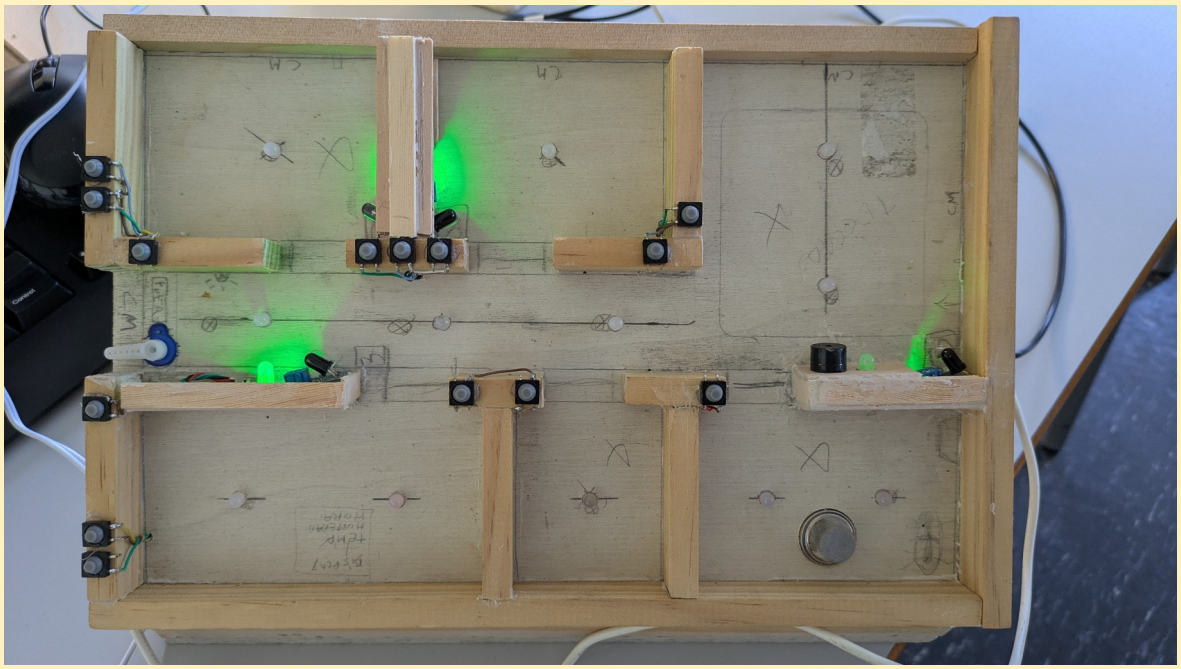
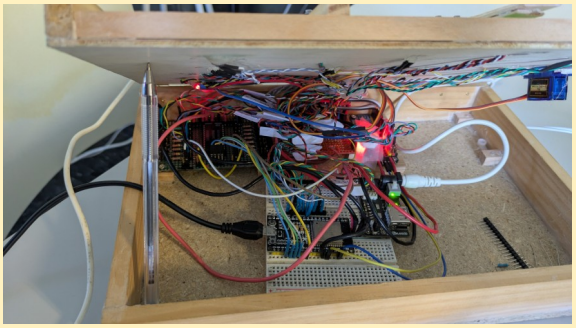
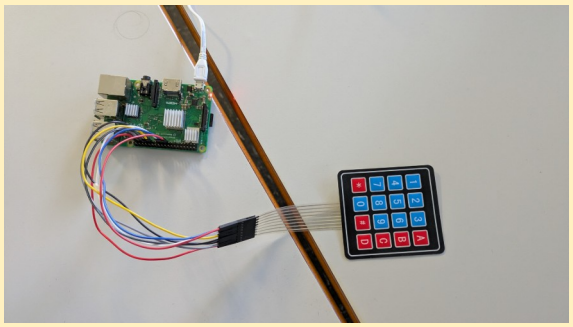
VIVIENDA DOMÓTICA







CIFO – LA VIOLETA

DICIEMBRE - 2024

Juan Francisco Ruiz
Cristian Dalmau
Marc Amor
Jose Antonio Aguilar

INDICE

DESCRIPCIÓN DEL PROYECTO

Este proyecto consiste en el desarrollo de una casa domótica Fig. 1, que permite gestionar diversos automatismos mediante pulsadores o un menú en pantalla (inicialmente serán visualizados en el monitor de un ordenador). El control se realiza a través de un teclado matricial de 16 teclas.

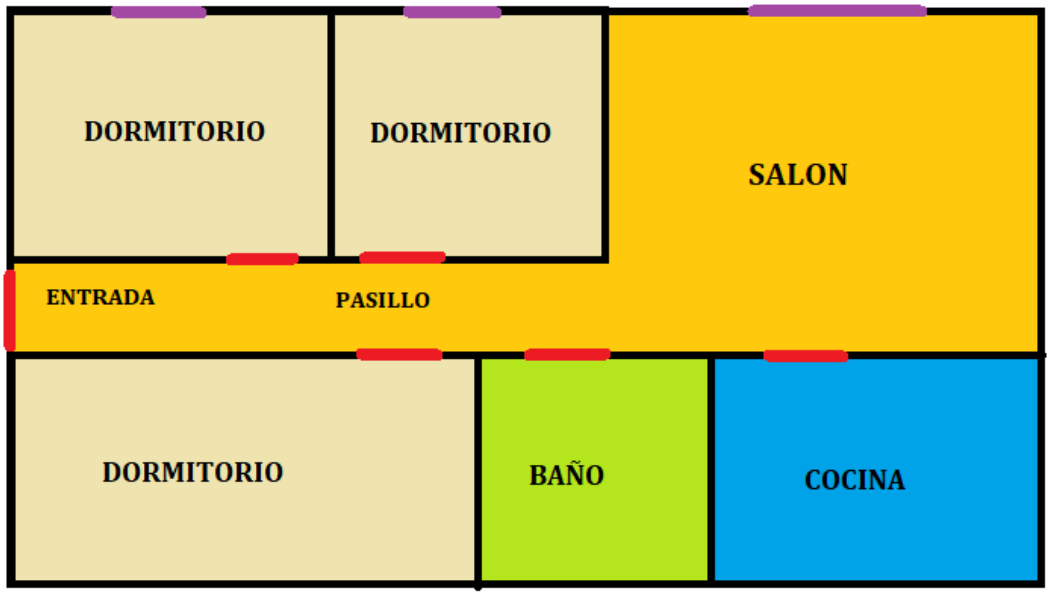


Fig. 1

Automatismos incluidos

1. **Luces**
   - Gestión de la iluminación en distintas estancias de la vivienda, como dormitorios, pasillos, cocina y baño.
2. **Puerta de entrada**
   - Apertura y cierre automatizados mediante un servo motor.
   - Incluye un sistema de detección de presencia que cierra la puerta automáticamente cuando no detecta personas.
3. **Ventanas**
   - Apertura y cierre automatizados mediante servo motores.

Sistema de alarma

El sistema de alarma cuenta con dos modos de armado y un sensor de gas para aumentar la seguridad.

1. **Modo Perimetral**
    - Usa sensores magnéticos “reed switches” (Fig. 2) instalados en las ventanas y en la puerta principal.
    - En caso de detección de intrusión, se activará una alarma sonora.

Fig. 2

2. **Modo General**
    - Combina sensores infrarrojos (IR) (Fig. 3) con los sensores magnéticos del modo perimetral.
    - Detecta movimiento dentro de la vivienda y activa la alarma sonora en caso de intrusión.

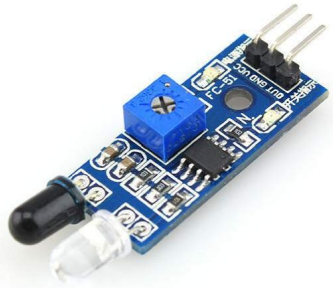


Fig. 3

3. **Alarma de gas**
    - Un sensor de gas (Fig. 4) ubicado en la cocina detecta posibles fugas.
    - Al activarse, emite una alarma sonora y pone en funcionamiento un extractor de humos.

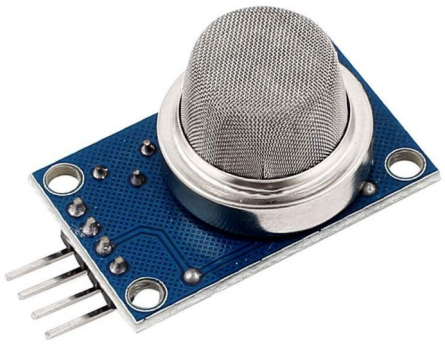


Fig. 4

Indicaciones adicionales del sistema de alarma (Fig. 5)

- Los sonidos de la alarma varían según el evento: armado, desarmado, detección de intrusión o alerta por gas.
- Un LED RGB indica el estado del sistema, mostrando diferentes colores según corresponda.

Fig. 5

MATERIAL NECESARIO PARA EL PROYECTO

Componentes electrónicos

- **LEDs blancos**: 12 unidades
- **LED RGB para alarma**: 1 unidad
- **LED RGB para puerta**: 1 unidad
- **Pulsadores**: 20 unidades
- **Servo motores**: 4 unidades
- **Sensores infrarrojos (IR)**: 4 unidades
- **Sensor de gas**: 1 unidad
- **Sensores magnéticos (reed switches)**: 4 unidades
- **Buzzer pasivo**: 1 unidad
- **PWM I2C de 16 canales**: 1 unidad
- **Placa multiplexor HEF4067B**: 1 unidad
- **Placa amplificadora para LEDs**: 1 unidad
- **Raspberry Pi 3**: 1 unidad
- **ESP32**: 1 unidad

Sistemas de alimentación

- **Alimentador MB-102**: 1 unidad
- **Alimentador 12V/1A:** 1 unidad
- **Alimentador Raspberry:** 1 unidad

Otros componentes

- **Extractor de gas**: 1 unidad
- **Cables**: Cantidad no determinada (ND)
- **Madera**: Cantidad no determinada (ND)

Bitácora del Proyecto

Día 1 y 2 (29/10/2024)

Actividades realizadas:

- Investigación del funcionamiento de la Raspberry Pi Pico.
- Exploración del código MicroPython y su compatibilidad con Python.
- Investigación sobre el multiplexor MCP23017.
- Búsquedas de información sobre estos temas, sin éxito.

Conclusión: Debido a la complejidad de estas tecnologías, se decidió abandonar esta vía por requerir demasiado tiempo de investigación y pruebas.

Progreso:

- Creación de las primeras clases del código: `Clase_LED` y `Clase_Pulsador`.

Día 3

Actividades realizadas:

- Investigación sobre la comunicación entre la Raspberry Pi y la Raspberry Pi Pico.
- Discusión grupal sobre un cambio de enfoque para simplificar el proyecto.
- Cálculo de los pines necesarios para los diferentes Shields y sensores.
- Creación de un plano para la vivienda a domotizar.

Conclusión: Se decidió desarrollar inicialmente un proyecto menos ambicioso enfocado en la Raspberry Pi, con la posibilidad de conectar otras placas en el futuro.

Día 4

Actividades realizadas:

- Investigación sobre librerías para servos y prueba de su funcionamiento.
- Determinación de limitaciones en servos convencionales: no es posible leer la posición sin un cuarto pin.
- Actualización de la `Clase_LED` para guardar y restaurar la intensidad de forma teórica.
- Mejora de la `Clase_Pulsador` para detectar pulsaciones prolongadas.
- Creación de la `Clase_MQ2` para detección de partículas digitales (detección invertida: HIGH en reposo, LOW cuando detecta).

Día 5

Actividades realizadas:

- Subida de los datasheets utilizados.
- Investigación sobre librerías I2C y el controlador LED PCA9685.
- Resolución de problemas para instalar librerías mediante un entorno virtual de Python.
- Instalación exitosa de las librerías necesarias y prueba del display 20x4.

Día 6

Actividades realizadas:

- Investigación sobre cómo posicionar datos en el display sin centrado automático.
- Experimentación con LEDs y el controlador PCA9685 (PWM).
- Implementación de un menú en el display.
- Actualización de la `Clase_Pulsador` para detectar pulsaciones únicas.
- Ajuste de la `Clase_LED` para funcionar correctamente con intensidades.

Día 7

Actividades realizadas:

- Solución de problemas en el código del display debido a cambios en librerías.
- Investigación sobre el almacenamiento de códigos con librerías en la Raspberry Pi Pico.
- Creación de la `Clase_FC51` para detección de infrarrojos (detección invertida).

Día 8

Actividades realizadas:

- Evaluación de la utilidad de la Raspberry Pi Pico por falta de tiempo para su aprendizaje.
- Desarrollo de la `Clase_Teclat4x4` para detectar teclas como una matriz.
- Adaptación del teclado a una matriz 3x3.
- Inicio del soldado de la placa para los LEDs de la maqueta.

Día 9

Actividades realizadas:

- Finalización de la placa de alimentación de los LEDs.
- Estudio y diseño de la placa para el multiplexor HEF4067B.
- Creación de versiones en MicroPython de las clases necesarias.

Día 10

Actividades realizadas:

- Problemas con la placa utilizada: tamaño pequeño, dificultad para soldar y pads oxidados.
- Cambio a una placa más grande de fibra de vidrio.
- Modificación de la `Clase_PCA9685` para usarla como controlador de intensidad para LEDs, descartando la `Clase_LED`.

Día 11

Actividades realizadas:

- Inicio de la nueva placa y su soldado.
- Creación de una segunda placa idéntica.
- Implementación de una función en la `Clase_Pulsador` para distinguir entre pulsaciones cortas (valor único) y largas (valor continuo).

Día 12

Actividades realizadas:

- Finalización de las dos placas.
- Mejora de la `Clase_PCA9685` para ajustar la intensidad de LEDs con pulsaciones cortas y largas.
- Resolución de problemas con un buzzer: creación de diccionarios de frecuencias y melodías predefinidas.

---

0. Buzzer Passiu

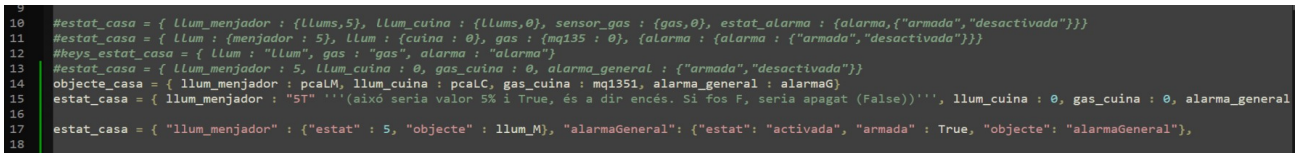
- Problemes amb els tons (notes), s’han eliminat les notes que donen errors per sortida de rang.
- S’ha creat un diccionari amb notes preestablertes amb les seues freqüències i un diccionari on es recullen les melodies preestablertes amb les notes de la melodia.

1. Comunicació entre Raspberry Pi 3 i Raspberry Pi Pico

- Es va iniciar la comunicació amb SPi per la “senzillesa” en la codificació i la velocitat de transmissió. Després de dos dies de feina intensiva no es va aconseguir que funcionés bidireccionalment de cap manera, ni individualment, és a dir, només una de les dues plaques envia a l’altra i l’altra ho llegeix.
- Es decideix canviar a port sèrie TX/RX (UART) perquè, en teoria, és més senzill de configurar/programar. Després d’altres dos dies intensius, s’aconsegueix la bidireccionalitat en diferit, és a dir, que una placa enviï i l’altra rebi. Però no és estable i dóna molts errors.
- Llavors, es decideix recuperar la idea inicial d’usar WIFI i en unes hores s’aconsegueix una comunicació estable.
- Tot i així, dóna errors i es dedica un dia sencer a estabilitzar la comunicació. S’arriba a una possible conclusió que el que passa quan es queda, aparentment, bloquejada la WIFI és que trigui massa en restablir-se la comunicació i sembli que no està funcionant quan s’està restablint automàticament. En tot cas, però, durant la investigació de resolució d’errors, es troba una manera de reiniciar el mòdul hardware mitjançant codi i s’implementa per a assolir una reconnexió molt més ràpida d’uns pocs segons en comptes de temps al voltant del minut.
- Una vegada estabilitzada, es procedeix a la realització de la classe. Com es vol usar una única classe per ambdues plaques, es genera un mòdul detector de placa amb l’ajut de la IA i es procedeix a usar-lo per a realitzar la classe única. Tot i poder ser més eficients fent una classe on ho fes i poder-ho cridar des de qualsevol classe, s’ha fet, en una primera instància, internament a la classe.

2. Diccionaris

- Es realitzarà amb diccionaris la comunicació, enviats en text pla com una string i descodificats i transformats amb regex en diccionaris al destí. Finalment, després d’analitzar diverses maneres de fer els diccionaris, es decideix usar diferents diccionaris: un estàtic on es registrarà la referència a l’objecte de cada element que serà fix un cop s’executi el codi i específic per a cada placa, i un altre diccionari que serà el dinàmic on registrarà l’estat de cada element, els quals seran actualitzats segons les circumstàncies del dia a dia de la casa i serà la part que s’anirà comunicant entre plaques.



Exemple de diverses versions descartades

3. 2024/12/09

- Es realitza l'entrega de la primera release on, per problemes trobats durant la codificació i la manca de temps, la funcionalitat acabada 100% usable és l'encesa i apagat de les llums de la vivenda en remot a través de la Raspberry Pi3 amb un teclat numèric. Els entrebancs més rellevants que ens hem trobat durant la codificació ha estat el tema de l'habilitació de la comunicació. Inicialment es va dedicar molt de temps en la comunicació per cable i posteriorment les dificultats amb l'estabilització de la connexió via WIFI.
- Un gran inconvenient ha estat la manca de prou temps per a la correcta recollida de requeriments i la posada en comú de les idees, seguiment de les tasques segons requeriments i objectius i algú més dedicat a la gestió que organitzés més que codifiqués, vaja, un PO (Product Owner) o un SCRUM master. En trobar-nos les dificultats de les comunicacions entre plaques, ens ha requerit uns 10 dies de dll-dg per a resoldre la tasca quan havíem calculat un parell o tres màxim. Com paral·lelament a aquesta qüestió i d'altres entrebancs trobats i a la construcció de la maqueta, s'ha anat creant codi, una vegada s'ha fet el merge dels diferents codis ens hem trobat els típics problemes d'integració i de deute tècnic. Per a resoldre dits bugs d'integració on el principal problema era la comunicació entre plaques, s'ha partit d'una versió molt senzilla, on només s'enviava i rebia un text del tipus "Hi world" i a partir d'aquí s'ha anat integrant part per part, mòdul per mòdul, línia a línia tot realitzant les proves d'integració pertinents per a assegurar que el codi era 100% funcional, o en el seu defecte, 100% lliure d'errors (la fiabilitat pura és una utopia) per a l'obtenció d'un codi funcional i estable.
- En resum, el codi de la primera release, és capaç de comunicar-se de la Raspberry Pi3 fins al ESP32-WROOM mitjançant un enviament per TX/RX via WIFI d'un diccionari en mode text dels canvis realitzats a la RaspPi3 amb les entrades d'un teclat numèric. Una vegada s'ha enviat i rebut el missatge en text, el ESP32 realitza una conversió del diccionari en format text a format diccionari. Amb aquest diccionari, que correspon als canvis realitzats, o millor dit, demanats/requerits des de la RaspPi3, la ESP32 realitza el canvi requerit. En aquesta entrega seria encendre o apagar els llums de la vivenda. Actualment tindríem un equivalent a un comandament a distància de les llums.

4. Requeriments Futuribles

- Acabar part sensors IR.
- Gestionar error per manca d'alimentació del mòdul PCA.
- Gestionar error per manca d'alimentació del mòdul X.
- Fer redundant el sistema per si hi ha falles i/o un sistema analògic/mecànic per a funcionament manual.

- Sectoritzar i separar les funcions en ESPs32VROOM diferents.
- Corregir (o revisar la totalitat de la lògica) els mètodes o funcions per a que no realitzin accions en la fase de captació de valors de sensors i que aquesta acció es faci en la dels actuadors en funció del valor del diccionari que s'ha actualitzat amb la fase dels sensors.
- Afegir entrada de SSID i PASSWORD a les classes WIFI, que ara mateix estan fixes i no com a paràmetre.
- Els sensors magnètics REED manca ser llegits pel chip multiplexor.
- Els sensors d'InfraRoig IR manca ser llegits pel chip multiplexor.
- Implementar un Display a la Raspberry i/o al ESP32.
- Implementar una web per al maneig dels estats remotament (per l'ESP32 o Raspberry).
- Usar fils d'execució, per exemple en l'enviament un, en el rebut un altre i el programa.
- Un mapa, tkinter e.g., de l'estat de les coses.
- Investigar codis preestablerts dels protocols domòtics: MQTT o ZIGBEE.

CONCLUSIONES

A través de este proyecto, hemos adquirido conocimientos prácticos sobre el uso de la Raspberry Pi, incluyendo prácticas con la Raspberry Pi Pico y la ESP32. Nos enfrentamos y superamos retos como aprender a utilizar MicroPython, introducir librerías mediante la creación de entornos virtuales, y lograr la comunicación entre una Raspberry Pi y una ESP32 a través de WiFi usando el protocolo TCP/IP.

Fue necesario desarrollar librerías específicas para cada sensor y placa utilizada, algunas partiendo desde cero. Este desafío implicó enfrentarse a la incertidumbre de si funcionarían con un código principal basado en diccionarios. Aunque esta estructura simplifica ciertos aspectos, también representa un reto importante por la complejidad añadida.

A pesar de los avances logrados, la falta de tiempo nos impidió ampliar el proyecto hacia funcionalidades más completas, como implementar un display para visualizar datos, controlar ventanas con servomotores, o incluir un extractor, funcionalidades que habrían permitido una experiencia de domótica más integral. La visión final era crear un sistema controlable desde un teléfono móvil, algo que quedó fuera de nuestro alcance en esta ocasión.

Sin embargo, el conocimiento adquirido es invaluable. Todos esperamos que estas experiencias sean útiles no solo en nuestra vida laboral, sino también como base para futuras creaciones y para mantener la pasión por el "Internet de las cosas" y la programación que hemos desarrollado.

Análisis de factores clave del proyecto

1. **Falta de liderazgo:**
    - Al ser un grupo sin jerarquías, aunque motivado, las decisiones fueron tomadas de manera improvisada. Esto llevó a desarrollar soluciones para posibles necesidades no previstas, lo que ocasionó esfuerzos en vano.

2. **Falta de organización:**
    - No se consideraron todas las necesidades del proyecto desde el principio. La maqueta, que debería haber sido el eje del desarrollo como "cliente", fue relegada frente a la programación, que era la fortaleza del curso.

- La falta de planificación entre las partes electrónicas y de programación generó incompatibilidades.

3. **Fallas en la improvisación:**
    - La ausencia de una definición clara del proyecto y sus necesidades resultó en pérdida de tiempo investigando controladores y librerías que no se ajustaban al propósito final.
    - Aunque las librerías individuales fueron creadas y probadas, no siempre resultaron compatibles entre sí.

Reflexión final

A pesar de la falta de organización inicial y del tiempo limitado, el proyecto fue llevado a cabo gracias a la tenacidad, el conocimiento y la experiencia del equipo en programación y electrónica. La ausencia de jerarquías no fue un obstáculo para mantener un ambiente positivo y motivador. Esto demuestra que, con compromiso y un buen ambiente de trabajo, es posible superar incluso los retos más desafiantes.

WEBGRAFÍA

1. Introducción y configuración

- **Iniciación con Raspberry Pi Pico: acceso y guardado de código**
  https://raspberrypi.cl/2022/06/27/raspberry-pi-pico-paso-a-paso/
- **Configuración de I2C**
  - Documentación oficial de SunFounder:
    https://docs.sunfounder.com/projects/davinci-kit-es/es/latest/appendix/i2c_configuration.html
  - Creación de un entorno virtual para I2C:
    https://docs.sunfounder.com/projects/davinci-kit-es/es/latest/appendix/create_virtual_environment.html#create-virtual

2. Programación de componentes

- **Lectura de pulsadores con Raspberry Pi**
  Video explicativo en YouTube:
  https://www.youtube.com/watch?v=YbjDjoQtngM
- **Uso del sensor DHT-11 (Temperatura y Humedad)**
  - Repositorio Adafruit para la biblioteca DHT:
    https://github.com/adafruit/DHT-sensor-library

3. Documentación técnica y datasheets

- **Convertidor de nivel lógico bidireccional de 8 canales (TXB0108)**
  - Ficha técnica (datasheet):
    https://cdn-shop.adafruit.com/datasheets/txb0108.pdf
  - Página del producto en Adafruit:
    https://www.adafruit.com/product/395

4. Conexión de periféricos

- **Conexión de un display LCD 20x4 a Raspberry Pi mediante I2C**
  Tutorial de OpenHardware.pe:
  https://openhardware.pe/contactando-un-display-lcd-de-20x4-a-rasperrt-pi-mediante-i2c-i2c-iii/

ANEXO CÓDIGO

CÓDIGO PRINCIPAL DE LA RASPBERRY PI 3

```
'''
Arxiu anterior:
    1-raspberry_bireccional_WIFI_20241129_BetaPasaPas_20241208.py -> amb autodetect
    2-Pi3_CasaDomotica_PrimeraRelease_v_0.py -> funciona, "primera" versió funcional provada amb èxit
    3-Pi3_CasaDomotica_PrimeraRelease_v_2.py -> corregint errors compatibilitat ESP32WROOM_CasaDomotica_PrimeraRelease_MAIN_v_1.py (ESP32)
    4-Pi3_CasaDomotica_PrimeraRelease_v_2.py ->
'''
'''
Accions en aquest ongoing:
    no se q collons poassa q no va
'''


#import socket
#import time
from classe_WIFI_RaspPI3_Beta import *
import re
from classe_teclat4x4_pi3 import *


        ###### Assignació PINs ######
PINs_fila = [10,9,11,5]
PINs_columna = [6,13,19,26]


        ###### Assignació variables ######
missatge_enviar = '{}'
Tecles = [["Llum_Cuina", "Llum_Lavabo", "Llum_Habitacio_1", "Alarma_Intrusio_Perimetral"], # -> [["1", "2", "3", "A"],
        ["Llum_Menjador", "Llum_Passadis", "Llum_Habitacio_2", "Alarma_Intrusio_Total"], # -> ["4", "5", "6", "B"],
        ["Llum_Habitacio_3", None, None, "Pols_Servo_Obrir"], # -> ["7", "8", "9", "C"],
        [None, "Pols_Timbre", None, "Pols_Servo_Tancar"]] # -> ["*", "0", "#", "D"]]
```

```
    # diccionari dinàmic
estat_objectes_casa = {
            "Llum_Cuina" : "100F", #Pot ser del mínim x% (x) per exemple fins al 100% (100), apagat (F) o encés (T) -> exemple xT 100F
            "Llum_Passadis" : "100F",
            "Llum_Menjador" : "100F",
            "Llum_Lavabo" : "100F",
            "Llum_Habitacio_1" : "100F",
            "Llum_Habitacio_2" : "100F",
            "Llum_Habitacio_3" : "100F",

            "Led_WIFI_activat" : False,
            "Led_WIFI_conectat" : False,
            "Led_WIFI_comunicant" : False, # Blinking

            "Pols_Llum_Cuina" : False, #Poden ser True or False (polsat/no polsat)
            "Pols_Llum_Passadis" : False,
            "Pols_Llum_Menjador" : False,
            "Pols_Llum_Lavabo" : False,
            "Pols_Llum_Habitacio_1" : False,
            "Pols_Llum_Habitacio_2" : False,
            "Pols_Llum_Habitacio_3" : False,
            "Pols_Servo_Obrir" : False,
            "Pols_Servo_Tancar" : False,
            "Pols_Timbre" : False,
            "Pols_Alarma_Perimetral" : False,
            "Pols_Alarma_Total" : False,

            "Proximitat_Passadis" : False, #Poden ser True or False (detectat/no detectat)
            "Proximitat_Menjador" : False,
            "Proximitat_Habitacio_2" : False,
            "Proximitat_Habitacio_3" : False,

            "Reed_PEntrada" : False, #Poden ser True or False (tancat/obert)
```

```
            "Reed_Menjador" : False,
            "Reed_Habitacio_2" : False,
            "Reed_Habitacio_3" : False,

            #"Gas_MQ135" : Gas_MQ135, -> 1

            "Servo_PEntrada" : False, # False està tancada 0, #0-90 graus obertura

            #"Alarma_Gas" : Alarma_Gas, -> 1
            "Alarma_Gas" : False, #Pot ser True or False (detectat/no detectat)
            "Alarma_Intrusio_Perimetral" : "FF", # Armat/activat {False, False} #Pot ser True or False tant l'armat (armat/desarmat) com detecció (detectat/no detectat)
            "Alarma_Intrusio_Total" : "FF" # Armat/activat {False, False} #Pot ser True or False tant l'armat (armat/desarmat) com detecció (detectat/no detectat)
            }

            ###### Creació de funcions ######

def to_diccionari(text):
    # Regex per capturar la clau i el valor
    pattern = r'"(\w+)"\s*:\s*({.*?}|\d+|"[^"]*")'
    matches = re.findall(pattern, text)

    diccionari = {}
    for key, value in matches:
        # Comprovem si el valor és un conjunt (set)
        if value.startswith("{"):
            # Eliminem les cometes i convertim els valors dins de les claus en tipus corresponents
            value = value.strip("{}").split(",")
            diccionari[key] = {val.strip() for val in value}
        elif value.isdigit():  # Si el valor és un número
            diccionari[key] = int(value)
        else:  # Si el valor és una cadena
            diccionari[key] = value.strip("")
```

```
    return diccionari



            ###### Creació objectes ######
COMs = WIFI()
Teclat = Teclat4x4(PINs_fila, PINs_columna, Tecles)

COMs.WIFI_reinicia()

            ###### MAIN ######
try:
    while True:

        # Realitza la comunicació amb l'ESP32
        missatge_rebut = COMs.WIFI_comunicacio(missatge_enviar)
##BORRARRRRRRRRRRRRR
        if missatge_rebut != '{}':
            print(missatge_rebut)

        missatge_enviar = '{}' #en la primera versió, es suposa que a cada tecla s'envia el missatge. pero es possible que en altres versions s'enviin ´es canvis en un enviament i en fils

        # Actualitza diccionari si es rebut algun canvi
        if missatge_rebut != '{}' and missatge_rebut is not None:
            diccionari_rebut = to_diccionari(missatge_rebut)
            claus_novetat = diccionari_rebut.keys()
            for key in claus_novetat:
                estat_objectes_casa[key] = diccionari_rebut[key]

        # Crea el missatge a enviar, si no hi ha canvis, envia text buit
        clau = Teclat.tecla()
        if clau != None: # Són tots Polsadors
#           print(clau)
```

```
            if "Llum" in clau:
#                print("Llum")
                fi = ''
                if estat_objectes_casa[clau][-1] == "F":
                    fi = "T"
                else:
                    fi = "F"
                estat_objectes_casa[clau] = estat_objectes_casa[clau][:-1]+fi
                missatge_enviar = missatge_enviar[:-1]+'"'+clau+f'":
"{estat_objectes_casa[clau]}"'+f'{missatge_enviar[len(missatge_enviar)-1]}'#estat_objectes_casa[clau]})' #en
la primera versió, es suposa que a cada tecla s'envia el missatge. pero es possible que en altres versions
s'enviin ´es canvis en un enviament i en fils
            elif "Intrusio" in clau: # Aquí només armarà o desarmarà l'alarma
#                print("Intrusio")
                ini = ''
                if estat_objectes_casa[clau][0] == "F":
                    ini = "T"
                else:
                    ini = "F"
                estat_objectes_casa[clau] = ini+estat_objectes_casa[clau][-1]
                missatge_enviar = missatge_enviar[:-1]+clau+f':
{estat_objectes_casa[clau]}'+f'{missatge_enviar[len(missatge_enviar)-1]}'
                    #estat_objectes_casa[clau]})'#en la primera versió, es suposa que a cada tecla s'envia el
missatge. pero es possible que en altres versions s'enviin ´es canvis en un enviament i en fils
            else:
#                print("Iogurt")
                estat_objectes_casa[clau] = not estat_objectes_casa[clau]
                missatge_enviar = missatge_enviar[:-1]+clau+f':
{estat_objectes_casa[clau]}'+f'{missatge_enviar[len(missatge_enviar)-1]}'
        #print(missatge_enviar)
        time.sleep(0.01)



        #time.sleep(1)  # Espera 5 segons abans del següent cicle


except socket.timeout:
    print("Timeout esperant connexió.")
```

```
except KeyboardInterrupt:
    #print("Inici aturat per l'usuari")
    #COMs.WIFI_tanca()
    #COMs.WIFI_desconecta()
    #COMs.WIFI_scan()
    print("Aturada manual per l'usuari")
except Exception as e:
    print(f"Error: {e}")

'''
while True:
    try:
        # Com a client: envia missatges a l'ESP32
        print("Connectant com a client al servidor de l'ESP32...")
        client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        client_socket.connect((esp_ip, esp_port))

        message = "Hola ESP"
        client_socket.send(message.encode('utf-8'))
        print(f"Missatge enviat a l'ESP32: {message}")

        client_socket.close()

        # Com a servidor: escolta i rep missatges de l'ESP32
        print("Esperant connexió de l'ESP32...")
        server_socket.settimeout(10)  # Timeout per no bloquejar indefinidament
        conn, addr = server_socket.accept()
        print(f"Connexió establerta amb: {addr}")

        # Rep el missatge
        data = conn.recv(1024).decode('utf-8')
        print(f"Missatge rebut de l'ESP32: {data}")

        conn.close()
```

```
        time.sleep(1)  # Espera 5 segons abans del següent cicle

    except socket.timeout:
        print("Timeout esperant connexió.")
    except Exception as e:
        print(f"Error: {e}")

'''

'''
# diccionaris estàtics
objectes_casa = {
            ###   "Llum_Cuina" : Llum_Cuina_PCA,
            ###   "Llum_Passadis" : Llum_Passadis_PCA,
            ###   "Llum_Menjador" : Llum_Menjador_PCA,
            ###   "Llum_Lavabo" : Llum_Lavabo_PCA,
            ###   "Llum_Habitació_1" : Llum_Habitació_1,
            ###   "Llum_Habitació_2" : Llum_Habitació_2,
            ###   "Llum_Habitació_3" : Llum_Habitació_3,

            ###   "Led_WIFI_activat" : LED_WIFI_act,
            ###    "Led_WIFI_conectat" : LED_WIFI_con,
            ###   "Led_WIFI_comunicant" : LED_WIFI_com,

              "Pols_Llum_Cuina" : Pols_Llum_Cuina_PCA, # Botó teclat 1
              "Pols_Llum_Passadis" : Pols_Llum_Passadis_PCA, # Botó teclat 5
              "Pols_Llum_Menjador" : Pols_Llum_Menjador_PCA, # Botó teclat 4
              "Pols_Llum_Lavabo" : Pols_Llum_Lavabo_PCA, # Botó teclat 2
              "Pols_Llum_Habitacio_1" : Pols_Llum_Habitacio_1, # Botó teclat 3
              "Pols_Llum_Habitacio_2" : Pols_Llum_Habitacio_2, # Botó teclat 6
              "Pols_Llum_Habitacio_3" : Pols_Llum_Habitacio_3, # Botó teclat 7
              "Pols_Servo_Obrir" : Pols_Servo_Obrir, # Botó teclat C
              "Pols_Servo_Tancar" : Pols_Servo_Tancar, # Botó teclat D
              "Pols_Timbre" : Pols_Timbre, # Botó teclat 0
              "Pols_Alarma_Perimetral" : Pols_Alarma_Perimetral, # Botó teclat A
```

```
            "Pols_Alarma_Total" : Pols_Alarma_Total, # Botó teclat B

          ###    "Proximitat_Passadis" : Proximitat_Passadis,
           ###   "Proximitat_Menjador" : Proximitat_Menjador,
           ###   "Proximitat_Habitacio_2" : Proximitat_Habitacio_2,
            ###  "Proximitat_Habitacio_3" : Proximitat_Habitacio_3,

            ###  "Reed_PEntrada" : Reed_PEntrada,
            ###  "Reed_Menjador" : Reed_Menjador,
            ###  "Reed_Habitacio_2" : Reed_Habitacio_2,
            ###  "Reed_Habitacio_3" : Reed_Habitacio_3,

             #"Gas_MQ135" : Gas_MQ135, -> 1

            ### "Servo_PEntrada" : Servo_PEntrada,

             #"Alarma_Gas" : Alarma_Gas, -> 1
            ### "Alarma_Gas" : Gas_MQ135,
            ### "Alarma_Intrusió_Perimetral" : Alarma_Intrusió_Perimetral, # Ha de contenir els sensors reed
            ### "Alarma_Intrusió_Total" : Alarma_Intrusió_Total # Ha de contenir els sensors de proximitat i els
reed
            }
'''
```

AGRADECIMIENTOS

Hemos de agradecer que este proyecto haya sido posible gracias en primer lugar a Joan Masdemont, que aparte de lo que nos enseño en clase, estuvo ahí cuando nos encallamos con el código, y con los retos que suponían por ejemplo comunicar la raspberry con otra placa.

También debemos agradecer al compañero Pere Martín Moraleja, que gracias a él, me confirmo que era necesario crear un entorno virtual para poder introducir librerías en el python de la raspberry pi.

Como no, mención especial a ChatGPT “chati” para los compañeros de clase. Que nos ayudo con la programación. Y en menor medida a Copilot que día a día sigue mejorando para poder igualarse a ChatGPT.

MAS INFORMACION

QR GITHUB

QR VIDEO PROYECTO

https://youtu.be/IMQz5hNNqFs

https://youtu.be/soCY5ggWark