FLOOD MONITORING AND EARLY WARNING:

INTRODUCTION:

Flood monitoring and early warning IoT projects aim to harness the power of interconnected sensors, data analytics, and communication technologies to create a comprehensive flood detection and notification ecosystem.

COMPONENTS USED:

| **Component** | **Purpose** | **Pin/Port** |
| --- | --- | --- |
| Arduino Board | Microcontroller platform | N/A |
| Ultrasonic Sensor | HC-SR04 or similar - for distance measurement | N/A |
| Buzzer | Audible alert when objects are too close | Pin 2 |
| Serial Monitor | Displaying distance measurements for debugging | Serial Communication (USB) |
| LED (Optional) | Visual indicator for the buzzer's state | Pin (Not used in the provided code) |
| Resistor (Optional) | Current limiting resistor for LED (if used) | Not used |

WORKING PRINCIPLE:

* The program starts by defining the pins used for the ultrasonic sensor (TRIG\_PIN and ECHO\_PIN) and the buzzer (BUZZER\_PIN). It also sets a distance threshold (DISTANCE\_THRESHOLD), which is the distance below which the buzzer will be activated.
* WE have used wokwi simulator
* Serial communication is initiated with a baud rate of 9600, allowing you to monitor the system's output via the Arduino Serial Monitor.
* TRIG\_PIN is set as an output, which will be used to trigger the ultrasonic sensor.
* ECHO\_PIN is set as an input, which will receive the echo signal from the sensor.
* BUZZER\_PIN is set as an output to control the buzzer.
* A trigger pulse is sent to the ultrasonic sensor by setting TRIG\_PIN to HIGH for a short duration (10 microseconds) and then brought back to LOW.
* This pulse initiates the ultrasonic wave transmission.
* The pulseIn() function is used to measure the time it takes for the ultrasonic signal to bounce off an object and return to the sensor.
* This time is measured in microseconds and stored in the duration\_us variable.
* The program then calculates the distance in centimeters based on the speed of sound and the time taken for the signal to travel to the object and back. The calculated distance is stored in the distance\_cm variable.
* The program checks if the distance cm is less than the defined DISTANCE THRESHOLD. If it is, it means an object is within the specified range, and it activates the buzzer by setting BUZZER PIN to HIGH. Otherwise, the buzzer is turned off by setting BUZZER PIN to LOW.
* The program also outputs the measured distance to the Serial Monitor for monitoring.
* There is a delay of 500 milliseconds before the loop repeats, providing a brief pause between distance measurements.

ARDUINO CODE:

#define TRIG\_PIN   23

#define ECHO\_PIN   22

#define BUZZER\_PIN 2

#define DISTANCE\_THRESHOLD 50

float duration\_us, distance\_cm;

void setup() {

**Serial**.begin (9600);

  pinMode(TRIG\_PIN, OUTPUT);

  pinMode(ECHO\_PIN, INPUT);

  pinMode(BUZZER\_PIN, OUTPUT);

}

void loop() {

  digitalWrite(TRIG\_PIN, HIGH);

  delayMicroseconds(10);

  digitalWrite(TRIG\_PIN, LOW);

  duration\_us = pulseIn(ECHO\_PIN, HIGH);

  // calculate the distance

  distance\_cm = 0.017 \* duration\_us;

  if (distance\_cm < DISTANCE\_THRESHOLD)

    digitalWrite(BUZZER\_PIN, HIGH);

  else

    digitalWrite(BUZZER\_PIN, LOW);

**Serial**.print("distance: ")

**Serial**.print(distance\_cm);

**Serial**.println(" cm");

  delay(500);

}

SIMULATION:

