INFORME PROYECTO

Máquina de Estados

Elaborado por:

Juan Camilo Benavides Salazar

104622020723

Jose Alejandro Rodriguez Dueñas

104622020707

Glenn Alexander Ward Ante

104622020698

Universidad del Cauca

Facultad de Ingeniería Electrónica y Telecomunicaciones

Ingeniería de Sistemas

Arquitectura Computacional

Popayán, Cauca, Colombia

30/11/2024

MÁQUINA DE ESTADOS

Planteamiento del problema

Problema principal

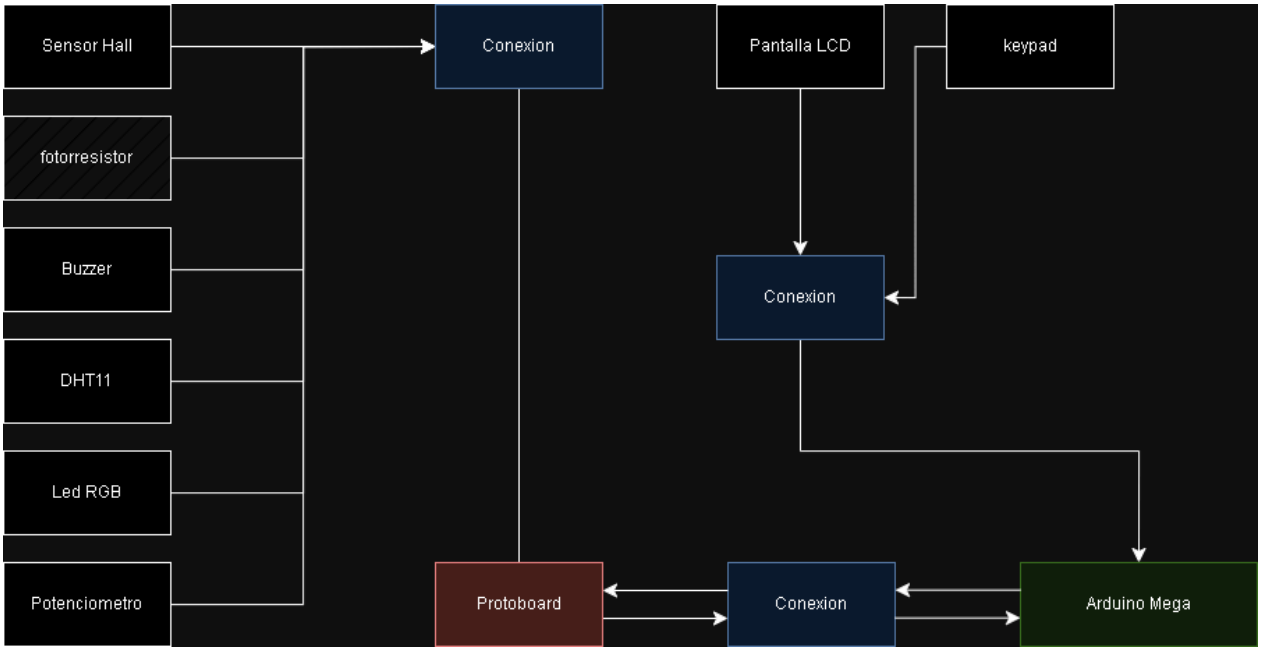
El diseño de sistemas electrónicos eficientes y versátiles requiere soluciones que permitan gestionar múltiples variables de manera simultánea. En este contexto, se busca implementar una máquina de estados finitos (MEF) utilizando una placa Arduino, capaz de manejar eventos complejos derivados de la interacción y monitoreo de diferentes variables ambientales.

Entre las variables que se deben considerar están la temperatura, la humedad, la luz, la presencia de infrarrojos y los campos magnéticos. Cada una de estas variables puede generar eventos que requieren respuestas específicas del sistema, lo que demanda un diseño robusto y escalable. El principal reto radica en organizar de forma lógica los posibles estados del sistema y las transiciones entre ellos, considerando las combinaciones de condiciones de las variables monitoreadas.

El proyecto debe garantizar un comportamiento eficiente y preciso, que permita:

- Detectar cambios en las condiciones ambientales.
- Tomar decisiones en tiempo real según los eventos detectados.
- Implementar transiciones claras y efectivas entre estados, asegurando que el sistema sea fácil de depurar, modificar y escalar.

Diagrama General del proyecto



Descripción de cada módulo que conforma el proyecto

Módulo de Sensores:

- **Sensor DTH11:** Diseñado para medir temperatura y humedad ambiental. Proporciona datos digitales que se procesan para monitorear las condiciones climáticas del entorno y generar eventos en la máquina de estados.
- **Sensor Hall:** Detecta campos magnéticos cercanos. Su función es identificar la presencia o ausencia de un campo magnético para activar o modificar estados del sistema según el contexto definido.
- **Sensor Fotoresistor:** Monitorea los niveles de luz ambiental. Se utiliza para detectar cambios en la iluminación que pueden influir en las transiciones de estados, como activar un modo nocturno o de alta luminosidad.

Módulo de Procesamiento

- **Arduino Mega:** Actúa como el núcleo del sistema, implementando la máquina de estados finitos (MEF). Procesa las señales de entrada de los sensores y controla las salidas hacia los actuadores. Su capacidad de

múltiples pines de E/S es crucial para manejar todas las conexiones necesarias.

Módulo de Actuadores

- **Pantalla LCD:** Permite mostrar información relevante del sistema, como las variables monitoreadas (temperatura, humedad, luz, etc.), el estado actual de la máquina de estados y las acciones tomadas en tiempo real.
- **Buzzer:** Genera alertas audibles para informar sobre eventos críticos detectados, como alta temperatura o presencia de campos magnéticos no deseados.
- **Led RGB:** Utilizado como indicador visual del estado del sistema. Cambia de color según el estado activo de la máquina (por ejemplo, verde para normal, rojo para alerta, azul para modo nocturno).
- **Consola Serial:** Proporciona una herramienta de depuración y comunicación. Envía información detallada al desarrollador sobre los eventos detectados, transiciones de estados y cualquier anomalía.

Módulo de Interfaz de Usuario

- **Pantalla LCD:** Facilita la interacción visual con el usuario, permitiendo que revise los valores actuales de las variables o elija configuraciones específicas.
- **KeyPad:** Permite al usuario navegar por menús, configurar parámetros del sistema, seleccionar modos operativos y activar funciones manualmente.

Diagrama de flujo del software de manera general

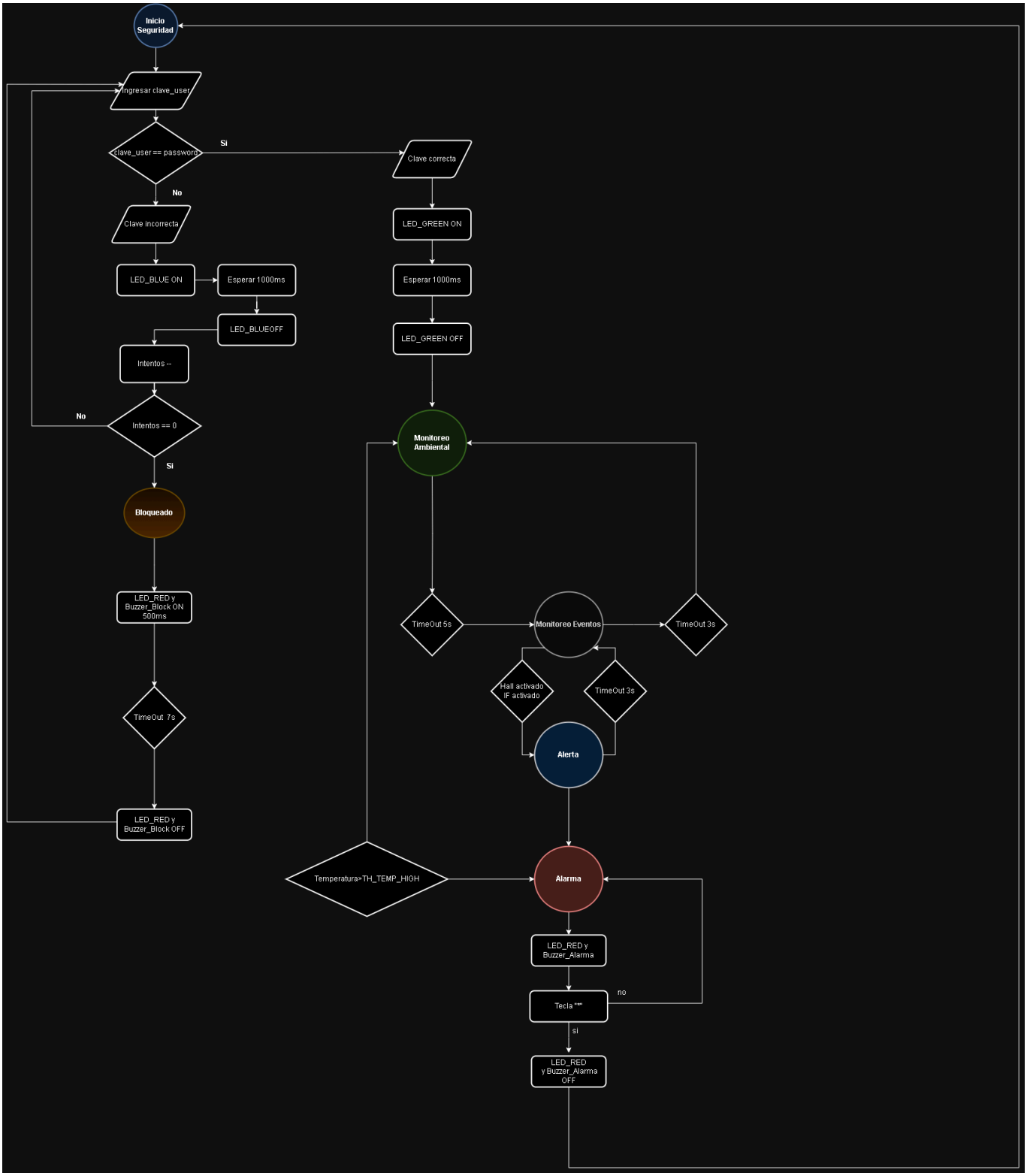
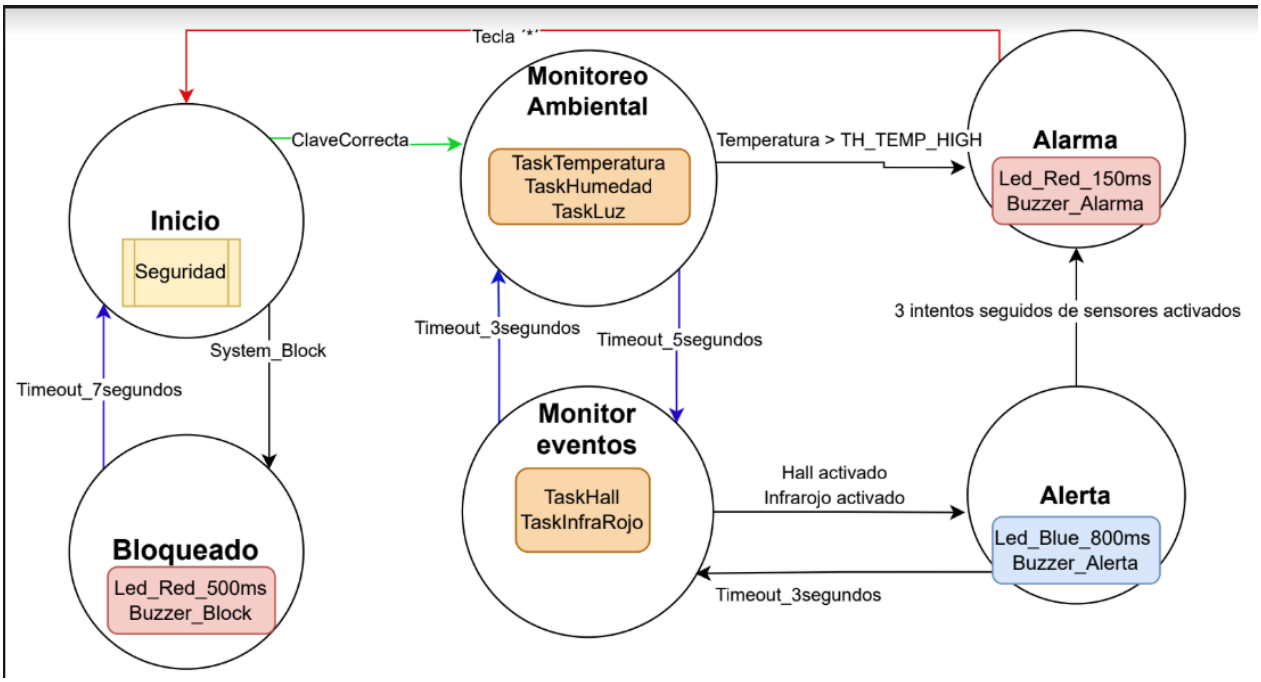


Diagrama de la máquina de estados a diseñar



Herramientas de desarrollo

Para el desarrollo del proyecto las herramientas que se están usando son:

Arduino IDE.	

Repositorio en github. Manejo de versiones de código fuente.	
WokWi. Simulador de proyectos arduino	

Componentes software

Interfaz gráfica de usuario (GUI)

LIBRERÍA	FUNCIONES	DESCRIPCIÓN	ESTADO
Keypad	customKeypad.getKey()	Devuelve el carácter de la tecla que ha sido presionada, permitiendo capturar la entrada del usuario en tiempo real.	Implementado
	keypadBuffer.push()	Añade un carácter al buffer, almacenando la entrada del usuario para procesarla más adelante.	
	keypadBuffer.len	Indica la longitud actual del contenido almacenado en el buffer.	
	keypadBuffer.str	Retorna el contenido completo del buffer como un string, ideal para procesar cadenas de entrada.	
	keypadBuffer.isFull()	Comprueba si el buffer ha alcanzado su capacidad máxima,	

		evitando sobrecargas de datos.	
	keypadBuffer.clear()	Vacía todo el contenido almacenado en el buffer, dejándolo listo para nuevas entradas.	
LiquidCrystal	lcd.begin()	Inicializa la comunicación con el lcd	Implementado
	lcd.print();	Muestra texto en la pantalla lcd	
	lcd.clear()	Limpia la pantalla lcd	
	lcd.setCursor();	Posiciona el cursor en una posición específica de la pantalla lcd	
	LiquidCrystal <name>(LCD_RS, LCD_EN, LCD_D4, LCD_D5, LCD_D6, LCD_D7);	Constructor	

Sensores:

LIBRERÍA	FUNCIONES	DESCRIPCIÓN	ESTADO
DHT sensor library	LiquidCrystal <name>(LCD_RS, LCD_EN, LCD_D4, LCD_D5, LCD_D6, LCD_D7)	Constructor que define las conexiones entre el Arduino y el LCD, estableciendo los pines necesarios para la comunicación.	Implemented
	lcd.begin()	Inicializa la pantalla LCD, configurando el número de columnas y filas según las especificaciones del hardware.	
	lcd.setCursor()	Establece la posición del cursor en una ubicación específica de la pantalla LCD,	

		indicada por las coordenadas de columna y fila.	
	lcd.clear()	Limpia todo el contenido de la pantalla LCD, dejando el cursor en la posición inicial.	
	lcd.print()	Muestra texto o valores en la pantalla LCD desde la posición actual del cursor.	

Actuadores:

LIBRERÍA	FUNCIONES	DESCRIPCIÓN	ESTADO
LiquidCrystal	lcd.begin()	Inicializa la comunicación con el lcd	Implementado
	lcd.print();	Muestra texto en la pantalla lcd	
	lcd.clear()	Limpia la pantalla lcd	
	lcd.setCursor();	Posiciona el cursor en una posición específica de la pantalla lcd	
	LiquidCrystal <name>(LCD_RS, LCD_EN, LCD_D4, LCD_D5, LCD_D6, LCD_D7);	Constructor	

Demás librerías:

LIBRERÍA	FUNCIONES	DESCRIPCIÓN	ESTADO
StateMachineLib	StateMachine::addTransition(fromState, toState, condition_function)	Agrega una transición entre dos estados, especificando una condición que debe cumplirse para realizar el cambio.	Implementado
	stateMachine.SetOnEntering(state_name, callback_function)	Asigna una función de retorno (callback) que se ejecutará automáticamente al entrar en el estado especificado	
	stateMachine.SetOnLeaving(state_name, callback_function)	Asigna una función (callback) que se ejecuta automáticamente al abandonar el estado especificado. Es útil para realizar tareas de limpieza, desactivar actuadores o guardar datos antes de realizar la transición a otro estado.	
AsyncTask Lib	AsyncTask::start()	Inicia la ejecución de una tarea asíncrona. La tarea comenzará a ejecutarse según las condiciones definidas.	Implementado
	AsyncTask::stop()	Detiene la ejecución de una tarea en curso, liberando recursos asociados a ella.	
	AsyncTask::update()	Verifica y ejecuta la tarea asíncrona si ha llegado el momento según el intervalo	

		establecido. Debe ser llamada periódicamente dentro del ciclo loop().	
	AsyncTask::reset() .	Reinicia el temporizador de la tarea, comenzando a contar nuevamente desde cero sin detener la tarea	

Alarma

Se ha implementado un sistema de alarma único, el cual se activa cuando se superan los intentos consecutivos con los sensores activados (Hall e Infrarrojo). Además, este sistema funciona como un mecanismo de bloqueo, activándose cuando se excede el número máximo de intentos en la introducción de la contraseña.

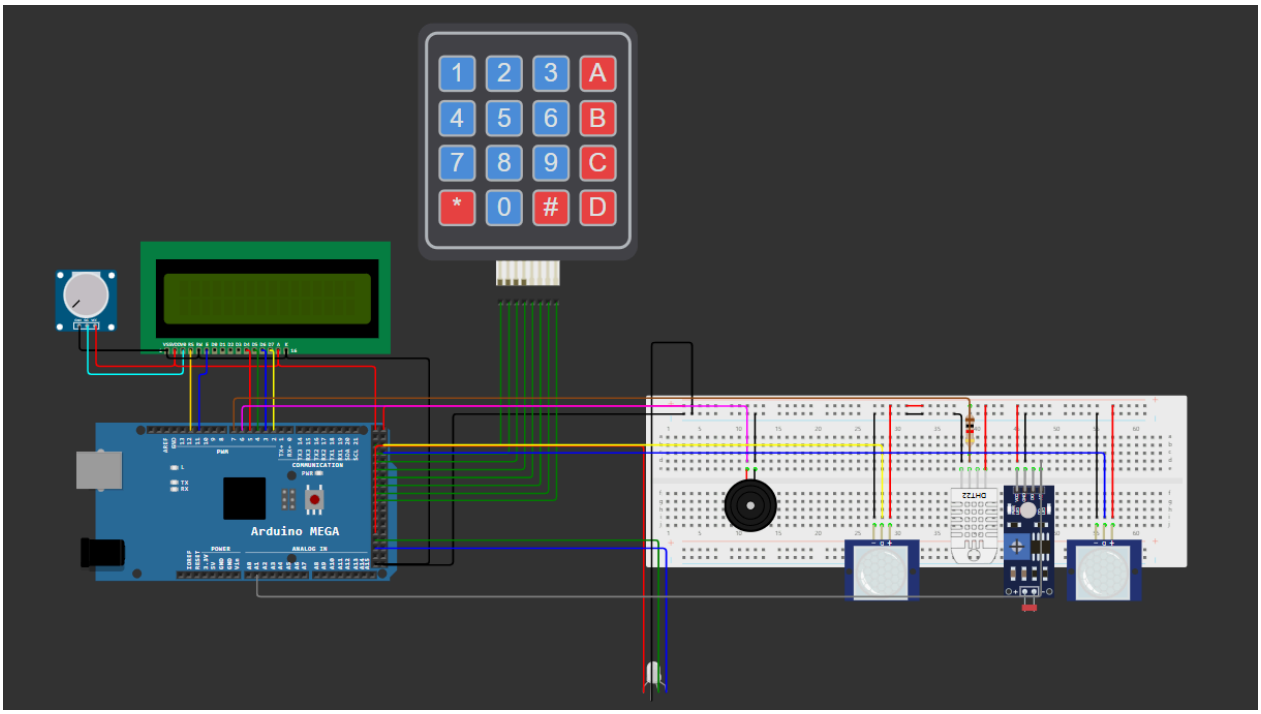
Proceso de pruebas

Test Backlog (min 10 pruebas)

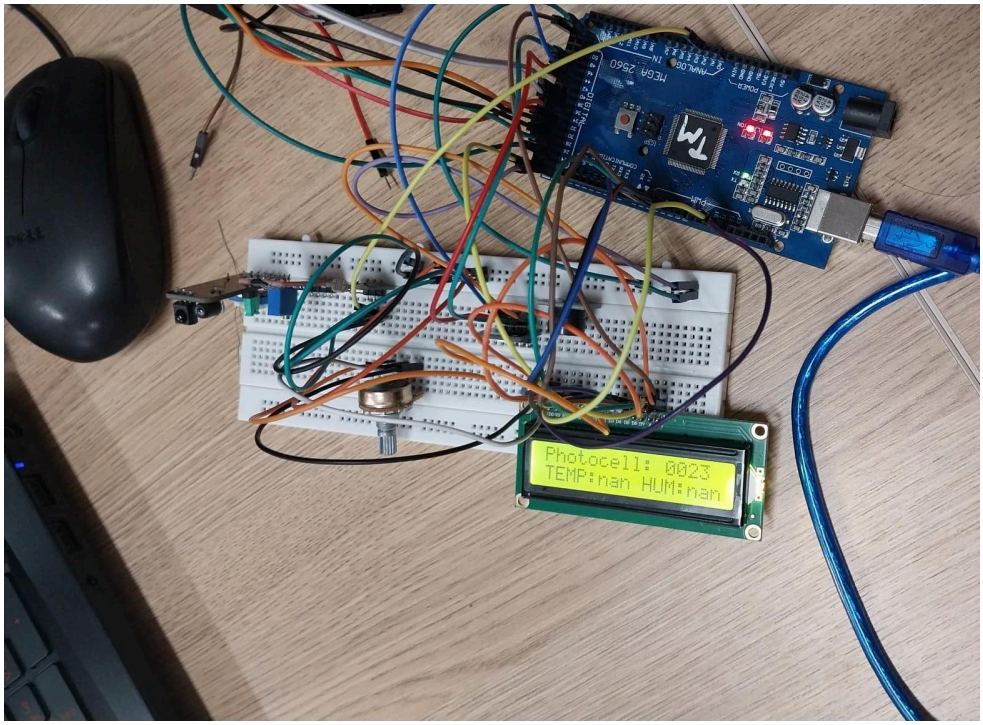
#	PRUEBA	DESCRIPCIÓN	PASÓ PRUEBA (SI/NO)	OBSERVACIONES
1	Verificar el teclado Keypad	Presionar el teclado y visualizar el carácter por el puerto serial.	Sí	Todos los caracteres se muestran correctamente en el serial.
2	Verificar el color del led dependiendo si la contraseña fue correcta o no.	Se ingresa la contraseña correcta el led debe ser de color verde, si falla 1 o 2 veces debe ser azul, si falla 3 veces de color rojo.	Si	El color del led fue correcto para cada situación.
3	Medir la humedad con el sensor DHT11	subir la humedad al 90% y luego bajarla al 10%.	No	La lectura de humedad es incorrecta para valores bajos.
4	Medir la luz con el fotoresistor	Cambiar la intensidad de la luz.	Sí	Las lecturas de luz corresponden con la realidad.

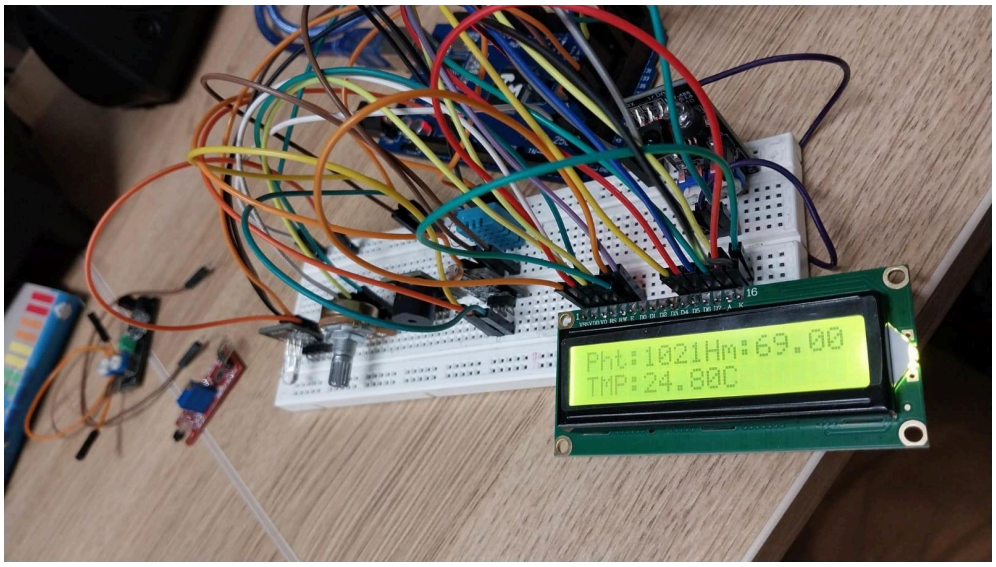
5	Configurar umbrales con el potenciómetro	Ajustar el potenciómetro y comprobar que los umbrales de temperatura y humedad se modifican correctamente.	Sí	Los umbrales se ajustan adecuadamente mediante el potenciómetro.
6	Visualizar alarmas en la pantalla LCD	Superar los umbrales de temperatura y verificar la visualización de las alarmas en la pantalla LCD.	Sí	Las alarmas se muestran correctamente en la pantalla LCD.
7	Alertas del LED RGB	Superar los umbrales de temperatura y verificar el cambio de color en del LED RGB.	Sí	El LED cambia de color según lo esperado.
8	Alertas del buzzer	Superar el número de intentos de los sensores y verificar que el buzzer emite sonidos de alarma	Sí	El buzzer suena correctamente cuando se superan los umbrales.
9	Función de bloqueo del sistema	Pasar el límite de intentos de contraseña.	Sí	El sistema se bloquea correctamente.
10	Tiempo de espera entre los estados.	Se toma el tiempo de espera que hay establecido entre el cambio de estados	Sí	El tiempo de espera es el correcto para cada cambio de estado.

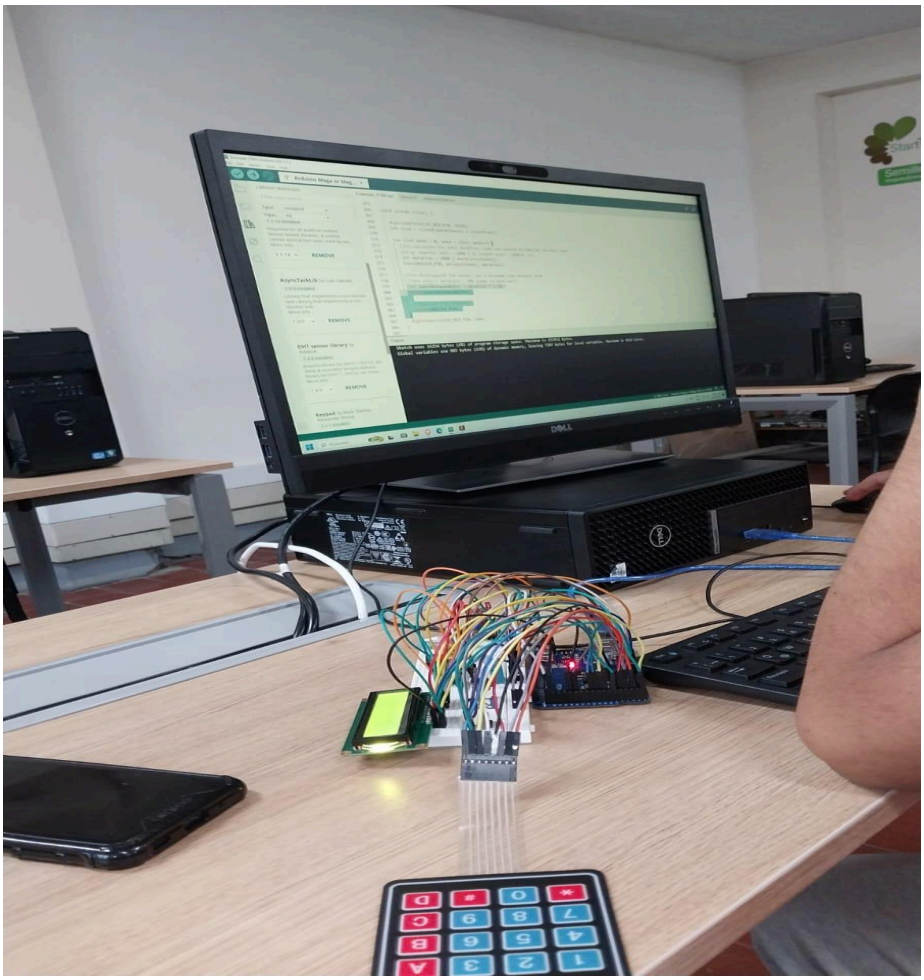
Diagrama final

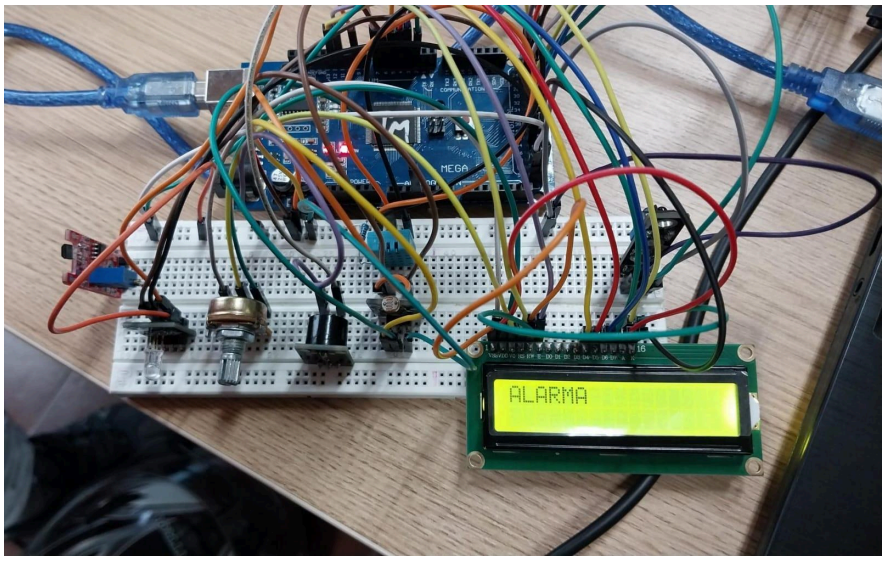


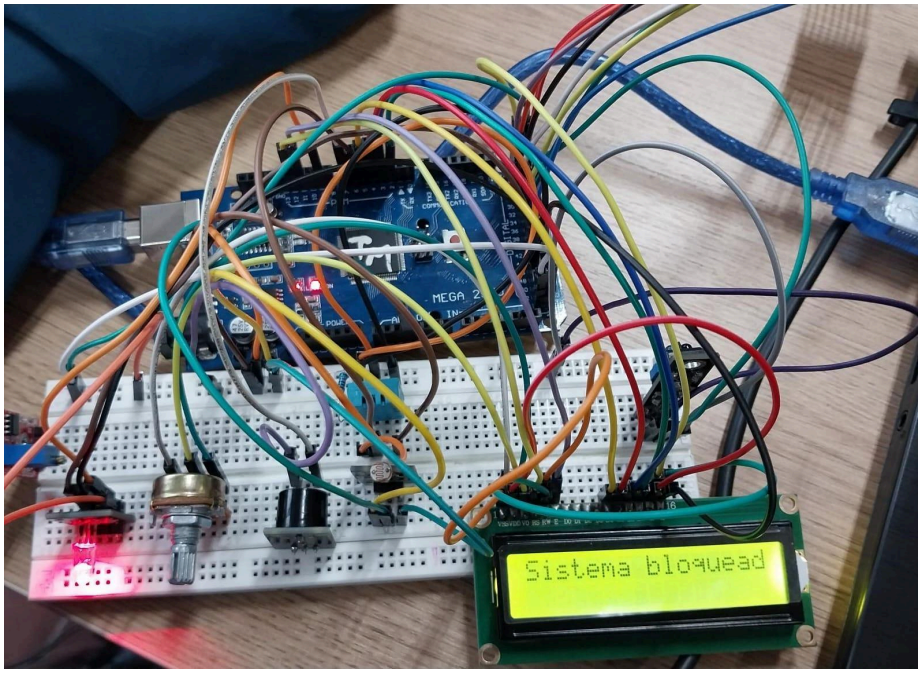
Fotografias evidencias (min 5)




Pht:1021Hm:69.00
TMP:24.80C







Problemas encontrados (min 4)

- Se encuentra un problema a la hora de hacer conexiones, a veces hubo puertos en el arduino o en la protoboard que no servían.
- Se detectó un problema en el entorno de desarrollo y pruebas Wokwi, ya que no estaban disponibles todos los componentes que se utilizarían para el desarrollo del proyecto.
- Se encontró un problema en los implementos que se prestaban, ya que algunos en varias ocasiones estaban dañados o presentaban problemas.

- Se encontró un problema con el arduino que en ocasiones generaba problemas en los portátiles en los cuales estaba conectado, generando un pantallazo azul haciendo apagar los equipos.

Link github:

https://github.com/Alexmax404/Proyecto-Arduino-AC