UNIVERSIDAD CATÓLICA BOLIVIANA "SAN PABLO"
UNIDAD ACADÉMICA REGIONAL COCHABAMBA
Departamento de Ingeniería y Ciencias Exactas
Internet de las Cosas (P1)

Practica I
Integración de sensores y actuadores en un objeto inteligente

Iturri Vargas Santiago Andres
Saavedra Lopez Santiago
Mgr. Marín García Eduardo Enrique

Cochabamba – Bolivia
Agosto de 2024

1. Introducción

1.1. Objetivo

El objetivo de este proyecto es diseñar e implementar un sistema de control basado en un microcontrolador ESP32, que utiliza un sensor ultrasónico para medir la distancia y activa un conjunto de LEDs según los umbrales de distancia predefinidos. El sistema tiene como propósito proporcionar una señal visual intuitiva mediante el parpadeo de los LEDs, lo cual se basa en la proximidad de un objeto detectado por el sensor ultrasónico. Esta implementación busca ser eficiente en términos de manejo del tiempo, evitando el uso de funciones de retardo bloqueante (delay()), y en su lugar utilizando una estructura orientada a objetos que optimiza la modularidad y reutilización del código.

1.2. Descripción de sistema

El sistema desarrollado es un prototipo de detección de proximidad que utiliza un sensor ultrasónico y un conjunto de tres LEDs para indicar la distancia de un objeto. El corazón del sistema es un microcontrolador ESP32, que se encarga de procesar los datos obtenidos del sensor y activar los LEDs correspondientes.

Componentes del Sistema:

1. **Sensor Ultrasónico (HC-SR04):** Este sensor mide la distancia de un objeto en función del tiempo que tarda un pulso ultrasónico en regresar al sensor después de rebotar en el objeto. Los pines de control del sensor (TRIG y ECHO) están conectados al ESP32.
2. **LEDs:** Se utilizan tres LEDs de colores (rojo, amarillo y verde), cada uno conectado a un pin digital del ESP32. Estos LEDs indican la proximidad del objeto al sensor:
    - **LED Rojo:** Parpadea cuando la distancia es menor a 10 cm.
    - **LED Amarillo:** Parpadea cuando la distancia está entre 10 cm y 40 cm.
    - **LED Verde:** Parpadea cuando la distancia está entre 40 cm y 80 cm.
3. **ESP32:** Es el microcontrolador encargado de leer la información del sensor ultrasónico y gestionar el encendido y parpadeo de los LEDs. El código está diseñado utilizando programación orientada a objetos, con clases dedicadas para el sensor ultrasónico y el control de los LEDs.

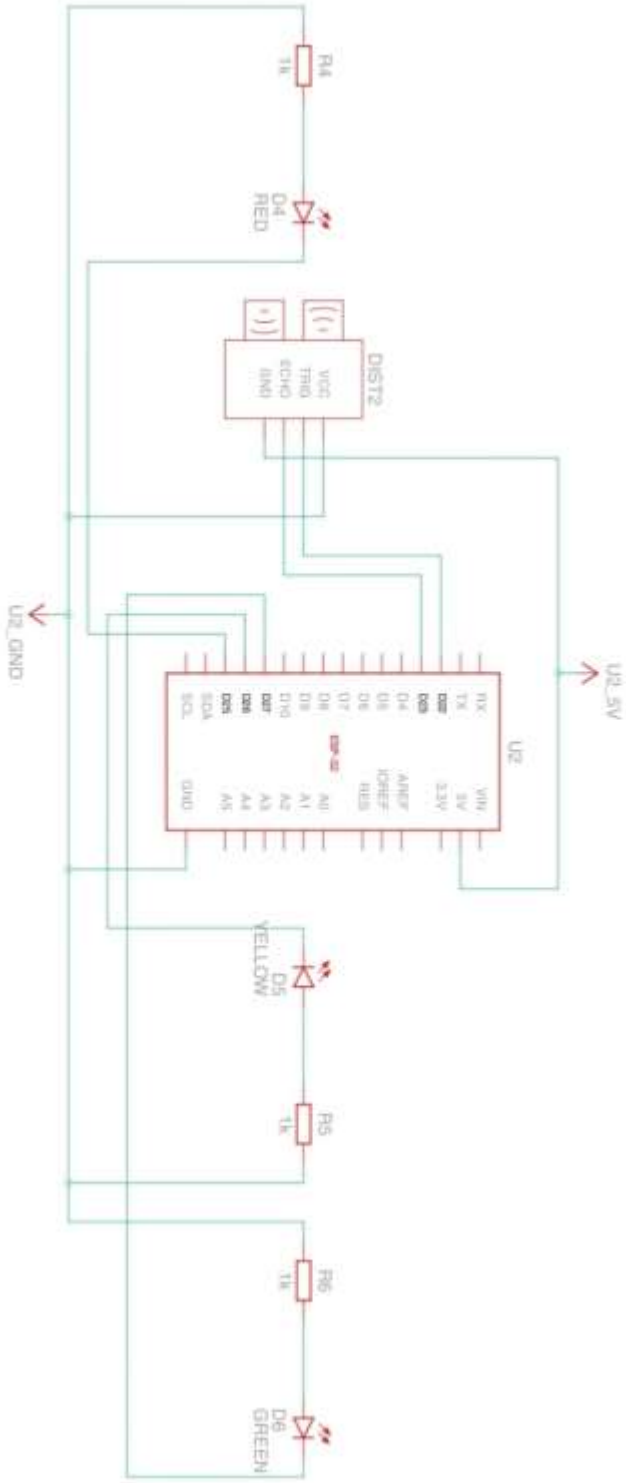
Funcionamiento del Sistema:

1. El ESP32 envía un pulso al sensor ultrasónico para iniciar la medición de distancia.
2. El sensor devuelve el tiempo que tardó el pulso en regresar, el cual es procesado para calcular la distancia.
3. Dependiendo de la distancia medida, el ESP32 selecciona el LED que debe parpadear.
4. El LED correspondiente parpadea a intervalos regulares, mientras que los otros LEDs permanecen apagados.

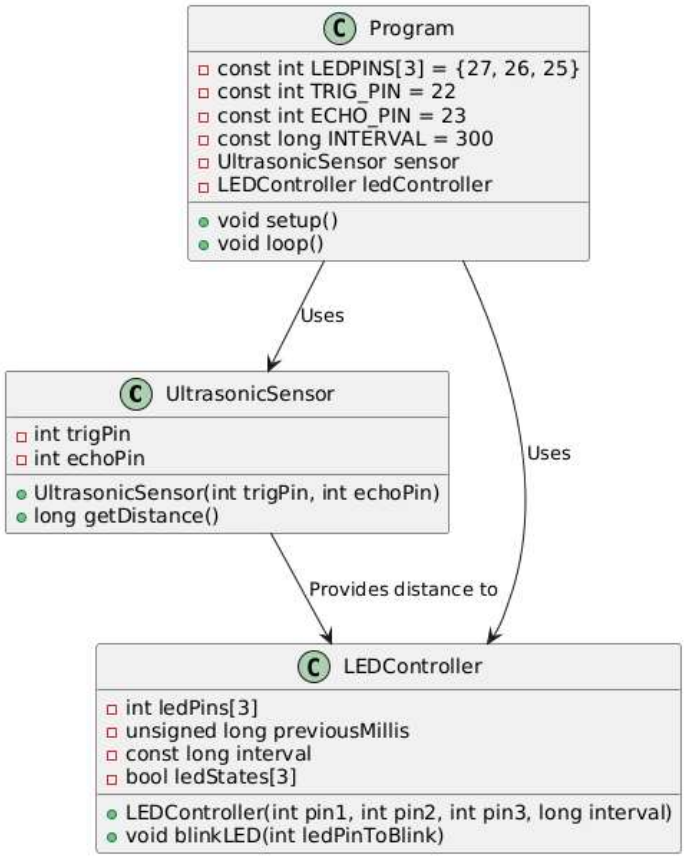
El sistema está diseñado para ser eficiente en términos de gestión del tiempo, utilizando la función millis() en lugar de delay() para manejar el parpadeo de los LEDs, lo que permite que el microcontrolador siga ejecutando otras tareas mientras los LEDs parpadean.

2. Análisis y Diseño

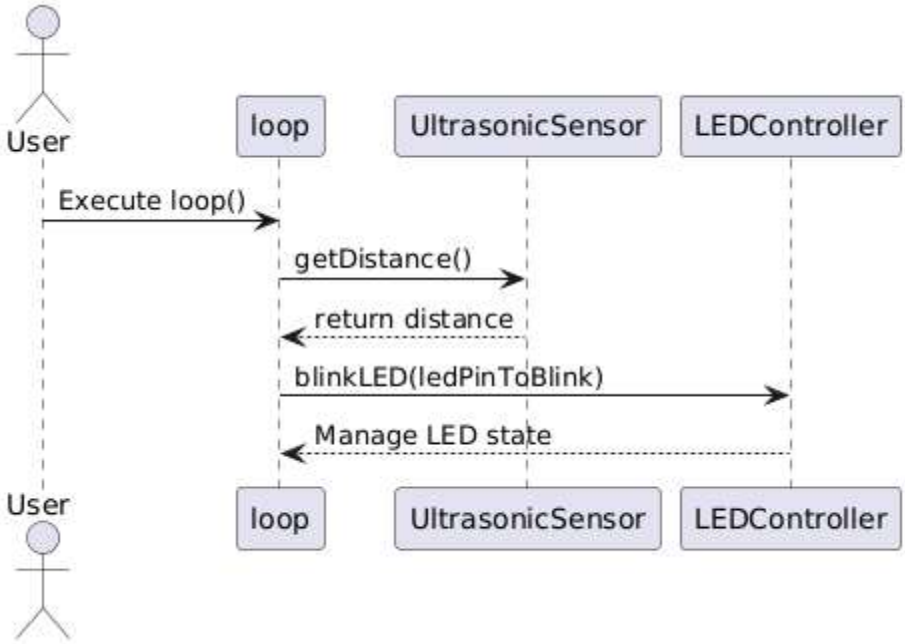
2.1. Diagramas de Circuito



2.2. Diagramas de Estructuras/Arquitecturas



2.3. Diagramas de Comportamiento



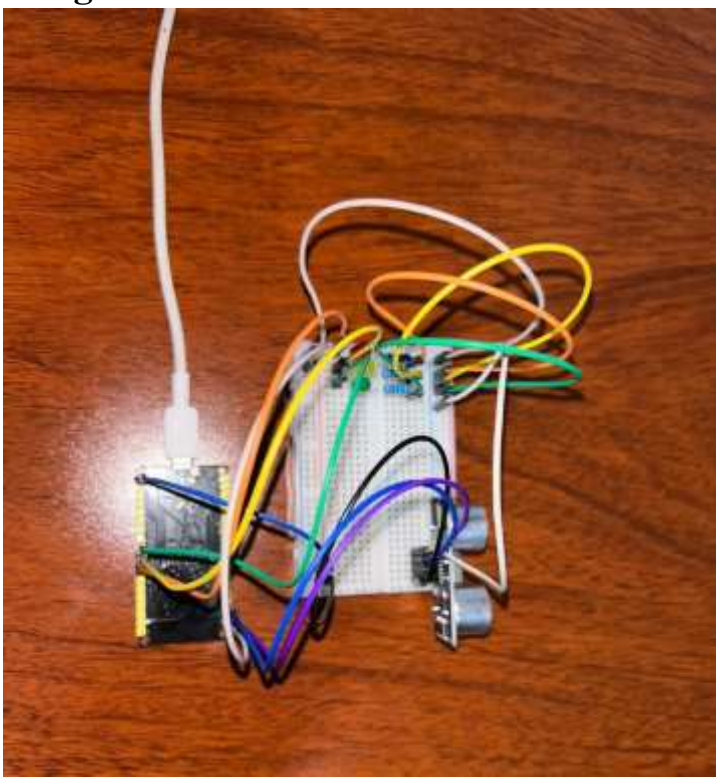
3. Desarrollo e implementación

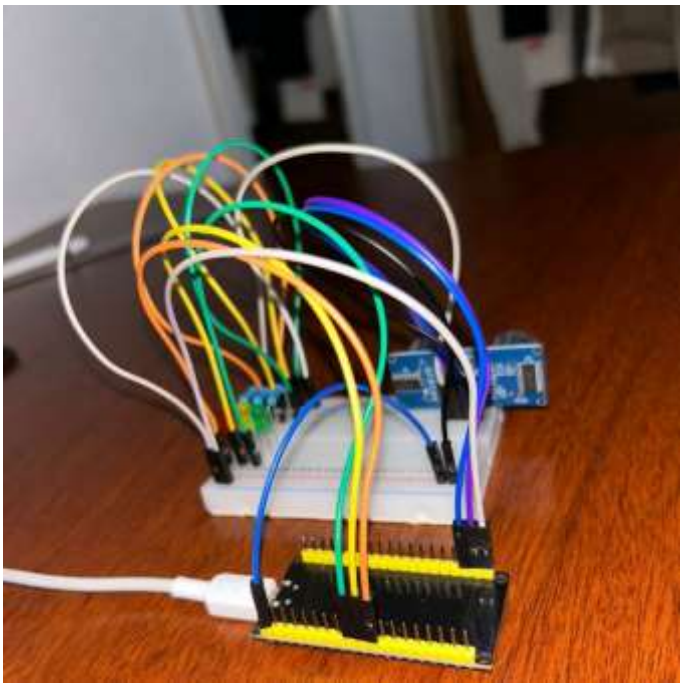
3.1. Código fuente

```
class UltrasonicSensor {
  private:
    int trigPin;
    int echoPin;

  public:
    UltrasonicSensor(int trigPin, int echoPin) {
      this->trigPin = trigPin;
      this->echoPin = echoPin;
      pinMode(trigPin, OUTPUT);
      pinMode(echoPin, INPUT);
    }

    long getDistance() {
      digitalWrite(trigPin, LOW);
      delayMicroseconds(2);
      digitalWrite(trigPin, HIGH);
      delayMicroseconds(10);
      digitalWrite(trigPin, LOW);

      long duration = pulseIn(echoPin, HIGH);
      return duration * 0.034 / 2;
    }
};

class LEDController {
  private:
    int ledPins[3];
    unsigned long previousMillis;
    const long interval;
    bool ledStates[3];

  public:
    LEDController(int pin1, int pin2, int pin3, long interval)
      : ledPins{pin1, pin2, pin3}, interval(interval), previousMillis(0) {
      for (int i = 0; i < 3; i++) {
        pinMode(ledPins[i], OUTPUT);
        ledStates[i] = LOW;
      }
    }

    void blinkLED(int ledPinToBlink) {
```

```
      unsigned long currentMillis = millis();
      if (currentMillis - previousMillis >= interval) {
        previousMillis = currentMillis;

        for (int i = 0; i < 3; i++) {
          if (ledPins[i] == ledPinToBlink)
            ledStates[i] = !ledStates[i];
          else
            ledStates[i] = LOW;
          digitalWrite(ledPins[i], ledStates[i]);
        }
      }
    }
};
Green yellow red
const int LEDPINS[] = {27, 26, 25};
const TRIG_PIN = 22;
const ECHO_PIN = 23;
const INTERVAL = 300;
UltrasonicSensor sensor(TRIG_PIN, ECHO_PIN);
LEDController ledController(LEDPINS[0], LEDPINS[1], LEDPINS[2], INTERVAL);

void setup() {
}

void loop() {
  long distance = sensor.getDistance();
  int ledPinToBlink = -1;

  if (distance < 10)
    ledPinToBlink = LEDPINS[2];
  else if (distance < 25)
    ledPinToBlink = LEDPINS[1];
  else if (distance < 40)
    ledPinToBlink = LEDPINS[0];

  ledController.blinkLED(ledPinToBlink);
}
```

3.2. Imágenes del circuito





4. Pruebas y validaciones

4.1. Documentación de errores

Para la documentación de errores realizamos 10 pruebas de 3 medidas distintas y sacamos los errores

Distancia Real (12 cm)			
N	**Distancia Medida**	**E. Abs**	**E. %**
1	11.44	0.56	0.0467
2	11.73	0.27	0.0225
3	11.75	0.25	0.0208
4	11.75	0.25	0.0208
5	12.04	0.04	0.0033
6	12.15	0.15	0.0125
7	12.43	0.43	0.0358
8	12.61	0.61	0.0508
9	12.61	0.61	0.0508
10	12.61	0.61	0.0508

Distancia Real (21 cm)			
N	**Distancia Medida**	**E. Abs**	**E. %**
1	20.45	0.55	0.0262
2	20.77	0.23	0.0110
3	20.77	0.23	0.0110
4	20.77	0.23	0.0110
5	21.06	0.06	0.0029
6	21.06	0.06	0.0029
7	21.13	0.13	0.0062
8	21.33	0.33	0.0157
9	21.33	0.33	0.0157
10	21.35	0.35	0.0167

Distancia Real (21 cm)			
N	**Distancia Medida**	**E. Abs**	**E. %**
1	32.3	0.7	0.0212
2	32.56	0.44	0.0133
3	32.56	0.44	0.0133
4	32.81	0.19	0.0058
5	33.06	0.06	0.0018
6	33.06	0.06	0.0018
7	33.32	0.32	0.0097
8	33.32	0.32	0.0097
9	33.59	0.59	0.0179
10	33.59	0.59	0.0179

5. Resultados y Conclusiones

5.1. Resultados

5.1.1. Para la Distancia Real de 12 cm:
La distancia medida varió entre 11.44 cm y 12.61 cm.
El error absoluto (E. Abs) varió de 0.04 cm a 0.61 cm.
El error porcentual (E. %) varió de 0.0033% a 0.0508%.

5.1.2. Para la Distancia Real de 21 cm (primera columna):
La distancia medida varió entre 20.45 cm y 21.35 cm.
El error absoluto (E. Abs) varió de 0.06 cm a 0.55 cm.
El error porcentual (E. %) varió de 0.0029% a 0.0262%.

5.1.3. Para la Distancia Real de 33 cm:
La distancia medida varió entre 32.3 cm y 33.59 cm.
El error absoluto (E. Abs) varió de 0.06 cm a 0.7 cm.
El error porcentual (E. %) varió de 0.0018% a 0.0212%.

5.2. Conclusiones

5.2.1. Precisión de Medición:

A medida que la distancia real aumenta, el error absoluto también tiende a aumentar, aunque el error porcentual generalmente disminuye. Esto sugiere que, aunque el sensor mantiene una cierta precisión relativa, la desviación en términos absolutos es más notable a distancias mayores.

5.2.2. Consistencia de los Errores:

Los errores absolutos y porcentuales son más pequeños cuando la distancia real es mayor, lo que indica que el sensor es más preciso a mayores distancias en términos relativos.

Para la distancia de 12 cm, el error porcentual fue el más alto, especialmente en las mediciones más alejadas de la distancia real (con un error máximo de 0.0508%).

5.2.3. Implicaciones para Aplicaciones Prácticas:

En aplicaciones donde se requiere alta precisión en distancias cortas (como 12 cm), se debe tener en cuenta el margen de error relativamente mayor.

Para distancias mayores, el sensor proporciona lecturas más consistentes y con menor error porcentual, lo que lo hace más adecuado para aplicaciones donde se midan distancias más largas.

6. Referencias

https://www.arduino.cc/
https://www.tinkercad.com/things/h5mNu0Xizdq-iot-1/editel
https://www.electroschematics.com/hc-sr04-datasheet/