Proyecto Final:

Smart Home

Fundamentos de Sistemas Embebidos

Autor: Francisco Javier Solano Tavera

1. Introducción

El siguiente trabajo representa el proyecto final de la asignatura de Fundamentos de Sistemas Embebidos, el cual corresponde al concentrador básico Smart Home.

El concentrador básico Smart Home, administra remotamente con un navegador web algunos dispositivos del hogar, por ejemplo, el desplegado de cámaras de vigilancia, el encendido y apagado de luces, el atenuado de luces, la detección de timbre, la apertura remota de la cochera y un programado de encendido y apagado de luces.

Es importante conocer algunos de los conceptos para entender el funcionamiento del proyecto, por lo que a continuación se presentan conceptos y antecedentes históricos.

Un sistema embebido es un sistema de cómputo de propósito específico y de características más restringidas que una computadora personal o supercomputadora. (Matamoros, 2021).

Según Valvano, 2003, las características básicas de los sistemas embebidos son las siguientes:

- Deben ser confiables,
- La confiabilidad, en inglés reliability R(t), es la probabilidad de que el
- sistema trabaje correctamente dado que está funcionando en t=0.
- La mantenibilidad, en inglés Maintainability M(d), es la probabilidad de
- que el sistema vuelva a trabajar correctamente d unidades de tiempo
- después de un fallo.
- La disponibilidad, en inglés Availability A(t), es la probabilidad de que el
- sistema esté funcionando en el tiempo t.
- La seguridad informática: consiste en disponer de una comunicación
- confidencial y autentificada.
- La creación de un sistema confiable debe ser considerada desde un
- comienzo, no como una consideración posterior.
- Deben ser eficientes en cuanto a la energía, al tamaño de código, al

- peso y al costo.
- Están dedicados a ciertas aplicaciones.
- Interfaces de usuario dedicadas (sin ratón, keyboard y pantalla)

La Raspberry Pi es un ordenador de bajo coste y tamaño reducido, tanto es así que cabe en la palma de la mano, pero puedes conectarle un televisor y un teclado para interactuar con ella exactamente igual que cualquier otra computadora. (Rodríguez, 2018).

la Raspberry Pi tiene la habilidad de interactuar con el mundo exterior, puede ser usada en una amplia variedad de proyectos digitales, desde reproductores de música y video, detectores de padres, estaciones meteorológicas hasta cajas de aves con cámaras infrarrojas. Queremos que veas que la Raspberry Pi puede ser usada por niños y adultos por todas partes del mundo, para aprender a programar y entender cómo funcionan las computadoras. (Ojeda, 2019).

La Raspberry Pi fue creada en febrero del 2012 por la Raspberry Pi Foundation, originalmente pensado para promover y enseñar las ciencias básicas de la computación en las escuelas y universidades de Reino Unido. Originalmente lanzaron dos modelos, el Modelo A y el Modelo B. (Ojeda, 2019).

En el 2012, después de su lanzamiento de la Raspberry Pi, se vendieron 500.000 unidades, y un mes después ya se había realizado la primera revisión “B” de la placa original. La primera unidad tenía 256 MB de ram y un procesador a 700 MHz, tenía el característico conector de 26 pines GPIO y salida de video por HDMI o RCA además de un conector de 3.5mm para el audio, el primer modelo carecía de puerto ethernet. (Delgado, 2020).

Objetivo

Implementar un concentrador que permita monitorear y administrar remotamente vía navegador los dispositivos inteligentes domésticos vinculados

2. Material

I. Plataforma

Equipo con máquina virtual con sistema operativo Linux

II. Intérprete de Python 3.5 instalado

3. Descripción del funcionamiento de componentes

Para el desarrollo del proyecto, su implementación fue por medio de la simulación, por lo que a continuación se presenta el circuito implementado en el simulador, en el que se pueden ver los pines con los que fue configurada la tarjeta. Esta configuración fue realizada por el profesor Matamoros, 2021.

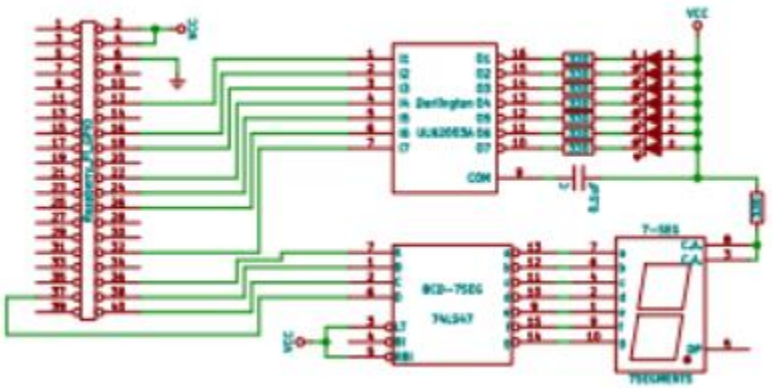


Imagen 1: Circuito implementado en el simulador

Según Velasco, 2020, la tarjeta de red de la Raspberry (aunque es Gigabit Ethernet la controla el controlador USB, por lo que su velocidad máxima es de 300 Mbps) para conectar nuestro Raspberry Pi por cable al router, así como los 4 puertos USB que aún siguen siendo USB 2.0. En el lado perpendicular nos encontraremos con el puerto HDMI, el puerto de salida de audio y vídeo compuesto y el conector micro-usb para conectarlo a la corriente. También menciona que los puertos CSI y DSI son para conectar una cámara y una pantalla fácilmente (marcados como display y camera), además del GPIO, uno de los pilares base de este micro-ordenador. Los 4 pines que nos encontramos detrás de los puertos USB sirven para configurar el PoE y, por último, en la parte inferior nos encontraremos con la ranura en la que va la micro-SD.

Entre sus componentes, se tiene un simple disipador (similar a un IHS de los procesadores de ordenador) que ayuda a mantenerla un poco más fresco. También, justo encima, se tiene una chapa con el logo de Raspberry Pi bajo la cual se esconde el nuevo chip de red de Adafruit, el que nos proporciona Wi-Fi de doble banda y Bluetooth 4.2. El chip LAN7515 se encargará de controlar tanto la tarjeta de red como los puertos USB del micro-ordenador. En la parte inferior encontraremos el chip Elpida B8132B4PB-8D-F que suministra 1 GB de RAM a este micro-ordenador. Por último, junto a la clavija micro-usb tendremos otro chip, el encargado de controlar la alimentación. (Velasco, 2020).

4. Descripción del funcionamiento de la tarjeta controladora

Según Ojeda 2019, la La Raspberry Pi cuenta con un GPIO de 40 pines, el cual permite el contacto con el mundo exterior, tanto por sensores como con actuadores, en este punto es importante conocer que el GPIO de Raspberry trabaja con un nivel de 3.3V, así que si quieres conectar sensores que operan a 5V necesitaras un conversor de niveles lógicos te recomendamos el MCI00582 comercializado por MCI Electronics. Debido que el procesador de la Raspberry Pi no tiene un conversor de analógico a digital integrado, por lo tanto si quieres leer sensores analógicos de usar un conversor ADC externo, en MCI Electronics puedes conseguir uno con el código MCI01856. Además cuentas con puertos de comunicación I2C, SPI y UART.

La siguiente imagen es una representación de la placa de la tarjeta:

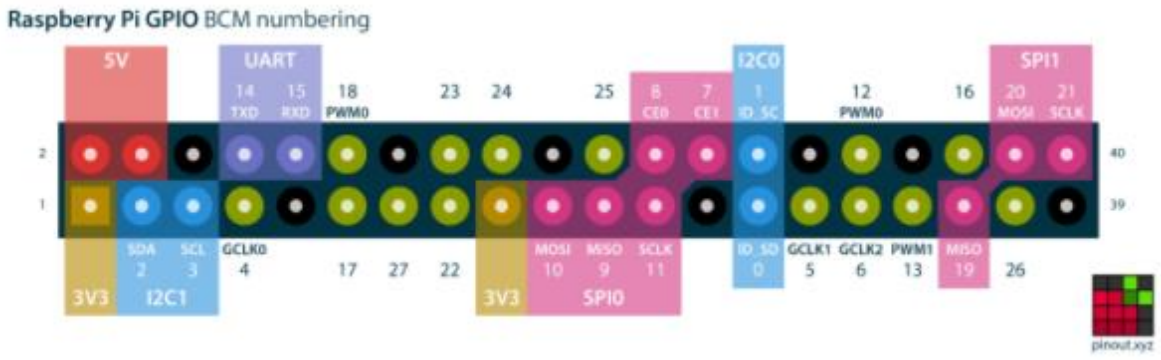


Imagen 2: Representación de la placa de la Raspberry Pi (Ojeda, 2019).

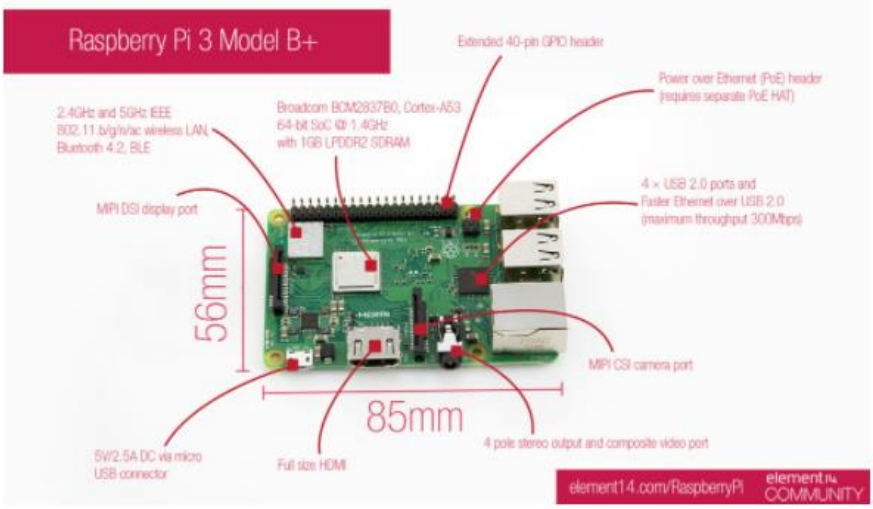


Imagen 3: Componentes de la Raspberry Pi (Ojeda, 2019).

5. Configuración de la tarjeta controladora

NOTA: Los siguientes pasos a configurar, siguen la secuencia del programa 2 del profesor José Mauricio Matamoros de María y Campos, UNAM.

Dirigirse al siguiente repositorio: https://github.com/Javier234/Proyecto-Final-Sistemas-Embebidos.git posteriormente, en una terminal de su equipo y en el lugar donde lo desee descargar, ejecute la siguiente línea de comandos:

git clone https://github.com/Javier234/Proyecto-Final-Sistemas-Embebidos.git

cd RPiVirtualBoard

Instalar las dependencias requeridas por el simulador con pip:

sudo apt install python3-tk

pip install --user -r requirements.txt

Probar el simulador con:

python3 ./webserver.py

Si todo es correcto, sale una ventana similar a la de la figura 1:

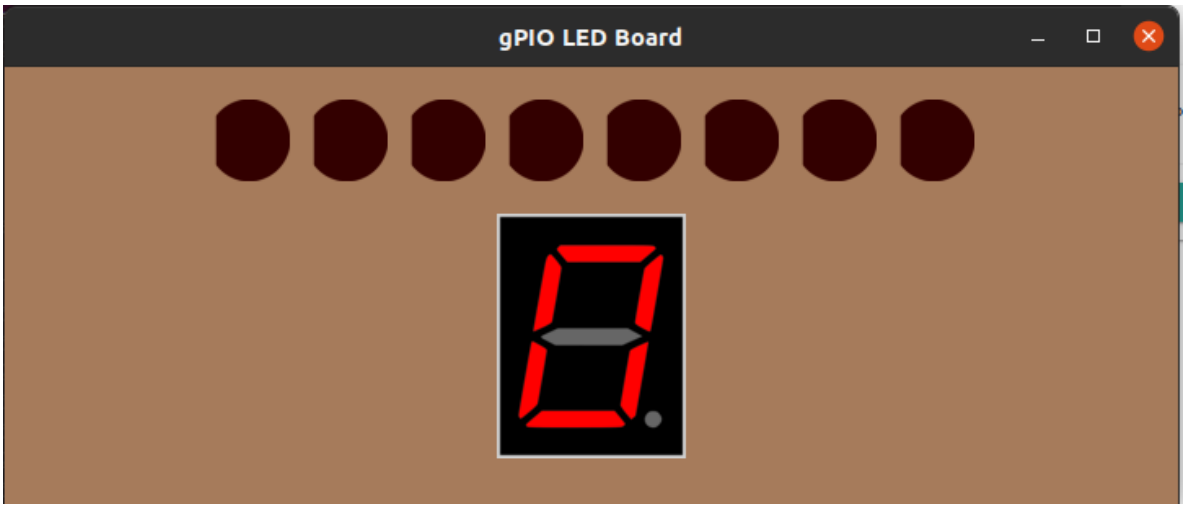


Figura 1. Simulador de la tarjeta con leds para la GPIO de la Raspberry Pi

6. Desarrollo de los componentes de software

A continuación, se muestra los módulos de código para las funciones que desempeñará la tarjeta.

El siguiente módulo de código muestra el encendido y apagado de luces. Se implementaron dos funciones, una para el encendido de luces y otra para el apagado de luces, para que, al momento de hacer esta acción vía remotamente en el navegador, se ejecuten las acciones por separado. Se utilizaron los 8 pines de la tarjeta simuladora.

```
123  def _casa_luces():
124      ledsoff()
125      pwm_off()
126
127  #while True: # Forever
128                      # Wait 500ms
129      GPIO.output(32, GPIO.HIGH) # Turn led on
130      GPIO.output(26, GPIO.HIGH)
131      GPIO.output(24, GPIO.HIGH)
132      GPIO.output(22, GPIO.HIGH)
133      GPIO.output(18, GPIO.HIGH)
134      GPIO.output(16, GPIO.HIGH)
135      GPIO.output(12, GPIO.HIGH)
136      GPIO.output(10, GPIO.HIGH)
137
138      print("Luces encendidas")
139
140      pass
```

```
143  """ APAGA LAS LUCES"""
144
145  def _casa_apagar_luces():
146      ledsoff()
147      pwm_off()
148
149      GPIO.output(32, GPIO.LOW)  # Turn led off
150      GPIO.output(26, GPIO.LOW)
151      GPIO.output(24, GPIO.LOW)
152      GPIO.output(22, GPIO.LOW)
153      GPIO.output(18, GPIO.LOW)
154      GPIO.output(16, GPIO.LOW)
155      GPIO.output(12, GPIO.LOW)
156      GPIO.output(10, GPIO.LOW)
157
158      print("Luces Apagadas")
```

Código 2: Encendido y Apagado de luces del archivo ledmanager.py

Para el desplegado de cámaras de vigilancia, se implementó el siguiente código que consta de dos funciones, una para activar la grabación de las cámaras y otra para detener la grabación. Para esta acción, se utilizó pwm en los pines 16, 18, 22 y 24 de la tarjeta simuladora.

```
    pwm6 = GPIO.PWM(24, 1)
    pwm5 = GPIO.PWM(22, 1)
    pwm4 = GPIO.PWM(18, 1)
    pwm3 = GPIO.PWM(16, 1)
```

```
def _casa_camaras():
    ledsoff()

    pwm6.start(50) #Se activa sexta camara
    pwm5.start(50) #Se activa quinta camara
    pwm4.start(50) #Se activa cuarta camara
    pwm3.start(50) #Se activa tercera camara
    print("Desplegado de camaras de vigilancia")

    pass

""" DESACTIVA CAMARAS DE VIGILANCIA"""
def _casa_camaras_apagadas():
    ledsoff()
    pwm_off()

    print("Apagando camaras de vigilancia")


    pass
```

Código 3: Desplegado de cámaras del archivo ledmanager.py

Para la detección del timbre de puerta, se implementó el código que se muestra a continuación. Se utilizó el pin 10 de la tarjera simuladora para que realizara la acción cuando se toque el timbre.

```
188  def _casa_timbre():
189      ledsoff()
190      pwm_off()
191      GPIO.output(10, GPIO.HIGH)
192      print("Timbre ON")
193      sleep(2)
194
195      GPIO.output(10, GPIO.LOW)
196      print("Timbre OFF")
197      sleep(1)
```

Código 4: Detección de timbre del archivo ledmanager.py

En la apertura remota de la puerta de la cochera, el código utilizado es el que se muestra en seguida. Se utilizaron los 8 pines de la tarjeta simuladora y se implementaron dos funciones, una para abrir la cochera y otra para cerrarla. El efecto de los leds funciona en cuando se está abriendo la cochera, hace una marquesina a la derecha, y cuando cierra la puerta de la cochera, hace una marquesina a la izquierda.

```
268 def _casa_abrir_cochera():
269     pwm_off()
270     ledsoff()
271 
272     sleep(0.5)                  # Wait 500ms
273     GPIO.output(10, GPIO.HIGH) # Turn led on
274     sleep(0.5)
275     GPIO.output(10, GPIO.LOW)  # Turn led off
276     GPIO.output(12, GPIO.HIGH) # Turn led on
277     sleep(0.5)
278     GPIO.output(12, GPIO.LOW)  # Turn led off
279     GPIO.output(16, GPIO.HIGH) # Turn led on
280     sleep(0.5)                  # Espera 500ms
281     GPIO.output(16, GPIO.LOW)  # Turn led off
282     GPIO.output(18, GPIO.HIGH) # Turn led on
283     sleep(0.5)                  # Espera 500ms
284     GPIO.output(18, GPIO.LOW)  # Turn led off
285     GPIO.output(22, GPIO.HIGH) # Turn led on
286     sleep(0.5)                  # Espera 500ms
287     GPIO.output(22, GPIO.LOW)  # Turn led off
288     GPIO.output(24, GPIO.HIGH) # Turn led on
289     sleep(0.5)                  # Espera 500ms
290     GPIO.output(24, GPIO.LOW)  # Turn led off
291     GPIO.output(26, GPIO.HIGH) # Turn led on
292     sleep(0.5)                  # Espera 500ms
293     GPIO.output(26, GPIO.LOW)  # Turn led off
294     GPIO.output(32, GPIO.HIGH) # Turn led on
295     sleep(0.5)                  # Espera 500ms
296     GPIO.output(32, GPIO.LOW)  # Turn led off
297     print("Abriendo cochera%")
298 
299     pass
```

```
302  def _casa_cerrar_cochera():
303
304      sleep(0.5)                  # Wait 500ms
305      GPIO.output(32, GPIO.HIGH) # Turn led on
306      sleep(0.5)
307      GPIO.output(32, GPIO.LOW)  # Turn led off
308      GPIO.output(26, GPIO.HIGH) # Turn led on
309      sleep(0.5)
310      GPIO.output(26, GPIO.LOW)  # Turn led off
311      GPIO.output(24, GPIO.HIGH) # Turn led on
312      sleep(0.5)                  # Espera 500ms
313      GPIO.output(24, GPIO.LOW)  # Turn led off
314      GPIO.output(22, GPIO.HIGH) # Turn led on
315      sleep(0.5)                  # Espera 500ms
316      GPIO.output(22, GPIO.LOW)  # Turn led off
317      GPIO.output(18, GPIO.HIGH) # Turn led on
318      sleep(0.5)                  # Espera 500ms
319      GPIO.output(18, GPIO.LOW)  # Turn led off
320      GPIO.output(16, GPIO.HIGH) # Turn led on
321      sleep(0.5)                  # Espera 500ms
322      GPIO.output(16, GPIO.LOW)  # Turn led off
323      GPIO.output(12, GPIO.HIGH) # Turn led on
324      sleep(0.5)                  # Espera 500ms
325      GPIO.output(12, GPIO.LOW)  # Turn led off
326      GPIO.output(10, GPIO.HIGH) # Turn led on
327      sleep(0.5)                  # Espera 500ms
328      GPIO.output(10, GPIO.LOW)  # Turn led off'''
329      print("Cerrando cochera%")
330      pass
331
```

Código 5: Apertura y cerrado de cochera del archivo ledmanager.py

En el atenuado de luces, se implementaron cuatro funciones, las cuales cada una hace una acción, configurando atenuación de las luces a un 75%, un 50%, a 25% y 10% por medio de pwm y el uso de todos los pines, el código es el siguiente.

```
48    pwm8 = GPIO.PWM(32, 1)
49    pwm7 = GPIO.PWM(26, 1)
50    pwm6 = GPIO.PWM(24, 1)
51    pwm5 = GPIO.PWM(22, 1)
52    pwm4 = GPIO.PWM(18, 1)
53    pwm3 = GPIO.PWM(16, 1)
54    pwm2 = GPIO.PWM(12, 1)
55    pwm1 = GPIO.PWM(10, 1)
```

```
204  def _casa_atenuar_75():
205      pwm_off()
206      ledsoff()
207      pwm8.start(75)
208      pwm7.start(75)
209      pwm6.start(75)
210      pwm5.start(75)
211      pwm4.start(75)
212      pwm3.start(75)
213      pwm2.start(75)
214      pwm1.start(75)
215      print("Atenuendo focos al 75%")
216
217      pass
219  def _casa_atenuar_50():
220      pwm_off()
221      ledsoff()
222      pwm8.start(50)
223      pwm7.start(50)
224      pwm6.start(50)
225      pwm5.start(50)
226      pwm4.start(50)
227      pwm3.start(50)
228      pwm2.start(50)
229      pwm1.start(50)
230      print("Atenuendo focos al 50%")
231
232      pass
```

```
234   def _casa_atenuar_25():
235       pwm_off()
236       ledsoff()
237       pwm8.start(25)
238       pwm7.start(25)
239       pwm6.start(25)
240       pwm5.start(25)
241       pwm4.start(25)
242       pwm3.start(25)
243       pwm2.start(25)
244       pwm1.start(25)
245       print("Atenuendo focos al 25%")
246
247       pass
```

```
249   def _casa_atenuar_10():
250       pwm_off()
251       ledsoff()
252       pwm8.start(10)
253       pwm7.start(10)
254       pwm6.start(10)
255       pwm5.start(10)
256       pwm4.start(10)
257       pwm3.start(10)
258       pwm2.start(10)
259       pwm1.start(10)
260       print("Atenuendo focos al 10%")
261
262       pass
```

Código 6: Atenuado de luces del archivo ledmanager.py

El siguiente código muestra el programado de luces, en el que se implementaron cuatro funciones, dos para para el programado del encendido de luces, y dos para el programado de apagado de luces, implementando tiempos de sleep para que cada cierto tiempo prenda o apague las luces, simulando 6 y 12 segundos para el encendido de luces y también 6 y 12 segundos para el apagado de luces. Se utilizaron los 8 pines de la tarjeta simuladora.

```
338  def _casa_programar_encendido_luces6():
339      ledsoff()
340      pwm_off()
341
342      sleep(6)                  # Wait 500ms
343      GPIO.output(32, GPIO.HIGH)  # Turn led off
344      GPIO.output(26, GPIO.HIGH)
345      GPIO.output(24, GPIO.HIGH)
346      GPIO.output(22, GPIO.HIGH)
347      GPIO.output(18, GPIO.HIGH)
348      GPIO.output(16, GPIO.HIGH)
349      GPIO.output(12, GPIO.HIGH)
350      GPIO.output(10, GPIO.HIGH)
351      print("Encendiendo luces en 6 segundos%")
352      pass
354  def _casa_programar_encendido_luces12():
355      ledsoff()
356      pwm_off()
357
358      sleep(12)                  # Wait 500ms
359      GPIO.output(32, GPIO.HIGH)  # Turn led off
360      GPIO.output(26, GPIO.HIGH)
361      GPIO.output(24, GPIO.HIGH)
362      GPIO.output(22, GPIO.HIGH)
363      GPIO.output(18, GPIO.HIGH)
364      GPIO.output(16, GPIO.HIGH)
365      GPIO.output(12, GPIO.HIGH)
366      GPIO.output(10, GPIO.HIGH)
367      print("Encendiendo luces en 12 segundos%")
368      pass
```

```
371  def _casa_programar_apagado_luces6():
372
373      pwm_off()
374
375      sleep(6)                  # Wait 500ms
376      GPIO.output(32, GPIO.LOW)  # Turn led off
377      GPIO.output(26, GPIO.LOW)
378      GPIO.output(24, GPIO.LOW)
379      GPIO.output(22, GPIO.LOW)
380      GPIO.output(18, GPIO.LOW)
381      GPIO.output(16, GPIO.LOW)
382      GPIO.output(12, GPIO.LOW)
383      GPIO.output(10, GPIO.LOW)
384      print("Apagando luces en 6 segundos%")
385      pass
```

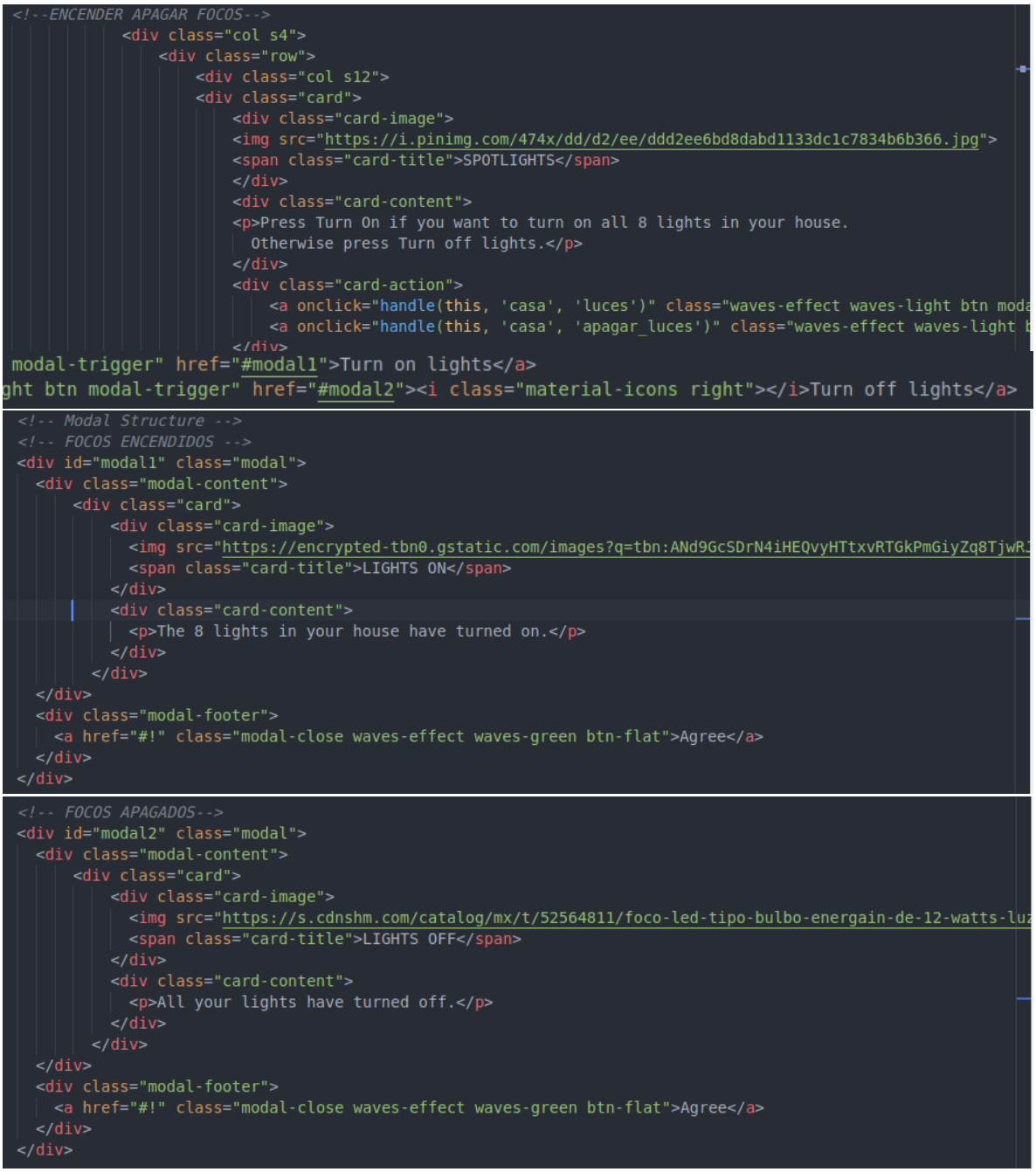
```
387  def _casa_programar_apagado_luces12():
388
389      pwm_off()
390
391      sleep(12)                  # Wait 500ms
392      GPIO.output(32, GPIO.LOW)  # Turn led off
393      GPIO.output(26, GPIO.LOW)
394      GPIO.output(24, GPIO.LOW)
395      GPIO.output(22, GPIO.LOW)
396      GPIO.output(18, GPIO.LOW)
397      GPIO.output(16, GPIO.LOW)
398      GPIO.output(12, GPIO.LOW)
399      GPIO.output(10, GPIO.LOW)
400      print("Apagando luces en 12 segundos%")
401      pass
```

Código 6: Programado de luces del archivo ledmanager.py

7. Integración de los componentes de software

Los códigos mostrados en la sección anterior (6), fueron enlazados con el archivo home.html, para que desde un navegador se administre remotamente los dispositivos domésticos vinculados.

Encendido y apagado de luces:



```
<!--ENCENDER APAGAR FOCOS-->
            <div class="col s4">
                <div class="row">
                    <div class="col s12">
                    <div class="card">
                        <div class="card-image">
                        <img src="https://i.pinimg.com/474x/dd/d2/ee/ddd2ee6bd8dabd1133dc1c7834b6b366.jpg">
                        <span class="card-title">SPOTLIGHTS</span>
                        </div>
                        <div class="card-content">
                        <p>Press Turn On if you want to turn on all 8 lights in your house.
                          Otherwise press Turn off lights.</p>
                        </div>
                        <div class="card-action">
                            <a onclick="handle(this, 'casa', 'luces')" class="waves-effect waves-light btn moda
                            <a onclick="handle(this, 'casa', 'apagar_luces')" class="waves-effect waves-light b
                        </div>
 modal-trigger" href="#modal1">Turn on lights</a>
ght btn modal-trigger" href="#modal2"><i class="material-icons right"></i>Turn off lights</a>
```

```
 <!-- Modal Structure -->
 <!-- FOCOS ENCENDIDOS -->
 <div id="modal1" class="modal">
   <div class="modal-content">
       <div class="card">
           <div class="card-image">
             <img src="https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcSDrN4iHEQvyHTtxvRTGkPmGiyZq8TjwRJ
             <span class="card-title">LIGHTS ON</span>
           </div>
           <div class="card-content">
             <p>The 8 lights in your house have turned on.</p>
           </div>
         </div>
   </div>
   <div class="modal-footer">
     <a href="#!" class="modal-close waves-effect waves-green btn-flat">Agree</a>
   </div>
 </div>
```

```
 <!-- FOCOS APAGADOS-->
 <div id="modal2" class="modal">
   <div class="modal-content">
       <div class="card">
           <div class="card-image">
             <img src="https://s.cdnshm.com/catalog/mx/t/52564811/foco-led-tipo-bulbo-energain-de-12-watts-luz
             <span class="card-title">LIGHTS OFF</span>
           </div>
           <div class="card-content">
             <p>All your lights have turned off.</p>
           </div>
         </div>
   </div>
   <div class="modal-footer">
     <a href="#!" class="modal-close waves-effect waves-green btn-flat">Agree</a>
   </div>
 </div>
```

Código 7: Encender apagar focos del archivo home.html

Código de las cámaras de vigilancia.

```
<!--CAMARAS DE VIGILANCIA-->
            <div class="col s4">
              <div class="row">
                  <div class="col s12">
                  <div class="card">
                      <div class="card-image">
                      <img src="https://lh3.googleusercontent.com/proxy/lYztjyEif1YC5B0un0oal0Xb02h0gYTrSoxwFSA
                      <span class="card-title">CAMERAS</span>
                      </div>
                      <div class="CAMARAS">
                      <p> Do you suspect danger around your house? Press turn on cameras.</p>
                      </div>
                      <div class="card-action">
                          <a onclick="handle(this, 'casa', 'camaras')" class="waves-effect waves-light btn moda
                          <a onclick="handle(this, 'casa', 'camaras_apagadas')" class="waves-effect waves-light
                      </div>
 modal-trigger" href="#modal3">Turn on cameras</a>
light btn modal-trigger" href="#modal4"><i class="material-icons right"></i>Turn off cameras</a>
```

```
<!-- CAMARAS ENCENDIDAS-->
  <div id="modal3" class="modal">
    <div class="modal-content">
        <div class="card">
            <div class="card-image">
              <img src="https://ak.picdn.net/shutterstock/videos/1035633218/thumb/1.jpg?ip=x480">
              <span class="card-title">CAMERAS ON</span>
            </div>
            <div class="card-content">
              <p>At this moment the security cameras have been activated.</p>
            </div>
          </div>
    </div>
    <div class="modal-footer">
      <a href="#!" class="modal-close waves-effect waves-green btn-flat">Agree</a>
    </div>
  </div>
```

```
 <!-- CAMARAS APAGADAS-->
 <div id="modal4" class="modal">
   <div class="modal-content">
       <div class="card">
           <div class="card-image">
             <img src="https://ak.picdn.net/shutterstock/videos/1007106691/thumb/1.jpg">
             <span class="card-title">CAMERAS OFF</span>
           </div>
           <div class="card-content">
             <p>Your cameras have been deactivated.</p>
           </div>
         </div>
   </div>
   <div class="modal-footer">
     <a href="#!" class="modal-close waves-effect waves-green btn-flat">Agree</a>
   </div>
 </div>
```

Código 8: Cámaras de vigilancia del archivo home.html

Para el atenuado de focos, se implementó el siguiente código en html.

```
<!--ATENUAR FOCOS-->
<div class="col s4">
  <div class="row">
    <div class="col s12">
    <div class="card">
      <div class="card-image">
      <img src="https://m.media-amazon.com/images/I/61yHbemiOfL._CR0,204,1224,1224_UX256.jpg">
      <span class="card-title"> ATTENUATE LIGHTS</span>
      </div>
      <div class="">
      <p>If the light bothers you, you can dim your lights as you want.</p>
      </div>
      <div class="card-action">
        <a onclick="handle(this, 'casa', 'atenuar_75')" class="waves-effect waves-light btn mod
        <a onclick="handle(this, 'casa', 'atenuar_50')" class="waves-effect waves-light btn mod
        <a onclick="handle(this, 'casa', 'atenuar_25')" class="waves-effect waves-light btn mod
        <a onclick="handle(this, 'casa', 'atenuar_10')" class="waves-effect waves-light btn mod
      </div>
modal-trigger" href="#modal5">Attenuate 75%</a>
modal-trigger" href="#modal6"><i class="material-icons right"></i>Attenuate 50%</a>
modal-trigger" href="#modal6"><i class="material-icons right"></i>Attenuate 25%</a>
modal-trigger" href="#modal6"><i class="material-icons right"></i>Attenuate 10%</a>
```

Código 9: Atenuado de luces del archivo home.html

El siguiente código representa el timbre

```
<!--TIMBRE-->
<div class="col s4">
  <div class="row">
    <div class="col s12">
    <div class="card">
      <div class="card-image">
      <img src="https://m.media-amazon.com/images/I/61qhg110-AL._CR0,521,986,986_UX256.jpg">
      <span class="card-title">DOORBELL</span>
      </div>
      <div class="card-content">
      <p>If you want to enter the house, ring the bell.</p>
      </div>
      <div class="card-action">
        <a onclick="handle(this, 'casa', 'timbre')" class="waves-effect waves-light btn moda
      </div>
modal-trigger" href="#modal7">Ring the doorbell</a>
```

```
<!-- TIMBRE -->
<div id="modal7" class="modal">
  <div class="modal-content">
    <div class="card">
      <div class="card-image">
        <img src="https://www.mndelgolfo.com/blog/wp-content/uploads/2017/09/elegir-timbre-para-puerta-ca
        <span class="card-title">THE BELL IS RINGING</span>
      </div>
      <div class="card-content">
        <p>You rang the bell.</p>
      </div>
    </div>
  </div>
  <div class="modal-footer">
    <a href="#!" class="modal-close waves-effect waves-green btn-flat">Agree</a>
  </div>
</div>
```

Código 10: Detección de Timbre del archivo home.html

Abertura de cochera:

```
<!--ABRIR CERRAR COCHERA-->
            <div class="col s4">
              <div class="row">
                  <div class="col s12">
                  <div class="card">
                      <div class="card-image">
                      <img src="https://i.pinimg.com/474x/a1/82/6d/a1826d39f5668d0b929f2cfabc98024b.jpg">
                      <span class="card-title">GARAGE</span>
                      </div>
                      <div class="card-content">
                      <p>Press open garage in case you want to put your car in or out.</p>
                      </div>
                      <div class="card-action">
                          <a onclick="handle(this, 'casa', 'abrir_cochera')" class="waves-effect waves-light btn
                          <a onclick="handle(this, 'casa', 'cerrar_cochera')" class="waves-effect waves-light bt
                      </div>
modal-trigger" href="#modal9">Open garage</a>
 modal-trigger" href="#modal10"><i class="material-icons right"></i>Close garage</a>
```

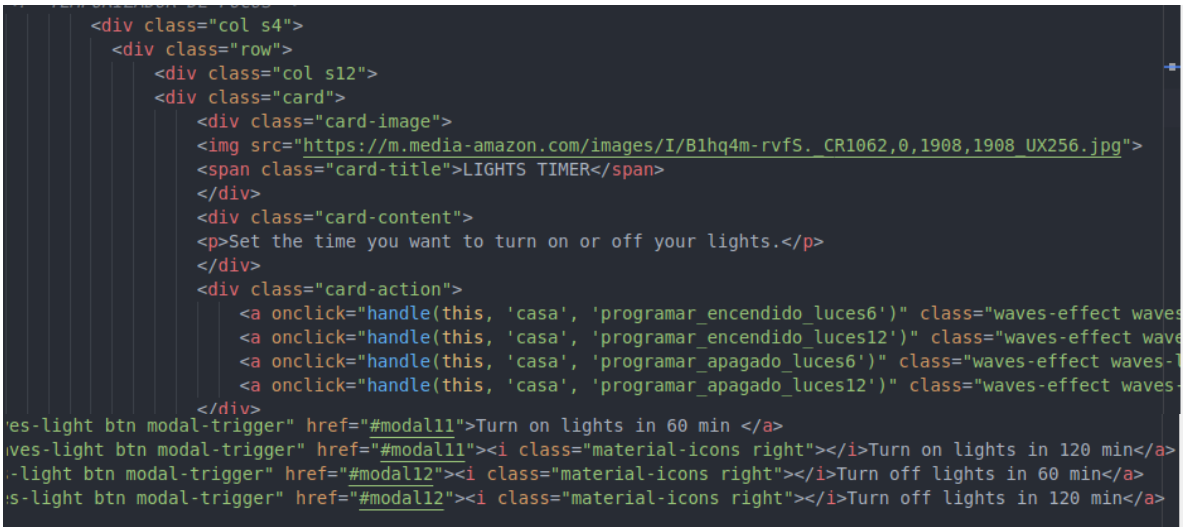

```
modal-trigger" href="#modal9">Open garage</a>
 modal-trigger" href="#modal10"><i class="material-icons right"></i>Close garage</a>
```


```
<!-- ABRIR COCHERA -->
<div id="modal9" class="modal">
  <div class="modal-content">
      <div class="card">
          <div class="card-image">
            <img src="https://casaydiseno.com/wp-content/uploads/2016/09/organizacion-garaje-consejos-espacio
            <span class="card-title">OPEN GARAGE</span>
          </div>
          <div class="card-content">
            <p>The garage is opening.</p>
          </div>
        </div>
  </div>
  <div class="modal-footer">
    <a href="#!" class="modal-close waves-effect waves-green btn-flat">Agree</a>
  </div>
</div>
```

```
<!-- CERRAR COCHERA -->
<div id="modal10" class="modal">
  <div class="modal-content">
      <div class="card">
          <div class="card-image">
            <img src="https://http2.mlstatic.com/D_NQ_NP_855516-MLM27311565295_052018-W.jpg">
            <span class="card-title">CLOSED GARAGE</span>
          </div>
          <div class="card-content">
            <p>The garage is closing.</p>
          </div>
        </div>
  </div>
  <div class="modal-footer">
    <a href="#!" class="modal-close waves-effect waves-green btn-flat">Agree</a>
  </div>
</div>
```

Código 11: Abertura y cerrado de cochera del archivo home.html

Para el programado de prendido y apagado de luces:



Código 12: Programado de luces del archivo home.html

El webserver.py fue el encargado de hacer la integración de los componentes de la página web para poder controlar vía remota en un navegador.

Para realizar la acción y ejecución del proyecto, en el servidor web se pone la IP de la máquina del usuario para que funcione como punto de acceso.

Para conocer la IP, ejecutar el siguiente comando

ip -4 address|grep inet

Una vez colocada esta IP en el archivo webserver.py en la línea 30, se prosigue a ejecutar la tarjeta simuladora con el siguiente comando:

python3 ./webserver.py

Al realizar la acción anterior, mostrará la tarjeta simuladora y terminal.

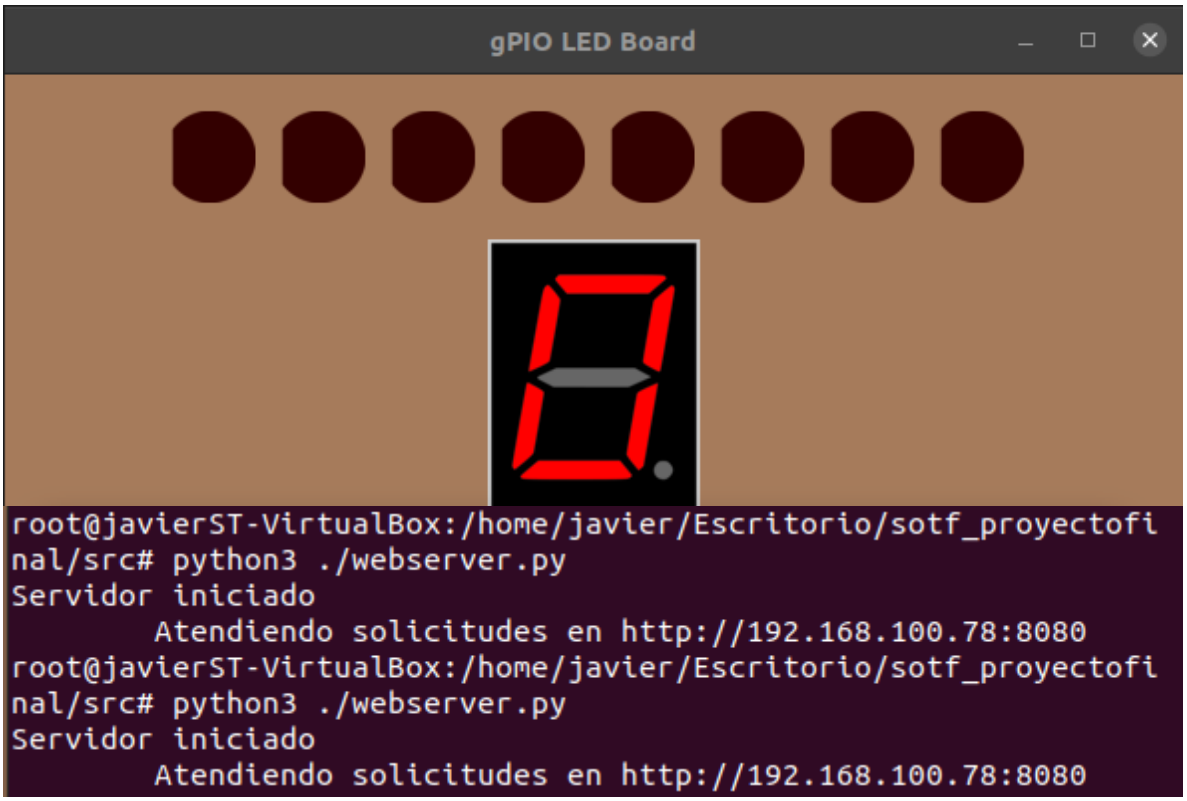


Imagen 4: Tarjeta Simuladora y terminal

En un navegador, escribiremos la IP y el puerto, como se muestra en la siguiente imagen:

Imagen 5: IP y puerto en navegador web

Estando en el navegador, empezaremos a administrar remotamente los dispositivos domésticos vinculados de la casa inteligente.

Botones para el encendido y apagado de luces:

Imagen 6: Encendido y apagado de luces

Para realizar la acción para encender los focos, se presiona Turn on lights, en la que prende los pines de la tarjeta simuladora y manda un mensaje de que la acción fue realizada.

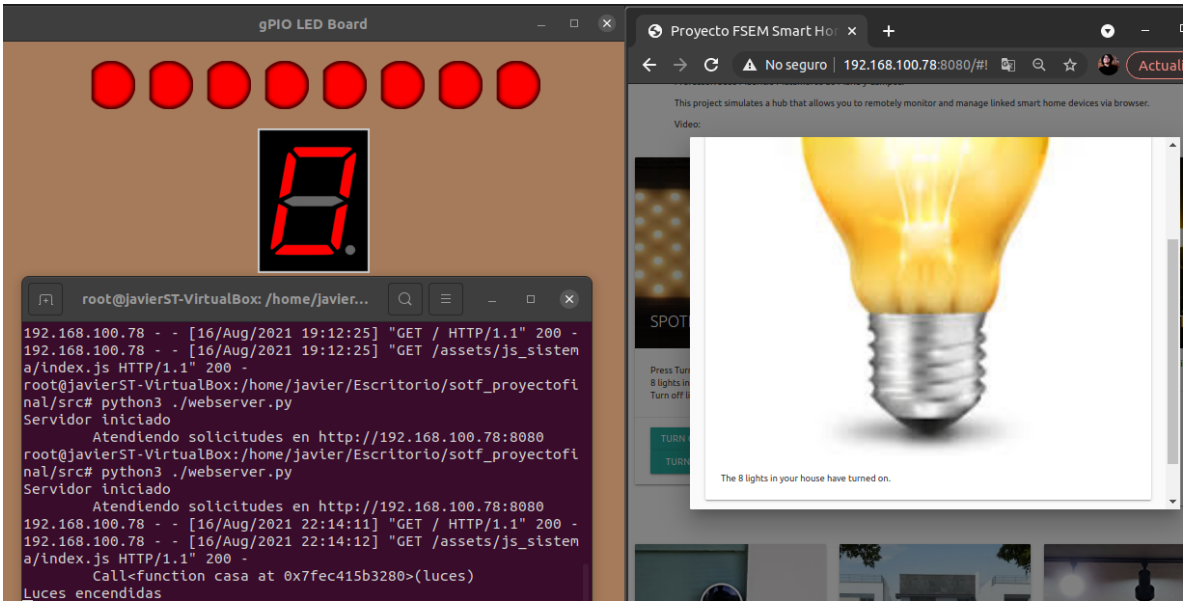


Imagen 7: botón Turn on lights encendido de luces

Para el apagado de luces se presiona Turn off lights en la que apagará los leds de la tarjeta.

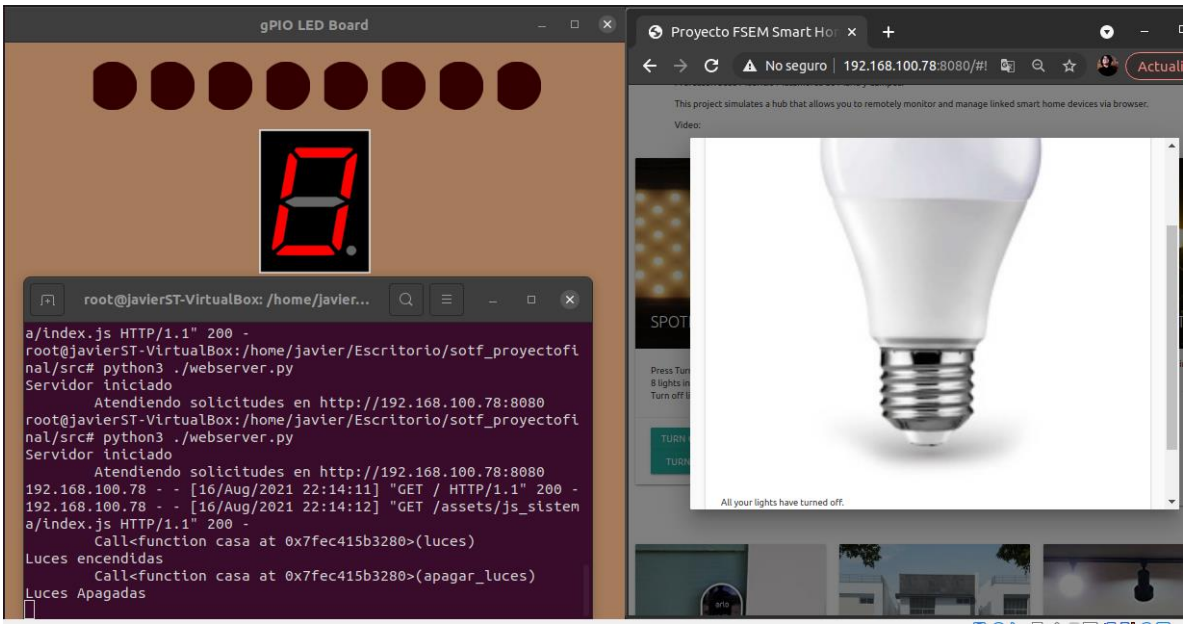


Imagen 8: Botón Turn off light para el apagado de luces

Botones para el desplegado de cámaras.

Imagen 9: Botones desplegado de cámaras

Para activar el grabado de cámaras, se presiona TURN ON CAMERAS. Para simular esta acción, los pines 16, 18, 22 y 24 están parpadeando, simulando que está grabando, al igual del desplegado del mensaje.

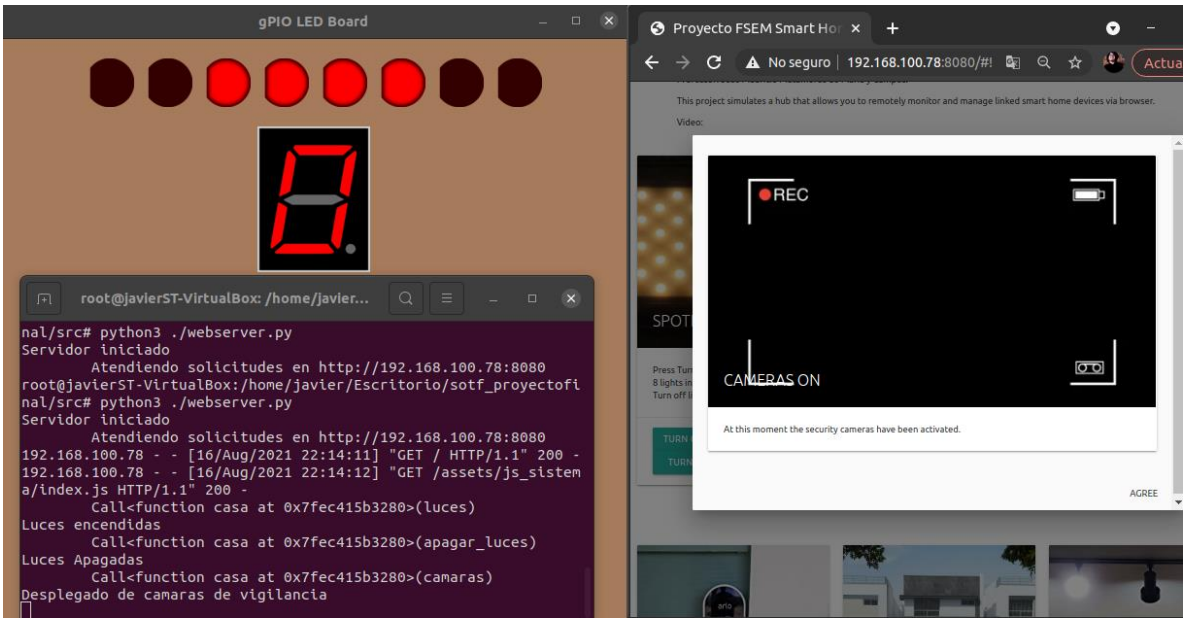


Imagen 10: Botón TURN ON CAMERAS para la activación de cámaras

Para el apagado de las cámaras, se presiona el botón TURN OFF CAMERAS, en la que simula apagar los pines.

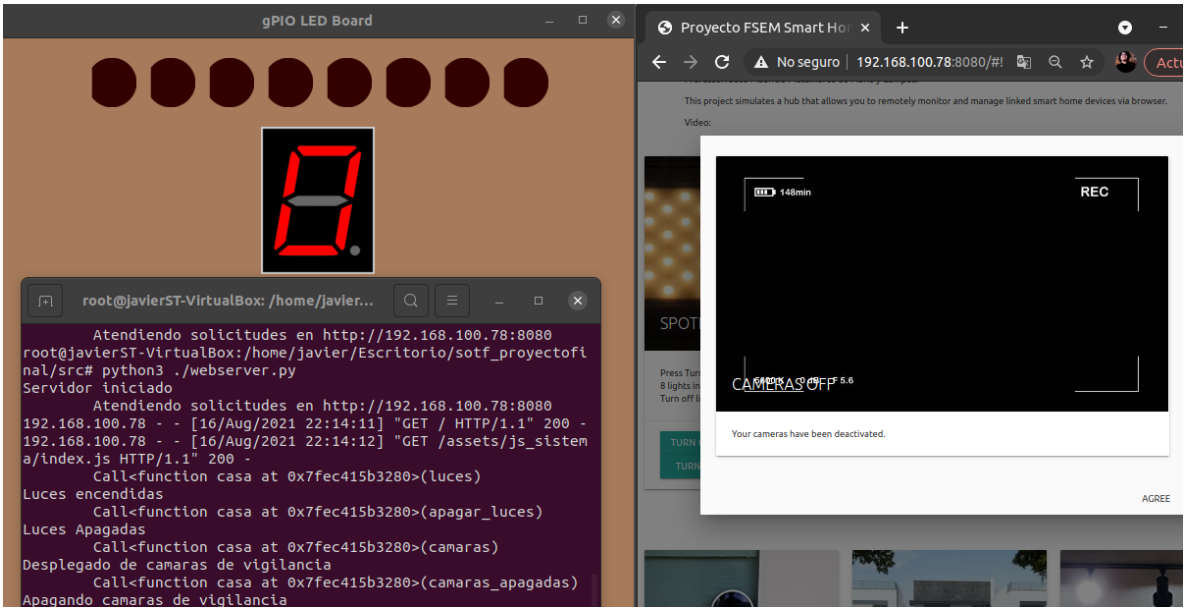


Imagen 11: Botón TURN OFF CAMERAS para el apagado de las cámaras.

Botones para el Atenuado.

Imagen 12: Botones de Atenuado de luces

Para realizar las atenuaciones, existen cuatro botones en los que cada uno corresponde a la atenuación que se desea para el hogar, por ejemplo, a 75%, 50%, 25% y 10%. En las imágenes siguientes se muestra la ejecución de los botones.

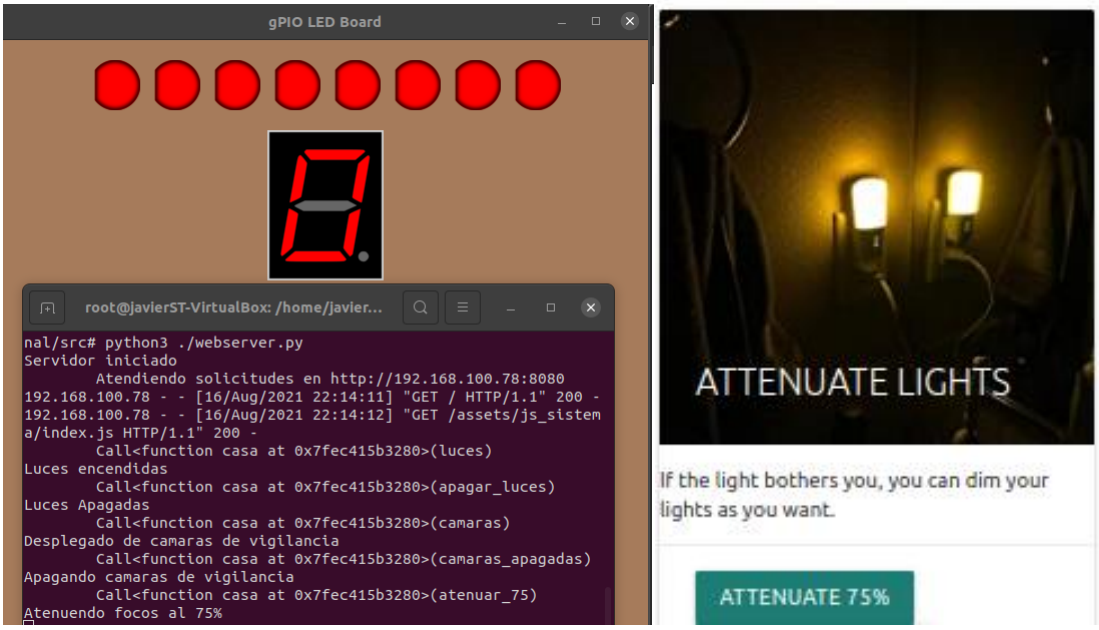


Imagen 13: Botón de Atenuado de luces a 75%

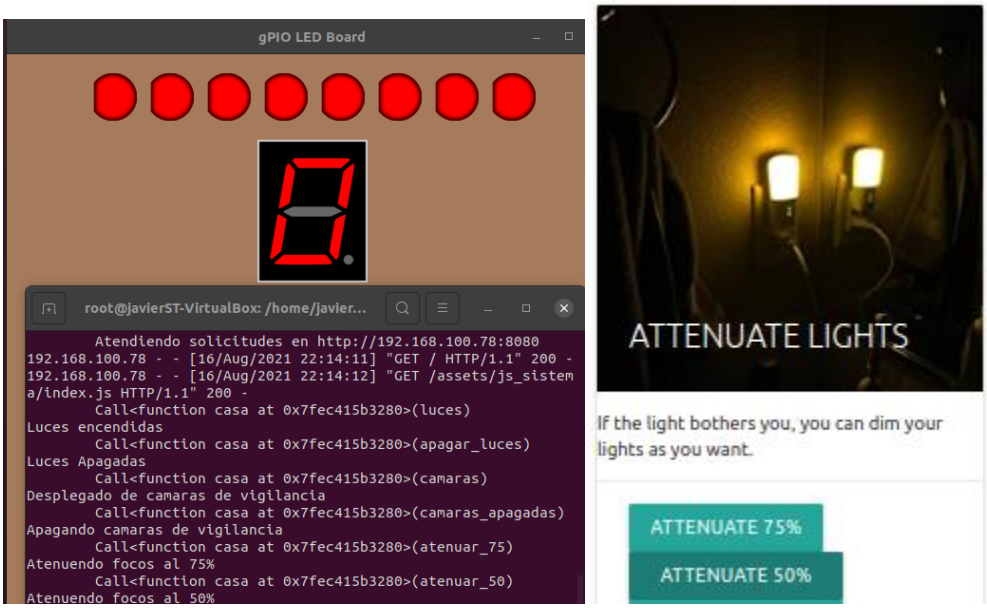


Imagen 14: Botón de Atenuado de luces a 50%

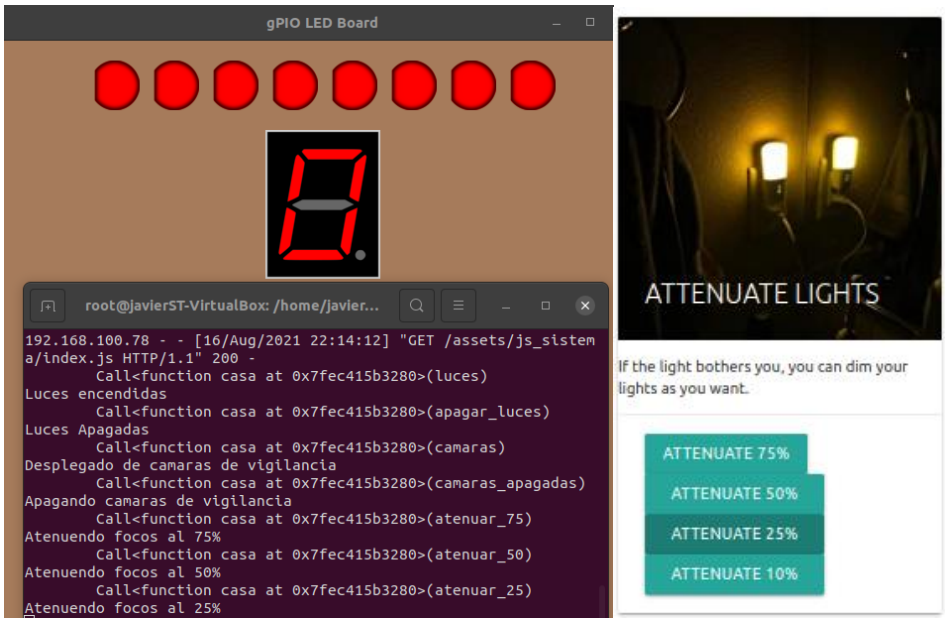


Imagen 15: Botón de Atenuado de luces a 25%

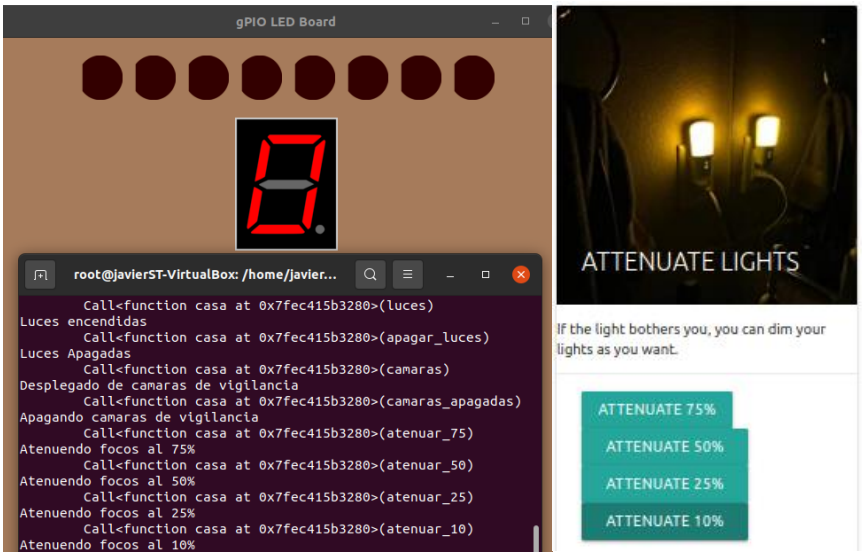


Imagen 16: Botón de Atenuado de luces a 10%

En la implementación del botón del timbre, prende un pin en el que índica que el timbre fue tocado, posteriormente se apaga, simulando que por cada vez que se toque el timbre de la puerta, este produce una acción, con el led prendido y la impresión en consola.

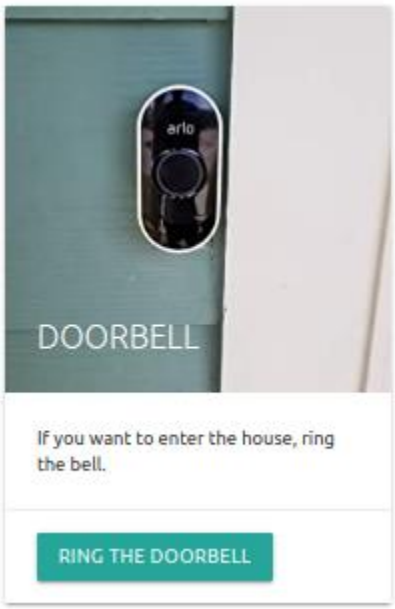


Imagen 17: Botón del timbre

Presionando el botón RING THE DOORBELL.

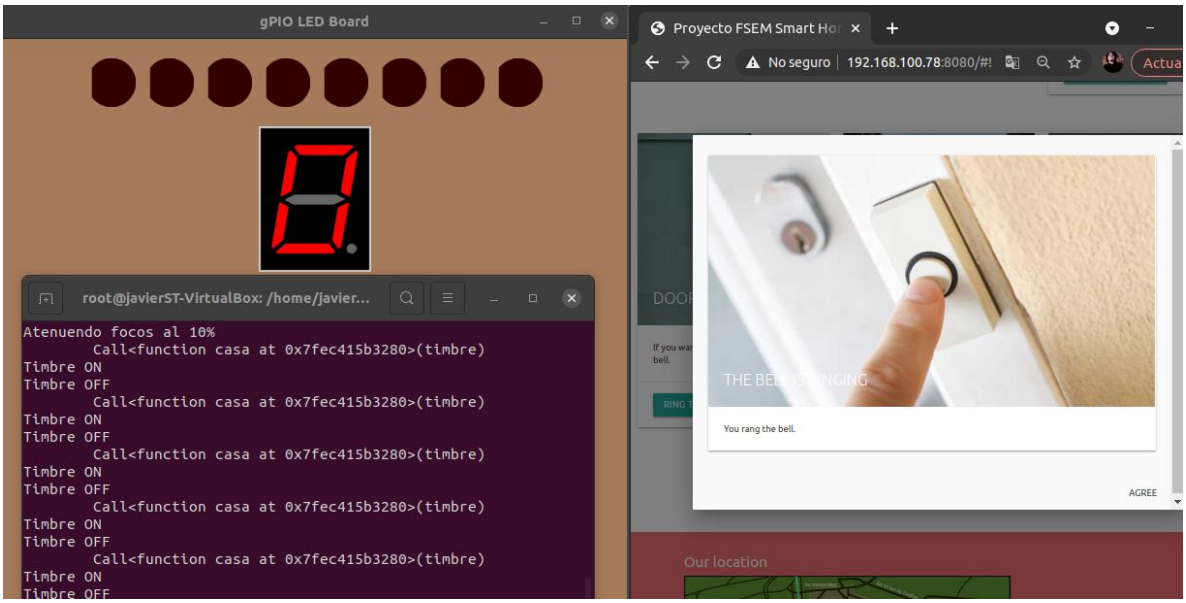


Imagen 18: Botón RING THE DOORBELL para el tocado del timbre

Para la abertura de la cochera, se tienen los siguientes botones, que es para abrirla y cerrarla.

Imagen 19: Botones para la abrir y cerrar la cochera

Presionando el botón OPEN GAREGE

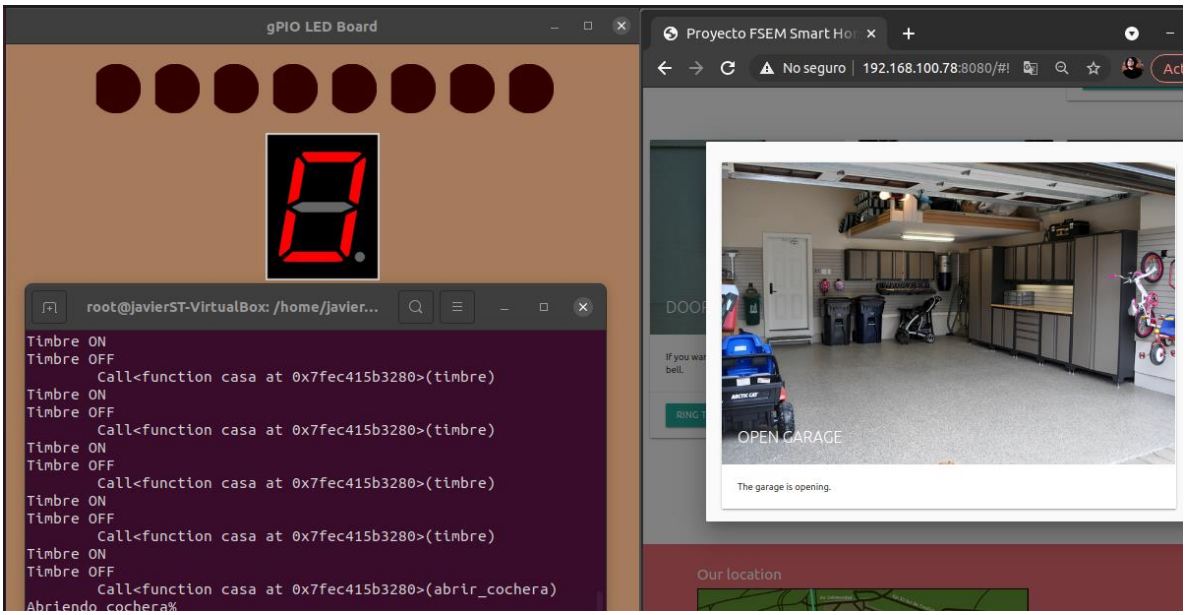


Imagen 20: Botón OPEN GARAGE

Para el cerado de la cochera:

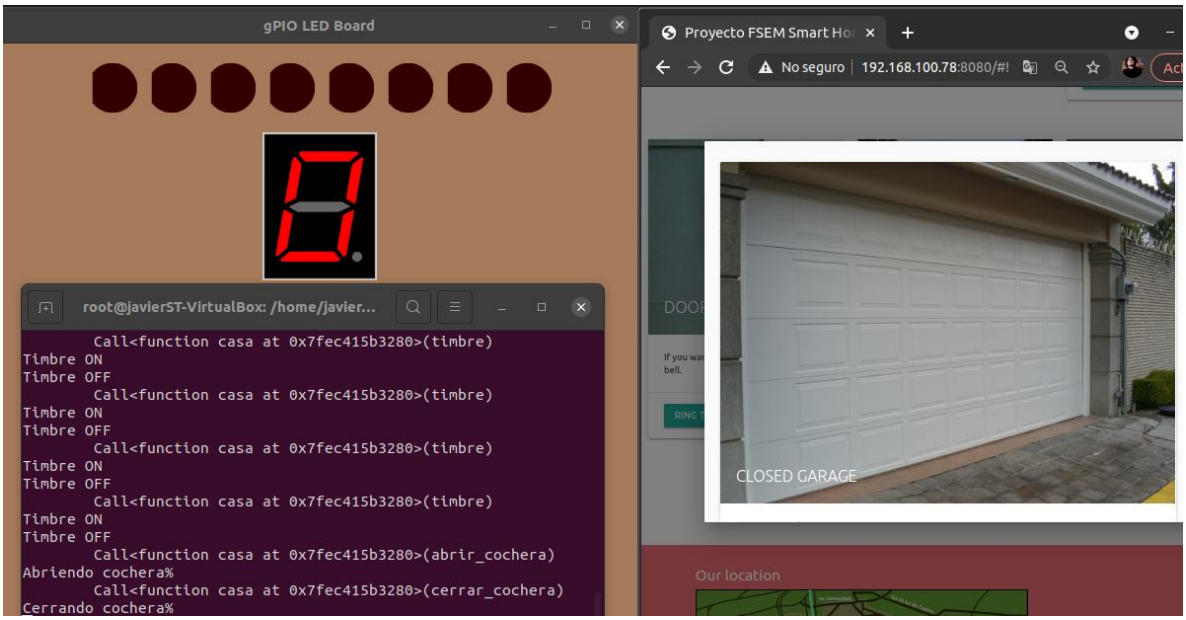


Imagen 21: Botón CLOSE GARAGE

Para el programado de encendido y apagado de luces, se implementaron los siguientes botones, cada uno con una programación, a 60 minutos y 120 minutos para el encendido y apagado (en simulación son segundos)

Imagen 22: Botones de programado de luces

A continuación, se muestra la acción de cada uno de estos botones.

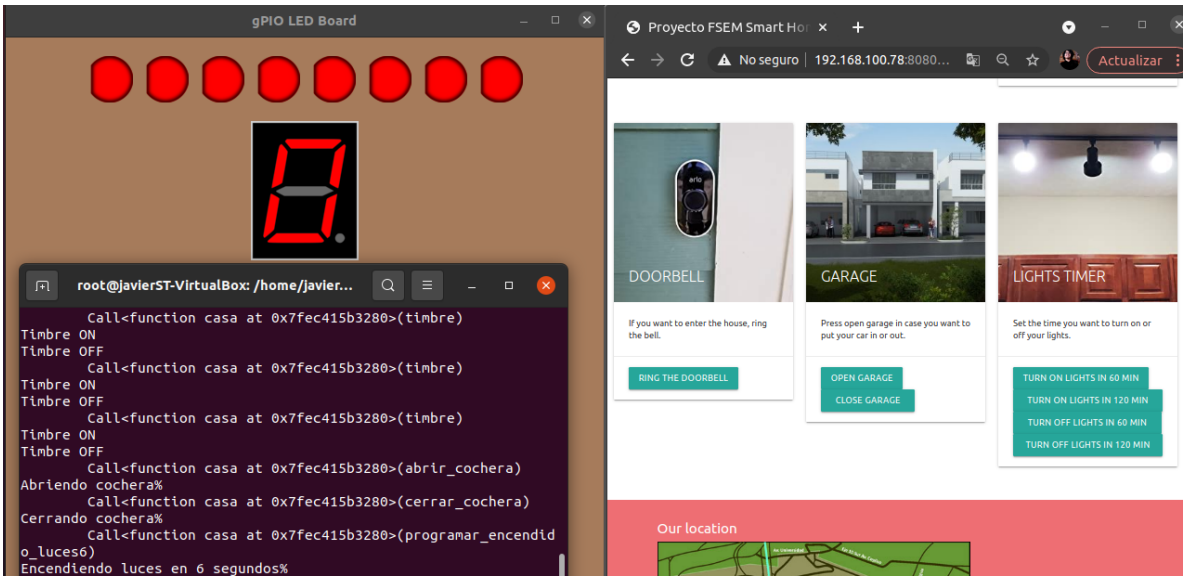


Imagen 23: Botón TURN ON LIGTHS IN 60 MIN

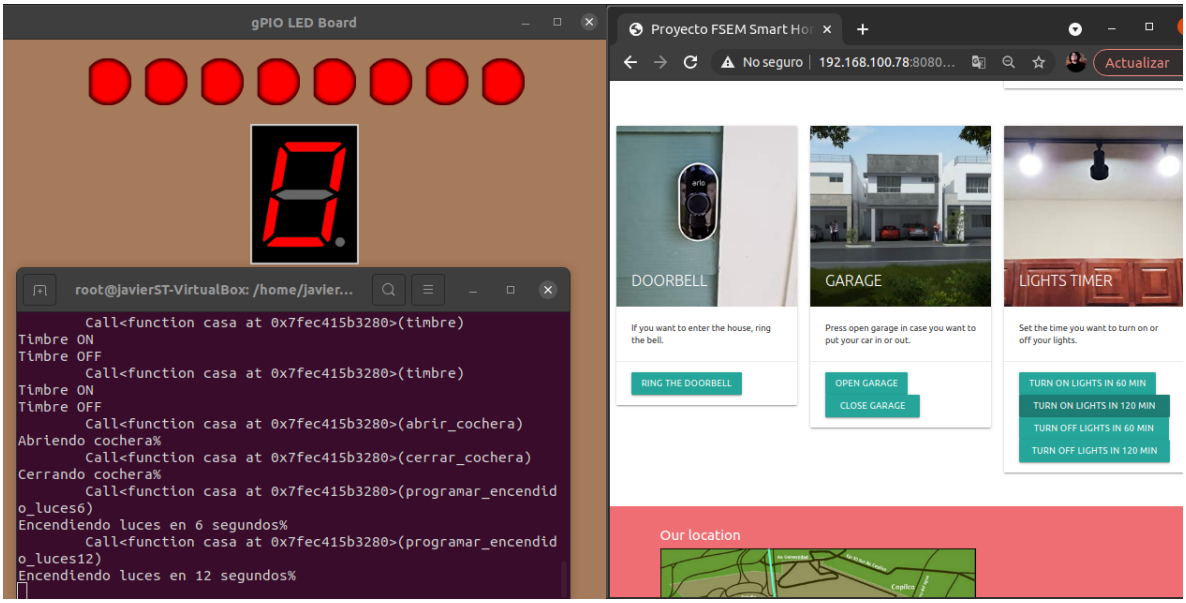


Imagen 24: Botón TURN ON LIGTHS IN 120 MIN

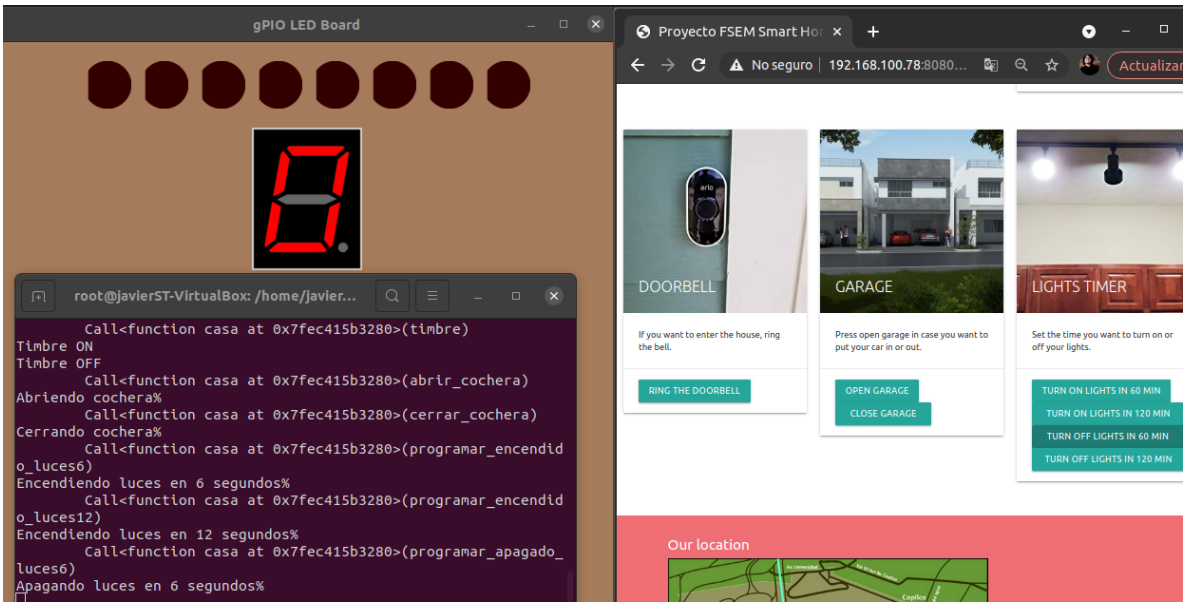


Imagen 25: Botón TURN OFF LIGTHS IN 60 MIN

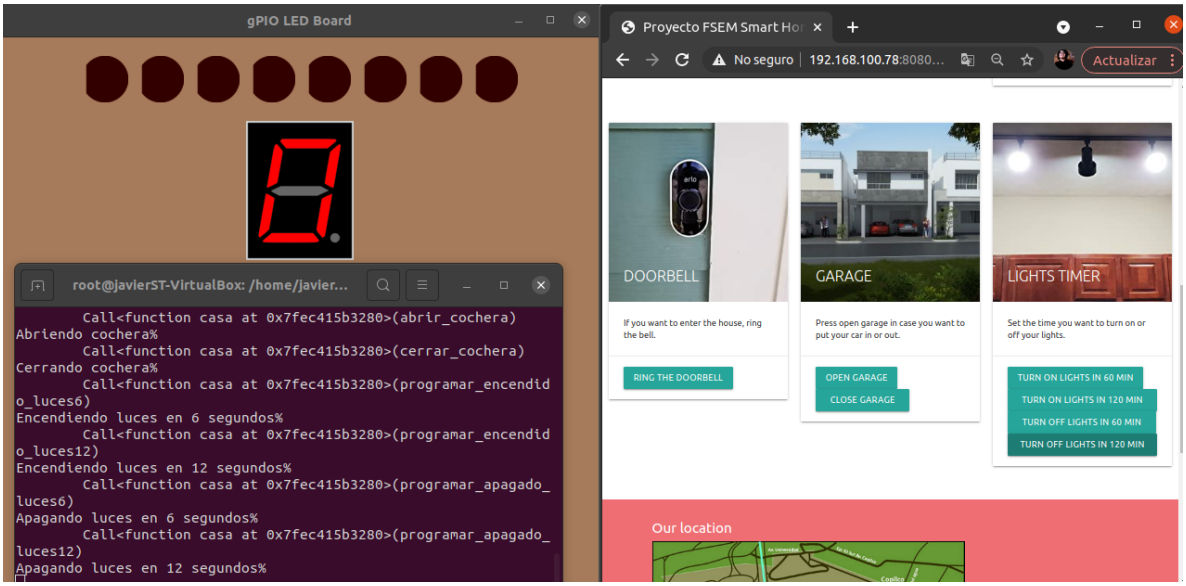


Imagen 26: Botón TURN OFF LIGTHS IN 120 MIN

8. Conclusiones

La realización del proyecto me ayudó a comprender la materia y me aclaró conocimientos, principalmente en la forma aplicativa y la forma en la que se enlazó el servidor web con el resto de los componentes, como con la página del html y el código en Python.

El funcionamiento de la Raspberry es muy importante, y gracias a este tipo de tecnologías, se pueden realizar pruebas, programas y ejecuciones de esta, como por ejemplo con el proyecto, que aunque no se pudo realizar de manera física, sí lo pudimos implementar de forma simulada.

Fue de gran ayuda los programas anteriores realizados en la materia, porque de esta forma se tomaron conceptos e ideas para la implementación exitosa del proyecto.

9. Fuentes de Consulta

- Matamoros de María y Campos José Mauricio. Apuntes Fundamentos de Sistemas Embebidos. UNAM. 2021.
- Valvano, J. (2003). Sistemas Embebidos. Recuperado de http://www.ieec.uned.es/investigacion/Dipseil/PAC/archivos/Informacion_de_referencia_ISE5_3_1.pdf
- Ojeda, L. (2019). Raspberry Pi. Recuperado de https://raspberrypi.cl/que-es-raspberry/
- Rodríguez, E. Raspberry Pi. Recuperado de https://www.xataka.com/makers/cero-maker-todo-necesario-para-empezar-raspberry-pi
- Delgado, A. (2020). ¿Qué es Raspberry Pi y para qué sirve? Recuperado de https://www.geeknetic.es/Raspberry-Pi/que-es-y-para-que-sirve
- Velasco, R. (2020). Análisis de Raspberry, Recuperado de https://hardzone.es/reviews/perifericos/analisis-raspberry-pi-3-modelo-b/