

Smart Water Fountain using the Wokwi simulator

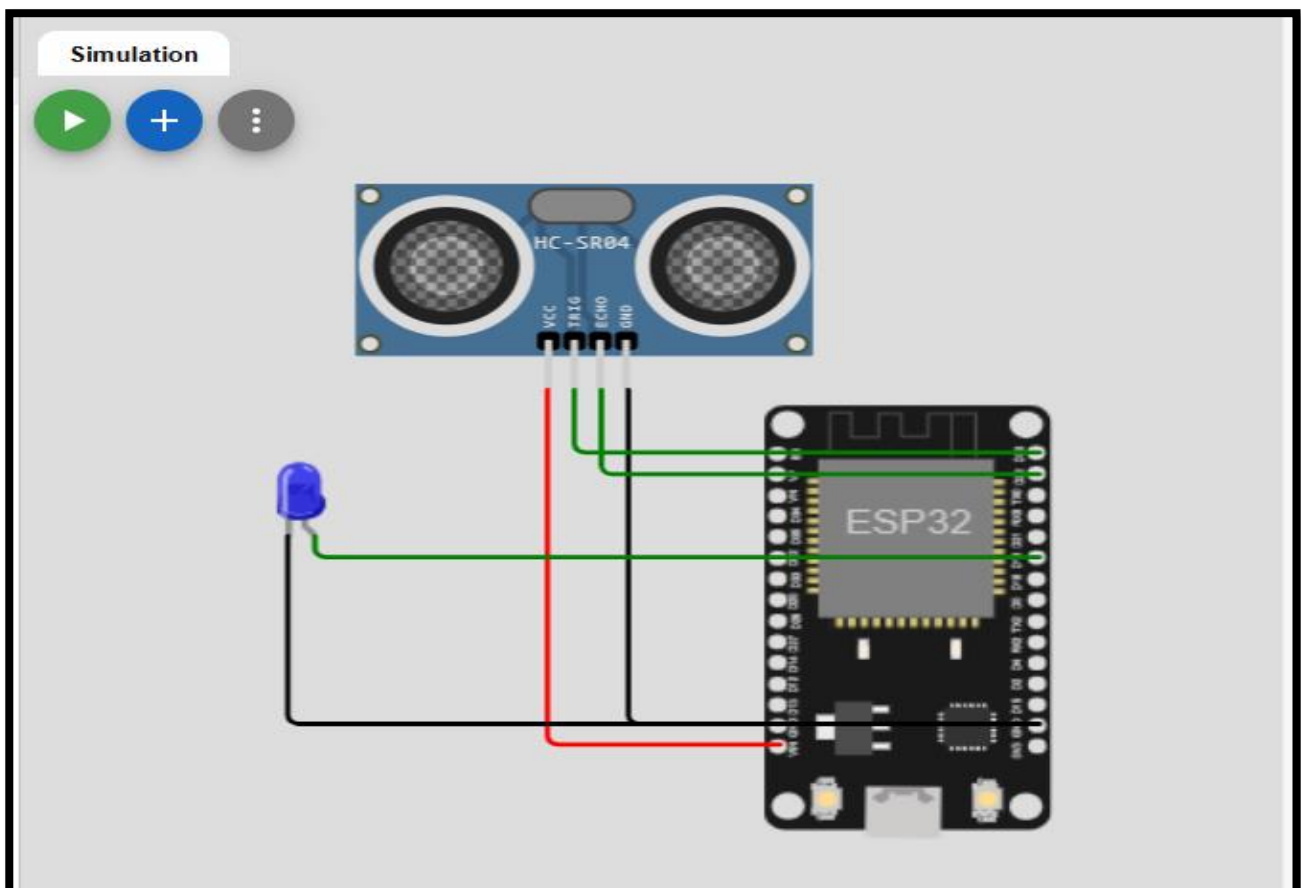
code: main.py

```
main.py  diagram.json  Library Manager  ▼
1  import machine
2  import time
3
4  # Pin assignments for the ultrasonic sensor
5  TRIGGER_PIN = 23  # GPIO23 for trigger
6  ECHO_PIN = 22     # GPIO22 for echo
7
8  # Pin assignment for the LED
9  LEAK_LED_PIN = 19 # GPIO19 for the LED
10
11 # Set the pin modes
12 trigger = machine.Pin(TRIGGER_PIN, machine.Pin.OUT)
13 echo = machine.Pin(ECHO_PIN, machine.Pin.IN)
14 leak_led = machine.Pin(LEAK_LED_PIN, machine.Pin.OUT)
15
16 # Function to measure distance using the ultrasonic sensor
17 def measure_distance():
18     # Generate a short trigger pulse
19     trigger.value(0)
20     time.sleep_us(5)
21     trigger.value(1)
22     time.sleep_us(10)
23     trigger.value(0)
24
25     # Measure the echo pulse duration to calculate distance
26     pulse_start = pulse_end = 0
27     while echo.value() == 0:
28         pulse_start = time.ticks_us()
29     while echo.value() == 1:
30         pulse_end = time.ticks_us()
31
32     pulse_duration = pulse_end - pulse_start
33
34     # Calculate distance in centimeters (assuming the speed of sound is 343 m/s)
35     distance = (pulse_duration * 0.0343) / 2 # Divide by 2 for one-way travel
36
37     return distance
38
39 # Function to check for a water leak
40 def check_for_leak():
41     # Measure the distance from the ultrasonic sensor
42     distance = measure_distance()
43
44     # Set the threshold distance for detecting a leak (adjust as needed)
45     threshold_distance = 10 # Adjust this value based on your tank setup
46
47     if distance < threshold_distance:
48         # If the distance is less than the threshold, a leak is detected
49         return True
50     else:
51         return False
52
53 # Main loop
54 while True:
55     if check_for_leak():
56         # Blink the LED to indicate a leak
57         leak_led.value(1) # LED ON
58         time.sleep(0.5)
59         leak_led.value(0) # LED OFF
60         time.sleep(0.5)
61     else:
62         leak_led.value(0) # LED OFF
63
64     time.sleep(1) # Delay between measurements
65
```

Sdiagram.json

```
main.py  diagram.json  Library Manager
1  {
2    "version": 1,
3    "author": "Uri Shaked",
4    "editor": "wokwi",
5    "parts": [
6      { "type": "wokwi-esp32-devkit-v1", "id": "esp", "top": -14.5, "left": 81.4, "attrs": {} },
7      { "type": "wokwi-led", "id": "led2", "top": 6, "left": -111.4, "attrs": { "color": "blue" } },
8      { "type": "wokwi-hc-sr04", "id": "ultrasonic1", "top": -113.7, "left": -71.3, "attrs": {} }
9    ],
10   "connections": [
11     [ "esp:TX0", "$serialMonitor:RX", "", [] ],
12     [ "esp:RX0", "$serialMonitor:TX", "", [] ],
13     [ "ultrasonic1:GND", "esp:GND.2", "black", [ "v0" ] ],
14     [ "ultrasonic1:ECHO", "esp:D22", "green", [ "v0" ] ],
15     [ "ultrasonic1:TRIG", "esp:D23", "green", [ "v0" ] ],
16     [ "ultrasonic1:VCC", "esp:VIN", "red", [ "v0" ] ],
17     [ "led2:A", "esp:D19", "green", [ "v0" ] ],
18     [ "led2:C", "esp:GND.1", "black", [ "v0" ] ]
19   ],
20   "dependencies": {}
21 }
```

CIRCUIT :-



STIMULATION OUTPUT:-

main.py

diagram.json

Library Manager

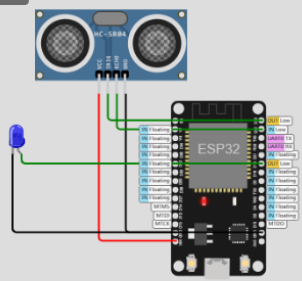
```
1 import machine
2 import time
3
4 # Pin assignments for the ultrasonic sensor
5 TRIGGER_PIN = 23 # GPIO23 for trigger
6 ECHO_PIN = 22    # GPIO22 for echo
7
8 # Pin assignment for the LED
9 LEAK_LED_PIN = 19 # GPIO19 for the LED
10
11 # Set the pin modes
12 trigger = machine.Pin(TRIGGER_PIN, machine.Pin.OUT)
13 echo = machine.Pin(ECHO_PIN, machine.Pin.IN)
14 leak_led = machine.Pin(LEAK_LED_PIN, machine.Pin.OUT)
15
16 # Function to measure distance using the ultrasonic sensor
17 def measure_distance():
18     # Generate a short trigger pulse
19     trigger.value(0)
20     time.sleep_us(5)
21     trigger.value(1)
22     time.sleep_us(10)
23     trigger.value(0)
24
25     # Measure the echo pulse duration to calculate distance
26     pulse_start = pulse_end = 0
27     while echo.value() == 0:
28         pulse_start = time.ticks_us()
29     while echo.value() == 1:
30         pulse_end = time.ticks_us()
```

Simulation

00:45.079

99%

Resume



```
rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:2
load:0x3fff0030,len:4728
load:0x40078000,len:14876
ho 0 tail 12 room 4
load:0x40080400,len:3368
entry 0x400805cc
```