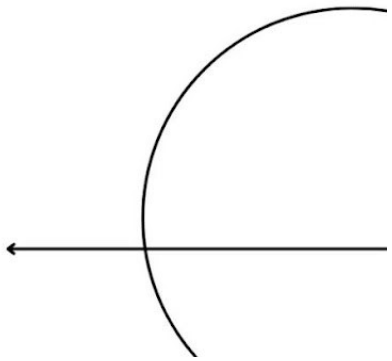
COA, LAB 1 REPORT

GROUP MEMBERS

- 150467 - KITHEKA LIZALOUREEN
- 166981 - EESHAN VAGHJIANI
- 145200 - AHMED LUKMAN
- 169688 - PAUL OBONYO
- 169768 - NATASHA KEMUNTO
- 169957 - KITTO JOSHUA
- 150647 Stephen Ndaba

2024



Lab Report: Distance Measurement and Feedback System

Introduction

This lab explores the integration of an ultrasonic sensor with an Arduino Uno to measure distance and provide feedback through LEDs and a buzzer. The system utilizes basic components like LEDs, a buzzer, and an LCD for displaying distance measurements.

Objectives

1. Interface an ultrasonic sensor with Arduino.

2. Measure distance in centimetres and inches.

3. Display distance on an LCD.

4. Control LEDs based on distance thresholds.

5. Emit varying tones from a buzzer based on distance.

Setup

- Connect HC-SR04 sensor to Arduino (trig to pin 2, echo to pin 3).

- LEDs (Red, Yellow, Green) connected to pins 4, 5, and 6 respectively.

- Buzzer connected to pin 7.

- LCD initialized via I2C for distance display.

Execution

- Sensor triggers ultrasonic pulses; echo duration calculates distance.

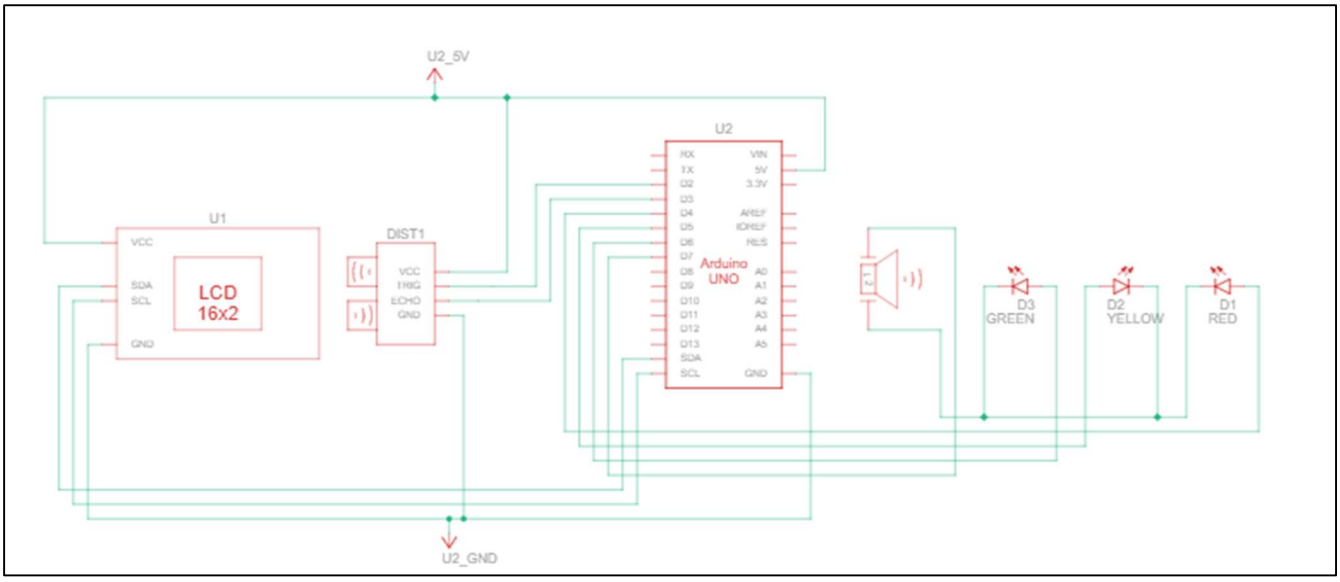
- LCD updates real-time distance in cm and inch.

- LEDs indicate proximity thresholds:
Red: <10 cm
Yellow: 10-20 cm
Green: >20 cm
Buzzer emits frequencies corresponding to distance.

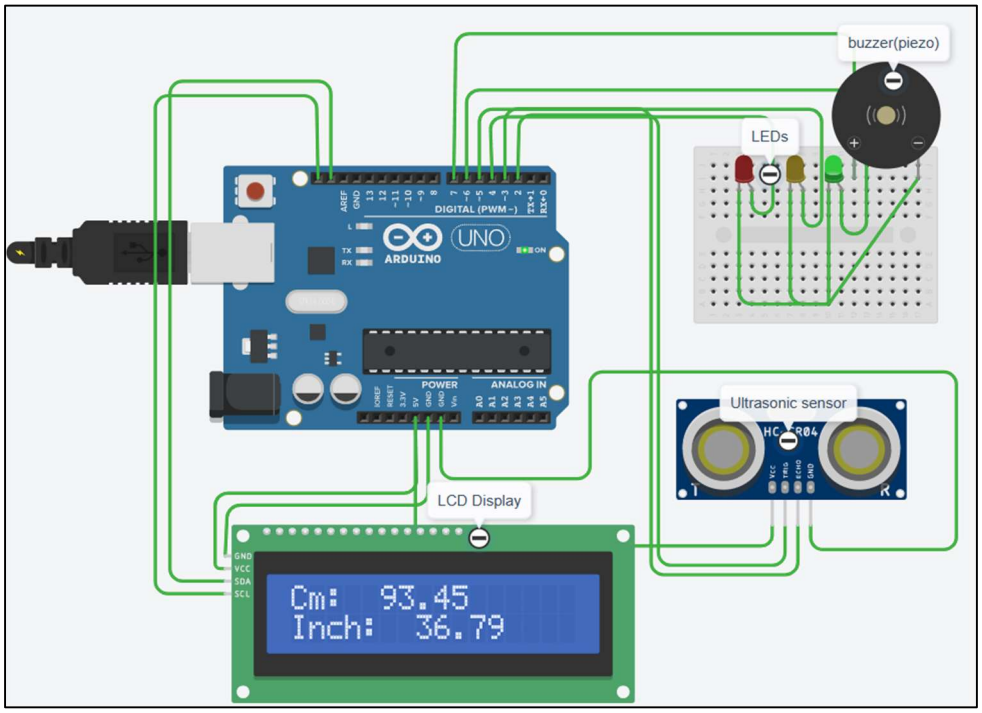
Conclusion

This lab demonstrated effective sensor integration with Arduino for distance measurement, providing visual and auditory feedback. It enhanced our understanding of microcontroller programming and sensor applications in practical scenarios.

SCHEMATIC DIAGRAM



Circuit View



code:

```
#include <LiquidCrystal_I2C.h>

// Define I2C address and LCD size
LiquidCrystal_I2C lcd(0x27, 16, 2);

// Define pin connections
int trig = 2;
int echo = 3;
int redLED = 4;
int yellowLED = 5;
int greenLED = 6;
int buzzer = 7;
long duration;
float distance_cm, distance_inch;
float previous_distance_cm = 0.0;
unsigned long previousMillis = 0;
const long interval = 100; // Interval for checking distance in milliseconds
const float threshold = 0.5; // Minimum change in distance to update display

void setup() {
  pinMode(trig, OUTPUT);
  pinMode(echo, INPUT);
  pinMode(redLED, OUTPUT);
  pinMode(yellowLED, OUTPUT);
  pinMode(greenLED, OUTPUT);
  pinMode(buzzer, OUTPUT);
```

```
  lcd.init();
  lcd.clear();
  lcd.backlight();

  // Print static text
  lcd.setCursor(0, 0);
  lcd.print("Cm: ");

  lcd.setCursor(0, 1);
  lcd.print("Inch: ");

  delay(2000);
}

void loop() {
  unsigned long currentMillis = millis();

  if (currentMillis - previousMillis >= interval) {
    // Save the last time you checked the distance
    previousMillis = currentMillis;

    // Trigger the ultrasonic sensor
    digitalWrite(trig, LOW);
    delayMicroseconds(2);
    digitalWrite(trig, HIGH);
    delayMicroseconds(10);
```

```
digitalWrite(trig, LOW);

// Read the duration of the echo pulse
duration = pulseIn(echo, HIGH);

// Calculate the distance in centimeters and inches
distance_cm = (duration / 2.0) * 0.0346;
distance_inch = distance_cm / 2.54;

// Check if the distance has changed significantly
if (abs(distance_cm - previous_distance_cm) > threshold) {
  // Format the distances to two decimal places
  char buffer_cm[10];
  dtostrf(distance_cm, 6, 2, buffer_cm);
  char buffer_inch[10];
  dtostrf(distance_inch, 6, 2, buffer_inch);

  // Clear previous distance values by printing spaces
  lcd.setCursor(4, 0);
  lcd.print("      ");
  lcd.setCursor(6, 1);
  lcd.print("      ");

  // Display the updated distances on the LCD
  lcd.setCursor(4, 0);
  lcd.print(buffer_cm);
```

```
  // Update the previous distance
  previous_distance_cm = distance_cm;
}

// Control the LEDs based on the distance
if (distance_cm < 10) {
  digitalWrite(redLED, HIGH);
  digitalWrite(yellowLED, LOW);
  digitalWrite(greenLED, LOW);
} else if (distance_cm < 20) {
  digitalWrite(redLED, LOW);
  digitalWrite(yellowLED, HIGH);
  digitalWrite(greenLED, LOW);
} else {
  digitalWrite(redLED, LOW);
  digitalWrite(yellowLED, LOW);
  digitalWrite(greenLED, HIGH);
}

// Control the buzzer frequency based on the distance
int buzzerFrequency = map(distance_cm, 0, 100, 1000, 100);
if (distance_cm < 100) {
  tone(buzzer, buzzerFrequency);
} else {
  noTone(buzzer);
}
```

OBSERVATION

- **Distance Measurement**: The system accurately measures distances up to 100 cm and displays them on the LCD in both centimetres and inches.

- **LED Indicators**:

Red LED consistently illuminates for distances less than 10 cm

Yellow LED responds accurately for distances between 10 and 20 cm.

Green LED lights up correctly for distances greater than 20 cm.

- **Buzzer Feedback**: The buzzer emits varying frequencies based on distance, with higher frequencies for closer distances and lower frequencies for further distances.

- **Responsiveness**: The system updates the LCD display and LED/buzzer feedback promptly, reflecting changes in distance measurements effectively.

- **Stability**: The system remains stable and responsive over extended periods of operation, with no significant drift or error in distance measurement.

BONUS TASK- JUMPMAN GAME

Introduction

This lab introduces the creation of a simple game using an Arduino Uno, an LCD display, and a push button. The game involves moving a character to avoid falling stones, displayed on a 16x2 LCD screen. This project combines elements of microcontroller programming, custom character creation, and basic game logic.

Objectives

1. Develop a simple game using an Arduino Uno.
2. Utilize an LCD display to show game elements.
3. Implement game controls using a push button.
4. Create custom characters for the LCD display.
5. Track and display the player's score and lives.

Setup

- **Components**:
    - Arduino Uno
    - 16x2 LCD display with I2C interface
    - Push button
    - Breadboard and jumper wires
- **Connections**:
    - Connect the LCD display to the Arduino using the I2C interface (SDA to A4, SCL to A5).
    - Connect the push button to digital pin 2

Execution

1. Initialize LCD and Custom Characters

- Define custom characters for stones, the hero with arms up and down, explosions, and a heart symbol.
- Initialize the LCD display and create custom characters in the `setup()` function.

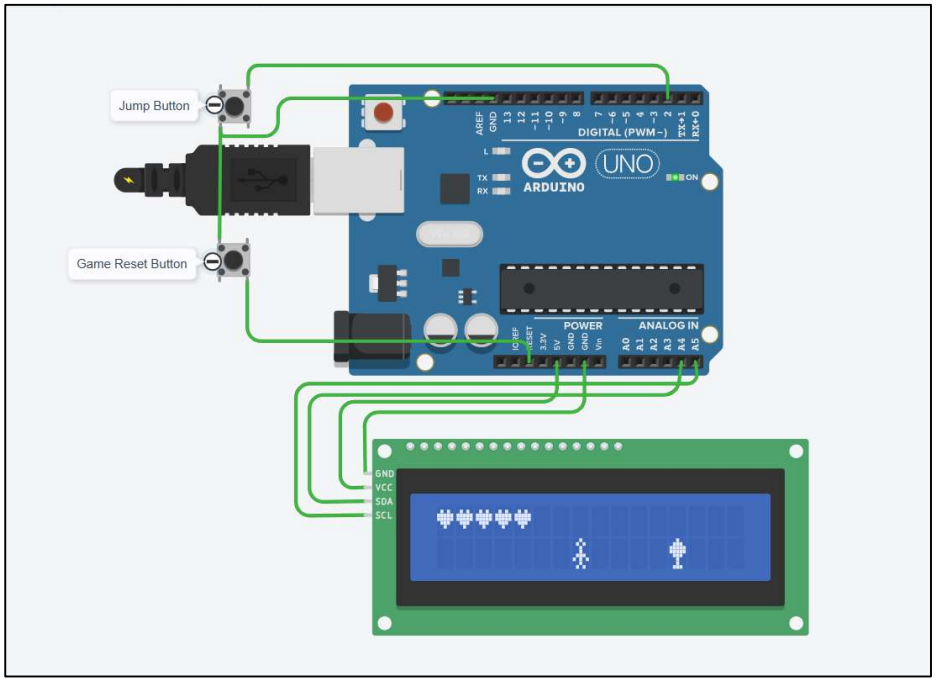
2. Game Logic

- **Character Movement**: The hero's position is controlled by the push button. The character moves up when the button is pressed and down when released.
- **Falling Stones**: Stones fall from the right side of the screen to the left. The game shifts stones and randomly generates new ones.
- **Collision Detection**: If a stone hits the hero when the button is not pressed, the hero loses a life, and the score decreases.
- **Life Display**: The number of remaining lives is displayed as heart symbols on the LCD.

3. Main Loop Execution

- Clear the LCD at the beginning of each loop.
- If the player has lives left, update stone positions, move the hero, show remaining lives, and increase the score.
- If the player has no lives left, display "GAME OVER" and the final score.

CIRCUIT VIEW



CONCLUSION

Ultra-Sonic Sensor Lab: demonstrated the effective use of an ultrasonic sensor to measure distance and provide real-time feedback through LEDs, an LCD display, and a buzzer. This lab highlighted the importance of precise sensor readings and the ability to translate these readings into meaningful visual and auditory signals. The successful integration of the ultrasonic sensor with various output components showcased the potential for creating responsive systems that interact with the environment.

BONUS: explored the development of a simple game using an Arduino, an LCD display, and a push button. This lab emphasized creating custom characters, handling user input, and implementing game logic within the constraints of a microcontroller environment. The game effectively utilized the LCD to display dynamic content, and the button input was used to control the game character, providing an engaging and interactive experience. The game also incorporated real-time score and life tracking, demonstrating the Arduino's capability to manage complex tasks. Together, these labs enhanced understanding of microcontroller programming, sensor integration, and interactive application development. They demonstrated how Arduino systems could be used to create both practical measurement tools and engaging interactive applications, underscoring the versatility and educational value of Arduino as a platform for learning electronics and programming.

References

-Arduino IDE documentation: [Arduino Documentation](https://www.arduino.cc/reference/en/)

-liquidCrystal_I2CLibrary:[LiquidCrystal_I2CLibrary]
(https://github.com/johnrickman/LiquidCrystal_I2C)

-Jump Man Game: [2024 , Jan 1 - CERN-OHL1.2+]
(https://projecthub.arduino.cc/crepeguy/jumpman-lcd-game-c9aea0)