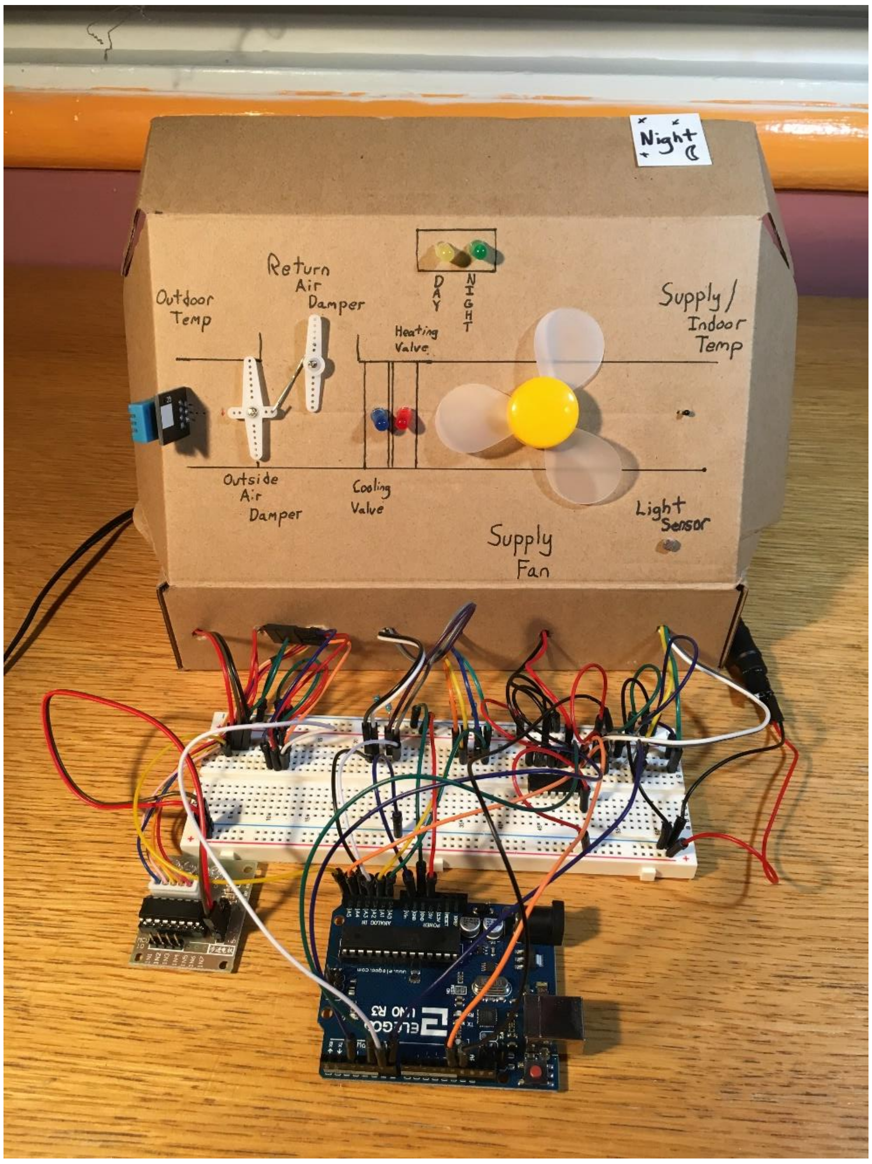
HVAC AHU Economizer Control Arduino Uno Project



By James B on October 10, 2018

Purpose

This project started after taking a 3-day Energy Auditing workshop and having a week's worth of Arduino tinkering at home. I figured, it would be a great way to nail down some HVAC economizer principles and further develop micro controller and C/C++ skills. Woohoo! The objective was to create a Packaged AHU economizer control system with the sensors available in the Elegoo Uno R3 Starter Kit (knock-off Arduino Uno). Simply, the goal was to have a few inputs that successfully control several outputs.

What the heck is an HVAC, AHU, Economizer?!

For the benefit of the non-energy geeks out there, let us dispel some acronym soup. Heating Ventilation and Air Conditioning (HVAC) is a fancy name for the building systems that condition and filter the outdoor air coming into the space.

An Air Handling Unit (AHU) is a large piece of equipment that rests on roof-tops of buildings that push and treat the air flowing into the building. We can think of them as big window ACs.

An economizer is an energy saving feature of an AHU. It usually consists of a couple actuators that control an outside and return air damper with control logic. The idea is to avoid using "mechanical" energy to cool or heat the air when it is possible to use outside air use OA to satisfy the building air requirements. We can think of this feature akin to opening a window.

Sensors and Devices

Type	Model	Device
Environment	West Coast Breeze	Upside down can of Dust Off
	"Glow-bulb Warming"	60-watt incandescent bulb
Inputs	Outdoor Air Temp (OAT)	DHT11 Temp & Humidity Module
	Supply Air Temp (SAT)	Thermistor
	Light Sensor	Photoresistor
Outputs	Night Indicator	Green LED
	Day Indicator	Yellow LED
	Cooling Valve	Blue LED
	Heating Valve	Red LED
	Outside Air Damper (OAD) & Return Air Damper (RAD) Actuators	SG90 Servo Motor
	Supply Fan	Hobby motor driven by an L293D w/ independent 9v power supply 2
Main Board	Controller	Elegoo Uno R3 powered with a 9v power supply
	Breadboard	Rail 1 has 5v from Elegoo and Rail 2 has 9v from power supply 1

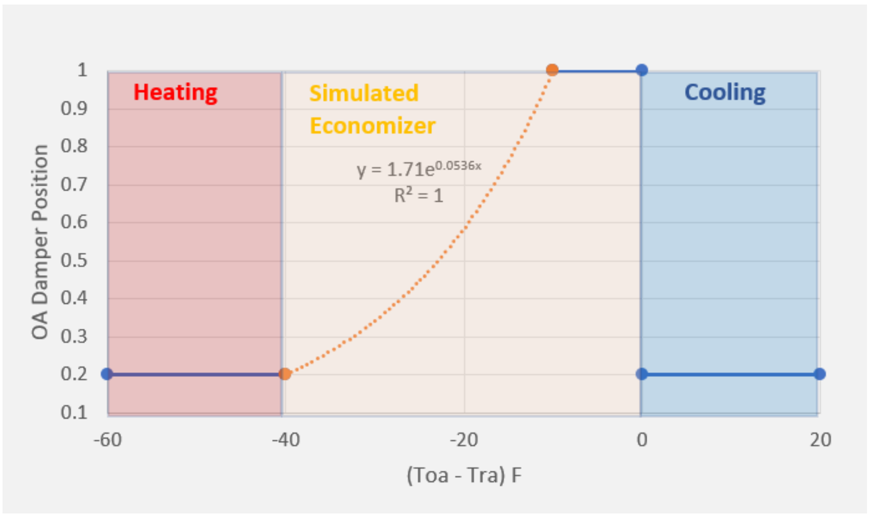
Control Logic

This system is representative of an AHU with an air-side economizer using non-integrated controls (i.e., 3 modes of operation). AHU are complex pieces of equipment and to model all the feedback loops with this limited demonstration would not be feasible. To keep within scope a list of assumptions were created.

Assumptions

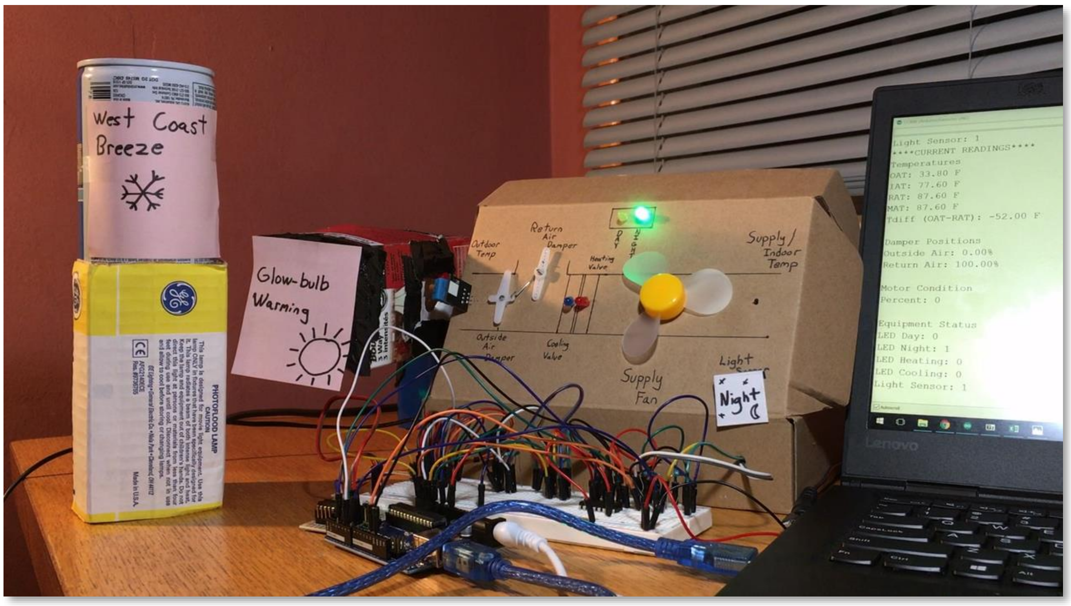
1) Return air temperature (RAT) = SAT +10F
2) Building is occupied after sunrise unoccupied after sunset
3) Minimum ventilation air requirements are 20% outside air (OA) requirement
4) Economizer control mode is enabled when -40F <= Tdiff < 0F (Tdiff = OAT-RAT)
5) Economizer function is exponential

I'd like to make note of a couple assumptions that are listed above. The RAT would typically have its own sensor, rather than being a function of SAT and is fairly constant. Another variance is the economizer control. Typical economizer controls are a function of RAT, OAT, MAT, and SAT. To simplify this, I modeled an actual economizer using OAD and Tdiff (i.e., OAT-RAT) variables and used an exponential regression model to create this simulation. In order to demonstrate the different sequences, the economizer mode was restricted to a Tdiff between -40F and 0F, rather than something more realistic like between -100F and 0F. The simulated AHU control operation is graphed below.



Control Sequence 1 – System Scheduling

The AHU will shut down when there is no light detected by the photoresistor. The night indicator will illuminate and the OAD position will close to 0%.

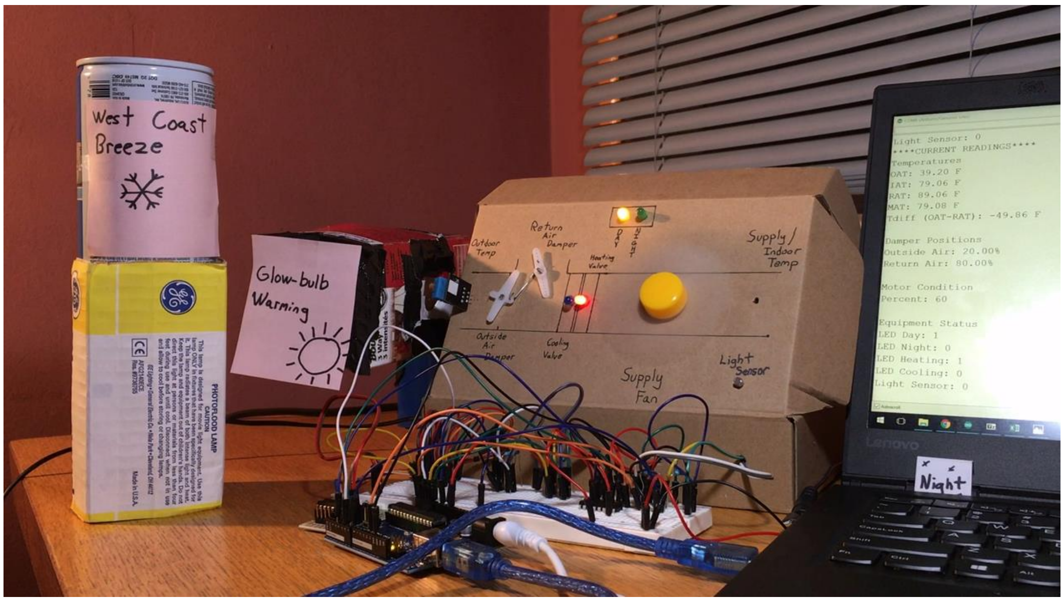


```
Light Sensor: 1
****CURRENT READINGS****
Temperatures
OAT: 33.80 F
IAT: 77.60 F
RAT: 87.60 F
MAT: 87.60 F
Tdiff (OAT-RAT): -52.00 F

Damper Positions
Outside Air: 0.00%
Return Air: 100.00%

Motor Condition
Percent: 0

Equipment Status
LED Day: 0
LED Night: 1
LED Heating: 0
LED Cooling: 0
Light Sensor: 1
```

When light is detected by the photoresistor the day indicator will illuminate, the supply fan will activate to 60% (arbitrary value), and the AHU control logic for heating and cooling will be enabled.

Control Sequence 2 – Mechanical Heating

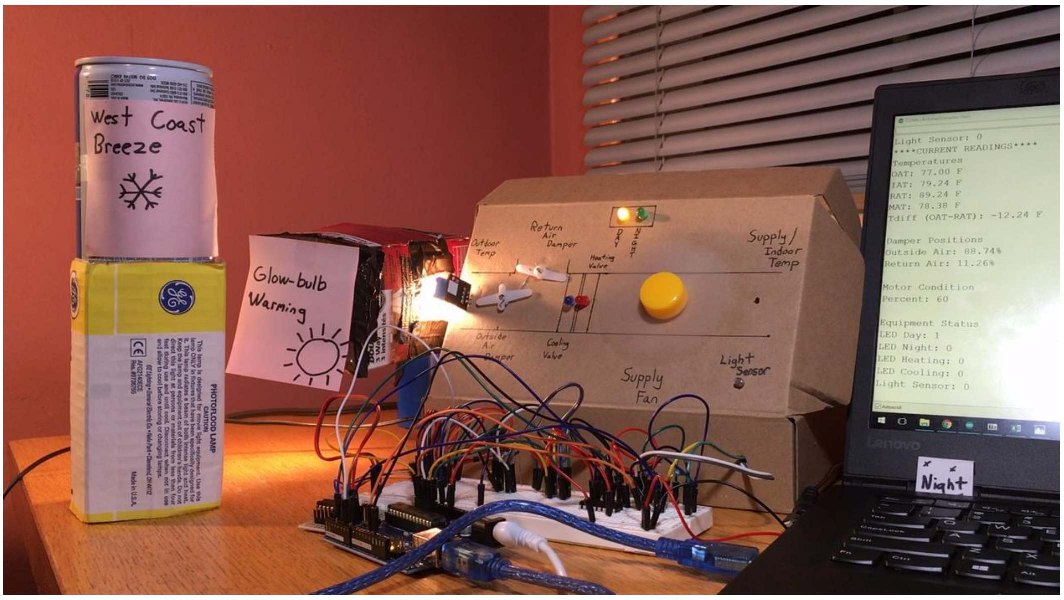
When Tdiff is < -40 then the OAD position = 20% and the heating valve is activated. The picture below simulates a cool 39F day from a "west coast breeze" (after using super cool air from the upside down Dust Off can) .



```
Light Sensor: 0
****CURRENT READINGS****
Temperatures
OAT: 39.20 F
IAT: 79.06 F
RAT: 89.06 F
MAT: 79.08 F
Tdiff (OAT-RAT): -49.86 F

Damper Positions
Outside Air: 20.00%
Return Air: 80.00%

Motor Condition
Percent: 60

Equipment Status
LED Day: 1
LED Night: 0
LED Heating: 1
LED Cooling: 0
Light Sensor: 0
```

Control Sequence 3 – Economizer Mode

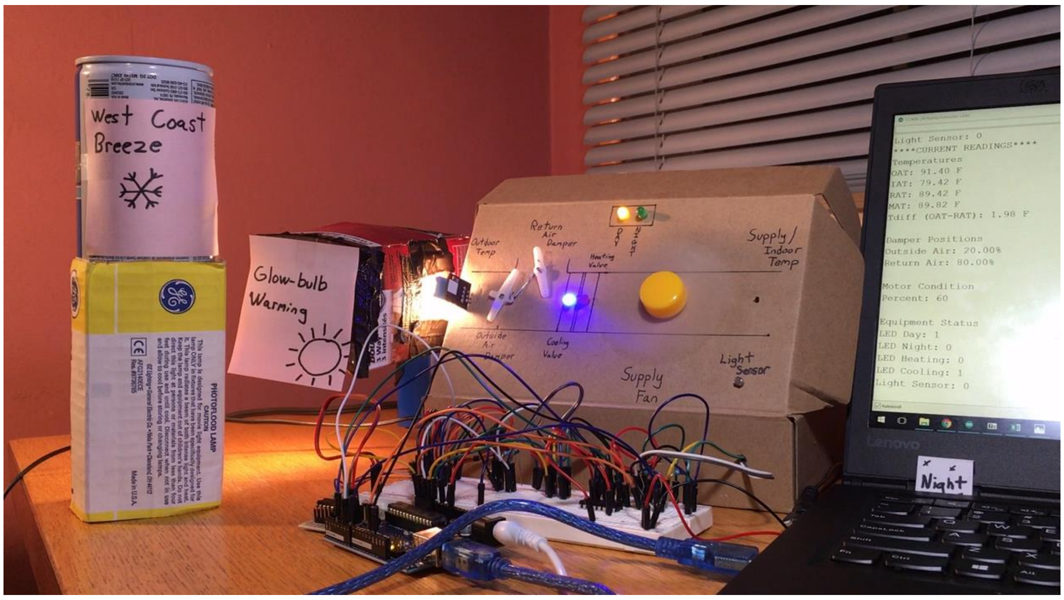
Economizer mode is enabled when Tdiff is between -40F and 0F. The OAD is an exponential function of Tdiff. The scenario below depicts a moderate 77F day with a "glow-bulb warming" sun that is heating up OAT.



```
Light Sensor: 0
****CURRENT READINGS****
Temperatures
OAT: 77.00 F
IAT: 79.24 F
RAT: 89.24 F
MAT: 78.38 F
Tdiff (OAT-RAT): -12.24 F

Damper Positions
Outside Air: 88.74%
Return Air: 11.26%

Motor Condition
Percent: 60

Equipment Status
LED Day: 1
LED Night: 0
LED Heating: 0
LED Cooling: 0
Light Sensor: 0
```

Control Sequence 4 – Mechanical Cooling

When the Tdiff is > 0 the OAD position = 20% and the heating valve is activated. In the last scenario the OAT reaches 91F and the cooling coil is activated.



```
Light Sensor: 0
****CURRENT READINGS****
Temperatures
OAT: 91.40 F
IAT: 79.42 F
RAT: 89.42 F
MAT: 89.82 F
Tdiff (OAT-RAT): 1.98 F

Damper Positions
Outside Air: 20.00%
Return Air: 80.00%

Motor Condition
Percent: 60

Equipment Status
LED Day: 1
LED Night: 0
LED Heating: 0
LED Cooling: 1
Light Sensor: 0
```

Test Video Procedure

The main idea was to vary the OAT to move the AHU through the several control sequences. The Test Video started by cooling the OAT sensor (not shown in the video) to 32F with the compressed air can then rapidly heat the OAT sensor with a 60-watt light bulb.

Lessons Learned

1. Voltage spikes cause faulty sensor readings. The OAT reading from the DHT11 temp & humidity module occasionally jumps by +/- 2 degrees. This usually happens when the servo is shuddering so I can only assume it is due to voltage fluctuations on the 5v rail. When I added a couple more grounding wires and plugged the power supply into the Arduino board OAT jumps decreased.

2. Stepper motors are tricky to integrate. I attempted to use a stepper motor for the RAD, but that became more complex when it was realized that the motor alone cannot remember its start position when powered down. (I kept the stepper motor driver in the system to add “wow factor” …and because I didn’t want to accidently bump any other wires loose trying to remove it!) To keep to scope, a linkage was added between the OAD and RAT, where the OAD is driven by the servo.
3. The Arduino IDE serial print is harsh on the eye. My next step is to make a simple GUI using Python to view all the sensor readings and status.

APPENDIX

Code

The C/C++ code was written using the Arduino IDE.

```
#include <Servo.h>
#include <dht.h>
dht DHT;
#define DHT11_PIN A5

//Pin Definitions
 int pinTimeDay=A1;
 int pinTimeNight=A0;
 int pinValveHeat=A2;
 int pinValveCool=A3;
 int pinMotor1=3;
 int pinMotor2=5;
 int pinThermistor = A4;
 int Vout=13;
 int pinServo = 6;
 int pinDayNight=7;

//Variable Definitions
 int timeDay=0;
 int timeNight=0;
 int valveHeat=0;
 int valveCool=0;
 int motorSpeed=0;
 int motorPercent=0;
 int Vo;
 float Toa;
 int Hoa;
 float R1 = 10000;
 float logR2, R2, Tia;
 float c1 = 1.009249522e-03, c2 = 2.378405444e-04, c3 = 2.019202697e-07;
 int pos = 0;
 float OAdampPos;
 float RAdampPos;
 Servo myOAdamper;
 int Tsp=70;
 float Tra;
 float Tdiff;


void setup(){
 Serial.begin(9600);
 pinMode(pinTimeDay,OUTPUT);
 pinMode(pinTimeNight,OUTPUT);
 pinMode(pinValveHeat,OUTPUT);
 pinMode(pinValveCool,OUTPUT);
 pinMode(pinMotor1,OUTPUT);
 pinMode(pinMotor2,OUTPUT);
 pinMode(Vout,OUTPUT);
 pinMode(pinDayNight,INPUT);
 myOAdamper.attach(pinServo);
}

void loop(){

//Day and Night System Schedule
 int DayNight = digitalRead(pinDayNight);
 if (DayNight == 0){
   digitalWrite(pinTimeNight,LOW);
   digitalWrite(pinTimeDay,HIGH);
 }
 else{
   motorPercent=0;
   OAdampPos = 0;
   digitalWrite(pinTimeNight,HIGH);
   digitalWrite(pinTimeDay,LOW);
```

```
    digitalWrite(pinValveCool,LOW);
    digitalWrite(pinValveHeat,LOW);
    Tra = Tia+10;   //adding 10F to simulate return air temp in summer
    Tdiff = Toa - Tra;
   }

//INDOOR AIR TEMPERATURE
  Vo = analogRead(pinThermistor);
  R2 = R1 * (1023.0 / (float)Vo - 1.0);
  logR2 = log(R2);
  Tia = (1.0 / (c1 + c2*logR2 + c3*logR2*logR2*logR2));
  Tia = Tia - 273.15;
  Tia = (Tia * 9.0)/ 5.0 + 32.0;

//OUTDOOR AIR TEMPERATURE
  int chk = DHT.read11(DHT11_PIN);
  Toa=DHT.temperature*1.8+32; //temp in F
  Hoa=DHT.humidity;


//CONTROL LOGIC
  if (DayNight == 0){

    motorPercent=60;
    Tra = Tia+10;   //adding 10F to simulate return air temp in summer
    Tdiff = Toa - Tra;

    if (-10 <= Tdiff && Tdiff < 0){
      OAdampPos = 100;
    }

    if (-40 <= Tdiff && Tdiff < -10) {
      OAdampPos = 100*1.71*(exp(0.0536*Tdiff));
    }

    if (Tdiff < -40) {
      OAdampPos = 20;
      digitalWrite(pinValveHeat,HIGH);
    }
      else{
        digitalWrite(pinValveHeat,LOW);
      }

    if (0 <= Tdiff) {
      OAdampPos = 20;
      digitalWrite(pinValveCool,HIGH);
    }
      else{
        digitalWrite(pinValveCool,LOW);
      }
  }

//SUPPLY FAN MOTOR CONTROL

  motorSpeed = motorPercent*(255./100.);  //converstion for 0-255 to 0-100
  analogWrite(Vout,motorSpeed);
  digitalWrite(pinMotor1,LOW);  //change Low/high outputs to change direction of fan
  digitalWrite(pinMotor2,HIGH);

//OUTSIDE AIR DAMPER
  /*Test: Happy range is 70 to 180 degrees
  Serial.println("Servo position?");
  while (Serial.available()==0) {}
  pos = Serial.parseInt();
  myPointer.write(pos);*/

  pos=-(110./100.)*OAdampPos+180.; //conversion for 70-180 deg to 0-100%
  myOAdamper.write(pos);

//RETURN AIR DAMPER
```

```
RAdampPos= 100 - OAdampPos;

//PRINT STATEMENTS
 Serial.println("****CURRENT READINGS****");
 Serial.println("Temperatures");
 Serial.print("OAT: ");
 Serial.print(Toa);
 Serial.println(" F");

 Serial.print("IAT: ");
 Serial.print(Tia);
 Serial.println(" F");

 Serial.print("RAT: ");
 Serial.print(Tra);
 Serial.println(" F");

 Serial.print("MAT: ");
 float MAT = Tra - (OAdampPos/100)*(Tra-Toa); //From OA%=(RAT-MAT)/(RAT-OAT)
 Serial.print(MAT);
 Serial.println(" F");

 Serial.print("Tdiff (OAT-RAT): ");
 Serial.print(Tdiff);
 Serial.println(" F");
 Serial.println(" ");

 Serial.println("Damper Positions");
 Serial.print("Outside Air: ");
 Serial.print(OAdampPos);
 Serial.println("%");

 Serial.print("Return Air: ");
 Serial.print(RAdampPos);
 Serial.println("%");
 Serial.println(" ");

 Serial.println("Motor Condition");
 Serial.print("Percent: ");
 Serial.println(motorPercent);
 Serial.println(" ");

 Serial.println("Equipment Status");
 Serial.print("LED Day: ");
 int PTD = digitalRead(pinTimeDay);
 Serial.print(PTD);
 Serial.println(" ");

 Serial.print("LED Night: ");
 int PTN = digitalRead(pinTimeNight);
 Serial.print(PTN);
 Serial.println(" ");

 Serial.print("LED Heating: ");
 int PVH = digitalRead(pinValveHeat);
 Serial.print(PVH);
 Serial.println(" ");

 Serial.print("LED Cooling: ");
 int PVC = digitalRead(pinValveCool);
 Serial.print(PVC);
 Serial.println(" ");

 Serial.print("Light Sensor: ");
 int PDN = digitalRead(pinDayNight);
 Serial.print(PDN);
 Serial.println(" ");

 delay(3000);

}
```