FLOOD MONITERING AND EARLY WARNING

PROJECT DESCRIPTION :

- Flooding is usually brought on by an increased quantity of water in a water system, like a lake, river overflowing. On occasion a dam fractures, abruptly releasing a massive quantity of water. The outcome is that a number of the water travels into soil, and 'flooding' the region.

- Rivers are involving river banks, in a station.
  Aside from lack of products and house and office property, streets infrastructure flood water consists of bacteria and sewage flow of waste sites and chemical spillage which leads to a variety of diseases afterwards.
  Flood predictions need information like:
  The speed of change in river stage on a realtime basis, which may help indicate the seriousness and immediacy of this threat.

- Understanding of the form of storm generating the moisture, such as length, intensity and areal extent, which is valuable for discovering potential seriousness of the flood.
  In this system we make use of a raspberry pi with water sensors, rain sensors to predict flood and alert respective authorities and sound instant alarm in nearby villages to instantly transmit information about possible floods using IOT.

- The water sensors are used to measure water level of 3 different locations. Also 3 different rain sensors are used to measure rain level of those 3 areas. These sensors provide information over the IOT using Raspberry Pi. On detection of conditions of flooding the system predicts the amount of time it would take to flood in a particular area and alerts the villages/areas that could be affected by it. The system also calculates the time it would take for flood to reach them and provides a time to people so that they can evacuate accordingly.

As we all know that Flood is one of the major well known Natural Disasters. When water level suddenly rises in dams, river beds etc. Alot of Destruction happens at surrounding places. It causes a huge amount of loss to our environment and living beings as well. So in these case, it is very important to get emergency alerts of the water level situation in different conditions in the river bed

PROJECT IMPLEMENTATION :

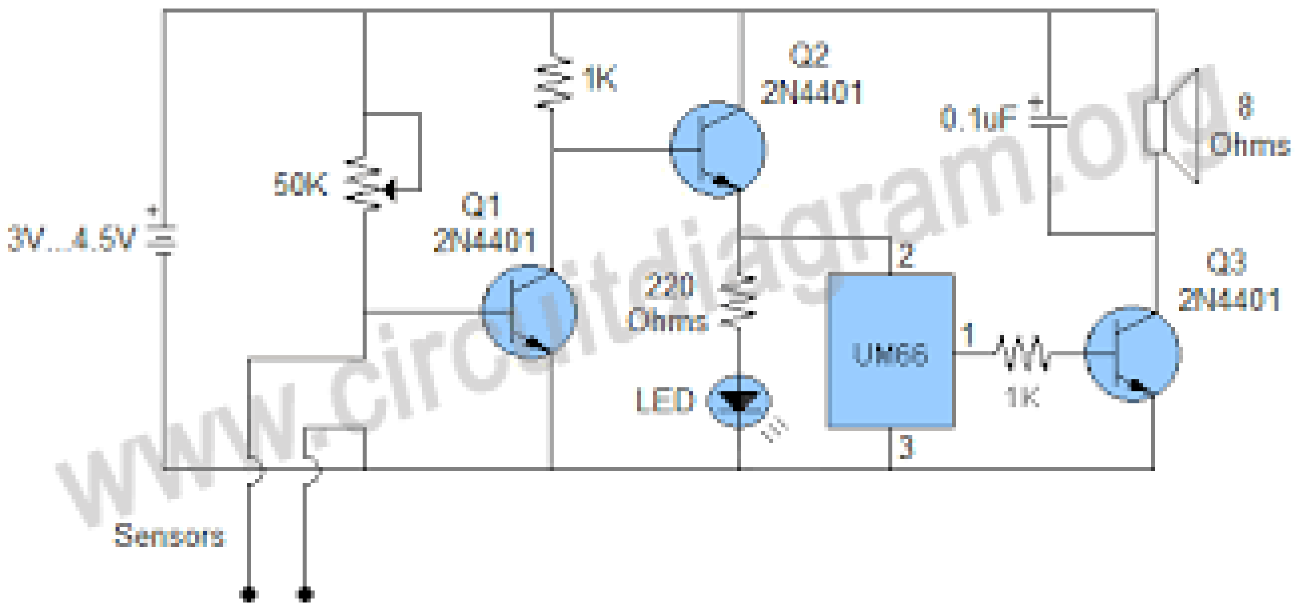
❖ Hardware Specifications

- Raspberry Pi 3
- Wifi Module
- LCD Display
- Water Sensor
- Rain Drop Sensor
- Resistors
- Capacitors
- Transistors
- Cables and Connectors
- Diodes
- PCB and Breadboards
- LED
- Transformer/Adapter
- Push Buttons
- Switch
- IC
- IC Sockets

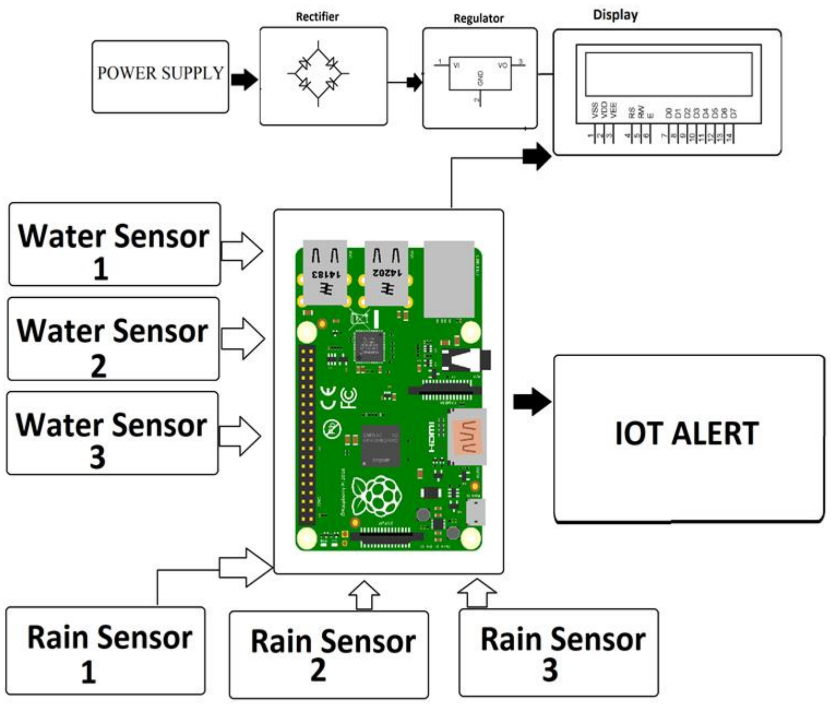
❖ Software Specifications

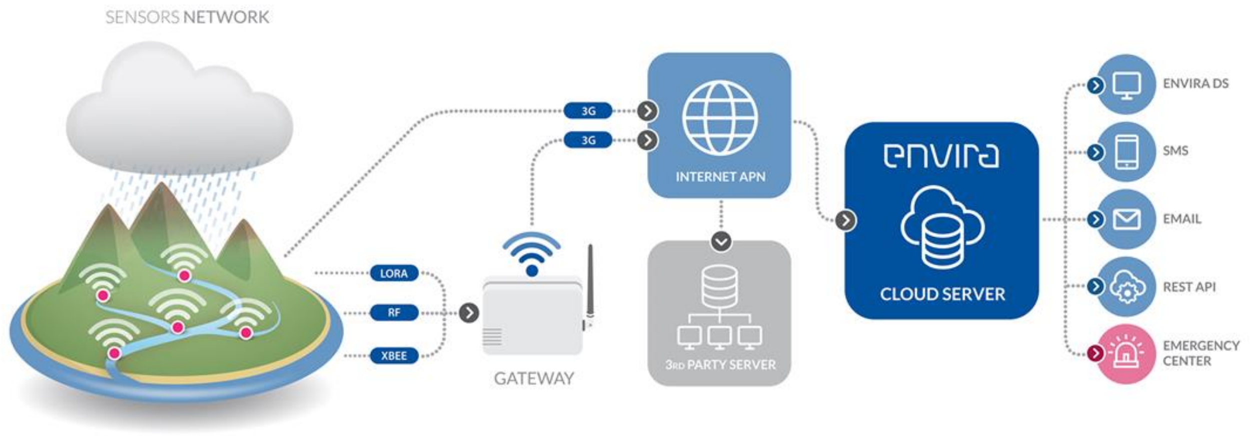
- Linux
- Programming Language: Python

CIRCUIT DIAGRAM :



Blockdiagram





BLOCK DIAGRAM :

- ❖ In this project we are using Ultrasonic Sensor, DHT11 Sensor, Flow Sensor, Raindrop Sensor, Turbidity Sensor to measure the parameters like water environmental temperature and humidity, Flow rate, level of water, rain intensity of the environment, how much amount of water is been contaminated in rivers. If any of the value exceeded message on the values will be uploaded using GSM Module and the exceeded values will be displayed in LCD screen and also it gives indication using LED's and alert will be given using voice sensor and also through SMS notification.

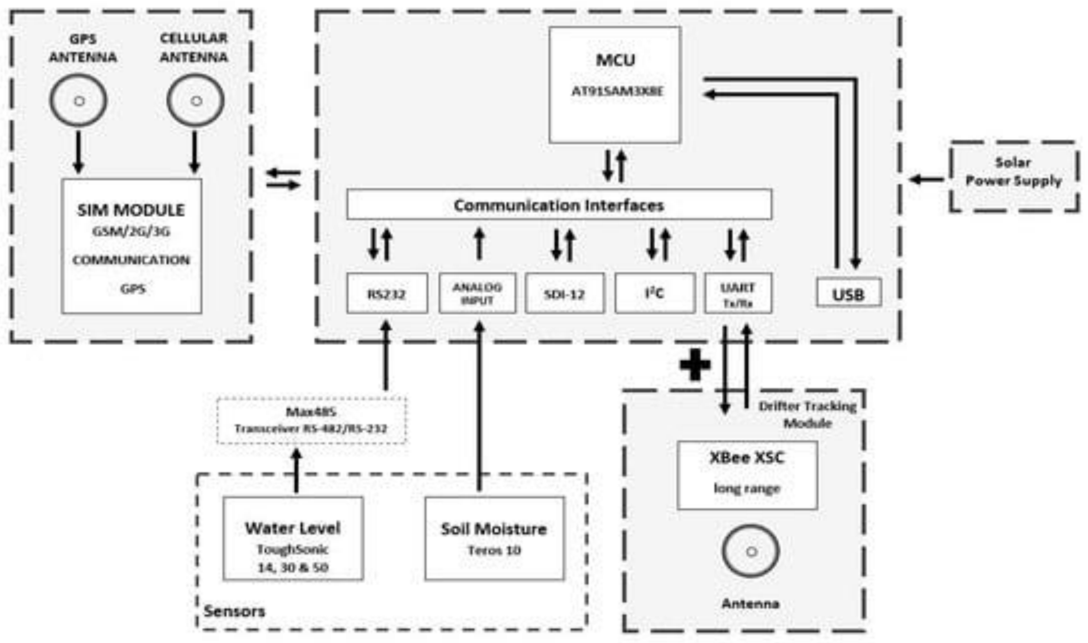


Fig1.1 RIVERCORE HARDWARE BLACK DIAGRAM

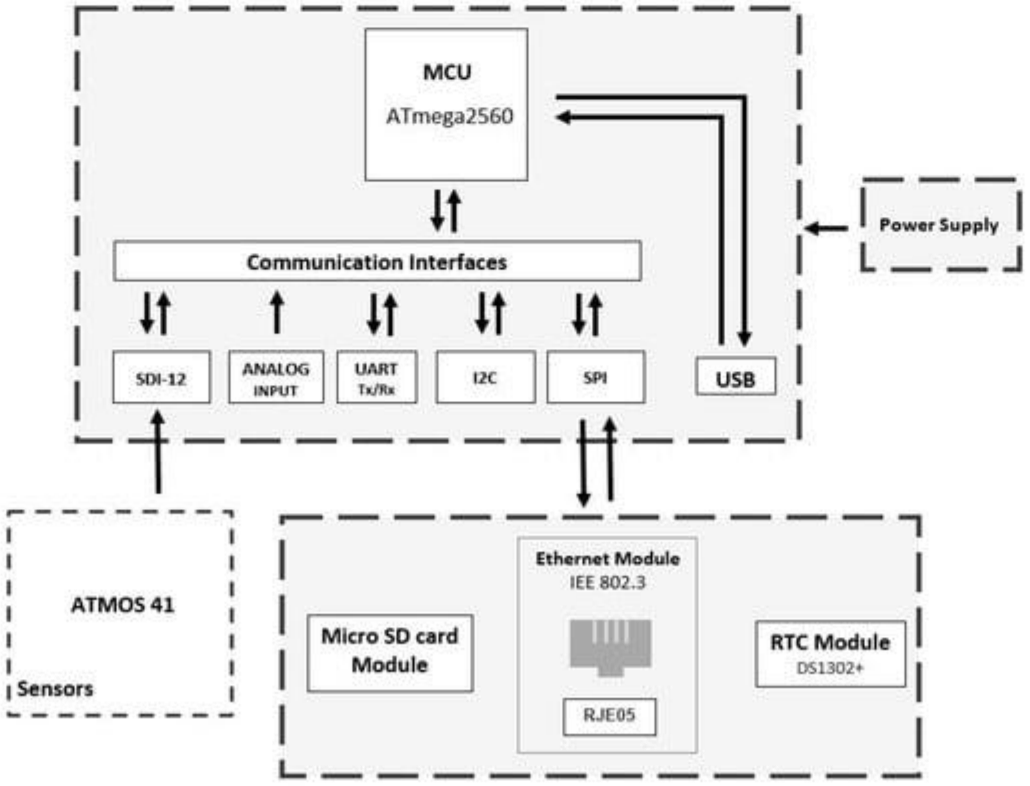


Fig 1.2: WATER STATION HARDWARE BLOCK DIAGRAM

SOURCE CODE :

FRONT END(Python code) ;

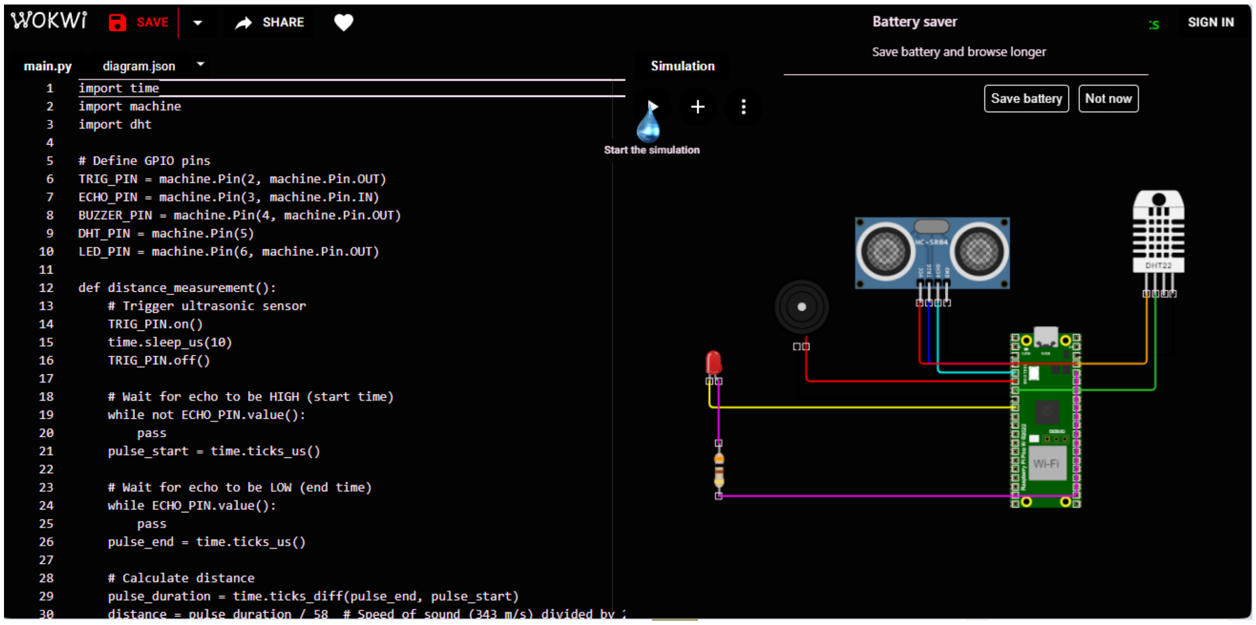
```
import time

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

    return distance

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

buzz_start_time = None  # To track when the buzzer started

while True:
    dist = distance_measurement()
    temp, humidity = read_dht_sensor()

    # Check if the distance is less than a threshold (e.g., 50 cm)
    if dist < 50:
        # Turn on the buzzer and LED
        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:  # 1 minute
        # Turn off the buzzer and LED after 1 minute
        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)
```

BACKEND CODE :

```
{
  "version": 1,
  "author": "Anonymous maker",
  "editor": "wokwi",
  "parts": [
    {
      "type": "board-pi-pico-w",
      "id": "pico",
      "top": -118.45,
      "left": 32.35,
      "attrs": { "env": "micropython-20231005-v1.21.0" }
    },
    {
      "type": "wokwi-hc-sr04",
    "id": "ultrasonic1",
    "top": -238.5,
    "left": -138.5,
    "attrs": { "distance": "257" }
  },
  {
    "type": "wokwi-buzzer",
    "id": "bz1",
    "top": -180,
    "left": -228.6,
    "attrs": { "volume": "0.1" }
  },
  { "type": "wokwi-dht22", "id": "dht1", "top": -268.5, "left": 167.4, "attrs": {} },
  { "type": "wokwi-led", "id": "led1", "top": -99.6, "left": -313, "attrs": { "color": "red" } },
  {
    "type": "wokwi-resistor",
    "id": "r1",
    "top": 33.6,
    "left": -317.35,
    "rotate": 90,
    "attrs": { "value": "300" }
  }
],
"connections": [
  [ "ultrasonic1:TRIG", "pico:GP2", "blue", [ "v0" ] ],
  [ "ultrasonic1:ECHO", "pico:GP3", "cyan", [ "v0" ] ],
  [ "ultrasonic1:GND", "pico:GND.1", "black", [ "v0" ] ],
  [ "bz1:2", "pico:GP4", "red", [ "v0" ] ],
    [ "bz1:1", "pico:GND.2", "black", [ "v0" ] ],
    [ "dht1:GND", "pico:GND.8", "black", [ "v0" ] ],
    [ "dht1:SDA", "pico:GP5", "limegreen", [ "v0" ] ],
    [ "ultrasonic1:VCC", "pico:3V3_EN", "red", [ "v0" ] ],
    [ "dht1:VCC", "pico:3V3_EN", "orange", [ "v0" ] ],
    [ "led1:C", "pico:GP6", "yellow", [ "v0" ] ],
    [ "led1:A", "r1:1", "magenta", [ "v0" ] ],
    [ "r1:2", "pico:3V3", "magenta", [ "h0" ] ]
  ],
  "dependencies": {}
}
```

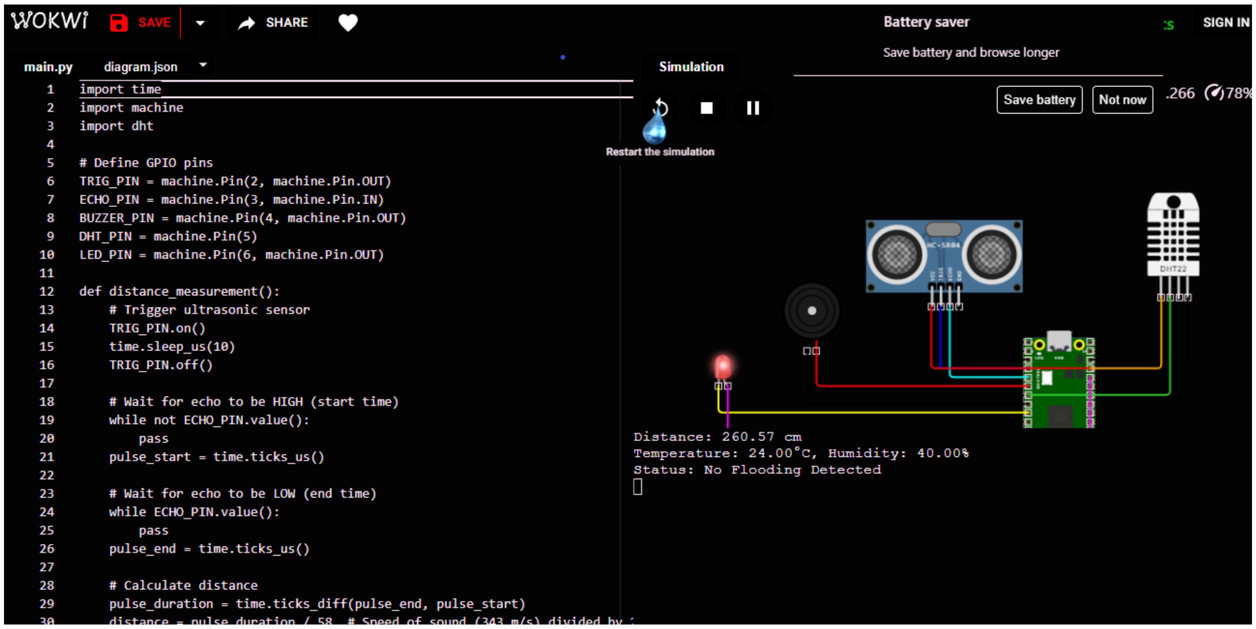
OUT PUT:

STEP 1: INITIALSTAGE



```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
```

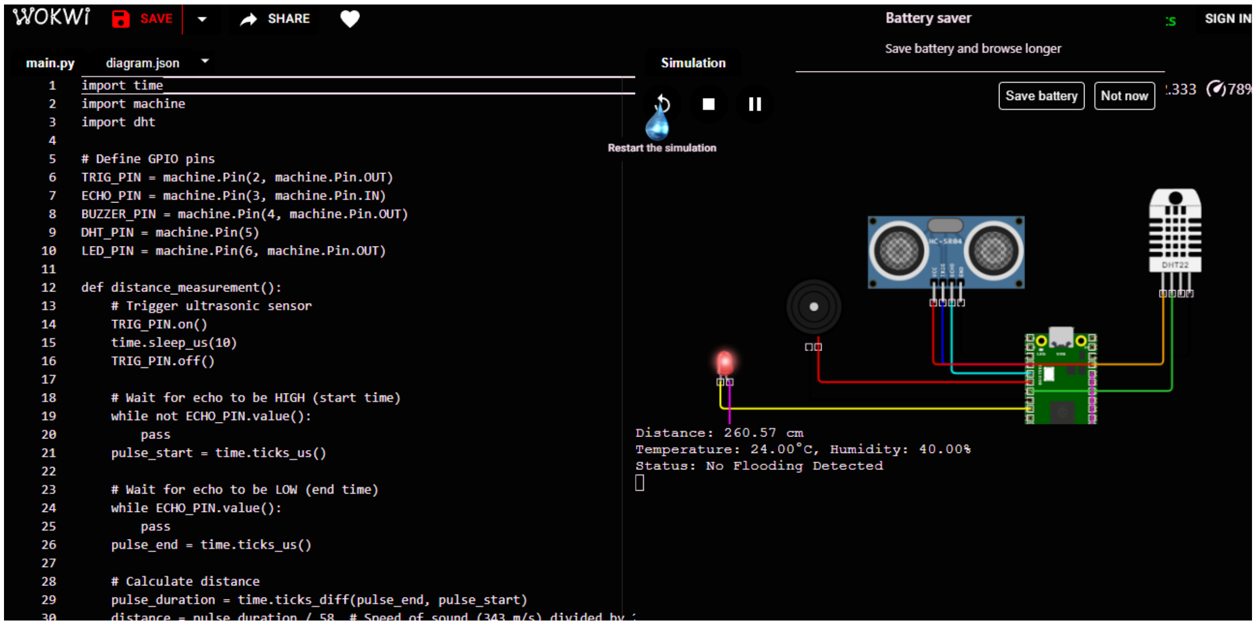
STEP 2:



```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
```

```
Distance: 260.57 cm
Temperature: 24.00°C, Humidity: 40.00%
Status: No Flooding Detected
```

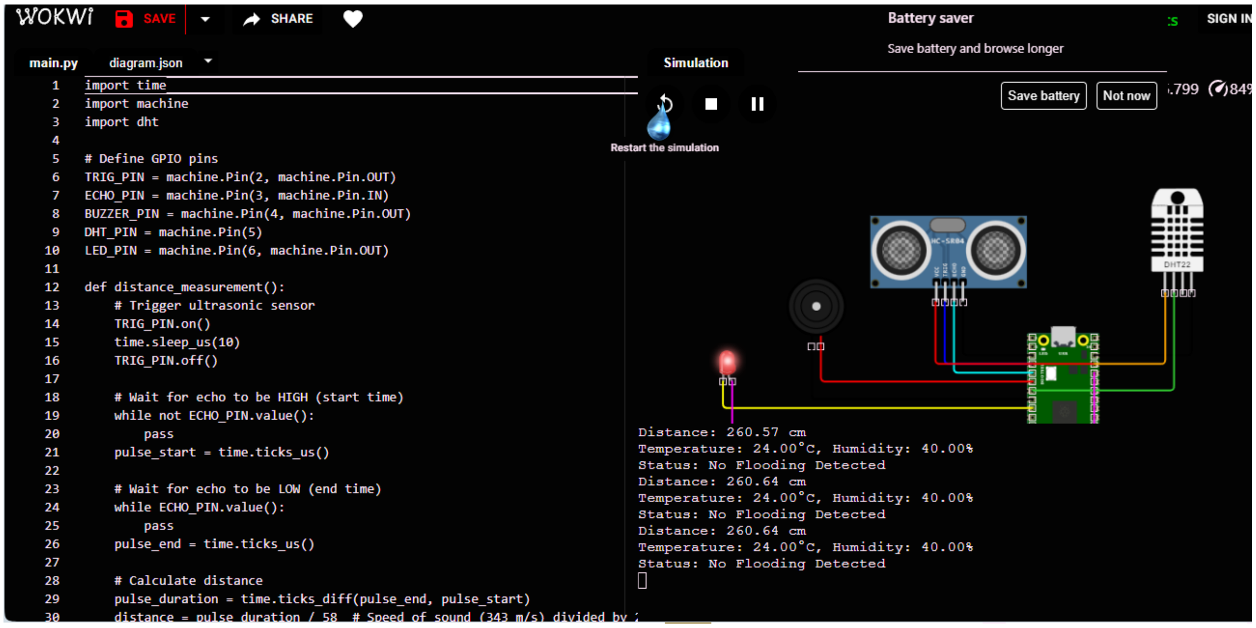
STEP 3: MIDDLE STAGE



```
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
```

```
Distance: 260.57 cm
Temperature: 24.00°C, Humidity: 40.00%
Status: No Flooding Detected
```

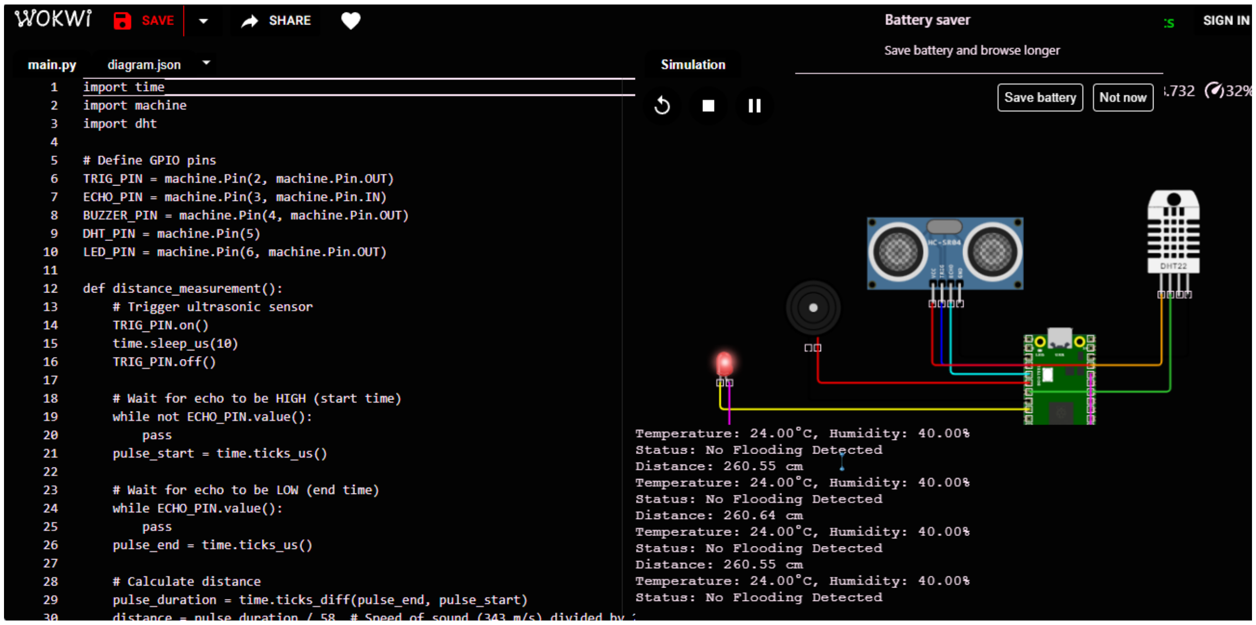
STEP4:



```
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
```

```
Distance: 260.57 cm
Temperature: 24.00°C, Humidity: 40.00%
Status: No Flooding Detected
Distance: 260.64 cm
Temperature: 24.00°C, Humidity: 40.00%
Status: No Flooding Detected
Distance: 260.64 cm
Temperature: 24.00°C, Humidity: 40.00%
Status: No Flooding Detected
```

STEP 5: FINAL STAGE



```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
```

```
Temperature: 24.00°C, Humidity: 40.00%
Status: No Flooding Detected
Distance: 260.55 cm
Temperature: 24.00°C, Humidity: 40.00%
Status: No Flooding Detected
Distance: 260.64 cm
Temperature: 24.00°C, Humidity: 40.00%
Status: No Flooding Detected
Distance: 260.55 cm
Temperature: 24.00°C, Humidity: 40.00%
Status: No Flooding Detected
```