```
#include <TimerOne.h>

/*
 * This Block will set all of the pins for inputs and outputs
 */
const int vPIN = A0;
const int iPIN = A2;
const int leadPIN = A4;
const int distPIN = A6;
const int PWMpin = 4;
const int buck_shut = 10;
const int dist_shut = 8;

// Sets the room temp value and beta value to be read from thermistor
const float room_temp = 298.15;
const float beta_const = 3977;

//Creats global variables for panel characteristics
float Pin, Vin, Iin, prev_Pin = 0, prev_Vin = 0;
float lead_temp, dist_temp;

//Initailizes a 35% duty cycle
int duty_cycle = 89;

void setup()
{
TCCR0B = TCCR0B & B11111000 | B00000010;    // set timer 0 for PWM frequency
of  7812.50 Hz
Serial.begin(9600);
}

void loop()
{
  MPPT();
   temp_sense();
   charge_shutdown();
 // send_bluetooth();
}

/*
 * This function is used to track the Max Power Point of the Solar Panel
 * and is used to adjust the duty cycle of the buck converters oscillations.
 */
```

```
void MPPT(){
 float senseV = analogRead(vPIN); //reads voltage pin
 float senseI = analogRead(iPIN); //reads current pin
 Vin = ((senseV + .5) * 5.0 / 1023.0) * 100;  //takes 10-bit data and
converts it to voltage
 Iin = ((senseI + .5) * 5.0 / 1023.0) * 10;   //takes 10-bit data and
converts it to current
 Pin = Vin * Iin;   //calculates the of the solar panel is inputing into the
system

//This series of if-statements follows the P&O MPPT algorithm
 if(Pin == prev_Pin){
    analogWrite(PWMpin, duty_cycle);
 }
 else if((Pin > prev_Pin && Vin > prev_Vin) || (Pin < prev_Pin && Vin <
prev_Vin)){
    if(duty_cycle < 255){
      duty_cycle++;
    }
  }
  else if((Pin>prev_Pin && Vin<prev_Vin) || (Pin<prev_Pin && Vin>prev_Vin)){
    if(duty_cycle > 0){
      duty_cycle--;
    }
  }

  analogWrite(PWMpin, duty_cycle); //writes new duty cycle
  prev_Pin = Pin; //sets previous power to current power for next loop
  prev_Vin = Vin; //sets previous voltage to current voltage for next loop

/*
 * This block is used to print to the serial monitor for the purposes of
debugging
 */
  String v1 = "Voltage = ";
  String v2 = v1 + Vin;
  String i1 = "Current = ";
  String i2 = i1 + Iin;
  String p1 = "Power = ";
  String p2 = p1 + Pin + "\n";

   Serial.println(v2);
   Serial.println(i2);
   Serial.println(p2);
```

```
}

void temp_sense(){
  float sense_lead = analogRead(leadPIN);
  float sense_dist = analogRead(distPIN);

//Used 1/T = 1/T0 + 1/B * ln( ( adcMax / adcVal ) - 1 ) equation to convert
from voltage
//to an temp^-1 value
  float inverse_lead = (1/room_temp) + 1/beta_const * log((1024 / sense_lead)
- 1);
  float inverse_dist = (1/room_temp) + 1/beta_const * log((1024 / sense_dist)
- 1);

//inverts temp and converts to celceius
  float lead_C_temp = (1/inverse_lead) - 273.15;
  float dist_C_temp = (1/inverse_dist) - 273.15;

//converts celcius to farenheit
  lead_temp = lead_C_temp * (9/5) + 32;
  dist_temp = dist_C_temp * (9/5) + 32;

/*
 * This block is used to print to the serial monitor for the purposes of
debugging
 */
  String L1 = "Lead-Acid Tempurature = ";
  String L2 = L1 + lead_temp + char(194) + char(176) + " F";
  String D1 = "Distributor Tempurature = ";
  String D2 = D1 + dist_temp + char(194) + char(176) + " F\n";

   Serial.println(L2);
   Serial.println(D2);
}

/*
 * This function will write HIGH or LOW signals to the Buck relay or
 * Distributor relay in order to shut the controller down if over-
 * heating occurs
 */
void charge_shutdown(){
   if(lead_temp > 113 || dist_temp > 113){
     digitalWrite(buck_shut, HIGH);
     digitalWrite(dist_shut, HIGH);
   }
   else{
     digitalWrite(buck_shut, LOW);
     digitalWrite(dist_shut, LOW);
   }
}

/* This block will be filled in by Manny and His blutooth app
void  send_bluetooth(){

}
*/
```