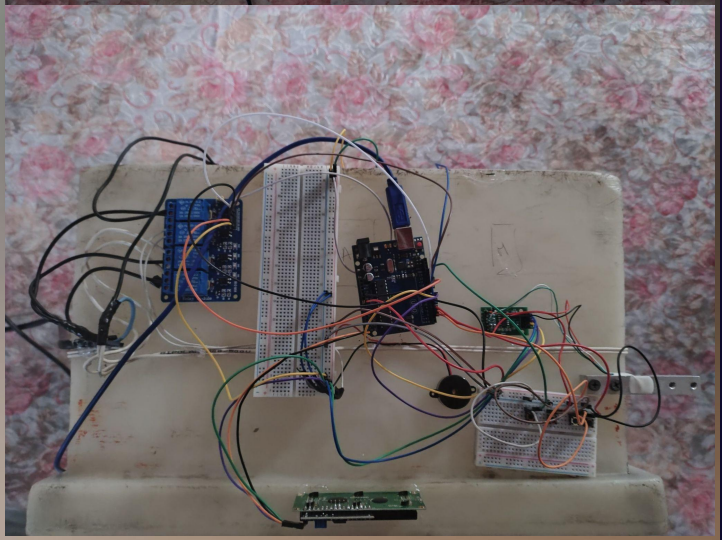


Prototipo de Asistente de Cocina Inteligente

integrante:
Uriel Gallardo(402550)
Valentino Rao(402308)
Mariano Condorí(406455)

Materia:
Informática 2

Grupo:
9

Introducción

En la actualidad, la cocina es un espacio que combina creatividad, precisión y tecnología. Sin embargo, muchos usuarios se enfrentan a desafíos al intentar mantener un control preciso de las preparaciones culinarias, como medir correctamente los ingredientes, calcular el contenido calórico, y gestionar los tiempos de cocción y descongelado.

El desarrollo de tecnologías avanzadas para la medición y control en la cocina se ha vuelto una prioridad en la industria de los electrodomésticos inteligentes, la integración de sensores de alta precisión y software de análisis en dispositivos domésticos está revolucionando la forma en que los usuarios interactúan con sus cocinas.

Mediante este proyecto, el problema central que se busca resolver es la falta de una herramienta integral en la cocina que combine múltiples funciones clave para la preparación de alimentos de manera precisa y eficiente.

Propuesta

1. ## Precisión en la Medición

   Medir correctamente los ingredientes es crucial para obtener resultados consistentes.

2. ## Gestión de Tiempos

   Controlar los tiempos de cocción y descongelado es esencial para evitar errores.

3. ## Control de Calorías

   Calcular el contenido calórico de las comidas es fundamental para una alimentación saludable.

4. ## Tendencias Alimentarias

   El interés por la alimentación saludable y las dietas personalizadas está en aumento.

Desarrollo de un asistente de cocina inteligente

Esté basado en microcontroladores y programación avanzada, este dispositivo combinará varias funciones en una sola unidad compacta:

1 Medición de Temperatura y Masa

Sensores integrados para medir la temperatura del ambientes y la masa de los ingredientes.

2 Cálculo de Calorías

Un módulo de software que calculará el contenido calórico de las comidas basadas en los ingredientes y las cantidades.

3 Recuento de Comidas y Control de Macronutrientes

Registro automático de las comidas elaboradas, con la capacidad de almacenar recetas diarias.

4 Medición de Humedad

Sensor para monitorear la humedad en la preparación de panes y masas, asegurando una fermentación y cocción adecuadas.

5 Pantalla Integrada

Una pantalla mostrará datos esenciales como la temperatura, tiempos de cocción, calorías, y estado de los alimentos.

7 Reloj y Temporizador

Gestión de tiempos de cocción y descongelado, avisando al usuario cuando un alimento ha alcanzado el punto deseado.

Beneficios del Asistente

Este desarrollo no solo facilitará las tareas diarias en la cocina, sino que también promoverá una alimentación más saludable y segura.

Facilidad de Uso

El asistente de cocina simplifica las tareas culinarias, ofreciendo una experiencia de usuario más cómoda y eficiente.

Alimentación Saludable

El control de calorías y macronutrientes ayuda a los usuarios a tomar decisiones más saludables.

Seguridad Alimentaria

La medición precisa de la temperatura y la humedad garantiza la seguridad de los alimentos.

Elementos del Proyecto

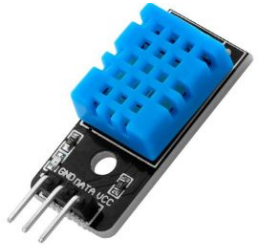
El asistente de cocina inteligente se compone de varios elementos electrónicos que trabajan en conjunto.

Buzzer

Genera señales acústicas para alertar al usuario.

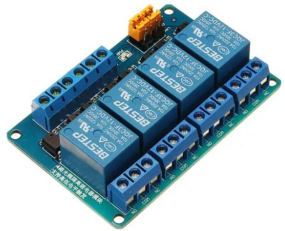
Sensor de Temperatura y Humedad DHT11

Mide la temperatura y la humedad del ambiente.



Modulo Relay de 4 Canales

Controlado por el microcontrolador para realizar movimientos precisos.



Pantalla LCD 16x02

La pantalla LCD muestra información esencial al usuario.

Temperatura	Tiempos de cocción	Calorías
Estado de los alimentos	Recetas	Información adicional

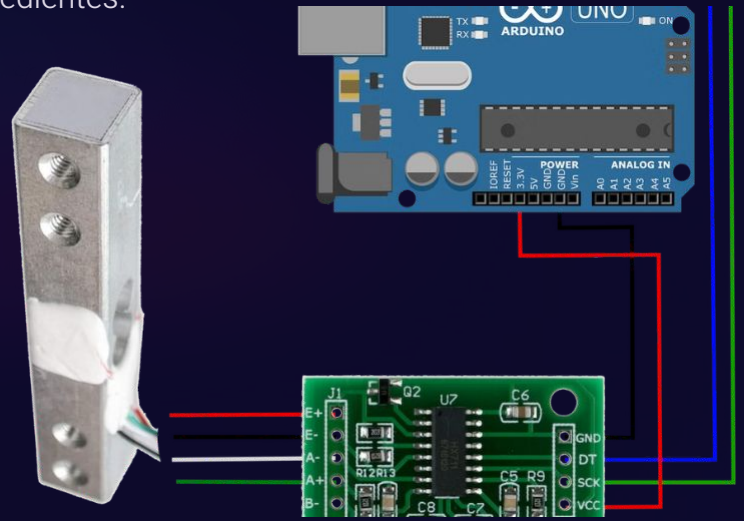
Módulo I2C

Facilita la comunicación.

Recibe y envia datos, efectivo en cuanto a cantidad de cables y disponibilidad de puertos.

Celda Galvánica y Amplificador Hx711

La celda galvánica mide la masa de los ingredientes.



Sensor de Carga

Convierte la fuerza aplicada en una señal eléctrica, amplificada mediante el módulo Hx711.

Códigos: Nutrición.h

```
class PuertoSerial {
    public:
        PuertoSerial(const std::string& puerto, int tasaBaudios) {
            serial_port = open(puerto.c_str(), O_RDWR);

            if (serial_port < 0) {
                std::cerr << "Error al abrir el puerto serial" << std::endl;
                exit(1);
            }

            struct termios tty;
            if(tcgetattr(serial_port, &tty) != 0) {
                std::cerr << "Error al obtener los atributos del puerto serial" << std::endl;
                exit(1);
            }

            cfsetispeed(&tty, tasaBaudios);
            cfsetospeed(&tty, tasaBaudios);

            tty.c_cflag &= ~PARENB;
            tty.c_cflag &= ~CSTOPB;
            tty.c_cflag &= ~CSIZE;
            tty.c_cflag |= CS8;

            tty.c_cflag &= ~CRTSCTS;
            tty.c_cflag |= CREAD | CLOCAL;

            tty.c_lflag &= ~ICANON;
            tty.c_lflag &= ~ECHO;
            tty.c_lflag &= ~ECHOE;
            tty.c_lflag &= ~ECHONL;
            tty.c_lflag &= ~ISIG;
            tty.c_iflag &= ~(IXON | IXOFF | IXANY);
            tty.c_iflag &= ~(ICRNL | INLCR);
            tty.c_oflag &= ~OPOST;
            tty.c_oflag &= ~ONLCR;
            tty.c_cc[VTIME] = 10;
            tty.c_cc[VMIN] = 0;

            if(tcsetattr(serial_port, TCSANOW, &tty) != 0) {
                std::cerr << "Error al configurar el puerto serial" << std::endl;
                exit(1);
            }
        }

        ~PuertoSerial() {
            close(serial_port);
        }

    std::string leerDatos() {
        std::string resultado;
        char bufferLectura[256];
        while (true) {
            memset(&bufferLectura, '\0', sizeof(bufferLectura));
            int bytesLeidos = read(serial_port, &bufferLectura, sizeof(bufferLectura) - 1);

            if (bytesLeidos < 0) {
                std::cerr << "Error al leer del puerto serial" << std::endl;
                return "";
            }

            resultado.append(bufferLectura, bytesLeidos);

            if (resultado.find('\n') != std::string::npos) {
                break;
            }
        }

        return resultado;
    }
    private:
        int serial_port;
```

Códigos: mainnutricion.cpp

```cpp
int main() {
    // Crear instancia de Persona con requerimientos diarios
    Persona usuario(2000, 50, 300, 70);

    // Leer datos de alimentos desde un archivo
    std::vector<DatosAlimento> datosAlimentos = leerDatosDeArchivo("alimentos.txt");

    // Mostrar los datos de alimentos leídos
    for (const auto& datos : datosAlimentos) {
        std::cout << "Alimento: " << datos.alimento << ", Peso: " << datos.peso << "g" << std::endl;
    }

    // Calcular y mostrar información nutricional
    InformacionNutricional totalNutricional = {0, 0, 0, 0};

    for (const auto& datos : datosAlimentos) {
        InformacionNutricional info = calcularInformacionNutricional(datos.alimento, datos.peso);

        totalNutricional.calorias += info.calorias;
        totalNutricional.proteinas += info.proteinas;
        totalNutricional.carbohidratos += info.carbohidratos;
        totalNutricional.lipidos += info.lipidos;

        std::cout << "Alimento: " << datos.alimento << std::endl;
        std::cout << "  Calorías: " << info.calorias << " kcal" << std::endl;
        std::cout << "  Proteínas: " << info.proteinas << " g" << std::endl;
        std::cout << "  Carbohidratos: " << info.carbohidratos << " g" << std::endl;
        std::cout << "  Lípidos: " << info.lipidos << " g" << std::endl;
    }

    // Comparar con requerimientos diarios del usuario
    std::cout << "\nResumen Total:\n";
    std::cout << "Calorías Totales: " << totalNutricional.calorias << " / " << usuario.obtenerCaloriasDiarias() << " kcal\n";
    std::cout << "Proteínas Totales: " << totalNutricional.proteinas << " / " << usuario.obtenerProteinasDiarias() << " g\n";
    std::cout << "Carbohidratos Totales: " << totalNutricional.carbohidratos << " / " << usuario.obtenerCarbohidratosDiarios() << " g\n";
    std::cout << "Lípidos Totales: " << totalNutricional.lipidos << " / " << usuario.obtenerLipidosDiarios() << " g\n";

    // Mostrar recetas para los alimentos leídos
    std::cout << "\nRecetas:\n";
    for (const auto& datos : datosAlimentos) {
        mostrarReceta(datos.alimento);
        std::cout << std::endl;
    }

    return 0;
```

Códigos: mainarduino.ino

```
void displayMenu() {
  if (inMenu) {
    lcd.clear();
    lcd.print("Opcion: ");
    lcd.print(currentOption + 1);

    if(currentOption > 1){

      lcd.clear();
      lcd.print("Comida: ");
      lcd.print(currentOption - 1);

      if(currentOption == 2){
        lcd.setCursor(0, 1);
        lcd.print("Pollo al Horno");
      }

      if(currentOption == 3){
        lcd.setCursor(0, 1);
        lcd.print("Pescado al Vapor");
      }

      if(currentOption == 4){
        lcd.setCursor(0, 1);
        lcd.print("Pizza Casera");
      }
    }
  }
}
```

```
void CalentarHorno(int temp, float Temp) {
    while(temp < Temp){

    temp= dht.readTemperature();

    lcd.clear();
    lcd.print("Calentando Horno");
    delay(1000);
    lcd.clear();

    lcd.print("Temp: ");
    lcd.print(temp);
    delay(1000);

    digitalWrite(RELAY1, LOW);
    digitalWrite(RELAY2, LOW);
  }

  if(temp > Temp){
    lcd.clear();
    lcd.print("Horno Caliente");
    delay(2000);
    digitalWrite(RELAY3, LOW);
    delay(1000);
    digitalWrite(RELAY3, HIGH);
  }
}
```

```
void CocinarOpcion1(){

  float temp = dht.readTemperature();
  float humidity = dht.readHumidity();

  lcd.clear();

  lcd.print("Pesar Pollo: ");
  delay(5000);
  static float pollo = 0;
  pollo = scale.get_units(1);
  lcd.clear();
  lcd.print("Pollo: ");
  lcd.print(pollo, 1);
  lcd.print(" g");
  delay(5000);

  lcd.clear();
  lcd.print("Pesar papas: ");
  delay(5000);
  static float papas = 0;
  papas = scale.get_units(1);
  lcd.clear();
  lcd.print("Papas: ");
  lcd.print(papas, 1);
  lcd.print(" g");
  delay(5000);

  pollo += papas;
  Serial.print("Pollo,");
  Serial.println(pollo);

  lcd.clear();
  lcd.print("MOMENTO PARA");
  lcd.setCursor(0, 1);
  lcd.print("CANCELAR");
  delay(10000);
```

```
while (millis() - startMillis < 150000) {

  float humidity_1 = dht.readHumidity();
  float temp1 = dht.readTemperature();

  if(controlluz == 0){
    digitalWrite(RELAY1, LOW);
    digitalWrite(RELAY2, LOW);
  }

  if (controlluz == 1) {
    digitalWrite(RELAY2, HIGH);
    digitalWrite(RELAY1, LOW);
  }

  if (controlmotor == 1) {
    digitalWrite(RELAY4, LOW);
  }

  if (controlmotor == 0) {
    digitalWrite(RELAY4, HIGH);
  }

  lcd.clear();
  delay(100);
  lcd.print("Temp: ");
  lcd.print(temp1);
  lcd.setCursor(0, 1);
  lcd.print("Humedad: ");
  lcd.print(humidity_1);
  lcd.setCursor(0, 1);
  delay(5000);
  lcd.clear();
  lcd.print("Cocinando...");
  delay(1000);
```