Chubster's Simple Heating System

Kevin Antonino *December 8, 2020*

1 Problem Description

Hamster Chubster The Second spends most of his time in a cage located in a washing room. During Winter months this room is usually around $60 - 65\ F$ which is in the lower bound of a Hamster's desired temperature which is $60 - 80\ F$. The goal of this project is to create a heating device that can raise the ambient temperature to a nominal $T_{hamster} = 72\ F$ for maximum hamster comfort. The mechanism for raising the temperature will be by pumping warm air into the cage at low speeds. This method is desirable over direct contact heating since errors in temperature can result in burns or the ignition of the shredded paper flooring. Therefore the device will be a fan-driven heating duct. The heat will come from a resistive circuit so that it can be modulated easily. The following sections will determine: the required heat dissipation to raise the temperature in steady laminar airflow, analysis of an electric circuit to dissipate this heat, the temperature control scheme, figures of prototype and test runs, and diagrams.

2 Required heat dissipation $\dot{Q}_{in}$

Assumptions for analysis for the simple heating duct:

1. The ambient air behaves as a simple, incompressible fluid following $PV = \rho RT$
2. The specific heat capacity at constant pressure C_p is constant for $T = 60 - 80\ F$
3. The water vapor in the air also behaves as an ideal gas
4. The water vapor negligibly effects the enthalpy of air

Therefore in the air duct control volume,

$$\dot{E}_{in} = \dot{E}_{out} \implies \dot{Q}_{in} + \dot{W}_{in} + \sum_{in} \dot{m}h = \dot{Q}_{out} + \dot{W}_{out} + \sum_{out} \dot{m}h$$

Ignoring work done by the fan, heat coming out of the duct, and any work done on the air:

$$\dot{Q}_{in} \sum_{in} \dot{m}h = \sum_{out} \dot{m}h$$

Solving for the heat rate and noticing the mass conservation in the duct from one entrance-exit:

$$\dot{Q}_{in} = \dot{m}(h_{out} - h_{in})$$

Using $\dot{m} = V_{air} * A_{duct} * \rho_{air}$ and for an ideal gas with constant C_p, $h = h(T) = C_p T$

$$\dot{Q}_{in} = [V_{air} A_{duct} \rho_{air}] * C_p (T_{out} - T_{in})$$

After some trial and error, an optimal duct size of 5 x 5 centimeters and air moving at 0.5 m/s will be used. ρ_{air} is calculated using the ideal gas relation with $P = 101.325$ kPa, $R = 0.2870 \frac{kJ}{kg * K}$, and $T = 298$ K $\implies \rho_{air} = 1.1847$ kg/m^3 The C_p used is 1.004 kJ/K For an understanding of the maximum heat dissipation required $T_{out} = T_{hamster} = 295.375$ K and T_{in} at absolute minimum condition is 288.706 K This results in:

$$\dot{Q}_{in} = 9.911 \approx 10\ W$$

3 Heating circuit

Keeping in mind the maximum 10 Watt demand, a circuit must be designed to dissipate this energy. A few assumptions will be required:

1. All the power consumed by a resistor will be translated to heat into the duct
2. The resistors used are Ohmic for all temperatures

Resistors will be used to provide this heat. This is because at the time of this project specialized resistive wires were not accessible within the time constraint. The orientation of the resistors will be in parallel to take advantage of the constant voltage from the source. The greatest wattage rating for the available resistors is 1 Watt per resistor. Since this is not recommended for continuous operation 0.5 Watts per resistor will be used. Since the power dissipated by the circuit is just the sum of each element, 20 resistors will be required. There resistance is:

$$P = IV = \frac{V^2}{R_{res}} \implies R_{res} = \frac{V^2}{P}$$

However using the assumption that $P_{res} = \dot{Q}_{res}$

$$R_{res} = \frac{V^2}{\dot{Q}}$$

A 12V 2A (max) power source will be used as it was readily available. Using this constraint along with the power requirement of 1/2 Watts:

$$R_{res} = 288\Omega$$

This will round up to 300Ω since there are no resistors at this exact resistance.

4 Heating circuit control

The heating circuit will be controlled by the Adafruit ESP32 Feather microcontroller from the Berkeley Mictrokit. The microcontroller will take a reference $T_{hamster}$ as well as measured temperature by the Velleman VMA339 temperature probe and sensor. This will be turned into a voltage from the model derived above and fed into the heating system.

$$V = \sqrt{R_{circuit}\dot{m}C_p(T_{ref} - T(t))}$$

When the ESP32 is turned on, the required voltage is calculated based on one measurement of the ambient air temperature. While the system is running the output temperature is measured at 5 second intervals to allow the circuit to be shut off in the event of over-heating, long before the housing can burn.

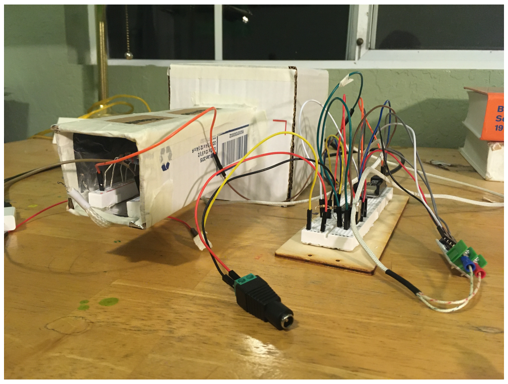
The voltage will be modulated by the STP16NF06L MOSFET from STMicroelectronics. One of the ESP32's PWM outputs is tuned to change the input voltage from the source by the use of this MOSFET. This component is able to switch 60V and 16A which is plenty for this application. In the gate circuit a 220Ω resistor will be used in series with the PWM output pin as a pull-down resistor so current is limited.

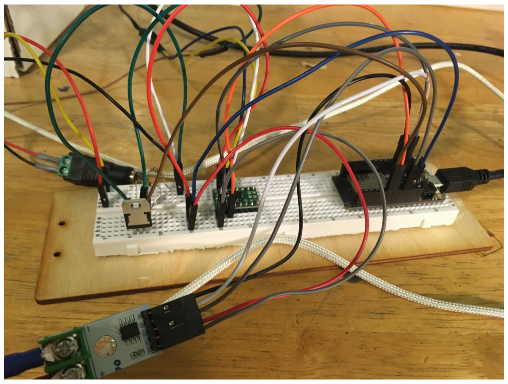
The Velleman temperature probe and sensor requires 3 GPIO pins on the ESP32 as well as a 3-5 volt source. The source will be the 3.3V USB output of the ESP32 microcontroller. The appropriate Velleman firmware library is loaded in the Arduino IDE sketch in the appendix. The microchip in the temperature sensor does not require a current-limiting resistor.

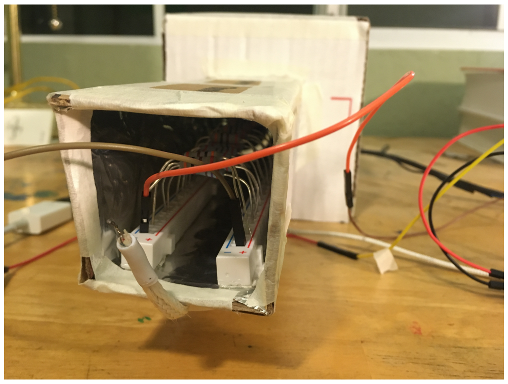
4.1 Fan control

For prototyping the fan will be an unused computer fan powered by the 3.3V USB output of the ESP32 microcontroller. It can later be designed to be powered by the Li-Ion battery from the Berkeley Microkit or from the 12 volt source. The fan will be modulated by the DRV8833 motor driver so that the fan voltage can be tuned to 0.5 m/s by one of the microcontroller's PWM digital outputs. Additionally the fan can be turned off by the ESP32 in the event of high temperatures as to not fuel potential fires. The fan's RGB pins are simply connected by jumper cables to apply the voltage.

5 Prototype and test runs


(a) Prototype of duct and circuit


(b) Close-up of circuit. The white PCB is the temperature sensor chip


(c) Close-up of duct with temperature probe and resistors

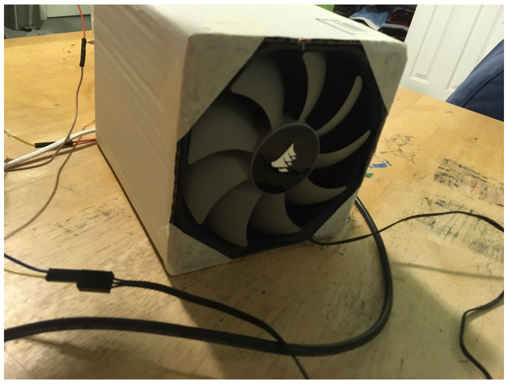
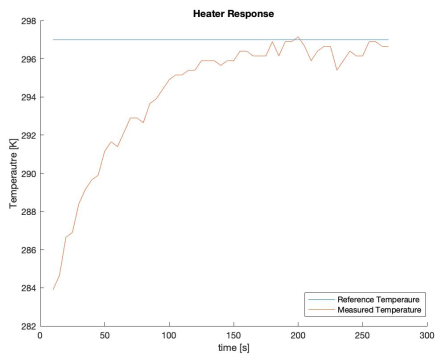
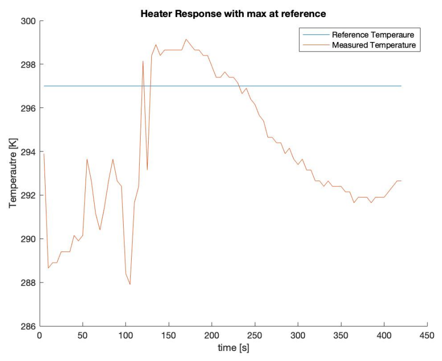

(d) Repurposed computer fan

Figure 1: Built Prototype

Below is a figure of the heating response of the system. While testing the temperature indoors was around 71 F so the heater was placed outside at ≈ 50 F, and the reference at 75 F or 297 K. The plot on the left shows that the heater can safely heat a volume of air at indoor extremes. The plot on the right studies the effect of the safety system by placing the max temperature allowable to 297 K. The right plot shows that if T_{max} is exceeded the inside temperature will drop.



(a) Figure of duct temperature over time while operating at extreme temperatures.



(b) Figure of safety system in action over the same range of temperatures

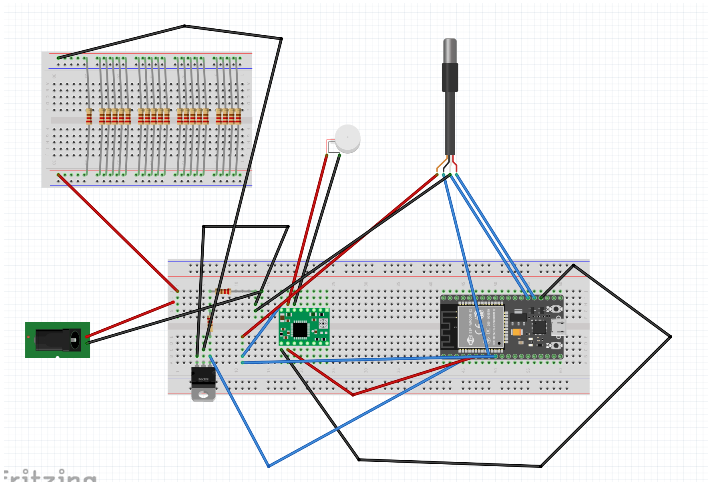
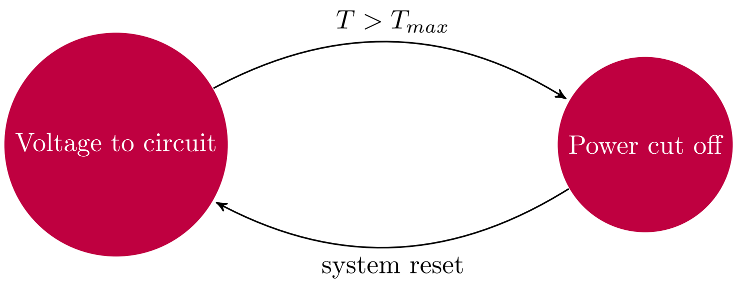


Figure 3: Circuit diagram. The grey cylinder is in place of the fan. The barrel on the left is the 12V source. Black wires go to ground, red wires are high potential. The blue wires are PWM or sensing pins.

The FSD is simple since there are only two states: one when voltage is applied to the resistors, the fan is running, and measurements are being taken. The other state is the "emergency state" where there is no voltage to the circuit and the fan is not spinning. The two states are illustrated in the MATLAB plots of temperature vs. time above.



A Arduino IDE code

```
/* This program will spin the DC motor fan based on PWM signal,
 * Modululate the power drawn to the heating circuit,
 * Read and put use feedback data from the temperature sensor
 */

#include <MAX6675.h> // Temp sensor library
#include <ESP32Encoder.h>
ESP32Encoder encoder;

//Inputs to the H-Bridge
const int bin1 = 27; //GPIO 27

//Modulation for heating Circuit
const int mod = 33; //GPIO 33

//Temperature Sensor
const int CS = 26;             // CS pin on MAX6675 GPIOA0
const int SO = 25;              // SO pin of MAX6675 GPIOA1
const int SCKK = 13;             // SCK pin of MAX6675
const int units = 1;            // Units to readout temp (0 = raw, 1 = C, 2 = F)
MAX6675 temp(CS,SO,SCKK,units); //Initialize library to sensor chip

// PWM
int fan_PWM = 250; // PWM for 0.5 m/s air flow
int mod_PWM = 0;

//Model Constants
const float R = 14.4; //Ohms
const float massFlux = 0.0014809; // kg/s
const float Cp = 1005; //J/(Kg*K)
const float Tref = 295; //Kelvin

//Model Variables
float V = 0;  //Voltage
float sensorTemp = 0.0;  // Temperature output variable
float T = 0; //Temperature for equation
const float maxTemp = 320; // Temp to shut down system
bool safetyState = 0;
int i = 1;

//Timer Variables
unsigned long total_time;
unsigned long passed_time;
int interval = 5000;

void setup() {
  Serial.begin(115200);

  //setup for PWM
  ledcAttachPin(bin1, 1);
  ledcSetup(1, 20000, 8);
  ledcAttachPin(mod, 2);
  ledcSetup(2, 20000, 8);
  ledcWrite(1, fan_PWM);
  ledcWrite(2,0);

  T = temp.read_temp() + 273.15; //Temperature of ambeint air in Kelvin
  V = sqrt(R*massFlux*Cp*(Tref-T)); //calculate voltage required
  V = ( V*(V<=12) + 12*(V>12) ); // does not draw > 12 V
  mod_PWM = map(V,0,12,0,243); //refined conversion factor

  total_time = millis();
}

void loop() {
  passed_time = millis();

  //measure T every 20 sec
  //check if on fire, if so run ISR

  if ((unsigned long) (passed_time - total_time ) >= interval ) {

  sensorTemp = temp.read_temp() + 273.15;

    if(safetyState == 0){
      safetyState = (sensorTemp > maxTemp); //check if its too hot

      Serial.print(i*interval/1000); //MATLAB formatted
      Serial.print(",");
      Serial.print(sensorTemp);
      Serial.print(",");
      Serial.print(Tref);
      Serial.print(",");
      Serial.print(V);
      Serial.print(",");
      Serial.print(mod_PWM);
      Serial.println(";");
      ledcWrite(2,mod_PWM); //Voltage to MOSFET

      i++;
      total_time = passed_time; // "reset" timer

    }else{

      ledcWrite(2,0); // MOSFET prevents voltage to heating circuit
      ledcWrite(1, 0); // Fan turns off as to not fuel any potential fires
      Serial.println("Emergency Shutoff");
      //to start again the ESP32 has to be manually reset
    }
  }

}
```