Universidad Nacional Autónoma de México

Facultad de Ingeniería

Fundamentos de Sistemas Embebidos (1858)

Proyecto Final - Documentación de Gimnasio Inteligente

INTEGRANTES:

Aguilar Luna Gabriel Daniel

Jiménez Juárez Jesús

Grupo: 3

Profesora: M.A. Ayesha Sagrario Román García

Semestre: 2021-1

Fecha de entrega: 21/enero/2021

Índice

Introducción

Las casas inteligentes son aquellas en las que las instalaciones eléctricas, de gas o de agua están monitorizadas y controladas a distancia desde un dispositivo electrónico. Normalmente suele ser un smartphone o computadora.

La automatización de los hogares, gracias a la telemedida y la domótica, está sirviendo para disponer de casas hechas a la medida, ajustadas a las necesidades particulares de cada individuo.

Una casa inteligente te puede avisar en el momento en el que exista una fuga de agua antes de que aparezca la gotera, del mismo modo que puede detectar incendios antes de ver el fuego.

Las capacidades de las casas inteligentes son muy amplias. Todo lo que tenga que ver con el encendido y apagado de los aparatos electrónicos e instalaciones de iluminación y climatización, combinado con la incorporación de sensores de presencia, de temperatura o de cualquier otro tipo, hacen que las posibilidades de personalización de las casas inteligentes sean casi infinitas.

Además de la seguridad y el confort que aportan las casas inteligentes, es muy importante la capacidad que tienen estos sistemas para el ahorro y la eficiencia energética. El diseño de las instalaciones de las casas inteligentes incluye aparatos electrónicos eficientes que, mediante su gestión integrada, consigue generar ahorros de agua, electricidad y combustible.

En nuestro caso nos tocó implementar un gimnasio en casa inteligente, el cual cuenta con sensores de movimiento, proximidad, temperatura, humedad, micrófonos, entre otros sensores que permiten que el gimnasio pueda cumplir con una gran variedad de tareas; algunas de ellas completamente automatizadas. Cuenta con lámparas inteligentes, tiras de led que se conectan a la red, un sistema de ventilación, persianas automáticas y más. El usuario puede interactuar con el sistema con comandos de voz a través del micrófono, con los bot de Telegram, con la aplicación BlueDot, una pantalla LCD que muestra información clave del estado del sistema y con todos los sensores antes mencionados.

Descripción del proyecto

Este proyecto tiene como finalidad la realización de un gimnasio inteligente dentro de una casa, aplicaremos varios códigos para que distintos materiales hagan su trabajo, que más adelante platicaremos de ellos. Se realizó por medio de la Raspberry Pi Modelo 4B todo está implementación, dentro del vídeo se ve el funcionamiento de cada uno de los programas y cuál va a ser su función dentro del Gimnasio Inteligente.

También realizamos una simulación de nuestro posible gimnasio inteligente, en donde todos los materiales ya están instalados en la infraestructura del cuarto.

Dentro de la documentación mencionaremos cuáles son los costos de cada uno de nuestros materiales, así como el diseño.

Esta simulación la realizamos en Visual Studio 2017, en el que nos basamos en un proyecto de la asignatura de Computación Gráfica antes implementada, nada más lo que hicimos fue realizar los modelos de las figuras que se muestran (como son pesas, ventiladores, bancas, mesas, etc), las animaciones que se van a mostrar y además la realización del cuarto.

Utilizaremos software como Telegram y Bluedot para poder mandar a realizar las tareas de manera remota, esto con el fin de facilitar el uso de nuestro proyecto. También haremos uso de Google Assistant implementado en la Raspberry, así como el Access Point.

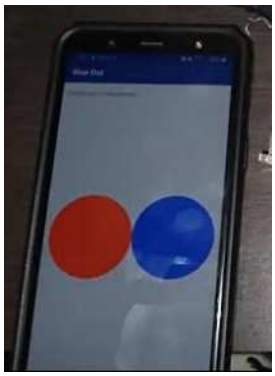
Esa fue una descripción general, ahora pasemos a lo que hace cada uno de los elementos.

Ledsmusicales.py

Los materiales para este programa son los siguientes:

- Bluedot
- 8 leds de diferentes colores
- Resistencias de 330 ohms
- Descarga del paquete *lightshowpi*

Este consta, como dice el nombre del archivo, de LEDs musicales, en donde se van a ir prendiendo y apagando las luces al ritmo de la canción que esté en reproducción. Está implementado con dos botones, el rojo es para iniciarlo y el azul es para finalizar el programa.

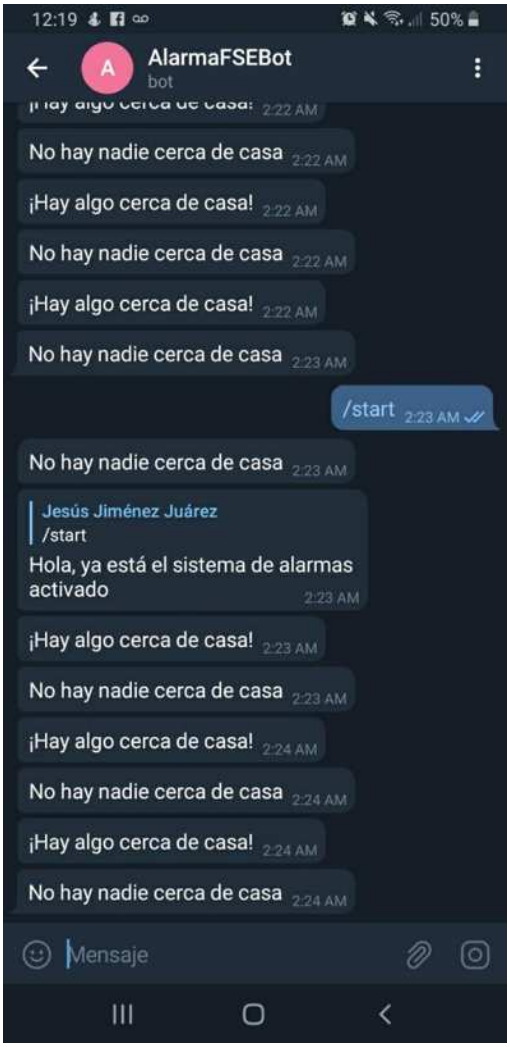


AlarmaPV.py

Los materiales para este programa son los siguientes:

- Bot de Telegram
- Resistencias de 330 ohms
- Sensor de proximidad

Este programa lo que hace es emitir una señal de alerta en caso de que detecte algún movimiento por la ventana. Dentro de Telegram, se debe de enviar el comando */start* para que le envíe un aviso para que el usuario esté informado de que algo entró y vaya a verificar qué sucedió.



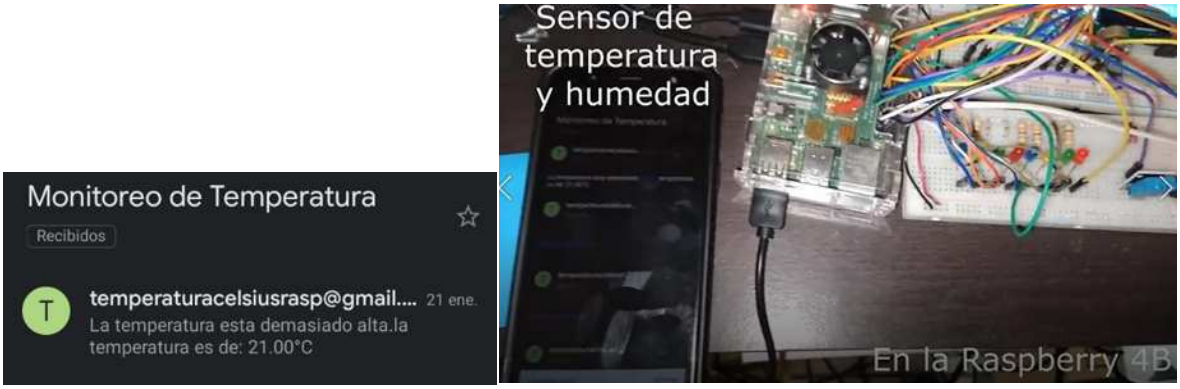
temperatura_humedad.py

Los materiales para este programa son los siguientes:

- Cuenta de correo electrónico para el envío de alerta de la temperatura.
- Resistencias de 330 ohms
- Sensor de temperatura DHT11
- Smart TV o alguna pantalla para poder visualizar la interfaz gráfica de la Raspberry Pi.

En la pantalla de la Smart TV se va a mostrar cuál es la temperatura y humedad que el sensor detectó en ese instante, nos estará dando estos datos entre 5 y 15 segundos y en caso de que llegue a detectar que la temperatura que registró supera

al que el usuario no quiere que sobrepase, lo mejor será que mande una señal de alerta.



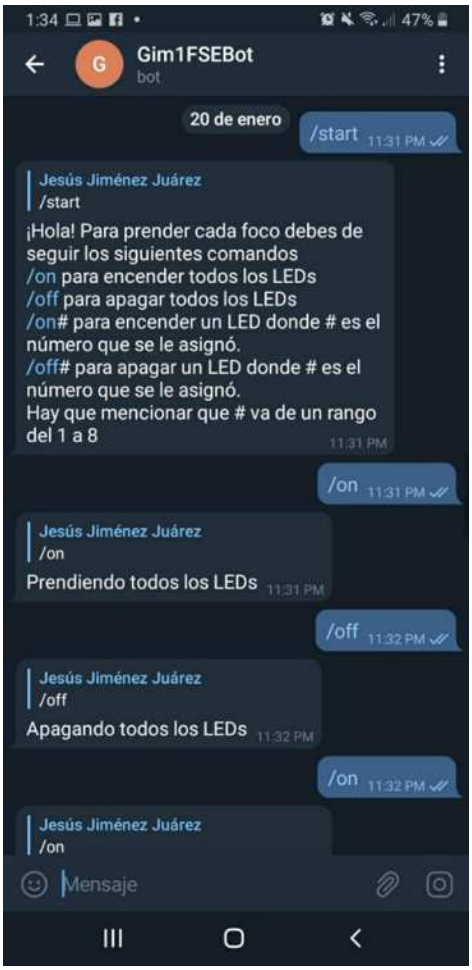
masterbot.py

Los materiales para este programa son los siguientes:

- LEDs blancos
- Resistencias de 330 ohms
- Bot de Telegram

Este código principal utiliza otros dos códigos que corre por medio de hilos, *bot1.py* y *bot2.py.* El primero lo que hace es apagar y prender las luces por medio de un bot de Telegram, con ella podemos utilizar los siguientes comandos.

- **/start** inicia el programa para después ir aceptando los comandos
- **/help** despliega los comandos necesarios para poder usarlos y mandarlos por el bot
- **/on** enciende todas las luces
- **/off** apaga todas las luces
- **/on#** enciende una luz en específico
- **/off#** apaga una luz en específico
- Hay que mencionar que # es el número que se le asignó a cada uno de los LEDs, el rango es de 1 a 8.



El segundo bot lo utilizamos para la atenuación de las luces, con ella podemos utilizar los siguientes comandos:

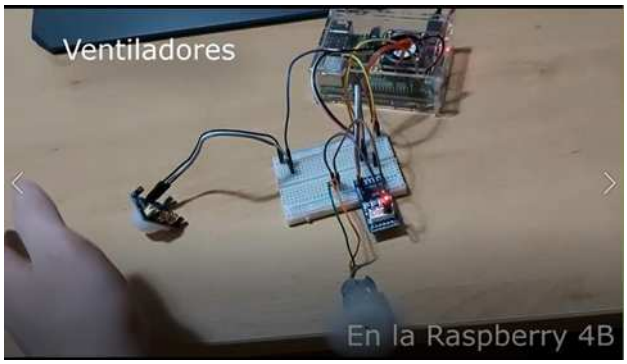
- **/help** despliega los comandos necesarios para poder usarlos y mandarlos por el bot
- Un número entre el 0 y el 100 para la atenuación de las luces, donde 0 es el mínimo y el 100 el máximo de que los LEDs puedan atenuarse.

ventilador.py

Los materiales para este programa son los siguientes:

- Sensor PIR
- Controlador de motor
- Motor DC de 5V

Este programa lo que hace es que si hay una mano cercana al sensor, encienda los ventiladores, cuando se vuelve a repetir lo anterior, es entonces cuando lo apaga.



LCD.py

Los materiales para este programa son los siguientes:

- LCD
- Potenciómetro de 10 kohms

Muestra la información cada cierto tiempo sobre los nombres de los integrantes del equipo. Hay que mencionar que esto no se realizó en la simulación

Materiales utilizados en general.

- Raspberry Pi 4B
- Micrófono USB
- SmartTV
- LCD
- 18 LEDs de diferentes colores
- Potenciómetro de 10 kohms
- Sensor de proximidad
- Sensor PIR
- Resistencias de diferentes valores

- Sensor de temperatura y humedad
- Motor DC de 5V
- Bluedot (Software PC)
- Telegram (Software en celular o PC)
- Visual Studio 2017 (Software en PC)
- Jumpers
- Protoboard
- Controlador de motor
- Cuenta de correo electrónico gmail para enviar la temperatura

Costos

Material	**Descripción**	**Costo (En México)**
Google Home	Estará instalado en un punto específico del cuarto en donde se pueda escuchar de buena manera al usuario que le va dando instrucciones sobre lo que tiene que hacer.	3,840.81
Sensor de movimiento	Estará instalado en las ventanas para estar atento ante intrusos que hayan ingresado por este lugar.	499.00
9 bombillas LED Inteligente WiFi de colores	Son los focos que utilizaremos como iluminación y estarán distribuidos en diferentes áreas. Serán inteligentes dado que pueden tener diferentes colores, además de apagarse y encenderse de manera remota.	2,700
10m Tira Leds	Se utilizará para que pueda realizar el juego de luces del que hablamos anteriormente (Luces musicales)	930
3 ventiladores de Techo Symphony con luz y Wi-Fi 54 pulgadas	Se instalarán para que puedan refrescar a la habitación.	25,134.15
Govee Sensor de humedad y de temperatura WiFi	Revisará el estado de la temperatura y humedad de la habitación	1,812.32
Raspberry Pi 4B	Será la cabeza de que todo funcione (Se recomienda ampliamente la de 8 GB para que procese todo de manera correcta)	2,229.00

Instalación	Se realizará la configuración necesaria para que pueda funcionar todo	7,500
Programación	Se van a realizar los códigos necesarios para que todo pueda funcionar con la Raspberry al mismo tiempo	20,000
	Total	64,645.13

NOTA: Otro material no previsto en la lista de costos, se vería con el usuario para la obtención de su permiso.

Diseño

Nos basamos en el siguiente modelo:

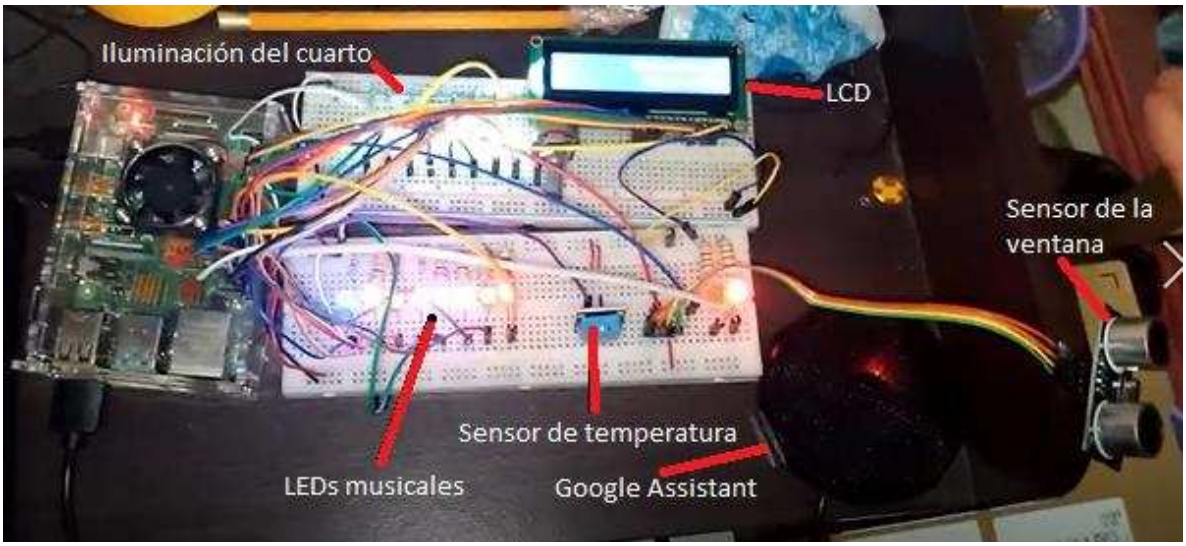
A continuación, se muestra el croquis del gimnasio con el que trabajamos:

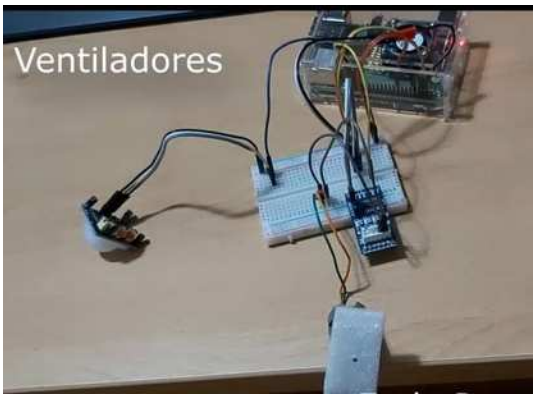
Identificación de elementos que funcionan con IoT

Dentro de nuestro proyecto, nos percatamos que todo funciona con IoT.

- **Iluminación Inteligente:** Esto se puede realizar con la utilización de Google Assistant, en donde con una simple instrucción que salió por medio de la voz del usuario, se pueden manejar de una manera sencilla todas las bombillas dentro de una habitación inteligente. También se puede realizar con sensores, aplicaciones como Telegram o BlueDot, entre otros.
- **Sensor de temperatura:** Al igual que el anterior, se pueden utilizar sensores, aplicaciones y Google Assistant. Pero nos dimos cuenta que, durante la realización del proyecto, que se pueden mandar correos electrónicos a los usuarios de que la temperatura ha sobrepasado al que el usuario había configurado como el máximo permitido dentro de la habitación
- **Sensor de proximidad:** Podemos mandar un mensaje por medio de Telegram al usuario de algún intruso dentro de nuestro hogar y también puede utilizarse para permitir el manejo de un objeto, como lo son las bombillas inteligentes o con los propios ventiladores.
- **Ventiladores Inteligentes:** Se pueden utilizar sensores como lo vimos en el proyecto, también pueden ser capaces de ser empleados por el sensor de temperatura, en donde si éste último ve que hay mucho calor dentro de la habitación, entonces es ahí en donde va a entrar al desquite el ventilador para disminuir la temperatura.
- **Google Assistant:** Esta vendría siendo una poderosa herramienta que puede dar información de lo que sea, así como la reproducción de música y mandar a reproducir vídeos a un dispositivo en específico.

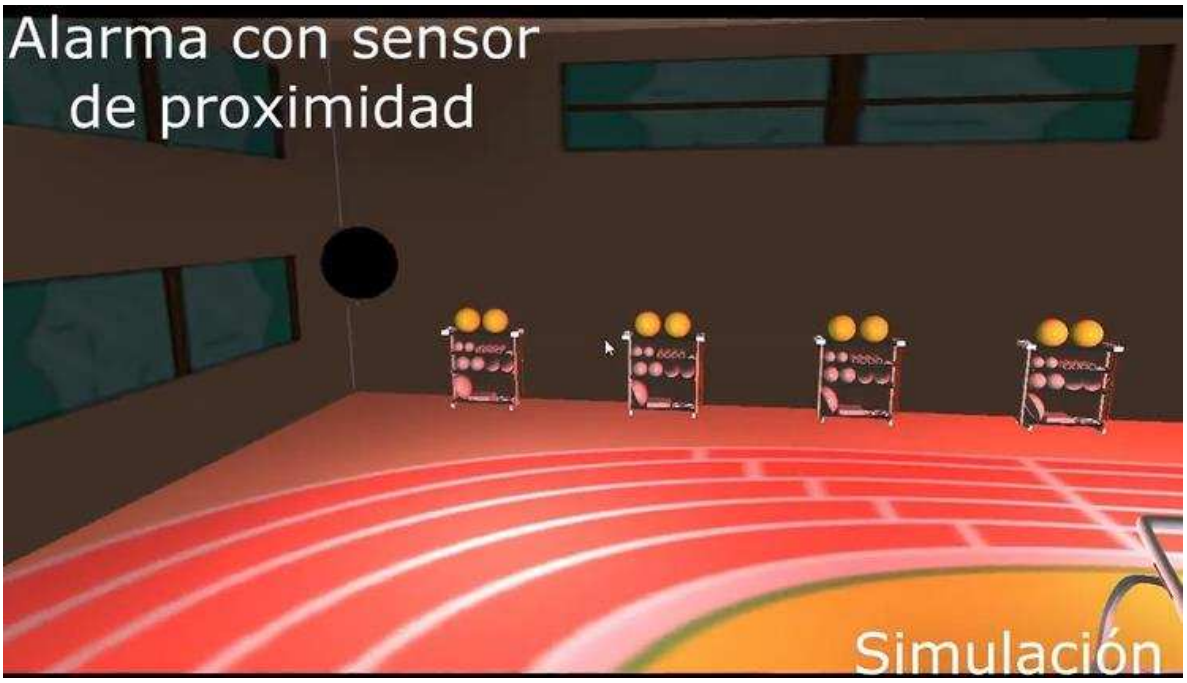
Implementación (Fotografías, imágenes de simulación, etc.)





Luces musicales

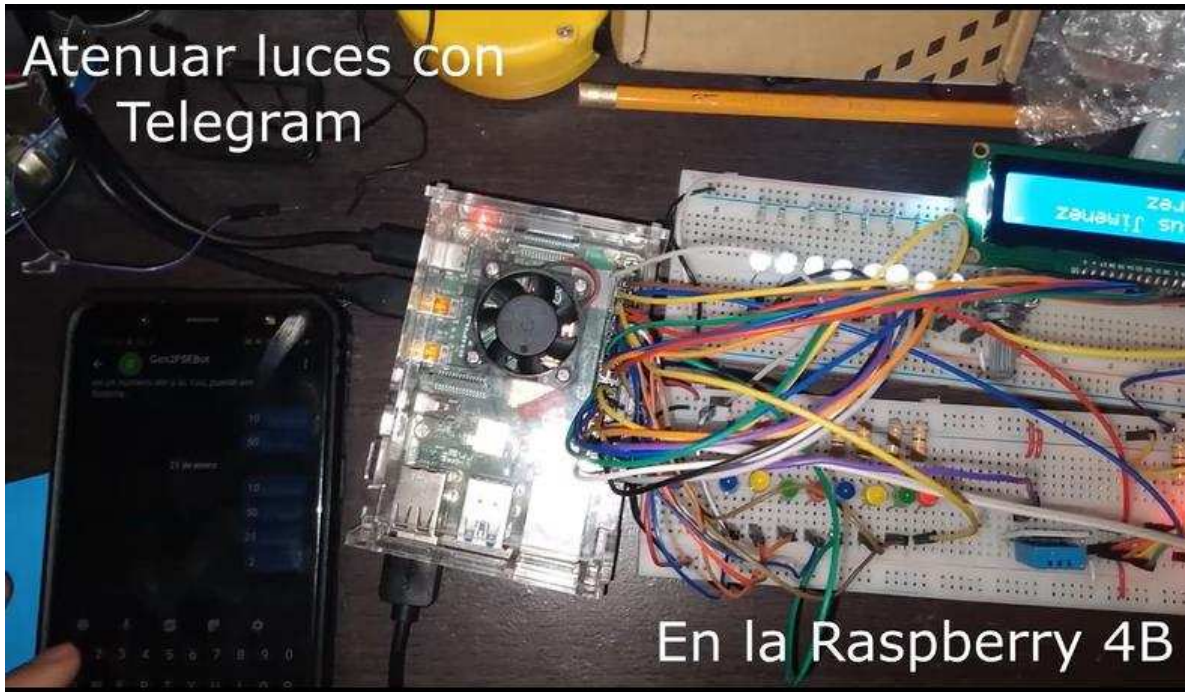
Alarma con sensor de proximidad



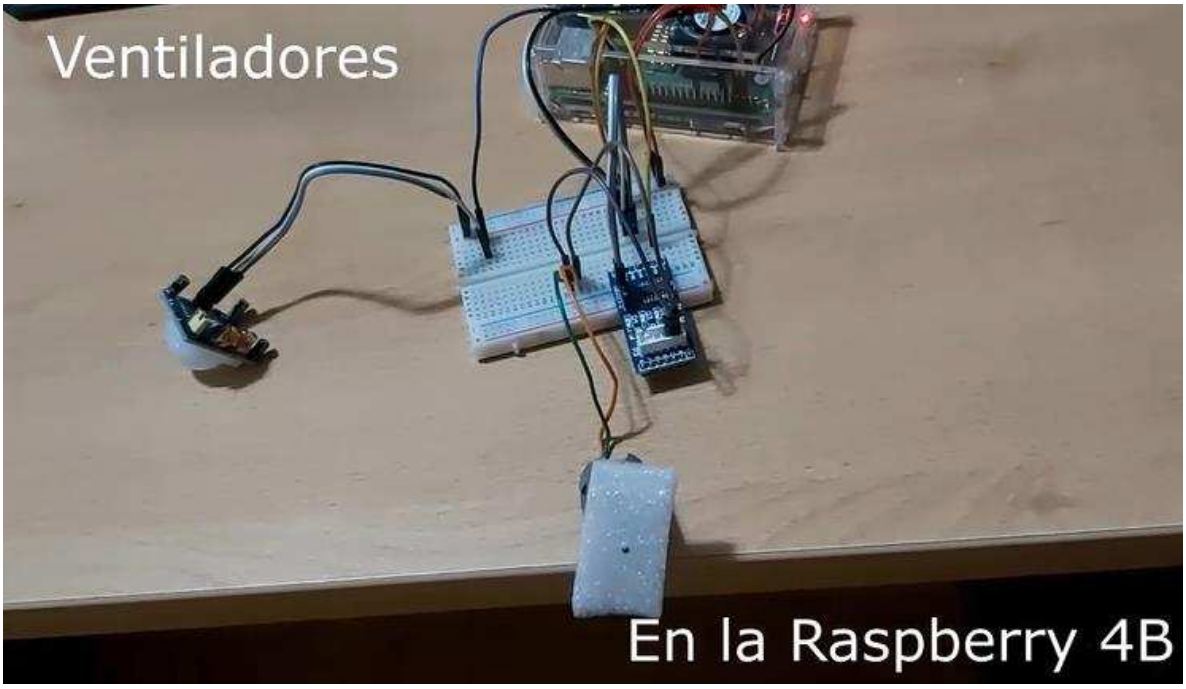
Sensor de temperatura y humedad

Iluminación


Apagar y/o prender
luces con Telegram
Simulación


Atenuar luces con
Telegram
En la Raspberry 4B

Ventiladores

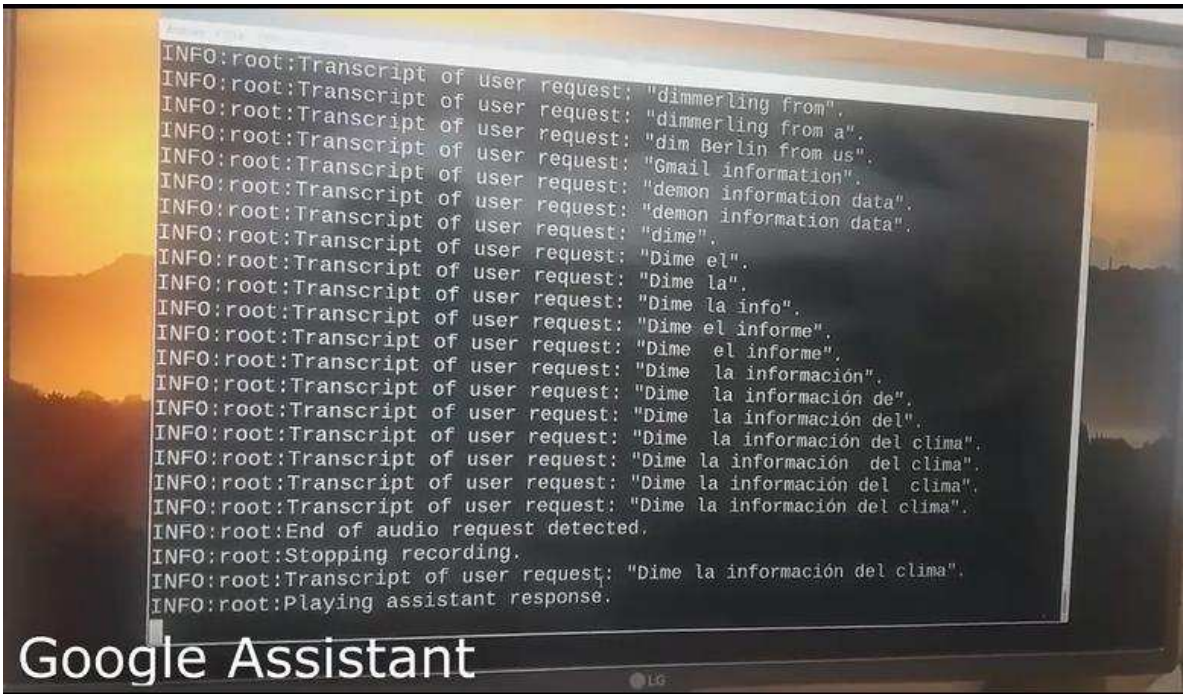



Ventiladores
Simulación

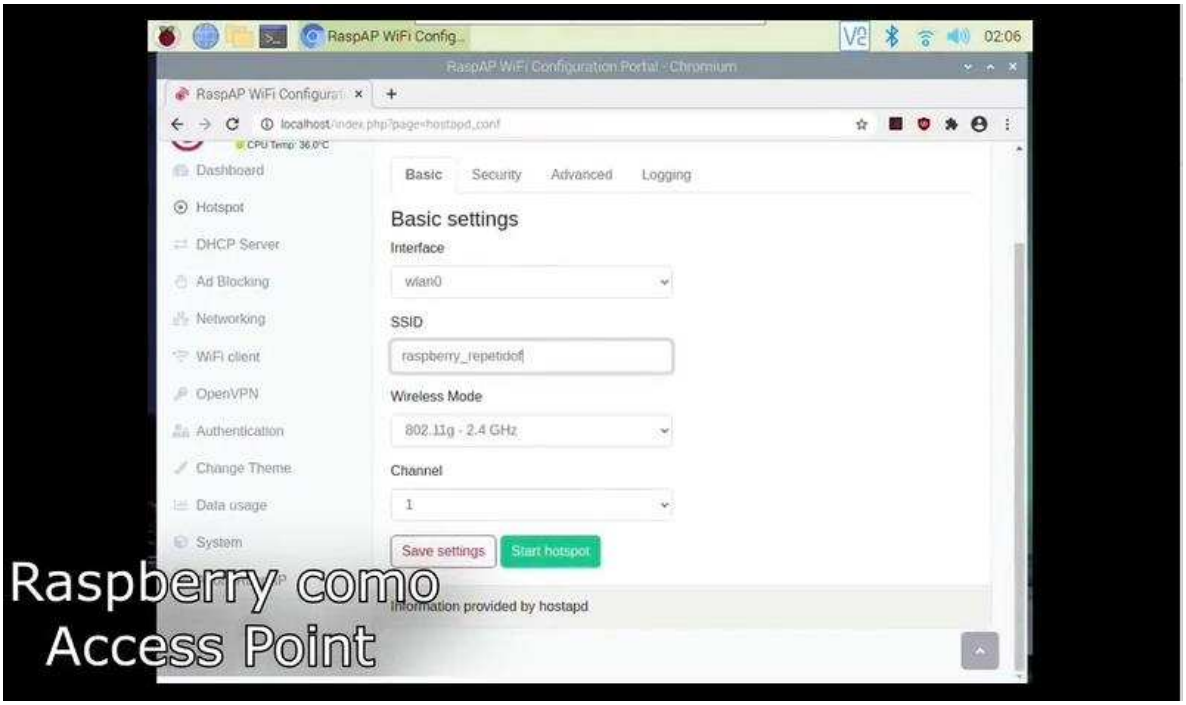
LCD


LCD
Proyecto Final
GIMNASIO
En la Raspberry 4B

Google Assistant



Access Point



Código

AlarmaPV

```
#!/usr/bin/python

# Realiza el funcionamiento de la alarma en una ventana, en caso de detectar algo
#cercano, se encenderá la alarma y avisará al usuario a través de un bot de telegram.

from gpiozero import DistanceSensor, LED
from signal import pause
import telebot
import pygame.mixer
from pygame.mixer import Sound # Load the required library

pygame.mixer.init()
ventana_sensor = DistanceSensor(14, 15, max_distance=1,threshold_distance=0.9)
sound = Sound('alerta.wav')

API_TOKEN = "1491889902:AAHHVW6R5-VXuHdyJIzM11_59dZGTp-BNG0"
bot = telebot.TeleBot(API_TOKEN)
chat_id = 0

#Manda un mensaje de ayuda al usuario por telegram
@bot.message_handler(commands=['help','start'])
def ayuda(message):
    global chat_id
    chat_id  = message.chat.id
    bot.reply_to(message,"""Hola, ya está el sistema de alarmas activado""")
#Función en donde se activa la alarma, pero intenta mandar un mensaje por medio del
#bot en Telegram, en caso de no poderse, se imprimirá en la Smart TV que no pudo
#realizarlo.
def activaAlarma(sound):
    global chat_id,bot
    try:
        sound.play()
        print("Detectando alguna proximidad")
        bot.send_message(chat_id, '¡Hay algo cerca de casa!')
    except:
        print("Da el comando /start o help para que se muestre en el bot el mensaje")
#Función en donde se desactiva la alarma, pero intenta mandar un mensaje por medio
#del bot en Telegram, en caso de no poderse, se imprimirá en la Smart TV que no pudo
#realizarlo.
def desactivaAlarma(sound):
    global chat_id,bot
    try:
        sound.stop()
        print("No se percibe nada")
        bot.send_message(chat_id, 'No hay nadie cerca de casa')
    except:
```

```
        print("Da el comando /start o help para que se muestre en el bot el mensaje")

ventana_sensor.when_in_range = lambda:activaAlarma(sound)
ventana_sensor.when_out_of_range = lambda:desactivaAlarma(sound)

bot.polling()
```

Bot1

```
#!/usr/bin/python
# Este programa lo que va a hacer es que, dependiendo del comando que el usuario
#ingrese en Telegram, entonces va a encender, apagar un LED en específico o todas al
#mismo tiempo.

import settings
import telebot

#LED
def on(pin):
        settings.ledList[pin] = 1
        return

def off(pin):
        settings.ledList[pin] = 0
        return

#LED
def onall():
        lista = [1,2,3,4,5,6,7,8]
        for pin in lista:
        settings.ledList[pin] = 1
        return

def offall():
        lista = [1,2,3,4,5,6,7,8]
        for pin in lista:
        settings.ledList[pin] = 0
        return

API_TOKEN = '1438902983:AAFVShpVkfuo7iNwMrOQ1Kd_K7DOc-4AP-4'

bot = telebot.TeleBot(API_TOKEN)

#Mandará un mensaje de ayuda al usuario en caso de no saber qué hacer con los
comandos /help y /start
# Handle '/help'
@bot.message_handler(commands=['help','start'])
def ayuda(message):
        offall()
```

```
        bot.reply_to(message, """\
¡Hola! Para prender cada foco debes de seguir los siguientes comandos
/on para encender todos los LEDs
/off para apagar todos los LEDs
/on# para encender un LED donde # es el número que se le asignó.
/off# para apagar un LED donde # es el número que se le asignó.
Hay que mencionar que # va de un rango del 1 a 8 \
""")

@bot.message_handler(commands=['on','On','oN','ON'])
def ledson(message):
        onall()
        bot.reply_to(message, """\
Prendiendo todos los LEDs\
""")

@bot.message_handler(commands=['off','Off','OfF','oFf','OFf','ofF','oFF','OFF'])
def ledsoff(message):
        offall()
        bot.reply_to(message, """\
Apagando todos los LEDs\
""")

@bot.message_handler(commands=['on1','On1','oN1','ON1'])
def ledon1(message):
        on(1)
        bot.reply_to(message, """\
Prendiendo el LED 1\
""")

@bot.message_handler(commands=['on2','On2','oN2','ON2'])
def ledon2(message):
        on(2)
        bot.reply_to(message, """\
Prendiendo el LED 2\
""")

@bot.message_handler(commands=['on3','On3','oN3','ON3'])
def ledson3(message):
        on(3)
        bot.reply_to(message, """\
Prendiendo el LED 3\
""")

@bot.message_handler(commands=['on4','On4','oN4','ON4'])
def ledson4(message):
        on(4)
        bot.reply_to(message, """\
Prendiendo el LED 4\
```

```
""")

@bot.message_handler(commands=['on5','On5','oN5','ON5'])
def ledson5(message):
        on(5)
        bot.reply_to(message, """\
Prendiendo el LED 5\
""")

@bot.message_handler(commands=['on6','On6','oN6','ON6'])
def ledson6(message):
        on(6)
        bot.reply_to(message, """\
Prendiendo el LED 6\
""")

@bot.message_handler(commands=['on7','On7','oN7','ON7'])
def ledson7(message):
        on(7)
        bot.reply_to(message, """\
Prendiendo el LED 7\
""")

@bot.message_handler(commands=['on8','On8','oN8','ON8'])
def ledson8(message):
        on(8)
        bot.reply_to(message, """\
Prendiendo el LED 8\
""")

@bot.message_handler(commands=['off1','Off1','OfF1','oFf1','OFf1','ofF1','oFF1','OFF1'])
def ledoff1(message):
        off(1)
        bot.reply_to(message, """\
Apagando el LED 1\
""")

@bot.message_handler(commands=['off2','Off2','OfF2','oFf2','OFf2','ofF2','oFF2','OFF2'])
def ledoff2(message):
        off(2)
        bot.reply_to(message, """\
Apagando el LED 2\
""")

@bot.message_handler(commands=['off3','Off3','OfF3','oFf3','OFf3','ofF3','oFF3','OFF3'])
def ledoff3(message):
        off(3)
        bot.reply_to(message, """\
Apagando el LED 3\
```

```
""")

@bot.message_handler(commands=['off4','Off4','OfF4','oFf4','OFf4','ofF4','oFF4','OFF4'])
def ledoff4(message):
        off(4)
        bot.reply_to(message, """\
Apagando el LED 4\
""")

@bot.message_handler(commands=['off5','Off5','OfF5','oFf5','OFf5','ofF5','oFF5','OFF5'])
def ledoff5(message):
        off(5)
        bot.reply_to(message, """\
Apagando el LED 5\
""")

@bot.message_handler(commands=['off6','Off6','OfF6','oFf6','OFf6','ofF6','oFF6','OFF6'])
def ledoff6(message):
        off(6)
        bot.reply_to(message, """\
Apagando el LED 6\
""")

@bot.message_handler(commands=['off7','Off7','OfF7','oFf7','OFf7','ofF7','oFF7','OFF7'])
def ledoff7(message):
        off(7)
        bot.reply_to(message, """\
Apagando el LED 7\
""")

@bot.message_handler(commands=['off8','Off8','OfF8','oFf8','OFf8','ofF8','oFF8','OFF8'])
def ledoff8(message):
        off(8)
        bot.reply_to(message, """\
Apagando el LED 8\
""")

# Handle all other messages with content_type 'text' (content_types defaults to ['text'])
@bot.message_handler(func=lambda message: True)
def echo_message(message):
        bot.reply_to(message, message.text)

bot.polling()
```

Bot2

```
#!/usr/bin/python
# Realiza la atenuación de las luces
```

```
import settings
import telebot

TOKEN = '1404908081:AAFyzExO0EMpH8hHj6GLTr18jLh5ww5LNFo'
bot = telebot.TeleBot(TOKEN)

def listener(mensajes):
    for m in mensajes:
        chat_id = m.chat.id
        texto = m.text
        print('ID: ' + str(chat_id) + ' - MENSAJE: ' + texto)
        if texto == "/help" or texto == "/start":
            break
        try:
            numero = int(texto)
            if numero >= 0  and numero <= 100:
                atenuacion(numero,m)
            else:
                bot.reply_to(m, """\
                Ingresa un número del 0 al 100\
                """)
        except:
            bot.reply_to(m, """\
            Ingresa un número del 0 al 100\
            """)
#Hace la atenuación
def atenuacion(numero,m):
    settings.intensidadLeds = numero/100
#Manda un mensaje para orientar al usuario
@bot.message_handler(commands=['help','start'])
def ayuda(message):
    bot.reply_to(message, """\
¡Hola! Para atenuar los LEDs, simplemente dé \
un número del 0 al 100, puede ser flotante.\
""")


bot.set_update_listener(listener)

bot.polling()
```

LCD

```
#!/usr/bin/python

#Este programa lo que va a realizar es imprimir en la LCD varios mensajes
import time
import RPi.GPIO as GPIO
from time import sleep
```

```
from signal import pause


# Assignment of GPIO pins
LCD_RS = 2
LCD_E  = 3
LCD_DATA4 = 10
LCD_DATA5 = 9
LCD_DATA6 = 11
LCD_DATA7 = 8

LCD_WIDTH = 16      # Characters per line
LCD_LINE_1 = 0x80   # Address of the first line of the display
LCD_LINE_2 = 0xC0   # Address of the second line of the display
LCD_CHR = GPIO.HIGH
LCD_CMD = GPIO.LOW
E_PULSE = 0.0005
E_DELAY = 0.0005

def lcd_send_byte(bits, mode):
    # Set pins to LOW
    GPIO.output(LCD_RS, mode)
    GPIO.output(LCD_DATA4, GPIO.LOW)
    GPIO.output(LCD_DATA5, GPIO.LOW)
    GPIO.output(LCD_DATA6, GPIO.LOW)
    GPIO.output(LCD_DATA7, GPIO.LOW)
    if bits & 0x10 == 0x10:
      GPIO.output(LCD_DATA4, GPIO.HIGH)
    if bits & 0x20 == 0x20:
      GPIO.output(LCD_DATA5, GPIO.HIGH)
    if bits & 0x40 == 0x40:
      GPIO.output(LCD_DATA6, GPIO.HIGH)
    if bits & 0x80 == 0x80:
      GPIO.output(LCD_DATA7, GPIO.HIGH)
    time.sleep(E_DELAY)
    GPIO.output(LCD_E, GPIO.HIGH)
    time.sleep(E_PULSE)
    GPIO.output(LCD_E, GPIO.LOW)
    time.sleep(E_DELAY)
    GPIO.output(LCD_DATA4, GPIO.LOW)
    GPIO.output(LCD_DATA5, GPIO.LOW)
    GPIO.output(LCD_DATA6, GPIO.LOW)
    GPIO.output(LCD_DATA7, GPIO.LOW)
    if bits&0x01==0x01:
      GPIO.output(LCD_DATA4, GPIO.HIGH)
    if bits&0x02==0x02:
      GPIO.output(LCD_DATA5, GPIO.HIGH)
    if bits&0x04==0x04:
      GPIO.output(LCD_DATA6, GPIO.HIGH)
    if bits&0x08==0x08:
```

```
    GPIO.output(LCD_DATA7, GPIO.HIGH)
  time.sleep(E_DELAY)
  GPIO.output(LCD_E, GPIO.HIGH)
  time.sleep(E_PULSE)
  GPIO.output(LCD_E, GPIO.LOW)
  time.sleep(E_DELAY)

def display_init():
  lcd_send_byte(0x33, LCD_CMD)
  lcd_send_byte(0x32, LCD_CMD)
  lcd_send_byte(0x28, LCD_CMD)
  lcd_send_byte(0x0C, LCD_CMD)
  lcd_send_byte(0x06, LCD_CMD)
  lcd_send_byte(0x01, LCD_CMD)

def lcd_message(message):
  message = message.ljust(LCD_WIDTH," ")
  for i in range(LCD_WIDTH):
    lcd_send_byte(ord(message[i]),LCD_CHR)


if __name__ == '__main__':
  # initialize
  GPIO.setmode(GPIO.BCM)
  GPIO.setwarnings(False)
  GPIO.setup(LCD_E, GPIO.OUT)
  GPIO.setup(LCD_RS, GPIO.OUT)
  GPIO.setup(LCD_DATA4, GPIO.OUT)
  GPIO.setup(LCD_DATA5, GPIO.OUT)
  GPIO.setup(LCD_DATA6, GPIO.OUT)
  GPIO.setup(LCD_DATA7, GPIO.OUT)

  display_init()

#Estará en un ciclo infinito en donde mostrará cada uno de los siguientes mensajes:
#Sistemas embebidos
#Jesús Jiménez Juárez
#Aguilar Luna Gabriel Daniel
#Proyecto Final - Gimnasio Inteligente
  while True:
    lcd_send_byte(LCD_LINE_1, LCD_CMD)
    lcd_message("Sistemas")
    lcd_send_byte(LCD_LINE_2, LCD_CMD)
    lcd_message("Embebidos")
    sleep(4)
    lcd_send_byte(LCD_LINE_1, LCD_CMD)
    lcd_message("Jesus Jimenez")
    lcd_send_byte(LCD_LINE_2, LCD_CMD)
    lcd_message("Juarez")
    sleep(4)
```

```
        lcd_send_byte(LCD_LINE_1, LCD_CMD)
        lcd_message("Gabriel Daniel")
        lcd_send_byte(LCD_LINE_2, LCD_CMD)
        lcd_message("Aguilar Luna")
        sleep(4)
        lcd_send_byte(LCD_LINE_1, LCD_CMD)
        lcd_message("Proyecto Final")
        lcd_send_byte(LCD_LINE_2, LCD_CMD)
        lcd_message("*GIMNASIO*")
        sleep(4)
    GPIO.cleanup()
```

Leds musicales

```
#!/usr/bin/python
#Este programa se utilizará para la función de los leds musicales, con el uso de BlueDot
from bluedot import BlueDot
from signal import pause
import os, sys
import threading

var = 0
var2 = 0
once = 0
once2 = 0
salir = 0
bd = BlueDot(cols=2, rows=1)
bd[0,0].color = "red"
#Función que importa el programa modulomusica
def funcMusic():
    import modulomusica
# Si presiona el botón 1, entonces va a realizar la ejecución de los leds musicales
def pressed_1(pos):
    global var,var2
    var = 1
    var2 = 0
# Si presiona el botón 2, saldrá del programa y matará la ejecución de los leds musicales
def pressed_2(pos):
    global var,var2
    var = 0
    var2 = 1

b1 = threading.Thread(target=funcMusic, name='modulomusica', daemon = True)

while salir == 0:

    bd[0,0].when_pressed = pressed_1
    bd[1,0].when_pressed = pressed_2
```

```
    if var:
        if var2 == 0:
            if once == 0:
                b1.start()
                print("button 1 pressed")
                once = 1
                once2 = 0

    else:
        if var2 == 1:
            if once2 == 0:
                print("button 2 pressed")
                kill = os.popen("sudo pgrep python").read()
                print(type(kill),kill[-6:-1])
                inst = ''
                for i in kill[-6:-1]:
                    print(i)
                    if ord(i) >= 48 and ord(i) <= 57:
                        print(i,type(i),ord(i))
                        inst = inst + i
                evento = "sudo kill" + " " + inst
                print(evento)
                os.system(evento)
                salir = 1
print("Luces musicales apagadas")
```

Master Bot

```
#!/usr/bin/python
# main.py
# Son hilos que realizan  los programas tanto de la atenuación (bot2.py) como el encendido (bot1.py) y el a
from gpiozero import PWMLED
import settings
import threading

settings.init()

def funcBot1():
    import bot1
def funcBot2():
    import bot2
#def funcBot3():
#    import temperatura_humedad

b1 = threading.Thread(target=funcBot1, name='bot1', daemon = True)
b2 = threading.Thread(target=funcBot2, name='bot2', daemon = True)
#b3 = threading.Thread(target=funcBot3, name='temperatura_humedad', daemon = True)
b1.start()
b2.start()
```

```
#b3.start()


mapeo_leds_gpio =
{1:PWMLED(26),2:PWMLED(19),3:PWMLED(13),4:PWMLED(6),5:PWMLED(5),6:PWMLED(0),7:PWMLED

while True:
    #print(settings.ledList)
    #print(settings.intensidadLeds)
    for x, y in mapeo_leds_gpio.items():
        if settings.ledList[x] == 1:
            y.value = settings.intensidadLeds
        else:
            y.value = 0
```

Módulo música

```
import os, sys
#Realizando el comando os.system en donde se va a reproducir música y se va a realizar
el juego de luces.
def moduloled():
    print("button 1 pressed")
    os.system("sudo python3 /home/pi/lightshowpi/py/synchronized_lights.py --
file=/home/pi/lightshowpi/music/sample/VivaLaVida.mp3")

moduloled()
```

Proyecto Final

```
#!/usr/bin/env python

import subprocess


# Iterable con las rutas de los scripts
scripts_paths = ("/home/pi/Documents/Fundamentos de Sistemas
Embebidos./Proyecto/masterbot.py",
                 "/home/pi/Documents/Fundamentos de Sistemas
Embebidos./Proyecto/temperatura_humedad.py",
                 "/home/pi/Documents/Fundamentos de Sistemas
Embebidos./Proyecto/Ledsmusicales.py",
                 "/home/pi/Documents/Fundamentos de Sistemas
Embebidos./Proyecto/LCD.py",
                 "/home/pi/Documents/Fundamentos de Sistemas
Embebidos./Proyecto/AlarmaPV.py")

# Creamos cada proceso
procesos = [subprocess.Popen(["python", script]) for script in scripts_paths]
```

```
# Esperamos a que todos los subprocesos terminen.
for proceso in procesos:
    proceso.wait()

# Resto de código a ejecutar cuando terminen todos los subprocesos.
```

Settings

```
#!/usr/bin/python

# settings.py

def init():
    global ledList, intensidadLeds
    ledList = [0,0,0,0,0,0,0,0,0,0]
    intensidadLeds = 1
```

Temperatura y humedad

```
#!/usr/bin/python

#temperatura_humedad.py
#Librerias
import RPi.GPIO as GPIO
import Adafruit_DHT
import smtplib
from email.mime.text import MIMEText
import time
import os, sys

#Configuraciones de los pines GPIO de la Raspberry
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(20,GPIO.OUT) #Pin que corresponde al led rojo
GPIO.setup(16,GPIO.OUT) #Pin que corresponde al led verde

#Variables Globales
sensor = Adafruit_DHT.DHT11
pin_temperatura = 21 #pin de la raspberry que recibe los datos de temperatura y
humedad del sensor
correo_origen = 'temperaturacelsiusrasp@gmail.com' #correo de origen
contraseña = '21tempB4yrrebpsar'
correo_destino = 'tachuyer@gmail.com' # correo de destino
temperatura_umbral = 20.0  #temperatura límite en °C

#Función para el envio de correo electrónico
def Enviar_correo(temperatura):
```

```
    msg = MIMEText(f"La temperatura esta demasiado alta.la temperatura es de: {temperatura:.2f}°C")
    msg['Subject'] = 'Monitoreo de Temperatura'
    msg['From'] = correo_origen
    msg['To'] = correo_destino
    server = smtplib.SMTP('smtp.gmail.com',587)
    server.starttls()
    server.login(correo_origen,contraseña)
    server.sendmail(correo_origen,correo_destino,msg.as_string())
    print("Su Email ha sido enviado.")
    server.quit()

while True:

    humedad, temperatura = Adafruit_DHT.read_retry(sensor, pin_temperatura)

    if humedad is not None and temperatura is not None:
        print(f'Temperatura={temperatura:.2f}*C  Humedad={humedad:.2f}%')

    else:
        print('Fallo la lectura del sensor.Intentar de nuevo')

    #Condicional que me permite controlar cuando la temperatura sobrepase el límite
    if temperatura > temperatura_umbral:

        GPIO.output(20,True) # enciende el led rojo como señal de advertencia
        GPIO.output(16,False) # apaga el led verde
        Enviar_correo(temperatura)

    else:

        GPIO.output(20,False) # led rojo apagado
        GPIO.output(16,True) # mantenga el led verde encendido
    time.sleep(0.01)
GPIO.cleanup()
```

Ventilador

```
#!/usr/bin/env python3
#By: Aguilar Luna Gabriel Daniel
#FSE grupo: 03

from gpiozero import MotionSensor, PWMLED
from signal import pause

sensor = MotionSensor(21)
salida = PWMLED(20)
```

```
salida.source = sensor

pause()
```

Evaluación de viabilidad (El equipo es factible para la venta o implementación en casas u oficinas)

Es viable para la venta e implementación para aquellas personas que deseen construir una casa inteligente, esto dado las siguientes ventajas:

- **No hay necesidad de hacer una nueva instalación:** Es favorable dado que no se gasta dinero de más removiendo la antigua instalación de luces y realizando modificaciones a la infraestructura del hogar
- **Puede usar lo nuevo en tecnología:** Así como el ser humano avanza con el tiempo, también lo hace lo nuevo que va saliendo respecto a las tecnologías. Cada vez se van comercializando más los objetos inteligentes y a un costo que al consumidor le puede beneficiar.
- **Puede ayudar a las personas con discapacidades:** Las personas que no tienen sus extremidades, puede utilizar su voz para manejar la iluminación, así como los ventiladores. Mientras que las personas sordomudas pueden utilizar los sensores para lo mismo
- **El usuario puede llegar a hacer tareas más fácil:** El encender/apagar luces de manera remota, poner los ventiladores a andar o hasta hablar con el cuarto con Google Assistant puede darle una mayor facilidad al usuario

Pero también tiene sus desventajas

- **Mantenimiento cada determinado tiempo:** Como todo, el mantenimiento debe de darse en los equipos instalados, pero este puede ser muy costoso en caso de algún desperfecto.
- **Dependencia del Wi-Fi:** En caso de no tener en ciertos momentos una señal decente de Internet, los objetos inteligentes tienden a ser inútiles por momentos.

No se recomienda a aquellas personas que quisieran cambiar su infraestructura por una más inteligente, también dependerá del tamaño de la habitación, porque hay objetos como el ventilador que puede llegar a ser muy grande para la habitación

Conclusiones

Aguilar Luna Gabriel Daniel

Durante el desarrollo de este proyecto, nos estuvimos ayudando mutuamente en la realización de los programas, así como ver en qué fallaba el otro. Hubo veces en los que tuve que investigar porque me costó trabajo realizar varios programas, como lo fue el de las luces de lightshowpi, en donde a ninguno de los dos nos funcionaba,

hasta que aceptamos la ayuda de uno de nuestros compañeros y facilitó que este proyecto terminase de manera satisfactoria.

El conocimiento sobre la materia es bastante grande porque es interesante cómo es el manejo de la Raspberry Pi y todas sus funciones en el entorno de los sistemas embebidos, así como en la IoT. Nos hace pensar de la importancia que tiene a nivel académico, dado que así más ingenieros se van preparando para que en un futuro tengan diferentes habilidades que aprendieron en la asignatura.

Jiménez Juárez Jesús

Este proyecto se pudo implementar todo lo visto tanto en teoría como en laboratorio, ya que la mayor parte de los materiales los utilizamos para diferentes actividades y, con ello, llegar a un objetivo que fue la realización del gimnasio inteligente. Cada programa tuvo sus dificultades dado que se presentaban diferentes retos, como por ejemplo en el del sensor de temperatura y humedad, donde nos dimos a la tarea de si se podía avisar al usuario de otra forma distinta a la vista en clase (Con Telegram o por medio de LEDs) y nos encontramos que también se podía mandar correos electrónicos en Python.

Se nos ocurrían ideas distintas a cada uno pero veíamos que teníamos muy pocas opciones de implementarlo dado que uno tenía el material del otro o ya no alcanzaba el espacio de los GPIO en la Raspberry (en mi caso). Aquí lo importante es el aprendizaje que uno se lleva al terminar este tipo de proyectos, en donde ya pasado un semestre más, me doy cuenta de la evolución que tengo respecto a los conocimientos que se tienen al paso del tiempo y, sin duda alguna, esta no es la excepción.

Fuentes de consulta

13. API - Input Devices — Gpiozero 1.5.1 Documentation. (s. f.). gpiozero. Recuperado 12 de octubre de 2020, de https://gpiozero.readthedocs.io/en/stable/api_input.html

14. API - Output Devices — Gpiozero 1.5.1 Documentation. (s. f.). gpiozero. Recuperado 16 de octubre de 2020, de https://gpiozero.readthedocs.io/en/stable/api_output.html

CDMX Electrónica. (2021, 21 enero). Raspberry PI 4 Modelo B Versión 2GB/4GB/8GB RAM. UNIT Electronics. https://uelectronics.com/producto/raspberry-pi-4-modelo-b-version-2gb-4gb-8gb-ram/

Amazon. (s. f.). Govee Sensor de humedad de temperatura WiFi, funciona con Alexa, termómetro inalámbrico higrómetro de temperatura monitor de humedad para casa, invernadero, bodega de vino, humidor (WiFi 5G no soportado): Amazon.com.mx: Industria, Empresas y Ciencia. Recuperado 21 de enero de 2021, de https://www.amazon.com.mx/Govee-temperatura-inal%C3%A1mbrico-exportaci%C3%B3n-invernadero/dp/B07TWMSNH5

Ventiladores de Techo con WI-FI. (s. f.). HUNTER FAN. Recuperado 22 de enero de 2021, de https://www.hunterfan.com.mx/collections/ventiladores-de-techo-con-wi-fi
10m Tira Led Wifi Alex,tira De Luces 300led Impermeable Rgb. (s. f.). Mercado Libre. Recuperado 22 de enero de 2021, de https://articulo.mercadolibre.com.mx/MLM-796758250-10m-tira-led-wifi-alextira-de-luces-300led-impermeable-rgb-_JM?searchVariation=60507795012#searchVariation=60507795012&position=13&type=item&tracking_id=c465f05d-ed20-4d85-9f95-a407d5b3f492
SENSOR DE MOVIMIENTO CON ALARMA RADIOSHACK J11 (BLANCO). (2021, 21 enero). RadioShack México. https://www.radioshack.com.mx/store/radioshack/en/Categor%C3%ADa/Todas/Hogar/Seguridad/Alarmas-y-sensores/SENSOR-DE-MOVIMIENTO-CON-ALARMA-RADIOSHACK-J11-%28BLANCO%29/p/91000
Bombilla LED Inteligente WiFi Focos Inteligentes de Colores Lámpara Dimmable Luces inteligentes Smart Bulb 6500K 7W RGBW 650LM con Remoto Controlado por Amazon Echo Alexa Google Home IFTTT E26: Amazon.com.mx: Hogar y Cocina. (s. f.). Amazon. Recuperado 21 de enero de 2021, de https://www.amazon.com.mx/Inteligente-Inalambrica-Inteligentes-Smartphone-Smartphone-Energy/dp/B07PSMPKQQ
Muela, C. (2018, 21 junio). Google Home ya a la venta en España y lo hemos probado: qué puedes y qué no puedes hacer con él. Xataka. https://www.xataka.com/accesorios/google-home-venta-espana-que-puedes-que-no-puedes-hacer#:%7E:text=Google%20Home%20y%20Google%20Home%20Mini%3A%20caracter%C3%ADsticas%20y%20diferencias,o%20coral%20(el%20segundo).
Zorrilla, V. A. P. B. J. (2017, 19 mayo). Raspberry Pi – Conectando un sensor de temperatura y humedad DHT11 – Internet de las Cosas. internetdelascosas. https://www.internetdelascosas.cl/2017/05/19/raspberry-pi-conectando-un-sensor-de-temperatura-y-humedad-dht11/