UNIVERSIDAD TECNOLÓGICA DE
SAN LUIS RIO COLORADO

APLICACION EN TKINTER

MTRA. IRENE GARCIA

ALUMNO: VICTOR MANUEL GALVAN COVARRUBIAS

TECNOLOGÍAS DE LA INFORMACIÓN

ÁREA DESARROLLO DE SOFTWARE MULTIPLATAFORMA

San Luis Rio Colorado, Sonora

Agosto, 2021

I Introducción

A lo largo de la carrera de TSU se imparte varias materias orientadas al desarrollo y aplicaciones de IOT. Esta se puede decir es la principal de esas materias.
En el quinto cuatrimestre se implementa es uso de sensores a través de programación en Python la cual con la ayuda de un Raspberry pi es posible manipularlos a los requisitos de cualquier proyecto. Para comprender lo que a continuación se presenta esto es necesario entender que es Python y Tkinter.

Python es un lenguaje de programación versátil multiplataforma y multiparadigma que se destaca por su código legible y limpio. La licencia de código abierto permite su utilización en distintos contextos sin la necesidad de abonar por ello y se emplea en plataformas de alto tráfico como Google, YouTube o Facebook.

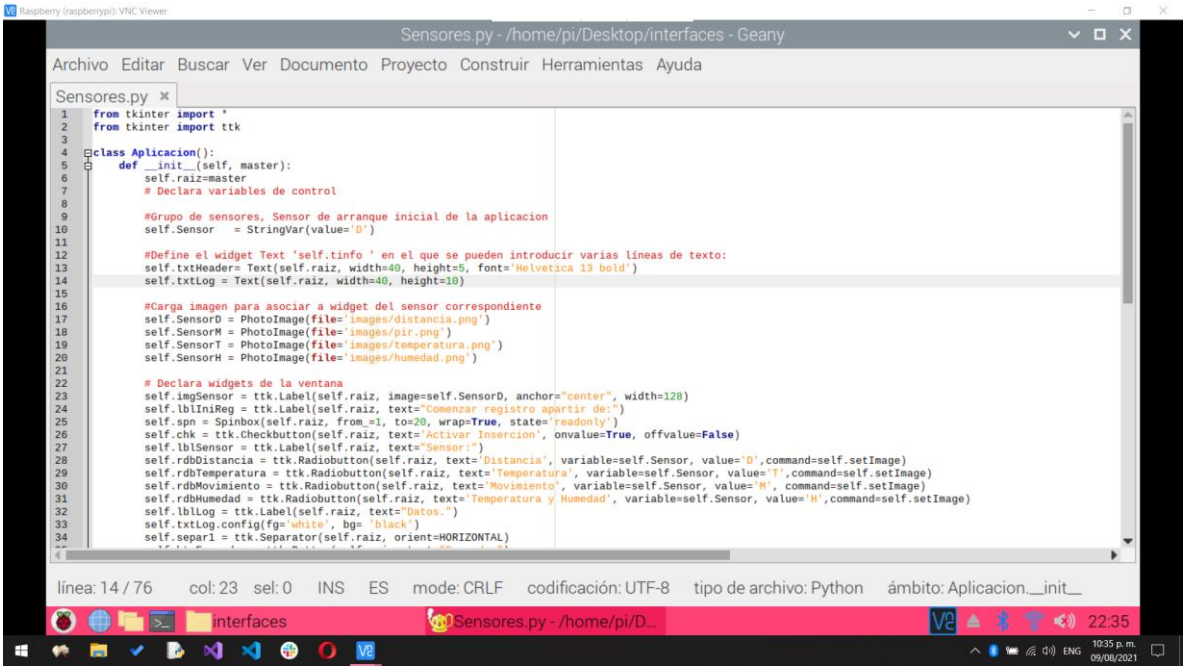
Tkinter es el paquete más utilizado para crear interfaces gráficas en Python. Es una capa orientada a objetos basada en Tcl (sencillo y versátil lenguaje de programación open-source) y Tk (la herramienta GUI estándar para Tcl).

Una vez entendido podemos continuar a lo largo del documento con la presentación del proyecto final de una de las materias más importantes de la carrera.
A continuación, se presentará el desarrollo de una interfaz gráfica la cual nos facilita la manipulación de los sensores vistos a lo largo del curso, así como también él envió de los datos registrados por estos a un servidor en la nube para su posterior interpretación y análisis.
Sin más que agregar a continuación se explica detalladamente el desarrollo de la interfaz en tres etapas principales.

II Desarrollo:

El proyecto consiste en implementar los cuatros sensores vistos en el curso (temperatura, distancia, humedad, movimiento) y adaptarlos a una sencilla interfaz gráfica la cual facilita su manipulación e inserción a una base de datos en MongoDB.

2.1 Etapa 1

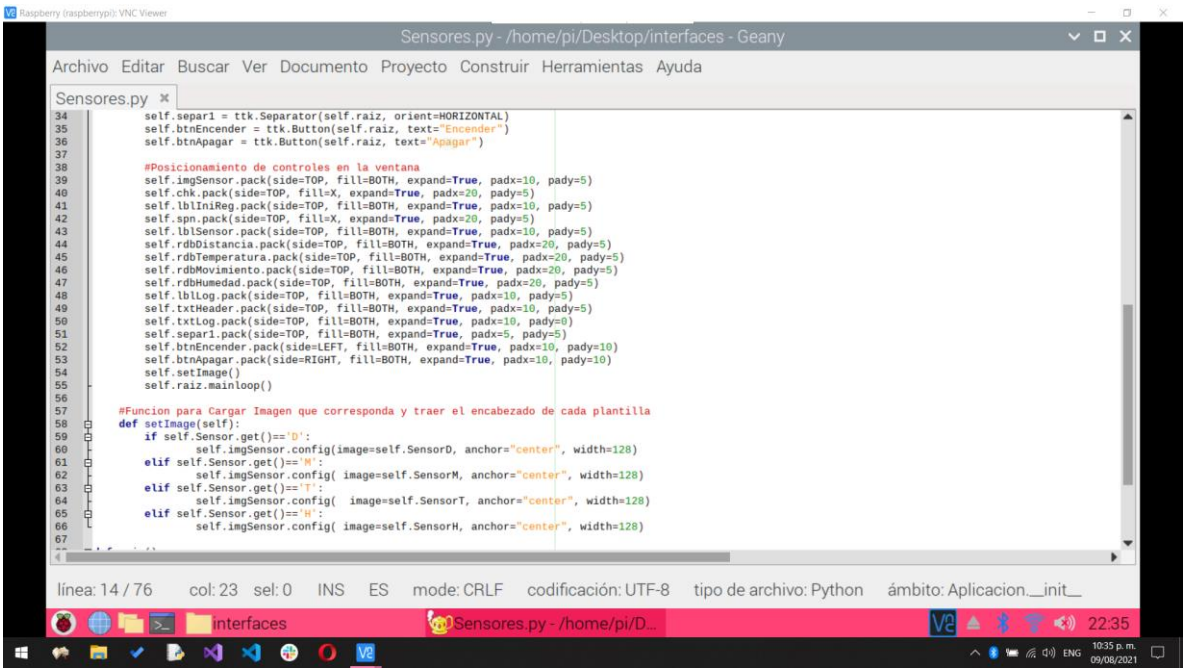



```
from tkinter import *
from tkinter import ttk

class Aplicacion():
    def __init__(self, master):
        self.raiz=master
        # Declara variables de control

        #Grupo de sensores, Sensor de arranque inicial de la aplicacion
        self.Sensor   = StringVar(value='D')

        #Define el widget Text 'self.tinfo ' en el que se pueden introducir varias lineas de texto:
        self.txtHeader= Text(self.raiz, width=40, height=5, font='Helvetica 13 bold')
        self.txtLog = Text(self.raiz, width=40, height=10)

        #Carga imagen para asociar a widget del sensor correspondiente
        self.SensorD = PhotoImage(file='images/distancia.png')
        self.SensorM = PhotoImage(file='images/pir.png')
        self.SensorT = PhotoImage(file='images/temperatura.png')
        self.SensorH = PhotoImage(file='images/humedad.png')

        # Declara widgets de la ventana
        self.imgSensor = ttk.Label(self.raiz, image=self.SensorD, anchor="center", width=128)
        self.lblIniReg = ttk.Label(self.raiz, text="Comenzar registro apartir de:")
        self.spn = Spinbox(self.raiz, from_=1, to=20, wrap=True, state='readonly')
        self.chk = ttk.Checkbutton(self.raiz, text='Activar Insercion', onvalue=True, offvalue=False)
        self.lblSensor = ttk.Label(self.raiz, text="Sensor:")
        self.rdbDistancia = ttk.Radiobutton(self.raiz, text='Distancia', variable=self.Sensor, value='D',command=self.setImage)
        self.rdbTemperatura = ttk.Radiobutton(self.raiz, text='Temperatura', variable=self.Sensor, value='T',command=self.setImage)
        self.rdbMovimiento = ttk.Radiobutton(self.raiz, text='Movimiento', variable=self.Sensor, value='M', command=self.setImage)
        self.rdbHumedad = ttk.Radiobutton(self.raiz, text='Temperatura y Humedad', variable=self.Sensor, value='H',command=self.setImage)
        self.lblLog = ttk.Label(self.raiz, text="Datos.")
        self.txtLog.config(fg='white', bg= 'black')
        self.separ1 = ttk.Separator(self.raiz, orient=HORIZONTAL)
```

Una vez importadas las librerías necesarias para trabajar con Tkinter se definen los widgets que componen a la interfaz gráfica, ocho en total para ser más precisos.

1. Primero se mostrará una imagen centrada la cual en base al sensor seleccionado seleccionará de la carpeta *images* la imagen correspondiente al mismo.
2. Continuando con los componentes se muestra un *checkbox* simple el cual activa y desactiva la inserción a MongoDB.
3. Lo siguiente que se puede apreciar es un s*pinbox,* este nos es útil solo si se activa la inserción a la base de datos puesto que establece la cantidad requerida para hacer el envió a la base de datos.
4. A continuación, se presenta un *radiobutton*, este es útil para cambiar al sensor que se desea utilizar.

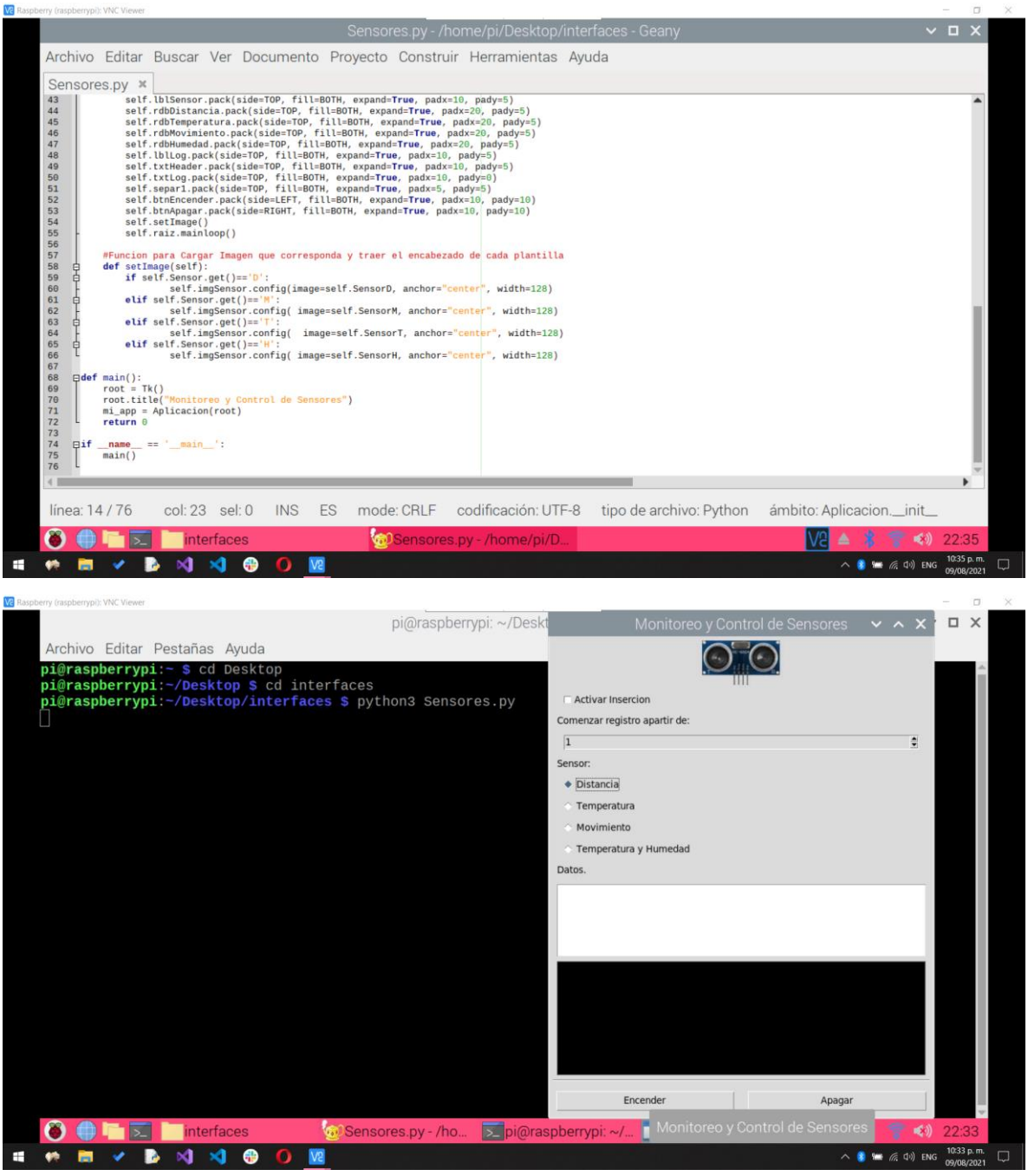
5. Ya por último se encuentran dos cajas te texto una es para el diseño de los encabezados que dependen de los datos que se arrojen con cada sensor.
6. Y la siguiente caja son las lecturas actuales que el sensor y la Raspberry esta arrojando.




```
        self.separ1 = ttk.Separator(self.raiz, orient=HORIZONTAL)
        self.btnEncender = ttk.Button(self.raiz, text="Encender")
        self.btnApagar = ttk.Button(self.raiz, text="Apagar")

        #Posicionamiento de controles en la ventana
        self.imgSensor.pack(side=TOP, fill=BOTH, expand=True, padx=10, pady=5)
        self.chk.pack(side=TOP, fill=X, expand=True, padx=20, pady=5)
        self.lblIniReg.pack(side=TOP, fill=BOTH, expand=True, padx=10, pady=5)
        self.spn.pack(side=TOP, fill=X, expand=True, padx=20, pady=5)
        self.lblSensor.pack(side=TOP, fill=BOTH, expand=True, padx=10, pady=5)
        self.rdbDistancia.pack(side=TOP, fill=BOTH, expand=True, padx=20, pady=5)
        self.rdbTemperatura.pack(side=TOP, fill=BOTH, expand=True, padx=20, pady=5)
        self.rdbMovimiento.pack(side=TOP, fill=BOTH, expand=True, padx=20, pady=5)
        self.rdbHumedad.pack(side=TOP, fill=BOTH, expand=True, padx=20, pady=5)
        self.lblLog.pack(side=TOP, fill=BOTH, expand=True, padx=10, pady=5)
        self.txtHeader.pack(side=TOP, fill=BOTH, expand=True, padx=10, pady=5)
        self.txtLog.pack(side=TOP, fill=BOTH, expand=True, padx=10, pady=0)
        self.separ1.pack(side=TOP, fill=BOTH, expand=True, padx=5, pady=5)
        self.btnEncender.pack(side=LEFT, fill=BOTH, expand=True, padx=10, pady=10)
        self.btnApagar.pack(side=RIGHT, fill=BOTH, expand=True, padx=10, pady=10)
        self.setImage()
        self.raiz.mainloop()

    #Funcion para Cargar Imagen que corresponda y traer el encabezado de cada plantilla
    def setImage(self):
        if self.Sensor.get()=='D':
                self.imgSensor.config(image=self.SensorD, anchor="center", width=128)
        elif self.Sensor.get()=='M':
                self.imgSensor.config( image=self.SensorM, anchor="center", width=128)
        elif self.Sensor.get()=='T':
                self.imgSensor.config(  image=self.SensorT, anchor="center", width=128)
        elif self.Sensor.get()=='H':
                self.imgSensor.config( image=self.SensorH, anchor="center", width=128)
```

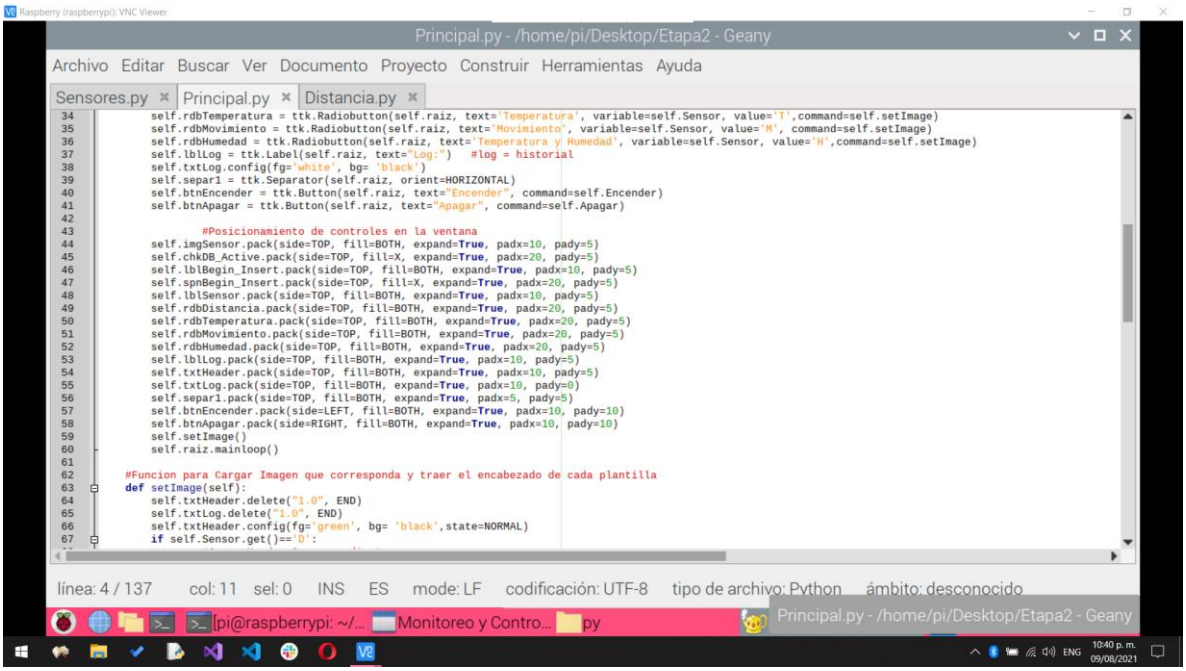
El diseño y el posicionamiento de los widgets está declarado en una sección separada de toda la funcionalidad, esto para un fácil acceso a los valores y posiciones que se requieren que tengan dentro de la ventana principal.

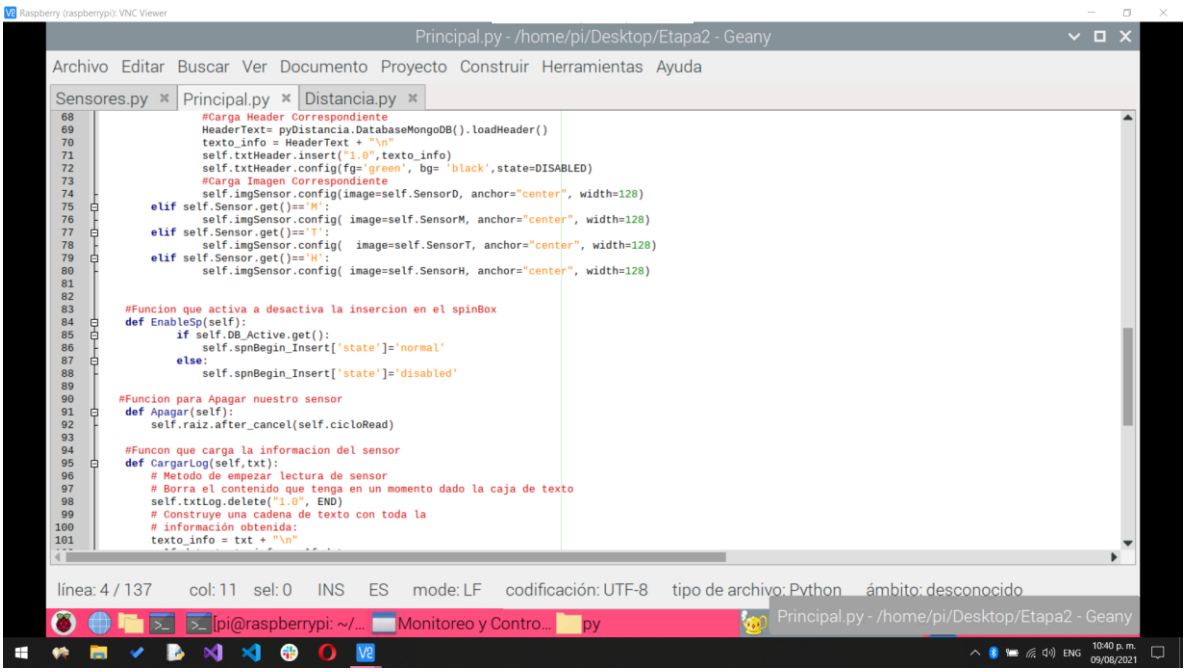
La función *setImage()* como su nombre lo indica es la encargada de cambiar la imagen en base al sensor seleccionado.

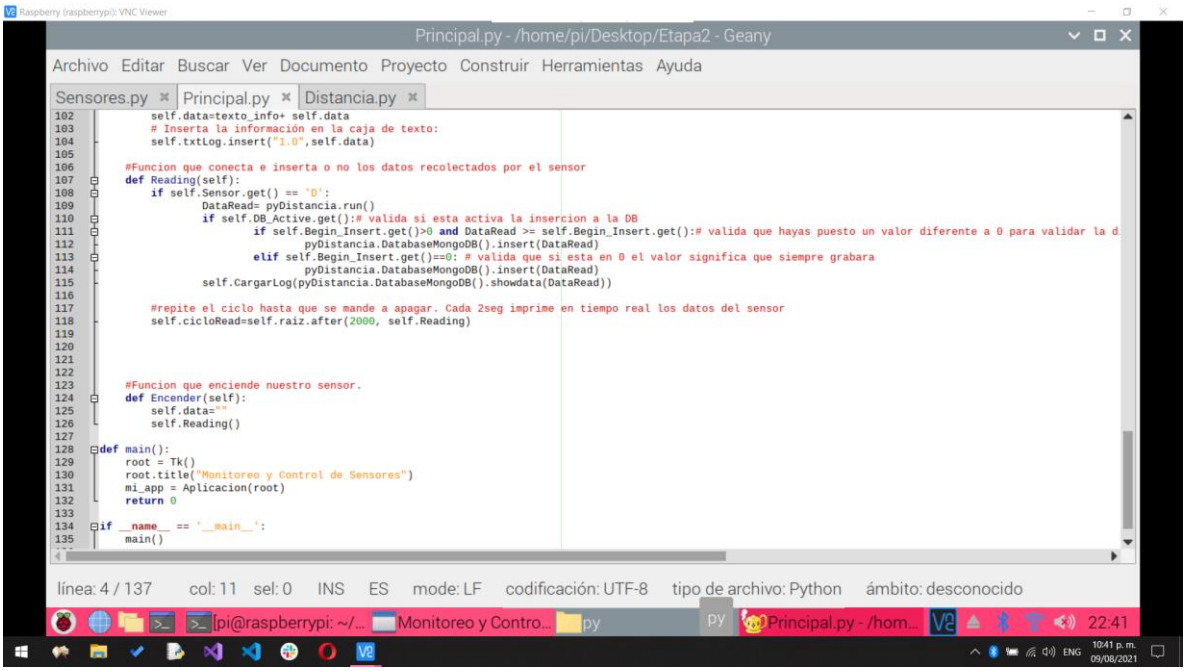


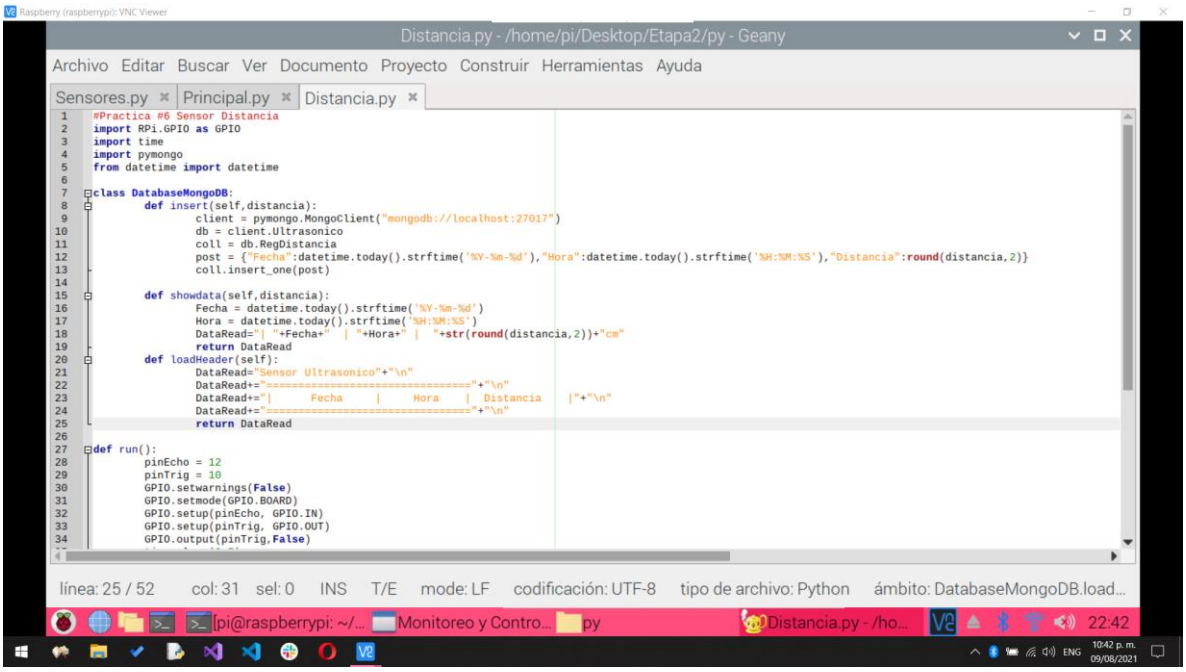
En la imagen anterior se pueden apreciar los componentes anteriormente descritos y su posicionamiento dentro de la ventana principal.

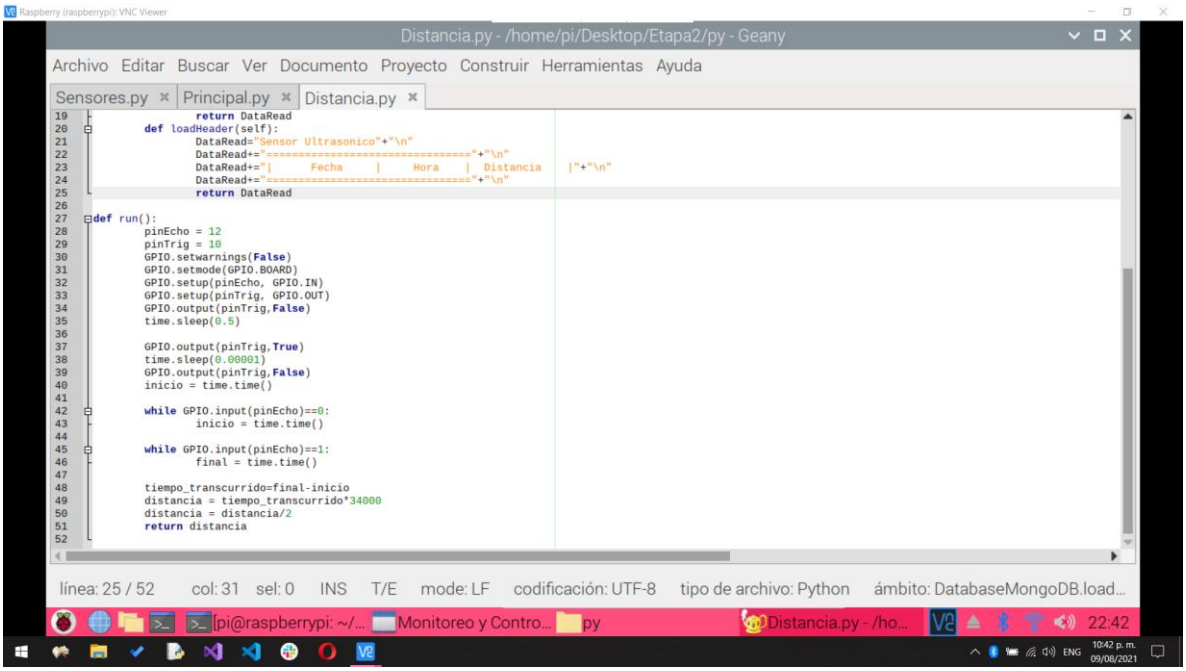
2.2 Etapa 2

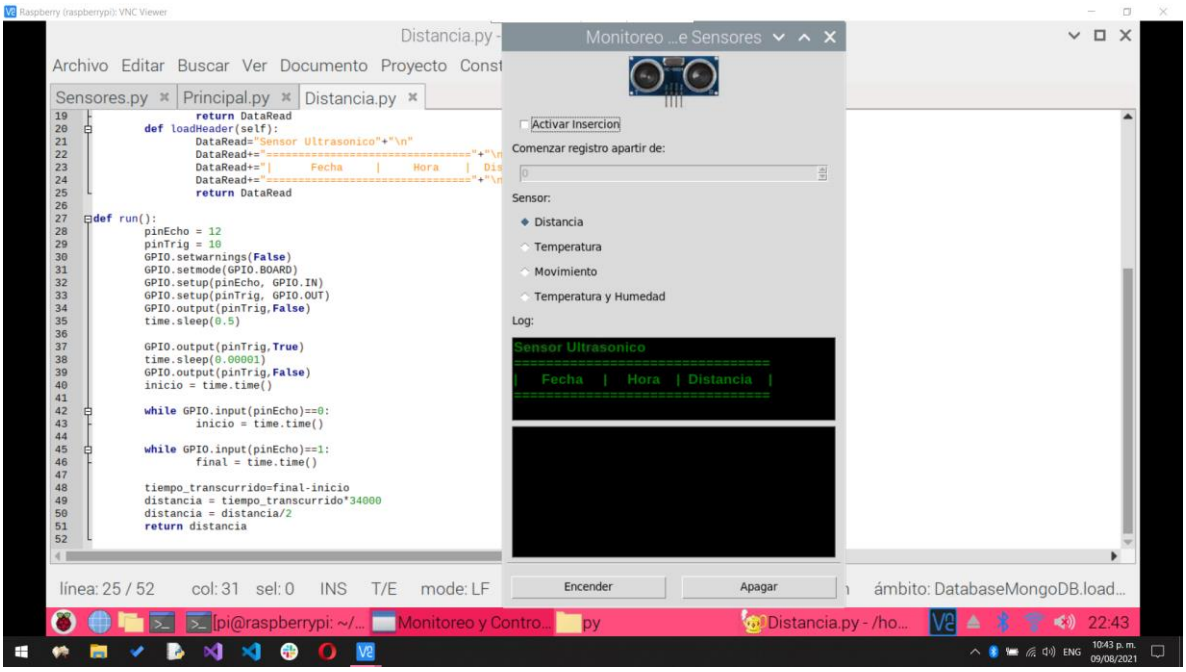

Principal.py - /home/pi/Desktop/Etapa2 - Geany
Archivo Editar Buscar Ver Documento Proyecto Construir Herramientas Ayuda
Sensores.py Principal.py Distancia.py
línea: 4 / 137 col: 11 sel: 0 INS ES mode: LF codificación: UTF-8 tipo de archivo: Python ámbito: desconocido


Principal.py - /home/pi/Desktop/Etapa2 - Geany
Archivo Editar Buscar Ver Documento Proyecto Construir Herramientas Ayuda
Sensores.py Principal.py Distancia.py
línea: 4 / 137 col: 11 sel: 0 INS ES mode: LF codificación: UTF-8 tipo de archivo: Python ámbito: desconocido


Principal.py - /home/pi/Desktop/Etapa2 - Geany
Archivo Editar Buscar Ver Documento Proyecto Construir Herramientas Ayuda
Sensores.py Principal.py Distancia.py
self.data=texto_info+ self.data
# Inserta la información en la caja de texto:
self.txtLog.insert("1.0",self.data)
#Funcion que conecta e inserta o no los datos recolectados por el sensor
def Reading(self):
if self.Sensor.get() == 'D':
DataRead= pyDistancia.run()
if self.DB_Active.get():# valida si esta activa la insercion a la DB
if self.Begin_Insert.get()>0 and DataRead >= self.Begin_Insert.get():# valida que hayas puesto un valor diferente a 0 para validar la d
pyDistancia.DatabaseMongoDB().insert(DataRead)
elif self.Begin_Insert.get()==0: # valida que si esta en 0 el valor significa que siempre grabara
pyDistancia.DatabaseMongoDB().insert(DataRead)
self.CargarLog(pyDistancia.DatabaseMongoDB().showdata(DataRead))
#repite el ciclo hasta que se mande a apagar. Cada 2seg imprime en tiempo real los datos del sensor
self.cicloRead=self.raiz.after(2000, self.Reading)
#Funcion que enciende nuestro sensor.
def Encender(self):
self.data=""
self.Reading()
def main():
root = Tk()
root.title("Monitoreo y Control de Sensores")
mi_app = Aplicacion(root)
return 0
if __name__ == '__main__':
main()
línea: 4 / 137 col: 11 sel: 0 INS ES mode: LF codificación: UTF-8 tipo de archivo: Python ámbito: desconocido
[pi@raspberrypi: ~/... Monitoreo y Contro... py Principal.py - /hom... 22:41
10:41 p. m. 09/08/2021


Distancia.py - /home/pi/Desktop/Etapa2/py - Geany
Archivo Editar Buscar Ver Documento Proyecto Construir Herramientas Ayuda
Sensores.py Principal.py Distancia.py
#Practica #6 Sensor Distancia
import RPi.GPIO as GPIO
import time
import pymongo
from datetime import datetime
class DatabaseMongoDB:
def insert(self,distancia):
client = pymongo.MongoClient("mongodb://localhost:27017")
db = client.Ultrasonico
coll = db.RegDistancia
post = {"Fecha":datetime.today().strftime('%Y-%m-%d'),"Hora":datetime.today().strftime('%H:%M:%S'),"Distancia":round(distancia,2)}
coll.insert_one(post)
def showdata(self,distancia):
Fecha = datetime.today().strftime('%Y-%m-%d')
Hora = datetime.today().strftime('%H:%M:%S')
DataRead="| "+Fecha+" | "+Hora+" | "+str(round(distancia,2))+"cm"
return DataRead
def loadHeader(self):
DataRead="Sensor Ultrasonico"+"\n"
DataRead+="============================="+"\n"
DataRead+="| Fecha | Hora | Distancia |"+"\n"
DataRead+="============================="+"\n"
return DataRead
def run():
pinEcho = 12
pinTrig = 10
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(pinEcho, GPIO.IN)
GPIO.setup(pinTrig, GPIO.OUT)
GPIO.output(pinTrig,False)
línea: 25 / 52 col: 31 sel: 0 INS T/E mode: LF codificación: UTF-8 tipo de archivo: Python ámbito: DatabaseMongoDB.load...
[pi@raspberrypi: ~/... Monitoreo y Contro... py Distancia.py - /ho... 22:42
10:42 p. m. 09/08/2021


Distancia.py - /home/pi/Desktop/Etapa2/py - Geany
Archivo Editar Buscar Ver Documento Proyecto Construir Herramientas Ayuda
Sensores.py
Principal.py
Distancia.py
return DataRead
def loadHeader(self):
DataRead="Sensor Ultrasonico"+"\n"
DataRead+="==========="+"\n"
DataRead+="| Fecha | Hora | Distancia |"+"\n"
DataRead+="==========="+"\n"
return DataRead
def run():
pinEcho = 12
pinTrig = 10
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(pinEcho, GPIO.IN)
GPIO.setup(pinTrig, GPIO.OUT)
GPIO.output(pinTrig,False)
time.sleep(0.5)
GPIO.output(pinTrig,True)
time.sleep(0.00001)
GPIO.output(pinTrig,False)
inicio = time.time()
while GPIO.input(pinEcho)==0:
inicio = time.time()
while GPIO.input(pinEcho)==1:
final = time.time()
tiempo_transcurrido=final-inicio
distancia = tiempo_transcurrido*34000
distancia = distancia/2
return distancia
línea: 25 / 52 col: 31 sel: 0 INS T/E mode: LF codificación: UTF-8 tipo de archivo: Python ámbito: DatabaseMongoDB.load...
[pi@raspberrypi: ~/... Monitoreo y Contro... py Distancia.py - /ho...
22:42
10:42 p. m.
09/08/2021


Monitoreo ...e Sensores
Activar Insercion
Comenzar registro apartir de:
Sensor:
Distancia
Temperatura
Movimiento
Temperatura y Humedad
Log:
Sensor Ultrasonico
| Fecha | Hora | Distancia |
Encender
Apagar
22:43
10:43 p. m.
09/08/2021

2.3 Etapa 3

III Conclusión

El IoT es uno de los aspectos más fuertes de esta carrera, ya es normal escuchar en cualquier parte este término. Realmente el IoT se está implementando a las tecnologías actuales y es uno de los principales campos en los cuales alguien se puede desempeñar completamente. La materia a la perspectiva de un alumno común fue abordada de una manera excelente porque realmente es más práctica que teoría. Jamás será lo mismo ver como se hace, construye o implementa algo a realmente hacerlo de primera fila. Esto de igual manera ayuda a la resolución de problemas que se presentan en tiempo real al desarrollar las practicas propuestas. Existen una gran variedad de sensores y actuadores disponibles para crear lo que nuestra imaginación nos permita. Por eso y más IoT es una de las mejores materias ya que nada está definido. Una persona puede simplemente ver una actividad y a través del análisis y arquitecturas IoT mejorar todo aspecto posible de ese proceso para maximizar rendimiento y recursos. Algo que no se puede hacer en las demás ramas ya que estas no representan tanto a la parte física de la industria.

El proyecto anteriormente presentado puede ser implementado en un futuro de igual manera con un sinfín de sensores disponibles para la Raspberry pi. El proporcionarle una interfaz gráfica limpia realmente es algo que un usuario inexperto agradece mucho. Ya que facilita el aprendizaje, así como la sencillez de operabilidad.

Los conocimientos aquí aprendidos cumplieron de manera excelente las expectativas esperadas al inicio de los cursos.