Proyecto Final:
Concentrador básico Smart Home

Fundamentos de Sistemas Embebidos

Luis Mauricio Barrientos Veana

1. OBJETIVO

Se elaborará la simulación de un concentrador que permite monitorear y administrar remotamente via navegador los dispositivos inteligentes domésticos vinculados.

2. MATERIAL

1. Plataforma:
   - Computadora con sistema operativo Linux,
   - Máquina virtual con sistema operativo Linux

2. Interprete de Python 3.5 instalado.

3. DESCRIPCIÓN DEL FUNCIONAMIENTO DE LA RASPBERRY PI

La página oficial de raspberrypi.org nos dice la Raspberry Pi es una computadora de bajo costo del tamaño de una tarjeta de crédito que se conecta a un monitor de computadora o televisor, y utiliza un teclado y un mouse estándar. Es un pequeño dispositivo que permite a personas de todas las edades explorar la computación y aprender a programar en lenguajes como Scratch y Python. Es capaz de hacer todo lo que se espera que haga una computadora de escritorio, desde navegar por Internet y reproducir videos de alta definición, hasta hacer hojas de cálculo, procesamiento de textos y jugar juegos.

Además, la Raspberry Pi tiene la capacidad de interactuar con el mundo exterior, y se ha utilizado en una amplia gama de proyectos de creadores digitales, desde máquinas de música y detectores de padres hasta estaciones meteorológicas y casas de pájaros con cámaras infrarroja.

Por eso en este proyecto se haráá uso de la Raspberry pi para simular un controlador del hogar inteligente, con una interfaz web que permitirá seleccionar diversos botones para manipular los siguientes elementos de un hogar:

- Encender/apagar focos
- Desplegado de cámaras de vigilancia
- Atenuado de focos
- Detección de timbre
- Apertura/cierre de cochera
- Programado de encendido/apagado de focos

4. Descripción de los componentes a utilizar

La página oficial de Raspberry pi nos indica lo siguiente sobre estos elementos:

GPIO pins: Una característica poderosa de la Raspberry Pi es la fila de pines GPIO a lo largo del borde superior de la

placa. GPIO significa Entrada/Salida de Propósito General. Estos pines son una interfaz física entre la Raspberry Pi y el mundo exterior. En el nivel más simple, puede pensar en ellos como interruptores que puede encender o apagar (entrada) o que el Pi puede encender o apagar (salida).

PWM: Le permite dar comportamientos analógicos a los dispositivos digitales, tales como LEDs. Esto significa que en lugar de que un LED simplemente esté encendido o apagado, puede controlar su brillo.

5. INFORMACIÓN SOBRE EL CUIDADO Y RIESGOS

En caso de realizar la configuración de manera física es importante resaltar las características generales de la Raspberry Pi para tener un manejo adecuado.

Si se siguen las instrucciones, utilizar los pines GPIO puede ser seguro. Sin embargo, conectar cables y fuentes de alimentación al azar en la Raspberry puede quemarla, principalmente si se usan los pines de 5V.

También pueden se puede descomponer si se conectan dispositivos que usan requieran mucho poder, como los motores.

6. INSTRUCCIONES

Paso 1: Configuración del Simulador

Nota: Para realizar el proyecto se tomó como base el simulador proporcionado por el profesor José Mauricio Matamoros de Maria y Campos en el programa 2. Por lo tanto, es necesario descargarlo y seguir los siguientes pasos.

Descargue el proyecto de https://github.com/barrientosmauricio/Proyecto-Final-Sistemas-Embebidos.git ejecutando la siguientelínea de comandos:

- git clone https://github.com/barrientosmauricio/Proyecto-Final-Sistemas-Embebidos.git
- cd src

A continuación, instale todas las dependencias requeridas por el simulador usando *pi*

- sudo apt install python3-tk
  - pip install --user -r requirements.txt

Para que se le permita al simulador funcionar de servidor web y controlar todos los elementos del puerto GPIO es necesario conocer su ip local y colocarla en el archivo webserver.py, para eso haga lo siguiente:

- ip -4 address|grep inet

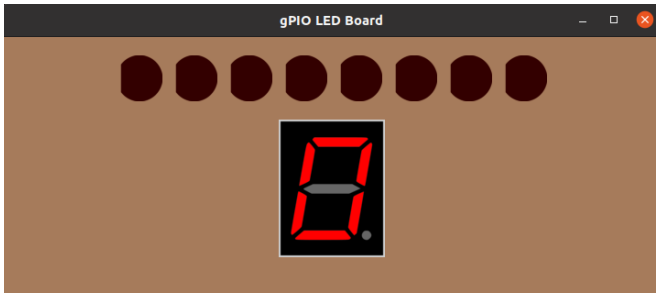
Abra el archivowebserver.py y coloque su ip en la línea 30:

```
30   address =
```

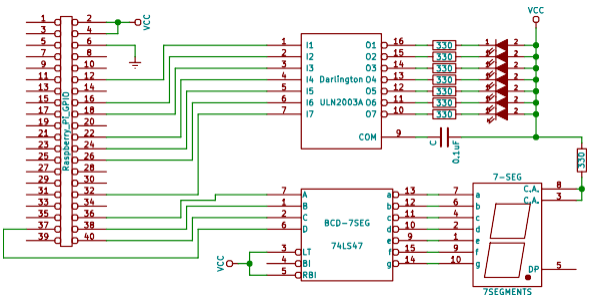
Finalmente, pruebe el simulador ejecutando la siguiente línea:

- python3 ./webserver.py

Si la configuración es correcta, verá una ventana similar a la de la Figura 1a. Este simulador implementa el circuito mostrado en la Figura 1b



(a) Simulador de tarjeta con leds para la GPIO de la Raspberry Pi



(b) Circuito implementado en el simulador

Figura 1: Simulador y circuito implementado

7. DESARROLLO DE LOS COMPONENTES DE SOFTWARE

Encender focos

El código mostrado en figura 2 muestra cómo se **encienden todos los 8 focos** de la casa utilizando la Raspberry Pi.

```
def _casa_luces():
    ledsoff()
    pwm_off()

#while True: # Forever
                    # Wait 500ms
    GPIO.output(32, GPIO.HIGH) # Turn led on
    GPIO.output(26, GPIO.HIGH)
    GPIO.output(24, GPIO.HIGH)
    GPIO.output(22, GPIO.HIGH)
    GPIO.output(18, GPIO.HIGH)
    GPIO.output(16, GPIO.HIGH)
    GPIO.output(12, GPIO.HIGH)
    GPIO.output(10, GPIO.HIGH)

    print("Luces encendidas")

    pass
```

Figura 2: Función def _casa_luces del archivo ledmanager.py

Apagar focos

El código mostrado en figura 4 muestra cómo se **activan y desactivan** todas las **cámaras de vigilancia** de la casa, que en este caso están asignadas a los pines 3,4,5 y 6 de izquierda a derecha en el simulador.

```
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
```

Figura 4: Función def _casa_camaras del archivo ledmanager.py

Tocar timbre

El código mostrado en figura 5 muestra cómo se **activa el timbre**, asignado al pin 1 de derecha a izquierda en el simulador.

```
def _casa_timbre():
    ledsoff()
    pwm_off()
    GPIO.output(10, GPIO.HIGH)
    print("Timbre ON")
    sleep(2)

    GPIO.output(10, GPIO.LOW)
    print("Timbre OFF")
    sleep(1)
```

Figura 5: Función def _casa_timbre del archivo ledmanager.py

Atenuar focos

El código mostrado en figura 6a y 6b muestra cómo se **atenúan los focos al 75% y al 50%.**

```
def _casa_atenuar_75():
    pwm_off()
    ledsoff()
    pwm8.start(75)
    pwm7.start(75)
    pwm6.start(75)
    pwm5.start(75)
    pwm4.start(75)
    pwm3.start(75)
    pwm2.start(75)
    pwm1.start(75)
    print("Atenuendo focos al 75%")

    pass
```

```
def _casa_atenuar_50():
    pwm_off()
    ledsoff()
    pwm8.start(50)
    pwm7.start(50)
    pwm6.start(50)
    pwm5.start(50)
    pwm4.start(50)
    pwm3.start(50)
    pwm2.start(50)
    pwm1.start(50)
    print("Atenuendo focos al 50%")

    pass
```

a) Atenuador al 75%

a) Atenuador al 50%

Figura 6: Funciones def _casa_atenuar75 y def _casa_atenuar_50 del archivo ledmanager.py

El código mostrado en figura 7a y 7b muestra cómo se **atenúan los focos al 25% y al 10%.**

```
def _casa_atenuar_25():
    pwm_off()
    ledsoff()
    pwm8.start(25)
    pwm7.start(25)
    pwm6.start(25)
    pwm5.start(25)
    pwm4.start(25)
    pwm3.start(25)
    pwm2.start(25)
    pwm1.start(25)
    print("Atenuendo focos al 25%")

    pass
```

a) Atenuador al 75%

```
def _casa_atenuar_10():
    pwm_off()
    ledsoff()
    pwm8.start(10)
    pwm7.start(10)
    pwm6.start(10)
    pwm5.start(10)
    pwm4.start(10)
    pwm3.start(10)
    pwm2.start(10)
    pwm1.start(10)
    print("Atenuendo focos al 10%")

    pass
```

a) Atenuador al 50%

Figura 7: Funciones def _casa_atenuar25 y def _casa_atenuar10 del archivo ledmanager.py

Abrir cochera

El código mostrado en figura 8 muestra el recorrido que hace la cochera de izquierda a derecha cuando se abre, simulándose en los 8 pines.

```
def _casa_abrir_cochera():
    pwm_off()
    ledsoff()

    sleep(0.5)                # Wait 500ms
    GPIO.output(10, GPIO.HIGH) # Turn led on
    sleep(0.5)
    GPIO.output(10, GPIO.LOW)  # Turn led off
    GPIO.output(12, GPIO.HIGH) # Turn led on
    sleep(0.5)
    GPIO.output(12, GPIO.LOW)  # Turn led off
    GPIO.output(16, GPIO.HIGH) # Turn led on
    sleep(0.5)                # Espera 500ms
    GPIO.output(16, GPIO.LOW)  # Turn led off
    GPIO.output(18, GPIO.HIGH) # Turn led on
    sleep(0.5)                # Espera 500ms
    GPIO.output(18, GPIO.LOW)  # Turn led off
    GPIO.output(22, GPIO.HIGH) # Turn led on
    sleep(0.5)                # Espera 500ms
    GPIO.output(22, GPIO.LOW)  # Turn led off
    GPIO.output(24, GPIO.HIGH) # Turn led on
    sleep(0.5)                # Espera 500ms
    GPIO.output(24, GPIO.LOW)  # Turn led off
    GPIO.output(26, GPIO.HIGH) # Turn led on
    sleep(0.5)                # Espera 500ms
    GPIO.output(26, GPIO.LOW)  # Turn led off
    GPIO.output(32, GPIO.HIGH) # Turn led on
    sleep(0.5)                # Espera 500ms
    GPIO.output(32, GPIO.LOW)  # Turn led off
    print("Abriendo cochera%")

    pass
```

Figura 8: Función def _casa_abrir_cochera del archivo ledmanager.py

Cerrar cochera

El código mostrado en figura 9 muestra el recorrido que hace la cochera de derecha a izquierda cuando se cierra, simulándose en los 8 pines.

```
def _casa_cerrar_cochera():

    sleep(0.5)                # Wait 500ms
    GPIO.output(32, GPIO.HIGH) # Turn led on
    sleep(0.5)
    GPIO.output(32, GPIO.LOW)  # Turn led off
    GPIO.output(26, GPIO.HIGH) # Turn led on
    sleep(0.5)
    GPIO.output(26, GPIO.LOW)  # Turn led off
    GPIO.output(24, GPIO.HIGH) # Turn led on
    sleep(0.5)                # Espera 500ms
    GPIO.output(24, GPIO.LOW)  # Turn led off
    GPIO.output(22, GPIO.HIGH) # Turn led on
    sleep(0.5)                # Espera 500ms
    GPIO.output(22, GPIO.LOW)  # Turn led off
    GPIO.output(18, GPIO.HIGH) # Turn led on
    sleep(0.5)                # Espera 500ms
    GPIO.output(18, GPIO.LOW)  # Turn led off
    GPIO.output(16, GPIO.HIGH) # Turn led on
    sleep(0.5)                # Espera 500ms
    GPIO.output(16, GPIO.LOW)  # Turn led off
    GPIO.output(12, GPIO.HIGH) # Turn led on
    sleep(0.5)                # Espera 500ms
    GPIO.output(12, GPIO.LOW)  # Turn led off
    GPIO.output(10, GPIO.HIGH) # Turn led on
    sleep(0.5)                # Espera 500ms
    GPIO.output(10, GPIO.LOW)  # Turn led off'''
    print("Cerrando cochera%")
    pass
```

Figura 9: Función def _casa_cerrar_cochera del archivo ledmanager.py

Programar encendido y apagado de focos

El código mostrado en figura 10 muestra cómo se **programa el encendido** de los focos de la casa en 6 segundos, que simula que son 60 minutos desde la interfaz web.

```
def _casa_programar_encendido_luces6():
    ledsoff()
    pwm_off()

    sleep(6)                # Wait 500ms
    GPIO.output(32, GPIO.HIGH)  # Turn led off
    GPIO.output(26, GPIO.HIGH)
    GPIO.output(24, GPIO.HIGH)
    GPIO.output(22, GPIO.HIGH)
    GPIO.output(18, GPIO.HIGH)
    GPIO.output(16, GPIO.HIGH)
    GPIO.output(12, GPIO.HIGH)
    GPIO.output(10, GPIO.HIGH)
    print("Encendiendo luces en 6 segundos%")
    pass
```

Figura 10: Función def _casa_programar_encendido_luces6 del archivo ledmanager.py

El código mostrado en figura 11 muestra cómo se programa el encendido de los focos de la casa en 12 segundos, que simula que son 120 minutos desde la interfaz web.

```
def _casa_programar_encendido_luces12():
    ledsoff()
    pwm_off()

    sleep(12)                # Wait 500ms
    GPIO.output(32, GPIO.HIGH)  # Turn led off
    GPIO.output(26, GPIO.HIGH)
    GPIO.output(24, GPIO.HIGH)
    GPIO.output(22, GPIO.HIGH)
    GPIO.output(18, GPIO.HIGH)
    GPIO.output(16, GPIO.HIGH)
    GPIO.output(12, GPIO.HIGH)
    GPIO.output(10, GPIO.HIGH)
    print("Encendiendo luces en 12 segundos%")
    pass
```

Figura 11: Función def _casa_programar_encendido_luces12 del archivo ledmanager.py

El código mostrado en figura 12 muestra cómo se programa el apagado de los focos de la casa en 6 segundos, que simula que son 60 minutos desde la interfaz web.

```
def _casa_programar_apagado_luces6():

    pwm_off()

    sleep(6)                  # Wait 500ms
    GPIO.output(32, GPIO.LOW)  # Turn led off
    GPIO.output(26, GPIO.LOW)
    GPIO.output(24, GPIO.LOW)
    GPIO.output(22, GPIO.LOW)
    GPIO.output(18, GPIO.LOW)
    GPIO.output(16, GPIO.LOW)
    GPIO.output(12, GPIO.LOW)
    GPIO.output(10, GPIO.LOW)
    print("Apagando luces en 6 segundos%")
    pass
```

Figura 12: Función def _casa_programar_apagado_luces6 del archivo ledmanager.py

El código mostrado en figura 13 muestra cómo se programa el apagado de los focos de la casa en 12 segundos, que simula que son 120 minutos desde la interfaz web.

```
def _casa_programar_apagado_luces12():

    pwm_off()

    sleep(12)                  # Wait 500ms
    GPIO.output(32, GPIO.LOW)  # Turn led off
    GPIO.output(26, GPIO.LOW)
    GPIO.output(24, GPIO.LOW)
    GPIO.output(22, GPIO.LOW)
    GPIO.output(18, GPIO.LOW)
    GPIO.output(16, GPIO.LOW)
    GPIO.output(12, GPIO.LOW)
    GPIO.output(10, GPIO.LOW)
    print("Apagando luces en 12 segundos%")
    pass
```

Figura 13: Función def _casa_programar_apagado_luces12 del archivo ledmanager.py

8. INTEGRACIÓN DE LOS COMPONENTES DE SOFTWARE

Archivo webserver.py

```python
import os
import sys
import json
import magic
from led_manager import casa
from http.server import BaseHTTPRequestHandler, HTTPServer
# import time
# import time

# Nombre o dirección IP del sistema anfitrión del servidor web
# address = "localhost"
address = "192.168.1.64"
# Puerto en el cual el servidor estará atendiendo solicitudes HTTP
# El default de un servidor web en producción debe ser 80
port = 8080


class WebServer(BaseHTTPRequestHandler):
    """Sirve cualquier archivo encontrado en el servidor"""
    def _serve_file(self, rel_path):
        if not os.path.isfile(rel_path):
            self.send_error(404)
            return
        self.send_response(200)
        mime = magic.Magic(mime=True)
        self.send_header("Content-type", mime.from_file(rel_path))
        self.end_headers()
        with open(rel_path, 'rb') as file:
            self.wfile.write(file.read())


    """Sirve el archivo de interfaz de usuario"""
    def _serve_ui_file(self):
        if not os.path.isfile("home.html"):
            err = "home.html not found."
            self.wfile.write(bytes(err, "utf-8"))
            print(err)
            return
        try:
            with open("home.html", "r") as f:
                content = "\n".join(f.readlines())
        except:
            content = "Error reading home.html"
        self.wfile.write(bytes(content, "utf-8"))

    def _parse_post(self, json_obj):
        if not 'action' in json_obj or not 'value' in json_obj:
            return
        switcher = {
            'casa' : casa

        }
        func = switcher.get(json_obj['action'], None)
        if func:
            print('\tCall{}({})'.format(func, json_obj['value']))
            func(json_obj['value'])
```

```python
    """do_POST controla todas las solicitudes recibidas vía POST, es
    decir, envíos de formulario. Aquí se gestionan los comandos para
    la Raspberry Pi"""
    def do_POST(self):
        # Primero se obtiene la longitud de la cadena de datos recibida
        content_length = int(self.headers.get('Content-Length'))
        if content_length < 1:
            return
        # Después se lee toda la cadena de datos
        post_data = self.rfile.read(content_length)
        # Finalmente, se decodifica el objeto JSON y se procesan los datos.
        # Se descartan cadenas de datos mal formados
        try:
            jobj = json.loads(post_data.decode("utf-8"))
            self._parse_post(jobj)
        except:
            print(sys.exc_info())
            print("Datos POST no recnocidos")

def main():
    # Inicializa una nueva instancia de HTTPServer con el
    # HTTPRequestHandler definido en este archivo
    webServer = HTTPServer((address, port), WebServer)
    print("Servidor iniciado")
    print ("\tAtendiendo solicitudes en http://{}:{}".format(
        address, port))

    try:
        # Mantiene al servidor web ejecutándose en segundo plano
        webServer.serve_forever()
    except KeyboardInterrupt:
        # Maneja la interrupción de cierre CTRL+C
        pass
    except:
        print(sys.exc_info())
    # Detiene el servidor web cerrando todas las conexiones
    webServer.server_close()
    # Reporta parada del servidor web en consola
    print("Server stopped.")


# Punto de anclaje de la función main
if __name__ == "__main__":
    main()
```

```python
    """do_GET controla todas las solicitudes recibidas vía GET, es
    decir, páginas. Por seguridad, no se analizan variables que lleguen
    por esta vía"""
    def do_GET(self):
        # Revisamos si se accede a la raiz.
        # En ese caso se responde con la interfaz por defecto
        if self.path == '/':
            # 200 es el código de respuesta satisfactorio (OK)
            # de una solicitud
            self.send_response(200)
            # La cabecera HTTP siempre debe contener el tipo de datos mime
            # del contenido con el que responde el servidor
            self.send_header("Content-type", "text/html")
            # Fin de cabecera
            self.end_headers()
            # Por simplicidad, se devuelve como respuesta el contenido del
            # archivo html con el código de la página de interfaz de usuario
            self._serve_ui_file()
        # En caso contrario, se verifica que el archivo exista y se sirve
        else:
            self._serve_file(self.path[1:])
```

Archivo home.html

Encender focos y cámaras de vigilancia

```
<!--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 modal-trigger" href="#modal1">Turn on lights</a>
                          <a onclick="handle(this, 'casa', 'apagar_luces')" class="waves-effect waves-light btn modal-trigger" href="#modal2"><i class="material-icons right"></i>Turn off lights</a>
                      </div>
                  </div>
                  </div>
              </div>
          </div>
<!--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/lYztjyEif1YC5B0un0oal0Xb02h0gYTrSoxwFSAfghd8SlIrtD1Vy_1bNeFWVGl7Lb7eCOKLW-nudELFmAW2v9fOk9dt_4Z5gWrfsNaONXB20NWjZk3yULSUi5nE3UQfWJyLyH0">
                    <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 modal-trigger" href="#modal3">Turn on cameras</a>
                        <a onclick="handle(this, 'casa', 'camaras_apagadas')" class="waves-effect waves-light btn modal-trigger" href="#modal4"><i class="material-icons right"></i>Turn off cameras</a>
                    </div>
                </div>
                </div>
            </div>
        </div>
```

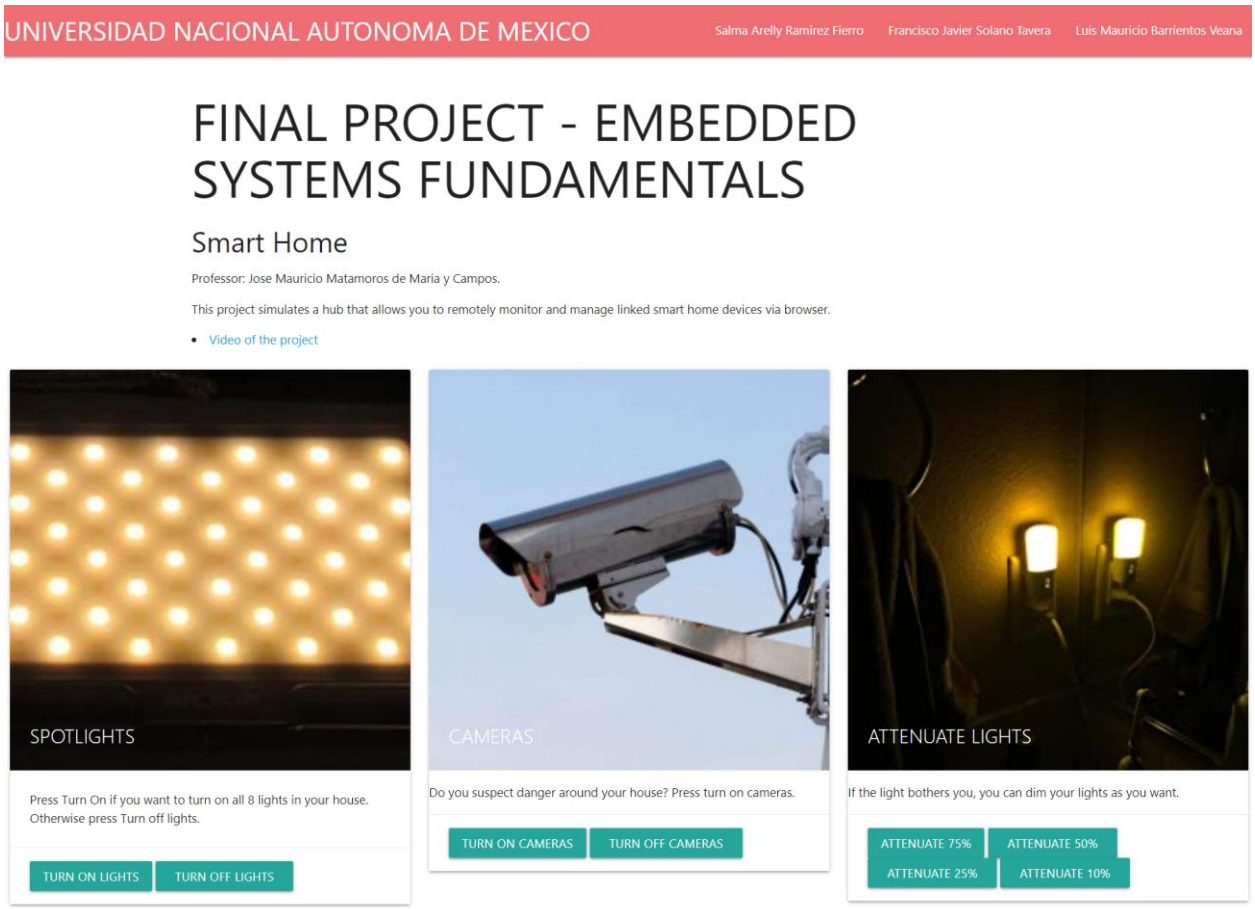
Atenuar focos y timbre

```
<!--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 modal-trigger" href="#modal5">Attenuate 75%</a>
                        <a onclick="handle(this, 'casa', 'atenuar_50')" class="waves-effect waves-light btn modal-trigger" href="#modal6"><i class="material-icons right"></i>Attenuate 50%</a>
                        <a onclick="handle(this, 'casa', 'atenuar_25')" class="waves-effect waves-light btn modal-trigger" href="#modal6"><i class="material-icons right"></i>Attenuate 25%</a>
                        <a onclick="handle(this, 'casa', 'atenuar_10')" class="waves-effect waves-light btn modal-trigger" href="#modal6"><i class="material-icons right"></i>Attenuate 10%</a>
                    </div>
                </div>
                </div>
            </div>
        </div>

        </div>
<!--SEGUNDA LINEA DE 3 BOTONES-->
        <div class="row">
<!--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/61qhgl10-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 modal-trigger" href="#modal7">Ring the doorbell</a>
                      </div>
                  </div>
                  </div>
              </div>
          </div>
```

Abrir/cerrar cochera y temporizador de focos

```
<!--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 modal-trigger" href="#modal9">Open garage</a>
                    <a onclick="handle(this, 'casa', 'cerrar_cochera')" class="waves-effect waves-light btn modal-trigger" href="#modal10"><i class="material-icons right"></i>Close garage</a>
                </div>
            </div>
            </div>
          </div>
        </div>
<!--TEMPORIZADOR DE 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/B1hq4m-rvfS._CR1062,0,1908,1908_UX256.jpg">
                <span class="card-title">LIGHTS TIMER</span>
                </div>
                <div class="card-content">
                <p>Set the time you want to turn on or off your lights.</p>
                </div>
                <div class="card-action">
                    <a onclick="handle(this, 'casa', 'programar_encendido_luces6')" class="waves-effect waves-light btn modal-trigger" href="#modal11">Turn on lights in 60 min </a>
                    <a onclick="handle(this, 'casa', 'programar_encendido_luces12')" class="waves-effect waves-light btn modal-trigger" href="#modal11"><i class="material-icons right"></i>Turn on lights in 120 min</a>
                    <a onclick="handle(this, 'casa', 'programar_apagado_luces6')" class="waves-effect waves-light btn modal-trigger" href="#modal12"><i class="material-icons right"></i>Turn off lights in 60 min</a>
                    <a onclick="handle(this, 'casa', 'programar_apagado_luces12')" class="waves-effect waves-light btn modal-trigger" href="#modal12"><i class="material-icons right"></i>Turn off lights in 120 min</a>
                </div>
            </div>
            </div>
          </div>
        </div>

      </div>
```

9. INTERFAZ FINAL



10. LINKS

- **Video: https://www.youtube.com/watch?v=5QWoXep6CeU**
- **Repositorio: https://github.com/barrientosmauricio/Proyecto-Final-Sistemas-Embebidos.git**

11. CONCLUSIONES

En este proyecto se cumplió con éxito la simulación de un concentrador básico en una casa inteligente; en un principio fue complicado visualizar todos los módulos que se requerían para completar el simulador, pero se tomó como base el simulador realizado en el programa 2 el cual tenía un esquema bien definido. Partiendo de ese punto se plantearon las soluciones del sistema definiendo los componentes de la interfaz y posteriormente se definieron las funciones relacionadas con cada componente de la interfaz web.

Al finalizar el trabajo me doy cuenta de que se debe llevar un proceso muy detallado para seleccionar los componentes de hardware y software que se requieren para implementar un sistema embebido, en este caso se pudieron haber separado los elementos del simulador para señalar de manera más clara el funcionamiento de cada acción en la casa, sin embargo, nos pareció que con los pines seleccionados fue suficiente para mostrar que los botones presionados por el usuario desde la interfaz web cumplían su función.

12. CUESTIONARIO

¿Cómo utilizar pwm en la Raspberry Pi?

¿Por qué es necesario configurar la ip en el sistema anfitrión?

13. BIBLIOGRAFÍA

Getting started with Raspberry Pi Pico - Control LED brightness with PWM | Raspberry Pi Projects

Computación física con Python - Pines GPIO | Proyectos Raspberry Pi

Matamoros, José M. Apuntes Fundamentos de Sistemas Embebidos. UNAM.