9600);
  // setting up the barometric pressure sensor
  if (!bme.begin()){
    Serial.println("Could not find a vliad BMP280 sensor, check wiring.");
    while (1);
  }
  pinMode(pressureInput, INPUT);
  pinMode(motor, OUTPUT);
  pinMode(mafPin, INPUT);
  pinMode(thermisterPin, INPUT);
  pinMode(offpin, INPUT);
  currOutput = 0;
  newOutput = 0;
  iloop = 0;
  ierr = 0;
  maxpressure = false;
  Stop = false;
  stay = false;
}

void loop() {
    unsigned long Time = millis();

    // you will only need one barometric pressure sensor, so this line is commented out for now
    //Serial.println(bme.readPressure());
    iloop++;

    // reading and converting the test pressure
    int inputVal = analogRead(pressureInput);
    float pressureVal = (5.0 * (float)inputVal/1023.0);
    float pressure = (((pressureVal/5.0)-.04)/.09)+0.06;

    // calculating the error
    float Error = setPressure - pressure;

    // calculating the integrated error
    int dT = Time-Time_old;
    Time_old = Time;
    float Error_old = Error;
```

```
  Error_old = Error_History[ierr-1];
  Error_History[ierr] = Error;
  ierr++;
  ierr = ierr % NINT;
  float Error_Int = 0;
  for (int i = 0; i<NINT; i++){
   Error_Int += Error_History[i];
  }
  Error_Int *= (float) dT/NINT;

  // if error is within a small margin, keep the PWM value constant and take measurements for
2 seconds
  // this time will need to be adjusted depending on you current limit for your motor controller
  if (Error < 0.20) {
    if (maxpressure == false) {
      testPressure = Time;
      maxpressure = true;
      takeDataTime = Time;
      dataPoint = 0;
      stay = true;
    } else {
      // Time - testPressure > 2000, the code will wait for two seconds before it stops taking
measurements
      // this corresponds to two seconds, although this value may need to be tweaked to ensure
that you are at steady state
      if (Time - testPressure >2000 && Stop == false){
       Stop = true;
       float mafAverage = 0.0;
       float tempAverage = 0.0;
       float pressureAverage = 0.0;
       int counter = 0;
       // after the controller has stopped taking measurements, it will average all
measurements taken over that time period and average them
       // if your time period is too long, you may need to increase the size of the mafData,
tempData, and pressureData arrays
       for (int i = 0; i < 50; i++) {
         if (mafData[i] != 0.0){
           mafAverage = mafData[i] + mafAverage;
           tempAverage = tempData[i] + tempAverage;
           pressureAverage = pressureData[i] + pressureAverage;
           counter++;
           delay(50);
         }
       }
```

```
      // printing out the average data values
      Serial.print("MAF Average Value: ");
      Serial.println(mafAverage/(float)counter);
      Serial.print("Temperature Average Value: ");
      Serial.println(tempAverage/(float)counter);
      Serial.print("Pressure Average Value: ");
      Serial.println(pressureAverage/(float)counter);


    }
    else {
      stay = true;
      // during the measurement period, data will be collected every 40ms
      if (Time - takeDataTime > 40){
        // reading and converting the airflow from the mass airflow sensor (see data sheet for
exact conversion)
        int airflow = analogRead(mafPin);
        float voltage = (float)airflow * (5.0 / 1023.0);
        float calculatedAirflow = 84.7 - (197.0*voltage) + (158.0*voltage*voltage) -
(45.1*voltage*voltage*voltage) + (6.12*voltage*voltage*voltage*voltage)- 9.68;
        mafData[dataPoint] = calculatedAirflow;

        // reading and converting the test pressure (see datasheet)
        int inputVal = analogRead(pressureInput);
        float pressureVal = (5.0 * (float)inputVal/1023.0);
        float pressure = (((pressureVal/5.0)-.04)/.09)+0.06;
        pressureData[dataPoint] = pressure;

        // reading and converting the temperature (see datasheet)
        float reading;
        reading = analogRead(thermisterPin);
        reading = (1023 / reading) - 1;
        reading = SERIESRESISTOR / reading;
        float steinhart;
        steinhart = reading / THERMISTORNOMINAL;
        steinhart = log(steinhart);
        steinhart /= BCOEFFICIENT;
        steinhart += 1.0 / (TEMPERATURENOMINAL + 271.15);
        steinhart = 1.0 / steinhart;
        steinhart -= 273.15;
        tempData[dataPoint] = steinhart;
        dataPoint++;
        takeDataTime = Time;
```

```
        }
      }
    }
  }

  // if the pressure is NOT within 0.2 of the desired test pressure, calculate the new output by
multiplying the proportional gain by the error
  // and summing with the integral gain multiplied by the integrated error
  float Output = proportionalGain * Error + integralGain * Error_Int;

  // constrain the output to values from 0 to 255 (PWM output)
  newOutput = constrain((int)Output, 0, 255);

  // set the output to 0 if the cycle is done
  if (Stop == true) {
    newOutput = 0;
  }
  // set the output to the current output if we are within the testing window
  else if (stay == true) {
    newOutput = currOutput;
  }

  // these if statements will slowly ramp the PWM values up and down
  // the PI controller will operate better without this ramping up and down, but this will prevent
you from tripping a breaker
  // as easily if your power draw is close to the breaker limit
  // note that if you are using the method, delay values may need to be tweaked to ensure that
the controller is not ramping
  // up too fast or too slow
  delay(5);
  if (newOutput > currOutput) {
    currOutput = currOutput + 1;
    analogWrite(motor, currOutput);
  }
  else if (currOutput > newOutput) {
    currOutput = currOutput - 1;
    analogWrite(motor, currOutput);
  }
 }
```