```
##########################
# GUI to control RPI GPIO
# 3.2.18
# by Julian Rogers
# feather_reset_etc8
##########################


IP_FEATHER4 = "xxx.xxx.xxx.xxx" #remote ip address

DEST_PORT_FEATHER4= xxxx

THIS_PORT = 5000     #port on this computer
BACKGROUND = "gray"
BUTTON_1 = "light yellow"       #buttons: advance, load, refresh
BUTTON_2 =  "cornflower blue"     #update buttons

global watchdog
watchdog = 0
global count1
count1 = 0
global count2
count2 = 0
global commnd
commnd = ""

from tkinter import *   #GUI

import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(4, GPIO.OUT)
GPIO.setup(17, GPIO.OUT)
GPIO.setup(27, GPIO.OUT)
GPIO.setup(22, GPIO.OUT)
GPIO.setup(26, GPIO.IN)

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

# create the root window
root = Tk()

# modify the window
root.title("Greenhouse Feather4 reset etc")
root.geometry("700x430")
root.configure(bg = BACKGROUND)
#root.attributes('-fullscreen', True)   #eliminates the title bar

# the following sets the window full screen
#w, h = root.winfo_screenwidth(), root.winfo_screenheight()
#root.geometry("%dx%d+0+0" % (w, h))

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

title_lab = Label(app, text = "Feather Supervisor", font = ("Arial Bold", 20), fg = "maroon", bg = "gray")
title_lab.grid(row = 1, column = 1, columnspan = 8)

blank_lab = Label(app, bg = "gray")
blank_lab.grid(row = 2, column = 1)

sys_1_but = Button(app, text = "GPIO(04)", font = ("Arial", 12), fg = "maroon", bg = "light blue")
sys_1_but.grid(row = 3, column = 1)

def on_off_1():
    but_col = sys_1_but.config('bg')[-1]
    if but_col == "light blue":
        but_col = "yellow"
        GPIO.output(4, True)
    else:
        but_col = "light blue"
        GPIO.output(4, False)

    sys_1_but.config(bg = but_col)


sys_1_but.config(command = on_off_1)


sys_2_but = Button(app, text = "GPIO(17)", font = ("Arial", 12), fg = "maroon", bg = "light blue")
sys_2_but.grid(row = 3, column = 2)

def on_off_2():
    but_col = sys_2_but.config('bg')[-1]
    if but_col == "light blue":
        but_col = "yellow"
        GPIO.output(17, True)
    else:
        but_col = "light blue"
        GPIO.output(17, False)

    sys_2_but.config(bg = but_col)


sys_2_but.config(command = on_off_2)


sys_3_but = Button(app, text = "Reset", font = ("Arial", 16), fg = "maroon", bg = "pink")
sys_3_but.grid(row = 10, column = 1)




def reset_feather():
    GPIO.output(27, True)
    sleep(0.1)
    GPIO.output(27, False)




sys_3_but.config(command = reset_feather)



blank_lab1 = Label(app, text = "", fg = BACKGROUND, bg = BACKGROUND)
blank_lab1.grid(row = 4, column = 1)

blank_lab2 = Label(app, text = "", fg = BACKGROUND, bg = BACKGROUND)
blank_lab2.grid(row = 9, column = 1)

ind_lab = Label(app, text = "  Running indicator  ", font = ("Arial", 16), fg = "maroon", bg = BACKGROUND)
ind_lab.grid(row = 5, column = 1)

run_ind = Button(app, text = "X", font = ("Arial", 10), fg = "white", bg = "white")
run_ind.grid(row = 5, column = 2)

watchdog_but = Button(app, text =" ", font = ("Arial", 10), fg = "maroon", bg = "white")
watchdog_but.grid(row = 5, column = 3)

time_lab = Label(app, text =" ", font = ("Arial", 16), fg = "maroon", bg = BACKGROUND)
time_lab.grid(row = 6, column = 1)

system_lab0 = Label(app, text ="Remote data: ", font = ("Arial", 16), fg = "maroon", bg = BACKGROUND)
system_lab0.grid(row = 7, column = 1)

system_lab = Label(app, text ="", font = ("Arial", 16), fg = "maroon", bg = BACKGROUND)
system_lab.grid(row = 7, column = 2)

#hour_lab = Label(app, text ="", font = ("Arial", 16), fg = "maroon", bg = BACKGROUND)
#hour_lab.grid(row = 8, column = 1)
time_but = Button(app, text ="", font = ("Arial", 16), fg = "maroon", bg = BACKGROUND)
time_but.grid(row = 7, column = 3)

auto_reset_but = Button(app, text ="Auto Reset", font = ("Arial", 16), fg = "green", bg = "white")
auto_reset_but.grid(row = 8, column = 1)

blank_lab3 = Label(app, text = "", fg = BACKGROUND, bg = BACKGROUND)
blank_lab3.grid(row = 11, column = 1)

text_lab = Label(app, text ="Command entry ", font = ("Arial", 16), fg = "maroon", bg = BACKGROUND)
text_lab.grid(row = 12 , column = 1)

entry1 = Entry(app, font = ("Arial", 16), fg = "black", bg = "white")
entry1.grid(row = 12, column = 2)

feedback_lab = Label(app, text ="", font = ("Arial", 16), fg = "blue", bg = BACKGROUND)
feedback_lab.grid(row = 13, column = 2)

conf_but = Button(app, text ="Confirm", font = ("Arial", 16), fg = "blue", bg = BACKGROUND)
conf_but.grid(row = 12, column = 3)

fb_lab = Label(app, text ="Last command: ", font = ("Arial", 16), fg = "maroon", bg = BACKGROUND)
fb_lab.grid(row = 13, column = 1)

test_lab = Label(app, text ="", font = ("Arial", 16), fg = "maroon", bg = BACKGROUND)
test_lab.grid(row = 14, column = 1)

def confirm():
    global commnd
    commnd = entry1.get()

conf_but.config(command = confirm)

def auto_toggle():
    but_col = auto_reset_but.config('bg')[-1]
    if but_col == "white":
        but_col = "red"
    else:
        but_col = "white"

    auto_reset_but.config(bg = but_col)
auto_reset_but.config(command = auto_toggle)


def get_data_remote():




    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    try:
        sock.sendto(bytes("r", "utf-8") , (IP_FEATHER4, DEST_PORT_FEATHER4))

        sock.settimeout(5)

        response = sock.recv(100)

    except socket.error:
        system_lab.config(text = "timed out!")
        response = bytes("----T---/", "utf-8")

    sock.close()

    response = str(response, "utf-8")
    system_lab.config(text = response)
    tim, other_data1 = response.split("T")

    try:
        tim_int = int(tim)

    except ValueError:
        tim_int = 0

    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)

    #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")

    #time_lab.config(text = "Comp time " + time.strftime('%H:%M'))



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

# updates screen every 1 seconds
def task():

    global commnd
    if commnd == "switch" or commnd=="heaton" or commnd=="heatoff" or commnd=="staton" or commnd=="statoff":
        feedback_lab.config(text = commnd)

        sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        try:
            sock.sendto(bytes(commnd, "utf-8") , (IP_FEATHER4, DEST_PORT_FEATHER4))

            sock.settimeout(5)

            response = sock.recv(100)
            entry1.delete(0,END)
            commnd = ""

        except socket.error:
            system_lab.config(text = "timed out!")
        sock.close()

    elif len(commnd) == 3:
        try:
            int(commnd)

            feedback_lab.config(text = commnd)
            sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
            try:
                sock.sendto(bytes(commnd, "utf-8") , (IP_FEATHER4, DEST_PORT_FEATHER4))

                sock.settimeout(5)

                response = sock.recv(100)
                entry1.delete(0,END)
                commnd = ""

            except socket.error:
                system_lab.config(text = "timed out!")


            sock.close()

        except  ValueError:
            entry1.delete(0,END)
            commnd = ""
            feedback_lab.config(text = "error")


    #feedback_lab.config(text = commnd)



    global watchdog
    global count1
    global count2

    count2 = count2 + 1

    if count2 == 30:
        get_data_remote()
        count2 = 0

    #clear_feedback()

    ind = GPIO.input(26)
    if ind == 1:
        watchdog = watchdog + 1
        run_ind.config(fg = "red", bg = "red")
    else:
        run_ind.config(fg = "white", bg = "white")
        watchdog = watchdog - 1
    count1 = count1 + 1
    if count1 == 30:
        count1 = 0

        if watchdog < 15:
            watchdog_but.config(text = "OK")

        else:
            watchdog_but.config(text = "FAIL")
            if auto_reset_but.config('bg')[-1] == "red":
                reset_feather()


        watchdog = 0

    time_lab.config(text = "Comp time " + time.strftime('%H:%M'))
    root.after(1000, task)


# calls the screen update every 1 seconds
root.after(1000, task)

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