

TECNICATURA SUPERIOR EN

Telecomunicaciones



SENSORES Y ACTUADORES

TRABAJO PRACTICO N° ----

2023

ACTIVIDAD: TAREA FINAL

INTEGRANTES:

- ALE, Ulises
- BAREA, Silvana
- ORSILI, José



Grupo nº 1



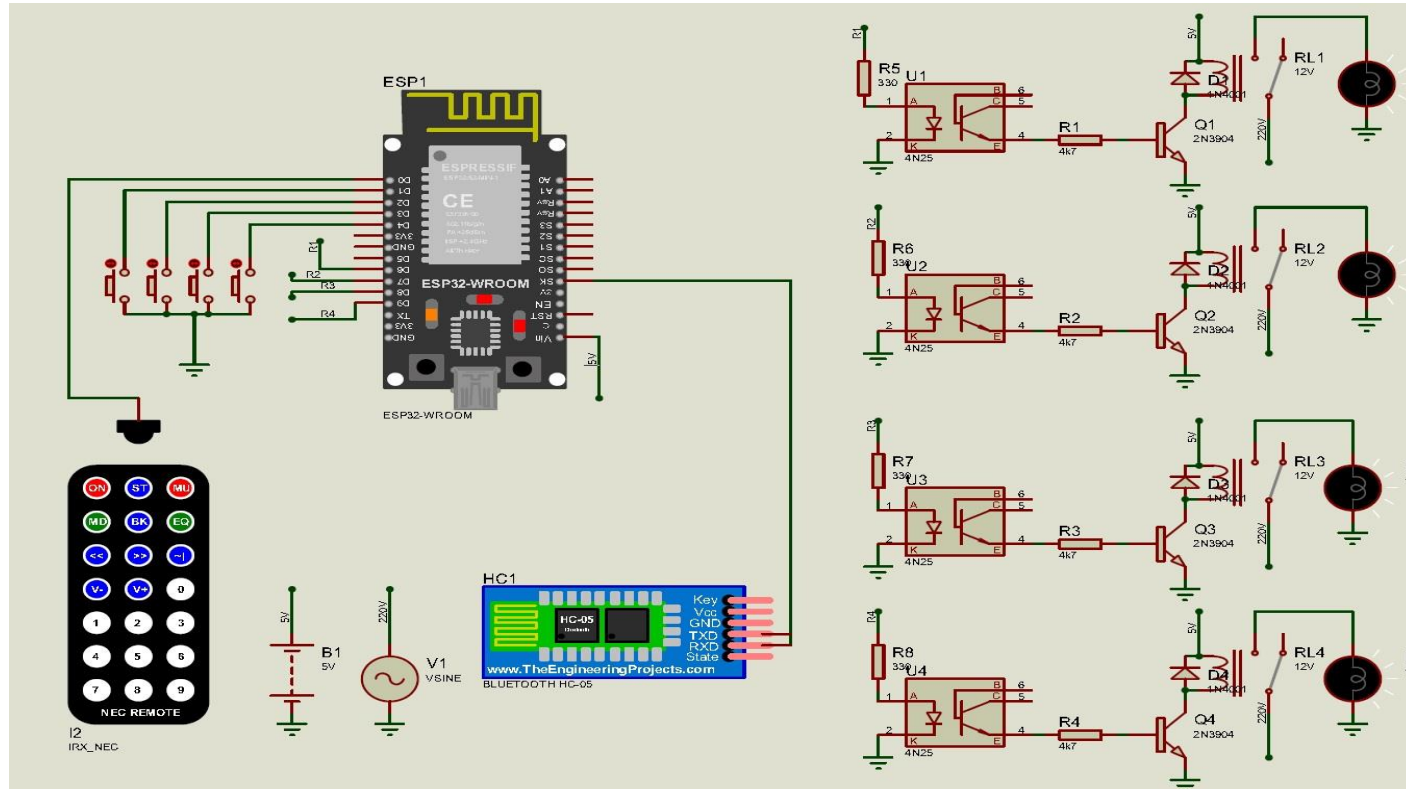
Domotización de una casa IoT, combinando Google, Alexa, Bluetooth, control remoto por Infrarrojos e interruptores manuales.

Descripción: Se implementa un Sistema IoT de Smart House, usando el ESP32. En este proyecto se definirá y creará un sistema domótico ESP RainMaker con Google Assistant, Alexa , Bluetooth, control remoto por infrarrojos e interruptores manuales . Se pueden controlar los relés y monitorear las lecturas de los sensores en Google Home y la aplicación Amazon Alexa desde cualquier parte del mundo. Sin WiFi, se pueden controlar los relés con Bluetooth, control remoto por infrarrojos e interruptores manuales. El ESP32 se conectará automáticamente con el Wi-Fi si el Wi-Fi está disponible.



Domotización de una casa IoT, combinando Google, Alexa, Bluetooth, control remoto por Infrarrojos e interruptores manuales.

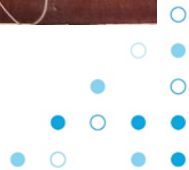
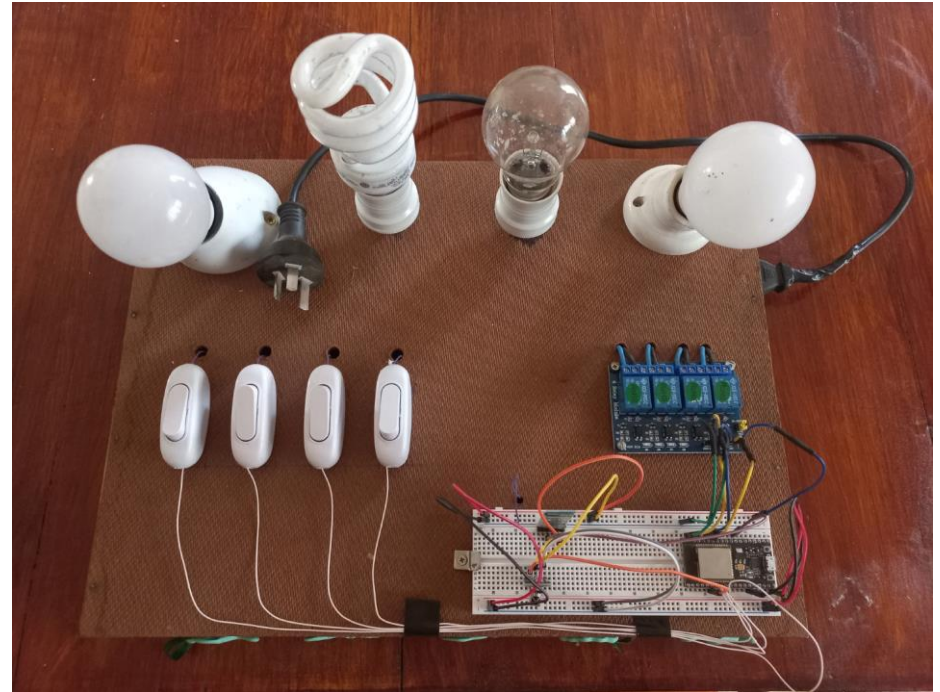
Realización del circuito utilizando Proteus



Domotización de una casa IoT, combinando Google, Alexa, Bluetooth, control remoto por Infrarrojos e interruptores manuales.

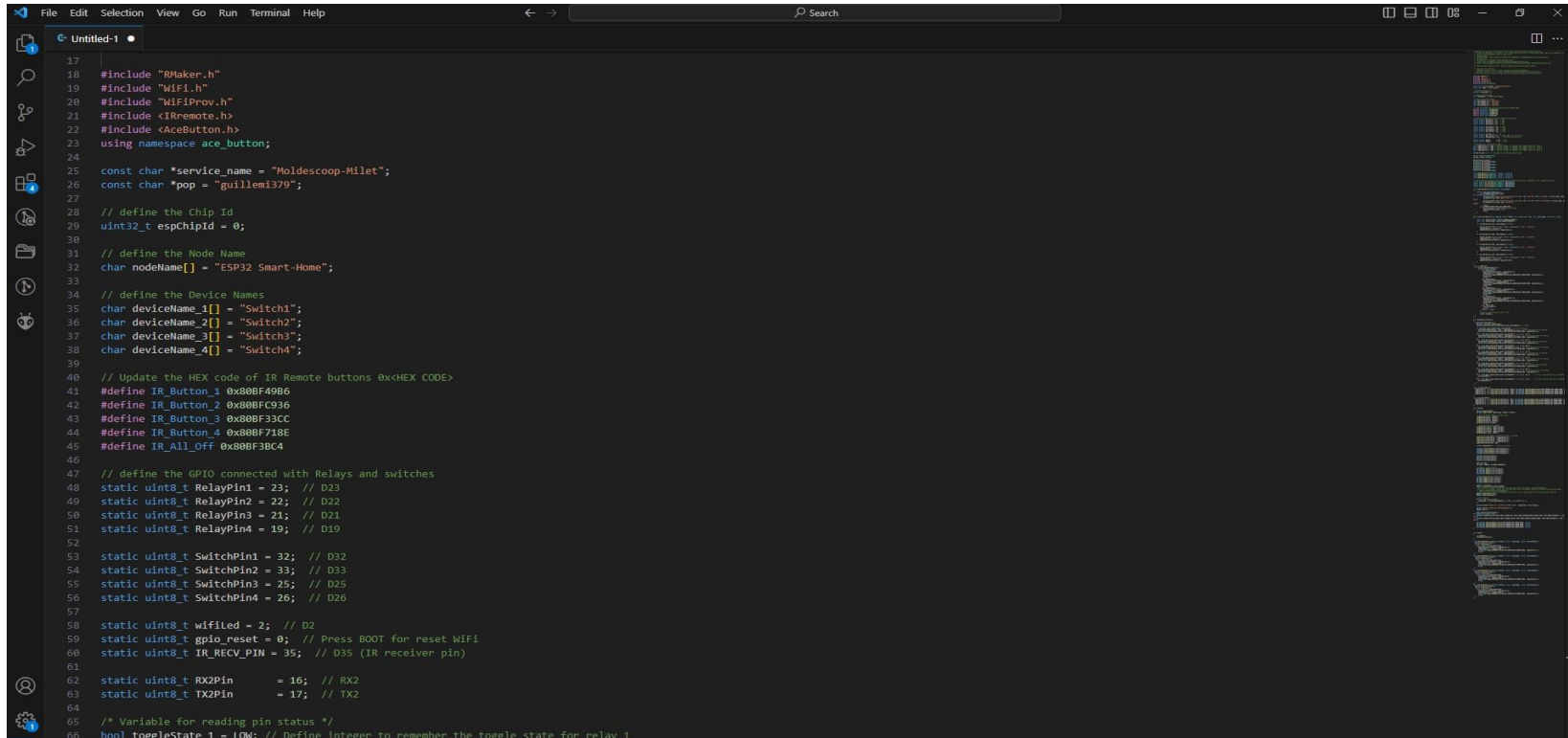
Selección de los componentes y armado del prototipo

- **Esp32 DvKit V1**
- **Celular**
- **Modulo rele 5v. 4 canales**
- **4 Focos**
- **Receptor IR 1838**
- **Modulo Bluetooth**
- **Control remoto**
- **Interruptores manuales**
- **Placa de desarrollo**
- **Cables**



Domotización de una casa IoT, combinando Google, Alexa, Bluetooth, control remoto por Infrarrojos e interruptores manuales.

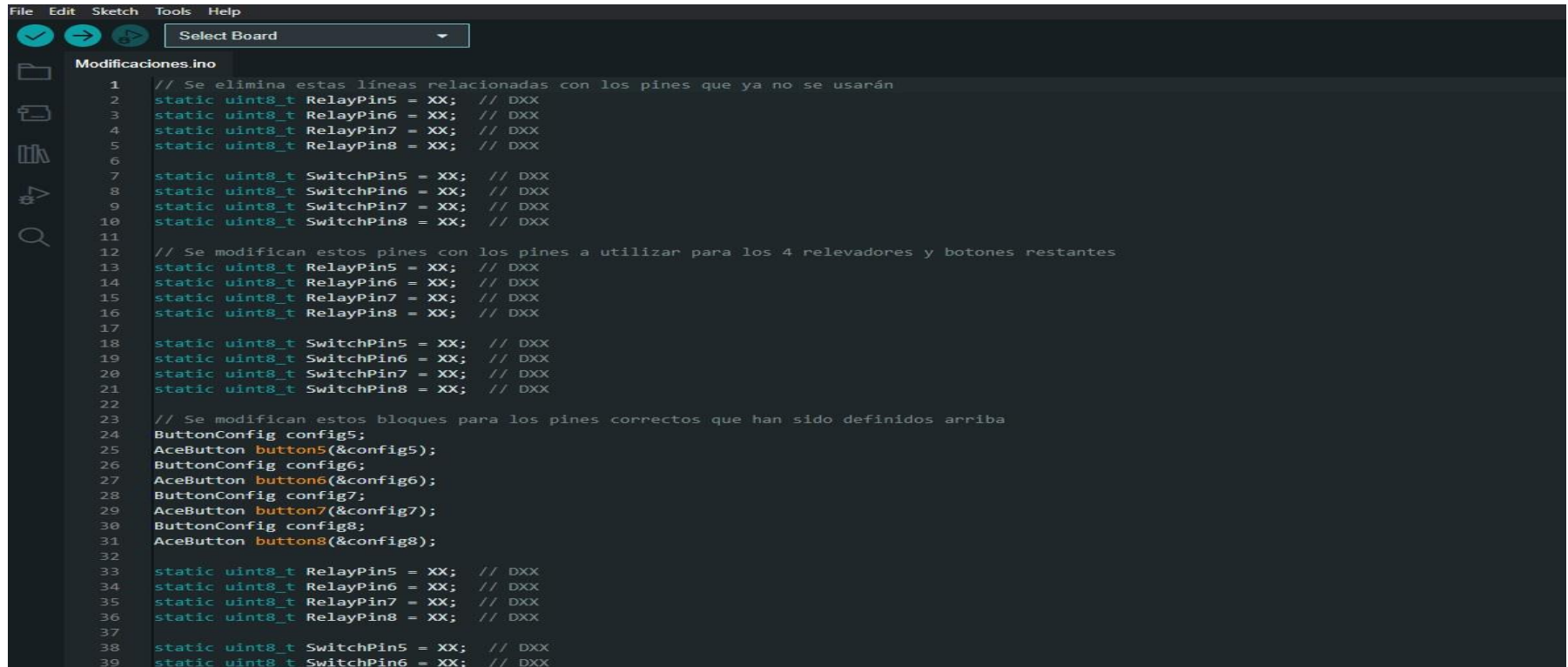
Codificación del programa

A screenshot of a code editor window titled 'Untitled-1'. The code is written in C++ and is for an ESP32-based IoT project. It includes headers for 'RMaker.h', 'Wifi.h', 'WifiProv.h', 'IRRemote.h', and 'AceButton.h'. It defines a service name 'Holdescoop-Milet' and a user 'guillemi379'. It sets up GPIO pins for four relays (D23, D22, D21, D19) and four switches (D32, D33, D25, D26). It also defines pins for WiFi (D2), a reset button (D0), an IR receiver (D35), and serial communication (D16, D17). The code uses the 'AceButton' library to handle IR remote controls and the 'Button' library for physical switches. It includes a toggle state variable for a relay.

```
17 #include "RMaker.h"
18 #include "Wifi.h"
19 #include "WifiProv.h"
20 #include <IRRemote.h>
21 #include <AceButton.h>
22 using namespace ace_button;
23
24
25 const char *service_name = "Holdescoop-Milet";
26 const char *pop = "guillemi379";
27
28 // define the Chip Id
29 uint32_t espChipId = 0;
30
31 // define the Node Name
32 char nodeName[] = "ESP32 Smart-Home";
33
34 // define the Device Names
35 char deviceName_1[] = "Switch1";
36 char deviceName_2[] = "Switch2";
37 char deviceName_3[] = "Switch3";
38 char deviceName_4[] = "Switch4";
39
40 // Update the HEX code of IR Remote buttons 0x<HEX CODE>
41 #define IR_Button_1 0x00BF49B6
42 #define IR_Button_2 0x00BFC936
43 #define IR_Button_3 0x00BF33CC
44 #define IR_Button_4 0x00BF71BE
45 #define IR_All_Off 0x00BF3BC4
46
47 // define the GPIO connected with Relays and switches
48 static uint8_t RelayPin1 = 23; // D23
49 static uint8_t RelayPin2 = 22; // D22
50 static uint8_t RelayPin3 = 21; // D21
51 static uint8_t RelayPin4 = 19; // D19
52
53 static uint8_t SwitchPin1 = 32; // D32
54 static uint8_t SwitchPin2 = 33; // D33
55 static uint8_t SwitchPin3 = 25; // D25
56 static uint8_t SwitchPin4 = 26; // D26
57
58 static uint8_t wifiled = 2; // D2
59 static uint8_t gpio_reset = 0; // Press BOOT for reset Wifi
60 static uint8_t IR_RECV_PIN = 35; // D35 (IR receiver pin)
61
62 static uint8_t RX2Pin = 16; // RX2
63 static uint8_t TX2Pin = 17; // TX2
64
65 /* Variable for reading pin status */
66 bool toggleState 1 = LOW; // Define integer to remember the toggle state for relay 1
```

Domotización de una casa IoT, combinando Google, Alexa, Bluetooth, control remoto por Infrarrojos e interruptores manuales.

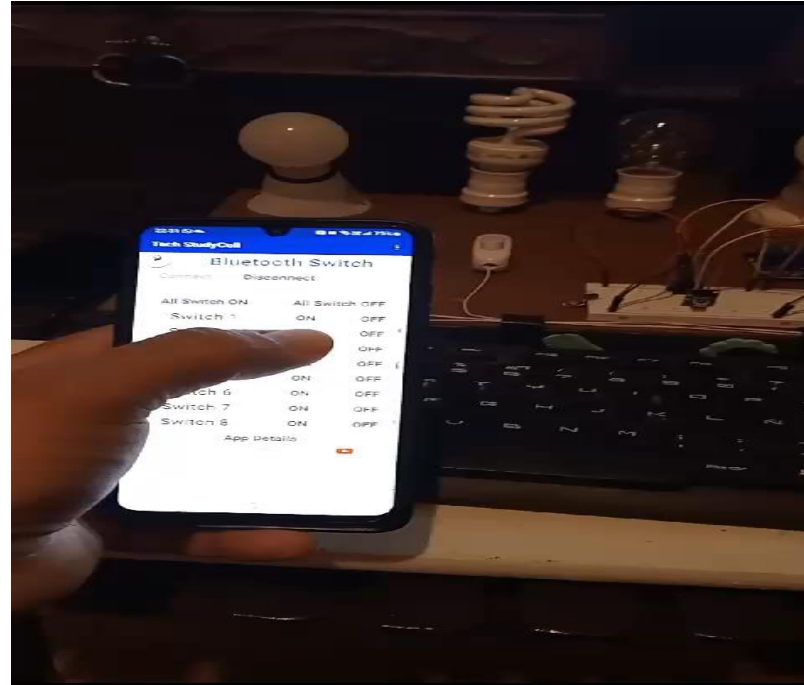
- Reformas en el código existente para adaptarlo a 4 pulsadores y 4 actuadores (relay)



```
1 // Se elimina estas líneas relacionadas con los pines que ya no se usarán
2 static uint8_t RelayPin5 = XX; // DXX
3 static uint8_t RelayPin6 = XX; // DXX
4 static uint8_t RelayPin7 = XX; // DXX
5 static uint8_t RelayPin8 = XX; // DXX
6
7 static uint8_t SwitchPin5 = XX; // DXX
8 static uint8_t SwitchPin6 = XX; // DXX
9 static uint8_t SwitchPin7 = XX; // DXX
10 static uint8_t SwitchPin8 = XX; // DXX
11
12 // Se modifican estos pines con los pines a utilizar para los 4 relevadores y botones restantes
13 static uint8_t RelayPin5 = XX; // DXX
14 static uint8_t RelayPin6 = XX; // DXX
15 static uint8_t RelayPin7 = XX; // DXX
16 static uint8_t RelayPin8 = XX; // DXX
17
18 static uint8_t SwitchPin5 = XX; // DXX
19 static uint8_t SwitchPin6 = XX; // DXX
20 static uint8_t SwitchPin7 = XX; // DXX
21 static uint8_t SwitchPin8 = XX; // DXX
22
23 // Se modifican estos bloques para los pines correctos que han sido definidos arriba
24 ButtonConfig config5;
25 AceButton button5(&config5);
26 ButtonConfig config6;
27 AceButton button6(&config6);
28 ButtonConfig config7;
29 AceButton button7(&config7);
30 ButtonConfig config8;
31 AceButton button8(&config8);
32
33 static uint8_t RelayPin5 = XX; // DXX
34 static uint8_t RelayPin6 = XX; // DXX
35 static uint8_t RelayPin7 = XX; // DXX
36 static uint8_t RelayPin8 = XX; // DXX
37
38 static uint8_t SwitchPin5 = XX; // DXX
39 static uint8_t SwitchPin6 = XX; // DXX
```


Domotización de una casa IoT, combinando Google, Alexa, Bluetooth, control remoto por Infrarrojos e interruptores manuales.

Video de Presentación



Domotización de una casa IoT, combinando Google, Alexa, Bluetooth, control remoto por Infrarrojos e interruptores manuales.

Conclusiones de la Experiencia

Problemas encontrados

- Dificultad en conseguir una placa experimental y soldador.
- El código original estaba diseñado para actuar con 8 dispositivos.
- Detectamos la ausencia de protección ante cortocircuitos en el diseño original.
- El software no respondía al control por voz.

Soluciones aplicadas

- Se utilizó protoboard.
- Se realizó la reforma del código eliminando partes no utilizadas adaptándolo a 4 dispositivos.
- Incorporamos un fusible en serie con la alimentación del sector de 220 voltios.
- Se habilitaron los permisos de la aplicación en Android del celular.





¡Muchas gracias!