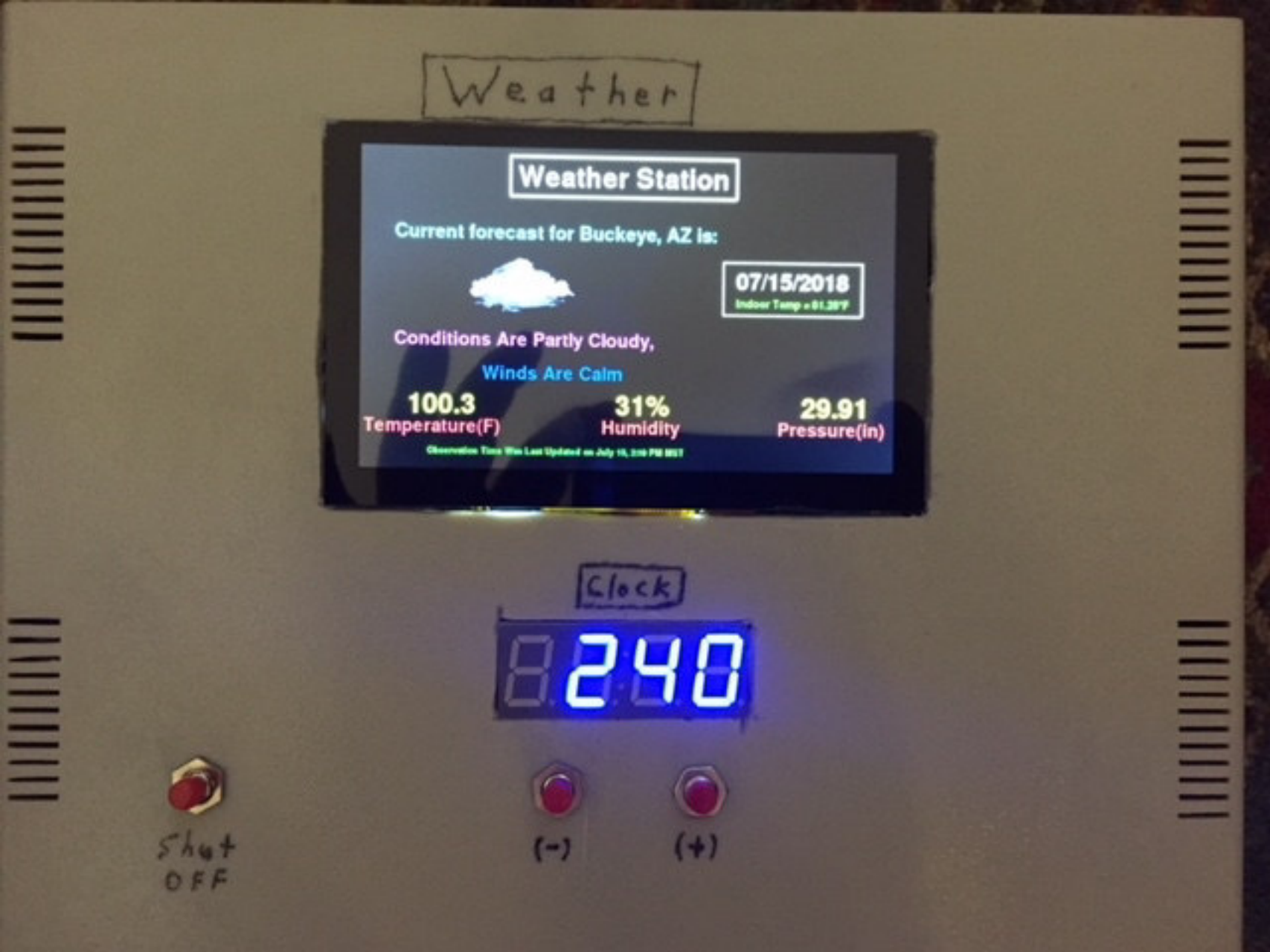
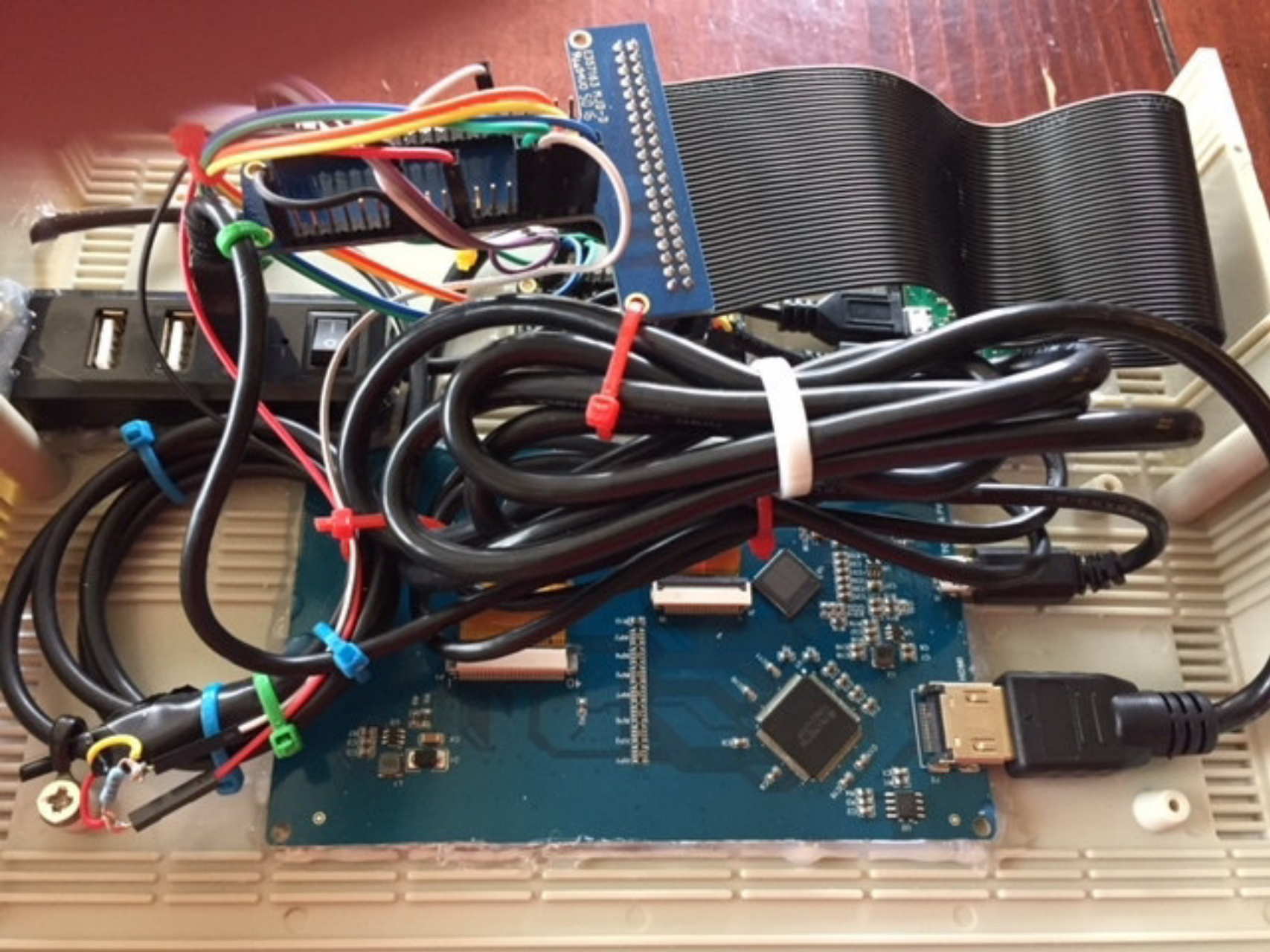
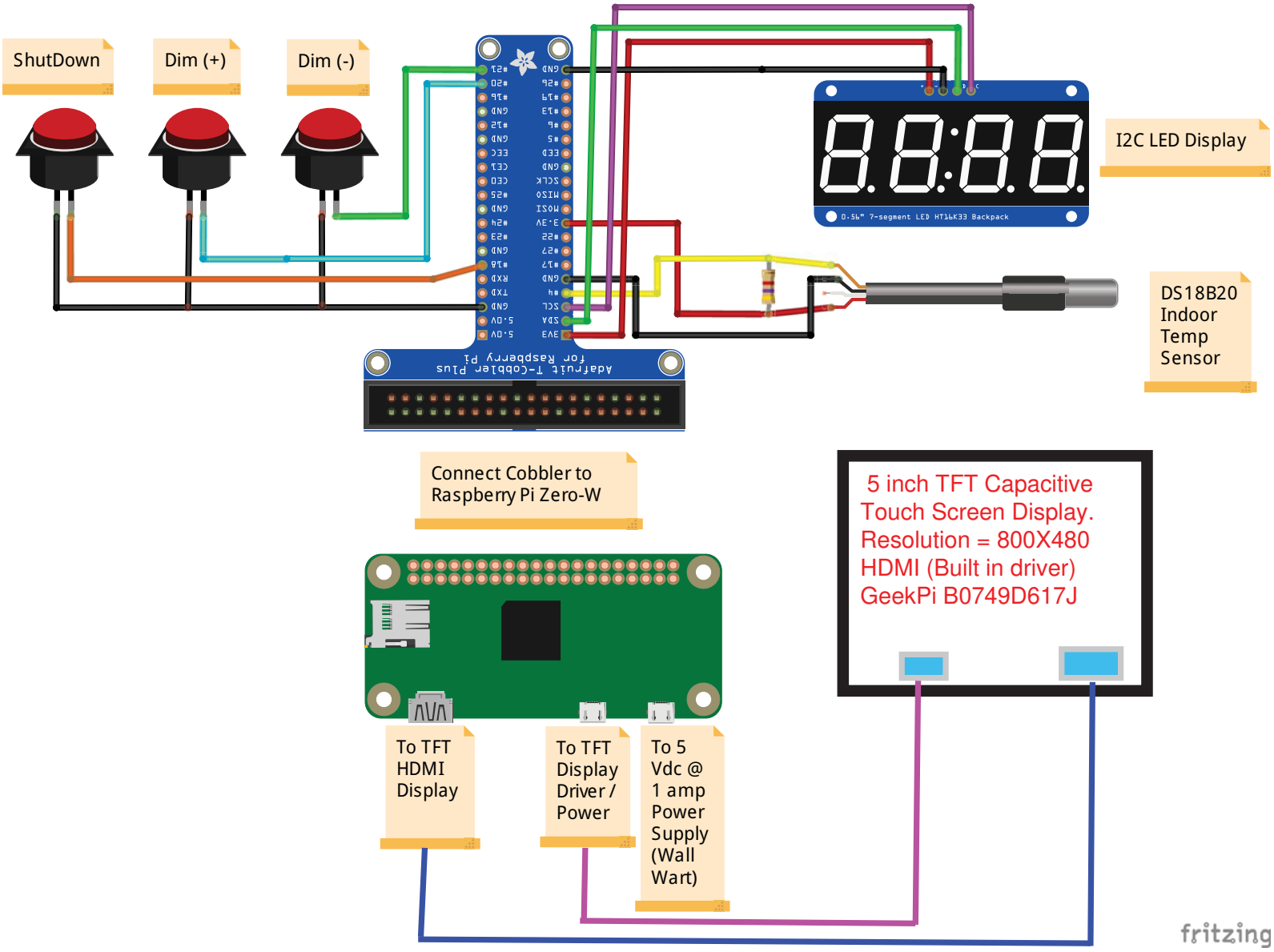

Weather
Weather Station
Current forecast for Buckeye, AZ is:
07/15/2018
Conditions Are Partly Cloudy,
Winds Are Calm
100.3
Temperature(F)
31%
Humidity
29.91
Pressure(in)
Clock
8:24:0
Shut
OFF
(-)
(+)



TFT Weather Display With Indoor Temperatureand LED Clock



```
#########################
# Weather Station,      #
# LED Clock and Indoor  #
# Temp                  #
# By, Roy H. Guerra Jr. #
# 7/10/18               #
#########################

"""
Pre-Reqs
---------

**Start Installing required packages by running the following commands**
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install vim git python-requests python-smbus i2c-tools python-imaging
python-smbus build-essential python-dev rpi.gpio python3 python3-pip libi2c-dev

**Install SM Bus**
  sudo apt install python-smbus

**Install i2c LED Backpack Python Drivers**
  git clone https://github.com/adafruit/Adafruit_Python_LED_Backpack
  cd Adafruit_Python_LED_Backpack/
  sudo python3 setup.py install

 Find IIC Address by typing in the following command on the terminal:
  sudo i2cdetect -y 1

**DHT11 Install** (Do not use, see TempDS1820.py for setup)
  git clone https://github.com/adafruit/Adafruit_Python_DHT.git
  cd Adafruit_Python_DHT/
  sudo python3 setup.py install

**To add a 5" custom TFT color display with touch screen perform**
  1) Open config File by:
  sudo nano /boot/config.txt
  2) Uncomment and/or add missing lines to the following:
  framebuffer_width=800
  framebuffer_height=480
  hdmi_force_hotplug=1
  hdmi_group=2
  hdmi_mode=87
  hdmi_cvt 800 480 60 6 0 0 0
 save config file by typing the following:
     (Cntrl + O) then "enter" then (Cntrl + X)
  3) If using a display without GPIO, connect:
     - HDMI TFT to HDMI Pi
     - TFT power to Pi USB for keyboard

**Disable screen blanking on pi zero-w***
  sudo nano ~/.config/lxsession/LXDE-pi/autostart
  add the following lines: (can only navigate by keyboard, no mouse)
    @ xset s 0 0
    @ xset s noblank
    @ xset s noexpose
    @ xset dpms 0 0 0
  save config file by typing the following:
     (Cntrl + O) then "enter" then (Cntrl + X)

***To add a Keyboard For TFT Touch Screen Displays**(only if not using Roy TochPad)
To get the Florence keyboard working you need to install at-spi2-core

sudo apt-get install at-spi2-core
sudo apt-get update
sudo apt-get install Florence

```

```
67  To Delete any unwanted programs that were installed use;
68  sudo apt-get --purge remove <package>
69
70  To delete any leftover orphans
71  sudo apt-get autoremove --purge
72
73  **To Dim Display add:***
74    Set brightness of entire display to specified value (16 levels, from
75     0 to 15).
76
77     if brightness < 0 or brightness > 15:
78        raise ValueError('Brightness must be a value of 0 to 15.')
79
80  Note- pygame display is set for (800X480) display, if screen changes size, you need to
    rescale pixels
81
82  **Place weather icons folder in /home/pi directory
83
84  **Program will re-start if internet connection is broke
85   **to stop program, use a keyboard or VNC (virtual connection) and hit the "escape"
     key
86  """
87
88  # Import Libraries
89  # ----------------
90  from TempDS1820 import read_temp # Import seperate Program and Function
91  from Roy_menu_touch_pad import result # Import seperate Touch Pad Program
92  from time import sleep
93  import RPi.GPIO as GPIO
94  import time
95  from datetime import datetime
96  from Adafruit_LED_Backpack import SevenSegment
97  import urllib.request as request # Use for Python 3
98  import json
99  import pygame
100 from pygame.locals import*
101 import os
102
103 # I/O Configuration
104 # -----------------
105 led_pin = 16 # LED pin to show sample collection
106 shutdown_pin = 18 # GPIO of shutdown push button
107 dim_up = 20 # GPIO of dim up button
108 dim_dwn = 21 # GPIO of dim down button
109
110 # Global Variables
111 # ----------------
112 intensity = 15
113
114 # Function to Detect shutdown Button Press (this is an interrupt)
115 # ---------------------------------------------------------------
116 def shutdownButtonPress(event, hold_time=3): # 3 sec. delay
117     sleep(hold_time)
118     if (GPIO.input(shutdown_pin) != GPIO.LOW):
119         return
120     safeClose()
121     shutdownPi()
122
123 # GPIO Software Configuration
124 # ---------------------------
125 GPIO.setmode(GPIO.BCM) # Use BCM pin numbers and not board numbers
126 GPIO.setwarnings(False)
127 GPIO.setup(dim_up, GPIO.IN, pull_up_down=GPIO.PUD_UP) # Enable internal Pull-Up Resistor
128 GPIO.setup(dim_dwn, GPIO.IN, pull_up_down=GPIO.PUD_UP) # Enable internal Pull-Up Resistor
129 GPIO.setup(shutdown_pin, GPIO.IN, pull_up_down=GPIO.PUD_UP) # Shutdown pushbutton mode
130 GPIO.setup(led_pin, GPIO.OUT) # LED Output
131 GPIO.add_event_detect(shutdown_pin, GPIO.FALLING, callback=shutdownButtonPress,
```

```
    bouncetime=200)

# Initial Setup Parameters
# ------------------------
pygame.init() # Initialize Pygame Screen

# use a (r, g, b) tuple as color
# ------------------------------
red = (255, 0, 0)
green = (0, 255, 0)
blue = (0, 0, 255)
yellow = (255, 255, 0)
magenta = (0, 255, 255)
white = (255, 255, 255)
gray = (127, 127, 127)
cyan = (255, 0, 255)
black = (0, 0, 0)

# Weatherunderground API Key
# --------------------------
#key = input('Please enter the API Key from Weather Underground: ')
key = "f94a68170dbe9cd3"

# Keyboard or Touch Screen Pad Input Command for Zip Code
# -------------------------------------------------------
zip_code = result() # Comes from the return command of Roy Touch Pad Program if used

# Comment out below with quotes (if not using a key board or virtual connection)
"""
zip_code = input('For which zip code would you like to see the weather? ')
if not zip_code.isdigit():
    raise ValueError("Zip Code is not a Positive Number")
elif ((len(zip_code) < 5) or (len(zip_code) > 5)):
    raise ValueError("Zip Code is not Five Digits")
print("zipcode = " + zip_code)
#zip_code = '85396' #Used for Testing
"""

# Weatherunderground URL to call
# ------------------------------
filename = "http://api.wunderground.com/api/" + key + "/geolookup/conditions/q/AZ/"+
zip_code +".json"

# 7 Segment Clock Display on i2c on address 0x70
# ----------------------------------------------
segment = SevenSegment.SevenSegment(address=0x70)

# Initialize the display, and set brightness level
# ------------------------------------------------
segment.begin()
segment.set_brightness(intensity) #0-15

# create the basic window/screen and a title/caption
# default is a black background
# ------------------------------
size = width, height = 800, 480
#screen = pygame.display.set_mode((800, 480)) # comment out after testing
screen = pygame.display.set_mode(size, pygame.FULLSCREEN)
pygame.mouse.set_visible(False) # Disable mouse
screen.fill(black)
pygame.display.set_caption("Weather Station By Roy H Guerra Jr")

# Fetch weather icon dictionary based on time of day
# --------------------------------------------------
def convert_icon(key):

    time_stamp = datetime.now().strftime("%H") # Get Hour
```

```
197
198         hr = int(time_stamp) # Convert to an Integer
199
200         day_icons={ # Daytime dictionary
201             "chanceflurries":("12.png"),
202             "chancerain":("9.png"),
203             "chancesleet":("8.png"),
204             "chancesnow":("35.png"),
205             "chancetstorms":("1.png"),
206             "clear":("32c.png"),
207             "cloudy":("26.png"),
208             "flurries":("12.png"),
209             "fog":("19.png"),
210             "hazy":("19.png"),
211             "mostlycloudy":("28.png"),
212             "mostlysunny":("34.png"),
213             "partlycloudy":("30.png"),
214             "scatteredclouds":("30.png"),
215             "partlysunny":("28.png"),
216             "rain":("9.png"),
217             "sleet":("17.png"),
218             "snow":("35.png"),
219             "sunny":("36c.png"),
220             "tstorms":("1.png"),
221             "overcast":("22.png"),
222             "na":("na.png")
223          }
224
225         night_icons={ # night time dictionary
226             "chanceflurries":("12.png"),
227             "chancerain":("9.png"),
228             "chancesleet":("8.png"),
229             "chancesnow":("35.png"),
230             "chancetstorms":("1.png"),
231             "clear":("31c.png"),
232             "cloudy":("26.png"),
233             "flurries":("12.png"),
234             "fog":("22.png"),
235             "hazy":("22.png"),
236             "mostlycloudy":("27.png"),
237             "mostlysunny":("33.png"),
238             "partlycloudy":("29.png"),
239             "scatteredclouds":("29.png"),
240             "partlysunny":("27.png"),
241             "rain":("9.png"),
242             "sleet":("17.png"),
243             "snow":("35.png"),
244             "sunny":("31c.png"),
245             "tstorms":("47.png"),
246             "overcast":("22.png"),
247             "na":("na.png")
248         }
249         # Logic to decide which dictionary to use
250         # ---------------------------------------
251         if (hr >= 20 or hr <= 6):
252             new = night_icons[key]
253         elif (hr < 20 and hr > 6):
254             new = day_icons[key]
255         else:
256             new = day_icons['na']
257
258         return new # Return the chosen weather icon
259
260
261     # Function to blink LED (Not Used)
262     # --------------------------------
263     def LEDblink(iterations):
```

```
    index = 0
    while (index < iterations):
        GPIO.output(led_pin, GPIO.HIGH)
        time.sleep(0.5)
        GPIO.output(led_pin, GPIO.LOW)
        time.sleep(0.5)
        index +=1

# Function to Read Intensity Buttons
# ----------------------------------
def Button_Read():
    global intensity
    if (GPIO.input(dim_up) == GPIO.LOW) and (intensity < 15 ):
        intensity +=1
    if (GPIO.input(dim_dwn) == GPIO.LOW) and (intensity != 0 ):
        intensity -=1
    #print("Intensity : " + str(intensity)) # Used for debug
    segment.set_brightness(intensity) #0-15

# Function to display time
# ------------------------
def tm():
    now = datetime.now() # Get current time
    hour = now.hour # Get current hour
    # convert to 12 hour time
    if (hour > 12):
        hour = hour - 12
    elif (hour == 0):
        hour = 12
    else:
        pass
    # Get time values and clear display
    minute = now.minute
    second = now.second
    segment.clear()
    # Set hours to blank out "0" if less than "10"
    if (hour < 10):
        segment.set_digit(1, hour % 10) # Ones
    else:
        segment.set_digit(0, int(hour / 10)) # Tens
        segment.set_digit(1, hour % 10) # Ones
    # Set minutes
    segment.set_digit(2, int(minute / 10)) # Tens
    segment.set_digit(3, minute % 10) # Ones
    segment.set_colon(second % 2) # Toggle colon at 1Hz
    segment.write_display() # Write the display buffer to the hardware
    sleep(0.25) # Wait a quarter second to prevent colon blinking
    Button_Read() # Goto to Button Read Function

# Function to print Text on Screen
# --------------------------------
def printText(txtText, Textfont, Textsize , Textx, Texty, Textcolor):
    myfont = pygame.font.SysFont(Textfont, Textsize) # Choose Font
    label = myfont.render(txtText, 1, Textcolor) # Apply to Text Label
    screen.blit(label, (Textx, Texty)) # Put OBJECT at Screen position
    pygame.display.update() # Update Screen

# Function to clear and update screen
# -----------------------------------
def clearScreen(): # Clears the pygame screen of all drawn objects.
    screen.fill(black)
    pygame.display.update()

# Function to get weather from weatherunderground in JSON
# -------------------------------------------------------
def getweather():
    mth = datetime.now().strftime("%m")
```

```
        dy = datetime.now().strftime("%d")
        yr = datetime.now().strftime("%Y")
        TempF = read_temp() # Pull from a separate program file
        #print("Indoor Temp: " + TempF) # Used for debug

        # Parse JavaScript Object Notation
        f = request.urlopen(filename)
        json_string = f.read().decode('utf-8')
        parsed_json = json.loads(json_string)
        location = parsed_json['location']['city']
        state = parsed_json['location']['state']
        temp_f = parsed_json['current_observation']['temp_f']
        hum = parsed_json['current_observation']['relative_humidity']
        wind = parsed_json['current_observation']['wind_string']
        pressure = parsed_json['current_observation']['pressure_in']
        weath = parsed_json['current_observation']['weather']
        observ = parsed_json['current_observation']['observation_time']

        # Used for degugging, comment out when program is working
        # -------------------------------------------------------
        #print ("Current forecast for %s, %s is: " % (location, state))
        #print ("Observation Time = %s " % (observ))
        #print ("Conditions are %s, Winds are %s " % (weath, wind))
        #print ("Temperature = %s Deg_F, Humidity = %s, Pressure = %s in" % (temp_f, hum,
        pressure))
        #print ("Month, day, year are " + mth + dy + yr)
        #print("Indoor Temp: " + TempF)


        # Pass these Texts with Fonts & Color to above function
        #          (Text, Font, Size, X, Y, Color)
        #-------------------------------------------------------
        pygame.draw.rect(screen, white, (220,12,340,60), 4) #For Title Box(X,Y,W,H)
        pygame.draw.rect(screen, white, (540,165,210,80), 2) #For Indoor Temp & Hum
        Box(X,Y,W,H)
        printText("Weather Station", "MS Comic Sans" , 60, 232, 20, white)
        printText("Current forecast for %s, %s is: " % (location, state), "MS Comic Sans" ,
        42, 50, 110, magenta)
        printText("Observation Time Was %s " % (observ), "MS Comic Sans" , 20, 100, 445,
        green)
        printText("Conditions Are %s, " % (weath), "MS Comic Sans", 40, 50, 270, cyan)
        printText("Winds Are %s " % (wind), "MS Comic Sans", 40, 180, 320, blue)
        printText(mth + "/" + dy + "/" + yr, "MS Comic Sans" , 50, 560, 175, white)
        printText("Indoor Temp = %s*F" % (TempF), "MS Comic Sans", 24, 560, 218, green)
        #printText("Indoor Hum: %s" % (Hum), "MS Comic Sans", 24, 575, 270, green) # Not Used

        # Pass this data with Fonts & Color to above function
        #          (Text, Font, Size, X, Y, Color)
        #-------------------------------------------------------
        printText("%s" % (temp_f), "MS Comic Sans", 60, 70, 360, yellow)
        printText("%s" % (hum), "MS Comic Sans", 60, 380, 360, yellow)
        printText("%s" % (pressure), "MS Comic Sans", 60, 660, 360, yellow)

        # Pass these Texts with Fonts & Color to above function
        #          (Text, Font, Size, X, Y, Color)
        #-------------------------------------------------------
        printText("Temperature(F)", "MS Comic Sans", 40, 5, 400, red)
        printText("Humidity", "MS Comic Sans", 40, 360, 400, red)
        printText("Pressure(in)", "MS Comic Sans", 40, 625, 400, red)

        # Search JSON Weather String for keywords
        # ---------------------------------------
        if "Sunny" in str(weath):
            y = ('sunny')
        elif "Flurries" in str(weath):
            y = ('flurries')
        elif "Fog" in str(weath):
```

```
            y = ('fog')
        elif "Hazy" in str(weath):
            y = ('hazy')
        elif "Clear" in str(weath):
            y = ('clear')
        elif "Cloudy" in str(weath):
            y = ('cloudy')
        elif "Snow" in str(weath):
            y = ('snow')
        elif "Sleet" in str(weath):
            y = ('sleet')
        elif "Rain" in str(weath):
            y = ('rain')
        elif "Thunderstorm" in str(weath):
            y = ('tstorms')
        elif "Partly Sunny" in str(weath):
            y = ('partlysunny')
        elif "Partly Cloudy" in str(weath):
            y = ('partlycloudy')
        elif "Scattered Clouds" in str(weath):
            y = ('scatteredclouds')
        elif "Mostly Sunny" in str(weath):
            y = ('mostlysunny')
        elif "Mostly Cloudy" in str(weath):
            y = ('mostlycloudy')
        elif "Chance Of Thunderstorm" in str(weath):
            y = ('chancetstorms')
        elif "Chance Of Snow" in str(weath):
            y = ('chancesnow')
        elif "Chance Of Sleet" in str(weath):
            y = ('chancesleet')
        elif "Chance Of Rain" in str(weath):
            y = ('chancerain')
        elif "Chance Of Flurries" in str(weath):
            y = ('chanceflurries')
        elif "Overcast" in str(weath):
            y = ('overcast')

        else:
            y = ('na')

        z = convert_icon(y) # Pass the weather icon key to the convert dictionary function

        img = pygame.image.load('/home/pi/icon_img/' + z) # Load the weather icon image

        screen.blit(img, (150,150)) # Blit icon to screen

        pygame.display.update()

        f.close() # Close URL request

    # Function to Shutdown Raspberry Pi Operating System
    # --------------------------------------------------
    def shutdownPi(): # Shutdown Raspberry Pi
        os.system("sudo shutdown -h now")

    # Function to Shutdown Raspberry Pi Resources
    # -------------------------------------------
    def safeClose():
        pygame.quit()
        GPIO.cleanup()

    getweather() # Goto function on startup

    i = 0 # Global counting variable

    # Main Program Loop:
```

```
# ------------------
while (True):
    try:
        for event in pygame.event.get(): #Escape pygame Full Screen if virtual or
        keyboard connection
            if event.type == KEYDOWN:
                if event.key == K_ESCAPE:
                    pygame.quit()
                    sys.exit()
        tm() # Goto Function
        i +=1 # Increment by "1"
        #LEDblink(2) # Blink Led function, results add 2 sec. delay
        if i >= 7200: # Update every 30 minutes = (0.25 from tm() delay section * 7200
        = seconds)
            clearScreen() # Goto function
            getweather() # Goto function
            i = 0 # reset counting variable

    except Exception as e:
        print("Exception: ") # Used for debug
        print(e) # Print Exception for debugging
        clearScreen() # Goto function
        j = 30
        while (j > 0):  # Loop until "0"
            printText("Resolving Issues", "MS Comic Sans" , 60, 220, 100, cyan)
            printText("Please Wait " + str(j) + " min", "MS Comic Sans" , 60, 220, 270,
            green)
            sleep (60) # 1 min wait
            j -=1 # count down
            clearScreen() # Goto function
        clearScreen() # Goto function
        pass
    # Uncomment the below statements if you want to shut down after runniing once
    #else:
        #safeClose() # Goto function
        #shutdownPi()  # Goto function
```

```
########################
# Touch Pad Menu       #
# By, Roy H. Guerra Jr. #
# 7/8/18               #
########################


# Import Modules
# --------------
import pygame
from pygame.locals import *
from time import sleep
import os

# Initialize pygame, and hide mouse
# ---------------------------------
pygame.init()
pygame.mouse.set_visible(True)

# Global Variables
# ----------------
zipcode = ''

# Function for printing Text
# --------------------------
def make_button(text, xpo, ypo, height, width, color):
    font=pygame.font.Font(None,42)
    label=font.render(str(text), 1, (color))
    screen.blit(label,(xpo,ypo))
    pygame.draw.rect(screen, blue, (xpo-10,ypo-10,width,height),5)

# Function for printing Labels
# ----------------------------
def make_label(text, xpo, ypo, fontsize, color):
    font=pygame.font.Font(None,fontsize)
    label=font.render(str(text), 1, (color))
    screen.blit(label,(xpo,ypo))

# define function that checks for touch location
# ----------------------------------------------
def on_touch():
    # get the position that was touched
    touch_pos = (pygame.mouse.get_pos() [0], pygame.mouse.get_pos() [1])
    #  x_min                 x_max   y_min                 y_max
    # clear button event
    if 351 <= touch_pos[0] <= 456 and 19 <= touch_pos[1] <= 72:
        button(10)
    # button 1 event
    if 29 <= touch_pos[0] <= 217 and 103 <= touch_pos[1] <= 138:
        button(1)
    # button 2 event
    if 262 <= touch_pos[0] <= 450 and 103 <= touch_pos[1] <= 138:
        button(2)
    # button 3 event
    if 29 <= touch_pos[0] <= 217 and 181 <= touch_pos[1] <= 211:
        button(3)
    # button 4 event
    if 262 <= touch_pos[0] <= 450 and 181 <= touch_pos[1] <= 211:
        button(4)
    # button 5 event
    if 29 <= touch_pos[0] <= 217 and 255 <= touch_pos[1] <= 287:
        button(5)
    # button 6 event
    if 262 <= touch_pos[0] <= 450 and 255 <= touch_pos[1] <= 287:
        button(6)
    # button 7 event
    if 29 <= touch_pos[0] <= 217 and 330 <= touch_pos[1] <= 361:
```

```
68          button(7)
69      # button 8 event
70      if 262 <= touch_pos[0] <= 450 and 330 <= touch_pos[1] <= 361:
71          button(8)
72      # button 9 event
73      if 29 <= touch_pos[0] <= 217 and 405 <= touch_pos[1] <= 437:
74          button(9)
75      # button 0 event
76      if 262 <= touch_pos[0] <= 450 and 405 <= touch_pos[1] <= 437:
77          button(0)
78
79  # Define each Button Press Action
80  # ------------------------------
81  def button(number):
82      #print "You pressed button ",number # Used for Debug
83      global zipcode
84      zipcode += str(number)
85      if number == 1:
86          screen.fill(black)
87          font=pygame.font.Font(None,48)
88          label=font.render("Number = 1", 1, (yellow))
89          screen.blit(label,[30,150])
90          pygame.display.flip()
91          sleep(0.5)
92      if number == 2:
93          screen.fill(black)
94          font=pygame.font.Font(None,48)
95          label=font.render("Number = 2", 1, (yellow))
96          screen.blit(label,[30,150])
97          pygame.display.flip()
98          sleep(0.5)
99      if number == 3:
100         screen.fill(black)
101         font=pygame.font.Font(None,48)
102         label=font.render("Number = 3", 1, (yellow))
103         screen.blit(label,[30,150])
104         pygame.display.flip()
105         sleep(0.5)
106     if number == 4:
107         screen.fill(black)
108         font=pygame.font.Font(None,48)
109         label=font.render("Number = 4", 1, (yellow))
110         screen.blit(label,[30,150])
111         pygame.display.flip()
112         sleep(0.5)
113     if number == 5:
114         screen.fill(black)
115         font=pygame.font.Font(None,48)
116         label=font.render("Number = 5", 1, (yellow))
117         screen.blit(label,(30,150))
118         pygame.display.flip()
119         sleep(0.5)
120     if number == 6:
121         screen.fill(black)
122         font=pygame.font.Font(None,48)
123         label=font.render("Number = 6", 1, (yellow))
124         screen.blit(label,(30,150))
125         pygame.display.flip()
126         sleep(0.5)
127     if number == 7:
128         screen.fill(black)
129         font=pygame.font.Font(None,48)
130         label=font.render("Number = 7", 1, (yellow))
131         screen.blit(label,(30,150))
132         pygame.display.flip()
133         sleep(0.5)
134     if number == 8:
```

```
            screen.fill(black)
            font=pygame.font.Font(None,48)
            label=font.render("Number = 8", 1, (yellow))
            screen.blit(label,(30,150))
            pygame.display.flip()
            sleep(0.5)
        if number == 9:
            screen.fill(black)
            font=pygame.font.Font(None,48)
            label=font.render("Number = 9", 1, (yellow))
            screen.blit(label,(30,150))
            pygame.display.flip()
            sleep(0.5)
        if number == 0:
            screen.fill(black)
            font=pygame.font.Font(None,48)
            label=font.render("Number = 0", 1, (yellow))
            screen.blit(label,(30,150))
            pygame.display.flip()
            sleep(0.5)
        if number == 10:
            screen.fill(black)
            font=pygame.font.Font(None,48)
            label=font.render("Clearing Zip Code Data", 1, (yellow))
            screen.blit(label,(30,150))
            pygame.display.flip()
            zipcode = ''
            sleep(0.5)

    # colors    R    G    B
    # ----------------------
    white   = (255, 255, 255)
    red     = (255,   0,   0)
    green   = (  0, 255,   0)
    blue    = (  0,   0, 255)
    black   = (  0,   0,   0)
    cyan    = ( 50, 255, 255)
    magenta = (255,   0, 255)
    yellow  = (255, 255,   0)
    orange  = (255, 127,   0)

    # Set Screen Size
    # ---------------
    size = width, height = 480, 480
    screen = pygame.display.set_mode(size,pygame.FULLSCREEN)
    #screen = pygame.display.set_mode(size) # Comment out after testing

    # Background Color
    # ----------------
    screen.fill(black)

    def Main_Menu():
        screen.fill(black)
        # Main Outer Border
        pygame.draw.rect(screen, blue, (0,0,480,480),10)
        # Box for Clear Button
        pygame.draw.rect(screen, cyan, (350,20,110,55), 3)
        # Clear Button Label
        make_label("  Clear", 353, 32, 40, magenta)
        # First Row Label
        make_label("ZipCode = " + zipcode, 32, 30, 48, green)
        # Second Row buttons 1 and 2
        make_button("          1", 30, 105, 55, 210, red)
        make_button("          2", 260, 105, 55, 210, red)
        # Third Row buttons 3 and 4
        make_button("          3", 30, 180, 55, 210, red)
        make_button("          4", 260, 180, 55, 210, red)
```

```
    # Fourth Row Buttons 5 and 6
    make_button("          5", 30, 255, 55, 210, red)
    make_button("          6", 260, 255, 55, 210, red)
    # Fifth Row Buttons 7 and 8
    make_button("          7", 30, 330, 55, 210, red)
    make_button("          8", 260, 330, 55, 210, red)
    # Sixth Row Buttons 9 and 0
    make_button("          9", 30, 405, 55, 210, red)
    make_button("          0", 260, 405, 55, 210, red)

def result():
    return zipcode

# While loop to Manage Touch Screen Inputs
# ----------------------------------------
while True:
    try:
        Main_Menu()
        for event in pygame.event.get():
            if event.type == pygame.MOUSEBUTTONDOWN:
                pos = (pygame.mouse.get_pos() [0], pygame.mouse.get_pos() [1])
                #print("X & Y: " + str(pos)) # Used to get mouse button positions for
                x,y(min & max)
                on_touch()
        #Ensure there is always a safe way to end the program if the touch screen fails
            if event.type == KEYDOWN:
                if event.key == K_ESCAPE:
                    pygame.quit()
                    sys.exit()
        pygame.display.update()
        sleep(0.5)
        if len(zipcode) == 5:
            Main_Menu()
            pygame.display.update()
            sleep(2)
            result()
            break
    except Exception as e:
        print("Exception: ") # Used for debug
        print(e) # Print Exception for debugging
        pygame.quit()
        sleep(2)
        pass
print("zipcode = " + zipcode) # Used for debug
#pygame.quit() # Comment out if using in another program
```

```
# ------------------------
# DS1820 Program Routine
# By Roy Guerra Jr.
# 7/11/18
# ------------------------

# Save program in Pi directory and call 'TempDS1820'
# Call program by executing 'from TempDS1820 import read_temp'

# To Install One Wire Interface (unless already connected with Cayenne)
# ------------------------------------------------------------------

# 1. To add support, we first need to open up the boot config file, this can be done by running the following command:
#    sudo nano /boot/config.txt
# 2. At the bottom of this file enter the following.
#    dtoverlay=w1-gpio
# 3. Once done save & exit by pressing ctrl x, enter, them ctrl y. Now reboot the Pi by running the following command.
#    sudo reboot
# 4. You can skip to downloading the code onto the Pi or follow the next few steps to check that the sensor is actually working.
# 5. Once the Raspberry Pi has booted back up we need to run modprobe so we can load the correct modules.
#    sudo modprobe w1-gpio pullup=1
#    sudo modprobe w1-therm
# 6. Now change into the devices directory and use ls to see the folder/files in this directory.
#    cd /sys/bus/w1/devices
#    ls
# 7. Now run the following command, change the numbering after cd to what has appeared in your directory by using the ls command. (If you have multiple sensors there will be more than one directory)
#    cd 28-000007602ffa
# 8. Now run the following command.
#    cat w1_slave (may not work with cayenne)
# Connections are as follows:
# Red wire to 3.3V, Black wire to ground and Yellow wire is data, pin #4.
# Install Options
# ===============
#To enable the one-wire interface you need to add the following line to #/boot/config.txt, before rebooting your Pi:
#  dtoverlay=w1-gpio
#   or
#  dtoverlay=w1-gpio,gpiopin=x
#if you would like to use a custom pin (default is BCM4)
# Aternatively you can enable the one-wire interface on demand using
# raspi-config, or the following:
# sudo modprobe w1-gpio
# Newer kernels (4.9.28 and later) allow you to use dynamic overlay
# loading instead, including creating multiple 1-Wire busses to be used
# at the same time:
# sudo dtoverlay w1-gpio gpiopin=4 pullup=0  # header pin 7
# sudo dtoverlay w1-gpio gpiopin=17 pullup=0 # header pin 11
# sudo dtoverlay w1-gpio gpiopin=27 pullup=0 # header pin 13
# once any of the steps above have been performed, and discovery is
# complete you can list the devices that your Raspberry Pi has discovered # via all 1-Wire busses (by default BCM4), like so:
# ls /sys/bus/w1/devices/



# Import Libraries
# ----------------
from time import sleep
import os
import glob
```

```
59
60
61  # Function to get Temperature:
62  # ----------------------------
63  os.system('modprobe w1-gpio')
64  os.system('modprobe w1-therm')
65
66  base_dir = '/sys/bus/w1/devices/'
67  device_folder = glob.glob(base_dir + '28*')[0]
68  device_file = device_folder + '/w1_slave'
69
70  def read_temp_raw():
71      f = open(device_file, 'r')
72      lines = f.readlines()
73      f.close()
74      return lines
75
76  def read_temp():
77      lines = read_temp_raw()
78      while lines[0].strip()[-3:] != 'YES':
79          time.sleep(0.2)
80          lines = read_temp_raw()
81      equals_pos = lines[1].find('t=')
82      if equals_pos != -1:
83          temp_string = lines[1][equals_pos+2:]
84          temp_c = float(temp_string) / 1000.0
85          temp_f = temp_c * 9.0 / 5.0 + 32.0
86          return str("%.2f" % temp_f) # or temp_c
87
88  # Main Program
89  # ------------
90  for num in range(3): #To run Test 3 times
91  #while True: # To run cont.  Comment out if not used
92    t = read_temp()
93    print('Temperature = ' + t)
94    sleep(1)
95
96
```