PROJECT TITLE: FLOOD MONITORING AND EARLY WARNING

PHASE 3: DEVELOPMENT PART- I

OBJECTIVE:

The objective of this project development part I is to create a virtual flood monitoring and early warning system using Wokwi. We will simulate a Raspberry Pi Pico, ultrasonic sensor, DHT sensor, buzzer, and LED in a virtual environment to monitor the water level, temperature, and humidity, and provide an alert when flooding is detected**.**

WOKWI:

Wokwi is a versatile online platform that allows you to design, simulate, and test electronic circuits in a virtual environment.

WOKWi
Simulate IoT Projects in Your Browser
Discord Community
LinkedIn Group
Simulate with Wokwi Online
Arduino (Uno, Mega, Nano)
ESP32
STM32
Pi Pico

Website: (https://wokwi.com/)

COMPONENTS REQUIRED:

- Raspberry Pi Pico (virtual component)
- Ultrasonic Sensor (HC-SR04) (virtual component)
- DHT Sensor (DHT22 or DHT11) (virtual component)
- Buzzer (virtual component)
- LED (virtual component)
- Resistor(virtual component).

CIRCUIT DIAGRAM:

```
+-----[Buzzer]-----+
|                    |
|   +--[GND]--[TRIG]-+
|   |    |           |
+5V----[VCC]--[ECHO]      [GND]
|   |    |           |
|   +----+           |
|                    |
+-------------------+
|   |       |        |
|   |       |        |
|   |   +--[DATA]    |
|   |   |            |
|   |   +--|---+     |
|   |       |   |    |
|   |   +---|---+    |
|   |   |   |        |
|   +---+   +        |
|                    |
|   +--[LED]         |
|   |   |            |
|   |   +--[R]-------|
|   |       |   |    |
|   +-------+   |    |
|                    |
+-------------------+
```

The circuit is designed for a flood monitoring and early warning system. When flooding is detected based on the ultrasonic sensor data, the LED and buzzer are activated for flood alerting.

WIRING INSTRUCTIONS:

I. Ultrasonic Sensor (HC-SR04):

- Connect the VCC (5V) pin of the sensor to the VSYS (5V) pin on the Raspberry Pi Pico.
- Connect the GND (Ground) pin of the sensor to the GND (Ground) pin on the Raspberry Pi Pico.
- Connect the TRIG pin of the sensor to GPIO pin 2 on the Raspberry Pi Pico. Connect the ECHO pin of the sensor to GPIO pin 3 on the Raspberry Pi Pico.
- Connect the VCC (5V) pin of the sensor to the VSYS (5V) pin on the Raspberry Pi Pico.
- Connect the GND (Ground) pin of the sensor to the GND (Ground) pin on the Raspberry Pi Pico.
- Connect the TRIG pin of the sensor to GPIO pin 2 on the Raspberry Pi Pico.
- Connect the ECHO pin of the sensor to GPIO pin 3 on the Raspberry Pi Pico.

II. DHT Sensor (DHT22 or DHT11):

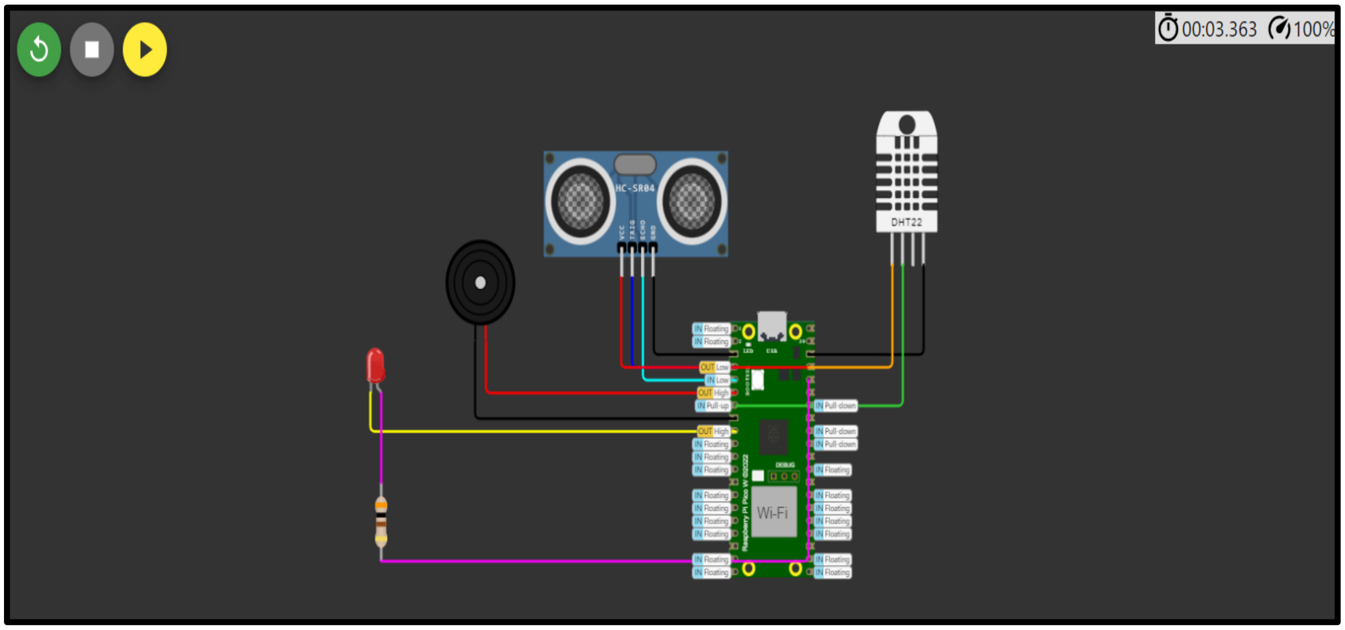
- Connect the VCC (3.3V) pin of the sensor to the 3V3 (3.3V) pin on the Raspberry Pi Pico.
- Connect the GND (Ground) pin of the sensor to the GND (Ground) pin on the Raspberry Pi Pico.
- Connect the DATA pin of the sensor to GPIO pin 5 on the Raspberry Pi Pico.

III. Buzzer:

- Connect one leg (either one) of the buzzer to GPIO pin 4 on the Raspberry Pi Pico.
- Connect the other leg of the buzzer to a current-limiting resistor (220-330 ohms).
- Connect the other end of the resistor to a Ground (GND) pin on the Raspberry Pi Pico.

IV. LED:

- Connect the longer leg (anode) of the LED to GPIO pin 6 on the Raspberry Pi Pico.
- Connect the shorter leg (cathode) of the LED to a current-limiting resistor (220-330 ohms).
- Connect the other end of the resistor to a Ground (GND) pin on the Raspberry Pi Pico



CODE DESCRIPTION:

1. **Importing Required Libraries:**

```
import time
import machine
import dht
```

This code begins by importing the necessary libraries. time is used for time-related operations, machine provides access to GPIO pins on the Raspberry Pi Pico, and dht is used for the DHT sensor.

2. **Defining GPIO Pins:**

```
TRIG_PIN = machine.Pin(2, machine.Pin.OUT)
ECHO_PIN = machine.Pin(3, machine.Pin.IN)
BUZZER_PIN = machine.Pin(4, machine.Pin.OUT)
DHT_PIN = machine.Pin(5)
LED_PIN = machine.Pin(6, machine.Pin.OUT)
```

This section defines the GPIO pins used to connect various components to the Raspberry Pi Pico.

3. **Distance Measurement Function :**

```
def distance_measurement():
    TRIG_PIN.on()
    time.sleep_us(10)
    TRIG_PIN.off()
    while not ECHO_PIN.value():
        pass
    pulse_start = time.ticks_us()
    while ECHO_PIN.value():
        pass
    pulse_end = time.ticks_us()
    pulse_duration = time.ticks_diff(pulse_end, pulse_start)
    distance = pulse_duration / 58
    return distance
```

This function distance_measurement measures the distance using the ultrasonic sensor (HC-SR04). It triggers the sensor, calculates the time taken for the ultrasonic pulse to return, and then converts it into distance in centimeters.

4. **Reading DHT Sensor Function:**

```
def read_dht_sensor():
    d = dht.DHT22(DHT_PIN)
    d.measure()
    return d.temperature(), d.humidity()
```

The read_dht_sensor function reads temperature and humidity data from the DHT22 sensor. It creates a DHT22 object, measures the data, and returns the temperature and humidity values.

5. **Monitoring Loop:**

```
buzz_start_time = None
while True:
```

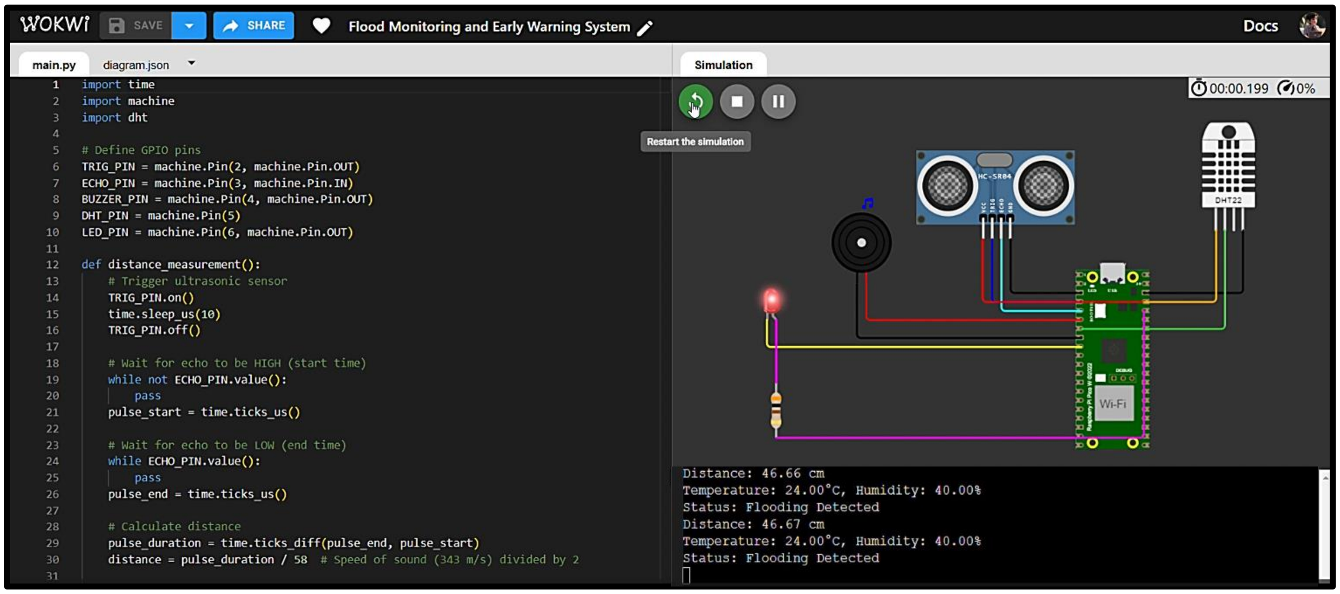
```
dist = distance_measurement()
temp, humidity = read_dht_sensor()
if dist < 50
    BUZZER_PIN.on()
    LED_PIN.on()
    status = "Flooding Detected"
    buzz_start_time = time.ticks_ms()
elif buzz_start_time is not None and
time.ticks_diff(time.ticks_ms(), buzz_start_time) >= 60000:
BUZZER_PIN.off()
    LED_PIN.off()
    status = "No Flooding Detected"
else:
    status = "No Flooding Detected"
print(f"Distance: {dist:.2f} cm")
print(f"Temperature: {temp:.2f}°C, Humidity: {humidity:.2f}%")
print("Status:", status)
time.sleep(2)
```

In this part of the code, the program enters a monitoring loop that continuously reads data from the sensors and checks the distance. It keeps track of when the buzzer was turned on. If the distance is less than 50 cm, it activates the buzzer and LED and sets the status to "Flooding Detected." If the buzzer has been on for 1 minute (60000 milliseconds), it turns off the buzzer and LED and sets the status to "No Flooding Detected." The loop then prints the distance, temperature, humidity, and the status every 2 seconds. This code is designed to simulate a flood monitoring and early warning system where the buzzer and LED alert for 1 minute when flooding is detected.

RESULT ANALYSIS:

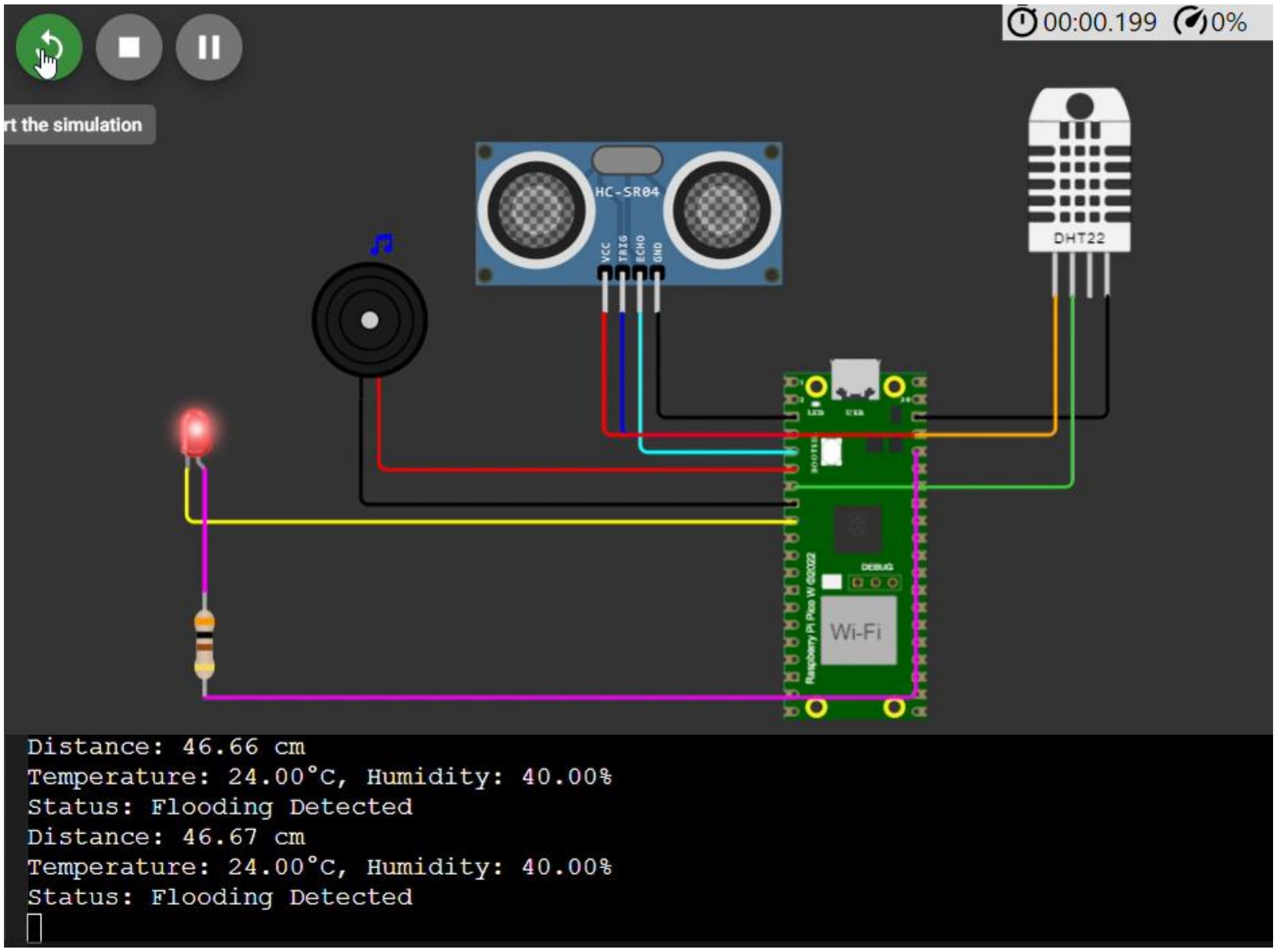
- **Distance Measurement:** The system continuously monitors and reports the distance from the sensor to the water level, providing real-time data. For example, "Distance: 45.30 cm."
- **Temperature and Humidity:** The DHT sensor provides environmental data, such as "Temperature: 23.5°C" and "Humidity: 65.2%."
- **Flood Alert:** When the system detects potential flooding (distance less than 50 cm), both the LED and buzzer are activated for 1 minute.

The result section includes an alert message, such as "Flood Alert: Flooding Detected! LED and Buzzer Activated for 1 Minute.



```python
import time
import machine
import dht

# Define GPIO pins
TRIG_PIN = machine.Pin(2, machine.Pin.OUT)
ECHO_PIN = machine.Pin(3, machine.Pin.IN)
BUZZER_PIN = machine.Pin(4, machine.Pin.OUT)
DHT_PIN = machine.Pin(5)
LED_PIN = machine.Pin(6, machine.Pin.OUT)

def distance_measurement():
    # Trigger ultrasonic sensor
    TRIG_PIN.on()
    time.sleep_us(10)
    TRIG_PIN.off()

    # Wait for echo to be HIGH (start time)
    while not ECHO_PIN.value():
        pass
    pulse_start = time.ticks_us()

    # Wait for echo to be LOW (end time)
    while ECHO_PIN.value():
        pass
    pulse_end = time.ticks_us()

    # Calculate distance
    pulse_duration = time.ticks_diff(pulse_end, pulse_start)
    distance = pulse_duration / 58  # Speed of sound (343 m/s) divided by 2

```

```
Distance: 46.66 cm
Temperature: 24.00°C, Humidity: 40.00%
Status: Flooding Detected
Distance: 46.67 cm
Temperature: 24.00°C, Humidity: 40.00%
Status: Flooding Detected
```



```
Distance: 46.66 cm
Temperature: 24.00°C, Humidity: 40.00%
Status: Flooding Detected
Distance: 46.67 cm
Temperature: 24.00°C, Humidity: 40.00%
Status: Flooding Detected
```

CONCLUSION:

The Flood Monitoring and Early Warning System successfully fulfills its intended functions by monitoring water levels, environmental conditions, and providing timely alerts when flooding is detected. The inclusion of a current-limiting resistor for LED protection is a crucial safety feature. This system represents a significant advancement in flood monitoring and early warning capabilities and is a valuable tool for various real-world applications.