



Universidad Nacional Autónoma de México

Facultad de Ingeniería

Laboratorio de Fundamentos de Sistemas Embebidos

Profesora: M.A. Ayesha Sagrario Román García

Documentación Proyecto

Invernadero Domestico

Grupo: 3

Alumno: Sánchez Escobar Fernando

Semestre 2021-2

Fecha de entrega: 08 de agosto del 2021

## Contenido

Introducción .....	2
Descripción del proyecto.....	3
Materiales .....	4
Costos .....	6
Diseño.....	6
Arquitectura General .....	7
Arquitectura del Software.....	8
Identificación de los elementos que funcionan con IoT .....	8
Google Assistant.....	8
Bot de Telegram .....	9
Implementación .....	9
Invernadero montado .....	9
Funcionamiento del Bot de Telegram .....	13
Funcionamiento Blue Dot.....	19
Códigos usados.....	21
greenHouse.py .....	21
creacionBD.py .....	21
interfaceBot.py.....	21
greenH_sensors.py.....	23
greenH_actions.py .....	26
showluces.py .....	29
Evaluación de viabilidad .....	30
Conclusión .....	30
Fuentes de consulta .....	31

## Introducción

Se entiende por invernadero a un lugar cerrado, estático y accesible a pie, este habitualmente cuenta con una cubierta exterior traslúcida de vidrio o de plástico, dentro del cual se puede obtener un microclima mediante el control de la temperatura, la humedad y de otros factores ambientales. Del mismo este puede contar con sistemas automáticos de riego y ventilación, lo cual se utiliza para la producción de cultivos de forma controlada. Mediante el avance de la tecnología el control de los factores del microclima es más sencillo de implementar, además de poder administrar estos controles de manera remota a través de Internet, siendo que no sea necesario el estar presente físicamente en el invernadero, haciendo esto posible gracias a la unión a la domótica.



*Ilustración 1. Invernadero, imagen recuperada de [https://www.freepik.es/vector-premium/ilustracion-isometrica-3d-invernadero-cultivo-plantulas-invernadero\\_5963001.htm](https://www.freepik.es/vector-premium/ilustracion-isometrica-3d-invernadero-cultivo-plantulas-invernadero_5963001.htm)*

La domótica es el conjunto de tecnologías aplicadas al control y la automatización inteligente de una vivienda permitiendo una gestión eficiente tanto en el uso de la energía como de los materiales, además de aportar seguridad y confort teniendo una comunicación entre el usuario y el sistema. En otras palabras “la integración de la tecnología en el diseño inteligente de un recinto cerrado es lo que se conoce como domótica”.



*Ilustración 2. Ejemplo sistema domótico, recuperado de <https://e-ficiencia.com/domotica-que-es-y-como-funciona/>*

En una instalación domótica los diferentes dispositivos que conforman el sistema domótico hacen uso de una red Wi-Fi con el fin de enviar y recibir información, así como para establecer una conexión con el usuario.

Los terminales siempre son dispositivos tales como electrodomésticos, dispositivos de iluminación, equipos de climatización, ventilación, en otras palabras cualquier equipo susceptible de disponer de una inteligencia o capacidad de comunicación con el sistema central programable, introduciendo una interfaz para su control. Los sensores dentro del conjunto van a recabar información sensible y la unidad central decidirá qué acciones realizar en relación a la información proporcionada por dichos sensores.

## Descripción del proyecto

El presente proyecto tiene como objetivo la construcción de un invernadero doméstico en el cual se implementarán las prácticas elaboradas a lo largo del semestre, uniendo todos los códigos en módulos específicos que permitan obtener información del invernadero y realizar acciones de manera automática dependiendo de la información obtenida, así como también permitir que el usuario sea capaz de obtener dicha información y se le muestre en una interfaz la cual también le permita ejecutar acciones en el invernadero.

Las acciones que se realizarán de manera automática en el invernadero dependiendo de la información obtenida por parte de los sensores serán las siguientes:

- ♣ Mediante un sensor de humedad de tierra se estará monitoreando la humedad de esta de tal manera que llegase a detectar que se encuentra seca enviará una señal que active la bomba de agua y riegue la tierra.
- ♣ Mediante el uso de un sensor de luz se monitoreará la cantidad de luz que se recibe, así cuando detecte que esta es mínima se activará la tira de leds con el fin de que las plantas reciban la luz necesaria.
- ♣ Se colocará un sensor ultrasónico por encima de las plantas, de esta manera cuando la distancia entre la planta y el sensor disminuya indicará que la planta ha crecido, recopilando esta información para ser mostrada al usuario.
- ♣ Haciendo uso de un sensor de humedad y temperatura se estarán monitoreando estos factores del invernadero, siendo que cuando se detecte que la temperatura es demasiada alta se activará un ventilador.

Para la interfaz se hará uso del Bot de Telegram como la interfaz que el usuario podrá usar para comunicarse con el invernadero, la información que se le mostrará al usuario y las acciones que este podrá realizar desde esta serán mediante los siguientes comandos:

- ♣ /help: listará todos los comandos de los que el usuario puede hacer uso y una descripción de lo que realiza cada uno de estos.
- ♣ /start: actualiza la información recopilada por los sensores y que se almacena en una base de datos.
- ♣ /temperatura: muestra el último registro hecho de la temperatura y humedad del invernadero.
- ♣ /grow: muestra el último registro que se tiene de cuanto aproximadamente han crecido las plantas.
- ♣ /light: muestra el estado actual de las luces del invernadero.
- ♣ /wet: informa si la tierra se encuentra húmeda o seca.

- ♣ /fresh: enciende el ventilador en caso de estar apagado, en caso contrario muestra un mensaje indicando que este se encuentra encendido.
- ♣ /water: enciende la bomba de agua en caso estar apagada durante tres segundos, en caso contrario muestra un mensaje indicando que esta se encuentra encendida.
- ♣ /lightup: enciende las luces en caso de estar apagadas, en caso contrario se muestra un mensaje indicando que se encuentran encendidas.
- ♣ /lightdown: apaga las luces en caso de estar encendidas, en caso contrario se muestra un mensaje indicando que se encuentran apagadas.

Adicionalmente se podrá hacer uso de Blue Dot para realizar algunas acciones en el invernadero y el asistente de Google para consultar información.

## Material

La lista de materiales usados en la implementación del proyecto se muestra a continuación.

Material	Descripción	Costo
<b>Sensor DHT11</b> 	Voltaje de Operación: 3V - 5V DC, Rango de medición de temperatura: 0 a 50 °C, Precisión de medición de temperatura: $\pm 2.0$ °C, Resolución Temperatura: 0.1°C, Rango de medición de humedad: 20% a 90% RH, Precisión de medición de humedad: 5% RH, Resolución Humedad: 1% RH, Tiempo de sensado: 1 seg, Interfaz digital: Single-bus (bidireccional)	\$10.00
<b>Sensor de humedad de suelo FC-28</b> 	Voltaje de alimentación: 3.3V - 5V DC (VCC), Corriente de operación: 35mA, Voltaje de señal de salida analógico (AO) : 0 a VCC, Voltaje de señal de salida digital (DO) : 3.3V/5V TTL, Opamp LM393 en modo comparador, umbral (threshold) regulable por potenciómetro, Superficie de electrodo: Estaño, Incluye: Electrodo, Placa y cable de conexión, Vida útil electrodo sumergido: 3 a 6 meses	\$45.00
<b>Sensor HC-SR04</b> 	Voltaje de trabajo 5V, Corriente 15mA, Frecuencia de trabajo 40HZ, Distancia máxima 4 m, Distancia mínima 2cm, Ángulo de apertura 15°, Margen de error 3mm o 0.3cm, Duración mínima del pulso de disparo (TTL) 10 $\mu$ S, Duración del pulso de eco de salida (TTL) 100-25000 $\mu$ S, Tiempo mínimo de espera entre mediciones 20 mS	\$50.00

<b>Sensor de luz</b> 	Celda fotoresistiva (Fotoresistencia) de 2 MOhms en la oscuridad y 50 KOhms máximos bajo la luz. Soporta 100 Vca.	\$25.00
<b>Tira de Leds</b> 	Serán colocados alrededor del espejo. Recortable cada 3 leds. Posicionamiento en ángulos de hasta 120°. Voltaje de operación de 12 VCC. Encapsulado IP65 a prueba de agua. Diseño anti-UV. Longitud de 5m, 300 LEDs tipo 5050. 5 metros de tira con 300 LEDs. Contiene autoadhesivo.	\$20.00 por metro
<b>Ventilador de 5V</b> 	Voltaje de operación: 5v Consumo: (.20A). Velocidad: 2000 + 10% RPM. Dimensiones: 30X30X7 mm.	\$148.00
<b>Mini bomba de agua de 6V</b> 	Voltaje de operación: 2.5-6 V, Elevación máxima: 40-110 cm, Flujo: 80-120l/h (2 litros por minuto), Tamaño de orificio de salida: 7.5mm, Tamaño de orificio de entrada: 5mm, Diámetro: aprox. 24mm, Longitud: Aprox. 45mm, Altura: Aprox. 30mm, Material: Plástico, Levante: ≈40cm-110cm, Modo de conducción: diseño sin escobillas, conducción magnética, Vida útil de trabajo continuo de 500 horas	\$65.00
<b>Circuito integrado L293D</b> 	Alimentación: 4.5 a 36 VDC, Corriente de salida: 600 mA. Corriente pico de salida: 1 A por canal (no repetitiva). Encapsulado: DIP de 16 pines. Alta inmunidad al ruido eléctrico. Protección contra exceso de temperatura. Diodos de protección (flyback) incorporados.	\$65.00




<b>Pila de 9V</b> 	Voltaje: 9 Vcc Capacidad nominal: 150 mAh Libre de mercurio y cadmio Dimensiones: 4.8 cm de alto x 2.6 cm de ancho x 1.7 cm de espesor	\$59.00 - \$130.00 dependiendo de la marca
<b>Videocámara USB con micrófono</b> 	1 micrófono incorporado capaz de transmitir a 10 metros, lente óptica importada de alta precisión, interfaz USB 2.0 o superior.	\$338.40
<b>Raspberry Pi 4B</b> 	Procesador Quad Core a 1.5GHz, 2GB RAM, soporta doble monitor (4Kp60) con salida en dos conectores micro HDMI, WiFi doble banda 2.4GHz y 5GHz, Bluetooth 5.0 y puerto Ethernet más rápido de un Gigabit.	\$1387.00

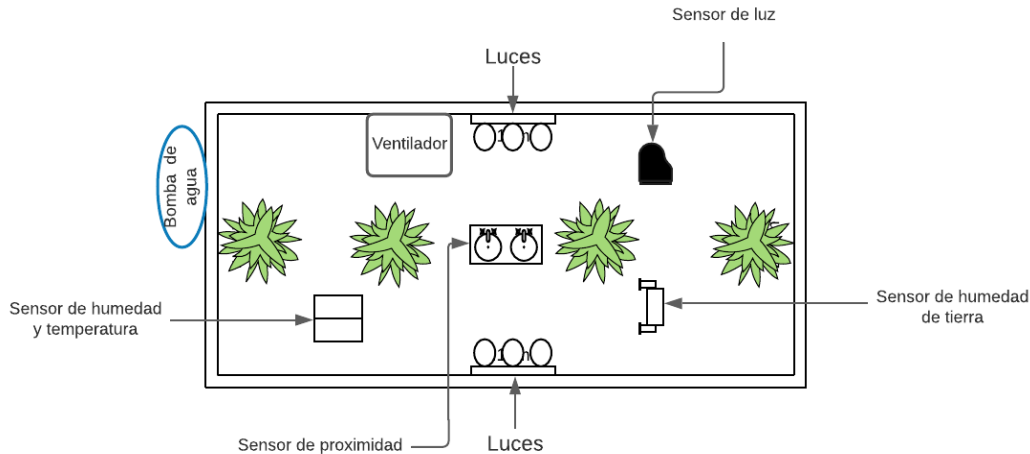
Tabla 1. Lista de materiales, descripción y costo.

## Costos

Basándose en la tabla de costos se puede estimar que el costo mínimo para implementar el proyecto es de \$2,212.40, esto dependerá de la tarjeta Raspberry Pi de la que se haga uso siendo que en mi caso hice uso del modelo Raspberry Pi 4B, pero esta puede ser remplazada por el modelo 4, 3 o 3B, además de que también dependerá si solo se adquiere la tarjeta o también de su carcasa, ventilador y demás. Cabe mencionar que no se está tomando en cuenta el precio del cableado a utilizar, así como de los materiales de soldadura que pueden llegar a requerirse.

## Diseño

El diseño del que se hace uso se muestra a continuación:

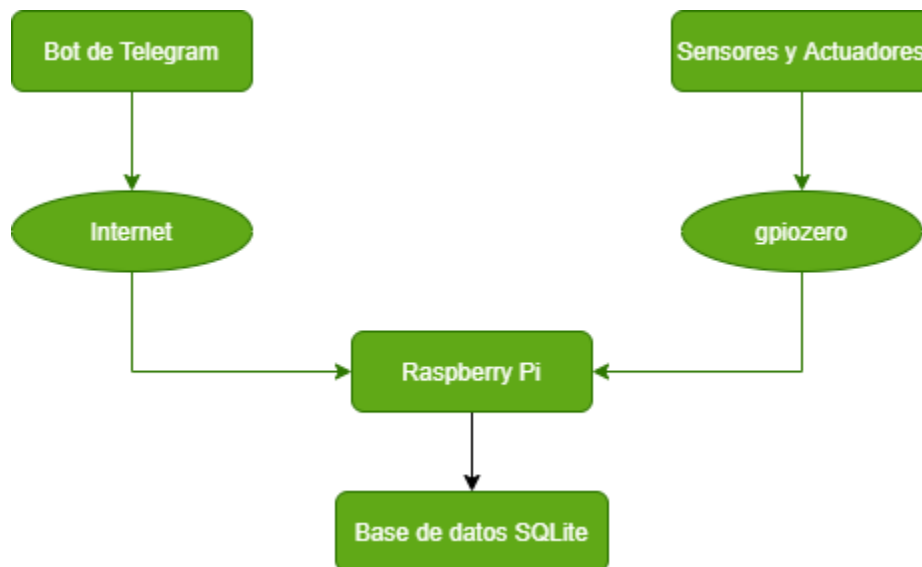


*Ilustración 3. Diseño del invernadero doméstico*

Todos los sensores y actuadores estarán conectados a la tarjeta Raspberry Pi, estos serán controlados por un programa que se dedica a monitorear el estado del invernadero, así como también podrán ser controlados mediante el Bot de Telegram y Blue Dot. Estas acciones dependerán del estado en que se encuentren cada uno de los actuadores. Adicionalmente podrán realizarse otras acciones mediante el Asistente de Google.

### Arquitectura General

La arquitectura del sistema que se desea implementar se puede ver en la siguiente figura.



*Ilustración 4. Arquitectura del sistema*

Por un lado se puede observar que tenemos al Bot de Telegram el cual gestiona la comunicación con el usuario a través de Internet. Por otro lado, los sensores y actuadores se conectan a la Raspberry Pi mediante los pines GPIO y conexiones USB. Todo ello se procesa en la placa Raspberry Pi, y a su vez, se almacena información en la Base de datos SQLite a la cual puede acceder al Bot.



## Arquitectura del Software

Se busca crear una arquitectura estructurada que permita trabajar de una manera eficiente y organizada; para ello se propone hacer una división de ficheros según su función dentro del Bot. El esquema sería el que vemos en la figura siguiente.

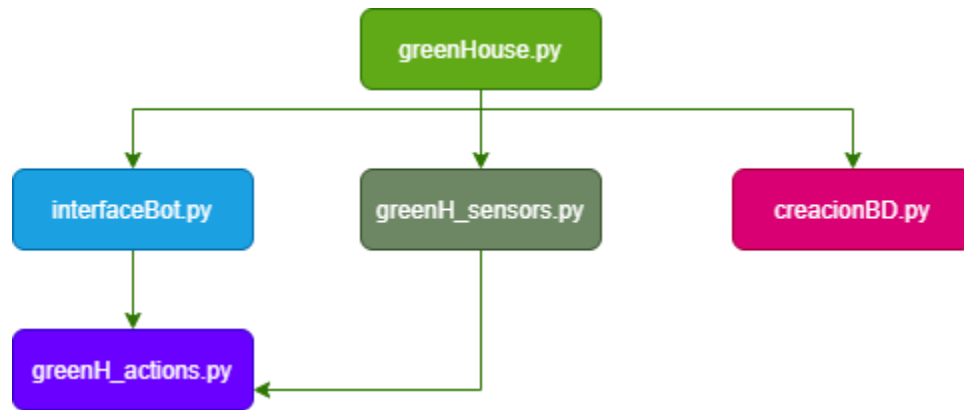


Ilustración 5. Arquitectura del software

El módulo “*greenHouse.py*” se encarga de comenzar y mandar la ejecución tanto del Bot como del *script* responsable de las lecturas de los sensores a segundo plano. Así mismo se encarga de ejecutar el *script* que crea la base de datos.

Después tendremos el módulo “*interfaceBot.py*” que actúa como el Bot de Telegram, este se encargará de recibir los mensajes de usuario y procesarlos correctamente.

El módulo “*greenH\_sensors.py*” se encarga del control de todos los sensores, proporcionando información sobre el estado de estos y del invernadero.

El módulo “*creacionBD.py*” se encarga de la creación de la base de datos, siendo que en caso de ya existir se imprime un mensaje por consola indicando que está ya ha sido creada previamente.

Finalmente el módulo “*greenH\_actions.py*” proporciona todas las funciones para activar y desactivar los actuadores del invernadero, estas serán usadas por el Bot de Telegram al ingresar el comando correspondiente.

Adicionalmente se tiene el módulo “*showLuces.py*” el cual hace uso del módulo Blue Dot para controlar el show de luces, dependiendo de la posición del botón que se presione iniciara un show de luces diferente, teniendo un total de cuatro shows de luces.

## Identificación de los elementos que funcionan con IoT

### Google Assistant

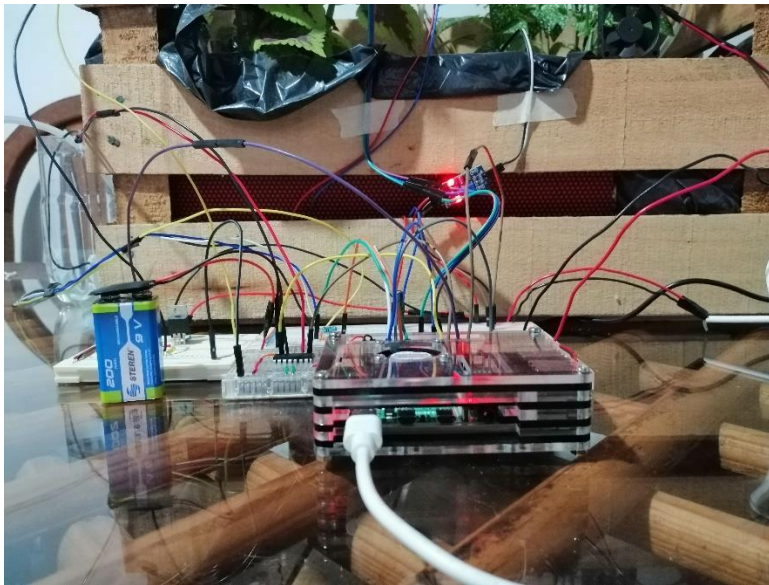
Se implementó esta tecnología con la finalidad de permitir al usuario pueda reproducir la música que quiera (siempre y cuando exista) mediante una orden, además de todas las funciones que ya están predeterminadas por el asistente de Google, cómo realizar preguntas, agendar citas, etc.

## Bot de Telegram

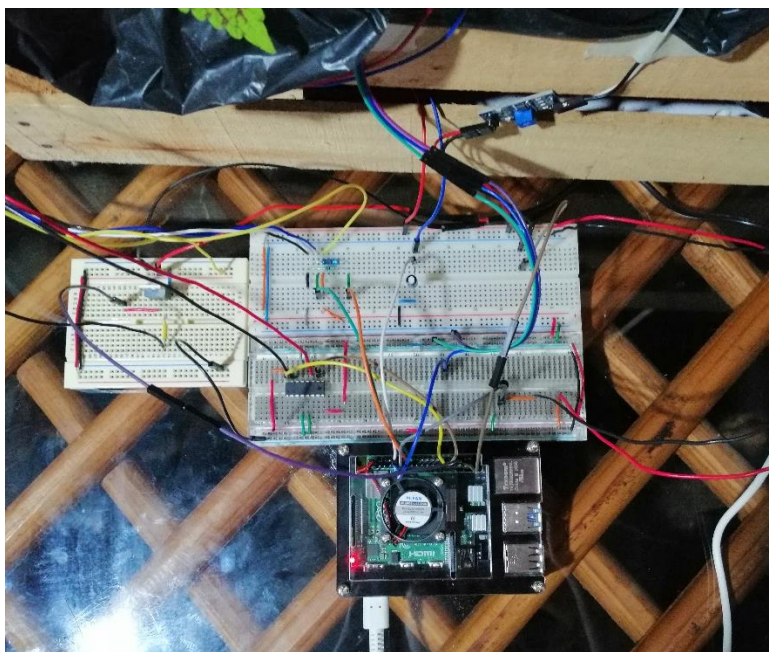
Mediante un Bot de Telegram se recuperará la información almacenada en la base de datos sobre los estados tanto de los actuadores como del invernadero, además de poder activar y desactivar estos dependiendo de los estados, siendo que cada vez que active o desactive uno de estos actualizará el estado correspondiente en la base de datos.

## Implementación

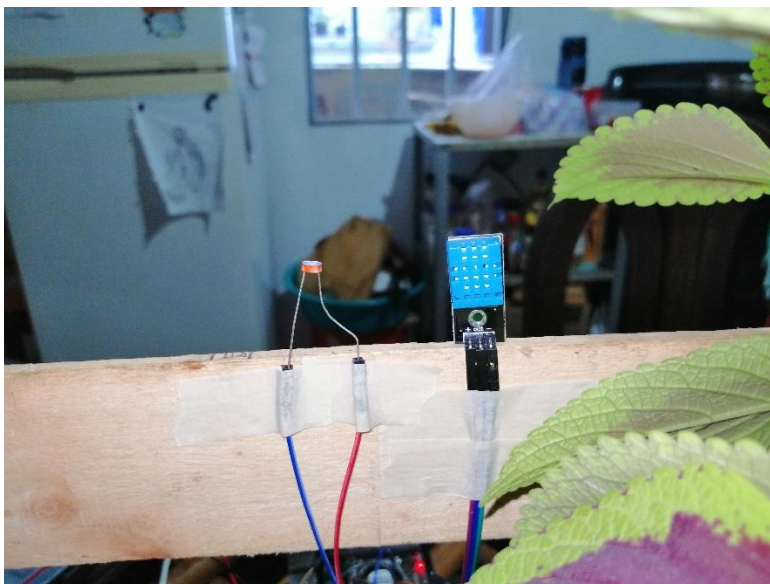
### Invernadero montado



*Ilustración 6. Circuito conectado y operando, vista frontal*



*Ilustración 7. Circuito conectado y operando, vista superior*

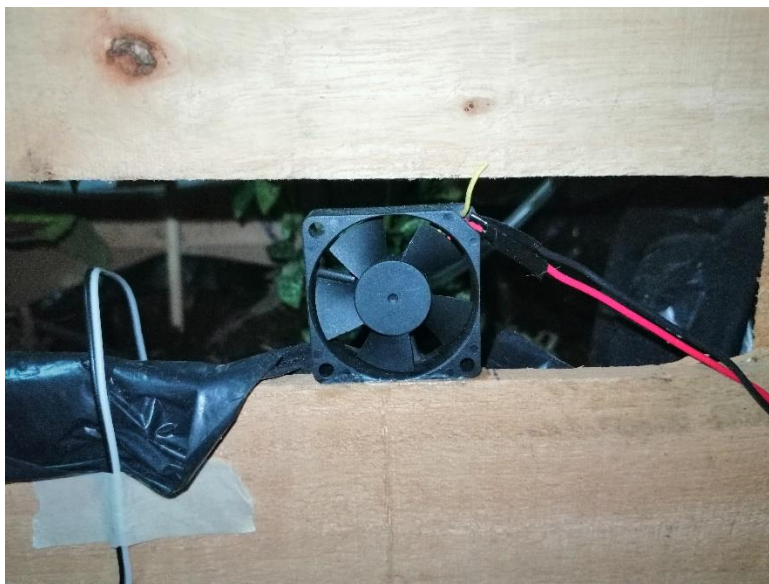


*Ilustración 8. Sensor de luz, sensor de humedad y temperatura DHT11*



*Ilustración 9. Sensor de humedad de tierra*





*Ilustración 10. Ventilador de 5V montado en el invernadero*



*Ilustración 11. Sensor HC-SR04 montado en el invernadero*



*Ilustración 12. Barra de leds de 12V montada en el invernadero - 1*



*Ilustración 13. Barra de leds de 12V montada en el invernadero - 2*

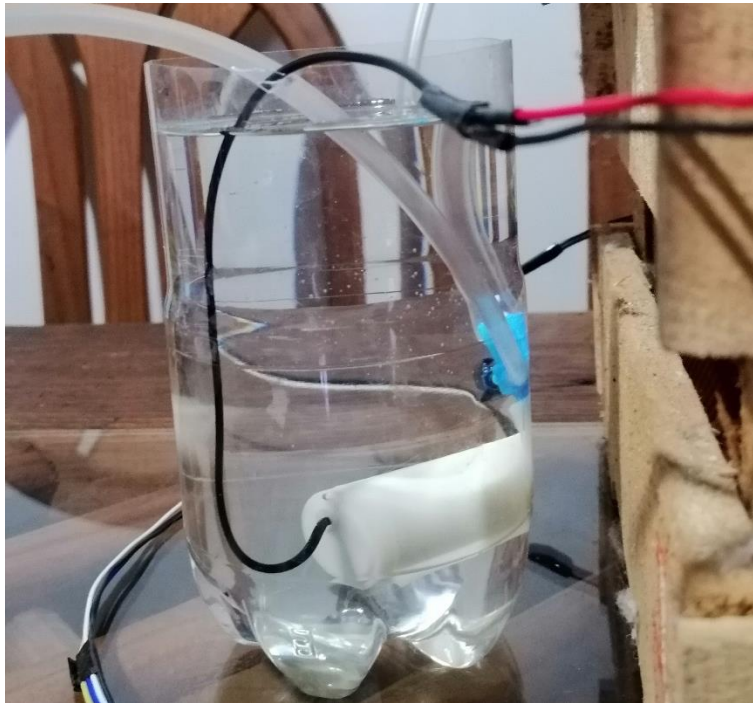


Ilustración 14. Mini bomba de agua sumergible colocada junto con las mangueras

## Funcionamiento del Bot de Telegram

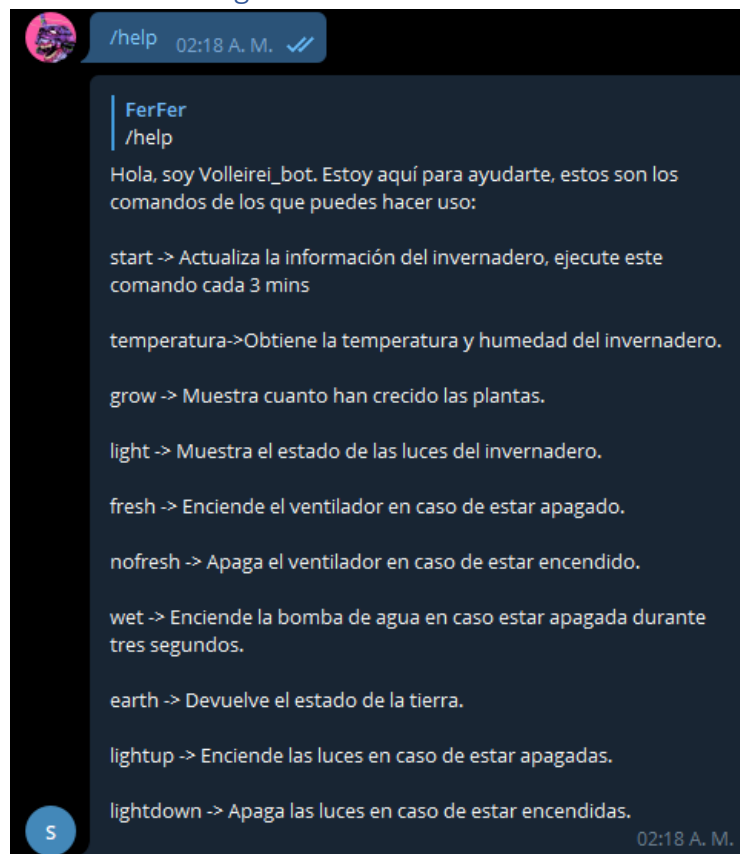


Ilustración 15. Muestra de la lista de comandos con /help

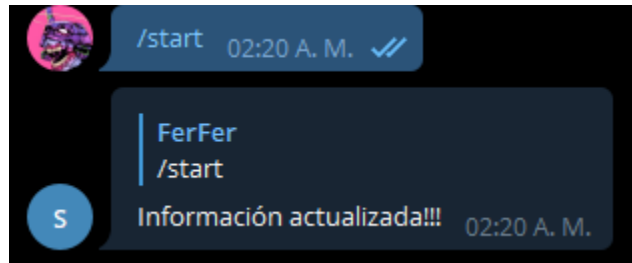


Ilustración 16. Funcionamiento del comando `/start`

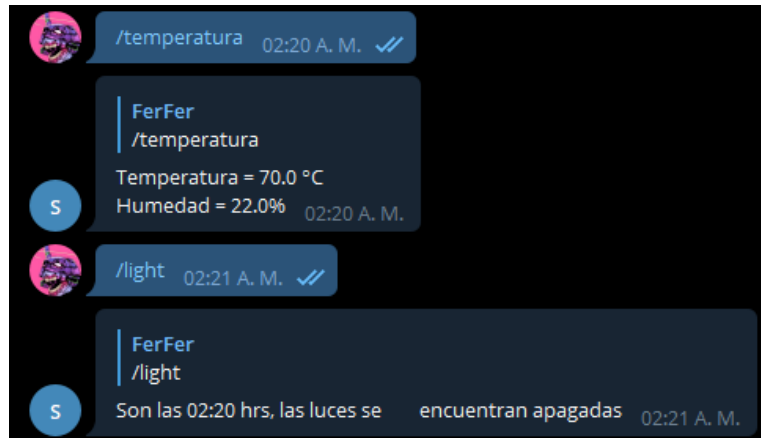


Ilustración 17. Comandos `/temperatura` y `/light`

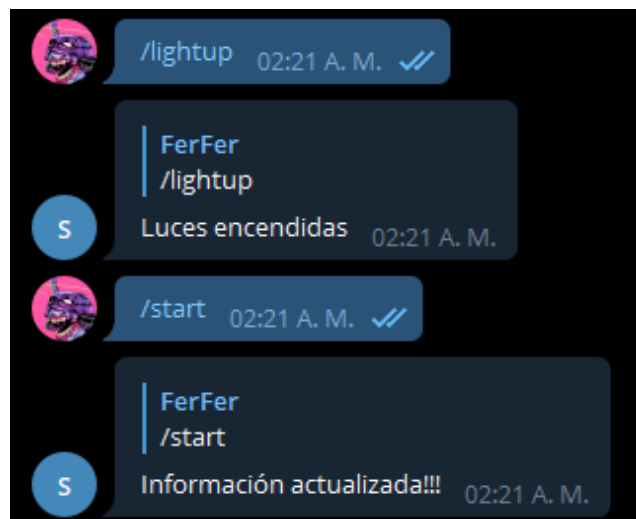


Ilustración 18. Encendiendo las luces con el comando `/lightup` y actualización de información



Ilustración 19. Barra de leds encendida con el comando /lightup

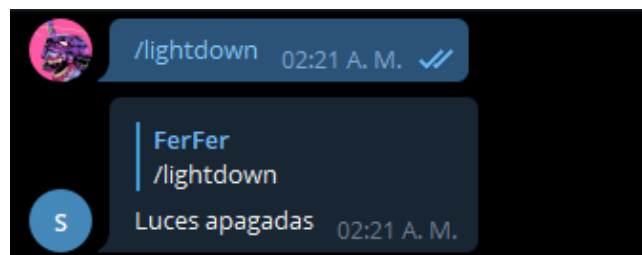


Ilustración 20. . Apagando las luces con el comando /lightdown



Ilustración 21. Barra de leds apagada con el comando /lightdown





Ilustración 22. Funcionamiento comandos `/grow` y `/earth`

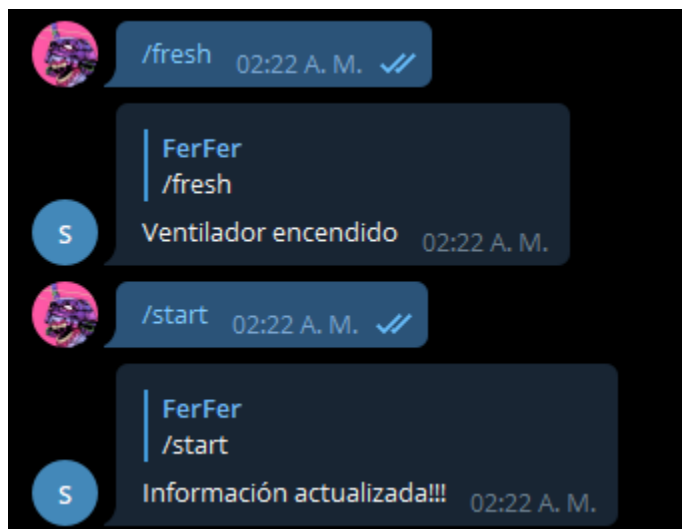
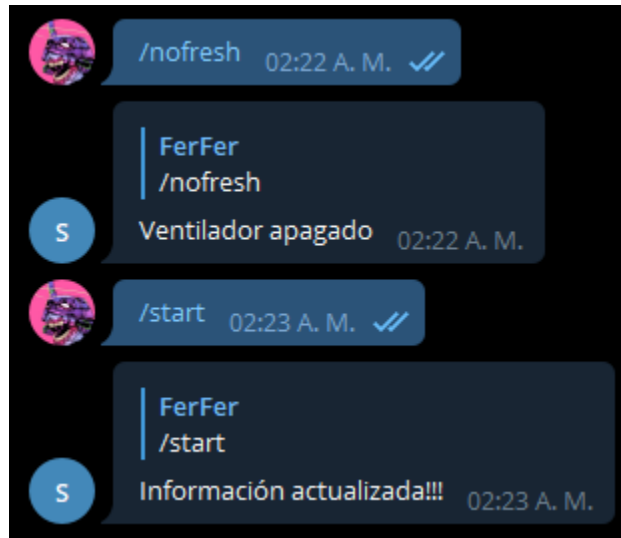


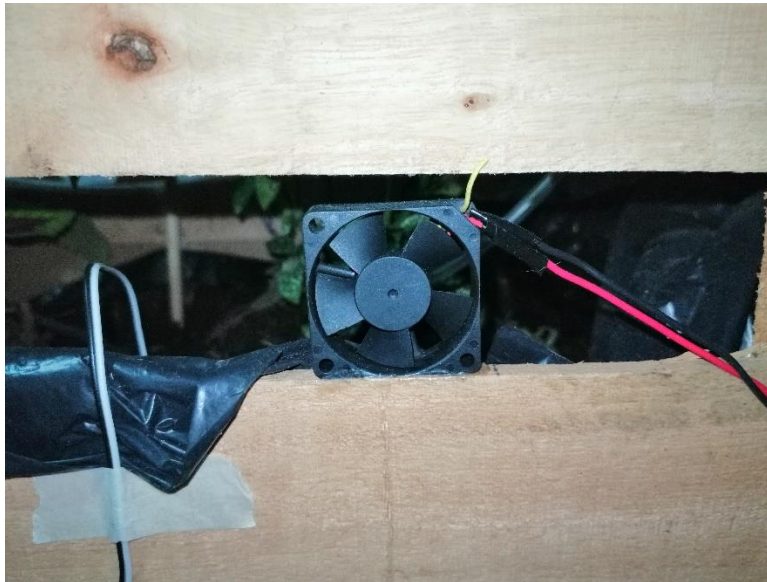
Ilustración 23. Funcionamiento comando `/fresh` y actualizando información



Ilustración 24. Ventilador encendido con el comando `/fresh`



*Ilustración 25. Funcionamiento comando /nofresh y actualizando información*



*Ilustración 26. Ventilador apagado mediante el comando /nofresh*



*Ilustración 27. Funcionamiento comando /wet*



*Ilustración 28. Bomba de agua operando mediante el comando /wet*

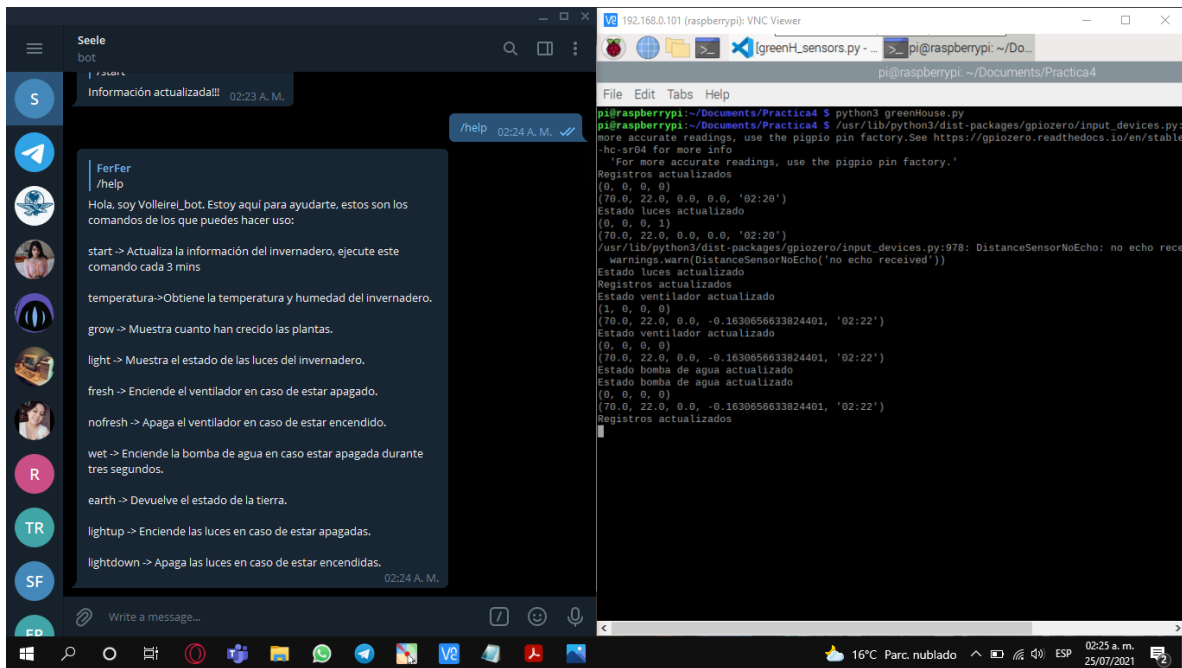


Ilustración 29. Bot de Telegram y terminal con el programa greenHouse.py

## Funcionamiento Blue Dot

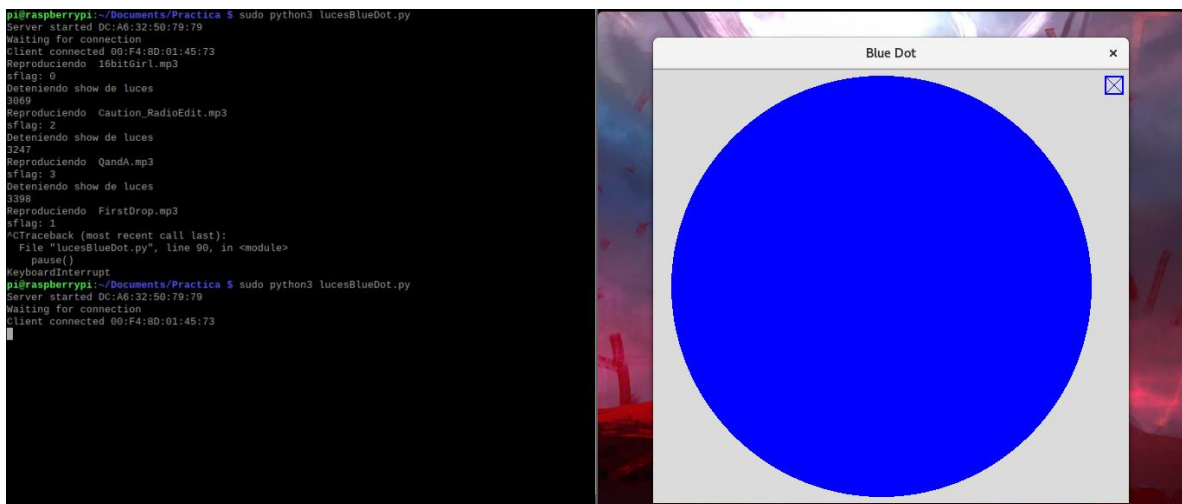
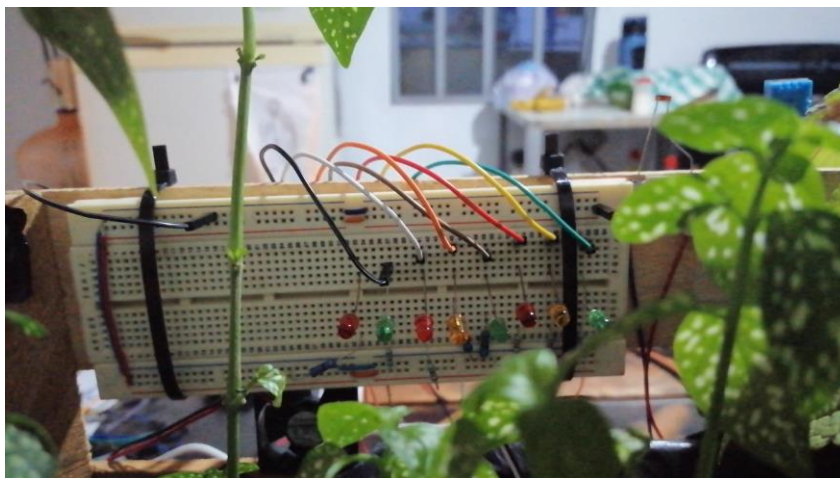
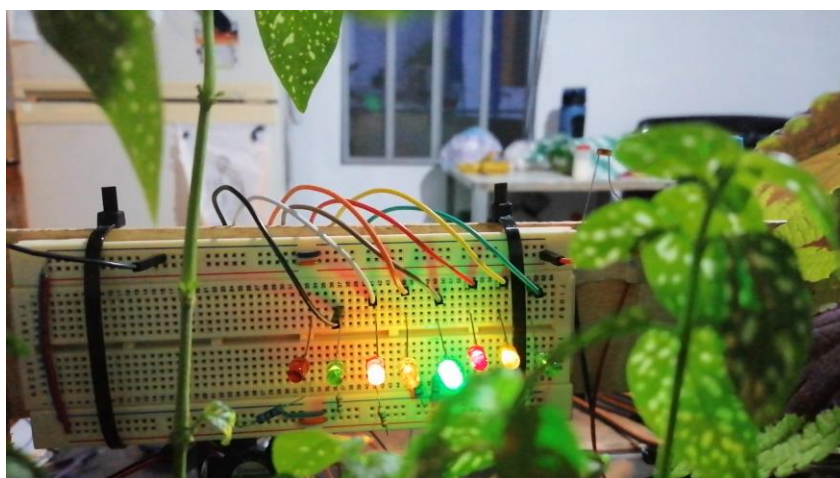


Ilustración 30. Terminal con los mensajes de showLuces.py y aplicación Blue Dot

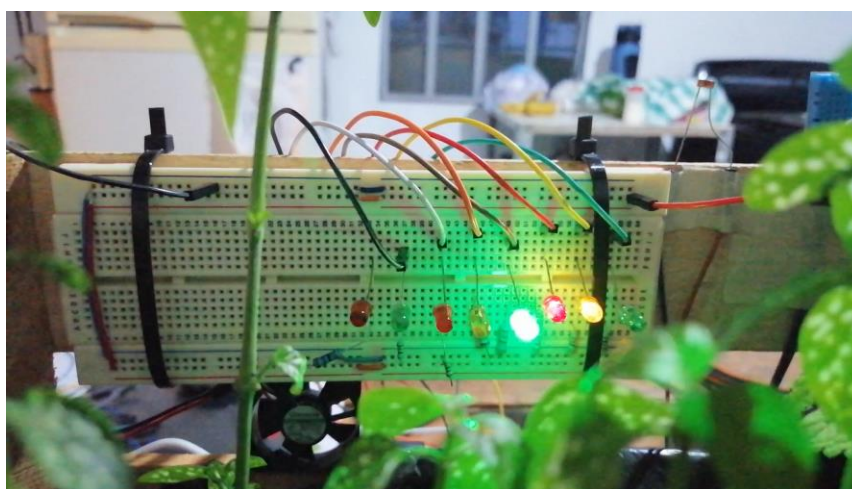




*Ilustración 31. Leds antes de comenzar con el show de luces*



*Ilustración 32. Iniciando show de luces*



*Ilustración 33. Show de luces en ejecución*

## Códigos usados

### greenHouse.py

```
import os

os.system('python3 creacionBD.py')

os.system('python3 interfaceBot.py&')
os.system('python3 greenH_sensors.py&')
```

### creacionBD.py

```
import sqlite3

# Conexión con la base de datos, en caso de no existir se crea
con = sqlite3.connect('greenHouse.db')
try:
    """ Se crean las tablas necesarias para almacenar los status de los actuadores y
    la información obtenida de los sensores """
    cursorObj = con.cursor()

    # Se crea la tabla greenhouse que almacenará la información obtenida de los sensores
    sql1 = '''CREATE TABLE greenhouse (humedad real, temperatura real, \
humTierra real, crecimiento real, hora text)'''

    cursorObj.execute(sql1)

    # Se crea la tabla sensorsStatus que almacenará los status de los actuadores
    sql2 = '''CREATE TABLE sensorsStatus (motorStatus integer, waterStatus integer,\
ledProxStatus integer, ledSLstatus integer)'''

    cursorObj.execute(sql2)

    print("Base de datos creada exitosamente")
except sqlite3.OperationalError:
    print("La base de datos ya ha sido creada previamente")

con.close() # Cerramos la conexión con la BD
```

### interfaceBot.py

```
import telebot
import os
import greenH_actions

API_TOKEN = '1838331528:AAGT9ruBtbRsVoC0QBF-E2S-lM7rgijTPwQ'

bot = telebot.TeleBot(API_TOKEN)

# Handle '/start' and '/help'
@bot.message_handler(commands=['help'])
def send_welcome(message):
    """Lista los comandos con los que cuenta el bot junto una pequeña
    descripción"""
    bot.reply_to(message, """\
Hola, soy Volleirei_bot. Estoy aquí para ayudarte, estos son los comandos de los que puedes
hacer uso:\
\n\nstart -> Actualiza la información del invernadero, ejecute este comando cada 3 mins\
\n\ntemperatura->Obtiene la temperatura y humedad del invernadero.\
\n\ngrow -> Muestra cuanto han crecido las plantas. \
\n\nlight -> Muestra el estado de las luces del invernadero. \
\n\nfresh -> Enciende el ventilador en caso de estar apagado. \
```

```

\n\nnofresh -> Apaga el ventilador en caso de estar encendido. \
\n\nwet -> Enciende la bomba de agua en caso estar apagada durante tres segundos. \
\n\nearth -> Devuelve el estado de la tierra. \
\n\nlightup -> Enciende las luces en caso de estar apagadas. \
\n\nlightdown -> Apaga las luces en caso de estar encendidas.
"""

@bot.message_handler(commands=['start'])
def actualizar_Info(message):
    """Obtiene los últimos registros almacenados en la base de datos"""
    greenH_actions.actualizarStatus()
    bot.reply_to(message, "Información actualizada!!!")

@bot.message_handler(commands=['temperatura'])
def medir_Temp(message):
    """Muestra la temperatura y humedad del invernadero"""
    humedad, temperatura = greenH_actions.getTemp_Hum()
    bot.reply_to(message, """Temperatura = {0:0.1f} °C \
\nHumedad = {1:0.1f}%""".format(humedad, temperatura))

@bot.message_handler(commands=['grow'])
def enviar_Crecimiento(message):
    """Obtiene y muestra el último registro del crecimiento de las plantas"""
    crecimiento = greenH_actions.getCrecimientoPlantas()
    bot.reply_to(message, "Las plantas han crecido {0:0.1f} cm".format(crecimiento))

@bot.message_handler(commands=['light'])
def enviar_EdoLuces(message):
    """Obtiene y muestra el estado de las luces del invernadero, así como la
    hora del sistema"""
    hora = greenH_actions.getHora()
    lightStaus = greenH_actions.getLedSlStatus()

    if lightStaus == 0:
        bot.reply_to(message, """Son las {0} hrs, las luces se\
    encuentran apagadas""".format(hora))
    elif lightStaus == 1:
        bot.reply_to(message, """Son las {0} hrs, y algo esta obstruyendo\
    la luz. Estas se encendieron""".format(hora))
    else:
        bot.reply_to(message, "Es de noche o madrugada, las luces permanecen apagadas")

@bot.message_handler(commands=['wet'])
def enviar_EdoTierra(message):
    """Activa la bomba de agua dependiendo del último status registrado en la
    BD, de activarla lo hace durante 3 segundos"""
    humT = greenH_actions.getWaterStatus()
    if humT == 0:
        bot.reply_to(message, "Regando las plantas")
        greenH_actions.setWaterStatus(1)
        greenH_actions.regarTierra()
        greenH_actions.setWaterStatus(0)
    else:
        bot.reply_to(message, "La bomba de agua ya esta encendida")

@bot.message_handler(commands=['fresh'])
def refrescar(message):
    """Activa el ventilador dependiendo del último status registrado en la BD"""
    statusFan = greenH_actions.getMotorStatus()
    if statusFan == 0:
        greenH_actions.encenderVentilador()
        greenH_actions.setMotorStatus(1)
        bot.reply_to(message, "Ventilador encendido")
    else:
        bot.reply_to(message, "El ventilador ya se encuentra encendido")

```

```

@bot.message_handler(commands=['nofresh'])
def apagarFan(message):
    """Apaga el ventilador dependiendo del último status registrado en la BD"""
    statusFan = greenH_actions.getMotorStatus()
    if statusFan == 1:
        greenH_actions.apagarVentilador()
        greenH_actions.setMotorStatus(0)
        bot.reply_to(message, "Ventilador apagado")
    else:
        bot.reply_to(message, "El ventilador ya se encuentra apagado")

@bot.message_handler(commands=['earth'])
def regar(message):
    """Obtiene y muestra el estado de la tierra del invernadero"""
    wetStatus = greenH_actions.getHumedadTierra()
    if wetStatus == 0:
        bot.reply_to(message, "La tierra esta humeda")
    else:
        bot.reply_to(message, "La tierra esta seca")

@bot.message_handler(commands=['lightup'])
def lightOn(message):
    """Dependiendo del último status en la BD enciendo o no las luces del invernadero"""
    lightSt = greenH_actions.getLedSlStatus()
    if lightSt == 0:
        greenH_actions.encenderLedSL()
        greenH_actions.setLedSlStatus(1)
        bot.reply_to(message, "Luces encendidas")
    else:
        bot.reply_to(message, "Las luces ya estan encendidas")

@bot.message_handler(commands=['lightdown'])
def lightOff(message):
    """Dependiendo del último status en la BD apaga o no las luces del invernadero"""
    lightSt = greenH_actions.getLedSlStatus()
    if lightSt == 1:
        greenH_actions.apagarLedSL()
        greenH_actions.setLedSlStatus(0)
        bot.reply_to(message, "Luces apagadas")
    else:
        bot.reply_to(message, "Las luces ya estan apagadas")

@bot.message_handler(func=lambda message: True)
def echo_message(message):
    """Responde con el mismo mensaje enviado por el usuario"""
    bot.reply_to(message, message.text)

bot.polling()

```

### greenH\_sensors.py

```

import sys
import Adafruit_DHT
import greenH_actions
import sqlite3
from gpiozero import InputDevice
from gpiozero import LightSensor
from gpiozero import DistanceSensor
from time import sleep, strftime

# Sensores y asignación de pines
SENSOR_DHT = Adafruit_DHT.DHT11 # Sensor de temperatura y humedad
PIN_DHT = 17 #Pin asignado
senLuz= LightSensor(18) #Sensor de luz (fotorresistencia)

```



```

# Sensor ultrasonico (HC-SR04)
sensor = DistanceSensor(23, 24) #Pin 23 --> Echo, Pin 24 --> Trigger
# Sensor de humedad terrestre
hum = InputDevice(26)

# Variables globales a usar
ventStatus = 0 # Status del ventilador
wbStatus = 0 # Status de la bomba de agua
lpStatus = 0 # Status del sensor HC-SR04
lslStatus = 0 # Status luces controladas por el sensor de luz
humedad,temperatura = -1.0,-1.0 # Humedad y temperatura del sensor DHT11
humTierra = 0 # Humedad sensor de humedad de tierra
primerMuestra = False
disInit = 0
crecimiento = 0.0
hr = "" # Hora de encendido/apagado luces
hora = int(strftime("%H"))
db = 'greenHouse.db' #Base de datos
regFlag = False

# Funciones
def medirHumedadTierra():
    """Obtiene la lectura proveniente del sensor de humedad de tierra, en caso
    de que la tierra este seca se activa la bomba de agua y se actualiza su status,
    en caso contrario se deja apagada"""

    global humTierra, wbStatus
    if hum.value == 0:
        humTierra = 0
        wbStatus = 0
    else:
        humTierra = 1
        wbStatus = 1
        greenH_actions.encenderWaterBomb()
        sleep(3)
        greenH_actions.apagarWaterBomb()
        wbStatus = 0

def leerTemperatura():
    """Obtiene la lectura tomada por el sensor DHT11 y se verifica que estas no
    sean nulas, en caso de serlo se colocan en 0"""

    global humedad, temperatura
    humedad, temperatura = Adafruit_DHT.read_retry(SENSOR_DHT, PIN_DHT)
    if humedad is None and temperatura is None:
        humedad, temperatura = 0, 0

def activarVentilador():
    """Enciende el ventilador en caso de que la temperatura sea mayor a 25°C"""

    global ventStatus
    if temperatura > 25.0:
        greenH_actions.encenderVentilador()
        ventStatus = 1
    elif temperatura <= 25.0:
        greenH_actions.apagarVentilador()
        ventStatus = 0
    else:
        ventStatus = 0

def monitorearLuz():
    """Obtiene la lectura tomada por el sensor de luz, siendo que en caso de no
    detectar suficiente luz y la hora del sistema este dentro del rango
    establecido se encienden las luces, en caso contrario esta se apagan"""

    global msgLuz, hora, hr

```

```

hr = strftime("%H:%M")

if hora in range(8,16) and senLuz.light_detected:
    greenH_actions.apagarLedSL()
    lslStatus = 0
elif hora in range(8,16) and not senLuz.light_detected:
    greenH_actions.encenderLedSL()
    lslStatus = 1
else:
    greenH_actions.apagarLedSL()
    lslStatus = 2

def obtenerPrimeraMuestra():
    """Toma una primera lectura con el sensor HC-SR04 y la almacena, transformandola
    a centimetros para posteriorenente ser usada para el cálculo" de cuanto han
    crecido las plantas"""

    global disInit, primerMuestra

    if primerMuestra == False:
        disInit = sensor.distance * 100
        primerMuestra = True

def sensarCrecimiento():
    """Realiza un lectura con el sensor HC-SR04, la transforma a centimetros y la
    usa para calcular cuanto han crecido aproximadamente las plantas, en caso de
    ser negativa la lectura se se coloca en cero"""

    global disInit, crecimiento
    distancia = sensor.distance * 100
    crecimiento = disInit - distancia
    greenH_actions.encenderLedProx()
    disInit = distancia
    greenH_actions.apagarLedProx()

    if distancia < 0.0:
        crecimiento = 0.0

def insertarRegistros():
    """Realiza una conexión con la BD, elimina las tablas en donde se almacenan
    los registros de los status y la información obtenida por los sensores,
    posteriormente volverlas a crear e insertar los nuevos registros.
    Finalmente realiza un commit y cierra la conexión."""

    global regFlag
    conn = sqlite3.connect(db)
    cur = conn.cursor()
    cur.execute("DROP TABLE IF EXISTS greenhouse")
    cur.execute("DROP TABLE IF EXISTS sensorsStatus")

    sql1 = '''CREATE TABLE greenhouse (humedad real, temperatura real, \
humTierra real, crecimiento real, hora text)'''
    cur.execute(sql1)

    sql2 = '''CREATE TABLE sensorsStatus (motorStatus integer, waterStatus integer,\
ledProxStatus integer, ledSLstatus integer)'''
    cur.execute(sql2)

    cur.execute("INSERT INTO greenhouse VALUES
(?,?,?,?,?)",[humedad,temperatura,humTierra,crecimiento,hr])
    cur.execute("INSERT INTO sensorsStatus VALUES
(?,?,?,?,?)",[ventStatus,wbStatus,lpStatus,lslStatus])
    conn.commit()
    conn.close()
    regFlag = False

```

```

def sincronizarRegistros():
    """Realiza una conexión con la BD, ejecuta una consulta y asigna los registros
    almacenados en las variables globales, posteriormente cierra la conexión y
    actualiza la bandera indicando que los registros han sido sincronizados"""

    global ventStatus, wbStatus, lpStatus, lslStatus, regFlag
    conn = sqlite3.connect(db)
    cur = conn.cursor()
    sql1 = '''SELECT * FROM sensorsStatus'''
    cur.execute(sql1)
    registros = cur.fetchone()
    ventStatus = registros[0]
    wbStatus = registros[1]
    lpStatus = registros[2]
    lslStatus = registros[3]
    conn.close()
    regFlag = True

def monitorearGH():
    """Realiza el monitoreo del invernadero, recopilando la información de su
    entorno por medio de los sensores, activando o desactivando los actuadores
    para finalmente insertar la información obtenida en la BD."""

    while True:
        obtenerPrimeraMuestra()
        leerTemperatura()
        activarVentilador()
        medirHumedadTierra()
        monitorearLuz()
        sensarCrecimiento()
        insertarRegistros()

        if regFlag == False:
            sincronizarRegistros()
            print("Registros actualizados")

        sleep(120)

if __name__ == "__main__":
    monitorearGH()

```

### greenH\_actions.py

```

import sqlite3
from gpiozero import LED
from gpiozero import Motor
from time import sleep

# Actuadores
vent = LED(27) # Ventilador
water = Motor(forward=5, backward=6) # Bomba de agua
ledProx = LED(25) # Led sensor HC-SR04
ledSL = LED(16) # Barra de LEDs

# Variables globales
statuss = () #Almacena los status de los sensores
info = () #Almacena la información obtenida por los sensores
db = 'greenHouse.db' #Nombre de la base de datos

# Funciones
def encenderLedSL():
    """Enciende las luces controladas por el sensor de luz"""
    ledSL.on()

```

```

def apagarLedSL():
    """Apaga las luces controladas por el sensor de luz"""
    ledSL.off()

def encenderLedProx():
    """Enciende el led perteneciente al sensor HC-SR04"""
    ledProx.on()

def apagarLedProx():
    """Apaga el led perteneciente al sensor HC-SR04"""
    ledProx.off()

def encenderVentilador():
    """Enciende el ventilador"""
    vent.on()

def apagarVentilador():
    """Apaga el ventilador"""
    vent.off()

def encenderWaterBomb():
    """Enciende la bomba de agua"""
    water.forward()

def apagarWaterBomb():
    """Apaga la bomba de agua"""
    water.stop()

def getTemp_Hum():
    """Retorna los valores de humedad y temperatura obtenido por el sensor
    DHT11 y almacenado en la Base de Datos"""
    return info[0], info[1]

def getHumedadTierra():
    """Retorna el valor de la humedad de tierra obtenido por el sensor de
    humedad de tierra y almacenado en la Base de Datos"""
    return info[2]

def getCrecimientoPlantas():
    """Retorna los centimetros que han crecido las plantas obtenido por el
    sensor HC-SR04 y almacenado en la Base de Datos"""
    return info[3]

def getHora():
    """Retorna la hora del sistema en que se tomo el registro"""
    return info[4]

def getMotorStatus():
    """Retorna el status almacenado en la Base de Datos correspondiente al
    ventilador"""
    return statuss[0]

def getWaterStatus():
    """Retorna el status almacenado en la Base de Datos correspondiente a
    la bomba de agua"""
    return statuss[1]

def getLedProxStatus():
    """Retorna el status almacenado en la Base de Datos correspondiente a
    al led del sensor HC-SR04"""
    return statuss[2]

def getLedSlStatus():
    """Retorna el status almacenado en la Base de Datos correspondiente a
    las luces controladas por el sensor de luz"""

```

```

        return statusss[3]

def setLedSlStatus(status):
    """
    Asigna el status recibido a las luces controladas por el sensor de luz status.
    Realiza una conexión con a BD y ejecutando la sentencia UPDATE de sql, una
    vez actualizado el registro cierra la conexión.
    Parameters:
        status (int): valor del status, 0 o 1
    """
    conn = sqlite3.connect(db)
    cur = conn.cursor()
    cur.execute("UPDATE sensorsStatus SET ledSLstatus = ?",[status])
    conn.commit()
    print("Estado luces actualizado")
    conn.close()

def setMotorStatus(status):
    """
    Asigna el status recibido al ventilador. Realiza una conexión con la BD y
    ejecutando la sentencia UPDATE de sql, una vez actualizado el registro
    cierra la conexión.
    Parameters:
        status (int): valor del status, 0 o 1
    """
    conn = sqlite3.connect(db)
    cur = conn.cursor()
    cur.execute("UPDATE sensorsStatus SET motorStatus = ?",[status])
    conn.commit()
    print("Estado ventilador actualizado")
    conn.close()

def setWaterStatus(status):
    """
    Asigna el status recibido a la bomba de agua. Realiza una conexión con la BD y
    ejecutando la sentencia UPDATE de sql, una vez actualizado el registro
    cierra la conexión.
    Parameters:
        status (int): valor del status, 0 o 1
    """
    conn = sqlite3.connect(db)
    cur = conn.cursor()
    cur.execute("UPDATE sensorsStatus SET waterStatus = ?",[status])
    conn.commit()
    print("Estado bomba de agua actualizado")
    conn.close()

def regarTierra():
    """Enciende la bomba de agua durante tres segundos y posteriormente la apaga"""
    encenderWaterBomb()
    sleep(3)
    apagarWaterBomb()

def actualizarStatus():
    """Realiza una conexión con la BD, ejecuta una consulta a esta y almacena
    los registros en las tuplas statusss e info"""
    global statusss, info
    conn = sqlite3.connect(db)
    cur = conn.cursor()
    sql1 = '''SELECT * from sensorsStatus'''
    cur.execute(sql1)
    statusss = cur.fetchall()
    #sensorsStatus (motorStatus integer, waterStatus integer, ledProxStatus integer,
    ledSLstatus integer)

    print(statusss)

```

```

sql2 = '''SELECT * from greenhouse'''
cur.execute(sql2)
info = cur.fetchone()
#greenhouse (humedad real,temperatura real,humTierra real, crecimiento real,hora text)

print(info)

conn.close()

```

## showluces.py

```

import os
import psutil
from signal import pause, SIGKILL, SIGINT, SIGTERM
from bluepy import BlueDot
from time import sleep

# Canciones que se reproduciran dependiendo de la posición del boton que se presione
songs = ('16bitGirl.mp3','FirstDrop.mp3','Caution_RadioEdit.mp3','QandA.mp3')
# Comando que inicia el show de luces
comando = 'sudo python3 /home/pi/lightshowpi/py/synchronized_lights.py --'
file=/home/pi/Music/'
# Bandera que indica si se esta o no ejecutando el show de luces
status = False
# Bandera que indica la canción que se esta reproduciendo
sflag = 4

def controlLightShow(pos):
    """
    Detecta la posición del botón de Blue Dot que se ha presionado y reproduce
    la comienza el show de luces con la asignación asignada a esa posición.

    Parameters:
        pos(bluepy): objeto de Blue Dot del cual se obtendrá la posición presionada.
    """

    global status, sflag
    process = ''

    if status == True:
        # Verifica si no hay algún show de luces en curso
        stopLightShow(songs[sflag])
        #sleep(3)

    else:
        """ Sino hay un show de curso de luces en uso se inicia el correspondiente
        a esa posición"""
        if pos.top:
            process = comando + songs[0] + '&'
            os.system(process)
            sflag = 0
            print("Reproduciendo ",songs[0])
        elif pos.bottom:
            process = comando + songs[1] + '&'
            os.system(process)
            sflag = 1
            print("Reproduciendo ",songs[1])
        elif pos.left:
            process = comando + songs[2] + '&'
            os.system(process)
            sflag = 2
            print("Reproduciendo ",songs[2])
        elif pos.right:
            process = comando + songs[3] + '&'
            os.system(process)
            sflag = 3
            print("Reproduciendo ",songs[3])

```

```

        # Se actualiza la bandera indicando que hay un show de luces en curso
        status = True
        print("sflag:", sflag)

def stopLightShow(song):
    """
    Detiene el show de luces en ejecución con la canción indicada.

    Parameters:
        song(string): nombre del archivo mp3
    """

    global status
    # Script de python que ejecuta el show de luces
    cmd = "/home/pi/lightshowpi/py/synchronized_lights.py"
    #cmd = Archivo mp3 con su ruta
    pcs = "--file=/home/pi/Music/" + song

    # Obtenemos todos los procesos que se encuentran ejecutando
    for process in psutil.process_iter():
        # Buscamos el proceso del show de luces
        if process.cmdline() == ["sudo", "python3", cmd, pcs]:
            print("Deteniendo show de luces")
            print(process.pid)
            #Matamos a todos los procesos hijos
            for child in process.children(recursive=True):
                child.kill()
            process.kill() # Matamos al proceso padre
            status = False

    if status == True:
        print("Proceso no encontrado...")

blueD = BlueDot()
blueD.when_pressed = controlLightShow
pause()

```

## Evaluación de viabilidad

En relación a un análisis de costo/beneficio entre los materiales y la implementación, encuentro viable la implementación del proyecto. Teniendo cuidado con los componentes asociados al manejo de los dispositivos, evitar el daño de estos protegiéndolos contra cambios de temperatura, exposición a la humedad, descargas eléctricas y que se encuentre resguardados de forma óptima tal que permita el crecimiento libre de las plantas.

## Conclusión

Al realizar este proyecto pude ver de mejor manera las aplicaciones de los módulos de la biblioteca gpiozero en escenarios de la vida real, siendo que en este caso se usaron para construir un invernadero doméstico, logrando implementar un sistema embebido puesto que a través de los sensores se recopilaba la información del entorno del invernadero y de acuerdo a esta se encendían o apagaban los actuadores. Del mismo modo con el uso del Bot de Telegram como la interfaz de usuario proporciono la funcionalidad del IoT, al poder consultar esta información sin estar físicamente en el invernadero. Proporcionando comodidad al usuario puesto que se han automatizado las acciones a realizar dentro de este.

## Fuentes de consulta

- ♣ insst. (S.F). ¿Qué es un invernadero? julio 13, 2021, de insst Sitio web: <https://www.insst.es/-/que-es-un-invernader-1>
- ♣ CEDOM. (S.F). Qué es Domótica. julio 13, 2021, de CEDOM Sitio web: <http://www.cedom.es/sobre-domotica/que-es-domotica>
- ♣ Sarachu, E. (2020). ¿Qué es la domótica? ¿Cómo funciona? julio 13, 2021, de E-ficiencia Sitio web: <https://e-ficiencia.com/domotica-que-es-y-como-funciona/>
- ♣ Passerby. (2016). RaspberryPi + FET LED lighting controller. julio 24, 2021, de Eletrical Engineering Sitio web: <https://electronics.stackexchange.com/questions/235460/raspberrypi-fet-led-lighting-controller/235463#235463>
- ♣ Nuttall, B., & Jones, D. (S.F). 13. API - Input Devices. julio 12, 2021, de gpiozero Sitio web: [https://gpiozero.readthedocs.io/en/stable/api\\_input.html](https://gpiozero.readthedocs.io/en/stable/api_input.html)
- ♣ Nuttall, B., & Jones, D. (S.F). 2. Basic Recipes. julio 12, 2021, de gpiozero Sitio web: <https://gpiozero.readthedocs.io/en/stable/recipes.html>
- ♣ Nuttall, B., & Jones, D. (S.F). 14. API - Output Devices. julio 12, 2021, de gpiozero Sitio web: [https://gpiozero.readthedocs.io/en/stable/api\\_output.html#](https://gpiozero.readthedocs.io/en/stable/api_output.html#)
- ♣ Rico, E. (S.F). os — Acceso portable a funciones específicas del sistema operativo. julio 12, 2021, de El módulo Python3 de la semana Sitio web: <https://rico-schmidt.name/pymotw-3/os/index.html>
- ♣ Python. (S.F). sqlite3 — DB-API 2.0 interface for SQLite databases. julio 14, 2021, de Python Sitio web: <https://docs.python.org/3/library/sqlite3.html>
- ♣ Anónimo. (S.F). ¿Cómo matar procesos y procesos hijos de Python? agosto 05, 2021, de It-swarm Sitio web: <https://www.it-swarm-es.com/es/python/como-matar-procesos-y-procesos-hijos-de-python/972186697/>
- ♣ Widdis, D. (S.F). psutil documentation. agosto 03, 2021, de psutil Sitio web: <https://psutil.readthedocs.io/en/latest/>
- ♣ Widdis, D. (S.F). psutil 5.8.0. agosto 03, 2021, de Pypi Sitio web: <https://pypi.org/project/psutil/>
- ♣ Python. (S.F). os — Miscellaneous operating system interfaces. julio 23, 2021, de Python Sitio web: <https://docs.python.org/3/library/os.html>
- ♣ Anónimo. (S.F). signal — Set handlers for asynchronous events. julio 23, 2021, de Python Sitio web: <https://docs.python.org/3/library/signal.html#module-signal>
- ♣ PyPi. (2020). bluedot 2.0.0. marzo 29, 2021, de PyPi Sitio web: <https://pypi.org/project/bluedot/>
- ♣ PyPi. (2020). Blue Dot. marzo 29, 2021, de PyPi Sitio web: <https://bluedot.readthedocs.io/en/latest/>