****

**FLOOD MONITORING AND EARLY WARNING SYSTEM**

**NAME :** PAVITHRASELVI.A

**REGNO : 953021106050**

**EMAIL : pavithraselvia.ug.21.ec@mtec.ac.in**

**COLLEGE CODE : 9530**

**COLLEGE NAME :** ST.MOTHER THERESA ENGINEERING

COLLEGE

**TEAM NAME :** proj 201033\_ Team \_1

CONTENT:

* + *ABSTRACT*
  + *INTRODUCTION*
  + *COMPONENTS*
  + *SOFTWARE*
  + *PROGRAMMING*
  + *PYTHON CODE*
  + *HARDWARE OUTPUT*

# *PYTHON CODE*

### ABSTRACT:

Floods are the natural disasters that cause catastrophic destruction and devastation of natural life , agriculture ,property and infrastructure every year .The objective of this project is to monitor the flood situation and send alert in case of danger in the form of text message .The main objective of this project is to detect rising water level in a river at a reasonable distance from the rail track/roadways and intimate that to the respective authorities through SMS ,to take appropriate action .Our project solves problem by implementing an early flood detection mechanism .In this project we will connect water level sensors at different water levels.

INTRODUCTION:

Here the Arduino uno is connected to water float sensors & ultrasonic sensor to analyses the water level.  The ultrasonic sensors measure the distance of the water level, and the Arduino micro-controller processes the signals from the sensors. Further, these calculated values will be passed to the Arduino which is been developed with Java, C++. The Arduino would give the alert message to the IoT module.

This system used to detect the current water level of flood around the road and will give real-time information to the motorists or commuters that has still not passing through the flooded areas to avoid problem.

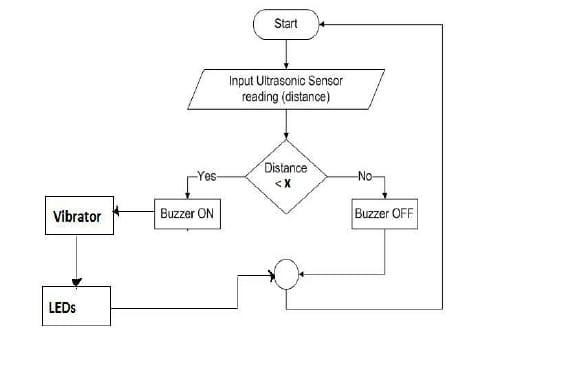
COMPONENTS:

1. Arduino Uno
2. Ultrasonic sensor
3. Buzzer
4. LED
5. Resister
6. Connecting jumping wire
7. Breadboard

### SOFTWARE :

### ARDUINO

### Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online.

****

**PROBLEM ANALYSIS:**

* We don’t have system which could inform People about flood risk .
* To prevent rising or running water from causing flooding.

            Distance= (time x speed)/2.

Here we have divided the product of speed and time by 2 because the time is the total time it took to reach the obstacle and return back. Thus the time to reach obstacle is just half the total time taken.

PROGRAMMING:

import time

# Simulated sensor functions

def read\_ultrasonic\_sensor():

# Simulated ultrasonic sensor (adjust as needed)

return 20 + (time.time() % 10)

def read\_rain\_sensor():

# Simulated rain sensor (adjust as needed)

return time.time() % 2

def send\_warning\_message():

print("Flood detected! Sending warning message...")

# Main monitoring loop

while True:

try:

ultrasonic\_distance = read\_ultrasonic\_sensor()

rain\_intensity = read\_rain\_sensor()

if rain\_intensity > 0.5:

print("Rain detected. Flood risk!")

if ultrasonic\_distance < 30:

print(f"Flood detected. Water level: {ultrasonic\_distance} cm")

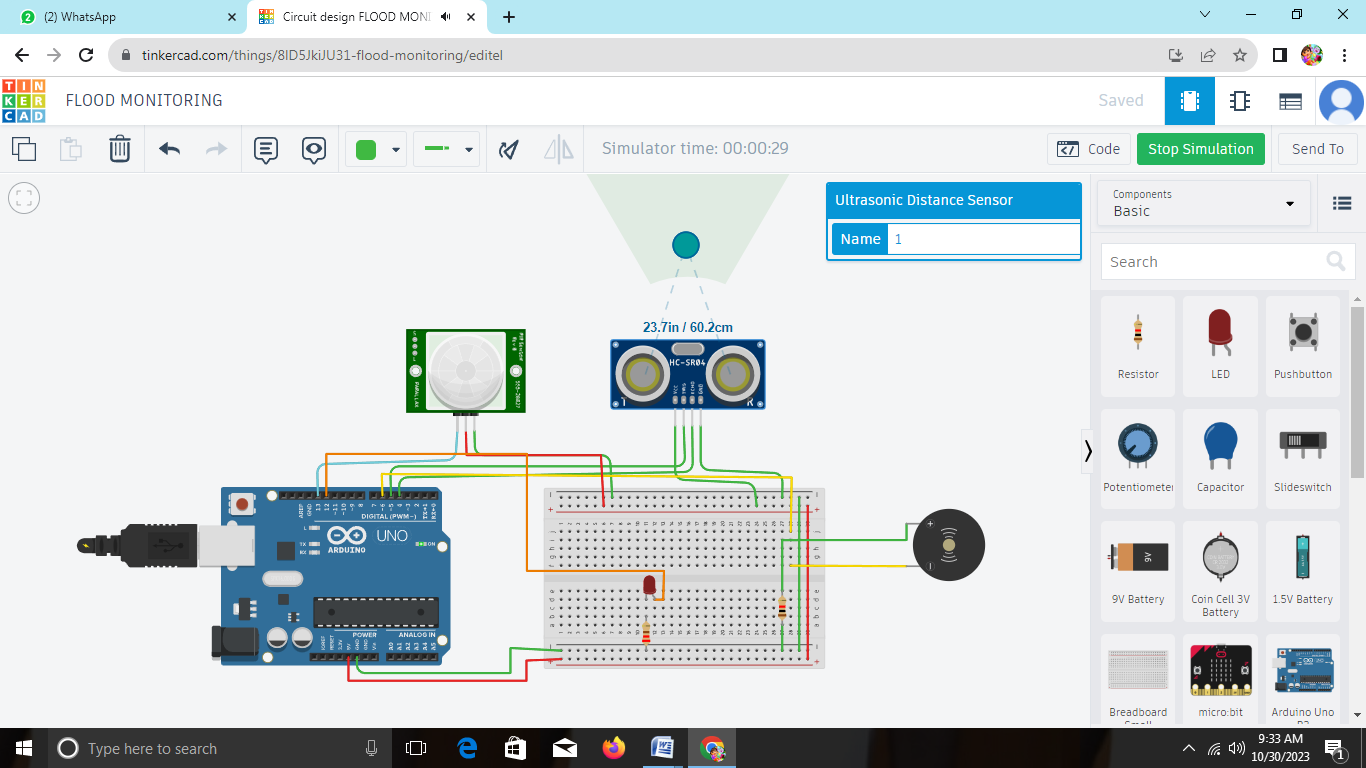
send\_warning\_message()

time.sleep(1) # Adjust the interval as needed

except KeyboardInterrupt:

break

HARDWARE OUTPUT:

****

**SOURCE CODE**

import time

# Simulated sensor functions

def read\_ultrasonic\_sensor():

# Simulated ultrasonic sensor (adjust as needed)

return 20 + (time.time() % 10)

def read\_rain\_sensor():

# Simulated rain sensor (adjust as needed)

return time.time() % 2

def send\_warning\_message():

print("Flood detected! Sending warning message...")

# Main monitoring loop

while True:

try:

ultrasonic\_distance = read\_ultrasonic\_sensor()

rain\_intensity = read\_rain\_sensor()

if rain\_intensity > 0.5:

print("Rain detected. Flood risk!")

if ultrasonic\_distance < 30:

print(f"Flood detected. Water level: {ultrasonic\_distance} cm")

send\_warning\_message()

time.sleep(1) # Adjust the interval as needed

except KeyboardInterrupt:

break

**// C++ code**

//

int PIR = 0;

int Distance = 0;

long readUltrasonicDistance(int triggerPin, int echoPin)

{

pinMode(triggerPin, OUTPUT); // Clear the trigger

digitalWrite(triggerPin, LOW);

delayMicroseconds(2);

// Sets the trigger pin to HIGH state for 10 microseconds

digitalWrite(triggerPin, HIGH);

delayMicroseconds(10);

digitalWrite(triggerPin, LOW);

pinMode(echoPin, INPUT);

// Reads the echo pin, and returns the sound wave travel time in microseconds

return pulseIn(echoPin, HIGH);

}

void setup()

{

pinMode(13, INPUT);

pinMode(12, OUTPUT);

pinMode(6, OUTPUT);

}

void loop()

{

PIR = digitalRead(13);

delay(10); // Wait for 10 millisecond(s)

if (PIR == HIGH) {

digitalWrite(12, HIGH);

delay(1); // Wait for 1 millisecond(s)

} else {

digitalWrite(12, LOW);

}

Distance = 0.01723 \* readUltrasonicDistance(5, 4);

if (Distance <= 100) {

tone(6, 880, 125); // play tone 69 (A5 = 880 Hz)

delay(125); // Wait for 125 millisecond(s)

} else {

noTone(6);

}

}