Instruction Manual

# Overview

This project automates the monitoring and classification of water for irrigation using sensor data and a machine learning model. It sends results to Firebase where the user can view the water suitability information and optionally notifies users via email when water is unsuitable.

# Tools/ services used

|  |  |
| --- | --- |
| Tools/ service | Description of use |
| Firebase | Used for web frontend and for the Realtime database feature, which allows seamless real-time updating of the database. |
| Arduino Cloud | Used to program the Arduino and view the serial monitor for configuration and debugging. |
| GitHub | Hosts the backend code (Flask + ML classifier), which is deployed via Render. |
| Render | Cloud service used to deploy and run the Flask backend that handles classification and email notifications. |

# Setup

This system uses an ESP32 microcontroller to collect environmental data, which is sent to a Flask server hosted on Render. The server processes and classifies the data using an AI model, logs it to Firebase Realtime Database, and sends notifications based on user settings configured via a web frontend. This manual explains how to set up each component of the system.

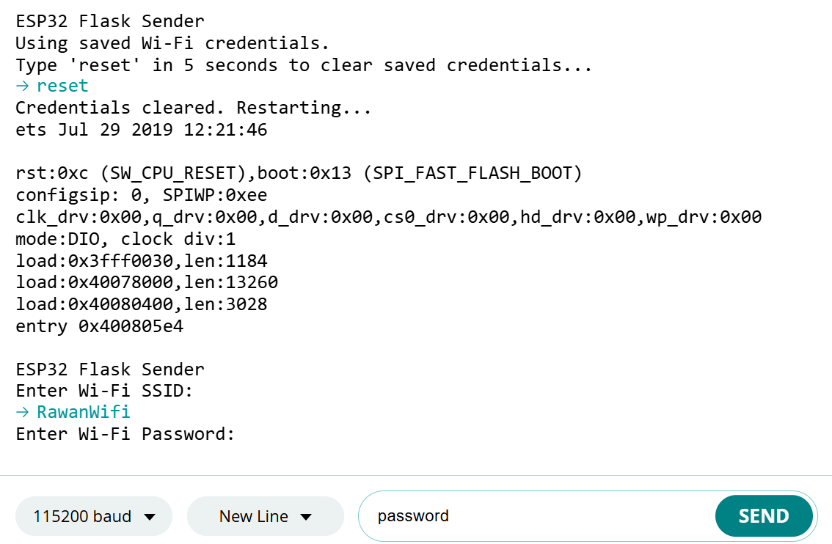
# Arduino Cloud:

* Sign up at Arduino Cloud.
* Add a new Sketch.
  + Insert the code to read from water quality sensors.
  + In the code make sure HTTP request logic to send sensor readings to the Flask server have the correct link and that it ends with(/classify). For example: http.begin( "https://your-flask-app.onrender.com/classify");
* Upload the sketch to the Arduino.
* The Arduino should connect to the last saved network automatically, if the network is not saved or need to be changed, then setup is required.
  + Open the serial monitor, if there is already a saved network type 'reset'

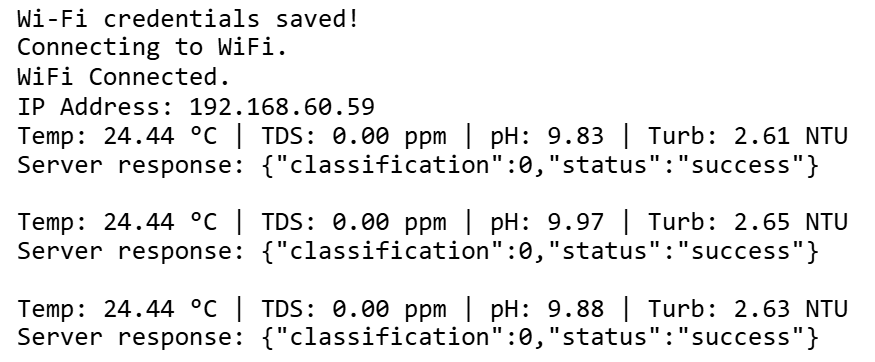
A screenshot of a computer

AI-generated content may be incorrect.

* + A message should appear and prompt the user to enter the network SSID and password



* + A message should appear to indicate that the credentials were saved.



* If all steps are done properly, the Arduino should be now able to send the readings to the flask server that is hosted on render.

# GitHub + Render

* Push your Flask project to a GitHub repository.
* Project should include those files:

|  |  |
| --- | --- |
| **File Name** | **Purpose** |
| .env | Stores sensitive environment variables like email credentials; **not pushed** to GitHub if .gitignore is properly configured. This is the file you must edit If you want to change the email source. It contains two variable one for the email and the other for the email app password (you can get that by following this link and creating an app: <https://myaccount.google.com/apppasswords>). Note: format is important here the variables should NOT have spaces or quotation marks ("). |
| .gitignore | Specifies which files/folders Git should ignore (e.g. .env, \_\_pycache\_\_/, .pkl files). |
| README.md | Contains basic documentation or description of the project, visible on GitHub's homepage. |
| app.py | Main Flask server code — handles data from the ESP32, runs the ML model, logs to Firebase, sends notifications. Make sure it gets the credentials from firebase-service-account.json file and includes a link to the database, for example:  cred = credentials.Certificate("firebase-service-account.json")  firebase\_admin.initialize\_app(cred, {  'databaseURL': 'https://aquaguard-tkmj6-default-rtdb.firebaseio.com/'  }) |
| firebase-service-account.json | Firebase Admin SDK credentials file used by the Flask server to access Firebase services securely. |
| render.yaml | Render deployment configuration file — defines what command to run, environment, and build instructions. |
| requirements.txt | Lists all Python libraries (like Flask, scikit-learn, firebase-admin) needed to run app.py. Make sure it includes all python library used as well as firebase-admin. |
| scaler.pkl | Serialized Python object (via pickle) that normalizes input sensor data before it’s fed to the ML model. |
| svm\_model.pkl | Pre-trained Support Vector Machine model file used to classify water as suitable or unsuitable. |

* Sign in with GitHub on Render
* When you sign up or log in to Render, you can choose “Sign in with GitHub.”
* This grants Render permission to access your GitHub repositories.
* Authorize Render to access your GitHub repositories
  + The first time you link GitHub, GitHub will ask you to authorize Render.
  + You can choose to allow access to all repositories or select specific ones.
* Create a New Web Service on Render
  + Click “New +” > “Web Service”
  + You’ll be asked to select a repository from your GitHub account.
  + Choose the correct repository (e.g., the one shown in your screenshot).
* Configure the service
  + Build Command: (e.g., pip install -r requirements.txt)
  + Start Command: (e.g., python app.py)
  + Environment Variables: Add any necessary variables like EMAIL\_USER, EMAIL\_APP\_PASSWORD, etc.
  + Runtime: Select Python and appropriate version.
* Deploy
  + Render pulls the code from GitHub and deploys it.
  + Any changes you push to GitHub can automatically redeploy the app if auto-deploy is enabled.
* Make sure the link is the same as the one uploaded in the Arduino, by clicking on the webservice.

# Firebase

1. Firebase console

* Go to Firebase Console and create a new project.
* Set Up the Realtime Database
  + - In the Firebase Console, select your project.
    - In the left sidebar, click “Build” > “Realtime Database”
    - Click “Create Database”
    - Select the region closest to you.
    - Choose Start in test mode (you can adjust rules later).
* Set Database Rules:

{

"rules": {

"sensor\_data\_classified": {

".read": true,

".write": true

}

}

}

* Create a Service Account for Server Access, so that the server knows were to save the readings and result
  + - Navigate to Project Settings (gear icon > Project settings)
    - Click the “Service Accounts” tab.
    - Click “Generate New Private Key”
    - This downloads a .json file (e.g., firebase-service-account.json) Save it in the GitHub repository.

1. Firebase studio for Easy GUI Management
   * Sign in with the same Google account.
   * Make sure .env file in the firebase project includes those variables and that they are correctly filled:
     + NEXT\_PUBLIC\_FIREBASE\_API\_KEY=
     + NEXT\_PUBLIC\_FIREBASE\_AUTH\_DOMAIN=
     + NEXT\_PUBLIC\_FIREBASE\_DATABASE\_URL=
     + NEXT\_PUBLIC\_FIREBASE\_PROJECT\_ID=
     + NEXT\_PUBLIC\_FIREBASE\_STORAGE\_BUCKET=
     + NEXT\_PUBLIC\_FIREBASE\_MESSAGING\_SENDER\_ID=
     + NEXT\_PUBLIC\_FIREBASE\_APP\_ID=