2023

Flood monitoring and early warning system

PHASE 4- DEVELOPMENT PART 2

METHA R.S

UNIVERSITY COLLEGE OF ENGINEERING , | Ariyalur

Feature engineering

Software requirements-

Arduino ide

The Arduino Integrated Development Environment (IDE) is a software application that provides a user-friendly interface for programming, compiling, and uploading code to Arduino-compatible microcontroller boards. Here's a brief explanation of the main features and functions of the Arduino IDE: Code Editor: The IDE includes a text editor where you write and edit your Arduino code. It supports the Arduino programming language, which is based on C and C++.Library Management: The IDE allows you to easily include and manage libraries, which are collections of pre-written code that extend the functionality of your Arduino projects. This simplifies complex tasks by providing ready-made functions and classes. Verification and Compilation: Once you've written your code, you can verify and compile it by clicking the "Checkmark" or "Upload" button. This process checks for syntax errors and generates a binary file that can be uploaded to the microcontroller .Serial Monitor: The IDE includes a serial monitor that allows you to send and receive data between your computer and the Arduino board. This is essential for debugging and real-time communication . Board Manager: You can select the specific type of Arduino board you're using, and the IDE will set the appropriate configurations for that board. It also allows you to install additional board definitions for non-standard Arduino-compatible boards. Port Selection: You can choose the communication port that your Arduino board is connected to, typically via USB. This is used for uploading the compiled code to the board.

Sketch Structure: Arduino code is organized into "sketches." A sketch typically contains two main functions: setup() (for initialization) and loop() (for the main program execution). You can write your code within these functions.

Programming Language Reference: The IDE provides access to the Arduino programming language reference, which documents the functions and libraries available for use in your projects.

Code Examples: The IDE includes a wide range of code examples that cover various aspects of Arduino programming. These examples serve as templates and learning resources.

Version Control: You can use version control systems (e.g., Git) in combination with the Arduino IDE to track changes in your code and collaborate with others on your projects.

Compatibility: The Arduino IDE is compatible with Windows, macOS, and Linux, making it accessible to a wide range of users.

Arduino library

Arduino libraries are pre-written sets of code that provide useful functions and tools for specific hardware components or tasks. When working on Arduino projects, you often need to include libraries to simplify and extend the functionality of your code. Here's a brief explanation of the libraries mentioned:

Ultrasonic Library: This library is used to interface with ultrasonic distance sensors, such as the HC-SR04 or JSN-SR04T. It simplifies the process of sending a pulse to the sensor and measuring the time it takes for the pulse to return, which can be used to calculate the distance to an object. The Ultrasonic library typically provides functions to initialize the sensor, trigger measurements, and read the distance in different units (e.g., centimeters, inches). Including this library in your project allows you to work with ultrasonic sensors without writing low-level code.

Thing Speak Library: Thing Speak is a cloud service that allows you to collect, store, and visualize data from your IoT projects. The Thing Speak library provides functions for interfacing with the Thing Speak platform through APIs. With this library, you can easily send data from your Arduino to your Thing Speak channel, making it available for real-time monitoring and analysis. It simplifies the process of formatting and sending data to Thing Speak 's servers, where you can create charts and visualizations based on the data received.

W i Fi Library: The Wi Fi library is a core library included with the Arduino IDE. It's used to establish a Wi-Fi connection between your Arduino board (e.g., an ESP8266 or ESP32) and a Wi-Fi network. The library provides functions for connecting to Wi-Fi networks, configuring network settings, and checking the status of the connection. It's essential for IoT projects that require internet connectivity.

To use these libraries in your Arduino projects, you typically need to:

- Include the library in your Arduino sketch by adding #include <Library Name. h> at the beginning of your code, where Library Name is the name of the library you want to use.
- Initialize and configure the library, such as setting up your Wi-Fi connection or initializing the ultrasonic sensor.
- Use the library's functions and methods to perform specific tasks, like reading sensor data, sending data to a cloud service, or connecting to Wi-Fi.

Wi fi network

You need access to a Wi-Fi network and its credentials (SSID and password) to connect your ESP32 or Arduino board to the internet.

Hardware Requirements –

Micro controller board -Microcontroller Board: You need an Arduino-compatible board (e.g., Arduino Uno, ESP32) to run your code. The board should have digital pins to connect the ultrasonic sensor and a buzzer. In your specific case, you mentioned using an ESP32.

Ultrasonic Sensor: The program is designed to work with an ultrasonic sensor. You should have a compatible sensor, such as the HC-SR04, which typically requires two pins for trigger and echo.

Buzzer: You need a buzzer that can be driven by your microcontroller. The buzzer typically has a positive (anode) and a negative (cathode) pin. Ensure it's compatible with your microcontroller's voltage levels.

Connections: Make sure you have appropriate wires and connectors to establish connections between the microcontroller, ultrasonic sensor, and buzzer.

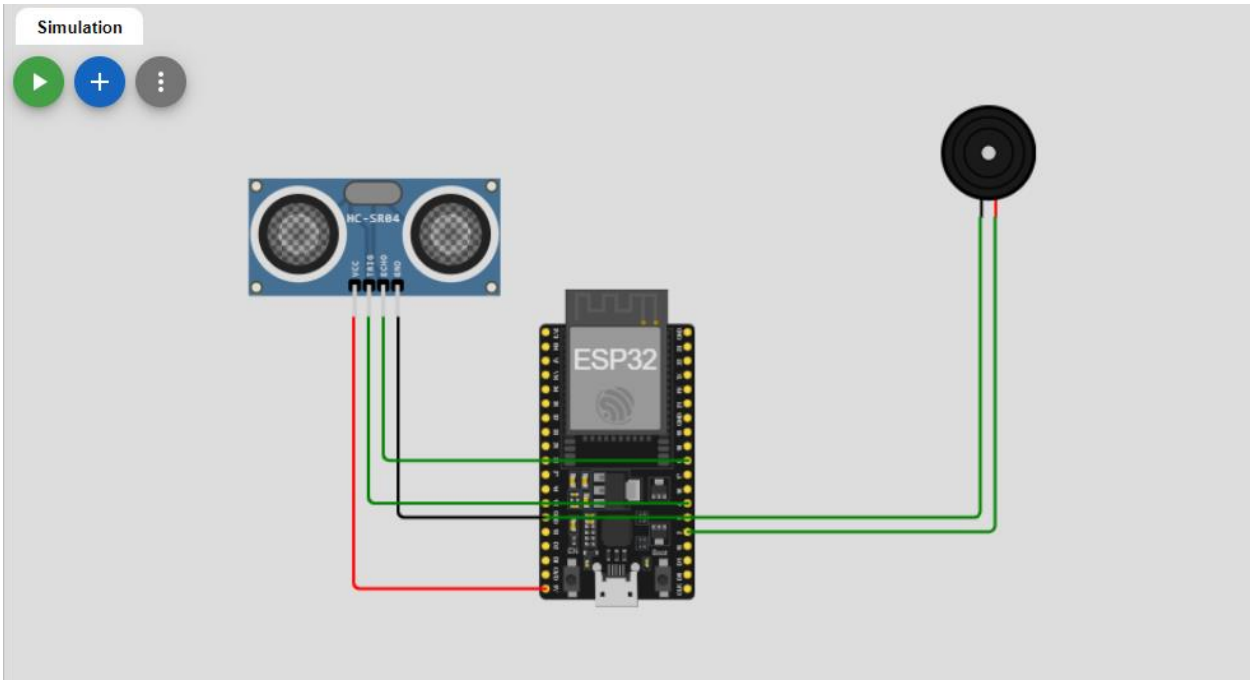
Power Supply: Ensure your microcontroller board is powered, either through a USB connection, an external power supply, or a battery, depending on your project's requirements.

Computer: A computer with the Arduino IDE installed to write, compile, and upload your code to the microcontroller.

Internet Connection: For the Thing Speak part of the code to work, your microcontroller board needs to be connected to the internet via Wi-Fi. Make sure your Wi-Fi network is functioning correctly.

Model training

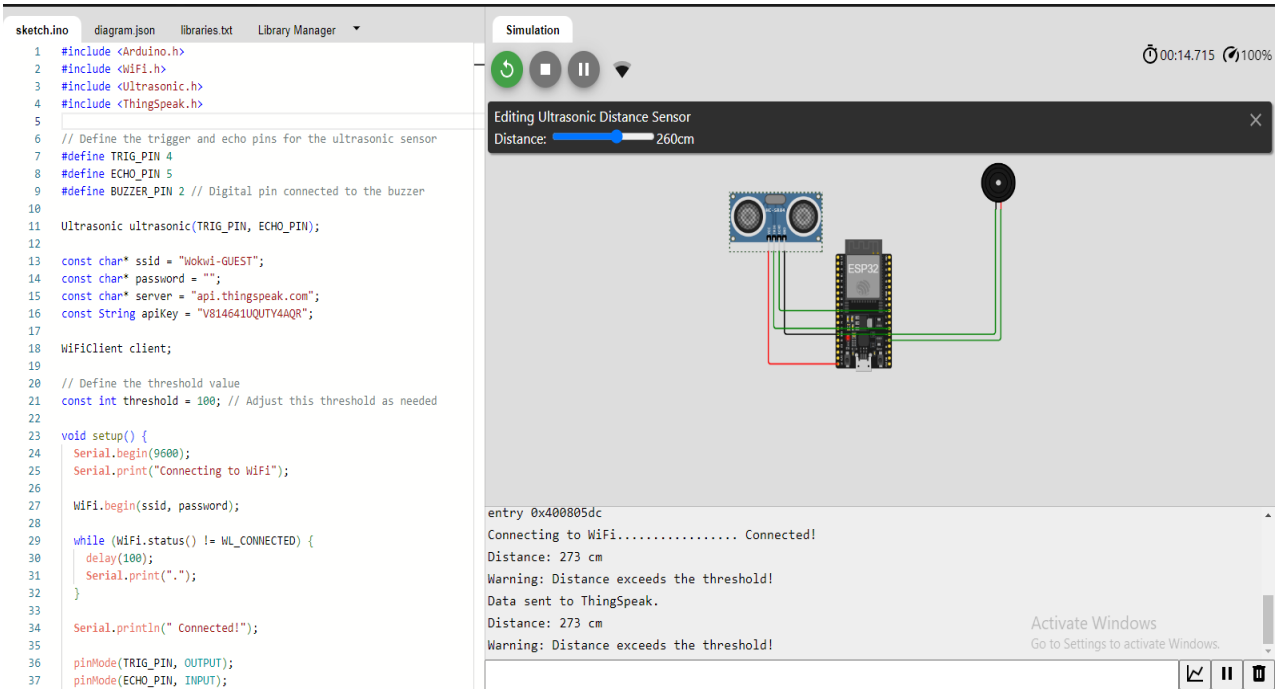
The ultrasonic sensor (HC-SR04 or similar) is used to measure distances to objects. It sends a high-frequency sound pulse and calculates the time it takes for the pulse to bounce back to the sensor. The distance is then computed using the speed of sound. The program connects to a Wi-Fi network using the ESP32's Wi-Fi capabilities .It periodically reads the distance from the ultrasonic sensor .If the measured distance exceeds a predefined threshold, it triggers a warning message in the serial monitor .The program sends the distance data to the Thing Speak cloud platform, which allows you to visualize and analyze your sensor data online.



Wokwi is a popular online electronics simulation platform that allows you to simulate and test your Arduino, ESP8266, and ESP32 projects in a virtual environment. It provides a user-friendly interface for building and simulating circuits, making it a great tool for learning and prototyping without physical hardware.To access the Wokwi Simulator, you can follow a series of steps to set up and run your virtual electronics project. Firstly, you should visit the Wokwi website at https://wokwi.com/. While creating an account on Wokwi is optional, it offers the advantage of saving your simulations and circuits across multiple devices for future access. Once you're on the platform, you can initiate your project by clicking the "Create a New Circuit" button, which will serve as the foundation for your simulation.

In order to replicate your real-world setup, you'll need to populate your virtual circuit with the appropriate components. Wokwi provides a user-friendly "Components" panel on the right, allowing you to search and select the components you require. To mirror your specific project, ensure that you include all the necessary elements.In the context of the provided MicroPython

code for an ESP32 and an ultrasonic sensor, you should set up these components in your virtual circuit. This involves adding virtual representations of the ESP32 and the ultrasonic sensor and establishing the connections between them as specified in your code.To breathe life into your project, you can proceed to the code editor section by clicking on the "Code" tab located in the left panel. Here, you can paste your MicroPython code, including the logic for controlling the components and performing any actions you want to simulate. Furthermore, if your code involves Wi-Fi connectivity for your ESP32, you have the option to configure a virtual Wi-Fi network within Wokwi. You can specify the SSID and password in your code to simulate the ESP32 connecting to a wireless network.When you're ready to observe the behavior of your project, you can click the "Run" button, initiating the simulation. Throughout the simulation, you can monitor the output via the virtual serial monitor, which is particularly useful for debugging and understanding how your code interacts with the components. Additionally, you can interact with the virtual components by using buttons, switches, and other input methods.Throughout the simulation, it's crucial to analyze the results. This includes observing the behavior of the virtual components, any visualizations you've incorporated into your code, and, most importantly, the data produced during the simulation. As you run the simulation, you have the opportunity to detect any issues or unexpected behaviors in your code, which can be invaluable for debugging and refinement.Should you wish to revisit your simulations later or share your projects with others, Wokwi allows you to save your circuits and simulations. This feature is particularly advantageous if you have created a Wokwi account, as it enables you to access your projects across various devices and share your work with colleagues or the wider community. Overall, the Wokwi Simulator provides a versatile and interactive platform for electronics enthusiasts, learners, and developers to test, validate, and fine-tune their projects in a virtual environment.



Arduino code

```
#include <Arduino.h>

#include <WiFi.h>
#include <Ultrasonic.h>
#include <ThingSpeak.h>

// Define the trigger and echo pins for the ultrasonic sensor
#define TRIG_PIN 4
#define ECHO_PIN 5
#define BUZZER_PIN 2 // Digital pin connected to the buzzer

Ultrasonic ultrasonic(TRIG_PIN, ECHO_PIN);

const char* ssid = "Wokwi-GUEST";
const char* password = "";
const char* server = "api.thingspeak.com";
const String apiKey = "V814641UQUTY4AQR";

WiFiClient client;

// Define the threshold value
const int threshold = 100; // Adjust this threshold as needed

void setup() {
  Serial.begin(9600);
  Serial.print("Connecting to WiFi");

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(100);
    Serial.print(".");
  }

  Serial.println(" Connected!");

  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  pinMode(BUZZER_PIN, OUTPUT); // Set the buzzer pin as an output
}

void loop() {
  // Read the distance from the ultrasonic sensor
```

```
  int distance = ultrasonic.read();

  // Print the distance to the serial monitor
  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" cm");

  // Check if the distance exceeds the threshold
  if (distance > threshold) {
    Serial.println("Warning: Distance exceeds the threshold!");
    // Sound the buzzer
    digitalWrite(BUZZER_PIN, HIGH);
  } else {
    digitalWrite(BUZZER_PIN, LOW); // Turn off the buzzer if the distance
is within the threshold
  }

  // Send the data to ThingSpeak
  if (client.connect(server, 80)) {
    String postStr = "field1=" + String(distance);
    postStr += "&key=" + apiKey;

    client.println("POST /update HTTP/1.1");
    client.println("Host: " + String(server));
    client.println("Connection: close");
    client.println("Content-Type: application/x-www-form-urlencoded");
    client.println("Content-Length: " + String(postStr.length()));
    client.println();
    client.println(postStr);

    delay(500); // Wait for the response

    Serial.println("Data sent to ThingSpeak.");
    client.stop();
  }

  delay(10000); // Delay for a longer interval before the next reading
(e.g., 10 seconds)
}
```

Things speak credentials

Channel ID: **2326843**

Author: mwa0000031493476

Access: Public

API key: V814641UQUTY4AQR

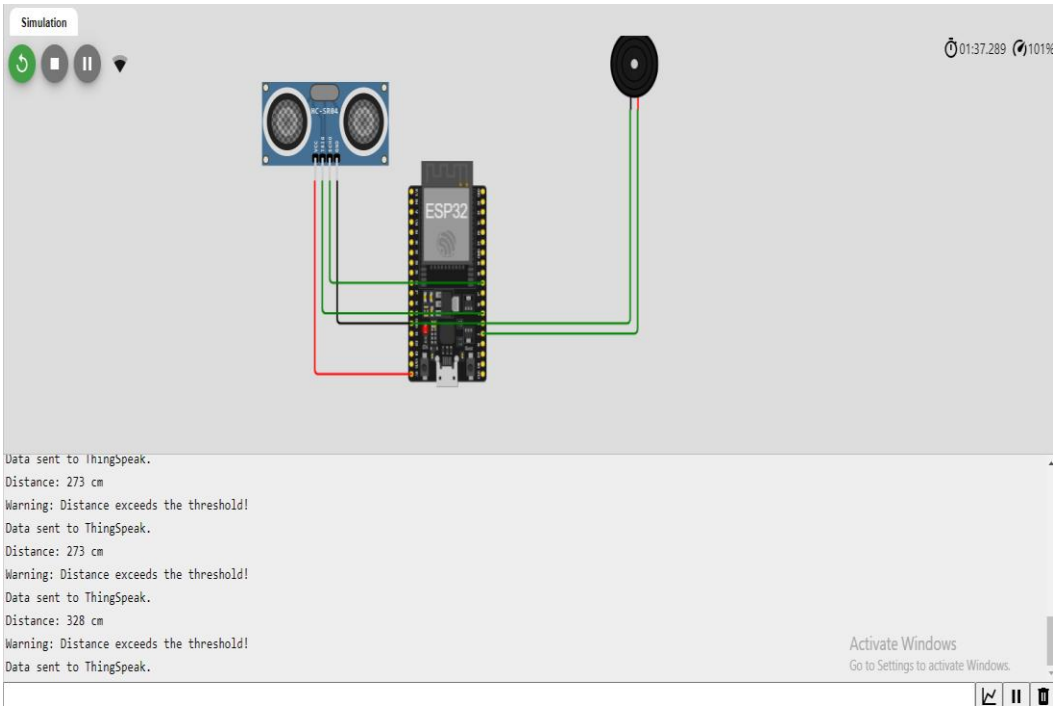
Channel ID (2326843): This is a unique identifier for a specific ThingSpeak channel. ThingSpeak channels are used to collect and store data from various IoT devices. Each channel is assigned a distinct Channel ID, which helps differentiate one channel from another. In your case, Channel ID 2326843 is the identifier for the specific channel where data is being sent or retrieved.

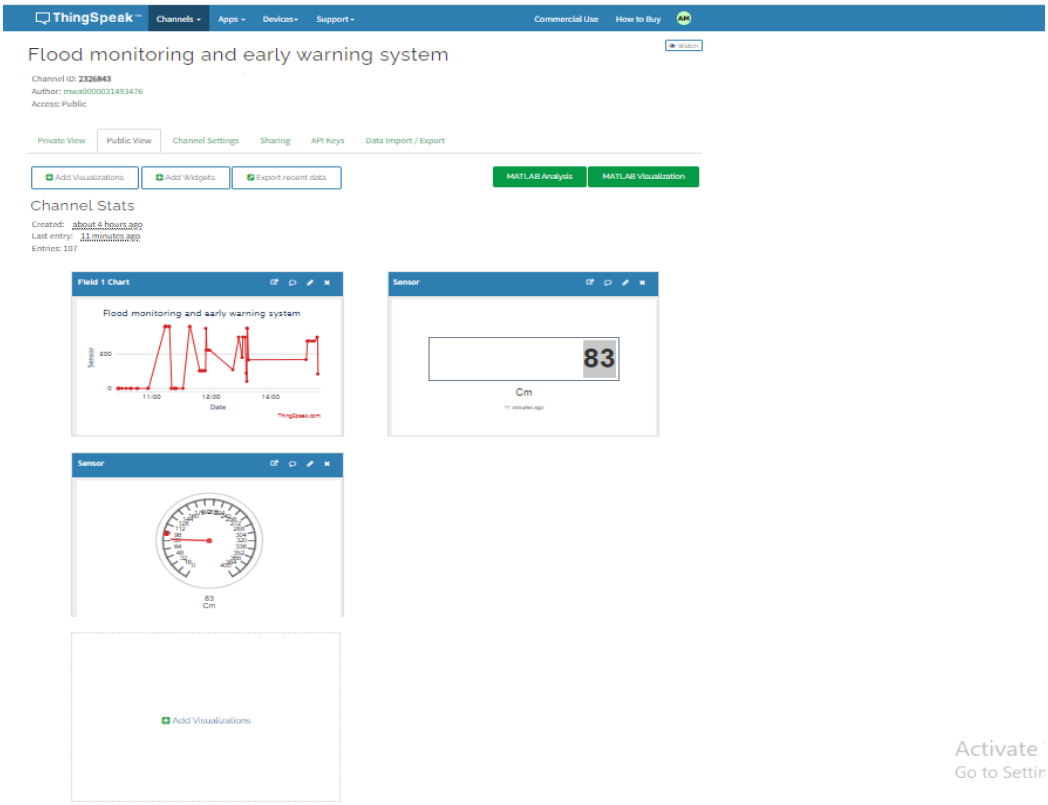
Author (mwa0000031493476): The "Author" typically represents the creator or owner of the ThingSpeak channel. It is the username or identifier associated with the individual who established and manages the channel. The author has control over the channel's settings and access permissions.

Access (Public): "Public" denotes the access level or visibility of the ThingSpeak channel. In this context, a "public" channel implies that the data stored in this channel is accessible to anyone, not just the channel owner. Public channels are often used for sharing data with the wider public or for open-source projects. Access can also be set to "private," where data is restricted to the channel owner or authorized users only.

API Key (V814641UQUTY4AQR): An API Key is a unique and secure authentication key that allows external applications or devices to interact with the ThingSpeak channel. It serves as a secure way to send data to or retrieve data from the channel. API Keys provide a level of security and control, ensuring that only authorized entities can access or modify the data in the channel. In your case, "V814641UQUTY4AQR" is the specific API Key associated with the channel, which can be used to send data to it.

Outputs:


Simulation
01:37.289
ESP32
Data sent to ThingSpeak.
Distance: 273 cm
Warning: Distance exceeds the threshold!
Data sent to ThingSpeak.
Distance: 273 cm
Warning: Distance exceeds the threshold!
Data sent to ThingSpeak.
Distance: 328 cm
Warning: Distance exceeds the threshold!
Data sent to ThingSpeak.
Activate Windows
Go to Settings to activate Windows.


ThingSpeak
Channels
Apps
Devices
Support
Commercial Use
How to Buy
Flood monitoring and early warning system
Channel ID: 2326843
Author: mwa0000031493476
Access: Public
Private View
Public View
Channel Settings
Sharing
API Keys
Data Import / Export
Add Visualizations
Add Widgets
Export recent data
MATLAB Analysis
MATLAB Visualization
Channel Stats
Created: about 4 hours ago
Last entry: 11 minutes ago
Entries: 107
Field 1 Chart
Flood monitoring and early warning system
Sensor
83
Cm
Sensor
83
Cm
Add Visualizations
Activate
Go to Settin

Project zip - ...\Downloads\flood monitoring and early warning system.zip

Wokwi link - https://wokwi.com/projects/380179605446953985

Things speak link -https://thingspeak.com/channels/2326843