

Instrumento de evaluación

Materia: Aplicaciones para IoT Unidad: 3 Tipo de Instrumento: _____
Actividad: Examen Parcial 3 Tipo de Reactivo: 6

Nombre del Maestro(a): Ing. Irma Irene García Razcón Calificación: _____

Nombre de Alumno(a): Victor Manuel Galvan Covarrubias Fecha: 23 agosto 2021

1. – Agregue el documento final de aplicación realizada en Tkinter para control y monitores de sensores. Valor 30 puntos.

El documento deberá contener:

1.- Portada (Logo de carrera, logo de la universidad, nombre de la materia, nombre del maestro, nombre del alumno y fecha).

2.- Introducción. Esta deberá dar una idea lo que contiene el documento, y agregar o mencionar a Python y Tkinter.

3.- Desarrollo:

3.1.- Etapa 1: Explicar el desarrollo de esta etapa (Diseño), explicar brevemente los 8 widgets que se están utilizando, agregar fotos del código completo y de la interfaz gráfica.

3.2.- Etapa 2: Explicar el desarrollo de esta etapa y como se ha agregado el archivo .py y el código del sensor.

3.3.- Etapa 3: Explicar el desarrollo de esta etapa y como se han ido agregando los archivos .py y el código del resto de los sensores.

4.- Conclusión.

Nota: El documento podrá ser realizado en español.

Esta hoja de instrucciones deberá ser la primera hoja de su documento, regresar en formato PDF. En esta plantilla de examen.

Tipo de reactivo

1. Falso/Verdadero
2. Respuesta corta
3. Opción múltiple
4. De relación
5. Preguntas de análisis
6. Casos prácticos
7. Ejercicios para resolver

Tipo de Instrumento

1. Rubrica
2. Lista de cotejo
3. Reactivos

FAC-EA-04
REV02



Universidad Tecnológica
de San Luis Río Colorado



**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 Río Colorado, Sonora

Agosto, 2021



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 el 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.

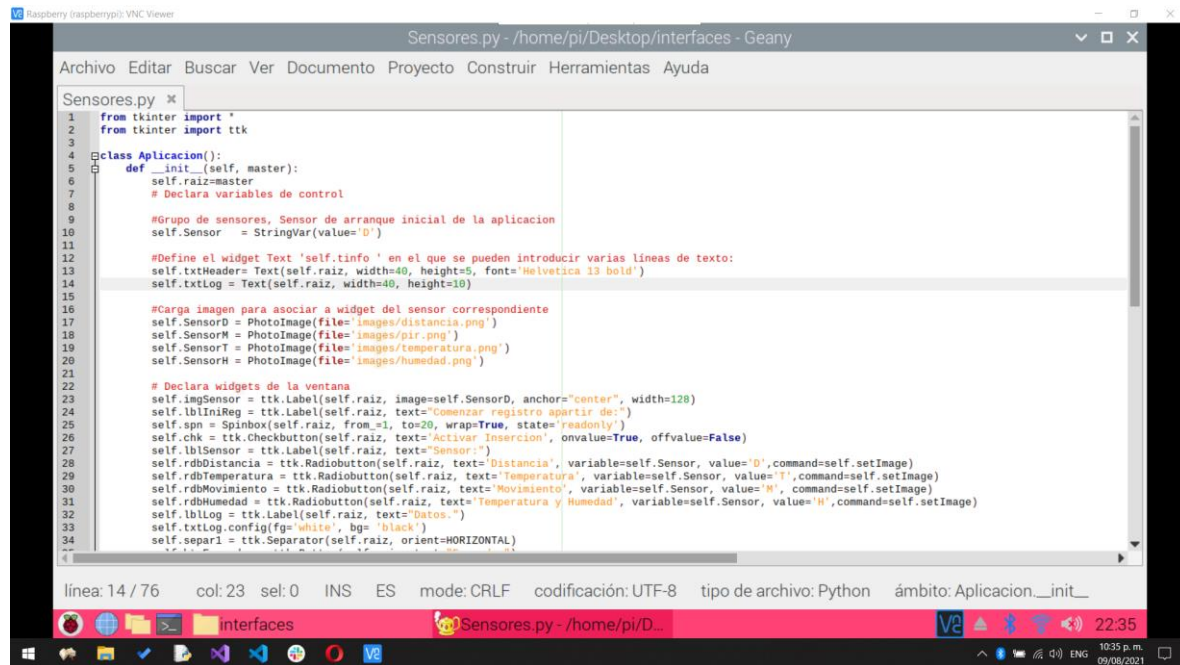
A continuación, se presentará el desarrollo de una interfaz gráfica la cual nos facilita la manipulación de los sensores vistos a lo largo del curso, así como también el envío de los datos registrados por estos a un servidor en la nube para su posterior interpretación y análisis.

Sin más que agregar a continuación se explica detalladamente el desarrollo de la interfaz en tres etapas principales.

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.

Etaa 1

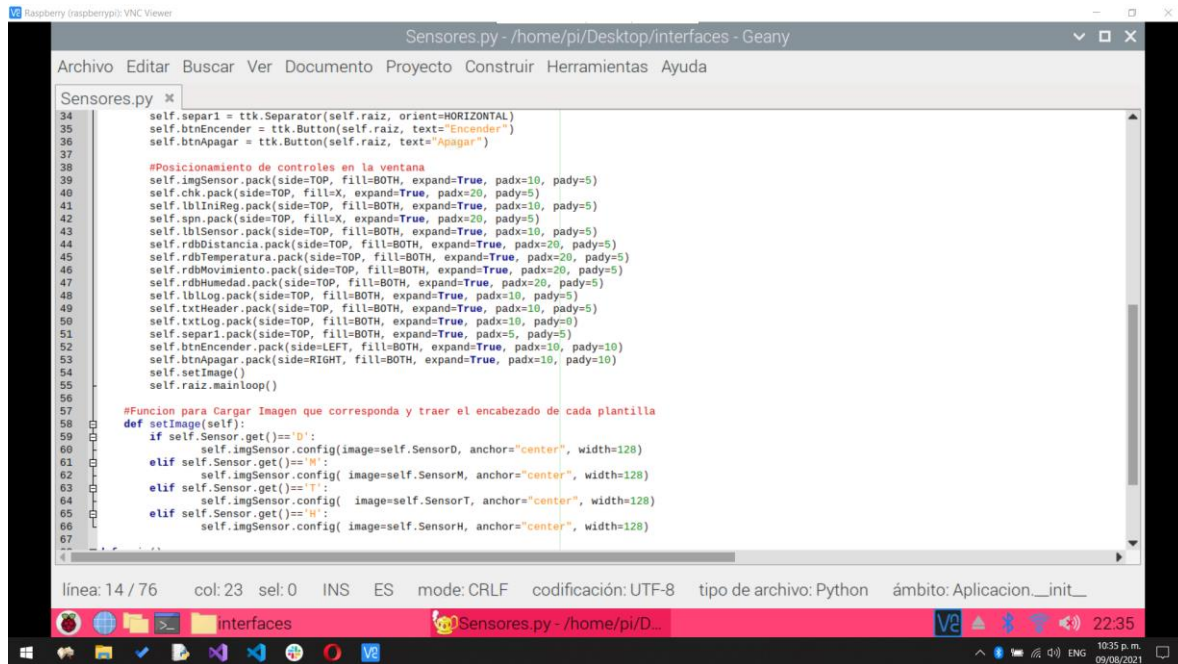


```
1 from tkinter import *
2 from tkinter import ttk
3
4 class Aplicacion():
5     def __init__(self, master):
6         self.raiz=master
7         # Declara variables de control
8
9         #Grupo de sensores, Sensor de arranque inicial de la aplicacion
10        self.Sensor = StringVar(value='D')
11
12        #Define el widget Text 'self.tinfo' en el que se pueden introducir varias lineas de texto:
13        self.txtHeader= Text(self.raiz, width=40, height=5, font='Helvetica 13 bold')
14        self.txtLog = Text(self.raiz, width=40, height=10)
15
16        #Carga imagen para asociar a widget del sensor correspondiente
17        self.SensorD = PhotoImage(file='images/distancia.png')
18        self.SensorM = PhotoImage(file='images/pir.png')
19        self.SensorT = PhotoImage(file='images/temperatura.png')
20        self.SensorH = PhotoImage(file='images/humedad.png')
21
22        # Declara widgets de la ventana
23        self.imgSensor = ttk.Label(self.raiz, image=self.SensorD, anchor="center", width=120)
24        self.lblIniReg = ttk.Label(self.raiz, text="Comenzar registro a partir de:")
25        self.spn = Spinbox(self.raiz, from_=1, to=20, wrap=True, state='readonly')
26        self.chk = ttk.Checkbutton(self.raiz, text="Activar insercion", onvalue=True, offvalue=False)
27        self.lblSensor = ttk.Label(self.raiz, text="Sensor:")
28        self.rdbDistancia = ttk.Radiobutton(self.raiz, text="Distancia", variable=self.Sensor, value='D', command=self.setImage)
29        self.rdbTemperatura = ttk.Radiobutton(self.raiz, text="Temperatura", variable=self.Sensor, value='T', command=self.setImage)
30        self.rdbMovimiento = ttk.Radiobutton(self.raiz, text="Movimiento", variable=self.Sensor, value='M', command=self.setImage)
31        self.rdbHumedad = ttk.Radiobutton(self.raiz, text="Humedad", variable=self.Sensor, value='H', command=self.setImage)
32        self.lblLog = ttk.Label(self.raiz, text="Datos:")
33        self.txtLog.config(fg='white', bg='black')
34        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 *spinbox*, este nos es útil solo si se activa la inserción a la base de datos puesto que establece la cantidad requerida para hacer el envío 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 de 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.

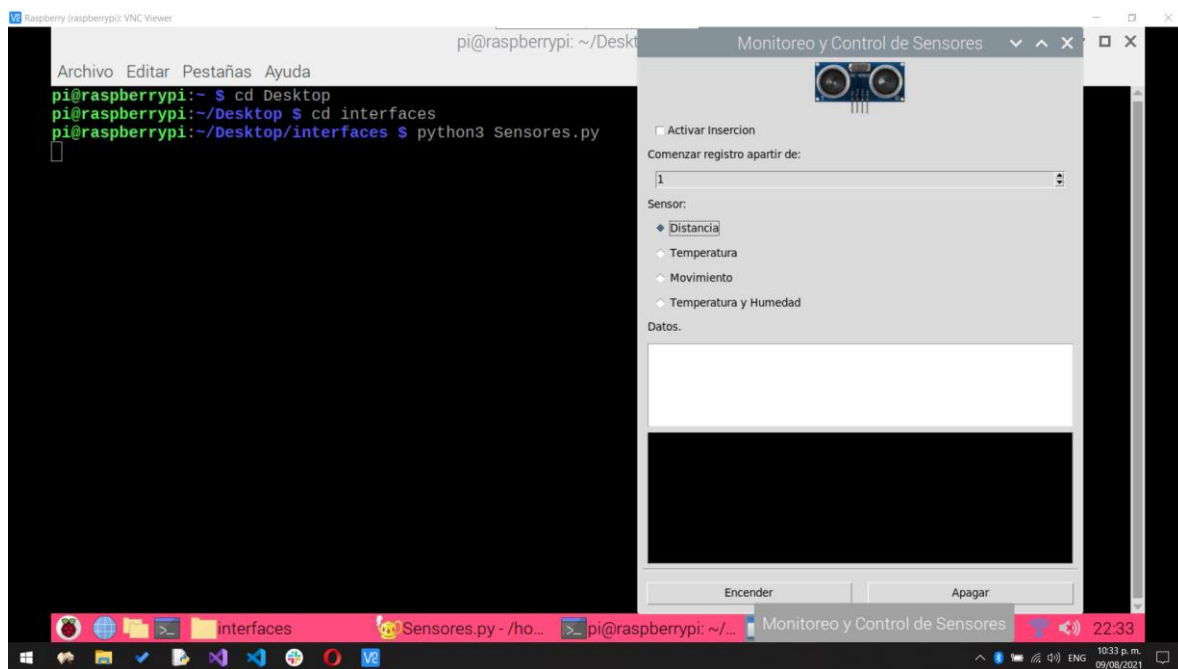


```
34 self.separ1 = ttk.Separator(self.raiz, orient=HORIZONTAL)
35 self.btnEncender = ttk.Button(self.raiz, text="Encender")
36 self.btnApagar = ttk.Button(self.raiz, text="Apagar")
37
38 #Posicionamiento de controles en la ventana
39 self.imgSensor.pack(side=TOP, fill=BOTH, expand=True, padx=10, pady=5)
40 self.chk.pack(side=TOP, fill=X, expand=True, padx=20, pady=5)
41 self.lblIniReg.pack(side=TOP, fill=BOTH, expand=True, padx=10, pady=5)
42 self.spn.pack(side=TOP, fill=X, expand=True, padx=20, pady=5)
43 self.lblSensor.pack(side=TOP, fill=BOTH, expand=True, padx=10, pady=5)
44 self.rdbDistancia.pack(side=TOP, fill=BOTH, expand=True, padx=20, pady=5)
45 self.rdbTemperatura.pack(side=TOP, fill=BOTH, expand=True, padx=20, pady=5)
46 self.rdbMovimiento.pack(side=TOP, fill=BOTH, expand=True, padx=20, pady=5)
47 self.rdbHumedad.pack(side=TOP, fill=BOTH, expand=True, padx=20, pady=5)
48 self.lblLog.pack(side=TOP, fill=BOTH, expand=True, padx=10, pady=5)
49 self.txtHeader.pack(side=TOP, fill=BOTH, expand=True, padx=10, pady=5)
50 self.txtLog.pack(side=TOP, fill=BOTH, expand=True, padx=10, pady=0)
51 self.separ1.pack(side=TOP, fill=BOTH, expand=True, padx=5, pady=5)
52 self.btnEncender.pack(side=LEFT, fill=BOTH, expand=True, padx=10, pady=10)
53 self.btnApagar.pack(side=RIGHT, fill=BOTH, expand=True, padx=10, pady=10)
54 self.setImage()
55 self.raiz.mainloop()
56
57 #Funcion para Cargar Imagen que corresponda y traer el encabezado de cada plantilla
58 def setImage(self):
59     if self.Sensor.get()=="D":
60         self.imgSensor.config(image=self.SensorD, anchor="center", width=128)
61     elif self.Sensor.get()=="M":
62         self.imgSensor.config( image=self.SensorM, anchor="center", width=128)
63     elif self.Sensor.get()=="T":
64         self.imgSensor.config( image=self.SensorT, anchor="center", width=128)
65     elif self.Sensor.get()=="H":
66         self.imgSensor.config( image=self.SensorH, anchor="center", width=128)
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
```

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.

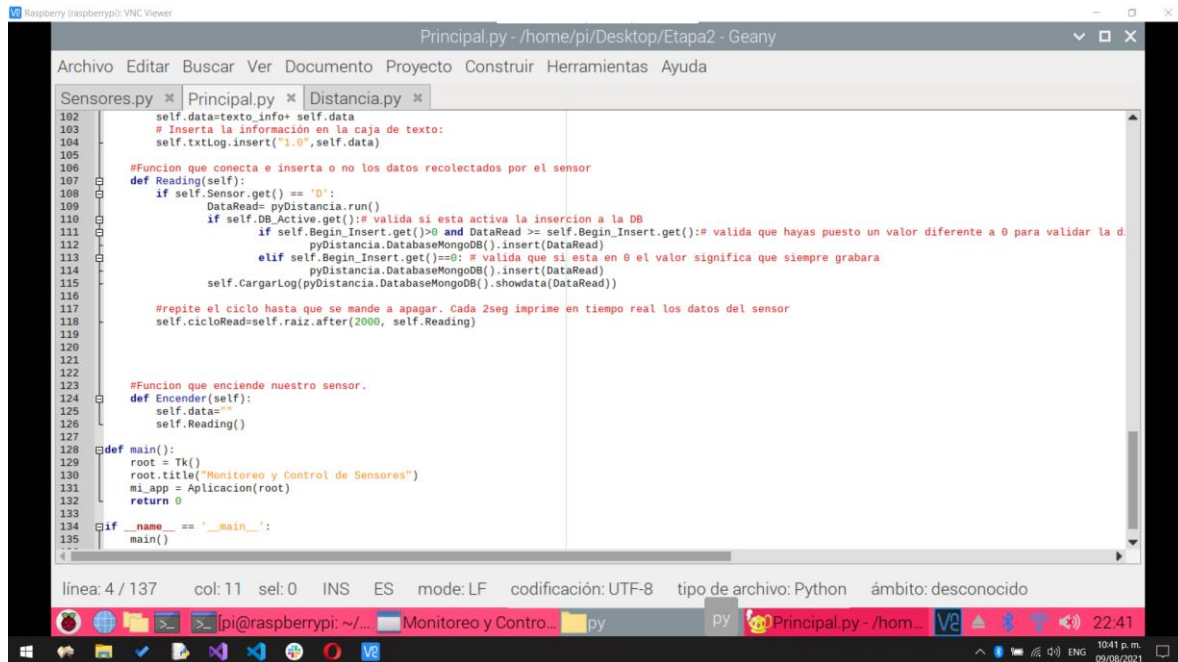
La función `setImage()` como su nombre lo indica es la encargada de cambiar la imagen en base al sensor seleccionado.

```
43 self.lblSensor.pack(side=TOP, fill=BOTH, expand=True, padx=10, pady=5)
44 self.rdbDistancia.pack(side=TOP, fill=BOTH, expand=True, padx=20, pady=5)
45 self.rdbTemperatura.pack(side=TOP, fill=BOTH, expand=True, padx=20, pady=5)
46 self.rdbMovimiento.pack(side=TOP, fill=BOTH, expand=True, padx=20, pady=5)
47 self.rdbHumedad.pack(side=TOP, fill=BOTH, expand=True, padx=20, pady=5)
48 self.lblLog.pack(side=TOP, fill=BOTH, expand=True, padx=10, pady=5)
49 self.txtHeader.pack(side=TOP, fill=BOTH, expand=True, padx=10, pady=5)
50 self.txtLog.pack(side=TOP, fill=BOTH, expand=True, padx=10, pady=5)
51 self.separ1.pack(side=TOP, fill=BOTH, expand=True, padx=5, pady=5)
52 self.btnEncender.pack(side=LEFT, fill=BOTH, expand=True, padx=10, pady=10)
53 self.btnApagar.pack(side=RIGHT, fill=BOTH, expand=True, padx=10, pady=10)
54 self.setImage()
55 self.raiz.mainloop()
56
57 #Funcion para Cargar Imagen que corresponda y traer el encabezado de cada plantilla
58 def setImage(self):
59     if self.Sensor.get()=="D":
60         self.imgSensor.config(image=self.SensorD, anchor="center", width=128)
61     elif self.Sensor.get()=="T":
62         self.imgSensor.config(image=self.SensorM, anchor="center", width=128)
63     elif self.Sensor.get()=="M":
64         self.imgSensor.config(image=self.SensorT, anchor="center", width=128)
65     elif self.Sensor.get()=="H":
66         self.imgSensor.config(image=self.SensorH, anchor="center", width=128)
67
68 def main():
69     root = Tk()
70     root.title("Monitoreo y Control de Sensores")
71     mi_app = Aplicacion(root)
72     return 0
73
74 if __name__ == "__main__":
75     main()
76
```



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

Etapas 2

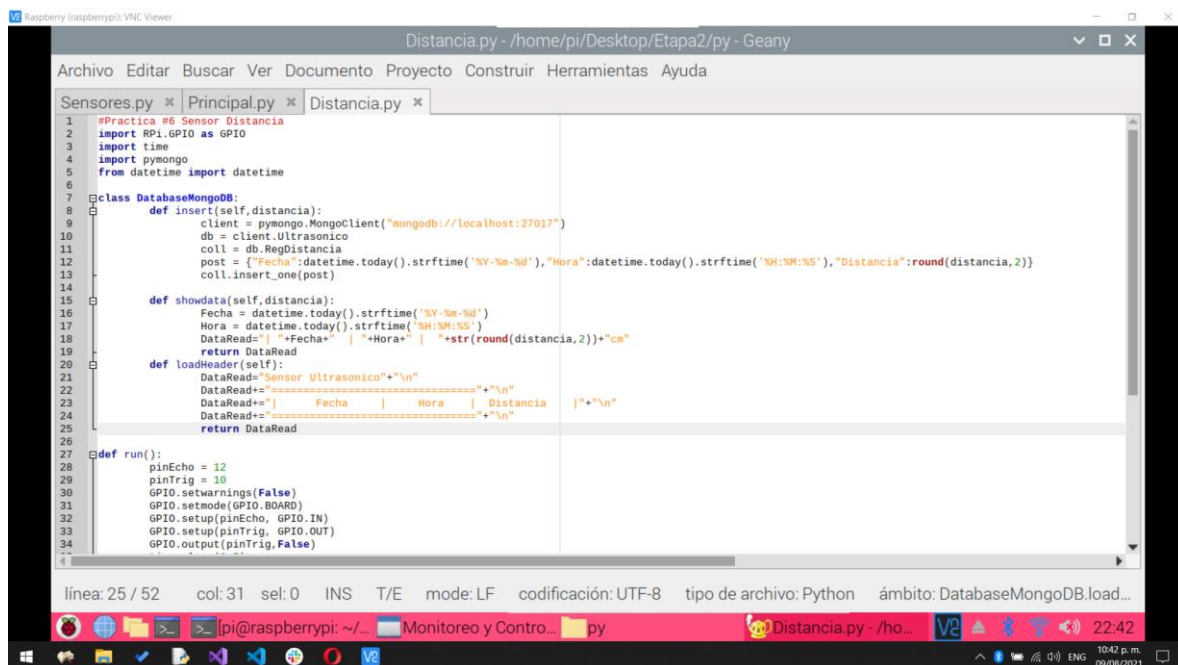


```
Principal.py - /home/pi/Desktop/Etapas2 - Geany
Archivo Editar Buscar Ver Documento Proyecto Construir Herramientas Ayuda
Sensores.py x Principal.py x Distancia.py x
102 self.data=texto.info+ self.data
103 # Inserta la información en la caja de texto:
104 self.txtLog.insert('1.0',self.data)
105
106 #Funcion que conecta e inserta o no los datos recolectados por el sensor
107 def Reading(self):
108     if self.Sensor.get() == '0':
109         DataRead= pyDistancia.run()
110         if self.DB.Active.get():# valida si esta activa la insercion a la DB
111             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
112                 pyDistancia.DatabaseMongoDB().insert(DataRead)
113             elif self.Begin_Insert.get()==0: # valida que si esta en 0 el valor significa que siempre grabara
114                 pyDistancia.DatabaseMongoDB().insert(DataRead)
115         self.CargarLog(pyDistancia.DatabaseMongoDB().showdata(DataRead))
116
117 #repite el ciclo hasta que se mande a apagar. Cada 2seg imprime en tiempo real los datos del sensor
118 self.cicloRead=self.raiz.after(2000, self.Reading)
119
120
121
122
123 #Funcion que enciende nuestro sensor.
124 def Encender(self):
125     self.data=""
126     self.Reading()
127
128 def main():
129     root = Tk()
130     root.title("Monitoreo y Control de Sensores")
131     mi_app = Aplicacion(root)
132     return 0
133
134 if __name__ == '__main__':
135     main()
136
137
línea: 4 / 137 col: 11 sel: 0 INS ES mode: LF codificación: UTF-8 tipo de archivo: Python ámbito: desconocido
1041 p.m. 09/08/2021
```

El sensor de distancia se agrega a la ventana principal a través del archivo creado.

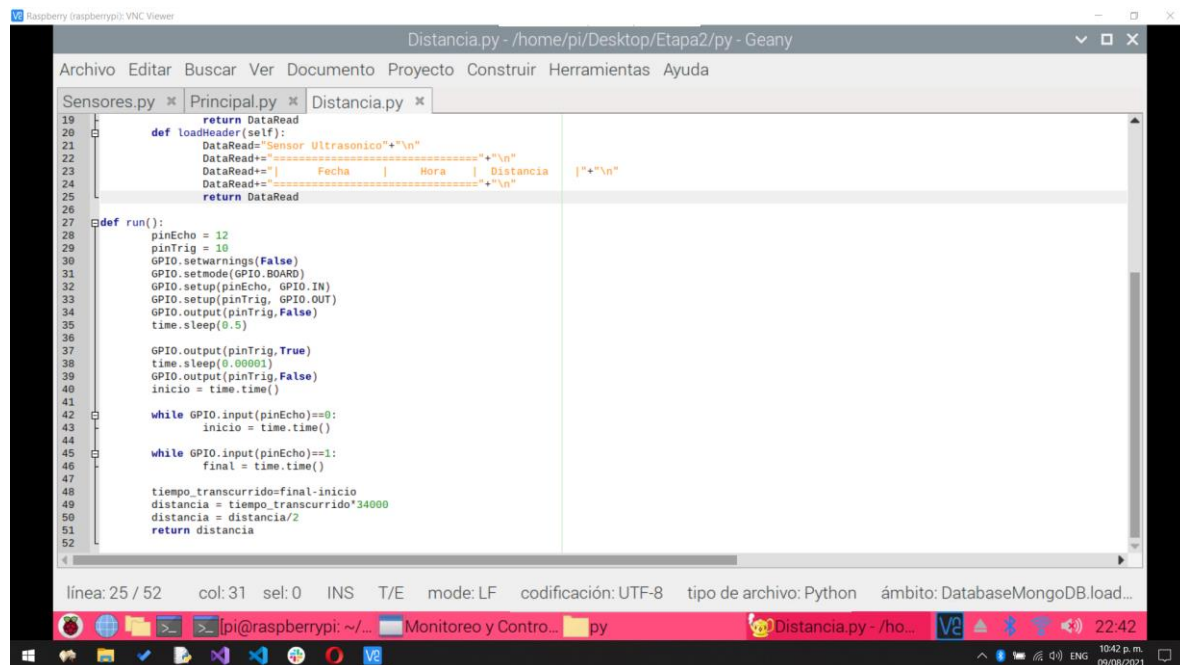
Este archivo contiene todo lo necesario para el funcionamiento del sensor:

- Envío a la MongoDB
- Impresión de registros en consola
- Funcionamiento del sensor mismo



```
Distancia.py - /home/pi/Desktop/Etapas2/py - Geany
Archivo Editar Buscar Ver Documento Proyecto Construir Herramientas Ayuda
Sensores.py x Principal.py x Distancia.py x
1 #Practica #6 Sensor Distancia
2 import RPi.GPIO as GPIO
3 import time
4 import pymongo
5 from datetime import datetime
6
7 class DatabaseMongoDB:
8     def insert(self,distancia):
9         client = pymongo.MongoClient("mongodb://localhost:27017")
10        db = client.Ultrasonico
11        coll = db.RegDistancia
12        post = {"Fecha":datetime.today().strftime('%Y-%m-%d'), "Hora":datetime.today().strftime('%H:%M:%S'), "Distancia":round(distancia,2)}
13        coll.insert_one(post)
14
15    def showdata(self,distancia):
16        Fecha = datetime.today().strftime('%Y-%m-%d')
17        Hora = datetime.today().strftime('%H:%M:%S')
18        DataRead="| "+Fecha+" | "+Hora+" | "+str(round(distancia,2))+".cm"
19        return DataRead
20
21    def loadHeader(self):
22        DataRead="Sensor Ultrasonico"+"\n"
23        DataRead+="===== "+"\n"
24        DataRead+="| Fecha | Hora | Distancia | "+"\n"
25        DataRead+="===== "+"\n"
26        return DataRead
27
28 def run():
29     pinEcho = 12
30     pinTrig = 10
31     GPIO.setwarnings(False)
32     GPIO.setmode(GPIO.BOARD)
33     GPIO.setup(pinEcho, GPIO.IN)
34     GPIO.setup(pinTrig, GPIO.OUT)
35     GPIO.output(pinTrig,False)
36
37
línea: 25 / 52 col: 31 sel: 0 INS T/E mode: LF codificación: UTF-8 tipo de archivo: Python ámbito: DatabaseMongoDB.load...
1042 p.m. 09/08/2021
```

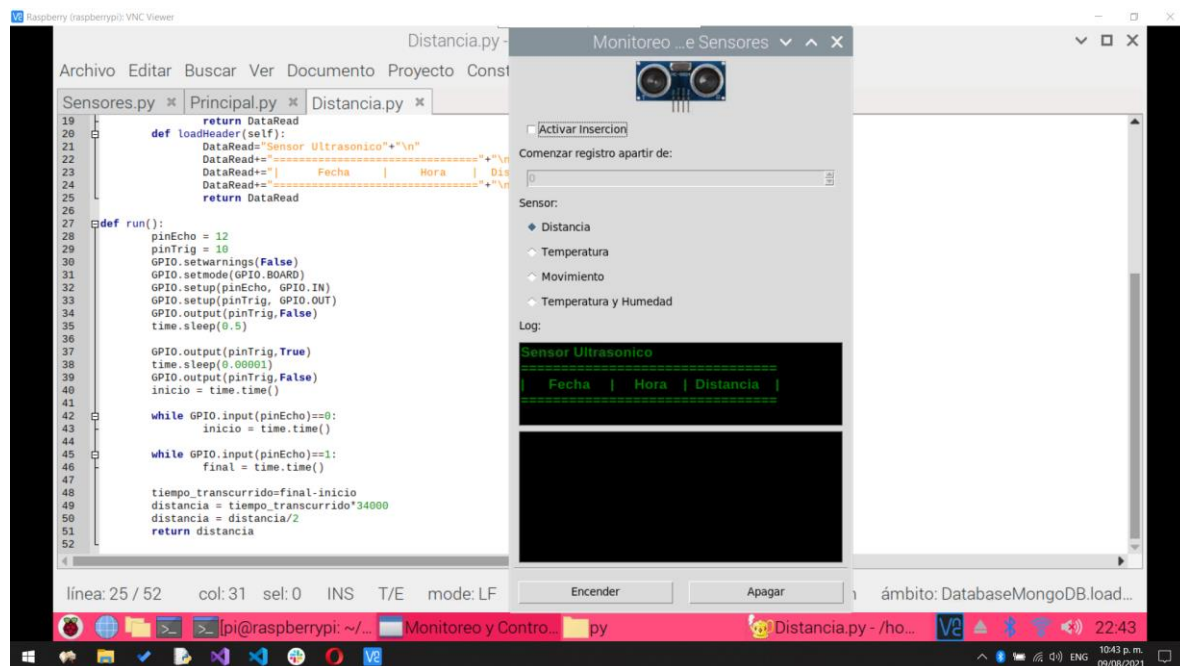

En base al nombre del archivo que recibe el sensor se es importado a la ventana principal. En este caso el archivo se llama Distancia.py por lo que en el documento siguiente se incluyen la línea `import py.Distancia as pyDistancia` para poder hacer uso del código creado.



The screenshot shows a code editor window titled 'Distancia.py - /home/pi/Desktop/Etapa2/py - Geany'. The editor has three tabs: 'Sensores.py', 'Principal.py', and 'Distancia.py'. The 'Distancia.py' tab is active, showing the following Python code:

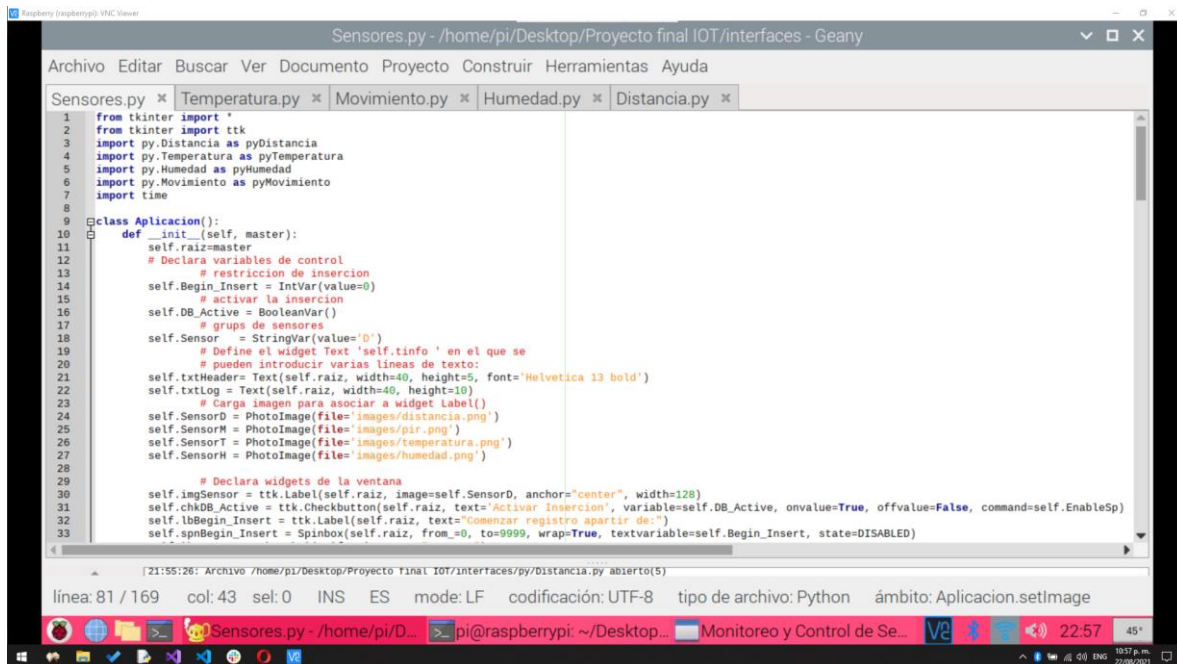
```
19     return DataRead
20
21     def loadHeader(self):
22         DataRead="Sensor Ultrasonico"+"\\n"
23         DataRead+="=====\\n"
24         DataRead+="| Fecha | Hora | Distancia |"+"\\n"
25         return DataRead
26
27 def run():
28     pinEcho = 12
29     pinTrig = 10
30     GPIO.setwarnings(False)
31     GPIO.setmode(GPIO.BOARD)
32     GPIO.setup(pinEcho, GPIO.IN)
33     GPIO.setup(pinTrig, GPIO.OUT)
34     GPIO.output(pinTrig,False)
35     time.sleep(0.5)
36
37     GPIO.output(pinTrig,True)
38     time.sleep(0.00001)
39     GPIO.output(pinTrig,False)
40     inicio = time.time()
41
42     while GPIO.input(pinEcho)==0:
43         inicio = time.time()
44
45     while GPIO.input(pinEcho)==1:
46         final = time.time()
47
48     tiempo_transcurrido=final-inicio
49     distancia = tiempo_transcurrido*34000
50     distancia = distancia/2
51     return distancia
52
```

The status bar at the bottom indicates 'línea: 25 / 52', 'col: 31', 'sel: 0', 'INS', 'T/E', 'mode: LF', 'codificación: UTF-8', 'tipo de archivo: Python', 'ámbito: DatabaseMongoDB.load...', and the system clock shows '10:42 p.m. 09/08/2021'.



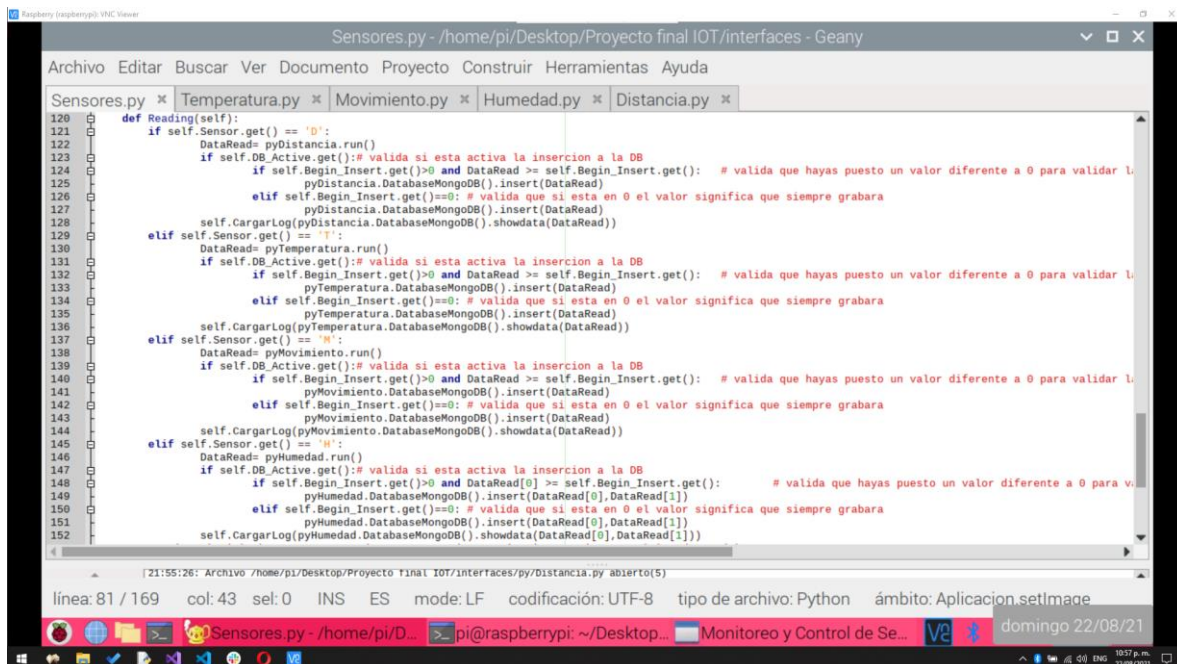
Una vez realizados los cambios y hechas las conexiones correspondientes al sensor este ya puede ser utilizado.

Etapla 3



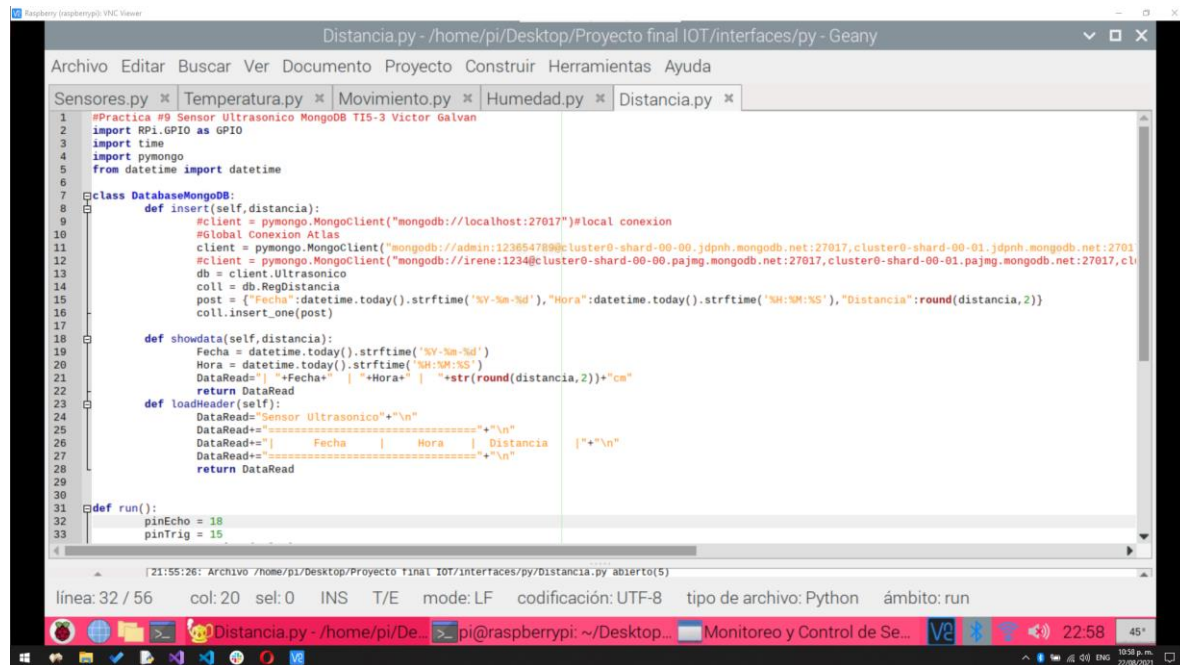
```
1 from tkinter import *
2 from tkinter import ttk
3 import py.Distancia as pyDistancia
4 import py.Temperatura as pyTemperatura
5 import py.Humedad as pyHumedad
6 import py.Movimiento as pyMovimiento
7 import time
8
9 class Aplicacion():
10     def __init__(self, master):
11         self.raiz=master
12         # Declara variables de control
13         # restriccion de insercion
14         self.Begin_Insert = IntVar(value=0)
15         # activar la insercion
16         self.DB_Active = BooleanVar()
17         # grupos de sensores
18         self.Sensor = StringVar(value='0')
19         # Define el widget Text 'self.tinfo' en el que se
20         # pueden introducir varias lineas de texto:
21         self.txtHeader = Text(self.raiz, width=40, height=5, font='Helvetica 13 bold')
22         self.txtLog = Text(self.raiz, width=40, height=10)
23         # Carga imagen para asociar a widget Label()
24         self.SensorD = PhotoImage(file='images/distancia.png')
25         self.SensorM = PhotoImage(file='images/pir.png')
26         self.SensorT = PhotoImage(file='images/temperatura.png')
27         self.SensorH = PhotoImage(file='images/humedad.png')
28
29         # Declara widgets de la ventana
30         self.imgSensor = ttk.Label(self.raiz, image=self.SensorD, anchor="center", width=128)
31         self.chkDB_Active = ttk.Checkbutton(self.raiz, text='Activar Insercion', variable=self.DB_Active, onvalue=True, offvalue=False, command=self.EnableSp)
32         self.lbBegin_Insert = ttk.Label(self.raiz, text="Comenzar registro apartir de:")
33         self.spnBegin_Insert = Spinbox(self.raiz, from_=0, to=9999, wrap=True, textvariable=self.Begin_Insert, state=DISABLED)
```

Se importan los sensores restantes al condigo de la ventana principal de la misma manera que en la etapa anterior. Otorgándole a cada uno su nombre correspondiente para así poderlos implementar de forma correcta y utilizar todas las funciones de la ventana principal.



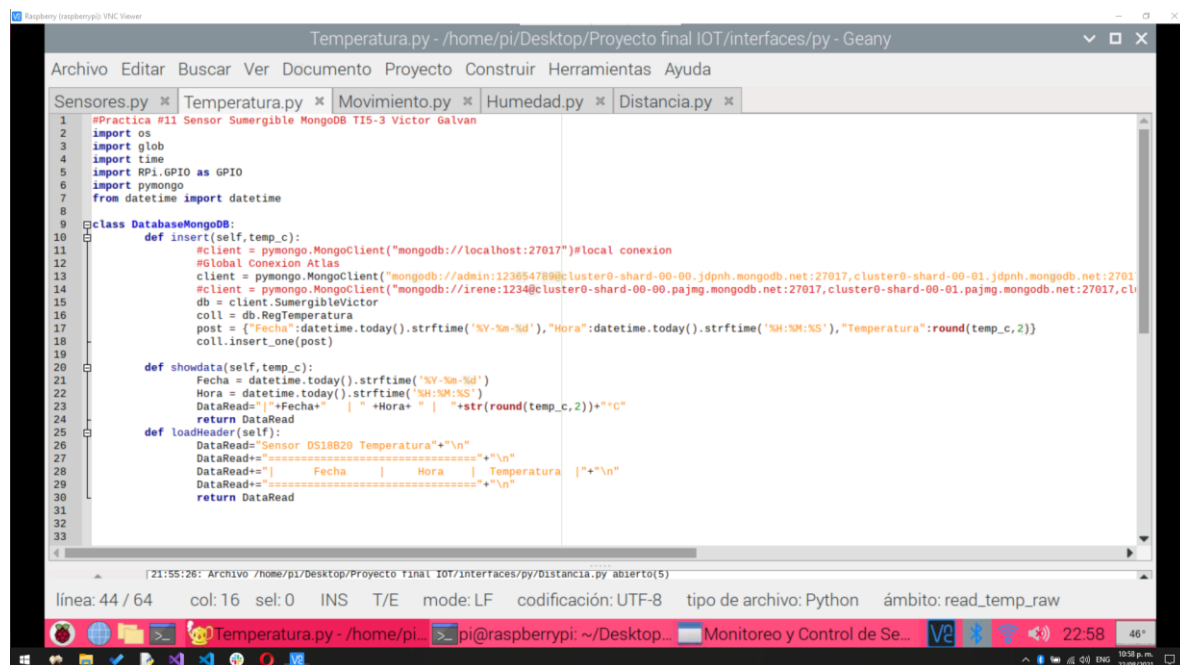
```
120 def Reading(self):
121     if self.Sensor.get() == 'D':
122         DataRead= pyDistancia.run()
123         if self.DB_Active.get():# valida si esta activa la insercion a la DB
124             if self.Begin_Insert.get()>0 and DataRead >= self.Begin_Insert.get(): # valida que hayas puesto un valor diferente a 0 para validar l
125                 pyDistancia.DatabaseMongoDB().insert(DataRead)
126             elif self.Begin_Insert.get()==0: # valida que si esta en 0 el valor significa que siempre grabara
127                 pyDistancia.DatabaseMongoDB().insert(DataRead)
128         self.CargarLog(pyDistancia.DatabaseMongoDB().showdata(DataRead))
129     elif self.Sensor.get() == 'T':
130         DataRead= pyTemperatura.run()
131         if self.DB_Active.get():# valida si esta activa la insercion a la DB
132             if self.Begin_Insert.get()>0 and DataRead >= self.Begin_Insert.get(): # valida que hayas puesto un valor diferente a 0 para validar l
133                 pyTemperatura.DatabaseMongoDB().insert(DataRead)
134             elif self.Begin_Insert.get()==0: # valida que si esta en 0 el valor significa que siempre grabara
135                 pyTemperatura.DatabaseMongoDB().insert(DataRead)
136         self.CargarLog(pyTemperatura.DatabaseMongoDB().showdata(DataRead))
137     elif self.Sensor.get() == 'M':
138         DataRead= pyMovimiento.run()
139         if self.DB_Active.get():# valida si esta activa la insercion a la DB
140             if self.Begin_Insert.get()>0 and DataRead >= self.Begin_Insert.get(): # valida que hayas puesto un valor diferente a 0 para validar l
141                 pyMovimiento.DatabaseMongoDB().insert(DataRead)
142             elif self.Begin_Insert.get()==0: # valida que si esta en 0 el valor significa que siempre grabara
143                 pyMovimiento.DatabaseMongoDB().insert(DataRead)
144         self.CargarLog(pyMovimiento.DatabaseMongoDB().showdata(DataRead))
145     elif self.Sensor.get() == 'H':
146         DataRead= pyHumedad.run()
147         if self.DB_Active.get():# valida si esta activa la insercion a la DB
148             if self.Begin_Insert.get()>0 and DataRead[0] >= self.Begin_Insert.get(): # valida que hayas puesto un valor diferente a 0 para v
149                 pyHumedad.DatabaseMongoDB().insert(DataRead[0],DataRead[1])
150             elif self.Begin_Insert.get()==0: # valida que si esta en 0 el valor significa que siempre grabara
151                 pyHumedad.DatabaseMongoDB().insert(DataRead[0],DataRead[1])
152         self.CargarLog(pyHumedad.DatabaseMongoDB().showdata(DataRead[0],DataRead[1]))
```

La activación de los sensores es realizada en base a la selección hecha por el usuario. Si el usuario selecciona el sensor de movimiento entonces a los métodos que se accederán serán a los contenidos dentro del archivo Movimiento.py y así respectivamente con cada sensor implementado dentro del proyecto.



```
1 #Practica #9 Sensor Ultrasonico MongoDB TI5-3 Victor Galvan
2 import RPi.GPIO as GPIO
3 import time
4 import pymongo
5 from datetime import datetime
6
7 class DatabaseMongoDB:
8     def insert(self, distancia):
9         #client = pymongo.MongoClient("mongodb://localhost:27017")#local conexion
10        #global Conexion Atlas
11        client = pymongo.MongoClient("mongodb://admin:123654789@cluster0-shard-00-00.jdpnh.mongodb.net:27017,cluster0-shard-00-01.jdpnh.mongodb.net:27017,cluster0-shard-00-02.jdpnh.mongodb.net:27017?ssl=true&replicaSet=atlas060201-shard-00-00&authSource=admin")
12        db = client.Ultrasonico
13        coll = db.RegDistancia
14        post = {'Fecha':datetime.today().strftime('%Y-%m-%d'), 'Hora':datetime.today().strftime('%H:%M:%S'), 'Distancia':round(distancia,2)}
15        coll.insert_one(post)
16
17     def showdata(self, distancia):
18        Fecha = datetime.today().strftime('%Y-%m-%d')
19        Hora = datetime.today().strftime('%H:%M:%S')
20        DataRead="| "+Fecha+" | "+Hora+" | "+str(round(distancia,2))+"cm"
21        return DataRead
22
23     def loadHeader(self):
24        DataRead="Sensor Ultrasonico"+"\\n"
25        DataRead+="=====\\n"
26        DataRead+="| Fecha | Hora | Distancia |"+"\\n"
27        DataRead+="=====\\n"
28        return DataRead
29
30
31
32 def run():
33     pinEcho = 18
34     pinTrig = 15
```

línea: 32 / 56 col: 20 sel: 0 INS T/E mode: LF codificación: UTF-8 tipo de archivo: Python ámbito: run



```
1 #Practica #11 Sensor Sumergible MongoDB TI5-3 Victor Galvan
2 import os
3 import glob
4 import time
5 import RPi.GPIO as GPIO
6 import pymongo
7 from datetime import datetime
8
9 class DatabaseMongoDB:
10     def insert(self, temp_c):
11         #client = pymongo.MongoClient("mongodb://localhost:27017")#local conexion
12         #global Conexion Atlas
13         client = pymongo.MongoClient("mongodb://admin:123654789@cluster0-shard-00-00.jdpnh.mongodb.net:27017,cluster0-shard-00-01.jdpnh.mongodb.net:27017,cluster0-shard-00-02.jdpnh.mongodb.net:27017?ssl=true&replicaSet=atlas060201-shard-00-00&authSource=admin")
14         db = client.SumergibleVictor
15         coll = db.RegTemperatura
16         post = {'Fecha':datetime.today().strftime('%Y-%m-%d'), 'Hora':datetime.today().strftime('%H:%M:%S'), 'Temperatura':round(temp_c,2)}
17         coll.insert_one(post)
18
19     def showdata(self, temp_c):
20        Fecha = datetime.today().strftime('%Y-%m-%d')
21        Hora = datetime.today().strftime('%H:%M:%S')
22        DataRead="| "+Fecha+" | "+Hora+" | "+str(round(temp_c,2))+"°C"
23        return DataRead
24
25     def loadHeader(self):
26        DataRead="Sensor DS18B20 Temperatura"+"\\n"
27        DataRead+="=====\\n"
28        DataRead+="| Fecha | Hora | Temperatura |"+"\\n"
29        DataRead+="=====\\n"
30        return DataRead
31
32
33
```

línea: 44 / 64 col: 16 sel: 0 INS T/E mode: LF codificación: UTF-8 tipo de archivo: Python ámbito: read_temp_raw

```
1 #Practica #10 Sensor PIR MongoDB TIIS-3 Victor Galvan
2 import RPi.GPIO as GPIO
3 import time
4 import pymongo
5 from datetime import datetime
6
7 class DatabaseMongoDB:
8     def insert(self,estado):
9         #client = pymongo.MongoClient("mongodb://localhost:27017")#local conexion
10        #Global Conexion Atlas
11        client = pymongo.MongoClient("mongodb://admin:123654789@cluster0-shard-00-00-jdph.mongodb.net:27017,cluster0-shard-00-01-jdph.mongodb.net:27017,cluster0-shard-00-02-jdph.mongodb.net:27017?ssl=true&replicaSet=atlas-shard-0&authSource=admin")
12        #client = pymongo.MongoClient("mongodb://irene:1234@cluster0-shard-00-00-pajmg.mongodb.net:27017,cluster0-shard-00-01-pajmg.mongodb.net:27017,cluster0-shard-00-02-pajmg.mongodb.net:27017?ssl=true&replicaSet=atlas-shard-0&authSource=admin")
13        db = client.Pir
14        coll = db.RegMovimiento
15        post = {"Fecha":datetime.today().strftime('%Y-%m-%d'),"Hora":datetime.today().strftime('%H:%M:%S'),"Movimiento":estado}
16        coll.insert_one(post)
17
18    def showdata(self,estado):
19        Fecha = datetime.today().strftime('%Y-%m-%d')
20        Hora = datetime.today().strftime('%H:%M:%S')
21        DataRead="| Fecha+ | Hora+ | "+ ("No Movimiento" if estado==0 else "Movimiento")
22        return DataRead
23
24    def loadHeader(self):
25        DataRead="Sensor PIR Movimiento"+ "\n"
26        DataRead+="=====+ "\n"
27        DataRead+="| Fecha | Hora | Movimiento | "+ "\n"
28        DataRead+="=====+ "\n"
29        return DataRead
30
31 estado = 0
32
33
```

[21:55:26: Archivo /home/pi/Desktop/Proyecto final IOT/interfaces/py/distancia.py abierto(5)]

línea: 15 / 48 col: 113 sel: 0 INS TAB mode: LF codificación: UTF-8 tipo de archivo: Python ámbito: DatabaseMongoDB...

```
1 #Practica #12 Sensor Temperatura y Humedad MongoDB TIIS-3 Victor Galvan
2 import RPi.GPIO as GPIO
3 import Adafruit_DHT
4 import time
5 import pymongo
6 from datetime import datetime
7
8 class DatabaseMongoDB:
9     def insert(self,temp,humedad):
10        #client = pymongo.MongoClient("mongodb://localhost:27017")#local conexion
11        #Global Conexion Atlas
12        client = pymongo.MongoClient("mongodb://admin:123654789@cluster0-shard-00-00-jdph.mongodb.net:27017,cluster0-shard-00-01-jdph.mongodb.net:27017,cluster0-shard-00-02-jdph.mongodb.net:27017?ssl=true&replicaSet=atlas-shard-0&authSource=admin")
13        #client = pymongo.MongoClient("mongodb://irene:1234@cluster0-shard-00-00-pajmg.mongodb.net:27017,cluster0-shard-00-01-pajmg.mongodb.net:27017,cluster0-shard-00-02-pajmg.mongodb.net:27017?ssl=true&replicaSet=atlas-shard-0&authSource=admin")
14        db = client.AmbienteVictor
15        coll = db.RegHumedad
16
17        post = {"Fecha":datetime.today().strftime('%Y-%m-%d'),"Hora":datetime.today().strftime('%H:%M:%S'),"Temperatura":temp,"Humedad":humedad}
18        coll.insert_one(post)
19
20    def showdata(self,temp,humedad):
21        Fecha = datetime.today().strftime('%Y-%m-%d')
22        Hora = datetime.today().strftime('%H:%M:%S')
23        DataRead="| Fecha+ | Hora+ | "+f"{temp:.2f}°C"+ " | "+f"{humedad:.2f}%"
24        return DataRead
25
26    def loadHeader(self):
27        DataRead="Sensor Temperatura y Humedad"+ "\n"
28        DataRead+="=====+ "\n"
29        DataRead+="| Fecha | Hora | Temperatura | Humedad | "+ "\n"
30        DataRead+="=====+ "\n"
31        return DataRead
32
33
```

[21:55:26: Archivo /home/pi/Desktop/Proyecto final IOT/interfaces/py/distancia.py abierto(5)]

línea: 36 / 48 col: 25 sel: 0 INS TAB mode: LF codificación: UTF-8 tipo de archivo: Python ámbito: run

Una vez creados los cuatro archivos de los sensores utilizados junto con las conexiones hechas se puede utilizar el programa completamente con todas sus funciones disponibles.

Sensores.py - /home/pi/Desktop/Proyecto final IoT

Archivo Editar Buscar Ver Documento Proyecto Construir Herramientas Ayuda

Sensores.py x Temperatura.py x Movimiento.py x Humedad.py x Distancia.py

```
1 from tkinter import *
2 from tkinter import ttk
3 import py.Distancia as pyDistancia
4 import py.Temperatura as pyTemperatura
5 import py.Humedad as pyHumedad
6 import py.Movimiento as pyMovimiento
7 import time
8
9 class Aplicacion():
10     def __init__(self, master):
11         self.raiz=master
12         # Declara variables de control
13         # restriccion de insercion
14         self.Begin_Insert = IntVar(value=0)
15         # activar la insercion
16         self.DB_Active = BooleanVar()
17         # grupos de sensores
18         self.Sensor = StringVar(value='0')
19         # Define el widget Text 'self.tinfo' en el que se
20         # pueden introducir varias lineas de texto:
21         self.txtHeader= Text(self.raiz, width=40, height=5, font='Helvetica 13 bold')
22         self.txtLog = Text(self.raiz, width=40, height=10)
23         # Carga imagen para asociar a widget Label()
24         self.SensorD = PhotoImage(file='images/distancia.png')
25         self.SensorM = PhotoImage(file='images/pir.png')
26         self.SensorT = PhotoImage(file='images/temperatura.png')
27         self.SensorH = PhotoImage(file='images/humedad.png')
28
29         # Declara widgets de la ventana
30         self.imgSensor = ttk.Label(self.raiz, image=self.SensorD, anchor="center", width=128)
31         self.chkDB_Active = ttk.Checkbutton(self.raiz, text='Activar Insercion', variable=self.DB_Active)
32         self.lbBegin_Insert = ttk.Label(self.raiz, text="Comenzar registro apartir de:")
33         self.spnBegin_Insert = Spinbox(self.raiz, from_=0, to=9999, wrap=True, textvariable=self.Begin_Insert)
```

[21:55:26: Archivo /home/pi/Desktop/Proyecto final IoT/interfases/py/distancia.py abierto(5)]

línea: 81 / 169 col: 43 sel: 0 INS ES mode: LF codificación: UTF-8 tip: Encender Apagar mage

Monitoreo de Sensores

Activar Insercion

Comenzar registro apartir de: 4

Sensor:

- Distancia
- Temperatura
- Movimiento
- Temperatura y Humedad

Log:

Sensor Ultrasonico

Fecha	Hora	Distancia
2021-08-22	23:14:26	3.36cm
2021-08-22	23:14:24	3.3cm
2021-08-22	23:14:21	3.36cm
2021-08-22	23:14:19	3.31cm

lf.EnableSp)

pi@raspberrypi: ~/Desktop/... Monitoreo y Control de Se... 23:14 45°

Sensores.py - /home/pi/Desktop/Proyecto final IoT

Archivo Editar Buscar Ver Documento Proyecto Construir Herramientas Ayuda

Sensores.py x Temperatura.py x Movimiento.py x Humedad.py x Distancia.py

```
1 from tkinter import *
2 from tkinter import ttk
3 import py.Distancia as pyDistancia
4 import py.Temperatura as pyTemperatura
5 import py.Humedad as pyHumedad
6 import py.Movimiento as pyMovimiento
7 import time
8
9 class Aplicacion():
10     def __init__(self, master):
11         self.raiz=master
12         # Declara variables de control
13         # restriccion de insercion
14         self.Begin_Insert = IntVar(value=0)
15         # activar la insercion
16         self.DB_Active = BooleanVar()
17         # grupos de sensores
18         self.Sensor = StringVar(value='0')
19         # Define el widget Text 'self.tinfo' en el que se
20         # pueden introducir varias lineas de texto:
21         self.txtHeader= Text(self.raiz, width=40, height=5, font='Helvetica 13 bold')
22         self.txtLog = Text(self.raiz, width=40, height=10)
23         # Carga imagen para asociar a widget Label()
24         self.SensorD = PhotoImage(file='images/distancia.png')
25         self.SensorM = PhotoImage(file='images/pir.png')
26         self.SensorT = PhotoImage(file='images/temperatura.png')
27         self.SensorH = PhotoImage(file='images/humedad.png')
28
29         # Declara widgets de la ventana
30         self.imgSensor = ttk.Label(self.raiz, image=self.SensorD, anchor="center", width=128)
31         self.chkDB_Active = ttk.Checkbutton(self.raiz, text='Activar Insercion', variable=self.DB_Active)
32         self.lbBegin_Insert = ttk.Label(self.raiz, text="Comenzar registro apartir de:")
33         self.spnBegin_Insert = Spinbox(self.raiz, from_=0, to=9999, wrap=True, textvariable=self.Begin_Insert)
```

[21:55:26: Archivo /home/pi/Desktop/Proyecto final IoT/interfases/py/distancia.py abierto(5)]

línea: 81 / 169 col: 43 sel: 0 INS ES mode: LF codificación: UTF-8 tip: Encender Apagar mage

Monitoreo de Sensores

Activar Insercion

Comenzar registro apartir de: 4

Sensor:

- Distancia
- Temperatura
- Movimiento
- Temperatura y Humedad

Log:

Sensor DS18B20 Temperatura

Fecha	Hora	Temperatura
2021-08-22	23:14:48	23.81°C
2021-08-22	23:14:45	23.75°C

lf.EnableSp)

pi@raspberrypi: ~/Desktop/... Monitoreo y Control de Se... domingo 22/08/21 23:14

Sensores.py - /home/pi/Desktop/Proyecto final IoT/interfases/py/distancia.py abierto(5)

Archivo Editar Buscar Ver Documento Proyecto Construir Herramientas Ayuda

Sensores.py x Temperatura.py x Movimiento.py x Humedad.py x Distancia.py

```

1 from tkinter import *
2 from tkinter import ttk
3 import py.Distance as pyDistancia
4 import py.Temperatura as pyTemperatura
5 import py.Humedad as pyHumedad
6 import py.Movimiento as pyMovimiento
7 import time
8
9 class Aplicacion():
10     def __init__(self, master):
11         self.raiz=master
12         # Declara variables de control
13         # restricción de inserción
14         self.Begin_Insert = IntVar(value=0)
15         # activar la inserción
16         self.DB_Active = BooleanVar()
17         # grupos de sensores
18         self.Sensor = StringVar(value='0')
19         # Define el widget Text 'self.tinfo' en el que se
20         # pueden introducir varias líneas de texto:
21         self.txtHeader= Text(self.raiz, width=40, height=5, font='Helvetica 13 bold')
22         self.txtLog = Text(self.raiz, width=40, height=10)
23         # Carga imagen para asociar a widget Label()
24         self.SensorD = PhotoImage(file='images/distancia.png')
25         self.SensorM = PhotoImage(file='images/pir.png')
26         self.SensorT = PhotoImage(file='images/temperatura.png')
27         self.SensorH = PhotoImage(file='images/humedad.png')
28
29         # Declara widgets de la ventana
30         self.imgSensor = ttk.Label(self.raiz, image=self.SensorD, anchor="center", width=128)
31         self.chkDB_Active = ttk.Checkbutton(self.raiz, text='Activar Inserción', variable=self.DB_Active)
32         self.lbBegin_Insert = ttk.Label(self.raiz, text='Comenzar registro a partir de:')
33         self.spnBegin_Insert = Spinbox(self.raiz, from_=0, to=9999, wrap=True, textvariable=self.Begin_Insert)

```

[21:55:26: Archivo /home/pi/Desktop/Proyecto final IoT/interfases/py/distancia.py abierto(5)]

línea: 81 / 169 col: 43 sel: 0 INS ES mode: LF codificación: UTF-8 tip: Encender Apagar image

Monitoreo y Control de Se...

Monitoreo de Sensores

Activar Inserción

Comenzar registro a partir de:

Sensor:

- Distancia
- Temperatura
- Movimiento
- Temperatura y Humedad

Log:

Sensor PIR Movimiento

Fecha	Hora	Movimiento
2021-08-22	23:15:20	Movimiento
2021-08-22	23:15:18	No Movimiento
2021-08-22	23:15:16	No Movimiento
2021-08-22	23:15:14	Movimiento
2021-08-22	23:15:12	No Movimiento
2021-08-22	23:15:10	No Movimiento

lf.EnableSp

Sensores.py - /home/pi/Desktop/Proyecto final IoT/interfases/py/distancia.py abierto(5)

Archivo Editar Buscar Ver Documento Proyecto Construir Herramientas Ayuda

Sensores.py x Temperatura.py x Movimiento.py x Humedad.py x Distancia.py

```

1 from tkinter import *
2 from tkinter import ttk
3 import py.Distance as pyDistancia
4 import py.Temperatura as pyTemperatura
5 import py.Humedad as pyHumedad
6 import py.Movimiento as pyMovimiento
7 import time
8
9 class Aplicacion():
10     def __init__(self, master):
11         self.raiz=master
12         # Declara variables de control
13         # restricción de inserción
14         self.Begin_Insert = IntVar(value=0)
15         # activar la inserción
16         self.DB_Active = BooleanVar()
17         # grupos de sensores
18         self.Sensor = StringVar(value='0')
19         # Define el widget Text 'self.tinfo' en el que se
20         # pueden introducir varias líneas de texto:
21         self.txtHeader= Text(self.raiz, width=40, height=5, font='Helvetica 13 bold')
22         self.txtLog = Text(self.raiz, width=40, height=10)
23         # Carga imagen para asociar a widget Label()
24         self.SensorD = PhotoImage(file='images/distancia.png')
25         self.SensorM = PhotoImage(file='images/pir.png')
26         self.SensorT = PhotoImage(file='images/temperatura.png')
27         self.SensorH = PhotoImage(file='images/humedad.png')
28
29         # Declara widgets de la ventana
30         self.imgSensor = ttk.Label(self.raiz, image=self.SensorD, anchor="center", width=128)
31         self.chkDB_Active = ttk.Checkbutton(self.raiz, text='Activar Inserción', variable=self.DB_Active)
32         self.lbBegin_Insert = ttk.Label(self.raiz, text='Comenzar registro a partir de:')
33         self.spnBegin_Insert = Spinbox(self.raiz, from_=0, to=9999, wrap=True, textvariable=self.Begin_Insert)

```

[21:55:26: Archivo /home/pi/Desktop/Proyecto final IoT/interfases/py/distancia.py abierto(5)]

línea: 81 / 169 col: 43 sel: 0 INS ES mode: LF codificación: UTF-8 tip: Encender Apagar image

Monitoreo y Control de Se...

Monitoreo de Sensores

Activar Inserción

Comenzar registro a partir de:

Sensor:

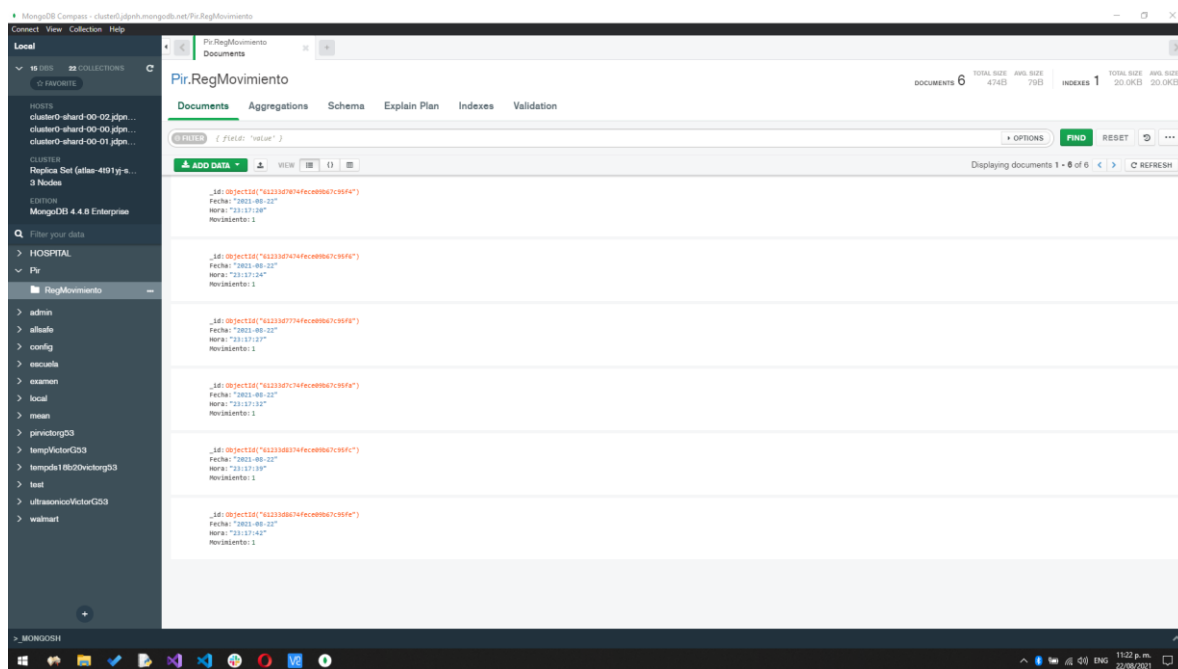
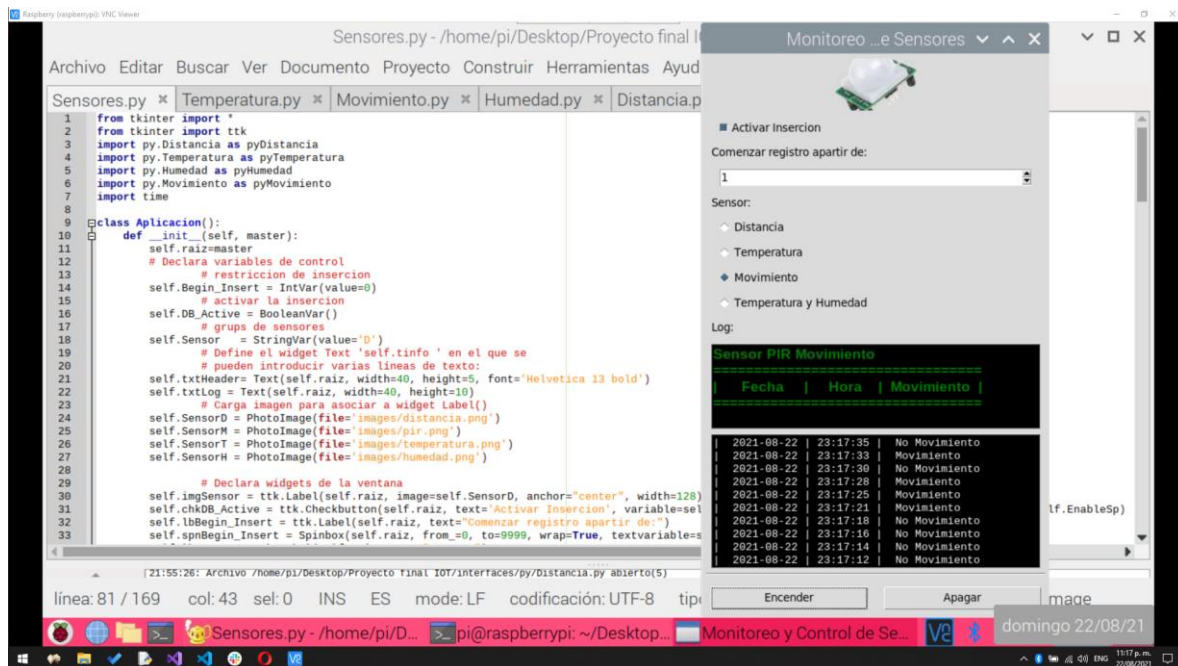
- Distancia
- Temperatura
- Movimiento
- Temperatura y Humedad

Log:

Sensor Temperatura y Humedad

Fecha	Hora	Temperatura	Humedad
2021-08-22	23:16:32	25.00°C	61.00%
2021-08-22	23:16:30	No Movimiento	
2021-08-22	23:16:28	25.00°C	61.00%
2021-08-22	23:16:26	25.00°C	61.00%
2021-08-22	23:16:24	25.00°C	62.00%
2021-08-22	23:16:22	24.00°C	62.00%
2021-08-22	23:16:20	25.00°C	63.00%
2021-08-22	23:16:18	25.00°C	63.00%
2021-08-22	23:16:16	25.00°C	63.00%
2021-08-22	23:16:14	25.00°C	63.00%
2021-08-22	23:16:12	25.00°C	63.00%
2021-08-22	23:16:10	25.00°C	63.00%
2021-08-22	23:16:08	25.00°C	63.00%
2021-08-22	23:16:06	25.00°C	63.00%
2021-08-22	23:16:04	25.00°C	63.00%
2021-08-22	23:16:02	25.00°C	63.00%
2021-08-22	23:16:00	25.00°C	63.00%

lf.EnableSp



La inserción a la base de datos como se puede apreciar en la imagen anterior es exitosa.

La aplicación en Tkinter ha sido terminada. Ya es posible el utilizar los cuatro sensores dentro de una interfaz gráfica agradable para un usuario inexperto en electrónica y programación.

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.