Final Project: Room Temperature Regulator

ECE 4220

Author: Jason Blickhan

Instructor: Guilherme DeSouza

May 2014

**Abstract**

The purpose of this project is to keep a room kept at a desired temperature without having to manually turn on and off a space heater. The project will implement a Raspberry Pi reading the current room temperature through a temperature sensor and an rf transmitter connected to a GPIO pin used to turn on a remote control outlet the space heater will be plugged into. The program will check the temperature and turn on the space heater as needed to keep the room at a desirable temperature.

**Introduction**

As I sat in my cold basement bedroom at my computer trying to think of a practical final project, I wiggled my fingers and did some arm circles to get the blood to my fingers considering they are always freezing, and it is so difficult typing with cold hands. Then it hit me! A final project that could benefit me for years to come and utilize concepts learned in class would be one that heats my room when I am just sitting around my room and don’t feel like messing with the thermostat. I already had a Raspberry Pi sitting around making a great paper weight that I hadn’t looked too far into, so this would be a great opportunity for me to force myself to learn how to use it.

**Problem Statement**

Before getting started, there are a few problems that need to be addressed. The first is to learn the Raspberry Pi and understand its GPIO pins along with what hardware to use with it and how to program it. I have never before used Python and decided this is as good a chance as ever to start learning it. Luckily Python is very easy to learn.

I looked around at hardware and found that a DS18B20 temperature sensor has been used with Raspberry Pi before and would be very compatible with what I am trying to do. I will also need a remote control outlet that will allow me to turn on and off the power to the space heater plugged into it. An rf receiver will be needed to figure out the signal being sent by the remote control and I will need to turn around and send that signal with an rf transmitter. The remote control outlets use a frequency of 315MHz so an rf receiver and transmitter pair that use the same frequency will be needed which can be cheaply found on Amazon.

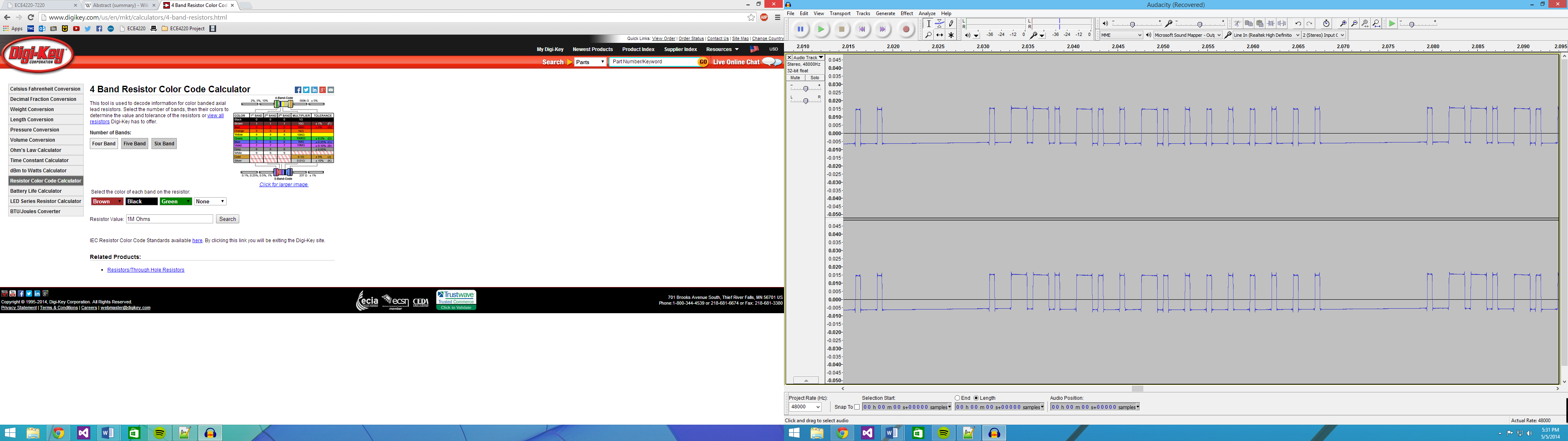
On the software side, the program will need to parse through the data the temperature sensor is reading in and determine if it is running. Then the correct value should be taken and converted into Fahrenheit. The program will need a separate thread checking the temperature constantly and sending the on and off signal accordingly through the rf transmitter. The signal will need to be sent to the transmitter via a GPIO pin on the Raspberry Pi.

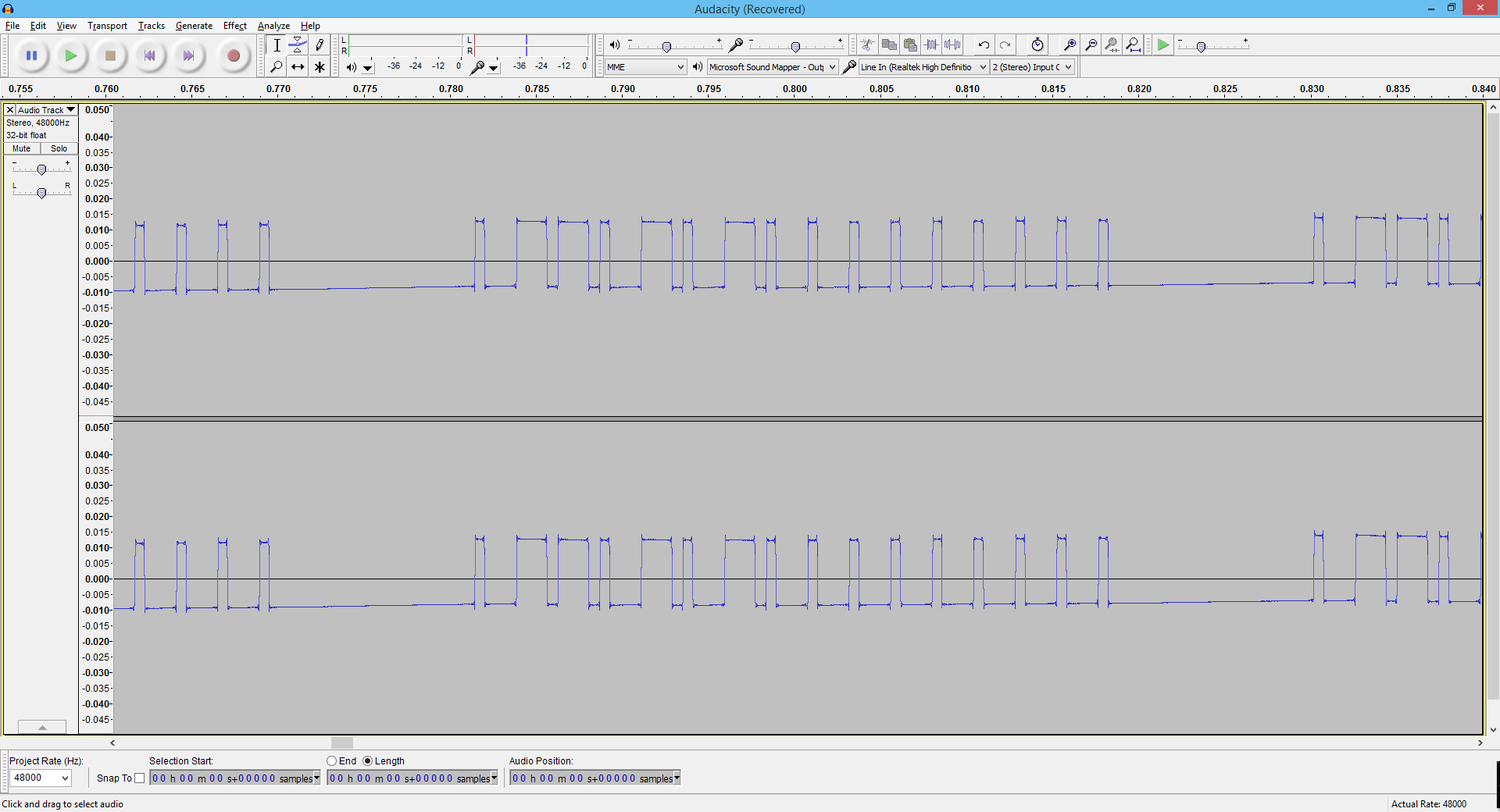
**Hardware Flowchart**

**Proposed Approach**

Seeing as there was a lot of online support and resources for the Raspberry Pi, I quickly learned to control the output voltages of the GPIO pins. I found that the DS18B20 would be the perfect temperature sensor to use with the Raspberry Pi since it could use the 5V power supply, ground, and pin 7 from the GPIO pins.There was also already Python code written to parse the data being read in from the sensor that I could incorporate into my program. This code goes to the file that the sensor is writing to and splits it into lines broken up by a newline character. It then checks for the letter ‘Y’ three characters from the end in those lines until it finds it. The line that contains this will also have the temperature after a “t=” which it looks for next. It finally strips the value and converts it to Celsius and Fahrenheit since it comes in as Celsius \* 1000.

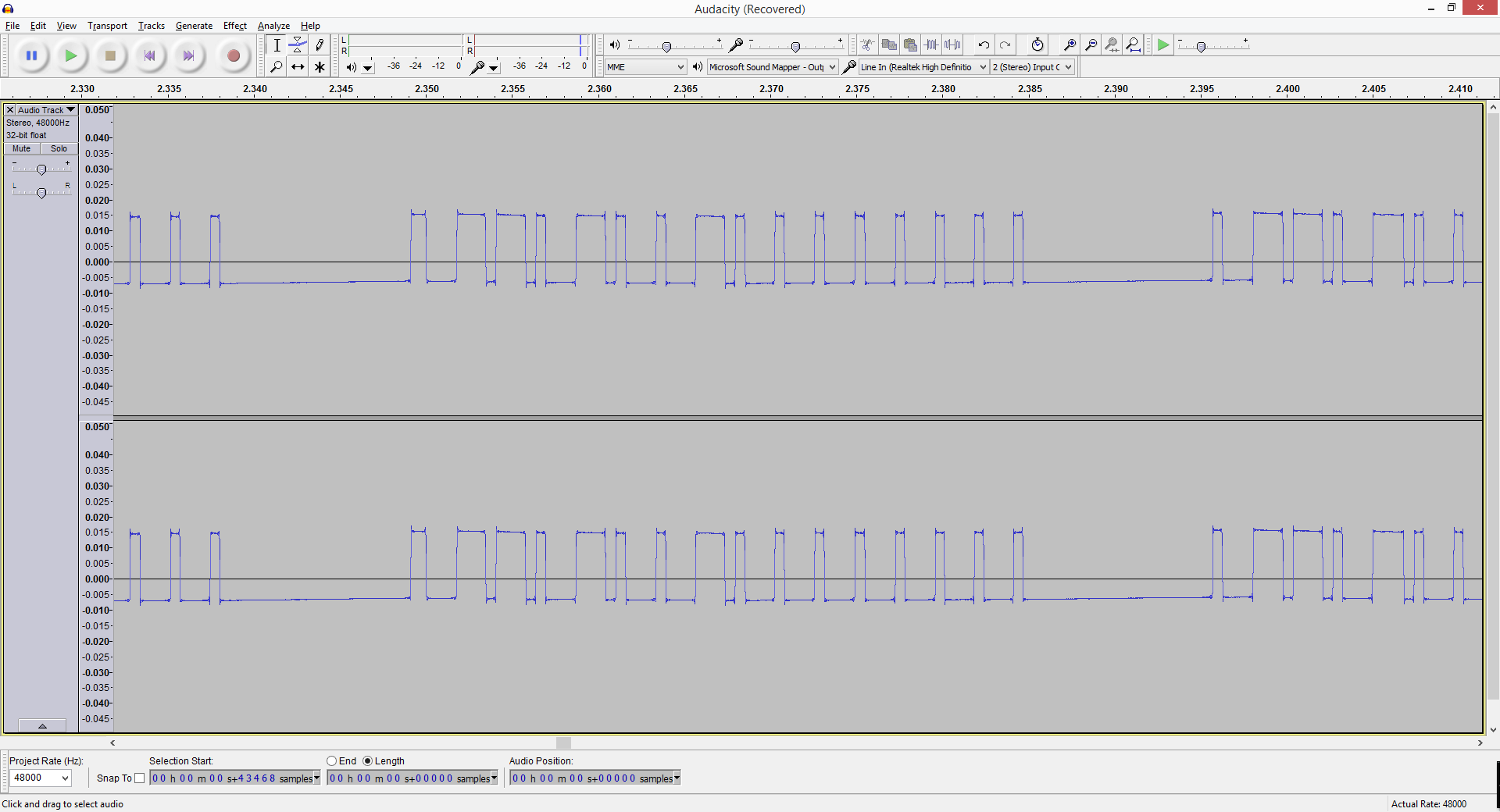
One of the tougher problems that needed to be solved was how to transmit a signal that would turn on a remote control outlet across the room. I found the solution with a 315 MHz rf transmitter and receiver. The receiver is only needed temporarily to obtain the correct signal from the remote control. This was done by connecting the 5V power supply and ground from the Raspberry Pi to the rf receiver. The transmitter and receiver each needed a 23cm antenna as should be used with 315 MHz. Then an audio cable was connected to ground and then to the data pin on the receiver through a 1M Ohm resistor in series while the other end was plugged into the line in jack on my desktop. The audio software Audacity was then used to record the signal sent each time the button to turn on the outlet on the remote control was pressed. I converted this signal to 0s and 1s corresponding to highs and lows so that it could be used in the program: 1000 1110 1110 1000 1110 1000 1000 1110 1000 1000 1000 1000 1000 1000 1000 1000 0000 0000 0000 0000. I then did the same for signal to turn off the remote control outlet: 1000 1110 1110 1000 1110 1000 1110 1000 1000 1000 1000 1000 1000 1000 1000 1000 0000 0000 0000 0000.

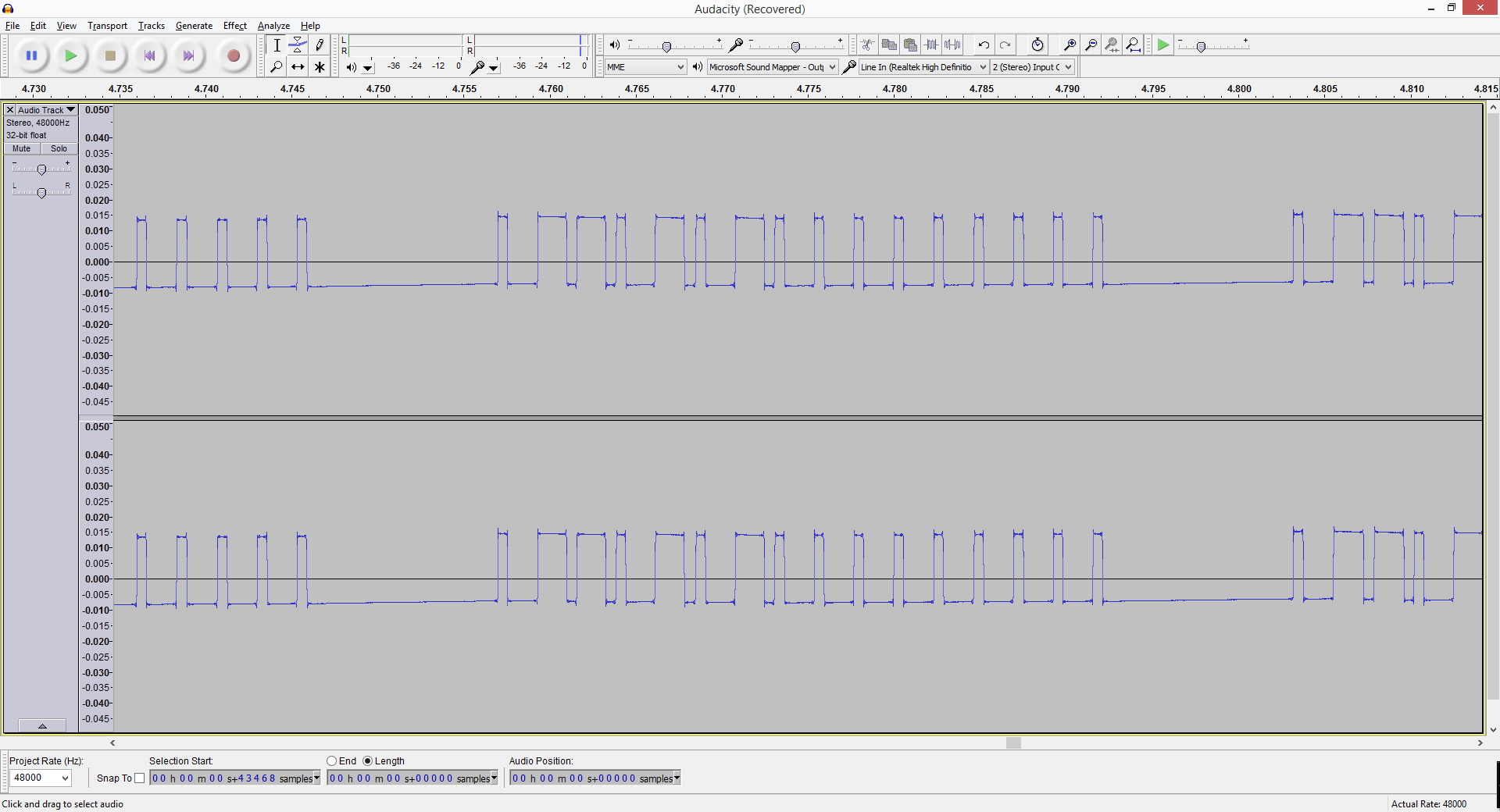


**On signal sent by Remote Control**

**Off signal sent by Remote Control**

Then in the program, I created a switchon() function that would go through those strings of 1s and 0s and send a voltage to the data pin of the transmitter on a 1 and turn the voltage off on a 0 and then sleeping for a very short amount of time. The amount of sleep between each bit checked came out to be 440us which I got by guessing around 220us and then comparing the transmitter signal to the remote control’s signal. It took a few tries until the value of sleep finally made the signals match pretty similarly. As soon as I got close to the right amount of sleep needed, the remote control outlet plugged into the wall turned on. I then copied this algorithm for the off signal and made a switchoff() function.

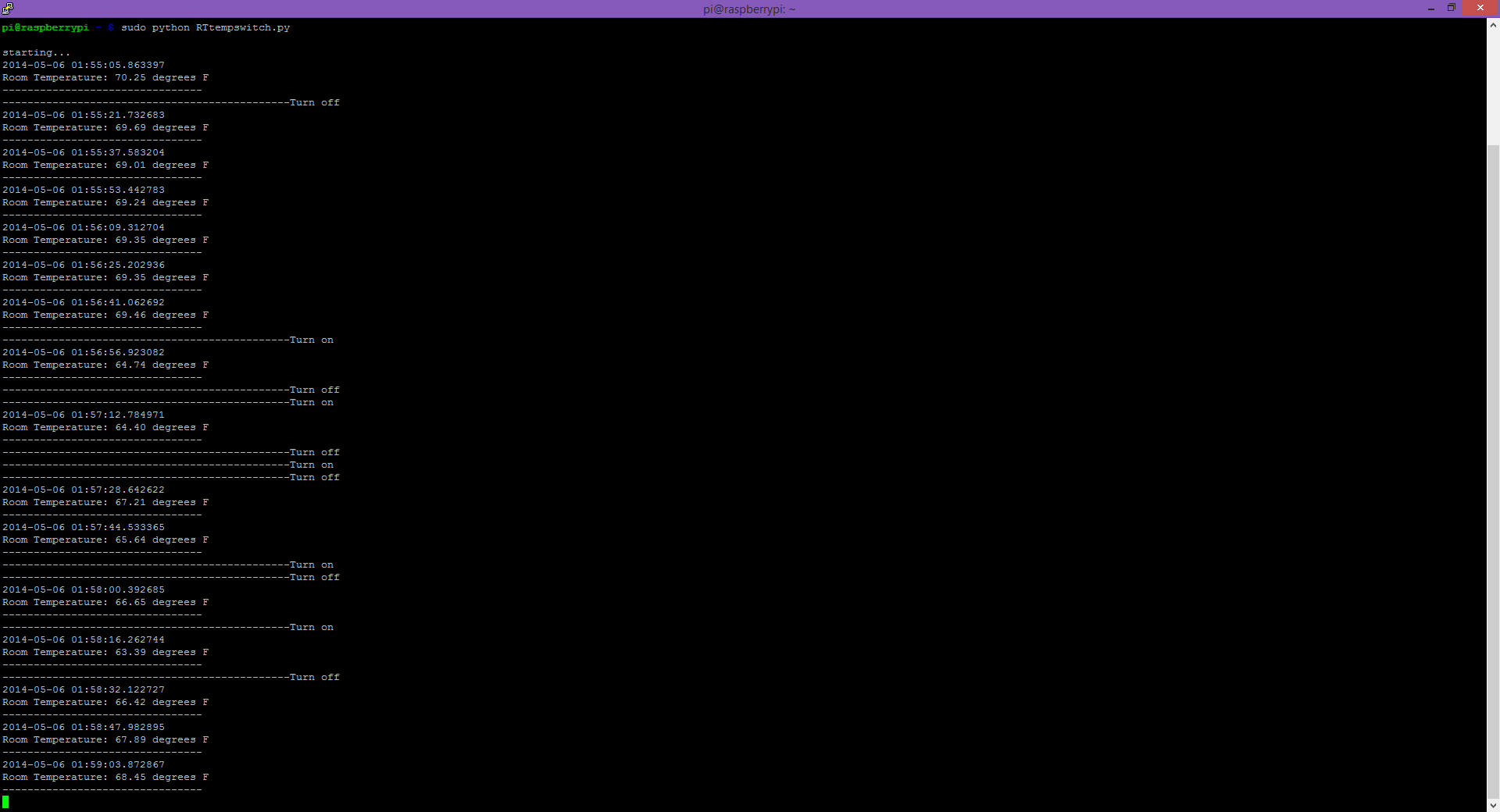
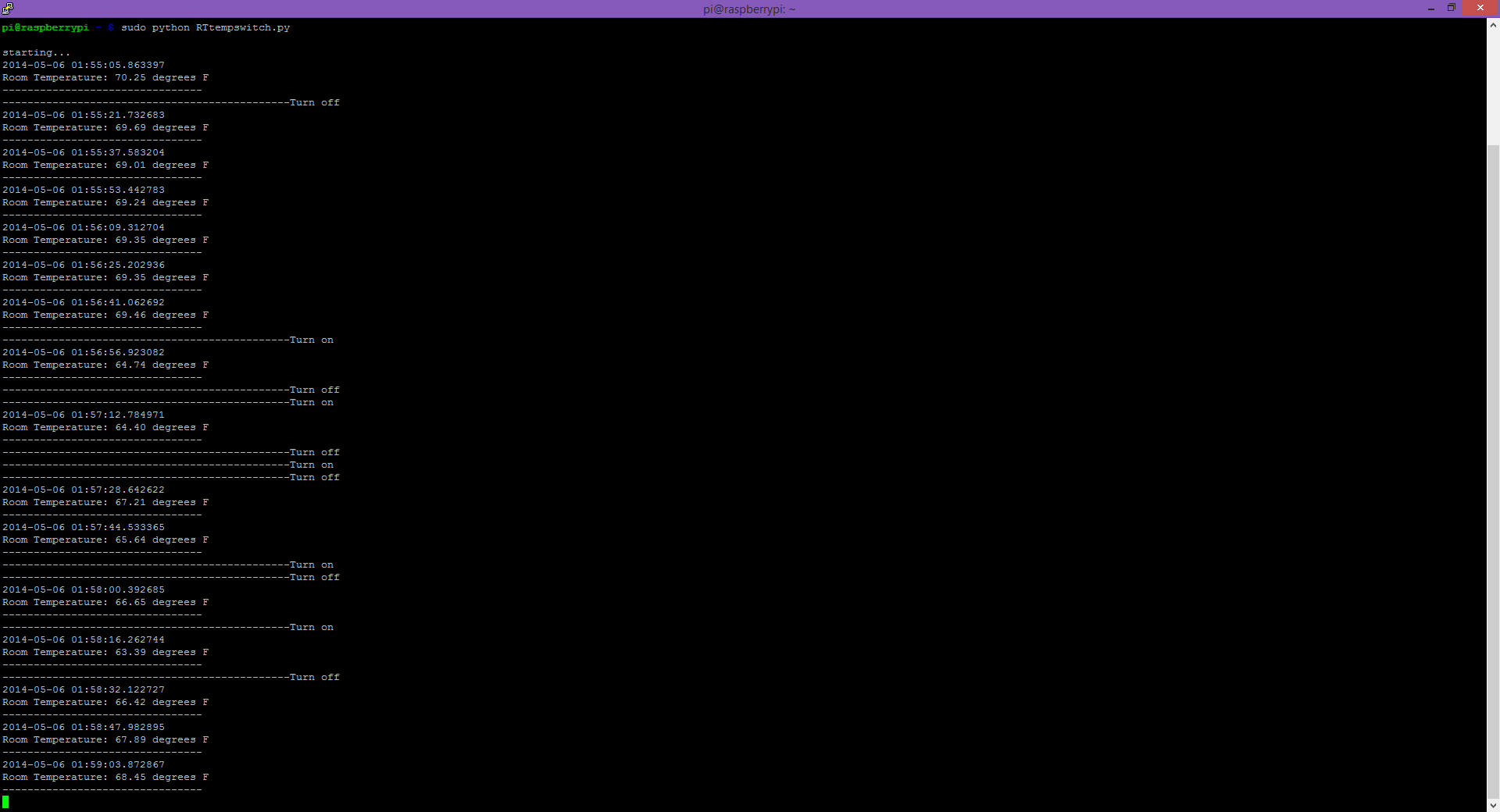
**On Signal sent by RF Transmitter**

**Off Signal sent by RF Transmitter**

Finally in the main function, a thread is created that starts running checktemp(). Checktemp() is a function that checks the current temperature and depending on if it is low or high will turn the remote control outlet on or off respectively by calling switchon() or switchoff(). If the temperature is below the desired minimum temperature and the variable that tells if the outlet is on or off says it is off, the outlet will be turned on. On the other hand if the temperature is at the desired temperature and the variable says the outlet is on, the outlet will be turned off. While this thread is running, there is an infinite while loop in main() that is displaying the current time and temperature every 15 seconds.

**Results**

The results of this project are successful in that it did exactly what I had hoped for keeping my room warmer. The checktemp() thread successfully gathers the current temperature from my bedroom and compares it to the desired temperature. It then sends a signal to turn on the remote control outlet that the spaceheater is plugged into and begins heating my room. When the desired temperature is reached, the thread sends the signal to turn off the outlet. While this is happening, the current temperature is being displayed to the screen every 15 seconds.



**Screenshot of program running while using cup of cold water and hand warmth to manually cool down and heat up temperature sensor to test checktemp() thread**

**Conclusion**

Ultimately, experience acquired through labs gave me the skills to complete this task. I was able to assemble hardware to an embedded device and configure it to carry out the job of keeping my room heated to a desirable temperature. By reading the current temperature using a DS18B20 temperature sensor and using a thread to constantly compare it to the desired temperature, I was able to make sure the space heater in my room would get turned on and stay on until it reached the desired temperature. This project was pretty fun in that it forced me to work with Raspberry Pi and interact with other hardware components that I wanted to get experience with, but hadn’t had time to do so. I learned a lot about working with Python and working with threads in Python. I also learned about Raspberry Pi and the GPIO pins and how to control hardware using them. This will be useful for future projects I want to undertake to do simple tasks around the house or for bigger tasks in my future classes and after graduation. Working with embedded systems has shown me that the possibilities of what I can accomplish are truly endless and how much fun it can be to see a project through and accomplish your goal.

**Bibliography**

"Adafruit's Raspberry Pi Lesson 11. DS18B20 Temperature Sensing." *Software*. N.p., n.d. Web. 20 April 2014.

**Code**

import thread

import os

import glob

import time

import datetime

import subprocess

from RPi import GPIO

from time import sleep

GPIO.setmode(GPIO.BOARD)

GPIO.setup(11, GPIO.OUT)

GPIO.setwarnings(False)

os.system('modprobe w1-gpio')

os.system('modprobe w1-therm')

base\_dir = '/sys/bus/w1/devices/'

device\_folder = glob.glob(base\_dir + '28\*')[0]

device\_file = device\_folder + '/w1\_slave'

oncode = "10001110111010001110100010001110100010001000100010001000100010000000000000000000"#corresponds to the signal that triggers on the rc outlet

offcode = "10001110111010001110100011101000100010001000100010001000100010000000000000000000"#corresponds to the signal that triggers off the rc outlet

slplen = 0.00044

def checktemp(lowtemp,hightemp):

onoff = 0

degC, degF = read\_temp()

if degF < lowtemp:

onoff = 0 #says heater is off if temp is low

if degF > hightemp:

onoff = 1 #says heater is on if temp is high

while 1:

degC, degF = read\_temp()

if degF < lowtemp and onoff == 0:#if temp is low and heater is off

print "----------------------------------------------Turn on"

switchon()

onoff = 1 #heater is on

if degF > hightemp and onoff == 1:#if temp is high and heater is on

print "----------------------------------------------Turn off"

switchoff()

onoff = 0 #heater is off

def switchon():

for i in range (0,9):#send signal 10 times

for j in range(0 ,len(oncode)):#send signal through rf transmitter

if(oncode[j] == '1'):

GPIO.output(11,True)

elif(oncode[j] == '0'):

GPIO.output(11,False)

sleep(slplen)#used to set correct length of signal

def switchoff():

for i in range (0,9):#send signal 10 times

for j in range(0 ,len(offcode)):#send signal through rf transmitter

if(offcode[j] == '1'):

GPIO.output(11,True)

elif(offcode[j] == '0'):

GPIO.output(11,False)

sleep(slplen)#used to set correct length of signal

def read\_temp\_raw():

catdata = subprocess.Popen(['cat',device\_file], stdout=subprocess.PIPE, stderr=subprocess.PIPE)#opens a pipe to read from the standard stream

out,err = catdata.communicate()

out\_decode = out.decode('utf-8')

lines = out\_decode.split('\n')#splits lines at newline character

return lines

def read\_temp():

lines = read\_temp\_raw() #lines from terminal

while lines[0].strip()[-3] != 'Y': #if the letter 3 from the end isn't Y(ES)

time.sleep(0.2)

lines = read\_temp\_raw()

equals\_pos = lines[1].find('t=') #find t= in string

if equals\_pos != -1:

temp\_string = lines[1][equals\_pos+2:]

temp\_c = float(temp\_string) / 1000.0 #converts string to float

temp\_f = temp\_c \* 9.0 / 5.0 + 32.0 #converts C to F

return temp\_c, temp\_f

def main():

print "\nstarting..."

thread.start\_new\_thread(checktemp, (65,66,))#args are (temp to turn on,temp to turn off,)

while 1:

degC,degF = read\_temp() #update temp

print str(datetime.datetime.now()) #show time

print"Room Temperature: %.2f degrees F" % degF

print "--------------------------------"

time.sleep(15)

main()