Tutorial: Proyecto Final Smart Home

Guadarrama Ortega César Alejandro
Fundamentos de Sistemas Embebidos

28 de mayo de 2022

Resumen

Este domunento explica de forma detallada el procedimiento de creación del proyecto "smart home". Comenzando por componenetes necesarios, consideraciones, creación del servidor web, librerías necesarias, explicación del código implementado, enlace a video demostrativo, etc. El video demostrativo se encuentra a traves de este enlace al video en YouTube: `https://youtu.be/i6VMrMJ1GQI`

1. Objetivo

Crear un proyecto de una casa inteligente. Este deberá incluir el encendido, apagado y control de iluminación interior, timbre inteligente, puerta de garaje inteligente, sistema de manejo de cámaras y como inclusión extra sistema de iluminación exterior a base de luz solar.

2. Lista de materiales

2.1. Materiales para el funcionamiento

- Raspberry Pi Model 3 o superior, con sistema opertivo Raspberry Pi OS (32 bits) instalado.
- Protoboard.
- Cables tipo jumpers.
- Cable para protoboard, rojo y negro 2m.
- 2 botones de dos terminales, diferente color.
- Resistores de 100K x1, 12K x1, 22K x1, 1K x1, 500 x1 y 220 OHMs x3.
- 5 diodos LED, blancos de preferencia.
- 1 Sensor ultrasónico HC-SR04.
- 1 Zumbador (Buzzer).
- 1 Micro servomotor Sg90.
- 1 Conector de batería de 9[V]
- 1 Batería de 9[V]
- 1 Fotoresistor.
- 1 Transistor 2N2222A

2.2. Materiales para la maqueta (opcionales)

- 1/2 de pliego de papel batería.
- 1/8 de pliego de papel corrugado.
- Pegamento blanco 850.
- Silicon frio UHU.

- Soldadura 2m.
- Soldador.
- Exacto.
- Tijeras.
- Vegetación artificial
- Carrito.

3. Descripción de los componentes

3.1. Botón Pulsador

Los botones pulsadores pueden explicarse como simples interruptores de control de energía de una máquina o aparato. Se trata generalmente de interruptores metálicos o termoplásticos destinados a facilitar el acceso al usuario.

3.2. Resistor

Se denomina resistor al componente electrónico diseñado para introducir una resistencia eléctrica determinada entre dos puntos de un circuito. Es un dispositivo electrónico de dos terminales y que no tiene polaridad, su principal función es la disipación de calor, proceso en el cual se convierte la energía eléctrica en energía térmica, es decir calor, su unidad de medida es ohm (Ω). Los resistores se utilizan en los circuitos para limitar el valor de la corriente o para fijar el valor de la tensión.

3.3. Diodo LED

Un diodo LED es un dispositivo que permite el paso de corriente en un solo sentido y que al ser polarizado emite un haz de luz. Trabaja como un diodo normal pero al recibir corriente eléctrica emite luz. Los LED trabajan aproximadamente con corriente de 2V. Para conectarlos a un voltaje distinto, se debe usar una resistencia.

Su funcionamiento es bastante simple, se conecta la corriente al semiconductor superior del diodo LED lo cual permitirá el paso de corriente eléctrica y hará que el semiconductor emita luz. Según el material del que esté elaborado el semiconductor, los diodos LED proyectarán luces de distintos colores.

3.4. Sensor Ultrasónico HC-SR04.

El sensor HC-SR04 es un sensor de distancia de bajo costo, su uso es muy frecuente en la robótica, utiliza transductores de ultrasonido para detectar objetos.

Su funcionamiento consiste en emitir un sonido ultrasónico por uno de sus transductores, y esperar que el sonido rebote de algún objeto presente, el eco es captador por el segundo transductor. La distancia es proporcional al tiempo que demora en llegar el eco.

3.5. Zumbador

El zumbador es un pequeño transductor capaz de convertir la energía eléctrica en sonido.

Para hacerlos funcionar solo basta conectar el positivo con el + y la tierra o negativo con el – de una batería o cualquier fuente de corriente directa.

Se basa en el efecto piezoeléctrico de los materiales, este efecto funciona de tal manera que cuando se aplica un voltaje el volumen del material cambia ligeramente.

3.6. Micro Servomotor Sg90

Un servo RC en miniatura estándar de la industria. Comúnmente utilizado en drones, aviones o helicópteros RC ligeros. El tamaño en miniatura estándar del SG90 lo hace excelente y conveniente para usar con proyectos mecatrónicos de microcontroladores. Cuenta con 180 grados de rotación con 0.125oz-in de torque

3.7. Fotoresistor

Un fotoresistor, o LDR (light-dependent resistor) es un dispositivo cuya resistencia varia en función de la luz recibida. Podemos usar esta variación para medir, a través de las entradas analógicas, una estimación del nivel del luz.

3.8. Transistor 2N2222A

Este dispositivo semiconductor es un transistor bipolar de juntura NPN. Su encapsulado es el TO-92, cuya estructura es plastico con tres terminales (Colector, Base, Emisor).

4. Descripción del funcionamiento de la Raspberry Pi

La Raspberry Pi es un pequeño ordenador para casi todo. Los usuarios de un ordenador conocen todas las bondades que tienen esta clase de aparatos. Si echamos un vistazo a las características del modelo, nos topamos con que el procesador es un chip que contiene todo lo que necesita para funcionar o SoC acompañado de características como una RAM a elegir, antena WiFi, Bluetooth, puertos USB de hasta 3.1, conexión a Internet por cable, puertos HDMI y conexión con salida de audio con minijack.

Los pines GPIO de la Raspberry Pi son los grandes olvidados, pero que sabiendo como utilizarlos nos abre un mundo de posibilidades. GPIO (General Purpose Input/Output) es, como su propio nombre indica, un sistema de E/S (Entrada/Salida) de propósito general, es decir, una serie de conexiones que se pueden usar como entradas o salidas para usos múltiples. Estos pines están incluidos en todos los modelos de Raspberry Pi, para que puedas realizar proyectos interesantes. En la figura 1 se muestran los pines GPIO de la Raspberry Pi.

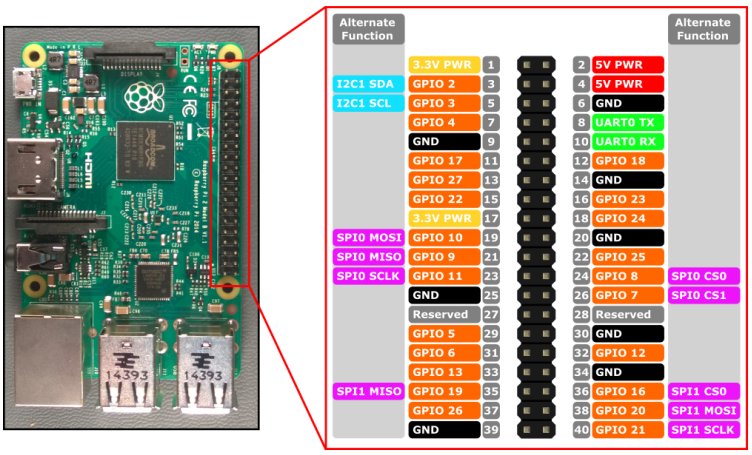

Figura 1: Raspberry Pi - Pines GPIO

5. Información sobre el cuidado de la salud, advertencias y cuidado de componentes

Aunque se están utilizando voltajes muy pequeños (3.3 [V] y 5[V]), hay que considerar el cuidado de los componentes electrónicos conectando de manera correcta la entrada de corriente y la tierra. Hay que evitar el contacto entre cables para no hacer un corto circuito que afecte a toda la conexión e incluso la tarjeta Raspberry Pi. Se recomienda colocar la Raspberry Pi en una base solida y seca a una distancia considerada de componentes estáticos. Finalmente, no hay que remover la tarjeta microSD mientras la Raspberry Pi esta en funcionamiento, podemos corromper el sistema operativo y tendremos que realizar la configuración de nuevo.

Figura 2: Señal de precaución - Riesgo eléctrico

6. Configuración de la tarjeta controladora (Raspberry Pi)

El software instalado en la Raspberry Pi es el Raspberry Pi OS de 32 bits, se pueden instalar versiones mejoradas del sistema incluso diferentes, para este proyecto se utilizó la anteriormente nombrada.

La instalación del sistema operativo es muy sencilla, solo debemos descargar un programa desde la página oficial llamado Raspberry Pi Imager este programa instalará la imagen del sistema en un dispositivo de almacenamiento por ejemplo una microSD, y quedara lista para colocarla en la Raspberry Pi. Se recomienta una memoria de 128 GB o más para un funcionamiento optimo. En la figura 3 se observa el software de escritura de imagen.

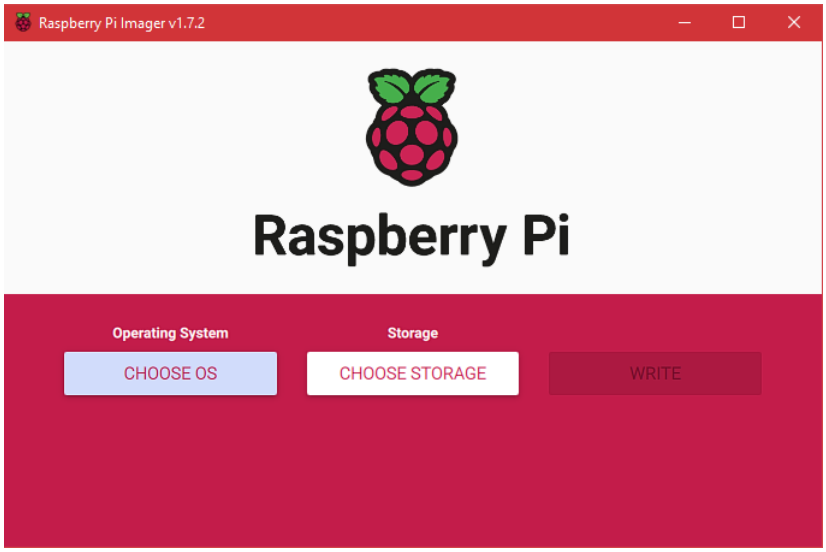

Figura 3: Software Raspberry Pi Imager

7. Desarrollo

Una vez tengamos los componentes, materiales y la Raspberry lista comenzamos a desarrollar el proyecto.

7.1. Raspberry Pi como punto de acceso inalámbrico

Comenzamos con la configuración de la Raspberry como punto de acceso inalambrico. Para operar como punto de acceso la Raspberry Pi necesita tener instalado el software apropiado, incluyendo un servidor DHCP para proporcionar a los dispositivos que se conecten una dirección IP. Se comienza por instalar los paquetes *DNSMasq y HostAPD*:

```
> apt-get install dnsmasq hostapd
> pip3 install -U python-magic
```

Detén los servicios para poder reconfigurarlos.

```
> systemctl stop dnsmasq
> systemctl stop hostapd
```

Para configurar una red independiente con servidor DHCP la Raspberry Pi se debe tener asignada una dirección IP estática en el adaptador inalámbrico que proveerá la conexión. Se configurará para una red privada clase C, es decir con direcciones IP del tipo 198.168.x.x. Así mismo, se supondrá que el dispositivo inalámbrico utilizado es *wlan0*. **Nota: Este procedmiento puede dejar al adaptador wi-fi sin conexión a internet, se recomienda colocar un cable de red (RJ45) al puerto Ethernet del dispositivo.**

Para configurar la dirección IP estática edite el archivo de configuración */etc/dhcpcd.conf* como superusuario usando *> sudo nano*. **Ejemplo: *> sudo nano /etc/dhcpcd.conf***

Dentro del archivo reemplaza el código que está por el siguiente:

```
interface wlan0
    static ip_address=192.168.1.254/24
    nohook wpa_supplicant
```

A continuación, reinicia el cliente DHCP

```
> service dhcpcd restart
```

El siguiente paso consiste en configurar el servidor DHCP, provisto por el servicio *dnsmasq*. De manera predeterminada el archivo de configuración */etc/dnsmasq.conf* contiene mucha información que no es necesaria, por lo que es más fácil comenzar desde cero. Respáldelo y cree uno nuevo con el siguiente texto:

```
# Use the require wireless interface - usually wlan0
interface=wlan0
# Reserve 20 IP addresses, set the subnet mask, and lease time
dhcp-range=192.168.1.200,192.168.1.220,255.255.255.0,24h
```

Esta configuración proporcionará 20 direcciones IP entre 192.168.1.200 y 192.168.1.220, válidas durante 24 horas. Ahora debe iniciarse el servidor DHCP

```
> systemctl start dnsmasq
```

Para configurar el punto de acceso se debe editar el archivo de configuración */etc/hostapd/hostapd.conf* con los parámetros adecuados. Respáldelo y cree uno nuevo con el siguiente texto:

```
# Wireless interface
interface=wlan0
# Specification: IEEE802.11
driver=nl80211
# The SSID or name of the network
ssid=Raspbberry_Apellido
# Password of the network
wpa_passphrase=12345678
wpa=2
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
# Mode and frequency of operation
hw_mode=g
# Broadcast channel
channel=5
wmm_enabled=0
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
rsn_pairwise=CCMP
```

Importante: Tanto el nombre de la red o SSID y la contraseña no deben entrecomillarse. La contraseña debe tener entre 8 y 64 caracteres. Cambie el SSID a Raspberry_Aplellido para evitar conflictos. En este caso la contraseña de la red es 12345678 y el SSID es Raspberry_Guadarrama

Ahora edite el archivo */etc/default/hostapd* y reemplace la línea que comienza con *DAEMON_CONF* con:

```
DAEMON_CONF="/etc/hostapd/hostapd.conf"
```

Finalmente, habilite los servicios para iniciar el punto de acceso:

```
> systemctl unmask hostapd
> systemctl enable hostapd
> systemctl start hostapd
```

Verifique que los servicios se están ejecutando

```
> systemctl status hostapd
> systemctl status dnsmasq
```

Para comprobar que se realizó la configuración de forma exitosa busca la red inalámbrica con el nombre que pusiste y accede a ella. Si te conectas a la Raspberry no habrá ningún problema ya que aun no subimos el servidor web de las peticiones.

7.2. Módulo de manipulación de los componentes electrónicos

En segmento de código de la figura 4 se muestran las librerías necesarias para el funcionamiento del módulo principal donde se manipulan los componentes electrónicos.

```
1  ##web_smart_home.py
2  #autor: Guadarrama Ortega César Alejandro
3
4  import RPi.GPIO as GPIO
5  import time
6  import sys
7  import cv2
8  from Ultrasonico_RaspberryPi import HCSR04
9
```

Figura 4: Librerías de web_smart_home.py

En segmento de código de la figura 5 muestra los pines GPIO que se usaran para la entrada y salida de los componentes electrónicos.

```
10 #Pines para usar
11 led=15
12 push_yellow = 13
13 BUZZER_PIN = 12  # Buzzer
14 push_red=11
15 servo_uno = 3
16 ultra = HCSR04(5,7)
```

Figura 5: Pines de uso - web_smart_home.py

En segmento de código de la figura 6 y 7 se muestran algunas varibles para que el sistema funcione correctamente.

```
19 #estados de botón
20 estadoLed = 0
21 estadoAnterior=0
22
23 #notas de timbre
24 notes = {  # Dict that contains the the first scale notes frequencies
25     'C': 0.002109, 'D': 0.001879, 'E': 0.001674, 'F': 0.001580, 'G': 0.001408,
26     'A': 0.001254, 'B': 0.001117, 'C1': 0.001054,
27 }
```

Figura 6: Estados del botón y notas del buzzer - web_smart_home.py

```
38 #variables servo
39 pulso = GPIO.PWM(servo_uno, 50)
40 pulso.start(2.5)
41
42 ##Funciones PWM
43 pwm_led=GPIO.PWM(led,500)
44 pwm_led.start(100)
```

Figura 7: Variables del servo y funciones PWM - web_smart_home.py

En la figura 8 se muestra la configuración inicial de los GPIO.

```
#Configuraciones
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(led,GPIO.OUT)
GPIO.setup(push_yellow,GPIO.IN)
GPIO.setup(push_red,GPIO.IN)
GPIO.setup(BUZZER_PIN, GPIO.OUT)
GPIO.setup(servo_uno, GPIO.OUT)
```

Figura 8: Configuraciones GPIO web_smart_home.py

El el segmento de código de la figura 9 se muetran las funciones principales para que funcione el timbre. Al presionar el timbre debe sonar el Zumbador con una nota especifica y desplegar una vista de la cámara 1.

```
#Funciones de timbre
def play_sound(duration, frequency):  # Play different frequencies
    for i in range(duration):
        GPIO.output(BUZZER_PIN, GPIO.HIGH)
        time.sleep(frequency)
        GPIO.output(BUZZER_PIN, GPIO.LOW)
        time.sleep(frequency)
def cam(camara):
    delay = 1/30 #tiempo de espera en cada frame
    #Seleccion de camaras
    if camara == 1:
        cap = cv2.VideoCapture('cam1.mov')
    if camara == 2:
        cap = cv2.VideoCapture('cam2.mov')
    #ciclo while de reproducción
    while (cap.isOpened()):
        ret,im=cap.read()
        if ret==False:
            break
        cv2.imshow('Imagen',im)
        if cv2.waitKey(1) & 0xFF == 27:
            break
        time.sleep(delay)
    cap.release()
    cv2.destroyAllWindows()
```

Figura 9: Funciones principales del timbre y cámara 1 - web_smart_home.py

El la figura 10 se muestra las funciones principales del control de encendido y apagado de LED's.

```
#LEDS
def ledson(word):
    if word == "On":
        pwm_led.ChangeDutyCycle(100)

def ledsoff(word):
    if word == "Off":
        pwm_led.ChangeDutyCycle(0)
```

Figura 10: Funciones principales de encendido y apagado de LED's - web_smart_home.py

La función que hace la llamada del timbre se muetra en la figura 11.

```
#Timbre
def timbre(word):
    if word == "On":
        print("se presiono timbre")
        for n in ['C', 'D', 'E', 'F', 'G', 'A', 'B', 'C1']:  # Play all notes
            play_sound(100, notes[n])
        cam(1)
```

Figura 11: Función de llamada del timbre - web_smart_home.py

Continuamos con la implementación del garaje inteligente este funciona a través de una variación de distancia entre el sensor ultrasónico y la pared de tope. Al cambiar la distancia dentro del sensor se abrirá la puerta de forma automática a través de un servomotor. Este segmento de código se muestra en la figura 12.

```
88   #Garage
89   def garage(word):
90       if word=="Automatic":
91           aux=True
92           while aux:
93               time.sleep(1)
94               distancia = ultra.distance()
95               print("%.1f"%distancia)
96               if distancia<5.5:
97                   for i in range(0,90):
98                       time.sleep(0.005)
99                       grados = (1.0/18.0)*(i)+2.5
100                      pulso.ChangeDutyCycle(grados)
101                  time.sleep(6)
102                  for i in range(90,0,-1):
103                      time.sleep(0.005)
104                      grados = (1.0/18.0)*(i)+2.5
105                      pulso.ChangeDutyCycle(grados)
106                  aux=False
107                      #sleep(6)
108      if word=="Open":
109          for i in range(0,90):
110              time.sleep(0.005)
111              grados = (1.0/18.0)*(i)+2.5
112              pulso.ChangeDutyCycle(grados)
113      if word=="Close":
114          for i in range(90,0,-1):
115              time.sleep(0.005)
116              grados = (1.0/18.0)*(i)+2.5
117              pulso.ChangeDutyCycle(grados)
```

Figura 12: Función principal del garaje inteligente - web_smart_home.py

Podemos variar la intensidad de las luces implementadas, este segmento de código se muestra en la figura 13.

```
118  #Cambiar iluminacion
119  def pwm(valor):
120      duty_s=valor
121      duty=int(duty_s)
122      pwm_led.ChangeDutyCycle(duty)
```

Figura 13: Función principal de variación de luminosidad - web_smart_home.py

Finalmente tenemos la implementación de cámaras, estas son dos una colocada enfrente del timbre y otra en la lateral superior de la fachada de la casa. Las cámaras representan videos cortos de 10 segundos a 20 fps. Estas se pueden visualizar independientemente de que se toque el timbre o no. El segmento de código que muestra esta implementación se muestra en la figura 14.

```
124  def camara(numero):
125      delay = 1/30 #tiempo de espera en cada frame
126      #Seleccion de camaras
127      if numero == "1":
128          cap = cv2.VideoCapture('cam1.mov')
129      if numero == "2":
130          cap = cv2.VideoCapture('cam2.mov')
131      #ciclo while de reproducción
132      while (cap.isOpened()):
133          ret,im=cap.read()
134          if ret==False:
135              break
136          cv2.imshow('Imagen',im)
137          if cv2.waitKey(1) & 0xFF == 27:
138              break
139          time.sleep(delay)
140      cap.release()
141      cv2.destroyAllWindows()
```

Figura 14: Función principal de selección de cámaras - web_smart_home.py

Todo el código implementado en esta o cualquier otra sección del documento se puede descargar directamente de mi repositorio en Github: `https://github.com/CesarGOC/ProjectFSE.git`

7.3. Módulo de manipulación de servidor web

El código es bastante extenso, pero en la figura 15 se muestra cómo se obtienen las llamadas al servidor y a través de estas la manipulación de las funciones principales para encender luces, abrir garaje, tocar el timbre, etc.

```
49      def _parse_post(self, json_obj):
50          if not 'action' in json_obj or not 'value' in json_obj:
51              return
52          switcher = {
53              'ledson' : ledson,
54              'ledsoff' : ledsoff,
55              'timbre' : timbre,
56              'garage' : garage,
57              'ledspwm' : pwm,
58              'cam' : camara
59
60          }
61          func = switcher.get(json_obj['action'], None)
62          if func:
63              print('\tCall{}({})'.format(func, json_obj['value']))
64              func(json_obj['value'])
```

Figura 15: Función principal de atendido de llamadas del servidor - web_server.py

También tenemos el apartado de la interfaz de usuario que se manda a llamar con el segmento de código de la figura 16.

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

Figura 16: Función principal para obtener la interfaz de usuario - web_server.py

7.4. Módulo de interfaz de usuario

Al igual que el apartado anterior el código es muy extenso, pero en la figura 17 se muestra la parte especifica donde se implementa el despliegue de los botones para empezar a controlar la Raspberry Pi.

```html
</style>
</head>
<body>
    <header><h1>Panel Smart Home</h1></header>
    <section class="container">
        <article class="column">
            <header><h2>Control iluminacion</h2></header>
            <button class="widebutton" onclick="handle(this, 'ledson', 'On')">Encer Luces</button>
            <button class="widebutton" onclick="handle(this, 'ledsoff', 'Off')">Apagar Luces</button>
            <header><h2>Control timbre</h2></header>
            <button class="widebutton" onclick="handle(this, 'timbre', 'On')">Tocar Timbre</button>
            <header><h2>Control garage</h2></header>
            <button class="widebutton" onclick="handle(this, 'garage', 'Automatic')">Activar sensor del garage</button>
            <button class="widebutton" onclick="handle(this, 'garage', 'Open')">Abrir garage</button>
            <button class="widebutton" onclick="handle(this, 'garage', 'Close')">Cerrar garage</button>
            <header><h2>Control de camaras</h2></header>
            <button class="widebutton" onclick="handle(this, 'cam', '1')">Abrir Camara 1 </button>
            <button class="widebutton" onclick="handle(this, 'cam', '2')">Abrir Camara 2</button>
        </article>
        <article class="column">
            <header><h2>Porcentaje de iluminacion</h2></header>
            <section class="container numpad">
                <button class="numbutton" onclick="handle(this, 'ledspwm', 25)">25%</button>
                <button class="numbutton" onclick="handle(this, 'ledspwm', 50)">50%</button>
                <button class="numbutton" onclick="handle(this, 'ledspwm', 75)">75%</button>
                <button class="numbutton" onclick="handle(this, 'ledspwm', 100)">100%</button>
            </section>
        </article>
    </section>
</body>
</html>
```

Figura 17: Implentación del desplique de botones - user_interface.html

Si abrimos el código user_interface.html en un navegador web podemos observar cómo se verá finalmente la página web cuando pongamos en marcha el programa. Esta vista se muestra en la figura 18.

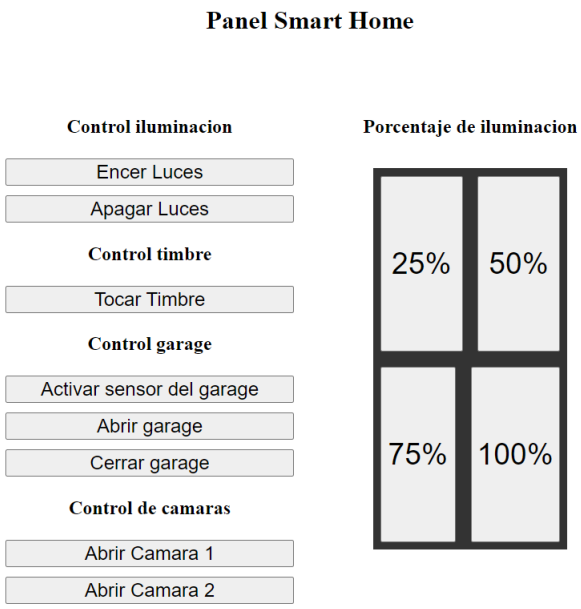

Figura 18: Implentación del desplique de botones - user_interface.html

8. Implementación de componentes electrónicos

En esta parte del documento se visualizarán los diagramas esquemáticos implementados para que funcione el proyecto, se vera de forma individual cada elemento y como es la conexión de este a una Raspberry Pi Model 3 o superior. El número de pin es arbitrario, al final de esta sección se mostraran los pines asociados a la implementación real.

8.1. Botón

El botón tiene dos terminales, una va al voltaje de 3.3 [V] de la Raspberry Pi y el otro a tierra. Para detectar la entrada se conecta un resistor de 220 OHMs entre la entrada de corriente y el cable del pin, la tierra va conectada directamente a la Raspberry Pi. Recordar que esta conexión es opcional, solo se utiliza en la implementación con la maqueta, el botón se usa en las luces y el timbre. En la figura 19 se muestra el esquematico de la conexion de un botón.

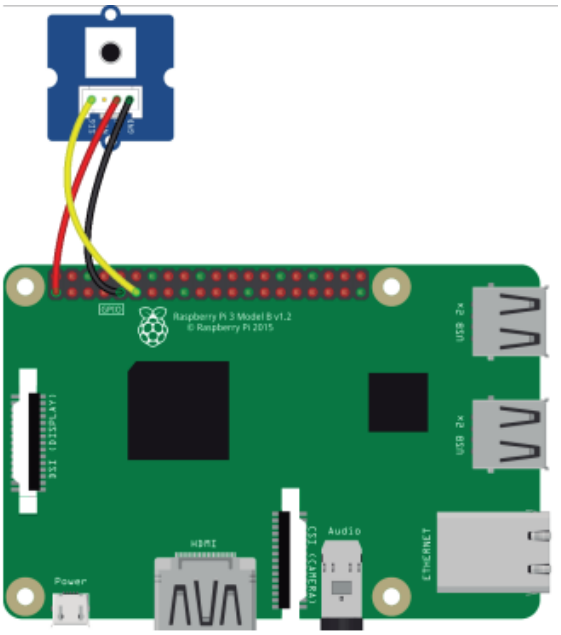

Figura 19: Diagrama esquematico de la conexión del botón

8.2. LED

El LED solo tiene dos conexiones + y - (ánodo y cátodo) la parte del ánodo va conectada al pin de la Raspberrry Pi esta debe tener un resistor de 220 OHMs, si que quieren conectar más LEDs se colocan en paralelo a la conexión principal. En la figura 20 se muestra el esquematico de la conexion de un LED.

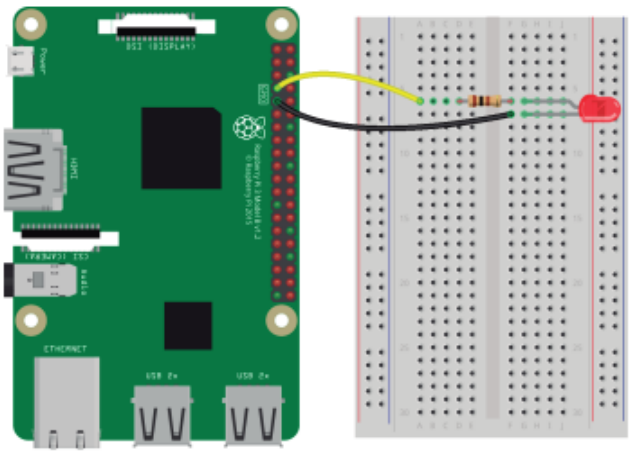

Figura 20: Diagrama esquematico de la conexión de un LED

8.3. Sensor Ultrasónico

El sensor ultrasónico tiene 4 terminales, Tierra, Echo, Triger y Vcc. La entrada de corriente se conecta a la salida de 5[V] de la Raspberry Pi y la tierra va conectada al pin GND de Raspberry Pi. El Triger se conecta a un pin de la Raspberry. En el Echo se hace un divisor de corriente como se muestra en la figura 21 este divisor se hace con los resistores de 10K y 22K OHMs. Después del divisor el Echo se conecta a un pin de la Raspberry Pi.

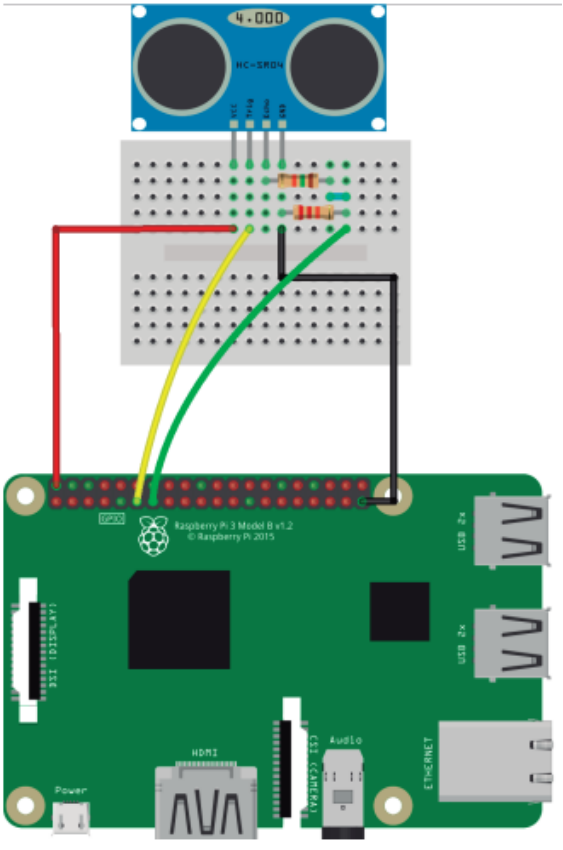

Figura 21: Diagrama esquematico de la conexión de un sensor ultrasónico

8.4. Zumbador

El zumbador o buzzer solo tiene dos terminales + y -, la parte positiva debe conectarse a un pin de la Raspberry Pi y la parte negativa a tierra. La conexión del buzzer se muestra en la figura 22.

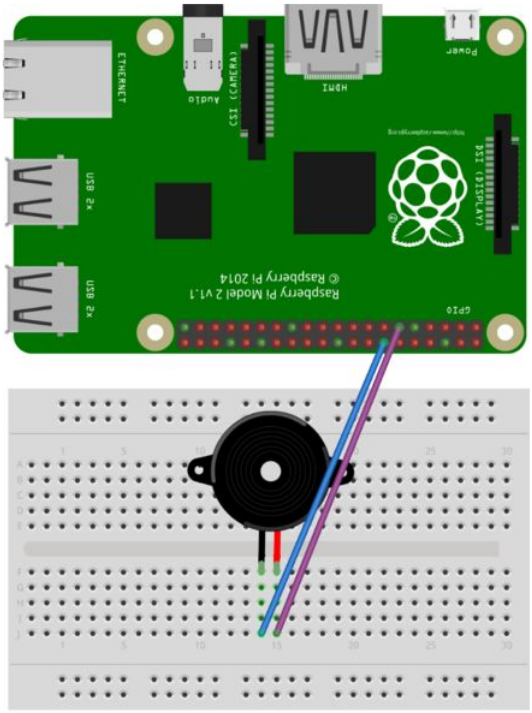

Figura 22: Diagrama esquematico de la conexión de un zumbador

8.5. Servomotor Sg90

Para finalizar esta sección en la figura 23 se muestra el diagrama del servomotor, este tiene tres terminales, Vcc, GND y Pin. La Vcc va conectada de igual forma a la salida de 5 [V] de la Raspberry Pi. La tierra va al GND común y el ultimo cable a un pin de la Raspberry Pi. **Nota: Por lo regular el sistema de colores es el siguiente, VCC-rojo, GND-café y Pin-Naranja o Amarillo.**

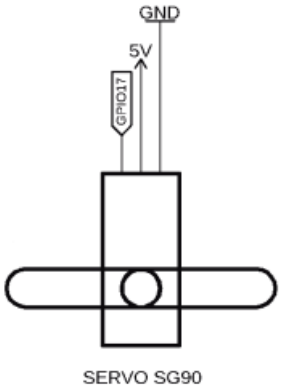

Figura 23: Diagrama de un servomotor convencional

Como añadido extra se implementó un encendido y apagado inteligente de luces externas, estas funcionan a través de un fotoresistor, un transistor y una batería externa. El diagrama esquemático se muestra en la figura 24.

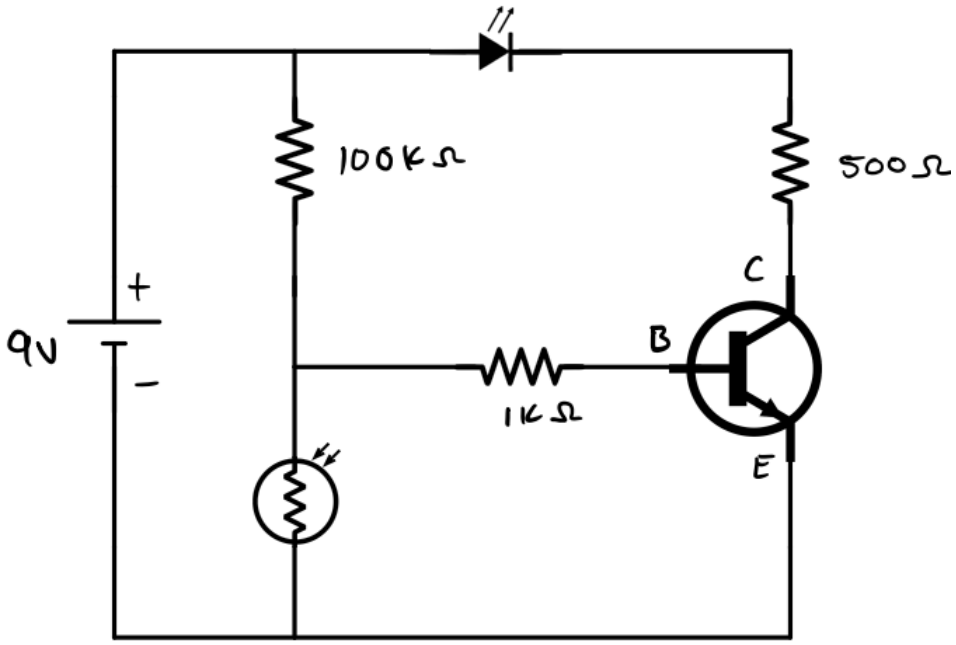

Figura 24: Plano final de la casa

La conexion final de los pines es la siguiente:

- Servomotro Pin GPIO 3.
- Ultrasónico Triger Pin GPIO 5.
- Ultrasónico Echo Pin GPIO 7.
- Botón 1 Pin GPIO 11.
- Botón 2 Pin GPIO 13..
- LEDs Pin GPIO 12.
- Vcc 5[V] Pin GPIO 2.
- Vcc 3.3[V] Pin GPIO 1.
- GND Pin GPIO 9

9. Diagrama de Maqueta (opcional)

Si se decide crear la maqueta por completo en la figura 24 se dejan unos planos para la creación de esta. Este diagrama esta baso en la versión final que se realizó en este proyecto. Tambien en la figura 25 se observa la simbología asociada.

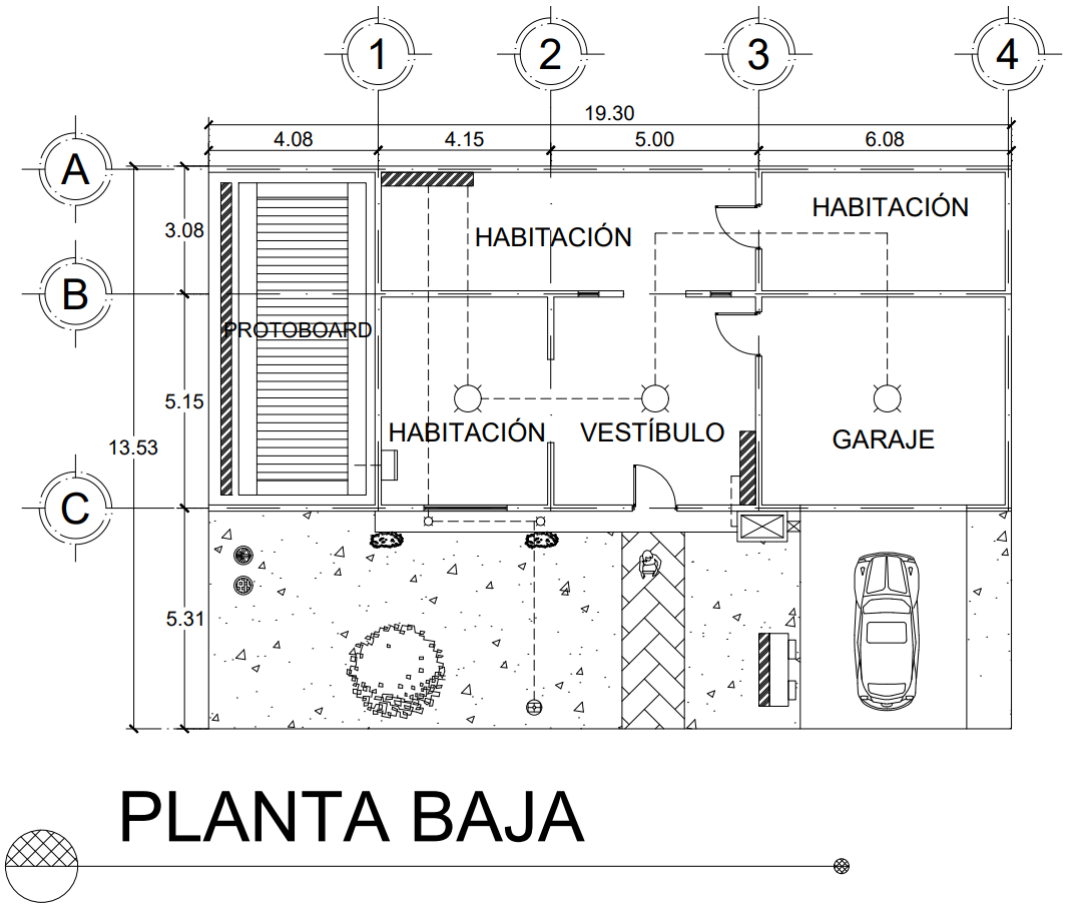

Figura 25: Plano final de la casa

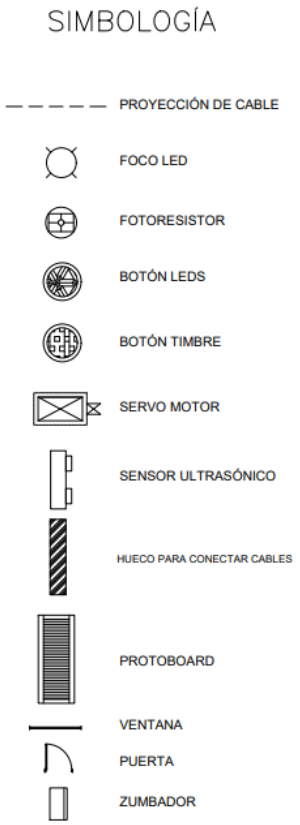

Figura 26: Simbología del plano de la de la casa

Nota: El plano completo se puede encontrar en el sepositorio: `https://github.com/CesarGOC/ProjectFSE.git`

10. Demostración

El video demostrativo del proyecto final se encuentra mediante este link de YouTube:`https://youtu.be/i6VMrMJ1GQI`. En este se explica el funcionamiento, materiales conexion, etc. En la figura 26 se muestra la Raspberry Pi utilizada en este caso es una Raspberry Pi Model 4 con 4GB de RAM.

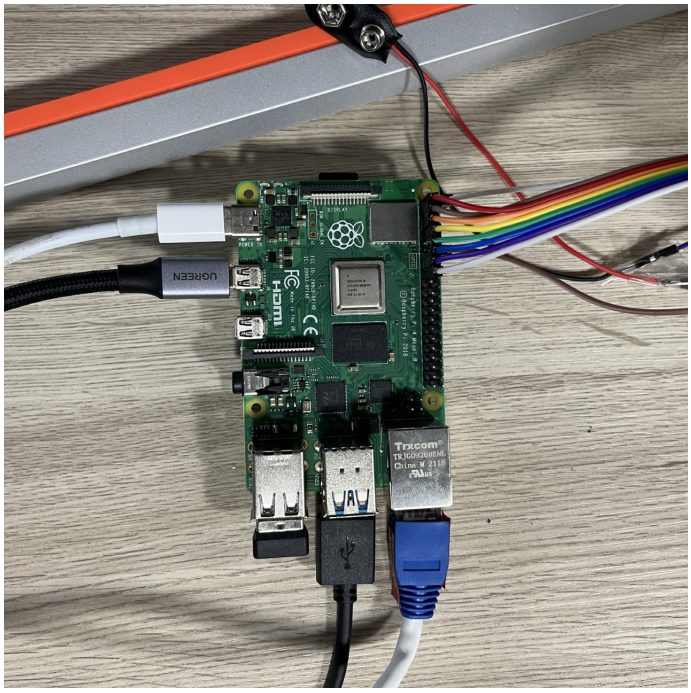

Figura 27: Foto de Raspberry Pi Model 4 utilizada para el proyecto

En la figura 27 se muestran las conexiones de la protoboard a la Raspberry Pi, se utilizo cableado de protoboar por debajo.

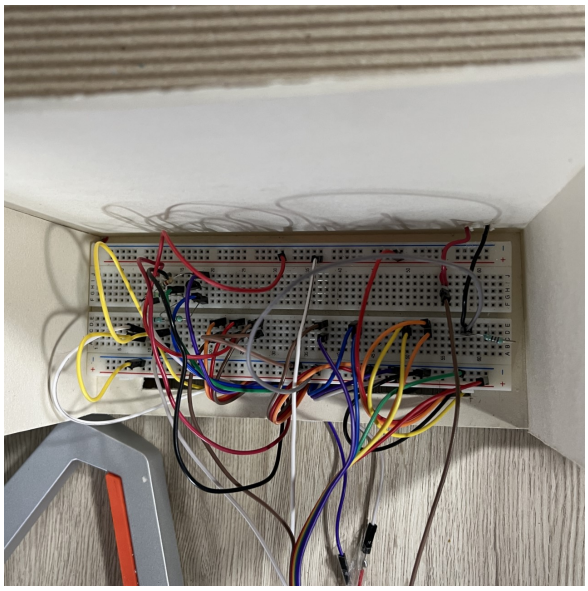

Figura 28: Foto de conexiones de la protoboard a la Raspberry Pi

En la figura 28 se muestran una vista frontal de la maqueta con la implementación terminada.

Figura 29: Foto frontal de la maqueta realizada

En la figura 28 se muestran una vista lateral de la maqueta con la implementación terminada.

Figura 30: Foto lateral de la maqueta realizada

11. Conclusiones

El proyecto fue se finalizo con los recursos mostrados. Se obtuvieron complicaciones por parte de la configuración de acceso inalámbrico y fallas en el sistema operativo instalado en la Raspberry Pi, pero estos problemas se pudieron resolver a la brevedad.

Al finalizar el proyecto se cumplió el objetivo de crear una Smart Home sencilla, aunque se encontraban diferentes formas de realizar las implementaciones de los diferentes dispositivos se realizo de tal forma que fuera más sencillo y barato de desarrollar.

La creación de este tutorial hace que se muestre cada secuencia realizada en este proyecto, mostrando

características, esquemáticos, conexiones y video demostrativo. Donde la persona que lo lea le interese el crear este tipo de proyectos que muestran las capacidades técnicas y tecnológicas que pueden tener los sistemas embebidos

12. Referencias

Matamoros, J. (2022). Facultad de Ingeniería, UNAM, Fundamentos de Sistemas Embebidos, Práctica 4: Control remoto de la Raspberry Pi via WiFi y servidor web.

Chhabra A. (2018). Life is On. Push buttons and their widely used applications across industries. Recuperado de https://blog.se.com/access-to-energy/2018/07/13/push-buttons-and-their-widely-used-applicati

Electrotec. (2022). El resistor. Recuperado de https://electrotec.pe/blog/resistor

BLED. (2022). Características y Ventajas de los Diodos LED. Recuperado de https://www.barcelonaled.com/blog/informacion-led/caracteristicas-y-ventajas-de-los-diodos-led/

Arencibia, J. ARDUINO: Sensor ultrasónico HC-SR04. Recuperado de https://www3.gobiernodecanarias.org/medusa/ecoblog/fsancac/2018/02/06/arduino-sensor-ultrasonico-hc-sr04/

Universidad Autónoma del Estado de Hidalgo. (2021). Arduino, Buzzer. Recuperado de http://ceca.uaeh.edu.mx/informatica/oas_final/OA4/buzzer.html

PCBOARD.ca. (2022). SG90 Micro Servo. Recuperado de https://www.pcboard.ca/sg90-servo

Valverde, J. (2022). ¿Que es un fotoresistor LDR?. Recuperado de https://jesusvalverdesite.wordpress.com/2017/03/07/que-es-un-fotoresistor-ldr/

Rambal Automotización y Robótica. (2022). Transistor 2N2222A NPN TO-92. Recuperdo de https://rambal.com/componentes/484-transistor-2n2222a-npn-to-92.html

Todo sobre los GPIO Raspberry Pi. Recuperadodehttps://www.comohacer.eu/gpio-raspberry-pi/

Raspberry Pi (2022). Recuperado de https://www.raspberrypi.com/

Raspberry Pi Org (2022). Recuperado de https://www.raspberrypi.org/