DISTANCE MEASUREMENT AND OBJECT DETECTION SYSTEM USING ULTRASONIC SENSOR WITH RASPBERRY PI PICO W

TEAM MEMBERS

- MOHAMED ASHAAN S
- SANTHOSHKUMAR S
- RAGUL E
- PRADEEBAN S

Aim

To design and implement a distance measurement and object detection system using the HC-SR04 ultrasonic sensor interfaced with a Raspberry Pi Pico W. The system will display the real-time distance on an I2C LCD screen and provide a visual alert using an LED when an object is detected within a specific range.

Apparatus Required

- Raspberry Pi Pico W
- HC-SR04 Ultrasonic Sensor Module
- I2C LCD Display Module (16x2)
- Breadboard
- Jumper Wires

PIN TABLE

Component Pin	Raspberry Pi Pico W Pin
HC-SR04 VCC	5V (VBUS)
HC-SR04 GND	GND
HC-SR04 Trig	GPIO 3
HC-SR04 Echo	GPIO 2
I2C LCD VCC	3.3V
I2C LCD GND	GND
I2C LCD SDA	GPIO 4

I2C LCD SCL	GPIO 5

COMPONENT DESCRIPTIONS

Raspberry Pi Pico W

The Raspberry Pi Pico W is a low-cost, high-performance microcontroller board featuring a dual-core processor. It includes built-in Wi-Fi and Bluetooth capabilities, making it ideal for a wide range of IoT applications, embedded systems, and hobbyist projects. Its GPIO pins allow for easy interfacing with various sensors and modules.

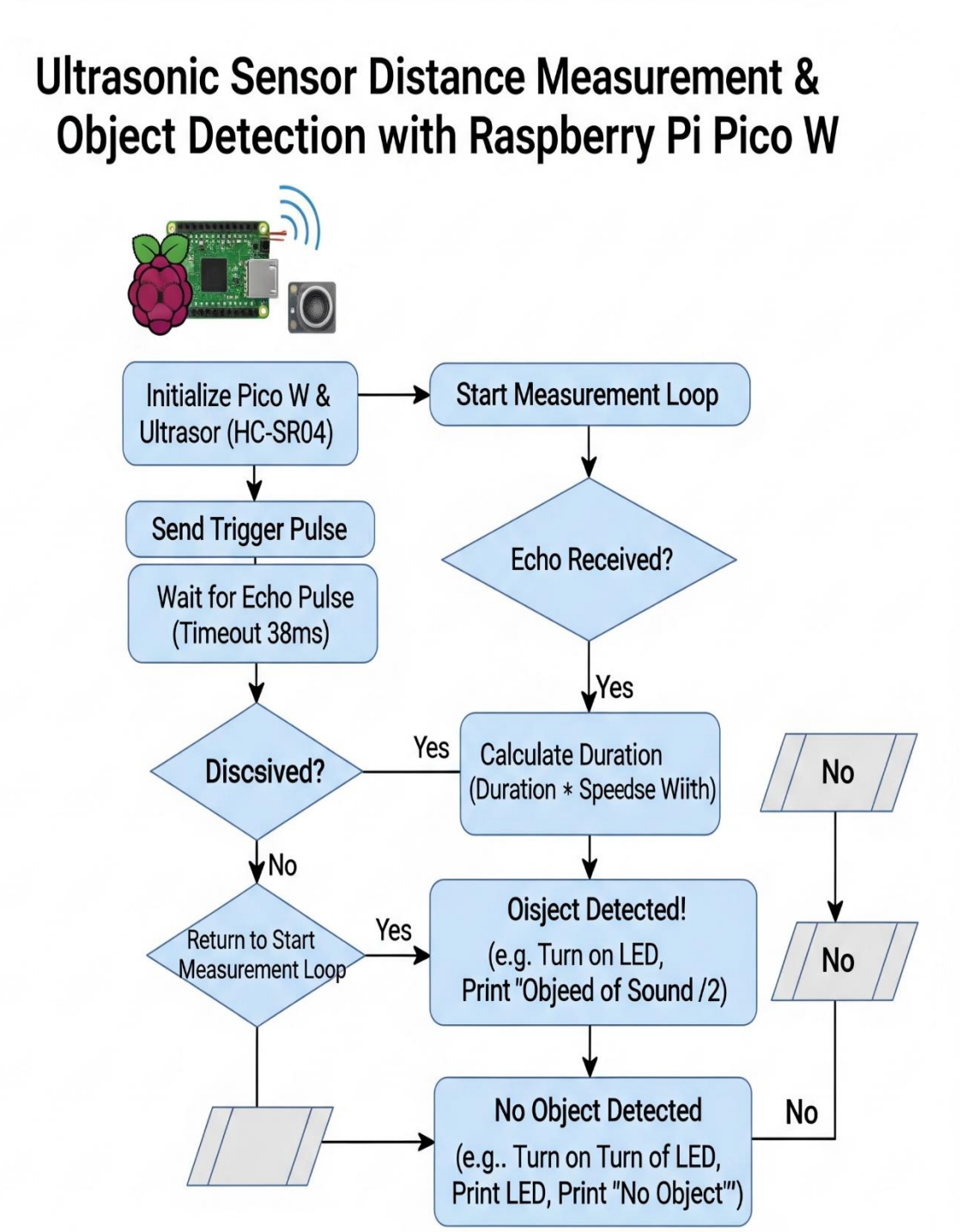
HC-SR04 Ultrasonic Sensor

The HC-SR04 is a non-contact distance measurement sensor. It operates by transmitting an ultrasonic pulse and measuring the time it takes for the echo to return. The distance to an object is calculated based on this time interval and the speed of sound. It is commonly used in robotics for obstacle avoidance and in automation projects for object detection.

I2C LCD Display

An I2C LCD (Liquid Crystal Display) module is a screen used for displaying text and simple graphics. This 16x2 version can display two rows of 16 characters. By using the I2C communication protocol, it simplifies wiring, requiring only two data lines (SDA and SCL) to connect to a microcontroller, in addition to power and ground.

FLOW CHART

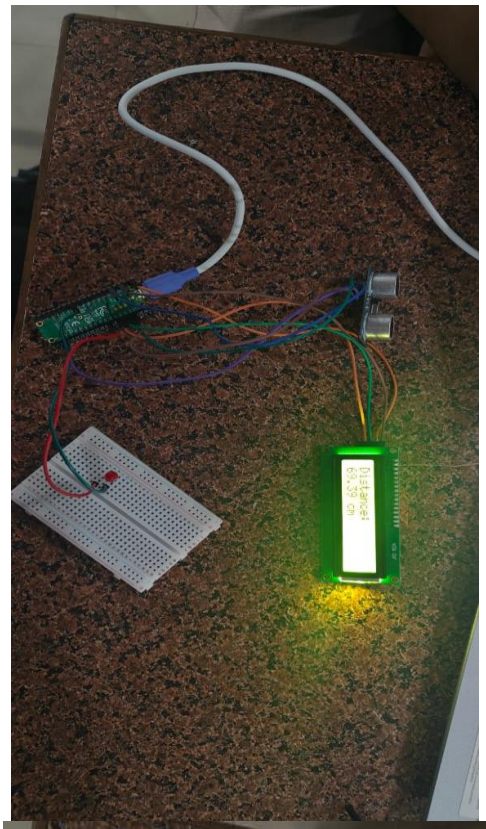

Ultrasonic Sensor Distance Measurement &
Object Detection with Raspberry Pi Pico W
Initialize Pico W &
Ultrasor (HC-SR04)
Start Measurement Loop
Send Trigger Pulse
Wait for Echo Pulse
(Timeout 38ms)
Echo Received?
Yes
Discsived?
Yes
Calculate Duration
(Duration * Speedse Wiith)
No
No
Return to Start
Measurement Loop
Yes
Oisject Detected!
(e.g. Turn on LED,
Print "Objeed of Sound /2)
No
No Object Detected
(e.g.. Turn on Turn of LED,
Print LED, Print "No Object"')
No

PROGRAM

```
from machine import Pin, SoftI2C
import utime
from pico_i2c_lcd import I2cLcd

# Define the GPIO pins for the HC-SR04 ultrasonic sensor
trigger = Pin(3, Pin.OUT)
echo = Pin(2, Pin.IN)

# Define the built-in LED pin for proximity alerts
led = Pin(25, Pin.OUT) # Onboard LED on Raspberry Pi Pico W

# --- LCD Configuration ---
I2C_ADDR = 0x27      # I2C address of the LCD
I2C_NUM_ROWS = 2     # Number of rows on the LCD
I2C_NUM_COLS = 16    # Number of columns on the LCD

# Initialize I2C communication for the LCD
i2c = SoftI2C(sda=Pin(4), scl=Pin(5), freq=400000)
lcd = I2cLcd(i2c, I2C_ADDR, I2C_NUM_ROWS, I2C_NUM_COLS)

def measure_distance():
    """
    Triggers the ultrasonic sensor and measures the distance to an object.
    Returns the distance in centimeters.
    """
    # Send a short pulse to trigger the sensor
```

```
    trigger.low()
    utime.sleep_us(2)
    trigger.high()
    utime.sleep_us(5) # A 10us pulse is recommended, but 5us works well
    trigger.low()

    # Wait for the echo pin to go high, marking the start of the echo pulse
    while echo.value() == 0:
        pulse_start = utime.ticks_us()

    # Wait for the echo pin to go low, marking the end of the echo pulse
    while echo.value() == 1:
        pulse_end = utime.ticks_us()

    # Calculate the duration of the pulse
    pulse_duration = pulse_end - pulse_start

    # Calculate distance using the speed of sound (343 m/s or 0.0343 cm/µs)
    # The duration is divided by 2 because the pulse travels to the object and back.
    distance_cm = (pulse_duration * 0.0343) / 2

    return distance_cm

# --- Main Program Execution ---

# Display an initial message on the LCD
lcd.putstr("Measuring...")
utime.sleep(2)
```

```
lcd.clear()


try:
    # Main loop to continuously measure and display distance
    while True:
        distance = measure_distance()

        # Clear the LCD and display the new distance reading
        lcd.clear()
        lcd.putstr("Distance:\n{:.2f} cm".format(distance))

        # Check if an object is within the 10 cm threshold
        if distance < 10:
            led.value(1)  # Turn on the LED for alert
        else:
            led.value(0)  # Turn off the LED

        # Wait for 1 second before the next measurement
        utime.sleep(1)

except KeyboardInterrupt:
    # Clean up resources if the program is stopped manually (Ctrl+C)
    lcd.backlight_off()
    lcd.display_off()
    led.value(0) # Ensure LED is off on exit
    print("Program stopped.")
```


Distance:
64.39 cm