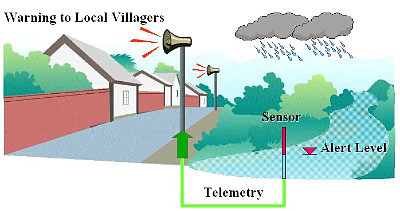
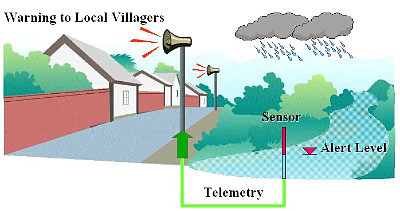
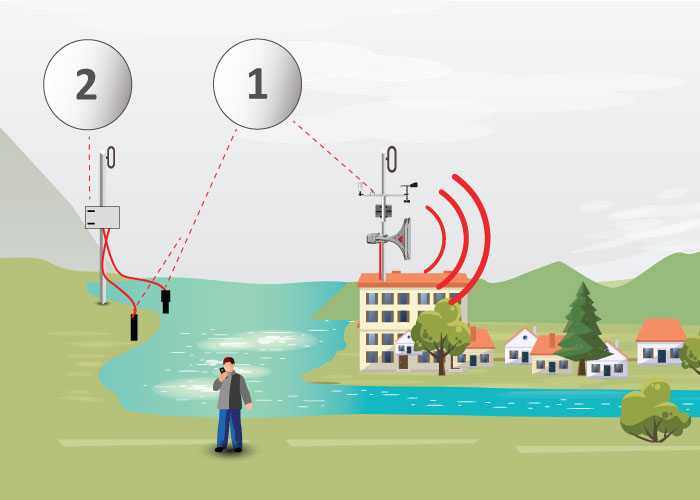
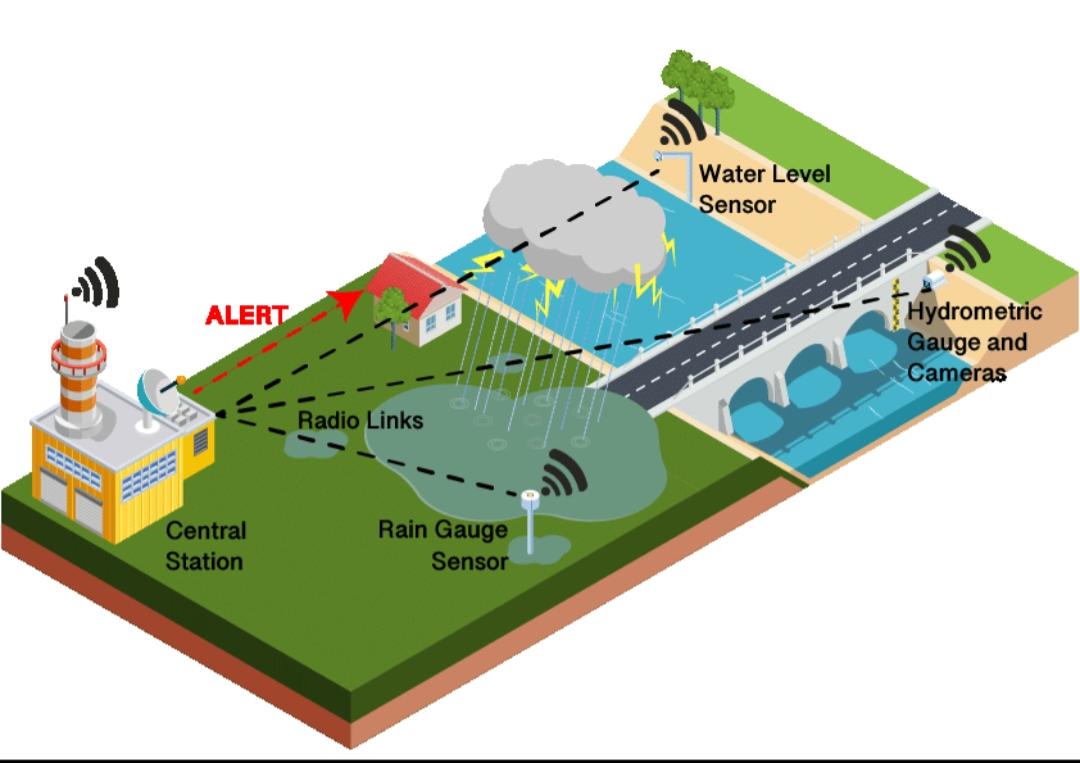
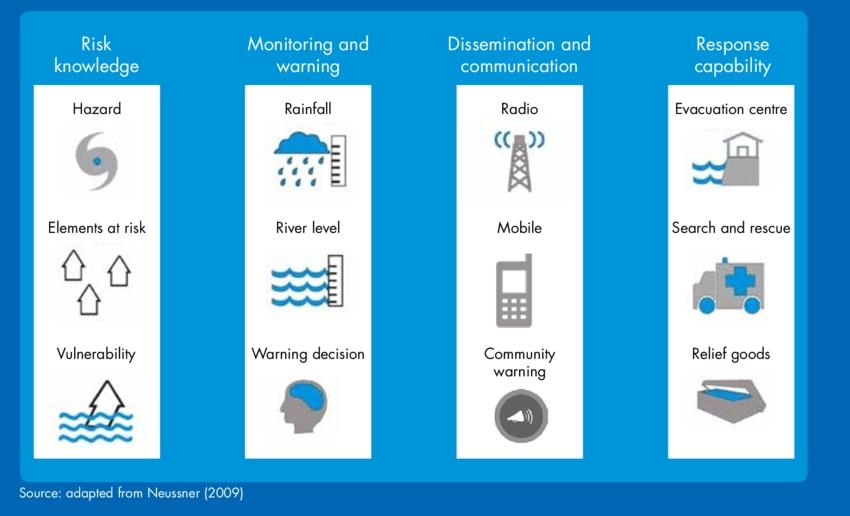
FLOOD MONITORING AND EARLY WARNING SYSTEM





# here’s the algorithm in simpler terms:

1.Collect Data Function:

Imagine we have a device that measures river water level and rainfall.

We make up random numbers to pretend we’re getting readings from these devices.

We return these pretend readings.

2.Generate Warning Function:

We set certain levels that we consider dangerous for the river water level and rainfall.

We check if our pretend data is higher than these danger levels.

If both are too high, we say there’s a flood warning for both.

If only the river level is too high, we warn about that.

If only the rainfall is too high, we warn about that.

If neither is too high, we say there’s no flood warning.

3.Main Simulation:

We start a loop that goes on forever (until we stop it).

In this loop, we:

Pretend to collect data (river level and rainfall).

Pretend to generate a warning based on this data.

Get the current time.

Print the time and the warning.

Wait for 15 minutes before doing it all over again.

So, it’s like pretending to check data from devices, deciding if there’s a flood, and logging this information every 15 minutes.

**program**

import random

Import time

# Simulated data collection function

Def collect\_data():

# Simulate data from a river gauge and rainfall sensor

River\_level = random.uniform(0.0, 10.0) # Simulated river level

Rainfall = random.uniform(0.0, 50.0) # Simulated rainfall

Return river\_level, rainfall

# Simulated warning generation function

Def generate\_warning(river\_level, rainfall):

# Set threshold values (you should determine appropriate values)

River\_level\_threshold = 8.0

Rainfall\_threshold = 30.0

# Check if thresholds are exceeded

If river\_level > river\_level\_threshold and rainfall > rainfall\_threshold:

Return “Flood Warning: River level and rainfall exceed thresholds”

Elif river\_level > river\_level\_threshold:

Return “Flood Warning: River level exceeds threshold”

Elif rainfall > rainfall\_threshold:

Return “Flood Warning: Rainfall exceeds threshold”

Else:

Return “No Flood Warning”

# Main simulation loop

While True:

river\_level, rainfall = collect\_data()

warning = generate\_warning(river\_level, rainfall)

timestamp = time.strftime("%Y-%m-%d %H:%M:%S")

print(f"{timestamp} - {warning}")

# Simulate data collection interval (e.g., every 15 minutes)

time.sleep(900) # Sleep for 900 seconds (15 minutes)

**Main.py:**

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)

**Diagram.jason:**

{

"version": 1,

"author": "keerthana",

"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": {}

}

