Objective:

* Flood Monitoring and Early warning to the people about Flood occurring.
* This can be done using ULTRASONIC

Sensors.

* There are simple Audrino code which

reads the water level from the sensors

and triggers the alarm.

* Use the ESP32 DevKit in Wokwi web-

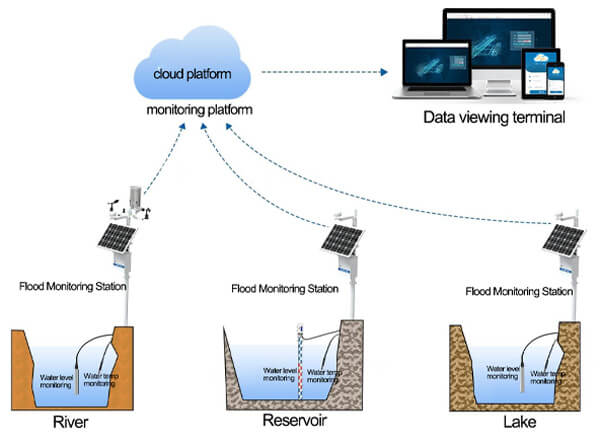
site to simulate the flood monitoring

code.

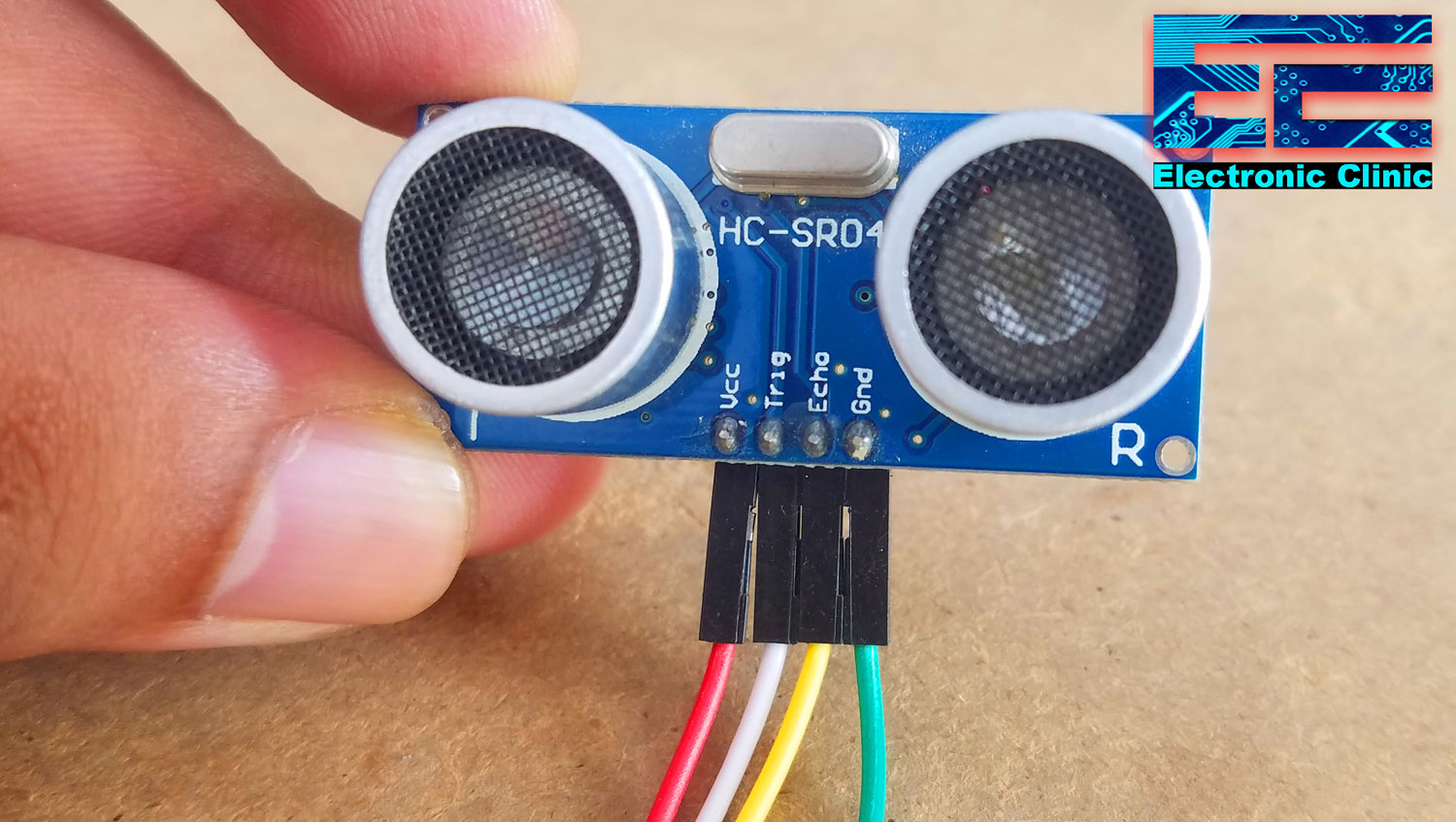
* This may give a correct warning so that we can avoid the death of the

fisherman by alerting them.

Flood Monitoring:



Ultrasonic sensor:



My Idea for solving this problem…

* To monitor the flood

we are choosing the ultrasonic sensors and

float sensors in the system.

* For early warning the ESP32 Dev Kit is used for the alarm setting process.
* Simulating the ESP32 kit using adurino coding

that I have already created for the alarm process.

Adurino Coding for Simulation…

#include <Keypad.h>

#include "Clock.h"

#include "Weather.h"

#include <WiFi.h>

#include <HTTPClient.h>

#include <ArduinoJson.h>

#include <LiquidCrystal\_I2C.h>

#define I2C\_ADDR    0x27

#define LCD\_COLUMNS 16

#define LCD\_LINES   2

uint8\_t state;

unsigned long clockTimer;

unsigned long weatherAPItimer;

unsigned long weatherDisplayTimer;

const char\* password = "";

const char\* ssid = "Wokwi-GUEST";

uint8\_t valIndex;

uint8\_t cursorPos;

char entered\_value [6];

const uint8\_t ROWS = 4;

const uint8\_t COLS = 4;

char keys[ROWS][COLS] = {

  { '1', '2', '3', 'A' },

  { '4', '5', '6', 'B' },

  { '7', '8', '9', 'C' },

  { '\*', '0', '#', 'D' }

};

uint8\_t colPins[COLS] = { 1, 0, 3, 2 };

uint8\_t rowPins[ROWS] = { 4, 5, 6, 7 };

LiquidCrystal\_I2C lcd(I2C\_ADDR, LCD\_COLUMNS, LCD\_LINES);

Clock rtc(&lcd);

Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);

void enterTime()

{

  state = 1;

  memset(&entered\_value[0], 0, sizeof(entered\_value));

  cursorPos = 0;

  valIndex = 0;

  lcd.clear();

  lcd.setCursor(0, 0);

  lcd.print("Set clock then  ");

  lcd.setCursor(0, 1);

  lcd.print("press # to save.");

  delay(3000);

  lcd.clear();

}

void enterAlarm()

{

  state = 2;

  memset(&entered\_value[0], 0, sizeof(entered\_value));

  cursorPos = 0;

  valIndex = 0;

  lcd.clear();

  lcd.setCursor(0, 0);

  lcd.print("Enter alarm time");

  lcd.setCursor(0, 1);

  lcd.print("press # to save.");

  delay(3000);

  lcd.clear();

}

void nextChar(char key)

{

  if (valIndex < 6)

  {

    entered\_value[valIndex] = key;

    lcd.setCursor(0, 0);

    lcd.print(entered\_value);

    cursorPos++;

    valIndex++;

  }

}

void eraseChar()

{

  if (valIndex > 0 )

  {

    valIndex--;

    cursorPos--;

    entered\_value[valIndex] = '\0';

    lcd.setCursor(cursorPos, 0);

    lcd.print(' ');

    lcd.setCursor(cursorPos, 0);

  }

}

void keyPadState0()

{

  char key = keypad.getKey();

  switch(key)

  {

    case 'A':

      enterAlarm();

      break;

    case 'C':

      enterTime();

      break;

    case '#':

      rtc.silence();

      break;

    case '\*':

      rtc.addToSnooze();

      break;

  }

}

void keyPadState1()

{

  char key = keypad.getKey();

  switch(key)

  {

    case '#':

      rtc.setTime(entered\_value);

      state = 0;

      break;

    case '\*':

      eraseChar();

      break;

    default:

      if (isDigit(key))

      {

        nextChar(key);

      }

      break;

  }

}

void keyPadState2()

{

  char key = keypad.getKey();

  switch(key)

  {

    case '#':

      rtc.setAlarm(entered\_value);

      state = 0;

      break;

    case '\*':

      eraseChar();

      break;

    default:

      if (isDigit(key))

      {

        nextChar(key);

      }

      break;

  }

}

void getInput()

{

  switch (state)

  {

    case 0:

      keyPadState0();

      break;

    case 1:

      keyPadState1();

      break;

    case 2:

      keyPadState2();

      break;

  }

}

void setup()

{

  pinMode(8, OUTPUT);

  pinMode(10, OUTPUT);

**Wire**.begin(18, 19);

  lcd.init();

  lcd.backlight();

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL\_CONNECTED)

  {

    lcd.print(".");

    delay(1000);

  }

  lcd.clear();

  if (!rtc.setTimeFromAPI())

  {

    char t [] = \_\_TIME\_\_;

    char compileTime [] = { t[0], t[1], t[3], t[4], t[6], t[7] };

    rtc.setTime(compileTime);

  }

  getWeather(lcd);

  printWeather(lcd);

}

void loop()

{

  if (state == 0)

  {

    unsigned long millisNow = millis();

    if (millisNow - clockTimer >= 1000)

    {

      clockTimer = millisNow;

      rtc.updateClock();

    }

    if (millisNow - weatherDisplayTimer >= 10000)

    {

      weatherDisplayTimer = millisNow;

      printWeather(lcd);

    }

    if (millisNow - weatherAPItimer >= 3600000)

    {

      weatherAPItimer = millisNow;

      getWeather(lcd);

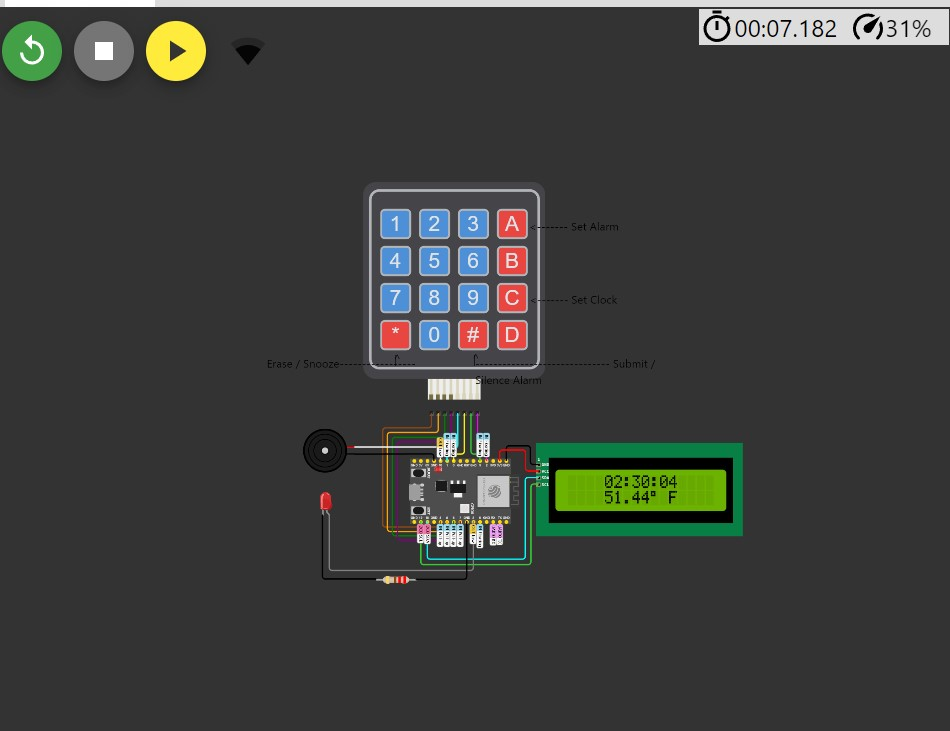
    }

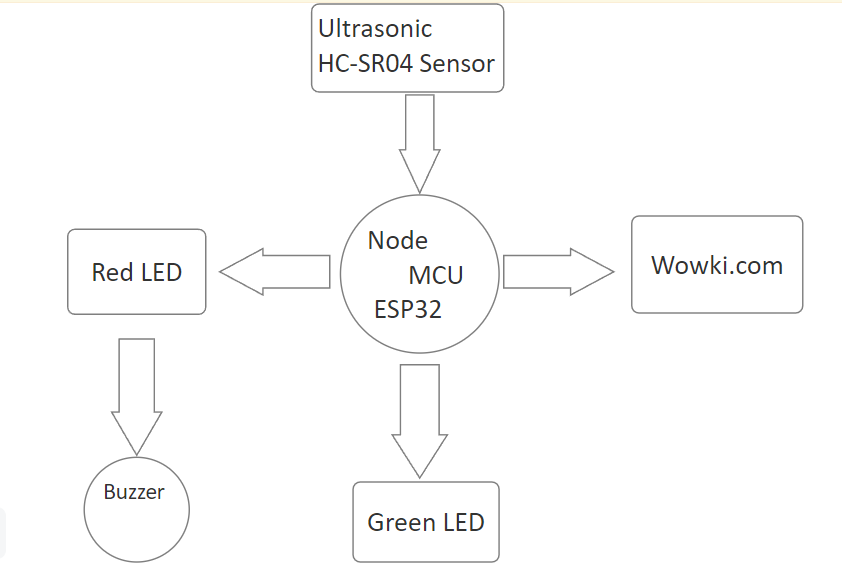
  }

  getInput();

}

Simulation process…



Block Diagram

Overview of my project model…

