



USAC
TRICENTENARIA
Universidad de San Carlos de Guatemala

Control de Ambiente

Fase 1

Docente:	Erick Bernal y Samuel Perez
Curso:	Laboratorio de Arquitectura de Computadores y Ensambladores 2
Grupo:	7
Integrantes:	Kevin Estuardo Secaida Molina 201602404 Oscar Eduardo Morales Giron 201603028 Edwin Sandoval Lopez 202010856 Pedro Luis Pu Tavico 202000562

Fecha: 24 de febrero de 2024

Índice

Introducción.....	3
Objetivos.....	4
Funcionamiento.....	5
Sensores.....	5
Medición de temperatura.....	5
Cantidad de luz en el ambiente.....	6
Medición de calidad de aire.....	6
Medición de proximidad.....	6
Arduino.....	6
Interacción con sensores.....	6
Botones.....	7
Ejecutar accion.....	7
LCD.....	7
Mostrar datos.....	7
Usos.....	7
Beneficios.....	7
Impacto ambiental.....	8
Fragmentos de código:.....	8
Librerías incluidas:.....	8
Declaración de Funciones.....	8
Inicialización.....	9
Guardado y recuperación de datos de EEPROM:.....	9
Funciones para Manejar Interrupciones de Botones.....	10
Bocetos:.....	10
Boceto de prototipo físico:.....	10
Prototipos propuestos.....	11
Prototipo Simulado.....	11
Prototipo físico.....	11
Descripción de las capas del Smart Connected Design Framework.....	12
Diagramas de flujo de la información:.....	12
Flujo de datos desde los botones hasta los sensores:.....	12

Introducción

Una estación meteorológica de IoT, como parte integral del proyecto único descrito, se convierte en el núcleo de nuestro sistema de monitoreo ambiental en dormitorios inteligentes. Al integrar diversos sensores y utilizar tecnología IoT, esta estación recopila datos cruciales sobre variables climáticas como temperatura, humedad, iluminación y concentración de CO₂. Estos datos se transmiten en tiempo real a una plataforma centralizada, donde son analizados y utilizados para optimizar el control climático en el dormitorio. Además de su papel en el ámbito residencial, estas estaciones tienen aplicaciones valiosas en campos como la agricultura, la meteorología y la investigación científica, contribuyendo así a un mayor entendimiento y gestión de nuestro entorno.

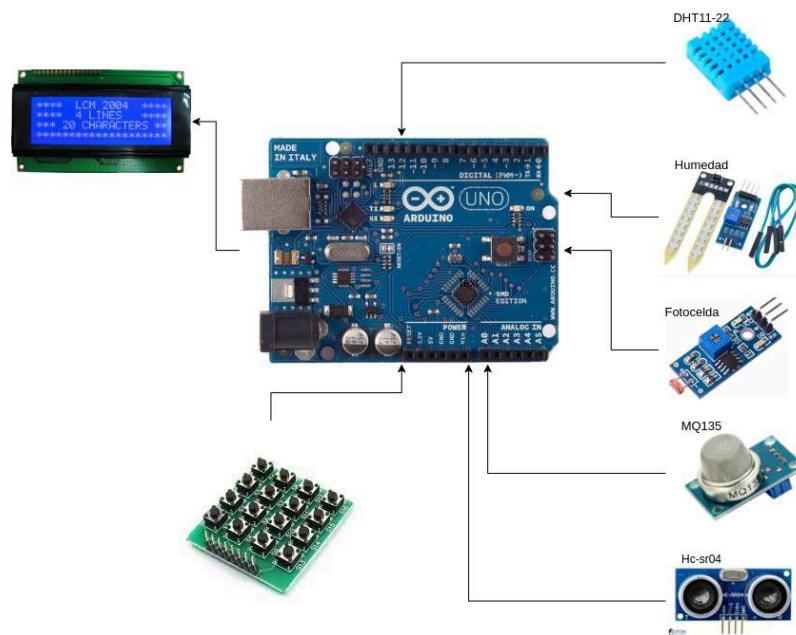
Objetivos

- Proporcionar una guía detallada para diseñar un dispositivo capaz de medir y registrar de manera precisa variables meteorológicas clave, como temperatura, humedad, iluminación y concentración de CO₂. Se enfocará en la selección adecuada de sensores y en la configuración óptima del hardware para garantizar una recolección de datos confiable y eficiente.
- Se busca capacitar al lector en el correcto uso e implementación del framework de IoT para el desarrollo de soluciones tecnológicas innovadoras. A través del manual, se proporcionará una comprensión sólida de los principios fundamentales del IoT y se guiará al lector en la aplicación práctica de estos conceptos para construir una estación meteorológica inteligente funcional.
- Enseñar cómo diseñar algoritmos eficaces de análisis de datos que interpreten la información meteorológica recopilada. Se enfocará en la creación de modelos predictivos para identificar patrones climáticos relevantes y optimizar el control climático en los dormitorios inteligentes. Los lectores aprenderán a utilizar los datos recolectados para tomar decisiones informadas y mejorar la calidad del ambiente interior.

Funcionamiento

El sistema consiste en una estación meteorológica IoT que recopila datos climáticos como temperatura, iluminación, humedad y concentración de CO₂ en el aire. Estos datos son procesados por un microcontrolador Arduino, que permite a los usuarios monitorear las condiciones climáticas en tiempo real y accionar dispositivos como un ventilador a través de botones físicos. Además, el microcontrolador almacena datos históricos en su memoria EEPROM para su posterior análisis.

A continuación se muestra el diagrama del funcionamiento lógico:



Sensores

Medición de temperatura

El dispositivo es capaz de medir la temperatura ambiente y proporcionar los datos precisos sobre la variación térmica, que existe en ese momento. Esto es posible a través del sensor de temperatura DHT11.



Cantidad de luz en el ambiente

El dispositivo es capaz de visualizar la cantidad de luz en el ambiente en una localidad determinada. Esto es posible a través del sensor de luz LDR1.



Medición de calidad de aire

El dispositivo es capaz de medir el CO₂ del ambiente para determinar la calidad del aire. Esto es posible a través del sensor MQ-135.



Medición de proximidad

El dispositivo es capaz de reconocer la proximidad de un objeto (persona), la acción se utilizará para comprobar la presencia o ausencia de un ser humano en la habitación. Esto es posible a través del sensor HC-SR04.



Arduino

Interacción con sensores

El dispositivo es capaz de comunicarse con los sensores a través de un arduino, para este dispositivo fue utilizado un Arduino Uno.



Botones

Ejecutar accion

Para controlar el dispositivo se implementaron 6 botones los cuales cuatro son para cada sensor, uno para el almacenamiento de datos en memoria y uno para mostrar los datos almacenados.



LCD

Mostrar datos

Los datos de cada sensor al presionar el botón correspondiente son mostrados en una pantalla LCD.



Usos

- Monitoreo en tiempo real de las condiciones climáticas en una habitación inteligente.
- Accionamiento remoto de dispositivos como un ventilador para mantener condiciones confortables.
- Análisis de datos históricos para identificar patrones climáticos y optimizar el control ambiental.

Beneficios

- Mejora del confort en el dormitorio al mantener condiciones climáticas óptimas.
- Ahorro energético al optimizar el uso de dispositivos de climatización.
- Facilita la visualización y comprensión de datos climáticos a través de una pantalla LCD.

Impacto ambiental

- Reducción del consumo energético al controlar eficientemente dispositivos de climatización.
- Mayor conciencia sobre las condiciones ambientales internas y su impacto en la calidad del aire.

Fragmentos de código:

Librerías incluidas:

Incluye las librerías necesarias para el manejo de dispositivos I2C, el sensor DHT11, la pantalla LCD I2C y el manejo de la memoria EEPROM.

```
#include <Wire.h>
#include <DHT.h>
#include <LiquidCrystal_I2C.h>
#include <EEPROM.h>
```

Declaración de Funciones

Se definen constantes para los pines utilizados. También se declaran variables para almacenar lecturas de sensores y estados de botones. Se declaran variables booleanas para indicar si se debe mostrar cada tipo de dato y para controlar las acciones de almacenamiento y recuperación de la EEPROM.

```
bool mostrarTemperatura = false;
bool mostrarLuminosidad = false;
bool mostrarCO2 = false;
bool mostrarProximidad = false;
volatile bool guardarEeprom = false;
volatile bool mostrarEeprom = false;
```

Inicialización

En el setup() se inicializan los dispositivos, se configuran los pines, se establecen las interrupciones y se configura la pantalla LCD.

```
void setup() {
    //Inicializaciones de los dispositivos.
    Serial.begin(9600);
    dht.begin();
    // Configuración de pines
    pinMode(trig, OUTPUT);
    pinMode(echo, INPUT);
    pinMode(MT, INPUT);
    pinMode(LA, INPUT);
    pinMode(MC, INPUT);
    pinMode(MP, INPUT);
    pinMode(GE, INPUT);
    pinMode(ME, INPUT);
    // Interrupciones
    attachInterrupt(digitalPinToInterrupt(GE), estado_guardarEeprom, FALLING);
    attachInterrupt(digitalPinToInterrupt(ME), estado_mostrarEeprom, FALLING);
    // Configuración de la LCD.
    lcd.init();
    lcd.backLight();
    lcd.print("ACE2 GRUPO 7");
    delay(2000); // Retraso de 2 segundos para salir del setup.
}
```

Guardado y recuperación de datos de EEPROM:

Estas funciones se encargan de guardar y recuperar datos en la memoria EEPROM del Arduino. Esto permite almacenar datos de forma permanente incluso después de apagar el dispositivo.

```
void guardarDatos(float dist_val, float lum_val, float co2_val, float temp_val, float hum_val) {
    int direccion = 0;
    EEPROM.put(direccion, dist_val);
    direccion += sizeof(float);
    EEPROM.put(direccion, lum_val);
    direccion += sizeof(float);
    EEPROM.put(direccion, co2_val);
    direccion += sizeof(float);
    EEPROM.put(direccion, temp_val);
    direccion += sizeof(float);
    EEPROM.put(direccion, hum_val);
}

void recuperarDatos(float& dist_val, float& lum_val, float& co2_val, float& temp_val, float& hum_val) {
    int direccion = 0;
    EEPROM.get(direccion, dist_val);
    direccion += sizeof(float);
    EEPROM.get(direccion, lum_val);
    direccion += sizeof(float);
    EEPROM.get(direccion, co2_val);
    direccion += sizeof(float);
    EEPROM.get(direccion, temp_val);
    direccion += sizeof(float);
    EEPROM.get(direccion, hum_val);
}
```

Funciones para Manejar Interrupciones de Botones

Estas funciones se ejecutan cuando se activan las interrupciones de los botones correspondientes y establecen las banderas para realizar las acciones de guardar y mostrar datos en la EEPROM.

```
void estado_guardarEeprom() {
    guardarEeprom = true;
    delay(50);
}

void estado_mostrarEeprom() {
    mostrarEeprom = true;
    delay(50);
}
```

Bocetos:

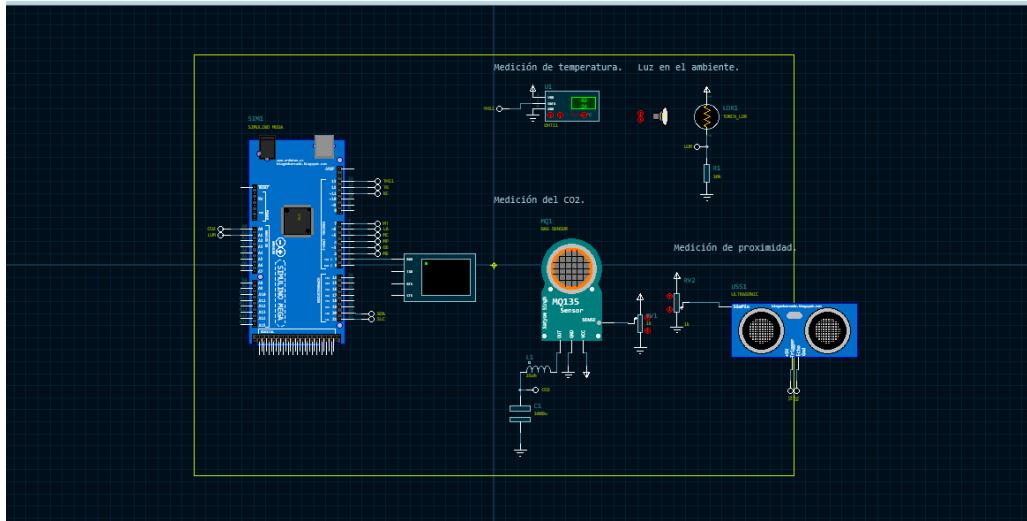
Boceto de prototipo físico:



El boceto representa el diseño del dispositivo físico el cual consiste en un microcontrolador Arduino conectado a sensores de temperatura, humedad, iluminación y CO₂.

Prototipos propuestos

Prototipo Simulado



El prototipo simulado consiste en un microcontrolador Arduino conectado a sensores de temperatura, humedad, iluminación y CO₂.

Prototipo físico



El prototipo físico consiste en un microcontrolador Arduino conectado a sensores de temperatura, humedad, iluminación y CO₂.

Descripción de las capas del Smart Connected Design Framework

1. Capa de Hardware: Incluye el microcontrolador Arduino, sensores de temperatura, humedad, iluminación y CO2.
2. Capa de Software: En esta fase se desarrollaron algoritmos para la adquisición de datos de los sensores y el envío de estos datos a la memoria EEPROM del Arduino.

Diagramas de flujo de la información:

Flujo de datos desde los botones hasta los sensores:

