

```

# GUI to control heating
# 31.8.16
# by Julian Rogers
# heat_control22a - trying to trap more errors!
# error handling now improved
# version has reduced window size to fit on Pi 7" screen
# this version does self-refresh
# need to investigate bug in stat update - done!
# note difference in window size & timeout compared to Windows version
# as data regarding date etc can't be fitted in screen-space available
# temp sensor 2 not yet implemented

IP = "192.168.1.177" #remote ip address
DEST_PORT = 8888 #destination port
THIS_PORT = 6000 #port on this computer

from tkinter import * #GUI

import datetime
import time
import socket #UDP
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

# create the root window
root = Tk()

# modify the window
root.title("mr-r's Boiler CONTROLLER")
root.geometry("700x430")
root.configure(bg = "teal")

# create a frame
app = Frame(root)
app.configure(bg = "teal")
app.grid()

lbl_time = Label(app, text = "Time > ", font = ( "Arial ", 16), fg = "black", bg = "teal")
lbl_time.grid(row = 2, column = 2, sticky = E)

#create labels for minutes and hours display
# This is a bodge for aesthetc reasons! Labels are unseen and just used to store values
# until I get round to a more efficient solution!
hour_lab = Label(app, text = "00", font = ("Arial", 4), fg = "teal", bg = "teal")
hour_lab.grid(row = 2, column = 3, sticky = E)

lbl_colon = Label(app, text = ":", font = ( "Arial", 4), fg = "teal", bg = "teal")
lbl_colon.grid(row = 2, column = 4, sticky = W)

mins_lab = Label(app, text = "00", font = ("Arial", 4), fg = "teal", bg = "teal")
mins_lab.grid(row = 2, column = 5, sticky = W)

time_but = Button(app, text = "00" + ":" + "00", font = ("Arial", 16), fg = "maroon", bg = "cornflower blue")
time_but.grid(row = 2, column = 3, sticky = W)

#-----

#blank row
blank_row = Label(app, text = "", fg = "black", bg = "teal")
blank_row.grid(row = 7, column = 1)

#-----

#heating and water labels
lbl_heating = Label(app, text = "HEATING", font = ( "Arial ", 16), fg = "maroon", bg = "teal")
lbl_heating.grid(row = 2, column = 6, columnspan = 4, sticky = E)

# effectively create blank column
lbl_row = Label(app, text = " ", bg = "teal")
lbl_row.grid(row = 2, column = 11)

lbl_water = Label(app, text = "WATER ", font = ( "Arial ", 16), fg = "maroon", bg = "teal")
lbl_water.grid(row = 2, column = 11, columnspan = 4, sticky = E)

#-----

# labels for heating and water on off times

```

```
lbl_time_on1 = Label(app, text = "Time on 1 > ", font = ( "Arial ", 16), fg = "black", bg = "teal")
lbl_time_on1.grid(row = 3, column = 6, sticky = E)
```

```
lbl_time_off1 = Label(app, text = "Time off 1 > ", font = ( "Arial ", 16), fg = "black", bg = "teal")
lbl_time_off1.grid(row = 4, column = 6, sticky = E)
```

```
lbl_time_on2 = Label(app, text = "Time on 2 > ", font = ( "Arial ", 16), fg = "black", bg = "teal")
lbl_time_on2.grid(row = 5, column = 6, sticky = E)
```

```
lbl_time_off2 = Label(app, text = "Time off 2 > ", font = ( "Arial ", 16), fg = "black", bg = "teal")
lbl_time_off2.grid(row = 6, column = 6, sticky = E)
```

```
lbl_advance_status = Label(app, text = "Advance ", font = ( "Arial ", 16), fg = "black", bg = "teal")
lbl_advance_status.grid(row = 5, column = 2, sticky = E)
```

```
#create a button to update on / off times on remote timer -----
```

```
def update_times():
    lbl_feedback.config(text = "updating on / off times ... ")
    #onoff_times = StringVar()
    onoff_times = water_on1_hour_label.cget("text") + "," + water_on1_mins_label.cget("text") + "," + \
        water_off1_hour_label.cget("text") + "," + water_off1_mins_label.cget("text") + "," + \
        water_on2_hour_label.cget("text") + "," + water_on2_mins_label.cget("text") + "," + \
        water_off2_hour_label.cget("text") + "," + water_off2_mins_label.cget("text") + "," + \
        heat_on1_hour_label.cget("text") + "," + heat_on1_mins_label.cget("text") + "," + \
        heat_off1_hour_label.cget("text") + "," + heat_off1_mins_label.cget("text") + "," + \
        heat_on2_hour_label.cget("text") + "," + heat_on2_mins_label.cget("text") + "," + \
        heat_off2_hour_label.cget("text") + "," + heat_off2_mins_label.cget("text")
    lbl_feedback_2.config(text = onoff_times)
    sock.sendto(bytes(onoff_times, "utf-8"), ("192.168.1.177", 8888))
```

```
update_times_but = Button(app, text = "Load on/off", font = ("Arial", 16), fg = "black", bg = "pink", command = update_times)
update_times_but.grid(row = 8, column = 12, columnspan = 5, sticky = W)
```

```
#-----####
```

```
# labels to show heat on/off times
```

```
heat_on1_hour_label = Label(app, text = "00", font = ("Arial", 16), fg = "black", bg = "teal")
heat_on1_hour_label.grid(row = 3, column = 7, sticky = E)
```

```
#colon label
lbl_colon2 = Label(app, text = ":", font = ( "Arial", 16), fg = "black", bg = "teal")
lbl_colon2.grid(row = 3, column = 8)
```

```
heat_on1_mins_label = Label(app, text = "00", font = ("Arial", 16), fg = "black", bg = "teal")
heat_on1_mins_label.grid(row = 3, column = 9, sticky = W)
```

```
heat_off1_hour_label = Label(app, text = "00", font = ("Arial", 16), fg = "black", bg = "teal")
heat_off1_hour_label.grid(row = 4, column = 7, sticky = E)
```

```
#colon label
lbl_colon3 = Label(app, text = ":", font = ( "Arial", 16), fg = "black", bg = "teal")
lbl_colon3.grid(row = 4, column = 8)
```

```
heat_off1_mins_label = Label(app, text = "00", font = ("Arial", 16), fg = "black", bg = "teal")
heat_off1_mins_label.grid(row = 4, column = 9, sticky = W)
```

```
#-----
```

```
heat_on2_hour_label = Label(app, text = "00", font = ("Arial", 16), fg = "black", bg = "teal")
heat_on2_hour_label.grid(row = 5, column = 7, sticky = E)
```

```
#colon label
lbl_colon3 = Label(app, text = ":", font = ( "Arial", 16), fg = "black", bg = "teal")
lbl_colon3.grid(row = 5, column = 8)
```

```
heat_on2_mins_label = Label(app, text = "00", font = ("Arial", 16), fg = "black", bg = "teal")
heat_on2_mins_label.grid(row = 5, column = 9, sticky = W)
```

```
heat_off2_hour_label = Label(app, text = "00", font = ("Arial", 16), fg = "black", bg = "teal")
heat_off2_hour_label.grid(row = 6, column = 7, sticky = E)
```

```
#colon label
lbl_colon4 = Label(app, text = ":", font = ( "Arial", 16), fg = "black", bg = "teal")
lbl_colon4.grid(row = 6, column = 8)
```

```

heat_off2_mins_label = Label(app, text = "00", font = ("Arial", 16), fg = "black", bg = "teal")
heat_off2_mins_label.grid(row = 6, column = 9, sticky = W)

#-----
#advance buttons

heat_advance_but = Button(app, text = "N", font = ("Arial", 16), fg = "maroon", bg = "pink")
heat_advance_but.grid(row = 5, column = 3, columnspan = 6, sticky = W)

def heat_adv():
    but_text = heat_advance_but.config('text')[-1]

    if but_text == "Y":
        but_text = "N"
    else:
        but_text = "Y"

    heat_advance_but.config(text = but_text)
    other_but = water_advance_but.config("text")[-1]
    lbl_feedback_2.config(text = but_text + other_but)
    sock.sendto(bytes(but_text + other_but, "utf-8"), ("192.168.1.177", 8888))
    get_data_remote()
    clear_feedback()
heat_advance_but.config(command = heat_adv)

water_advance_but = Button(app, text = "N", font = ("Arial", 16), fg = "maroon", bg = "pink")
water_advance_but.grid(row = 5, column = 4, columnspan = 6, sticky = W)

def water_adv():
    but_text = water_advance_but.config('text')[-1]

    if but_text == "Y":
        but_text = "N"
    else:
        but_text = "Y"

    water_advance_but.config(text = but_text)
    other_but = heat_advance_but.config("text")[-1]
    lbl_feedback_2.config(text = but_text + other_but)
    sock.sendto(bytes(other_but + but_text, "utf-8"), ("192.168.1.177", 8888))
    get_data_remote()
    clear_feedback()

water_advance_but.config(command = water_adv)

#-----
# labels to show water on/off times
water_on1_hour_label = Label(app, text = "00", font = ("Arial", 16), fg = "black", bg = "teal")
water_on1_hour_label.grid(row = 3, column = 12, sticky = E)

#colon label
lbl_colon4 = Label(app, text = ":", font = ("Arial", 16), fg = "black", bg = "teal")
lbl_colon4.grid(row = 3, column = 13)

water_on1_mins_label = Label(app, text = "00", font = ("Arial", 16), fg = "black", bg = "teal")
water_on1_mins_label.grid(row = 3, column = 14, sticky = W)

water_off1_hour_label = Label(app, text = "00", font = ("Arial", 16), fg = "black", bg = "teal")
water_off1_hour_label.grid(row = 4, column = 12, sticky = E)

#colon label
lbl_colon5 = Label(app, text = ":", font = ("Arial", 16), fg = "black", bg = "teal")
lbl_colon5.grid(row = 4, column = 13)

water_off1_mins_label = Label(app, text = "00", font = ("Arial", 16), fg = "black", bg = "teal")
water_off1_mins_label.grid(row = 4, column = 14, sticky = W)

#-----

water_on2_hour_label = Label(app, text = "00", font = ("Arial", 16), fg = "black", bg = "teal")
water_on2_hour_label.grid(row = 5, column = 12, sticky = E)

#colon label
lbl_colon6 = Label(app, text = ":", font = ("Arial", 16), fg = "black", bg = "teal")
lbl_colon6.grid(row = 5, column = 13)

```

```

water_on2_mins_label = Label(app, text = "00", font = ("Arial", 16), fg = "black", bg = "teal")
water_on2_mins_label.grid(row = 5, column = 14, sticky = W)

water_off2_hour_label = Label(app, text = "00", font = ("Arial", 16), fg = "black", bg = "teal")
water_off2_hour_label.grid(row = 6, column = 12, sticky = E)

#colon label
lbl_colon7 = Label(app, text = ":", font = ("Arial", 16), fg = "black", bg = "teal")
lbl_colon7.grid(row = 6, column = 13)

water_off2_mins_label = Label(app, text = "00", font = ("Arial", 16), fg = "black", bg = "teal")
water_off2_mins_label.grid(row = 6, column = 14, sticky = W)

#-----

#create blank row in effect

blank_row2 = Label(app, text = " ", bg = "teal")
blank_row2.grid(row = 9, column = 6)

#label to show switch status on remote timer
lbl_switch_status = Label(app, text = "Switch ", font = ("Arial", 16), fg = "black", bg = "teal")
lbl_switch_status.grid(row = 4, column = 2, sticky = E)

sw_heatstat_but = Button(app, text = "T", font = ("Arial", 16), fg = "maroon", bg = "grey")
sw_heatstat_but.grid(row = 4, column = 3, sticky = W)

sw_waterstat_but = Button(app, text = "T", font = ("Arial", 16), fg = "maroon", bg = "grey")
sw_waterstat_but.grid(row = 4, column = 4, sticky = W)

#-----

#create blank row in effect

blank_row3 = Label(app, text = " ", bg = "teal")
blank_row3.grid(row = 12, column = 6)

# labels to show remote temperature sensor values etc
lbl_temp_sensor1 = Label(app, text = "Temperature sensor 1 > ", font = ("Arial", 16), fg = "black", bg = "teal")
lbl_temp_sensor1.grid(row = 7, column = 2, colspan = 4, sticky = E)

lbl_temp_sensor2 = Label(app, text = "Temperature sensor 2 > ", font = ("Arial", 16), fg = "black", bg = "teal")
lbl_temp_sensor2.grid(row = 8, column = 2, colspan = 4, sticky = E)

temp_sensor1_label = Label(app, text = "00.0", font = ("Arial bold", 16), fg = "maroon", bg = "teal")
temp_sensor1_label.grid(row = 7, column = 6, sticky = W)

temp_sensor2_label = Label(app, text = "00.0", font = ("Arial bold", 16), fg = "maroon", bg = "teal")
temp_sensor2_label.grid(row = 8, column = 6, sticky = W)

lbl_thermostat = Label(app, text = "Stat", font = ("Arial", 16), fg = "black", bg = "teal")
lbl_thermostat.grid(row = 9, column = 2, colspan = 2, sticky = W)

#option menu to set thermostat value

var3 = StringVar(app)
var3.set("18") # initial value
thermostat_opt_men = OptionMenu(app, var3, "10", "11", "12", "13", "14", "15", "16", "17", "18", "19", "20", "21", "22")
thermostat_opt_men.config(font = ("Arial", 14), fg = "black", bg = "pink", width = 2)
thermostat_opt_men.grid(row = 10, column = 1, colspan = 2, sticky = W)

#-----

#create a button to update thermostat setting
def update_thermo():
    lbl_feedback.config(text = "updating stat setting... ")
    thermo_str = var3.get() + "0"
    lbl_feedback_2.config(text = thermo_str)
    sock.sendto(bytes(thermo_str, "utf-8"), ("192.168.1.177", 8888))
    get_data_remote()
    clear_feedback()

update_thermo_but = Button(app, text = "Update Stat", font = ("Arial", 16), fg = "black", bg = "pink", command = update_thermo)
update_thermo_but.grid(row = 10, column = 3, colspan = 3, sticky = W)

```

```

#create blank row in effect

blank_row4 = Label(app, text = " ", bg = "teal")
blank_row4.grid(row = 16, column = 2)

blank_row5 = Label(app, text = " ", bg = "teal")
blank_row5.grid(row = 17, column = 2)

#-----
#create a button to refresh all values as stored on timer
def get_data_remote():

    response = bytes("0/00,00,00,00,00,00,00,00,00,00,00,00,00,00,00T0S0A0t0", "utf-8")
    lbl_feedback.config(text = "Data from remote is...")

    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    try:
        sock.sendto(bytes("r", "utf-8"), ("192.168.1.177", 8888))

        sock.settimeout(5)

        response = sock.recv(100)

    except socket.error:
        lbl_feedback.config(text = "timed out!")
        response = bytes("0/00,00,00,00,00,00,00,00,00,00,00,00,00,00,00T0S0A0t0", "utf-8")

    sock.close()

    response = str(response, "utf-8")
    lbl_feedback_2.config(text = response)
    try:
        tim, other_data1 = response.split("/")

    except ValueError:
        response = bytes("0/00,00,00,00,00,00,00,00,00,00,00,00,00,00,00T0S0A0t0", "utf-8")
        response = str(response, "utf-8")
        tim, other_data1 = response.split("/")

    on_offs, other_data2 = other_data1.split("T")
    temp_sensor, other_data3 = other_data2.split("S")
    switch_mode, other_data4 = other_data3.split("A")
    advance_mode, thermo_setting = other_data4.split("t")

    thermo_setting = thermo_setting[0:2]
    thermostat_opt_men.config(var3.set(thermo_setting))

    #lbl_feedback_2.config(text = on_offs)

    tim_int = int(tim)
    clock_hrs = tim_int // 60
    clock_mins = tim_int % 60

    if clock_hrs < 10:
        clock_hrs_str = "0" + str(clock_hrs)
    else:
        clock_hrs_str = str(clock_hrs)

    if clock_mins < 10:
        clock_mins_str = "0" + str(clock_mins)
    else:
        clock_mins_str = str(clock_mins)

    hour_lab.config(text = clock_hrs_str)
    mins_lab.config(text = clock_mins_str)
    time_but.config(text = clock_hrs_str + ":" + clock_mins_str)

    a_status_int = int(advance_mode)
    adv_status = ("N", "Y")
    h_adv_status = adv_status[a_status_int % 2]
    w_adv_status = adv_status[a_status_int // 2]

    s_mode_int = int(switch_mode)
    sw_condition = ("T", "O", "C")

    w_mode_str = sw_condition[s_mode_int // 3]
    h_mode_str = sw_condition[s_mode_int % 3]

```

```

water_advance_but.config(text = w_adv_status)
heat_advance_but.config(text = h_adv_status)

sw_heatstat_but.config(text = h_mode_str)
sw_waterstat_but.config(text = w_mode_str)

won1_hrs,won1_mins,woff1_hrs,woff1_mins,won2_hrs,won2_mins,woff2_hrs,woff2_mins,\
hon1_hrs,hon1_mins,hoff1_hrs,hoff1_mins,hon2_hrs,hon2_mins,hoff2_hrs, hoff2_mins = on_offs.split(", ",15)

heat_on1_hour_label.config(text = hon1_hrs)
heat_on1_mins_label.config(text = hon1_mins)
heat_off1_hour_label.config(text = hoff1_hrs)
heat_off1_mins_label.config(text = hoff1_mins)
heat_on2_hour_label.config(text = hon2_hrs)
heat_on2_mins_label.config(text = hon2_mins)
heat_off2_hour_label.config(text = hoff2_hrs)
heat_off2_mins_label.config(text = hoff2_mins)
water_on1_hour_label.config(text = won1_hrs)
water_on1_mins_label.config(text = won1_mins)
water_off1_hour_label.config(text = woff1_hrs)
water_off1_mins_label.config(text = woff1_mins)
water_on2_hour_label.config(text = won2_hrs)
water_on2_mins_label.config(text = won2_mins)
water_off2_hour_label.config(text = woff2_hrs)
water_off2_mins_label.config(text = woff2_mins)

temperature = float(temp_sensor)

temperature = temperature / 10

temp_sensor = str(temperature)
temp_sensor1_label.config(text = temp_sensor )

# determine whether boiler is actually on or off
# first is it timed to be on?
wonh1 = int(won1_hrs)
wonm1 = int(won1_mins)
wonm1 = wonh1 * 60 + wonm1
woffh1 = int(woff1_hrs)
woffm1 = int(woff1_mins)
woffm1 = woffh1 * 60 + woffm1

wonh2 = int(won2_hrs)
wonm2 = int(won2_mins)
wonm2 = wonh2 * 60 + wonm2
woffh2 = int(woff2_hrs)
woffm2 = int(woff2_mins)
woffm2 = woffh2 * 60 + woffm2

honh1 = int(hon1_hrs)
honm1 = int(hon1_mins)
honm1 = honh1 * 60 + honm1
hoffh1 = int(hoff1_hrs)
hoffm1 = int(hoff1_mins)
hoffm1 = hoffh1 * 60 + hoffm1

honh2 = int(hon2_hrs)
honm2 = int(hon2_mins)
honm2 = honh2 * 60 + honm2
hoffh2 = int(hoff2_hrs)
hoffm2 = int(hoff2_mins)
hoffm2 = hoffh2 * 60 + hoffm2

if (honm1 <= tim_int < hoffm1) or (honm2 <= tim_int < hoffm2):
    heat = True
else:
    heat = False
if (wonm1 <= tim_int < woffm1) or (wonm2 <= tim_int < woffm2):
    water = True
else:
    water = False

# next what about advance status?
if a_status_int == 1 or a_status_int == 3:
    heat = not heat

```

```

if a_status_int == 2 or a_status_int == 3:
    water = not water

# finally what about the over ride switches?
if w_mode_str == "O":
    water = False

if w_mode_str == "C":
    water = True

if h_mode_str == "O":
    heat = False

if h_mode_str == "C":
    heat = True

if heat and temperature < float(thermo_setting):
    bheat_but.config(bg = "orange red", fg = "orange red")
if heat and temperature >= float(thermo_setting):
    bheat_but.config(bg = "sandy brown", fg = "sandy brown")
if not heat:
    bheat_but.config(bg = "black", fg = "black")

if water:
    bwater_but.config(bg = "orange red", fg = "orange red")
else:
    bwater_but.config(bg = "black", fg = "black")

# end of boiler on off question!

#i = datetime.now()
today = datetime.date.today()
tt = today.timetuple()
dayof_week = tt.tm_wday
month_num = tt.tm_mon
month_num = month_num - 1
dayof_mon = tt.tm_mday
hour = tt.tm_hour
mins = tt.tm_min
year = tt.tm_year
year = year % 2000

day = ("Mon","Tue","Wed","Thu","Fri","Sat","Sun")
#dayofweek_opt_men.config(var2.set(day[dayof_week]))
month_names = ("Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec")

lbl_feedback_3.config(text = "Comp time " + time.strftime("%H:%M"))

refresh_but = Button(app, text = "Refresh", font = ("Arial", 16), fg = "black", bg = "pink", command = get_data_remote)
refresh_but.grid(row = 13, column = 1, columnspan = 2, sticky = W)

#-----

# option menues to set minutes and hours
var6 = StringVar(app)
var6.set("00") # initial value
hour_opt_men = OptionMenu(app,var6,"00","01","02","03","04","05","06","07","08","09","10",\
    "11","12","13","14","15","16","17","18","19","20","21","22",\
    "23")
hour_opt_men.config(font = ("Arial", 14), fg = "black", bg = "cornflower blue", width = 2)
hour_opt_men.grid(row = 10, column = 12, columnspan = 2, sticky = W)

# minutes only multiples of 5 as 60 options do not fit on the Pi screen!
var7 = StringVar(app)
var7.set("00") # initial value
min_opt_men = OptionMenu(app,var7,"00","05","10","15","20","25","30","35","40","45","50","55")
min_opt_men.config(font = ("Arial", 14), fg = "black", bg = "cornflower blue", width = 2)
min_opt_men.grid(row = 10, column = 14, columnspan = 2, sticky = W)

lbl_time_set = Label(app, text = "Time Set hh:mm", font = ( "Arial ", 16), fg = "black", bg = "teal")
lbl_time_set.grid(row = 9, column = 12, columnspan = 6, sticky = W)

def upd_time():
    mins_lab.config(text = var7.get())

```

```

hour_lab.config(text = var6.get())
time_but.config(text = var6.get() + ":" + var7.get())

def update_hon1():
    heat_on1_hour_label.config(text = var6.get())
    heat_on1_mins_label.config(text = var7.get())

def update_hoff1():
    heat_off1_hour_label.config(text = var6.get())
    heat_off1_mins_label.config(text = var7.get())

def update_hon2():
    heat_on2_hour_label.config(text = var6.get())
    heat_on2_mins_label.config(text = var7.get())

def update_hoff2():
    heat_off2_hour_label.config(text = var6.get())
    heat_off2_mins_label.config(text = var7.get())

def update_won1():
    water_on1_hour_label.config(text = var6.get())
    water_on1_mins_label.config(text = var7.get())

def update_woff1():
    water_off1_hour_label.config(text = var6.get())
    water_off1_mins_label.config(text = var7.get())

def update_won2():
    water_on2_hour_label.config(text = var6.get())
    water_on2_mins_label.config(text = var7.get())

def update_woff2():
    water_off2_hour_label.config(text = var6.get())
    water_off2_mins_label.config(text = var7.get())

#update heat times from time entry
update_hon1_but = Button(app, text = "U", font = ("Arial", 16), fg = "black", bg = "cornflower blue", command = update_hon1)
update_hon1_but.grid(row = 3, column = 10, columnspan = 1, sticky = W)

update_hoff1_but = Button(app, text = "U", font = ("Arial", 16), fg = "black", bg = "cornflower blue", command = update_hoff1)
update_hoff1_but.grid(row = 4, column = 10, columnspan = 1, sticky = W)

update_hon2_but = Button(app, text = "U", font = ("Arial", 16), fg = "black", bg = "cornflower blue", command = update_hon2)
update_hon2_but.grid(row = 5, column = 10, columnspan = 1, sticky = W)

update_hoff2_but = Button(app, text = "U", font = ("Arial", 16), fg = "black", bg = "cornflower blue", command = update_hoff2)
update_hoff2_but.grid(row = 6, column = 10, columnspan = 1, sticky = W)

#update water times from time entry
update_won1_but = Button(app, text = "U", font = ("Arial", 16), fg = "black", bg = "cornflower blue", command = update_won1)
update_won1_but.grid(row = 3, column = 15, columnspan = 1, sticky = W)

update_woff1_but = Button(app, text = "U", font = ("Arial", 16), fg = "black", bg = "cornflower blue", command = update_woff1)
update_woff1_but.grid(row = 4, column = 15, columnspan = 1, sticky = W)

update_won2_but = Button(app, text = "U", font = ("Arial", 16), fg = "black", bg = "cornflower blue", command = update_won2)
update_won2_but.grid(row = 5, column = 15, columnspan = 1, sticky = W)

update_woff2_but = Button(app, text = "U", font = ("Arial", 16), fg = "black", bg = "cornflower blue", command = update_woff2)
update_woff2_but.grid(row = 6, column = 15, columnspan = 1, sticky = W)

#-----
# feedback label 1
lbl_feedback = Label(app, text = "", font = ("Arial Bold", 13), fg = "black", bg = "light sea green")
lbl_feedback.grid(row = 10, column = 6, columnspan = 6, sticky = W)

# feedback label 2
lbl_feedback_2 = Label(app, text = "", font = ("Arial Bold", 8), fg = "black", bg = "light sea green")
lbl_feedback_2.grid(row = 13, column = 6, columnspan = 13, sticky = W)

# feedback label 3
lbl_feedback_3 = Label(app, text = "", font = ("Arial Bold", 13), fg = "black", bg = "light sea green")
lbl_feedback_3.grid(row = 2, column = 4, columnspan = 6, sticky = W)

#-----
# clear feedback

```



```

def clear_feedback():
    lbl_feedback.config(text = "")
    lbl_feedback_2.config(text = "")
    lbl_feedback_3.config(text = "")

#-----
# reset button
def reset_timer():
    lbl_feedback.config(text = "")
    lbl_feedback_2.config(text = "")
    lbl_feedback_3.config(text = "")
    sock.sendto(bytes("xxxx", "utf-8"), ("192.168.1.177", 8888))
    get_data_remote()

reset_but = Button(app, text = "reset", font = ("Arial", 16), fg = "black", bg = "grey", command = reset_timer)
reset_but.grid(row = 13, column = 3, columnspan = 2, sticky = E)

#-----
# boiler on "buttons)

bheat_but = Button(app, font = ("Arial Bold", 16), bg = "white", fg = "white", text = "X")
bheat_but.grid(row = 2, column = 10, sticky = W)

bwater_but = Button(app, font = ("Arial Bold", 16), bg = "white", fg = "white", text = "X")
bwater_but.grid(row = 2, column = 15, sticky = W)

#-----
#update first immediately after prog is run
get_data_remote()

# calls a function after given time
def after(self, ms, func = None, *args):
    """call function after a given time"""

# updates screen every 10 seconds
def task():
    get_data_remote()
    clear_feedback()
    root.after(10000, task)

# calls the screen update every 10 seconds
root.after(10000, task)

#-----
# kick off the window's event-loop
root.mainloop()

```