**Código fuente Módulo Concentrador y Procesador**

**Contenido**

[Módulo de Cómputo Sensible al Contexto (ventana principal) 2](#_Toc532490909)

[Módulo de Adquisición de Variables, MAV (ventana secundaria) 16](#_Toc532490910)

[Adquisición de variables (usado en MAV) 26](#_Toc532490911)

[Conexión para la adquisición de variables 30](#_Toc532490912)

[Conexión con Firebase 31](#_Toc532490913)

[Enviar variables a plataforma FitoSmart 33](#_Toc532490914)

[Datos para envío a FitoSmart 39](#_Toc532490915)

[Información WEB 40](#_Toc532490916)

[Geolocalización 41](#_Toc532490917)

[Información del clima 42](#_Toc532490918)

[Lectura de datos de configuración 42](#_Toc532490919)

[Datos de configuración 43](#_Toc532490920)

[Conectar antena externa a red WiFi 43](#_Toc532490921)

# Módulo de Cómputo Sensible al Contexto (ventana principal)

#!/usr/bin/env python

from tkinter import \*

import tkMessageBox

import Tkinter as tk

import re

from datetime import datetime as dtime

from datetime import timedelta

from threading import Thread

from time import sleep

##si habilito estas siguientes opciones no controlo bien la picamera att omar:

import sys

sys.path.append('/home/pi/Desktop/fitotron\_code/mav')

sys.path.append('/home/pi/Desktop/fitotron\_code/firebase')

sys.path.append('/home/pi/Desktop/fitotron\_code/send-data')

from FitoGeolocation import \*

from FitoWebData import \*

from FitoWeather import \*

from FitoConfig import \*

from FitoFirebase import \*

from FitoMavGetVars import \*

from FitoSendData import \*

from dataForSend import \*

##deshabilite estas opciones para manejar bien la picamera att omar

#from FitoSendData import \*

#from FitoMai import \*

#from Fito\_Act import \*

import os

#download and install pillow:

# http://www.lfd.uci.edu/~gohlke/pythonlibs/#pillow

from PIL import Image, ImageTk

import json

class Object:

def toJSON(self):

return json.dumps(self, default=lambda o: o.\_\_dict\_\_,

sort\_keys=True, indent=4)

# Here, we are creating our class, Window, and inheriting from the Frame

# class. Frame is a class from the tkinter module. (see Lib/tkinter/\_\_init\_\_)

class Fitocsc(Frame):

# config vars

cloud\_api = "";

appname = "";

deviceId = "";

ip = "0.0.0.0";

port = 3000;

timer = 0;

upd\_hour = "00:00:00"

# device info

description = ""

address = ""

lat = 0.0

lng = 0.0

# Define settings upon initialization. Here you can specify

def \_\_init\_\_(self, master=None):

# parameters that you want to send through the Frame class.

Frame.\_\_init\_\_(self, master)

#reference to the master widget, which is the tk window

self.master = master

self.init\_config();

#with that, we want to then run init\_window, which doesn't yet exist

self.init\_window()

#Creation of init\_window

def init\_window(self):

# changing the title of our master widget

self.master.title(self.appname)

# allowing the widget to take the full space of the root window

self.pack(fill=BOTH, expand=1)

# creating a menu instance

mainmenu = Menu(self.master)

self.master.config(menu=mainmenu)

# create the file object)

fitotron = Menu(mainmenu)

# adds a command to the menu option, calling it exit, and the

# command it runs on event is client\_exit

fitotron.add\_command(label="Get Web Info", command=self.getDeviceData)

fitotron.add\_command(label="Get Context Data", command=self.getContext)

fitotron.add\_command(label="Mod Adq Imagenes", command=self.setMai)

fitotron.add\_command(label="Mod Adq Variables", command=self.setMav)

#fitotron.add\_command(label="Mod Actuadores", command=self.setAct)

fitotron.add\_command(label="Exit", command=self.client\_exit)

#added "file" to our menu

mainmenu.add\_cascade(label="File", menu=fitotron)

# create the file object)

setup = Menu(mainmenu)

# adds a command to the menu option, calling it exit, and the

mainmenu.add\_cascade(label="Setup", menu=setup)

setup.add\_command(label="Parameters", command=self.create\_parameters\_window)

setup.add\_command(label="Update hour", command=self.create\_updatehour\_window)

def init\_config(self):

myFitoConfig = FitoConfig();

myData = myFitoConfig.getConfigData();

texto = open('/home/pi/timers.txt','a')

texto = open('/home/pi/timers.txt')

tx = texto.read()

n = tx.split()

texto.close()

self.upd\_hour = n[2]

#tkMessageBox.showinfo("FitoSmart - Config", myData)

self.cloud\_api = myData['cloud\_api'];

self.appname = myData['appname'];

self.deviceid = myData['deviceid'];

self.ip = myData['ip'];

self.port = myData['port'];

self.timer = myData['timer'];

#print self.appname

def validate(self, new\_text):

if not new\_text: # the field is being cleared

self.entered\_number = 0

return True

try:

self.entered\_number = int(new\_text)

return True

except ValueError:

return False

def showImg(self):

load = Image.open("fitotron.jpg")

render = ImageTk.PhotoImage(load)

# labels can be text or images

img = Label(self, image=render)

img.image = render

img.place(x=0, y=50)

#img.grid(row=1, column=1)

def setMai(self):

os.system("python /home/pi/Desktop/fitotron\_code/mai/FitoMaiGui.py")

def setMav(self):

os.system("python /home/pi/Desktop/fitotron\_code/mav/FitoMavGui.py")

def setAct(self):

os.system("python /home/pi/Desktop/fitotron\_code/csc/FitoAct.py")

def getDeviceData(self):

myDeviceData = FitoWebData();

myDevData = myDeviceData.getDeviceData(self.cloud\_api, self.deviceid);

data = myDevData[0]

self.description = data['descripcion']

self.address = data['calle'] + ", " + data['colonia'] + " " + data['ciudad'] + " " + data['estado'] + " " + "mexico"

#self.address = data['estado'] + "+" + "mexico"

calle = data['calle'] + ", " + data['colonia']

ciudad = data['ciudad'] + " " + data['estado'] + ", " + "mexico"

lblDeviceName = Label(self, text="Id Fitotron: " + self.deviceid)

lblDeviceName.pack()

lblDeviceName = Label(self, text="Descripcion: ")

lblDeviceName.pack()

lblDeviceName = Label(self, text=self.description)

lblDeviceName.pack()

myGeolocation = FitoGeolocation();

lat, lng = myGeolocation.getGeolocation(self.address);

self.lat = lat

self.lng = lng

#tkMessageBox.showinfo("FitoSmart - Geolocation", myMsg)

lblAddress = Label(self, text= "Ubicacion: ")

lblAddress.pack()

lblAddress = Label(self, text= calle)

lblAddress.pack()

lblAddress = Label(self, text= ciudad)

lblAddress.pack()

lblAddress = Label(self, text= "Ubicacion: lat: " + str(self.lat) + " lng: " + str(self.lng))

lblAddress.pack()

def getContext(self):

#self.getDeviceData();

myWeather = FitoWeather();

myMsg, weather, wind = myWeather.getWeather(self.lat, self.lng);

#tkMessageBox.showinfo("FitoSmart - Weather", myMsg)

lblContext = Label(self, text='Datos del Contexto')

lblContext.pack()

lblContext = Label(self, text='Temperatura : ' + str(myMsg['temp']))

lblContext.pack()

lblContext = Label(self, text='Nubes : ' + str(weather[0]['description']))

lblContext.pack()

lblContext = Label(self, text='Vel. viento : ' + str(wind['speed']))

lblContext.pack()

lblContext = Label(self, text='Nivel del mar : ' + str(myMsg['sea\_level']))

lblContext.pack()

lblContext = Label(self, text='Humedad : ' + str(myMsg['humidity']))

lblContext.pack()

lblContext = Label(self, text='Presion : ' + str(myMsg['pressure']))

lblContext.pack()

def create\_parameters\_window(self):

#self.counter += 1

t = tk.Toplevel(self)

t.wm\_title("Window #%s" % "Parameters setup")

def saveParameters():

device\_config = Object()

device\_config.cloud\_api = txt\_cloud\_api.get()

device\_config.appname = txt\_appname.get()

device\_config.deviceid = txt\_deviceid.get()

device\_config.ip = txt\_ip.get()

device\_config.port = txt\_port.get()

device\_config.timer = txt\_timer.get()

url = "device.config"

with open(url, "w") as f:

f.write(device\_config.toJSON())

tkMessageBox.showinfo("FitoSmart - sendData", device\_config.toJSON())

def salir():

exit()

l = tk.Label(t, text="cloud\_api").grid(row=0)

l = tk.Label(t, text="appname").grid(row=1)

l = tk.Label(t, text="deviceid").grid(row=2)

l = tk.Label(t, text="ip").grid(row=3)

l = tk.Label(t, text="port").grid(row=4)

l = tk.Label(t, text="timer").grid(row=5)

txt\_cloud\_api = tk.Entry(t) #Crear objetos de entrada de texto

txt\_appname = tk.Entry(t)

txt\_deviceid = tk.Entry(t)

txt\_ip = tk.Entry(t)

txt\_port = tk.Entry(t)

txt\_timer = tk.Entry(t)

txt\_cloud\_api.grid(row=0, column=1) #Situar los objetos

txt\_appname.grid(row=1, column=1)

txt\_deviceid.grid(row=2, column=1)

txt\_ip.grid(row=3, column=1)

txt\_port.grid(row=4, column=1)

txt\_timer.grid(row=5, column=1)

txt\_cloud\_api.insert(END, self.cloud\_api) #Escribir los valores predefinidos en los objetos

txt\_appname.insert(END, self.appname)

txt\_deviceid.insert(END, self.deviceid)

txt\_ip.insert(END, self.ip)

txt\_port.insert(END, self.port)

txt\_timer.insert(END, self.timer)

l = tk.Button(t, text='Guardar', command=saveParameters).grid(row=6, column=0, sticky="w", pady=4)

#l = tk.Button(t, text='Salir', command=t.quit).grid(row=6, column=1, sticky="w", pady=4) #command=t.quit

def create\_updatehour\_window(self):

t = tk.Toplevel(self)

t.wm\_title("Window #%s" % "Hour setup")

def validHour(dataH):

time\_re = re.compile(r'^(?:[01]?\d|2[0-3]):[0-5]\d:[0-5]\d$')

return time\_re.search(dataH)

def set\_hour():

tempH = txt\_updatehour.get()

if not validHour(tempH):

l = tk.Label(t, text="Use HH:MM:SS 24hrs").grid(row=1, column=1)

else:

texto = open('/home/pi/timers.txt','r')

tx = texto.read()

n = tx.split()

n[2] = tempH

texto = open('/home/pi/timers.txt','w')

for w in n:

texto.write(w+'\n')

texto.close()

l = tk.Label(t, text=" Saved, reboot app ").grid(row=1, column=1)

print "Nueva hora de actualizacion: ", tempH

print "Reiniciar para aplicar cambio"

l = tk.Label(t, text="Every day at:").grid(row=0, column=0)

txt\_updatehour = tk.Entry(t) #Crear objetos de entrada de texto

txt\_updatehour.grid(row=0, column=1) #Situar los objetos

txt\_updatehour.insert(END, self.upd\_hour) #Escribir el valor predefinido

l = tk.Button(t, text='Save', command=set\_hour).grid(row=1, column=0, sticky="w", pady=4)

def client\_exit(self):

#Cancelar temporizadores antes de cerrar ventana

texto = open('/home/pi/timers.txt','r')

tx = texto.read()

n = tx.split()

n[0] = "0"

n[1] = "0"

n[3] = "0"

n[4] = "0"

n[5] = "0"

texto = open('/home/pi/timers.txt','w')

for w in n:

texto.write(w+'\n')

texto.close()

print "Cerrando..."

sleep(7)

root.destroy()

#============================ Temporizadores =============================#

#Temporizador para subir datos a Firebase y FitoSmart

class Temporizador\_1(Thread):

def \_\_init\_\_(self, hora, delay, funcion):

super(Temporizador\_1, self).\_\_init\_\_()

self.\_estado = True

self.hora = hora

self.delay = delay

self.act = funcion

def stop(self):

self.\_estado = False

def run(self):

aux = dtime.strptime(self.hora, '%H:%M:%S')

hora = dtime.now()

hora = hora.replace(hour = aux.hour, minute=aux.minute, second=aux.second, microsecond=0)

if hora <= dtime.now():

hora += timedelta(days=1) #days=1, seconds

horaP = hora - timedelta(seconds=14) #hora para pausar el Timer2

pausa = False

print "Ejecutando actualizacion automatica"

print 'Proxima ejecucion programada el {0} a las {1}'.format(hora.date(), hora.time())

texto = open('/home/pi/timers.txt','r')

tx = texto.read()

n = tx.split()

texto.close()

if n[6] == '0':

print "Sin Fitotron asignado para actualizacion automatica"

else:

if n[6] == "12": fit = "1"

if n[6] == "13": fit = "2"

if n[6] == "14": fit = "3"

print "Fitotron",fit,"asignado para actualizacion automatica"

while self.\_estado:

# Leer timers.txt para apagar o pausar el Temporizador 1

texto = open('/home/pi/timers.txt','r')

tx = texto.read()

n = tx.split()

texto.close()

if n[0] == "0":

self.stop()

break

# Pausar la carga de instrucciones 14 seg antes del envio automatico

if horaP <= dtime.now() and pausa == False:

if n[3] == '0' and not n[6] == '0':

print "Pausando carga de instrucciones"

# Pausa momentaneamente a Timer2

texto = open('/home/pi/timers.txt','r')

tx = texto.read()

n = tx.split()

rp = n[4] #respaldo del estado anterior

n[4] == '1'

texto = open('/home/pi/timers.txt','w')

for w in n:

texto.write(w+'\n')

texto.close()

pausa = True

# Si se ejecuta se suma un dia a la fecha objetivo

if hora <= dtime.now():

if n[3] == '0' and not n[6] == '0':

self.act(n[6])

print 'Actualizacion ejecutada el {0} a las {1}'.format(hora.date(), hora.time())

#hora += timedelta(days=1)

#print 'Proxima atualizacion programada el {0} a las {1}'.format(hora.date(), hora.time())

#horaP = hora - timedelta(seconds=14) #hora para pausar el Timer2

texto = open('/home/pi/timers.txt','r')

tx = texto.read()

n = tx.split()

n[4] == rp

texto = open('/home/pi/timers.txt','w')

for w in n:

texto.write(w+'\n')

texto.close()

print 'Estado anterior de carga de instrucciones restaurado'

pausa = False

hora += timedelta(days=1)

print 'Proxima atualizacion programada el {0} a las {1}'.format(hora.date(), hora.time())

horaP = hora - timedelta(seconds=14) #hora para pausar el Timer2

if n[3] == '1': print 'Actualizacion automatica pausada'

if n[6] == '0': print "Fitotron no asignado"

sleep(self.delay)

if self.\_estado == False:

print 'Cancelando actualizacion de variables'

#Temporizador para leer instrucciones de Firebase

class Temporizador\_2(Thread):

segA = 0

def \_\_init\_\_(self, delay, funcion):

super(Temporizador\_2, self).\_\_init\_\_()

self.\_estado = True

self.delay = delay

self.funcion = funcion

def stop(self):

self.\_estado = False

def run(self):

tconf = FitoConfig()

paused = False

first = True

print "Ejecutando carga de instrucciones"

#print "cada", self.delay, "seg."

print "Vinculando actuadores con Firebase, espere..."

while self.\_estado:

# Leer timers.txt para apagar o pausar el Temporizador 2

texto = open('/home/pi/timers.txt','r')

tx = texto.read()

n = tx.split()

texto.close()

if n[1] == "0":

self.stop()

break

if n[4] == "0":

self.funcion()

if paused == True:

print "Carga de instrucciones ejecutandose"

paused = False

if first == True:

print "Vinculado a Firebase, hecho"

first = False

if n[4] == "2":

if paused == False: print "Carga de instrucciones pausada"

paused = True

sleep(self.delay)

if self.\_estado == False:

print 'Cancelando carga de instrucciones'

#========================================================================#

###=================================================================================###

fire = FitoFirebase()

send = FitoSendData()

datasend = dataForSend()

mav = FitoMavGetVars()

#Funcion para apagar temporizadores si se cierra la ventana desde el icono x

def cerrando():

#Cancelar temporizadores antes de cerrar ventana

texto = open('/home/pi/timers.txt','r')

tx = texto.read()

n = tx.split()

n[0] = "0"

n[1] = "0"

n[3] = "0"

n[4] = "0"

n[5] = "0"

texto = open('/home/pi/timers.txt','w')

for w in n:

texto.write(w+'\n')

texto.close()

print "Cerrando..."

sleep(7)

root.destroy()

#Funciones que seran automaticas usando los temporizadores

def subirDatosFitotron(nip):

if nip == "12": fit = "1"

if nip == "13": fit = "2"

if nip == "14": fit = "3"

# Solicitar variables al Fitotron

print "Solicitando variables a los modulos"

va = mav.getTotalVars(nip)

# Subir a firebase

print "Actualizar en Firebase..."

rf = fire.sendToFirebase(va, "s", nip)

print rf

# Subir a FitoSmart

print "Actualizando variables en FitoSmart..."

rn = send.sendData2(va, fit)

print "Hecho"

# Datos respaldo instrucciones

# NOTA: Por ahora, todo lo relacionado al Rele del Modulo Monitoreo Nutriente NO se

#esta usando. Aun no cuenta con su instruccion en Firebase.

MR = [-1,-1,-1,-1,-1,-1,-1] #CN/Re1, CN/Re2, CN/Re3, MN/Re, MA1/Re, MA2/Re, MA3/Re

#first = True #Primera carga, actualizar modulos con datos de Firebase

def leerInstFirebase():

#print "Estado bombas(1,2,3) y luces(1,2,3): ",MR[0],MR[1],MR[2],MR[4],MR[5],MR[6]

#ejB = False #Modulo Control Nutriente

ejA1 = False #Modulo Monitoreo Ambiente 1

ejA2 = False #Modulo Monitoreo Ambiente 2

ejA3 = False #Modulo Monitoreo Ambiente 2

#ejN = False #Modulo Monitoreo Nutriente

# Primero leer y guardar instrucciones de firebase

resp = fire.getFromFirebase()

#print "Lectura firebase:",resp.b1, resp.b2, resp.b3, resp.i1, resp.i2, resp.i3

# Leer txt

texto = open("/home/pi/timers.txt","r")

tx = texto.read()

n = tx.split()

texto.close()

########### Bombas individuales ############

def bombaI():

if not resp.b1 == MR[0]:

if resp.b1 == 1: sw = "ON"

else: sw = "OFF"

mav.bombas("R1",sw)

ejB = True

MR[0] = resp.b1

if not resp.b2 == MR[1]:

if resp.b2 == 1: sw = "ON"

else: sw = "OFF"

mav.bombas("R2",sw)

ejB = True

MR[1] = resp.b2

if not resp.b3 == MR[2]:

if resp.b3 == 1: sw = "ON"

else: sw = "OFF"

mav.bombas("R3",sw)

ejB = True

MR[2] = resp.b3

# Si se realizo cambio desde MAV, solo respaldar datos del Firebase

#evitando que se envien instrucciones repetidas

if n[5] == "1":

#print "Ejecutado desde MAV"

MR[0] = resp.b1

MR[1] = resp.b2

MR[2] = resp.b3

#MR[3] = resp.n

MR[4] = resp.i1

MR[5] = resp.i2

MR[6] = resp.i3

texto = open("/home/pi/timers.txt","r")

tx = texto.read()

n = tx.split()

n[5] = "0"

texto = open("/home/pi/timers.txt","w")

for w in n:

texto.write(w+"\n")

#### Empezar a comparar para ejecutar instrucciones ####

"""if resp.b1 == resp.b2 and resp.b1 == resp.b3:

# Bombas en conjunto

temp1 = resp.b1

if MR[0] == MR[1] and MR[0] == MR[2]:

temp2 = MR[0]

if not temp1 == temp2:

if temp1 == 1: sw = "ON"

else: sw = "OFF"

mav.bombas("RT",sw)

ejB = True

# Respaldar estados actuales ya modificados para siguiente lectura

MR[0] = temp1

MR[1] = temp1

MR[2] = temp1

else:

# Bombas por separado

bombaI()

else:

# Bombas por separado

bombaI()"""

######## Rele Monitoreo Nutriente ########

"""if not resp.n == MR[3]:

if resp.n == 1: sw = "ON"

else: sw = "OFF"

L1 = mav.MN\_con(sw)

MR[3] = resp.n

ejN = True"""

########### Iluminacion ###########

## Monitoreo Ambiente 1 (12) ##

"""if not resp.i1 == MR[4]:

if resp.i1 == 1: sw = "ON"

else: sw = "OFF"

L1 = mav.MA\_con("12",sw)

MR[4] = resp.i1

ejA1 = True

## Monitoreo Ambiente 2 (13) ##

if not resp.i2 == MR[5]:

if resp.i2 == 1: sw = "ON"

else: sw = "OFF"

L2 = mav.MA\_con("13",sw)

MR[5] = resp.i2

ejA2 = True"""

## Monitoreo Ambiente 3 (14) ##

if not resp.i3 == MR[6]:

if resp.i3 == 1: sw = "ON"

else: sw = "OFF"

L3 = mav.MA\_con("14",sw)

MR[6] = resp.i3

ejA3 = True

### Actualizar en FitoSmart y/o Firebase###

#if ejB == True:

#Bo = Object()

#Bo.R1 = MR[0]

#Bo.R2 = MR[1]

#Bo.R3 = MR[2]

#resp = send.sendDataB(Bo)

#print "Datos actualizados en FitoSmart"

#if ejN == True:

# Aqui actualizar en Firebase lo correspondiente al rele de

#Modulo Monitoreo Nutriente

if ejA1 == True:

r = fire.sendEspFirebase("/Modulo\_Monitoreo\_Ambiente\_1","Re",L1)

if ejA2 == True:

r = fire.sendEspFirebase("/Modulo\_Monitoreo\_Ambiente\_2","Re",L2)

if ejA3 == True:

r = fire.sendEspFirebase("/Modulo\_Monitoreo\_Ambiente\_3","Re",L3)

horapred = "00:00:00" #Hora predefinida para subir datos

# Al iniciar modulo central, crear archivo para temporizadores

texto = open('/home/pi/timers.txt','a')

texto = open('/home/pi/timers.txt')

tx = texto.read()

n = tx.split()

if len(n) == 0 or len(n) < 7:

#Si no hay datos o no existe el archivo, crear nuevo con datos predefinidos

#Datos: Apagar Timer1, apagar Timer2, Hora de ejecucion Timer1, pausar Timer1, pausar Timer2, Aviso de MAV, Modulo Ambiente para Timer1

texto = open('/home/pi/timers.txt','w')

texto.write('1\n1\n'+horapred+'\n0\n0\n0\n0')

horatemp = horapred

texto.close()

else:

#Si ya tiene datos:

horatemp = n[2] #Hora de ejecucion del Temporizador\_1

texto = open('/home/pi/timers.txt','w')

texto.write('1\n1\n'+horatemp+'\n0\n0\n0\n'+n[6])

texto.close()

t1 = Temporizador\_1(horatemp, 1, subirDatosFitotron)

t2 = Temporizador\_2(2, leerInstFirebase)

t1.start()

sleep(1)

#t2.start()

sleep(1)

# root window created. Here, that would be the only window, but

# you can later have windows within windows.

root = Tk()

root.geometry("460x300")

#creation of an instance

app = Fitocsc(root)

#cuando se cierre la ventada desde el icono x, se ejecuta la funcion cerrando

root.protocol("WM\_DELETE\_WINDOW", cerrando)

#mainloop

root.mainloop()

# Módulo de Adquisición de Variables, MAV (ventana secundaria)

# -\*- coding: latin-1 -\*-

from tkinter import \*

from datetime import datetime as dtime

import tkMessageBox

import Tkinter as tk

import sys

sys.path.append('/home/pi/Desktop/fitotron\_code/csc')

sys.path.append('/home/pi/Desktop/fitotron\_code/send-data')

sys.path.append('/home/pi/Desktop/fitotron\_code/firebase')

from FitoConfig import \*

from FitoGeolocation import \*

from FitoWebData import \*

from FitoMavGetVars import \*

from FitoSendData import \*

from FitoFirebase import \*

from PIL import Image, ImageTk

import json

class Object:

def toJSON(self):

return json.dumps(self, default=lambda o: o.\_\_dict\_\_,

sort\_keys=True, indent=4)

# Here, we are creating our class, Window, and inheriting from the Frame

# class. Frame is a class from the tkinter module. (see Lib/tkinter/\_\_init\_\_)

class FitoMavGui(Frame):

# config vars

cloud\_api = "";

appname = "";

deviceId = "";

ip = "0.0.0.0";

port = 0;

timer = 0;

lat = 0

lng = 0

ipcsc = '0.0.0.0'

# device info

#codebar = 0

#url\_image0 = ""

#url\_image1 = ""

###

mav = FitoMavGetVars() #Objeto clase FitoMavGetVars()

enviar = FitoSendData() #Objeto clase FitoSendData()

fireb = FitoFirebase() #Objeto clase FitoFirebase()

# Define settings upon initialization. Here you can specify

def \_\_init\_\_(self, master=None):

# parameters that you want to send through the Frame class.

Frame.\_\_init\_\_(self, master)

#reference to the master widget, which is the tk window

self.master = master

self.init\_config();

#with that, we want to then run init\_window, which doesn't yet exist

self.init\_window()

#Creation of init\_window

def init\_window(self):

# changing the title of our master widget

self.master.title("MAV - Modulo de adquisicion de Variables")

self.entrytext = StringVar()

self.entrytext1 = StringVar()

self.myCmdCodebar = Label(self.master, text="Control").grid(row=0,column=0)

self.myCmdCodebar = Label(self.master, text="Nivel").grid(row=0,column=1)

self.myCmdCodebar = Label(self.master, text="Nutriente").grid(row=0,column=2)

self.myCmdCodebar = Label(self.master, text="Ambiente 1").grid(row=0,column=3)

self.myCmdCodebar = Label(self.master, text="Ambiente 2").grid(row=0,column=4)

self.myCmdCodebar = Label(self.master, text="Ambiente 3").grid(row=0,column=5)

########################## MCN

self.myCmdSend = Button(self.master, text="B1 ON", command=self.Bomba1on, height = 1, width = 4) #h=2,w=10

self.myCmdSend.grid(row=1,column=0)

self.myCmdSend.configure(state = NORMAL)

self.myCmdSend = Button(self.master, text="B1 OFF", command=self.Bomba1off, height = 1, width = 4)

self.myCmdSend.grid(row=1,column=1)

self.myCmdSend.configure(state = NORMAL)

self.myCmdSend = Button(self.master, text="B2 ON", command=self.Bomba2on, height = 1, width = 4)

self.myCmdSend.grid(row=2,column=0)

self.myCmdSend.configure(state = NORMAL)

self.myCmdSend = Button(self.master, text="B2 OFF", command=self.Bomba2off, height = 1, width = 4)

self.myCmdSend.grid(row=2,column=1)

self.myCmdSend.configure(state = NORMAL)

self.myCmdSend = Button(self.master, text="B3 ON", command=self.Bomba3on, height = 1, width = 4)

self.myCmdSend.grid(row=3,column=0)

self.myCmdSend.configure(state = NORMAL)

self.myCmdSend = Button(self.master, text="B3 OFF", command=self.Bomba3off, height = 1, width = 4)

self.myCmdSend.grid(row=3,column=1)

self.myCmdSend.configure(state = NORMAL)

self.myCmdSend = Button(self.master, text="Bs ON", command=self.BombasTon, height = 1, width = 4)

self.myCmdSend.grid(row=4,column=0)

self.myCmdSend.configure(state = NORMAL)

self.myCmdSend = Button(self.master, text="Bs OFF", command=self.BombasToff, height = 1, width = 4)

self.myCmdSend.grid(row=4,column=1)

self.myCmdSend.configure(state = NORMAL)

self.myCmdSend = Button(self.master, text="Nivel", command=self.NivelLiq, height = 1, width = 4)

self.myCmdSend.grid(row=5,column=0)

self.myCmdSend.configure(state = NORMAL)

################################## MMN

self.myCmdSend = Button(self.master, text="ON", command=self.NutOn, height = 1, width = 4)

self.myCmdSend.grid(row=1,column=2)

self.myCmdSend.configure(state = NORMAL)

self.myCmdSend = Button(self.master, text="OFF", command=self.NutOff, height = 1, width = 4)

self.myCmdSend.grid(row=2,column=2)

self.myCmdSend.configure(state = NORMAL)

self.myCmdSend = Button(self.master, text="Valores", command=self.NutVar, height = 1, width = 4)

self.myCmdSend.grid(row=3,column=2)

self.myCmdSend.configure(state = NORMAL)

######################## MMA 1

self.myCmdSend = Button(self.master, text="ON", command=self.Amb1On, height = 1, width = 8)

self.myCmdSend.grid(row=1,column=3)

self.myCmdSend.configure(state = NORMAL)

self.myCmdSend = Button(self.master, text="OFF", command=self.Amb1Off, height = 1, width = 8)

self.myCmdSend.grid(row=2,column=3)

self.myCmdSend.configure(state = NORMAL)

self.myCmdSend = Button(self.master, text="Valores", command=self.Amb1Var, height = 1, width = 8)

self.myCmdSend.grid(row=3,column=3)

self.myCmdSend.configure(state = NORMAL)

self.myCmdSend = Button(self.master, text="Nivel", command=self.Amb1Di, height = 1, width = 8)

self.myCmdSend.grid(row=4,column=3)

self.myCmdSend.configure(state = NORMAL)

self.myCmdSend = Button(self.master, text="Fitotron 1\nEnviar datos", command=self.enviarDatos1, height = 1, width = 8)

self.myCmdSend.grid(row=5,column=3)

self.myCmdSend.configure(state = NORMAL)

self.myCmdSend = Button(self.master, text="Asignar\nFitotron 1", command=self.selFitotron1, height = 1, width = 8)

self.myCmdSend.grid(row=6,column=3)

self.myCmdSend.configure(state = NORMAL)

######################## MMA 2

self.myCmdSend = Button(self.master, text="ON", command=self.Amb2On, height = 1, width = 8)

self.myCmdSend.grid(row=1,column=4)

self.myCmdSend.configure(state = NORMAL)

self.myCmdSend = Button(self.master, text="OFF", command=self.Amb2Off, height = 1, width = 8)

self.myCmdSend.grid(row=2,column=4)

self.myCmdSend.configure(state = NORMAL)

self.myCmdSend = Button(self.master, text="Valores", command=self.Amb2Var, height = 1, width = 8)

self.myCmdSend.grid(row=3,column=4)

self.myCmdSend.configure(state = NORMAL)

self.myCmdSend = Button(self.master, text="Nivel", command=self.Amb2Di, height = 1, width = 8)

self.myCmdSend.grid(row=4,column=4)

self.myCmdSend.configure(state = NORMAL)

self.myCmdSend = Button(self.master, text="Fitotron 2\nEnviar datos", command=self.enviarDatos2, height = 1, width = 8)

self.myCmdSend.grid(row=5,column=4)

self.myCmdSend.configure(state = NORMAL)

self.myCmdSend = Button(self.master, text="Asignar\nFitotron 2", command=self.selFitotron2, height = 1, width = 8)

self.myCmdSend.grid(row=6,column=4)

self.myCmdSend.configure(state = NORMAL)

####################### MMA 3

self.myCmdSend = Button(self.master, text="ON", command=self.Amb3On, height = 1, width = 8)

self.myCmdSend.grid(row=1,column=5)

self.myCmdSend.configure(state = NORMAL)

self.myCmdSend = Button(self.master, text="OFF", command=self.Amb3Off, height = 1, width = 8)

self.myCmdSend.grid(row=2,column=5)

self.myCmdSend.configure(state = NORMAL)

self.myCmdSend = Button(self.master, text="Valores", command=self.Amb3Var, height = 1, width = 8)

self.myCmdSend.grid(row=3,column=5)

self.myCmdSend.configure(state = NORMAL)

self.myCmdSend = Button(self.master, text="Nivel", command=self.Amb3Di, height = 1, width = 8)

self.myCmdSend.grid(row=4,column=5)

self.myCmdSend.configure(state = NORMAL)

self.myCmdSend = Button(self.master, text="Fitotron 3\nEnviar datos", command=self.enviarDatos3, height = 1, width = 8)

self.myCmdSend.grid(row=5,column=5)

self.myCmdSend.configure(state = NORMAL)

self.myCmdSend = Button(self.master, text="Asignar\nFitotron 3", command=self.selFitotron3, height = 1, width = 8)

self.myCmdSend.grid(row=6,column=5)

self.myCmdSend.configure(state = NORMAL)

###################

text = open('/home/pi/timers.txt','r')

txt = text.read()

w = txt.split()

text.close()

if w[3] == '1': self.myCmdCodebar = Label(self.master, text=" Pausado ").grid(row=8,column=5) #Actualizacion

if w[4] == '2': self.myCmdCodebar = Label(self.master, text=" Pausado ").grid(row=8,column=3) #Instrucciones

self.myCmdSend = Button(self.master, text="Pausa cargar\ninstrucciones", command = self.pausaInst, height = 1, width = 8)

self.myCmdSend.grid(row=7,column=3)

self.myCmdSend.configure(state = NORMAL)

self.myCmdSend = Button(self.master, text="Pausa envio\nautomatico", command = self.pausaAut, height = 1, width = 8)

self.myCmdSend.grid(row=7,column=5)

self.myCmdSend.configure(state = NORMAL)

"""self.myCmdSend = Button(self.master, text="Detener Automatico\nprueba", command=self.detenerAut, height = 2, width = 10)

self.myCmdSend.grid(row=7,column=4)

self.myCmdSend.configure(state = NORMAL)"""

################### SALIR

self.myCmdSend = Button(self.master, text="Salir", command=self.client\_exit, height = 1, width = 4)

self.myCmdSend.grid(row=7,column=0)

self.myCmdSend.configure(state = NORMAL)

def init\_config(self):

myFitoConfig = FitoConfig();

myData = myFitoConfig.getConfigData();

#tkMessageBox.showinfo("FitoSmart - Config", myData)

self.cloud\_api = myData['cloud\_api'];

self.appname = myData['appname'];

self.deviceid = myData['deviceid'];

self.ip = myData['ip'];

self.port = myData['port'];

self.timer = myData['timer'];

print self.appname

#self.getDeviceData(); #Obtener datos con internet

def getDeviceData(self):

myDeviceData = FitoWebData();

myDevData = myDeviceData.getDeviceData(self.cloud\_api, self.deviceid);

data = myDevData[0]

self.description = data['descripcion']

self.address = data['calle'] + ", " + data['colonia'] + " " + data['ciudad'] + " " + data['estado'] + " " + "mexico"

#self.address = data['estado'] + "+" + "mexico"

myGeolocation = FitoGeolocation();

lat, lng = myGeolocation.getGeolocation(self.address);

self.lat = lat

self.lng = lng

############### Funcion de aviso ###############

def aviso(self):

texto = open("/home/pi/timers.txt","r")

tx = texto.read()

n = tx.split()

n[5] = "1"

texto = open("/home/pi/timers.txt","w")

for w in n:

texto.write(w+"\n")

texto.close()

############# Asignar Fitotron ################

def asignFitotron(self, ftn):

texto = open("/home/pi/timers.txt","r")

tx = texto.read()

n = tx.split()

n[6] = ftn

texto = open("/home/pi/timers.txt","w")

for w in n:

texto.write(w+"\n")

texto.close()

########################### MCN ############################

def Bomba1on(self):

x = self.mav.bombas("R1","ON");

resp = x.R1+"\n"+x.R2+"\n"+x.R3

self.fireb.sendEspFirebase("Modulo\_Bombas","Bomba1",1)

self.aviso()

tkMessageBox.showinfo("Respuesta: ", resp)

def Bomba1off(self):

x = self.mav.bombas("R1","OFF");

resp = x.R1+"\n"+x.R2+"\n"+x.R3

self.fireb.sendEspFirebase("Modulo\_Bombas","Bomba1",0)

self.aviso()

tkMessageBox.showinfo("Respuesta: ", resp)

def Bomba2on(self):

x = self.mav.bombas("R2","ON");

resp = x.R1+"\n"+x.R2+"\n"+x.R3

self.fireb.sendEspFirebase("Modulo\_Bombas","Bomba2",1)

self.aviso()

tkMessageBox.showinfo("Respuesta: ", resp)

def Bomba2off(self):

x = self.mav.bombas("R2","OFF");

resp = x.R1+"\n"+x.R2+"\n"+x.R3

self.fireb.sendEspFirebase("Modulo\_Bombas","Bomba2",0)

self.aviso()

tkMessageBox.showinfo("Respuesta: ", resp)

def Bomba3on(self):

x = self.mav.bombas("R3","ON");

resp = x.R1+"\n"+x.R2+"\n"+x.R3

self.fireb.sendEspFirebase("Modulo\_Bombas","Bomba3",1)

self.aviso()

tkMessageBox.showinfo("Respuesta: ", resp)

def Bomba3off(self):

x = self.mav.bombas("R3","OFF");

resp = x.R1+"\n"+x.R2+"\n"+x.R3

self.fireb.sendEspFirebase("Modulo\_Bombas","Bomba3",0)

self.aviso()

tkMessageBox.showinfo("Respuesta: ", resp)

def BombasTon(self):

x = self.mav.bombas("RT","ON");

resp = x.R1+"\n"+x.R2+"\n"+x.R3

self.fireb.sendBombFirebase(1)

self.aviso()

tkMessageBox.showinfo("Respuesta: ", resp)

def BombasToff(self):

x = self.mav.bombas("RT","OFF");

resp = x.R1+"\n"+x.R2+"\n"+x.R3

self.fireb.sendBombFirebase(0)

self.aviso()

tkMessageBox.showinfo("Respuesta: ", resp)

def NivelLiq(self):

x = self.mav.nivel();

resp = x.N1+"\n"+x.N2+"\n"+x.N3

tkMessageBox.showinfo("Respuesta: ", resp)

######################## MMN ############################

def NutVar(self):

x = self.mav.MN\_var();

resp = x.CO2+"\n"+x.PH+"\n"+x.CE+"\n"+x.TL+"\n"+x.Re

tkMessageBox.showinfo("Respuesta: ", resp)

def NutOn(self):

resp = self.mav.MN\_con("ON");

self.fireb.sendEspFirebase("Modulo\_Monitoreo\_Nutriente","Re",resp)

#aviso()

tkMessageBox.showinfo("Respuesta: ", resp)

def NutOff(self):

resp = self.mav.MN\_con("OFF");

self.fireb.sendEspFirebase("Modulo\_Monitoreo\_Nutriente","Re",resp)

#aviso()

tkMessageBox.showinfo("Respuesta: ", resp)

######################### MMA 1 ##########################

def Amb1On(self):

resp = self.mav.MA\_con("12","ON");

self.fireb.sendEspFirebase("Modulo\_iluminacion","iluminacion1",1)

self.aviso()

self.fireb.sendEspFirebase("Modulo\_Monitoreo\_Ambiente\_1","Re",resp)

tkMessageBox.showinfo("Respuesta: ", resp)

def Amb1Off(self):

resp = self.mav.MA\_con("12","OFF");

self.fireb.sendEspFirebase("Modulo\_iluminacion","iluminacion1",0)

self.aviso()

self.fireb.sendEspFirebase("Modulo\_Monitoreo\_Ambiente\_1","Re",resp)

tkMessageBox.showinfo("Respuesta: ", resp)

def Amb1Var(self):

x = self.mav.MA\_var("12");

resp = x.Ta+"\n"+x.Hr+"\n"+x.Tl+"\n"+x.I+"\n"+x.Tin+"\n"+x.Al+"\n"+x.Pr+"\n"+x.Re

tkMessageBox.showinfo("Respuesta: ", resp)

def Amb1Di(self):

resp = self.mav.MA\_dist("12");

tkMessageBox.showinfo("Respuesta: ", resp)

def enviarDatos1(self):

print "Solicitando variables a modulos"

v = self.mav.getTotalVars("12")

print "Enviando a Firebase"

self.fireb.sendToFirebase(v,"s","12")

print "Hecho"

print "Enviando a FitoSmart..."

msg = self.enviar.sendData2(v, "1")

tkMessageBox.showinfo("Respuesta: ", msg)

def selFitotron1(self):

self.asignFitotron("12")

self.fireb.sendEspFirebase("Fitotron Actual","Fitotron",1)

print "Fitotron 1 asignado para actualizacion automatica"

tkMessageBox.showinfo("Respuesta: ", "Hecho")

######################### MMA 2 ##########################

def Amb2On(self):

resp = self.mav.MA\_con("13","ON");

self.fireb.sendEspFirebase("Modulo\_iluminacion","iluminacion2",1)

self.aviso()

self.fireb.sendEspFirebase("Modulo\_Monitoreo\_Ambiente\_2","Re",resp)

tkMessageBox.showinfo("Respuesta: ", resp)

def Amb2Off(self):

resp = self.mav.MA\_con("13","OFF");

self.fireb.sendEspFirebase("Modulo\_iluminacion","iluminacion2",0)

self.aviso()

self.fireb.sendEspFirebase("Modulo\_Monitoreo\_Ambiente\_2","Re",resp)

tkMessageBox.showinfo("Respuesta: ", resp)

def Amb2Var(self):

x = self.mav.MA\_var("13");

resp = x.Ta+"\n"+x.Hr+"\n"+x.Tl+"\n"+x.I+"\n"+x.Tin+"\n"+x.Al+"\n"+x.Pr+"\n"+x.Re

tkMessageBox.showinfo("Respuesta: ", resp)

def Amb2Di(self):

resp = self.mav.MA\_dist("13");

tkMessageBox.showinfo("Respuesta: ", resp)

def enviarDatos2(self):

print "Solicitando variables a modulos"

v = self.mav.getTotalVars("13")

print "Enviando a Firebase"

self.fireb.sendToFirebase(v,"s","13")

print "Hecho"

print "Enviando a FitoSmart..."

msg = self.enviar.sendData2(v, "2")

tkMessageBox.showinfo("Respuesta: ", msg)

def selFitotron2(self):

self.asignFitotron("13")

self.fireb.sendEspFirebase("Fitotron Actual","Fitotron",2)

print "Fitotron 2 asignado para actualizacion automatica"

tkMessageBox.showinfo("Respuesta:","Hecho")

######################### MMA 3 ##########################

def Amb3On(self):

resp = self.mav.MA\_con("14","ON");

self.fireb.sendEspFirebase("Modulo\_iluminacion","iluminacion3",1)

self.aviso()

self.fireb.sendEspFirebase("Modulo\_Monitoreo\_Ambiente\_3","Re",resp)

tkMessageBox.showinfo("Respuesta: ", resp)

def Amb3Off(self):

resp = self.mav.MA\_con("14","OFF");

self.fireb.sendEspFirebase("Modulo\_iluminacion","iluminacion3",0)

self.aviso()

self.fireb.sendEspFirebase("Modulo\_Monitoreo\_Ambiente\_3","Re",resp)

tkMessageBox.showinfo("Respuesta: ", resp)

def Amb3Var(self):

x = self.mav.MA\_var("14");

resp = x.Ta+"\n"+x.Hr+"\n"+x.Tl+"\n"+x.I+"\n"+x.Tin+"\n"+x.Al+"\n"+x.Pr+"\n"+x.Re

tkMessageBox.showinfo("Respuesta: ", resp)

def Amb3Di(self):

resp = self.mav.MA\_dist("14");

tkMessageBox.showinfo("Respuesta: ", resp)

def enviarDatos3(self):

print "Solicitando variables a modulos"

v = self.mav.getTotalVars("14")

print "Enviando a Firebase"

self.fireb.sendToFirebase(v,"s","14")

print "Hecho"

print "Enviando a FitoSmart..."

msg = self.enviar.sendData2(v, "3")

tkMessageBox.showinfo("Respuesta: ", msg)

def selFitotron3(self):

self.asignFitotron("14")

self.fireb.sendEspFirebase("Fitotron Actual","Fitotron",3)

print "Fitotron 3 asignado para actualizacion automatica"

tkMessageBox.showinfo("Respuesta:","Hecho")

##################### Pausar Timer 1, envio automatico ######################

def pausaAut(self):

texto = open("/home/pi/timers.txt", "r")

tx = texto.read()

n = tx.split()

if n[3] == '0':

n[3] = '1'

self.myCmdCodebar = Label(self.master, text=" Pausado ").grid(row=8,column=5)

print "Actualizacion automatica pausada"

else:

n[3] = '0'

self.myCmdCodebar = Label(self.master, text="Ejecutando").grid(row=8,column=5)

print "Actualizacion automatica restaurada"

texto = open("/home/pi/timers.txt", "w")

for w in n:

texto.write(w+"\n")

##################### Pausar Timer 2, cargar instrucciones ######################

def pausaInst(self):

texto = open("/home/pi/timers.txt", "r")

tx = texto.read()

n = tx.split()

if n[4] == '0':

n[4] = '2'

self.myCmdCodebar = Label(self.master, text=" Pausado ").grid(row=8,column=3)

print "Pausando desde boton, espere..."

elif n[4] == '2':

n[4] = '0'

self.myCmdCodebar = Label(self.master, text="Ejecutando").grid(row=8,column=3)

print "Activando desde boton, espere..."

texto = open("/home/pi/timers.txt", "w")

for w in n:

texto.write(w+"\n")

texto.close()

#### Prueba ####

def enviarFirebase(self):

R = self.mav.getTotalVars("ALL");

resp = self.fire.sendToFirebase(R, "ALL")

tkMessageBox.showinfo("Respuesta: ", resp)

##### Cerrar ventana MAV #####

def client\_exit(self):

exit()

# root window created. Here, that would be the only window, but

# you can later have windows within windows.

root = Tk()

root.geometry("460x250") #540x440

#creation of an instance

app = FitoMavGui(root)

#mainloop

root.mainloop()

# Adquisición de variables (usado en MAV)

from FitoGetMav import \*

class Object:

def toJSON(self):

return json.dumps(self, default=lambda o: o.\_\_dict\_\_,

sort\_keys=True, indent=4)

class FitoMavGetVars:

mydata = FitoGetMav()

def ejemplo1(self):

print "Enviando instruccion"

resp = self.mydata.getDeviceData("http://192.168.0.10","CN-R1-ON")

return resp

def ejemplo2(self, mod):

cad = "Recibi: "+mod

return cad

##################### Modulo Control Nivel ####################

def bombas(self, rele, switch):

inst = "CN-"+rele+"-"+switch

resp = self.mydata.getDeviceData("http://192.168.0.10",inst)

resp = resp.replace("#","")

vecresp = resp.split("/")

vars = Object()

vars.R1 = vecresp[1]

vars.R2 = vecresp[2]

vars.R3 = vecresp[3]

return vars

def nivel(self):

resp = self.mydata.getDeviceData("http://192.168.0.10","CN-Valores")

resp = resp.replace("#","")

vecresp = resp.split("/")

vars = Object()

vars.N1 = vecresp[1]

vars.N2 = vecresp[2]

vars.N3 = vecresp[3]

return vars

################## Modulo Monitoreo Nutriente #################

def MN\_var(self):

resp = self.mydata.getDeviceData("http://192.168.0.11","MMN-Valores")

resp = resp.replace("-","")

vecresp = resp.split("/")

sce = vecresp[3]+"/"+vecresp[4]

vars = Object()

vars.CO2 = vecresp[1]

vars.PH = vecresp[2]

vars.CE = sce

vars.TL = vecresp[5]

vars.Re = vecresp[6]

return vars

def MN\_con(self, switch):

inst = "MMN-R1-"+switch

resp = self.mydata.getDeviceData("http://192.168.0.11",inst)

resp = resp.replace("-","")

vecresp = resp.split("/")

resp = vecresp[1]

return resp

################# Modulo Monitoreo Ambiente (12, 13, 14) #################

def MA\_con(self, modulo, switch):

ipadd = "http://192.168.0."+modulo #IP del modulo, terminacion 12, 13, o 14

inst = "MA-R1-"+switch

resp = self.mydata.getDeviceData(ipadd,inst)

resp = resp.replace("#","")

vecresp = resp.split("/")

vaR1 = vecresp[1]

return vaR1

def MA\_var(self, modulo):

ipadd = "http://192.168.0."+modulo

resp = self.mydata.getDeviceData(ipadd,"MA-Valores")

resp = resp.replace("#","")

vecresp = resp.split("/")

vars = Object()

vars.Ta = vecresp[1]

vars.Hr = vecresp[2]

vars.Tl = vecresp[3]

vars.I = vecresp[4]

vars.Tin = vecresp[5]

vars.Al = vecresp[6]

vars.Pr = vecresp[7]

vars.Re = vecresp[8]

return vars

def MA\_dist(self, modulo):

ipadd = "http://192.168.0."+modulo

resp = self.mydata.getDeviceData(ipadd,"MA-Nivel")

resp = resp.replace("#","")

vecresp = resp.split("/")

varN = vecresp[1]

return varN

def getTotalVars(self, Amb):

# Preparar objeto para sendData de la clase FitoSendData y/o para

#sendToFirebase de la clase FitoFirebase

# Amb: monitoreo ambiente especificado del "12" al "14"; "ALL" toma

#las variables de todos

data\_vars = Object()

data\_vars.CN = Object()

data\_vars.MN = Object()

####### Modulo Control Nivel #######

nv = self.nivel()

####### Modulo Monitoreo Nutriente #######

nut = self.MN\_var()

###### Modulo(s) Monitoreo Ambiente ######

if not Amb == "ALL":

data\_vars.MA = Object() #Objeto para un solo modulo monitoreo ambiente

Avar = self.MA\_var(Amb)

Adis = self.MA\_dist(Amb)

else:

data\_vars.MA1 = Object() #Objetos para cada modulo monitoreo ambiente

data\_vars.MA2 = Object()

data\_vars.MA3 = Object()

Avar1 = self.MA\_var("12")

Adis1 = self.MA\_dist("12")

Avar2 = self.MA\_var("13")

Adis2 = self.MA\_dist("13")

Avar3 = self.MA\_var("14")

Adis3 = self.MA\_dist("14")

#### guardar variables en data\_vars por modulos ####

data\_vars.CN.N1 = nv.N1

data\_vars.CN.N2 = nv.N2

data\_vars.CN.N3 = nv.N3

data\_vars.MN.CO2 = nut.CO2

data\_vars.MN.PH = nut.PH

data\_vars.MN.CE = nut.CE

data\_vars.MN.TL = nut.TL

data\_vars.MN.Re = nut.Re

if not Amb == "ALL":

data\_vars.MA.Ta = Avar.Ta

data\_vars.MA.Hr = Avar.Hr

data\_vars.MA.Tl = Avar.Tl

data\_vars.MA.I = Avar.I

data\_vars.MA.Tin = Avar.Tin

data\_vars.MA.Al = Avar.Al

data\_vars.MA.Pr = Avar.Pr

data\_vars.MA.Re = Avar.Re

data\_vars.MA.Dis = Adis

else:

data\_vars.MA1.Ta = Avar1.Ta

data\_vars.MA1.Hr = Avar1.Hr

data\_vars.MA1.Tl = Avar1.Tl

data\_vars.MA1.I = Avar1.I

data\_vars.MA1.Tin = Avar1.Tin

data\_vars.MA1.Al = Avar1.Al

data\_vars.MA1.Pr = Avar1.Pr

data\_vars.MA1.Re = Avar1.Re

data\_vars.MA1.Dis = Adis1

data\_vars.MA2.Ta = Avar2.Ta

data\_vars.MA2.Hr = Avar2.Hr

data\_vars.MA2.Tl = Avar2.Tl

data\_vars.MA2.I = Avar2.I

data\_vars.MA2.Tin = Avar2.Tin

data\_vars.MA2.Al = Avar2.Al

data\_vars.MA2.Pr = Avar2.Pr

data\_vars.MA2.Re = Avar2.Re

data\_vars.MA2.Dis = Adis2

data\_vars.MA3.Ta = Avar3.Ta

data\_vars.MA3.Hr = Avar3.Hr

data\_vars.MA3.Tl = Avar3.Tl

data\_vars.MA3.I = Avar3.I

data\_vars.MA3.Tin = Avar3.Tin

data\_vars.MA3.Al = Avar3.Al

data\_vars.MA3.Pr = Avar3.Pr

data\_vars.MA3.Re = Avar3.Re

data\_vars.MA3.Dis = Adis3

return data\_vars

# Conexión para la adquisición de variables

import httplib2 as http

from urlparse import urlparse

class FitoGetMav:

def getDeviceData(self, cloud\_api, deviceid):

headers = {

'Content-Type': 'application/json'

}

url = cloud\_api

path = "/" + deviceid

target = urlparse(url+path)

method = 'GET'

body = ''

h = http.Http()

response, content = h.request(target.geturl(),method,body,headers)

data = content

#print response

return data

#mydata=FitoWebData()

#dato=mydata.getDeviceData("http://fitotron-api.appspot.com","2017")

#dato=mydata.getDeviceData("http://192.168.1.100","LUZ")

#print dato

# Conexión con Firebase

from firebase import firebase

firebase = firebase.FirebaseApplication('https://fitotronapp.firebaseio.com', None)

class Object:

def toJSON(self):

return json.dumps(self, default=lambda o: o.\_\_dict\_\_, sort\_keys=True, indent=4)

class FitoFirebase:

def sendToFirebase(self, vars, mod = "ALL", num = "12"):

# vars: Objeto obtenido de getTotalVars de clase FitoMavGetVars

# mod: Cantidad de modulos monitoreo ambiente, "S" = un modulo, "ALL" = Todos los modulos. Este dato debe coincidir con vars

# num: Numero del modulo monitoreo ambiente (fitotron) del cual se obtienen los datos

# Crear diccionarios de control nivel y monitoreo ambiente

varCN = {'Nivel\_1': vars.CN.N1, 'Nivel\_2': vars.CN.N2, 'Nivel\_3': vars.CN.N3}

varMN = {'CO2': vars.MN.CO2, 'pH': vars.MN.PH, 'CE': vars.MN.CE, 'TL': vars.MN.TL, 'Re': vars.MN.Re}

# Enviar CN y MN a FireBase

firebase.patch("/Modulo\_Control\_Nivel", varCN)

firebase.patch("/Modulo\_Monitoreo\_Nutriente", varMN)

# Verificar cantidad de modulos monitoreo ambiente y crear diccionarios, depues enviar a FireBase

if mod == "S" or mod == "s":

if num == "12":

varMA1 = {'Ta': vars.MA.Ta, 'Hr': vars.MA.Hr, 'Tl': vars.MA.Tl, 'I': vars.MA.I, 'Tin': vars.MA.Tin, 'Al': vars.MA.Al, 'Pr': vars.MA.Pr, 'Re': vars.MA.Re, 'Dis': vars.MA.Dis}

firebase.patch("/Modulo\_Monitoreo\_Ambiente\_1", varMA1)

if num == "13":

varMA2 = {'Ta': vars.MA.Ta, 'Hr': vars.MA.Hr, 'Tl': vars.MA.Tl, 'I': vars.MA.I, 'Tin': vars.MA.Tin, 'Al': vars.MA.Al, 'Pr': vars.MA.Pr, 'Re': vars.MA.Re, 'Dis': vars.MA.Dis}

firebase.patch("/Modulo\_Monitoreo\_Ambiente\_2", varMA2)

if num == "14":

varMA3 = {'Ta': vars.MA.Ta, 'Hr': vars.MA.Hr, 'Tl': vars.MA.Tl, 'I': vars.MA.I, 'Tin': vars.MA.Tin, 'Al': vars.MA.Al, 'Pr': vars.MA.Pr, 'Re': vars.MA.Re, 'Dis': vars.MA.Dis}

firebase.patch("/Modulo\_Monitoreo\_Ambiente\_3", varMA3)

elif mod == "ALL":

varMA1 = {'Ta': vars.MA1.Ta, 'Hr': vars.MA1.Hr, 'Tl': vars.MA1.Tl, 'I': vars.MA1.I, 'Tin': vars.MA1.Tin, 'Al': vars.MA1.Al, 'Pr': vars.MA1.Pr, 'Re': vars.MA1.Re, 'Dis': vars.MA1.Dis}

varMA2 = {'Ta': vars.MA2.Ta, 'Hr': vars.MA2.Hr, 'Tl': vars.MA2.Tl, 'I': vars.MA2.I, 'Tin': vars.MA2.Tin, 'Al': vars.MA2.Al, 'Pr': vars.MA2.Pr, 'Re': vars.MA2.Re, 'Dis': vars.MA2.Dis}

varMA3 = {'Ta': vars.MA3.Ta, 'Hr': vars.MA3.Hr, 'Tl': vars.MA3.Tl, 'I': vars.MA3.I, 'Tin': vars.MA3.Tin, 'Al': vars.MA3.Al, 'Pr': vars.MA3.Pr, 'Re': vars.MA3.Re, 'Dis': vars.MA3.Dis}

firebase.patch("/Modulo\_Monitoreo\_Ambiente\_1", varMA1)

firebase.patch("/Modulo\_Monitoreo\_Ambiente\_2", varMA2)

firebase.patch("/Modulo\_Monitoreo\_Ambiente\_3", varMA3)

else:

print "Argumentos invalidos en metodo sendToFirebase de la\nclase FitoFirebase."

return "Hecho"

def sendAmbFirebase(self, vars, dis, num):

### Funcion de prueba ###

if num == "12":

varMA1 = {'Ta': vars.Ta, 'Hr': vars.Hr, 'Tl': vars.Tl, 'I': vars.I, 'Tin': vars.Tin, 'Al': vars.Al, 'Pr': vars.Pr, 'Re': vars.Re, 'Dis': dis}

firebase.patch("/Modulo\_Monitoreo\_Ambiente\_1", varMA1)

if num == "13":

varMA2 = {'Ta': vars.Ta, 'Hr': vars.Hr, 'Tl': vars.Tl, 'I': vars.I, 'Tin': vars.Tin, 'Al': vars.Al, 'Pr': vars.Pr, 'Re': vars.Re, 'Dis': dis}

firebase.patch("/Modulo\_Monitoreo\_Ambiente\_2", varMA2)

if num == "14":

varMA3 = {'Ta': vars.Ta, 'Hr': vars.Hr, 'Tl': vars.Tl, 'I': vars.I, 'Tin': vars.Tin, 'Al': vars.Al, 'Pr': vars.Pr, 'Re': vars.Re, 'Dis': dis}

firebase.patch("/Modulo\_Monitoreo\_Ambiente\_3", varMA3)

def getFromFirebase(self):

bombas = firebase.get('Modulo\_Bombas', None)

luces = firebase.get('Modulo\_iluminacion', None)

datos = Object()

datos.b1 = bombas['Bomba1']

datos.b2 = bombas['Bomba2']

datos.b3 = bombas['Bomba3']

datos.i1 = luces['iluminacion1']

datos.i2 = luces['iluminacion2']

datos.i3 = luces['iluminacion3']

return datos

def sendEspFirebase(self, folder, label, value):

# Actualizar un valor en especifico

var = {label:value}

firebase.patch(folder, var)

return "Hecho"

def getEspFirebase(self, ruta):

value = firebase.get(ruta, None)

return value

def sendBombFirebase(self, value):

# Actualizar todos los valores en Modulo\_Bombas de una vez

var = {'Bomba1':value, 'Bomba2':value, 'Bomba3':value}

firebase.patch('Modulo\_Bombas', var)

return "Hecho"

# Enviar variables a plataforma FitoSmart

#from picamera import PiCamera,Color

from time import sleep

from dataForSend import \*

import zmq

import sys

from random import random

import base64

import time,datetime

import json

sys.path.append("/home/pi/Desktop/fitotron\_code/csc")

from FitoConfig import \*

class Object:

def toJSON(self):

return json.dumps(self, default=lambda o: o.\_\_dict\_\_,

sort\_keys=True, indent=4)

class FitoSendData:

def sendData(self, vars, ip, numport, lat, lng, id\_device = 96, id\_planta = 100, url\_image0 = "/home/pi/nueva.jpg", url\_image1 = "/home/pi/programas/planta.jpg"):

port = numport

if len(sys.argv) > 1:

port = sys.argv[1]

int(port)

if len(sys.argv) > 2:

port1 = sys.argv[2]

int(port1)

context = zmq.Context()

print "Connecting to server..."

print ip

socket = context.socket(zmq.REQ)

socket.connect ("tcp://"+ ip + ":%s" % port)

# toma lateral - 0

if not url\_image0 == "n":

image=open(url\_image0,"rb")

image\_read=image.read()

print("enviando imagen 0 .jpg")

pack = (image\_read)

#for dato in pack:

socket.send(image\_read)

message\_url0=socket.recv()

print message\_url0

# toma lateral - 1

if not url\_image1 == "n":

image=open(url\_image1,"rb")

image\_read=image.read()

print("enviando imagen 1 .jpg")

pack = (image\_read)

#for dato in pack:

socket.send(image\_read)

message\_url1=socket.recv()

print message\_url1

today = '%s' % datetime.datetime.now()

print "Fecha actual: ", today

data\_vars = Object()

data\_vars.values = Object()

data\_vars.fecha = today

if not id\_device == "n": data\_vars.code\_device = id\_device

if not id\_planta == "n": data\_vars.id\_planta = id\_planta

if not url\_image0 == "n": data\_vars.urlImageLat = message\_url0

if not url\_image1 == "n": data\_vars.urlImageSup = message\_url1

####### Variables de Modulo Control Nivel ######

data\_vars.values.N1 = vars.CN.N1

data\_vars.values.N2 = vars.CN.N2

data\_vars.values.N3 = vars.CN.N3

###### Variables de Modulo Monitoreo Nutriente ######

data\_vars.values.co2 = vars.MN.CO2

data\_vars.values.ph = vars.MN.PH

data\_vars.values.ce = vars.MN.CE

data\_vars.values.TL = vars.MN.TL

data\_vars.values.ReN = vars.MN.Re

###### Variables de Modulo Monitoreo Ambiente ######

data\_vars.values.temp = vars.MA.Ta

data\_vars.values.hume = vars.MA.Hr

data\_vars.values.Tl = vars.MA.Tl

data\_vars.values.I = vars.MA.I

data\_vars.values.Tin = vars.MA.Tin

data\_vars.values.Al = vars.MA.Al

data\_vars.values.Pr = vars.MA.Pr

data\_vars.values.ReA = vars.MA.Re

data\_vars.values.Dis = vars.MA.Dis

data\_vars.values.LAT = lat

data\_vars.values.LONG = lng

vars = data\_vars.toJSON()

#print "datos enviados: ",vars

socket.send(vars)

message=socket.recv()

#print "Mensaje: ",message

socket.close()

return message

def sendDataB(self, vars, id\_planta = 100, url\_image0 = "/home/pi/nueva.jpg", url\_image1 = "/home/pi/programas/planta.jpg"):

data = dataForSend()

ip, port, lat, lng, id\_device = data.getData()

context = zmq.Context()

print "Connecting to server..."

print ip

socket = context.socket(zmq.REQ)

socket.connect ("tcp://"+ ip + ":%s" % port)

today = '%s' % datetime.datetime.now()

print "Fecha actual: ", today

# toma lateral - 0

if not url\_image0 == "n":

image=open(url\_image0,"rb")

image\_read=image.read()

print("enviando imagen 0 .jpg")

pack = (image\_read)

#for dato in pack:

socket.send(image\_read)

message\_url0=socket.recv()

print message\_url0

# toma lateral - 1

if not url\_image1 == "n":

image=open(url\_image1,"rb")

image\_read=image.read()

print("enviando imagen 1 .jpg")

pack = (image\_read)

#for dato in pack:

socket.send(image\_read)

message\_url1=socket.recv()

print message\_url1

data\_vars = Object()

data\_vars.values = Object()

if not id\_planta == "n": data\_vars.id\_planta = id\_planta

if not url\_image0 == "n": data\_vars.urlImageLat = message\_url0

if not url\_image1 == "n": data\_vars.urlImageSup = message\_url1

data\_vars.fecha = today

data\_vars.code\_device = id\_device

data\_vars.id\_planta = id\_planta

data\_vars.values.Bo1 = vars.R1

data\_vars.values.Bo2 = vars.R2

data\_vars.values.Bo3 = vars.R3

data\_vars.LAT = lat

data\_vars.LONG = lng

vars = data\_vars.toJSON()

#print "datos enviados: ",vars

socket.send(vars)

message=socket.recv()

socket.close()

return message

def sendData2(self, vars, ftn, id\_planta = 100, url\_image0 = "/home/pi/nueva.jpg", url\_image1 = "/home/pi/programas/planta.jpg"):

data = dataForSend()

ip, port, lat, lng, id\_device = data.getData()

if len(sys.argv) > 1:

port = sys.argv[1]

int(port)

if len(sys.argv) > 2:

port1 = sys.argv[2]

int(port1)

context = zmq.Context()

print "Connecting to server..."

print ip

socket = context.socket(zmq.REQ)

socket.connect ("tcp://"+ ip + ":%s" % port)

# toma lateral - 0

if not url\_image0 == "n":

image=open(url\_image0,"rb")

image\_read=image.read()

print("enviando imagen 0 .jpg")

pack = (image\_read)

#for dato in pack:

socket.send(image\_read)

message\_url0=socket.recv()

print message\_url0

# toma lateral - 1

if not url\_image1 == "n":

image=open(url\_image1,"rb")

image\_read=image.read()

print("enviando imagen 1 .jpg")

pack = (image\_read)

#for dato in pack:

socket.send(image\_read)

message\_url1=socket.recv()

print message\_url1

today = '%s' % datetime.datetime.now()

print "Fecha actual: ", today

data\_vars = Object()

data\_vars.values = Object()

data\_vars.fecha = today

if not id\_device == "n": data\_vars.code\_device = id\_device

if not id\_planta == "n": data\_vars.id\_planta = id\_planta

if not url\_image0 == "n": data\_vars.urlImageLat = message\_url0

if not url\_image1 == "n": data\_vars.urlImageSup = message\_url1

####### Variables de Modulo Control Nivel ######

data\_vars.values.N1 = vars.CN.N1

data\_vars.values.N2 = vars.CN.N2

data\_vars.values.N3 = vars.CN.N3

###### Variables de Modulo Monitoreo Nutriente ######

data\_vars.values.co2 = vars.MN.CO2

data\_vars.values.ph = vars.MN.PH

data\_vars.values.ce = vars.MN.CE

data\_vars.values.TL = vars.MN.TL

data\_vars.values.ReN = vars.MN.Re

###### Variables de Modulo Monitoreo Ambiente ######

data\_vars.values.temp = vars.MA.Ta

data\_vars.values.hume = vars.MA.Hr

data\_vars.values.Tl = vars.MA.Tl

data\_vars.values.I = vars.MA.I

data\_vars.values.Tin = vars.MA.Tin

data\_vars.values.Al = vars.MA.Al

data\_vars.values.Pr = vars.MA.Pr

data\_vars.values.ReA = vars.MA.Re

data\_vars.values.Dis = vars.MA.Dis

###### Fitotron correspondiente ######

data\_vars.values.Fitotron = ftn

data\_vars.values.LAT = lat

data\_vars.values.LONG = lng

vars = data\_vars.toJSON()

#print "datos enviados: ",vars

socket.send(vars)

message=socket.recv()

#print "Mensaje: ",message

socket.close()

return message

def sendAmbData(self, vars, id\_planta = 100, url\_image0 = "/home/pi/nueva.jpg", url\_image1 = "/home/pi/programas/planta.jpg"):

data = dataForSend()

ip, port, lat, lng, id\_device = data.getData()

if len(sys.argv) > 1:

port = sys.argv[1]

int(port)

if len(sys.argv) > 2:

port1 = sys.argv[2]

int(port1)

context = zmq.Context()

print "Connecting to server..."

print ip

socket = context.socket(zmq.REQ)

socket.connect ("tcp://"+ ip + ":%s" % port)

# toma lateral - 0

if not url\_image0 == "n":

image=open(url\_image0,"rb")

image\_read=image.read()

print("enviando imagen 0 .jpg")

pack = (image\_read)

#for dato in pack:

socket.send(image\_read)

message\_url0=socket.recv()

print message\_url0

# toma lateral - 1

if not url\_image1 == "n":

image=open(url\_image1,"rb")

image\_read=image.read()

print("enviando imagen 1 .jpg")

pack = (image\_read)

#for dato in pack:

socket.send(image\_read)

message\_url1=socket.recv()

print message\_url1

data\_vars = Object()

data\_vars.values = Object()

if not id\_planta == "n": data\_vars.id\_planta = id\_planta

if not url\_image0 == "n": data\_vars.urlImageLat = message\_url0

if not url\_image1 == "n": data\_vars.urlImageSup = message\_url1

data\_vars.fecha = today

data\_vars.code\_device = id\_device

data\_vars.id\_planta = id\_planta

###### Variables de Modulo Monitoreo Ambiente ######

data\_vars.values.temp = vars.MA.Ta

data\_vars.values.hume = vars.MA.Hr

data\_vars.values.Tl = vars.MA.Tl

data\_vars.values.I = vars.MA.I

data\_vars.values.Tin = vars.MA.Tin

data\_vars.values.Al = vars.MA.Al

data\_vars.values.Pr = vars.MA.Pr

data\_vars.values.ReA = vars.MA.Re

data\_vars.values.Dis = vars.MA.Dis

data\_vars.values.LAT = lat

data\_vars.values.LONG = lng

ars = data\_vars.toJSON()

#print "datos enviados: ",vars

socket.send(vars)

message=socket.recv()

#print "Mensaje: ",message

socket.close()

return message

# Datos para envío a FitoSmart

import sys

sys.path.append('/home/pi/Desktop/fitotron\_code/csc')

from FitoConfig import \*

from FitoGeolocation import \*

from FitoWebData import \*

class dataForSend:

def getData(self):

ip = ""

port = ""

cloud\_api = ""

device\_id = ""

lat = ""

lng = ""

address = ""

fitoc = FitoConfig()

fitog = FitoGeolocation()

fitow = FitoWebData()

dataC = fitoc.getConfigData();

ip = dataC['ip']

port = dataC['port']

cloud\_api = dataC['cloud\_api']

device\_id = dataC['deviceid']

dataW = fitow.getDeviceData(cloud\_api, device\_id)

data = dataW[0]

address = data['calle'] + ", " + data['colonia'] + " " + data['ciudad'] + " " + data['estado'] + " " + "mexico"

#lat, lng = fitog.getGeolocation(address)

lat = 18.8061187

lng = -97.1723751

return ip, port, lat, lng, device\_id

# Información WEB

#from picamera import PiCamera,Color

import httplib2 as http

from urlparse import urlparse

import json

class FitoWebData:

def getDeviceData(self, cloud\_api, deviceid):

headers = {

'Content-Type': 'application/json'

}

url = cloud\_api

path = "/getdevice?device=" + deviceid

target = urlparse(url+path)

method = 'GET'

body = ''

#print target

h = http.Http()

response, content = h.request(

target.geturl(),

method,

body,

headers)

# assume that content is a json reply

# parse content with the json module

data = json.loads(content)

print data

#djson = json.loads(data)

return data #['coord']

# Geolocalización

#from picamera import PiCamera,Color

#import httplib2 as http

#from urlparse import urlparse

import googlemaps

#from googlemaps import GoogleMaps

import json

class FitoGeolocation:

def getGeolocation(self, address):

gmaps = googlemaps.Client(key='AIzaSyDWDUcf4fsQETW14FyRhT0-wMHdbyAaP4g')

#gmaps = GoogleMaps('AIzaSyC8RayL0qVmqFu9NFttSCT5FTEexBdCavQ')

print address

# Geocoding an address

geocode\_result = gmaps.geocode(address)

print geocode\_result

#data = json.loads(content)

lat = geocode\_result[0]['geometry']['location']['lat']

lng = geocode\_result[0]['geometry']['location']['lng']

#print data

#djson = json.loads(data)

print lat

print lng

return lat, lng #['coord']

# Información del clima

#from picamera import PiCamera,Color

import httplib2 as http

from urlparse import urlparse

import json

class FitoWeather:

def getWeather(self, lat, lng):

headers = {

'Content-Type': 'application/json'

}

url = "http://api.openweathermap.org"

path = "/data/2.5/weather?lat=" + str(lat) + "&lon=" + str(lng) + "&APPID=2af72ef62258728d72777bef612f2a3e&units=metric&lang=es"

target = urlparse(url+path)

method = 'POST'

body = ''

h = http.Http()

response, content = h.request(

target.geturl(),

method,

body,

headers)

data = json.loads(content)

print data['weather']

print data['wind']

#djson = json.loads(data)

return data['main'], data['weather'], data['wind']

# Lectura de datos de configuración

import json

class FitoConfig:

def getConfigData(self):

url = "/home/pi/Desktop/fitotron\_code/csc/device.config"

#print url

with open(url) as f:

content = f.read()

# assume that content is a json reply

# parse content with the json module

print content

data = json.loads(content)

#print data

#djson = json.loads(data)

return data #['coord']

# Datos de configuración

{

"appname": "Fitotron - Modulo de Computo Sensible al Contexto",

"cloud\_api": "http://fitotron-api.appspot.com",

"deviceid": "2017",

"ip": "35.237.167.229",

"port": "3000",

"timer": "0"

}

# Conectar antena externa a red WiFi

#!/usr/bin/env python

# -\*- coding: latin-1 -\*-

from tkinter import \*

from PIL import ImageTk

from time import sleep

import tkMessageBox

import Tkinter as tk

import os

class connect(Frame):

def \_\_init\_\_(self, master=None):

Frame.\_\_init\_\_(self, master) #parameters that you want to send through the bFrame class

self.master = master #reference to the master widget, which is the tk window

#self.init\_config();

self.init\_window() #with that, we want to then run init\_window, which doesn't yet exist

#Creation of init\_window

def init\_window(self):

ifaces = open('/etc/network/interfaces','r')

texto = ifaces.read()

ifaces.close()

vec = texto.split('\n')

def guardaRed():

red = txt\_ssid.get()

pwd = txt\_pass.get()

if red == '' or pwd == '':

pass

else:

red = ' wpa-ssid "{0}"'.format(red)

pwd = ' wpa-psk "{0}"'.format(pwd)

vec[20] = red

vec[21] = pwd

ifaces = open('temp','w')

for x in range(len(vec)):

if x < (len(vec)-1): ifaces.write(vec[x]+'\n')

ifaces.close()

os.system("sudo cp temp /etc/network/interfaces")

os.system("rm temp")

sleep(2)

os.system("sudo reboot")

self.master.title("Red WiFi")

txt\_ssid = tk.Entry()

txt\_pass = tk.Entry()

txt\_ssid.grid(row=0, column=1)

txt\_pass.grid(row=1, column=1)

self.myCmdCodebar = Label(self.master, text="Nombre de Red").grid(row=0, column=0)

self.myCmdCodebar = Label(self.master, text="ContraseÃ±a ").grid(row=1, column=0)

self.myCmdSend = Button(self.master, text='Hecho', command=guardaRed, height = 1, width = 4)

self.myCmdSend.grid(row=2, column=0)

self.myCmdSend.configure(state=NORMAL)

def init\_config(self):

pass

#root window created. Here, that would be the only window, but

# you can later have windows within windows.

root = Tk()

root.geometry("270x80") #540x440

#creation of an instance

app = connect(root)

#mainloop

root.mainloop()