UNIVERSIDAD DE ORIENTE

NÚCLEO DE ANZOÁTEGUI

ESCUELA DE INGENIERÍA Y CIENCIAS APLICADAS

DEPARTAMENTO DE COMPUTACIÓN Y SISTEMAS

PROYECTOS DIGITALES AVANZADOS

PROYECTO FINAL:

SISTEMA DE SEGURIDAD CON NOTIFICACIONES AL CORREO

Profesor:
Pedro Rene
Cabrera

Bachiller:
Claudia Rodríguez
C.I: 27.943.668

Barcelona, febrero de 2025

1. OBJETIVOS DEL PROYECTO

1.1. Objetivo Principal: Desarrollar un sistema de seguridad basado en proximidad utilizando el Raspberry Pi Pico W, que detecte objetos cercanos y genere alertas en tiempo real vía email, mejorando la seguridad mediante monitoreo y notificaciones automatizadas.

1.2. Objetivos Específicos.

- **1.2.1.** Aplicar conocimientos adquiridos a lo largo del semestre en el desarrollo del sistema.
- **1.2.2.** Implementar conectividad WiFi, permitiendo el escaneo de redes disponibles y poder escoger otra red de WiFi en caso de que la escogida no esté disponible
- **1.2.3.** Registrar eventos de detección en un archivo de texto para mantener un historial de actividad.
- **1.2.4.** Integrar y configurar componentes electrónicos como el sensor ultrasónico, keypad matricial, pantalla OLED y un LED indicador para el monitoreo.
- **1.2.5.** Mantener el sistema de alarma alerta cuando se detecte un sospechoso en un rango de 0 a 30cm y almacenar estas muestras en una lista.
- **1.2.6.** Incorporar autenticación mediante pin de seguridad, utilizando el keypad matricial, para que no envíe la alarma en caso de ser el propietario.
- **1.2.7.** Enviar notificaciones por correo en caso de intentos fallidos de autenticación o detección de actividad sospechosa.
- **1.2.8.** Implementación de un mecanismo para evitar falsas alarmas, comparando la muestra anterior con la actual para verificar si el sujeto sospechoso se ha alejado.
- **1.2.9.** Almacenar eventos relevantes en el registro de actividad para seguimiento posterior.
- **1.2.10.** Permitir la interacción del usuario mediante opciones para continuar o detener el monitoreo según sea necesario utilizando el keypad.

2. DESARROLLO

2.1. Planteamiento del Problema

En la actualidad, la seguridad es un aspecto fundamental en la vida cotidiana debido al riesgo constante de intrusos que pueden ingresar a edificios o residencias. Si bien existen múltiples soluciones tecnológicas para disminuir estos riesgos, los sistemas de seguridad tradicionales suelen ser costosos, complejos de instalar y difíciles de mantener. Además, en muchos casos, no están diseñados con un enfoque centrado en la experiencia del usuario, lo que limita su accesibilidad y facilidad de uso.

El avance en tecnologías como los sensores ultrasónicos, las pantallas OLED y la conectividad mediante redes WiFi (IoT) ha permitido el desarrollo de soluciones más accesibles, eficientes y personalizables. No obstante, muchos sistemas de seguridad IoT presentan deficiencias en autenticación y control de acceso, lo que puede derivar en notificaciones erróneas o falsas alarmas. La falta de mecanismos que permitan verificar si la persona que interactúa con el dispositivo es el propietario o un intruso representa un desafío importante en la confiabilidad de estos sistemas.

Este proyecto final para la materia de **PROYECTOS DIGITALES AVANZADOS** tiene como objetivo desarrollar un sistema de seguridad basado en proximidad mediante un dispositivo IoT utilizando un Raspberry Pi Pico W. El sistema integrará un sensor ultrasónico para detectar la presencia de objetos en un rango de distancias, una pantalla OLED para mostrar información relevante y un teclado matricial que permitirá la autenticación mediante un pin de seguridad. Para reforzar la seguridad, el sistema implementará un mecanismo de validación que permitirá únicamente al usuario propietario desactivar la alarma, por alarma se entiende la notificación por correo. En caso de una detección sospechosa o intentos fallidos de autenticación, se enviará una notificación por correo electrónico con los detalles del evento.

Uno de los principales desafíos de este proyecto es la integración eficiente de todas las funcionalidades, asegurando un sistema fuerte, confiable y fácil de usar. Además, se implementará un mecanismo para reducir falsas alarmas, comparando la distancia de la detección actual con la anterior; si se detecta un alejamiento del objeto sospechoso, el sistema reiniciará el historial de detección. Con esta solución, no solo se busca mejorar la seguridad del entorno monitoreado, sino también desarrollar una herramienta accesible, adaptable y de fácil interacción para el usuario final.

2.2. Solución del Problema

El código implementa varias funciones que se han implementado en las prácticas a lo largo del semestre. A continuación, se explicará cada una de ellas con más detalle:

1. **init_keypad y scan:** Estas funciones son responsables de la inicialización del teclado matricial 4x4 y del escaneo de las teclas presionadas. La función init_keypad configura las filas del teclado en un estado bajo, preparándolas para su lectura, mientras que **scan** verifica cuál de las teclas ha sido presionada, leyendo la fila y columna correspondientes.

2. **tecla_cancelar_presionada(oled):** Esta función recibe como parámetro la instancia de la pantalla OLED y tiene como tarea preguntar al usuario, al final de cada ciclo de monitoreo, si desea continuar. Si el usuario presiona la tecla asterisco (*), la ejecución del programa se cancela. En caso contrario, el monitoreo continúa. La función retorna **True** si se presiona la tecla de cancelación, y **False** si no se presiona. El usuario tiene un segundo para presionar la tecla asterisco.

3. **obtener_fecha_hora_actual:** Esta función obtiene la fecha y hora actual del sistema, formateándolas en un formato adecuado para su visualización o envío. Los valores retornados son la fecha y la hora del momento en que se ejecuta la función.

4. **send_email(mensaje, distance, fecha, hora):** Esta función envía un correo de alerta cuando un objeto es detectado a una distancia peligrosa. Utiliza el servicio SMTP de Gmail para enviar el correo, incluyendo en el mensaje la distancia del objeto, la fecha y la hora de la detección. El correo se envía al correo establecido en el código (en este caso, para efectos

prácticos, tanto el remitente como el destinatario, son los mismos). La función retorna **True** si el correo fue enviado correctamente.

5. **leer_password(oled):** Esta función permite al usuario ingresar un pin de seguridad mediante el teclado matricial. Tiene un límite de tiempo de 10 segundos para ingresar la contraseña. Si el usuario no ingresa nada o la contraseña es incorrecta, el sistema enviará un correo de alerta. Si la contraseña es correcta, el sistema no enviará el correo. La función devuelve la contraseña ingresada (si es correcta o incorrecta) o una cadena vacía si el tiempo se agotó sin recibir una entrada.

6. **monitoreo():** Esta es la función principal que coordina todas las demás funciones. Se encarga de instanciar el sensor ultrasónico, la pantalla OLED y el LED azul. Monitorea de manera continua la distancia hasta que el usuario decida detenerlo. Se tiene una lista denominada **"ultimas_distancias"** y una variable de tipo entero denominada **"cantidad".**

Cantidad hace referencia al número de muestras tomadas y **ultimas_distancias** almacena las muestras tomadas siempre y cuando se cumpla la condicional de que las distancias sean decrecientes. En caso de que no sean decrecientes, el contenido de la lista ultimas_distancias se borra e inicia el muestreo de nuevo.

Una vez tomada las siete distancias, se toma una octava muestra que tiene que entrar en un rango de 0 a 30cm, en ese caso, pasa a emitirse la alerta y pedir el pin de seguridad para verificar si es un usuario propietario o es un intruso. Si la contraseña es incorrecta, se enviará un correo de alerta. La función también maneja casos de error cuando el sensor no mide correctamente (por ejemplo, cuando el valor de la distancia es -0.02).

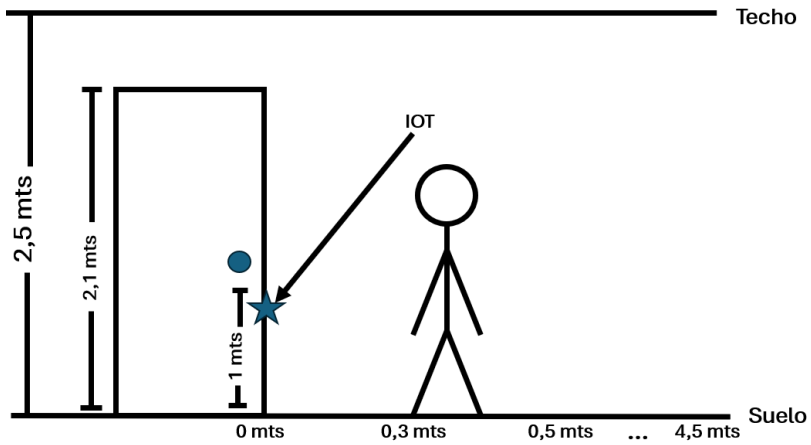
7. **mostrar_oled(oled, message, n):** Esta función muestra un mensaje en la pantalla OLED durante un período de tiempo determinado (especificado por el parámetro n). Divide el mensaje en palabras y la muestra de manera continua en la pantalla.

8. **escanear_redes(oled):** Esta función escanea las redes WiFi cercanas. Los nombres de las redes (SSIDs) detectadas se almacenan en una lista, la cual se mostrará en la pantalla OLED. Antes de agregar cada red a la lista, se verifica que el nombre no esté vacío.

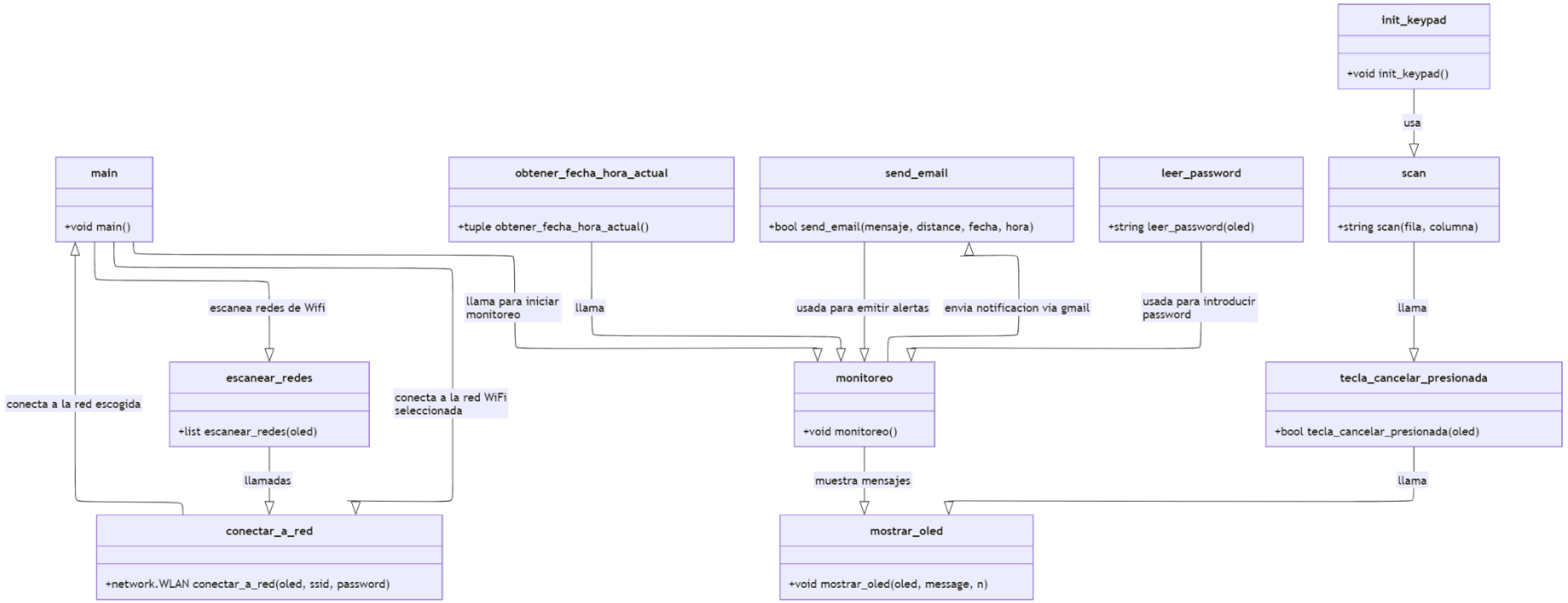
9. **conectar_a_red(oled, ssid, password):** Esta función permite al dispositivo conectarse a una red WiFi seleccionada. Recibe como parámetros el nombre de la red (SSID) y la contraseña, e intenta conectar el dispositivo a la red. Si la conexión no se establece en un tiempo determinado (10 segundos), se le pedirá al usuario que seleccione otra red.

10. **main():** La función principal del programa. Se encarga de escanear las redes WiFi-disponibles y permite al usuario seleccionar una red a la que conectarse. Una vez establecida la conexión, se inicia el proceso de monitoreo de distancias. En este proceso, si se detecta un objeto cercano, se le solicita al usuario ingresar una contraseña para decidir si se enviará un correo de alerta o no, aunque ya esta parte fue explicada en las funciones anteriormente mencionadas.

2.3. Imagen ilustrativa de su uso

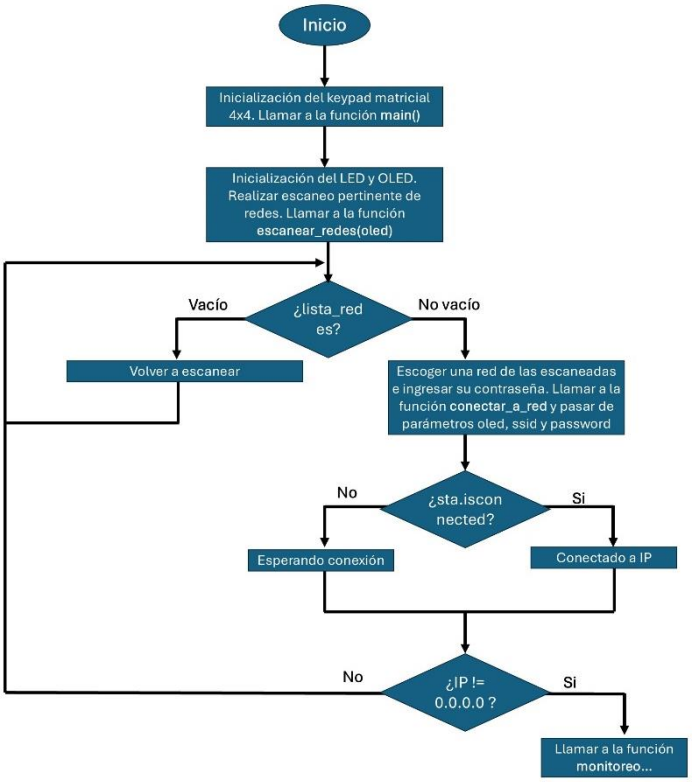


2.4. Diagrama UML

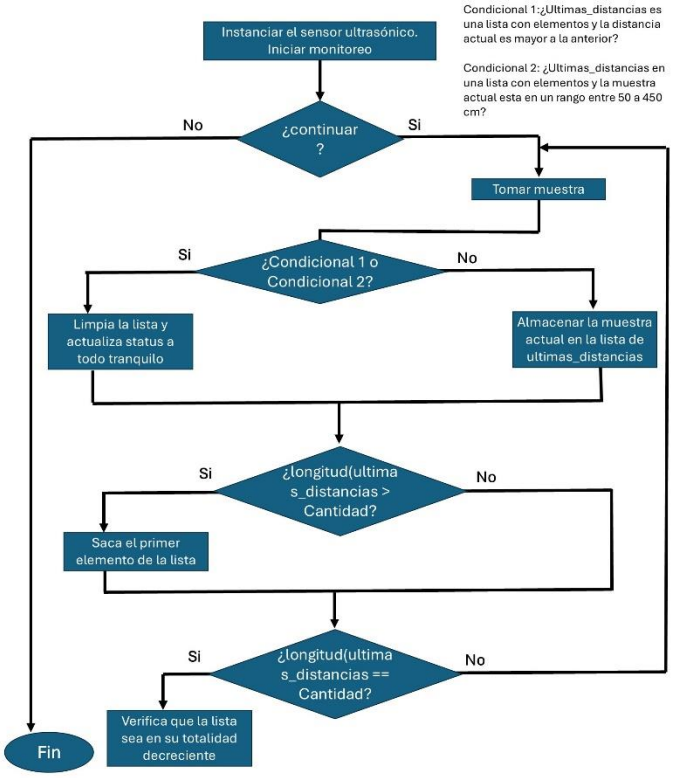


2.5. Diagrama de Flujo

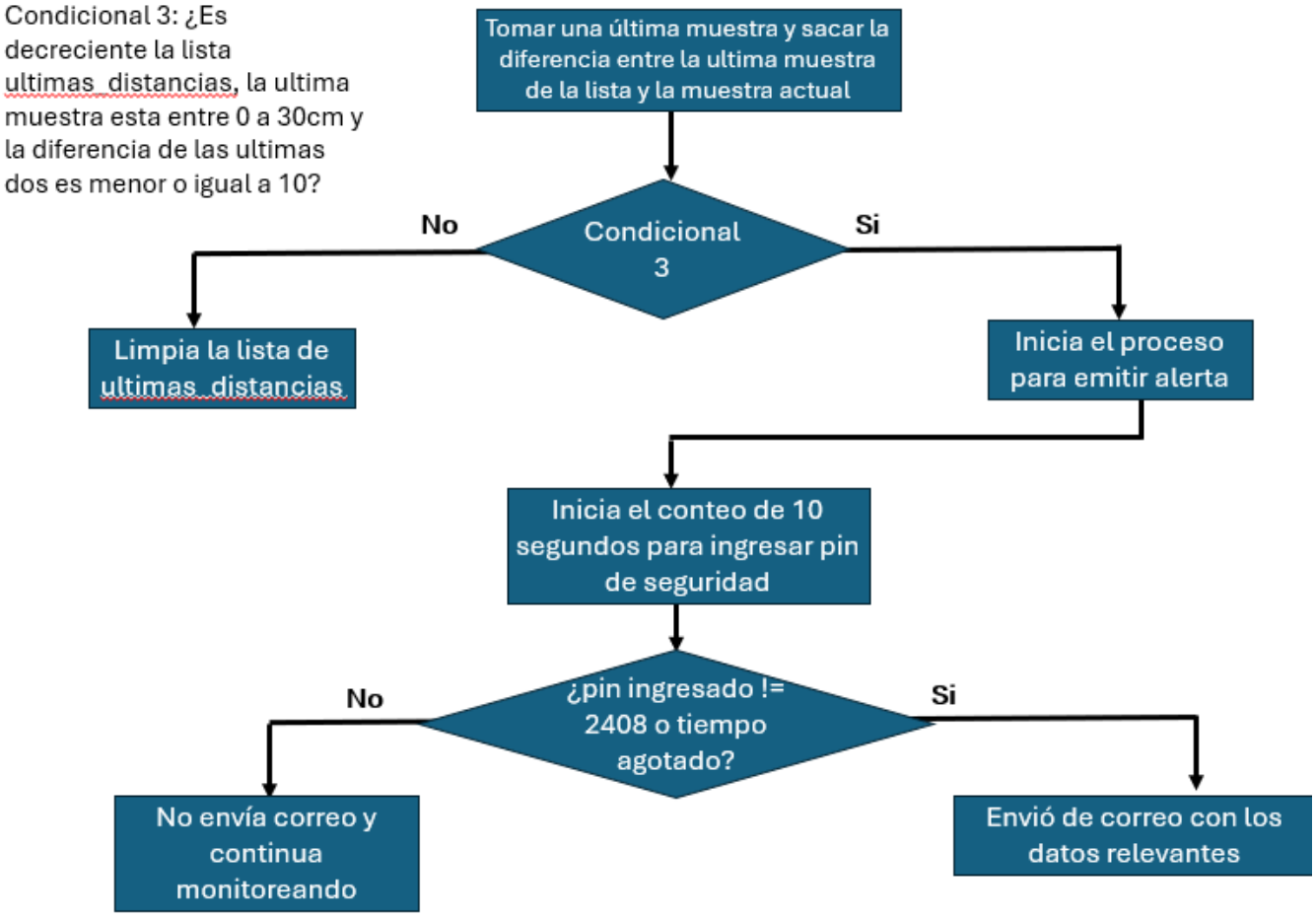
2.5.1. Parte I: Conexión a la red WiFi



2.5.2. Parte II: Monitoreo



2.5.3. Parte III: Comparación de datos



Sea una de las dos alternativas, retorna a lo que es el monitoreo, específicamente a la parte de tomar muestras.

2.6. Observaciones.

- Un problema que se presento fue la lectura erronea del sensor ultrasonico que tomaba el valor de “-0.02” esto se opto por hacer una advertencia diciendo que la muestra fue mal tomada.
- Asi mismo, se recuerda que los valores de las variables como: ultimas_distancias, cantidad, el correo utilizado y los tiempos al tomar las distancias son modificables y los utilizados son para efectos prácticos. En situaciones reales, el correo emisor puede ser el de una empresa, como la cantidad de distancias tomadas puede ser mayor y los tiempos entre distancias puede ser de 1 a 2 minutos. Tambien se hace referencia al tiempo estipulado para ingresar el pin de seguridad.

3. ANEXOS

3.1. Código:

```
from time import sleep, time as timer
from machine import Pin, I2C
import network
import umail
import socket
from ssd1306 import SSD1306_I2C
from hcsr04 import HCSR04
import time

# -- configuracion del keypad -- #
TECLA_ARRIBA  = const(0)
TECLA_ABAJO = const(1)

teclas = [['1', '2', '3', 'A'], ['4', '5', '6', 'B'], ['7', '8', '9', 'C'], ['*', '0', '#', 'D']]
filas = [2, 3, 4, 5]
columnas = [6, 7, 8, 9]

fila_pines = [Pin(nombre_pin, mode=Pin.OUT) for nombre_pin in filas]
columna_pines = [Pin(nombre_pin, mode=Pin.IN, pull=Pin.PULL_DOWN) for nombre_pin in columnas]

# -- init_keypad: inicializacion del keypad -- #
def init_keypad():
  for fila in range(0, 4):
    fila_pines[fila].low()

# -- scan (fila, columna): toma que tecla se esta presionado -- #
def scan(fila, columna):
  fila_pines[fila].high()
  tecla = None
  if columna_pines[columna].value() == TECLA_ABAJO:
    tecla = TECLA_ABAJO
  if columna_pines[columna].value() == TECLA_ARRIBA:
    tecla = TECLA_ARRIBA
  fila_pines[fila].low()
  return tecla

# -- tecla_cancelar_presionada(): esta es para finalizar ejecucion del programa -- #
def tecla_cancelar_presionada(oled):
  start_time = time.time()  # temporizador
  while time.time() - start_time < 1:  # max 5 seg para  que cancele
    for fila in range(4):
      for columna in range(4):
        tecla = scan(fila, columna)
        if tecla == TECLA_ABAJO and teclas[fila][columna] == "*":
          print("Tecla '*' presionada. Cancelando...")
          mostrar_oled(oled, "Status: Ha presionado *. Saliendo..", 1)
          return True
    sleep(0.1)  # mini pausa, nunca quitar

  # no se presiono la tecla
  #print("Tiempo agotado. Cancelando automaticamente...")
  #mostrar_oled(oled, "Tiempo agotado. Cancelando automaticamente...", 1)
  return False

# -- obtener_fecha_hora_actual: funcion para obtener fecha y hora -- #
def obtener_fecha_hora_actual():
  current_time = timer()
  local_time = time.localtime(current_time)
  year = local_time[0]
  month = local_time[1]
  day = local_time[2]
  hour = local_time[3]
  minute = local_time[4]
  second = local_time[5]
  fecha = "{:02}/{:02}/{}".format(day, month, year)
  hora = "{:02}:{:02}:{:02}".format(hour, minute, second)
  return fecha, hora
```

```
# -- send_email(cuerpo del mensaje, distancia, fecha, hora): envio del mensaje por correo -- #
def send_email(mensaje, distance, fecha, hora):
    sender_email = "claudiaelena091@gmail.com" # correo emisor
    sender_name = "Claudia" # nombre del enviador
    sender_app_password = "nxip rybc pfqc eqju" # config del gmail
    email_subject = 'Alerta: Objeto detectado cerca' # asunto
    recipient_email = "claudiaelena091@gmail.com" # quien lo va a recibir
    try:
        smtp = umail.SMTP('smtp.gmail.com', 465, ssl=True)
        smtp.login(sender_email, sender_app_password)
        smtp.to(recipient_email)
        smtp.write("From:" + sender_name + "<" + sender_email + ">\n")
        smtp.write("Subject:" + email_subject + "\n")
        smtp.write(mensaje)
        smtp.send()
        smtp.quit()
        print(f"Correo enviado. Por favor, revisar bandeja de entrada de: {recipient_email}")
    except Exception as e:
        print(f"Error al enviar el correo: {e}")
    return True

# -- leer_password(oled): lee la contraseña de seguridad desde el keypad con temporizador -- #
def leer_password(oled):
    password = ""
    start_time = timer()  #  timer
    mostrar_oled(oled, "Ingrese password:", 0.5)
    while len(password) < 4:  # pass de 4 digitos
        if timer() - start_time > 10:  # verificacion de tiempo
            cadena = "Tiempo agotado. Password incorrecta."
            print(cadena)
            mostrar_oled(oled, cadena, 2)
            return ""  # como no recibio nada, mejor retorna vacio

        tecla_presionada = False
        for fila in range(4):
            for columna in range(4):
                tecla = scan(fila, columna)
                if tecla == TECLA_ABAJO:
                    tecla_presionada = True
                    print("Tecla presionada:", teclas[fila][columna])
                    password += teclas[fila][columna]
                    mostrar_oled(oled, f"Ingrese password: {password}", 0.2)
                    sleep(0.5)  # mini pausa para que no escriba varias veces
        if not tecla_presionada:
            sleep(0.1)  # mini pausa
    return password  # regresa contraseña ingeresada de 4 digitos

def monitoreo():

    # crear archivo para guardar un registro
    with open("registros.txt", "w") as file:
        file.write("identificador | fecha | hora | distancia | correo enviado\n")

    # instanciar sensor, pantalla oled, y led
    sensor = HCSR04(trigger_pin=27, echo_pin=26, echo_timeout_us=10000)
    oled = SSD1306_I2C(128, 64, I2C(0, scl=Pin(17), sda=Pin(16), freq=400000))
    azul = Pin(14, Pin.OUT)

    # var bool para continuar monitoreo
    continuar = True
```

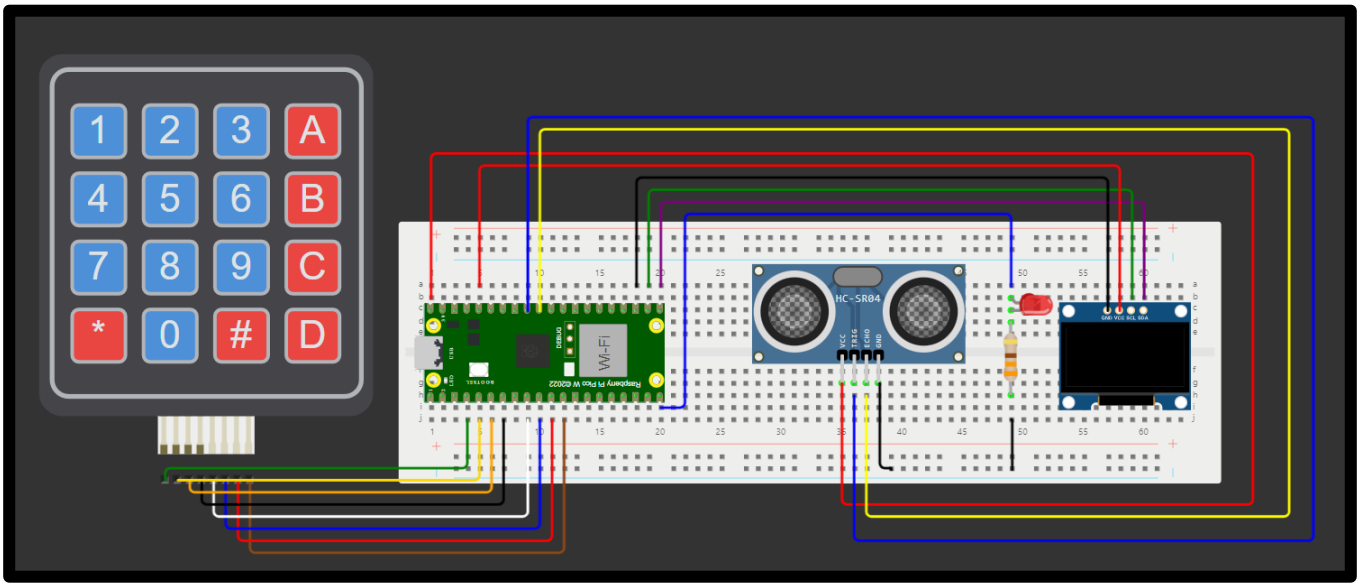
```
  # mostrar esto para saber que ya estas en la func de monitoreo()
  cadena = "Status: CONEXION ESTABLECIDA"
  mostrar_oled(oled, cadena, 3)

  # lista para almacenar las distancias
  ultimas_distancias = []

  # cantidad de veces que hace el monitoreo de distancia
  # para estar seguros y no emitir falsas alarmas
  cantidad = 7

  # contraseña para desactivar el envio de correo
  contra = "2408"

  # bucle de monitoreo
  # la var continuar se altera si se deja presionado
  # el asterisco en el keypad
  while continuar:
    distance = round(sensor.distance_cm(), 2)
    print("Distancia:", distance, "cm")
    azul.value(1)

    # aqui cambia la var continuar
    if tecla_cancelar_presionada(oled):
      continuar = False
      mostrar_oled(oled, "Status: Finalizando monitoreo...", 2)
      sta = network.WLAN(network.STA_IF)
      sta.active(False)
      break

    # if por si da margen de error y reinicia la lista
    if distance == -0.02:
      print("Margen de error. Sensor no tomo la medida correcta")
      cadena = "STATUS: Muestra mal tomada. Intentando de nuevo"
      mostrar_oled(oled, cadena, 0.5)
      ultimas_distancias.clear()

    # caso contrario de que si la tome bien
    else:
      mostrar_oled(oled, f"Distancia: {distance} cm", 0.30)
      fecha, hora = obtener_fecha_hora_actual()

      # condicional para verificar que la distancia anterior sea menor que la actual
      # si se cumple, elimina las distancias tomadas y reinicia la lista de
      # ultimas_distancias

      # se le añadio una condicional de 30 <= distance <= 60 para avisar
      # que no hay peligro, peligro existe si la distancia es menor a 30 cm

      if ultimas_distancias and distance > ultimas_distancias[-1] or ultimas_distancias and 50 <= distance <= 450:
        ultimas_distancias.clear()
        print("Patron de distancias reiniciado debido a aumento en la distancia.")
        mostrar_oled(oled, "STATUS: Todo tranquilo, sin peligro...", 2)


      # si no es el caso anterior, la añade
      ultimas_distancias.append(distance)

      # esta condicional es para asegurar que la lista d eultimas_distancias no supere
      # el tamaño que se establecio con la var cantidad, si llega a pasar eso, elimina
      # el primer elemento
      if len(ultimas_distancias) > cantidad:
        ultimas_distancias.pop(0)

      # si la cantidad de elementos en ultimas_distancias coincide con la cantidad de muestras
      # entra en el bucle
      if len(ultimas_distancias) == cantidad:
```

```
        # esto es para verificar que si sea
decreciente y marcarla como true
        es_decreciente =
all(ultimas_distancias[i] >
ultimas_distancias[i + 1] for i in
range(cantidad - 1))

        # pausa para verificar donde se
encuentra la persona
        # si esta abriendo la puerta

        sleep(2)  # pausa antes de tomar la
nueva medida
        distance =
round(sensor.distance_cm(), 2)
        print("Comparacion de distancia:",
distance, "cm")
        mostrar_oled(oled, f"Distancia:
{distance} cm", 0.30)

        # sacar diferencia de la tomada y la
ultima de la lista
        diferencia_ultimas_dos =
abs(distance - ultimas_distancias[-1]) <= 10
        print(f"distance: {distance} ////
ultimas_distancias: {ultimas_distancias}")

        if distance < ultimas_distancias[-1]:
            ultimas_distancias.clear()

        # entonces, si la lista es decreciente
y la ult distancia esta entre 0 a 30 y
        # la diferencia de las ultimas_dos es
menor o igual a 10
        if es_decreciente and 0 <= distance
<= 30 and diferencia_ultimas_dos:
            # mensaje para advertir que si
entro al bucle
            cadena = f"ALERTA! Intruso
detectado. Ult. distancia: {distance}cm"
            print(cadena)
            mostrar_oled(oled, cadena, 2.5)

            # alerta usando el led azul
            azul.value(1)
            sleep(1)
            azul.value(0)
            sleep(1.5)
            azul.value(1)
            sleep(1)

            # mensaje para advertir que puede
ingresar el pin de seguridad
            cadena = "ADVERTENCIA: Tiene 10
segundos para ingresar el pin de seguridad o
el correo sera enviado."
            print(cadena)
            mostrar_oled(oled, cadena, 2.5)

            # llamada a la funcion de leer la
contraseña del keypad
            password = leer_password(oled)

            # verificacion (contra se definio
anteriormente)
            if password == contra:
                cadena = "Status: Password
correcta, no se enviara el correo..."
                print(cadena)
                mostrar_oled(oled, cadena, 2.5)

            # envio de correo
            else:
                cadena = "Status: Password no
valida. Enviando correo..."
                print(cadena)
                mostrar_oled(oled, cadena, 2.5)
                mensaje = f"Alerta! Se ha
detectado un intruso acercandose
lentamente. Datos tomados del
monitoreo.... \nDistancia:
{distance}cm\nHora: {hora}\nFecha: {fecha}"
                mostrar_oled(oled, f"Correo:
{mensaje}", 3)
                enviado = send_email(mensaje,
distance, fecha, hora)

                # si resulto true
                if enviado:
                    mostrar_oled(oled, "Status:
Correo enviado", 2)
```

```
                with open("registros.txt", "a") as
file:

file.write(f"Alerta_{fecha}_{hora} | {fecha} |
{hora} | {distance} cm | Si\n")
                ultimas_distancias.clear()
                print(ultimas_distancias) #
verificar que si se limpio
            else:
                ultimas_distancias.clear()  #
reiniciar el patron si no se cumplen ambas
condiciones
                print("Patron de distancias
reiniciado.")

                azul.value(0)
    print("Monitoreo finalizado.")



# -- mostrar_oled(instancia, mensaje str, y
tiempo que permanece en la pantalla)
def mostrar_oled(oled, message, n):
    oled.fill(0)
    ancho_caracter = 7
    max_columna = 120
    fila = 0
    columna = 0
    palabras = message.split()
    for palabra in palabras:
        ancho_palabra = len(palabra) *
ancho_caracter
        if columna + ancho_palabra >
max_columna:
            fila += 16
            columna = 0
        if fila >= 50:
            oled.show()
            sleep(n)
            oled.fill(0)
            fila = 0
            columna = 0
        oled.text(palabra, columna, fila)
        columna = columna + 7
        columna += ancho_palabra +
ancho_caracter
    oled.show()
    sleep(n)

# -- escanear_redes(oled): escaneo de wifis
cercanas 2.4ghz -- #
def escanear_redes(oled):
    print("Escaneando redes WiFi cercanas...")
    sta = network.WLAN(network.STA_IF)
    sta.active(True)
    redes = sta.scan()
    lista_redes = []
    for red in redes:
        ssid = red[0].decode('utf-8').strip()
        if ssid:  # verifica que no sea una cadena
vacia
            lista_redes.append(ssid)
    return lista_redes

# -- conectar_a_red(oled instancia, nombre
de la red, contraseña): conectar a un ap -- #
def conectar_a_red(oled, ssid, password):
    cadena = f"Conectando a {ssid}..."
    mostrar_oled(oled, cadena, 2)
    print(cadena)
    sta = network.WLAN(network.STA_IF)
    sta.active(True)
    sta.connect(ssid, password)
    a = 1
    while not sta.isconnected():
        print("Esperando conexion...")
        mostrar_oled(oled, f"Esperando
conexion con {ssid}", 2)
        sleep(2)
        a = a + 1
        if a >= 5:
            break
    ip = sta.ifconfig()[0]
    print(f"IP: {ip}")
    if ip == "0.0.0.0":
        print("Retomando a escoger red.
Conexion no aceptada. Verifique
disponibilidad.")
        mostrar_oled(oled, "Conexion no
establecida... Retornando", 2)
    return sta
```

```
# -- main -- #
def main():
  azul = Pin(14, Pin.OUT)
  oled = SSD1306_I2C(128, 64, I2C(0,
scl=Pin(17), sda=Pin(16), freq=400000))

  azul.value(0)
  oled.fill(0)
  sleep(2)

  cadena = "SISTEMA DE SEGURIDAD DE
DISTANCIA"
  mostrar_oled(oled, cadena, 4)
  print(cadena)

  cadena = "Status: Escaneando redes de
WiFi disponibles.."
  mostrar_oled(oled, cadena, 4)
  print(cadena)
  ssid = ""
  password = ""

  sta = network.WLAN(network.STA_IF)
  sta.active(False)

  while not sta.isconnected():
    lista_redes = escanear_redes(oled)
    if lista_redes:
      cadena = "Se han encontrado redes,
verifique la consola"
      mostrar_oled(oled, cadena, 3)
      print(cadena)
    else:
      cadena = "No se han encontrado
redes, verifique que si exista"
      mostrar_oled(oled, cadena, 3)
      print(cadena)
        cadena = "Selecciona una red en
consola."
    mostrar_oled(oled, cadena, 3)
    print(cadena)
    for i, red in enumerate(lista_redes):  #
enlistar redes
      print(f"{i + 1}. {red}")

    opcion_red = int(input("Marque el
numero de la red a la que desea conectar (0
para salir): "))
    if opcion_red < 1 or opcion_red >
len(lista_redes):
      print("Opcion invalida. Saliendo...")
      return

    ssid = lista_redes[opcion_red - 1]
    cadena = f"Red seleccionada: {ssid}.
Revise consola."
    print(cadena)
    mostrar_oled(oled, cadena, 3)
    password = input("Ingrese la password
de la red: ")
    print(f"Conectando a la red: {ssid} con
password: {password}")
    sta = conectar_a_red(oled, ssid,
password)

  mostrar_oled(oled, f"Conectado a: {ssid}..
Redirigiendo..", 3)
  monitoreo()

if __name__ == "__main__":
  init_keypad()
  main()
```

3.2. Diagrama Circuital



3.3. Bibliografía

Teclado de Membrana Matricial 4x4 con Raspberry Pi Pico. Códigos MicroPython validación de password. Enlace: https://www.youtube.com/watch?v=Ew-fO-uTuFA

NUEVO SISTEMA DE ALARMA DE SEGURIDAD PARA CASAS Y NEGOCIOS 2020 - GSM WIFI. Enlace: https://www.youtube.com/watch?v=xuAoIu2S_IE

La guía definitiva para el sistema de alarma contra intrusiones en 2024: proteger completamente su hogar. Enlace: https://www.roombanker.com/es/blog/intrusion-alarm-system-complete-guide/

Falsas alarmas y cómo evitarlas. Enlace: https://ajax.systems/es/blog/false-alarms-and-how-to-avoid-them/

Por qué los detectores de movimiento reaccionan a los animales y cómo evitarlo. Enlace: https://ajax.systems/es/blog/what-is-pet-immunity-in-motion-detectors-and-how-to-use-it-correctly/

Como Funciona un sistema de alarmas. Enlace: https://www.youtube.com/watch?v=NMhn1rJv_jg

How to Send Email With Raspberry Pi Pico W. Enlace: https://www.youtube.com/watch?v=tfp-Futa-lw