AquaGrow – A Smart Aquaponics Gardening System

Design Specification

Project by:
Tam Nguyen
Department of Computer and Information Science
(662) 371-5507
tnnguye1@go.olemiss.edu

Project sponsor:
Eric Hodges, M.Sc.
Department of Biomolecular Sciences
(405) 249-9733
ehodges@go.olemiss.edu

Project Overview

"Aquaponics is a revolutionary system for growing plants by fertilizing them with the waste water from fish in a sustainable closed system. A combination of aquaculture and hydroponics, aquaponics gardening is an amazingly productive way to grow organic vegetables, greens, herbs, and fruits, while providing the added benefits of fresh fish as a safe, healthy source of protein." – Sylvia Bernstein

Currently, aquaponics system owners have to monitor many aspects of their systems such as temperature, sunlight, pH level, nitrate level, water level, etc. In order to maintain an optimal growing condition for plants, these aspects must be regularly supervised, most often in a manual way. Therefore, there is a demand in a flexible, automated and electrically-controlled aquaponics system. This project aims to create such product by combining the use of multiple sensors, a Wi-Fi integrated microcontroller and a web interface. It will allow users to monitor and control their aquaponics systems remotely and effortlessly.

User Requirements

The main target users of AquaGrow are those who own an aquaponics gardening setup. The user expectations for the product are as follow:

1. Regarding the hardware:
    - Monitoring features: The sensor system will provide the users with information on atmospheric temperature and humidity, growing light intensity, growing bed water level, fish tank's water temperature, fish tank's pH level. These data will then be available on a display monitor next to the aquaponics system as well as on the web interface.
    - Controlling features: The system will be able to control (turn on/off) the electric water pump inside the fish tank, adjust the intensity of the growing light, and potentially adjust the water temperature in the fish tank by a heating mat.
2. Regarding the software:
    - The users will be able to create an account and maintain a portfolio of plants they've been growing. The users can access all the information regarding their plants and current data from the sensors on any device by logging into their account.
    - The users will be provided with a suggested list of plants/herbs/vegetables that are suitable to be grown in their area (based on USDA Open Plant Hardiness Zones).
    - When users 'add' a new plant, they have the options to pick a preset of appropriate temperature, light, pH level… for the chosen plant. If they opt to not use the preset, they can set up their own desired numbers.
    - The values from above will act as the threshold for the system. The users will receive notifications when the data received from the sensors fall out of range.

- The users will also receive notifications regarding system status (whether something is on or off), fish feeding reminders, growing tips, etc.
- The users will be able to turn on/off the pump, the light and the heating mat remotely from the web interface. This is called 'manual control'. They can also use 'automatic control' by setting up the operating schedule for each of those devices so they will stay turned on/off according to the set schedule.
- The users will be provided with a dashboard. The dashboard will provide many information such as a suggested timeline of the plant(s) they're growing, real-time data of the sensors and graphs regarding those data over a period of time.

Development Environment

1. Hardware (all hardware is provided and has been ordered by the sponsor)
   - ESP32 Wi-Fi and Bluetooth integrated microcontroller: this device will be connected to all the sensors and the water pump, grow light and heating mat. It acts as a server that sends the sensors' data to our application and receive user's instructions to control the system.
   - Display module
   - Dimmable grow light
   - Submersible water pump
   - Atmospheric temperature/humidity sensor
   - Ambient light sensor
   - Non-contact water level sensor
   - Water temperature sensor
   - pH circuit and pH probe as the pH sensor

- Necessary tools for a normal aquaponics setup: fish tank, grow bed, grow medium, water pipe, filter…

2. Software technologies:
   - ESP32 web server and sensors integration: C/C++ and the Arduino IDE.
   - Backend: NodeJS, Express – provide a fast and easy way to implement a RESTful API
   - Frontend: React, Redux, React Router – dynamic user interface
   - Database: MongoDB and Mongoose. The reasons for choosing a NoSQL database (MongoDB) instead of a relational database (MySQL) are:
     - The application doesn't require a strict schema architecture.
     - MongoDB allows dynamic schema, which gives developers the flexibility to scale and modify the data schema without modifying any of the existing data.
     - It's more suitable for real-time app with a large amount of frequently incoming data.
     - The use of JavaScript across the whole technology stack (NodeJS, React and MongoDB are all technically JavaScript)
   - Socket programming – our application needs to communicate with ESP32 and/or the API asynchronously to receive the sensor data, as well as to send instructions down to control the system. Therefore, the application needs a bi-directional communication capability. Socket.io – a JavaScript library for real-time applications, will be considered to handle the communication between the web application (client) and the ESP32/our API (server).
   - SMS Notifications – providing the users the option to receive notifications via SMS with Twilio API.
   - Version control: Git and GitHub.
   - Design tools:

- Prototyping tool: Sketch/Adobe XD
- UI library: Ant Design for React

3. Datasets:
    - USDA Plant Hardiness Zone Datasets:

      http://prism.oregonstate.edu/projects/plant_hardiness_zones.php
4. Testing tools:
    - API Testing: Postman
    - Continuous Integration: Travis CI – automated testing integrated with GitHub repository.
    - Responsive website test: https://sizzy.co
    - Unit testing: Mocha and Chai testing libraries
5. Other tools:
    - Task management: Trello

Deployment Environment

The deployment environment would include all the above hardware, software and database. In addition, the REST API and the web application will be deployed on Heroku. Heroku allows continuous deployment from a GitHub repository, so the code base that works in local development environment should work when deployed on Heroku as well.

Architecture

The project consists of 4 main components:

- The aquaponics setup (all hardware, sensors and ESP32 microcontroller)
- The database

- RESTful API
- The web application

Their relationships are described in the following diagram:

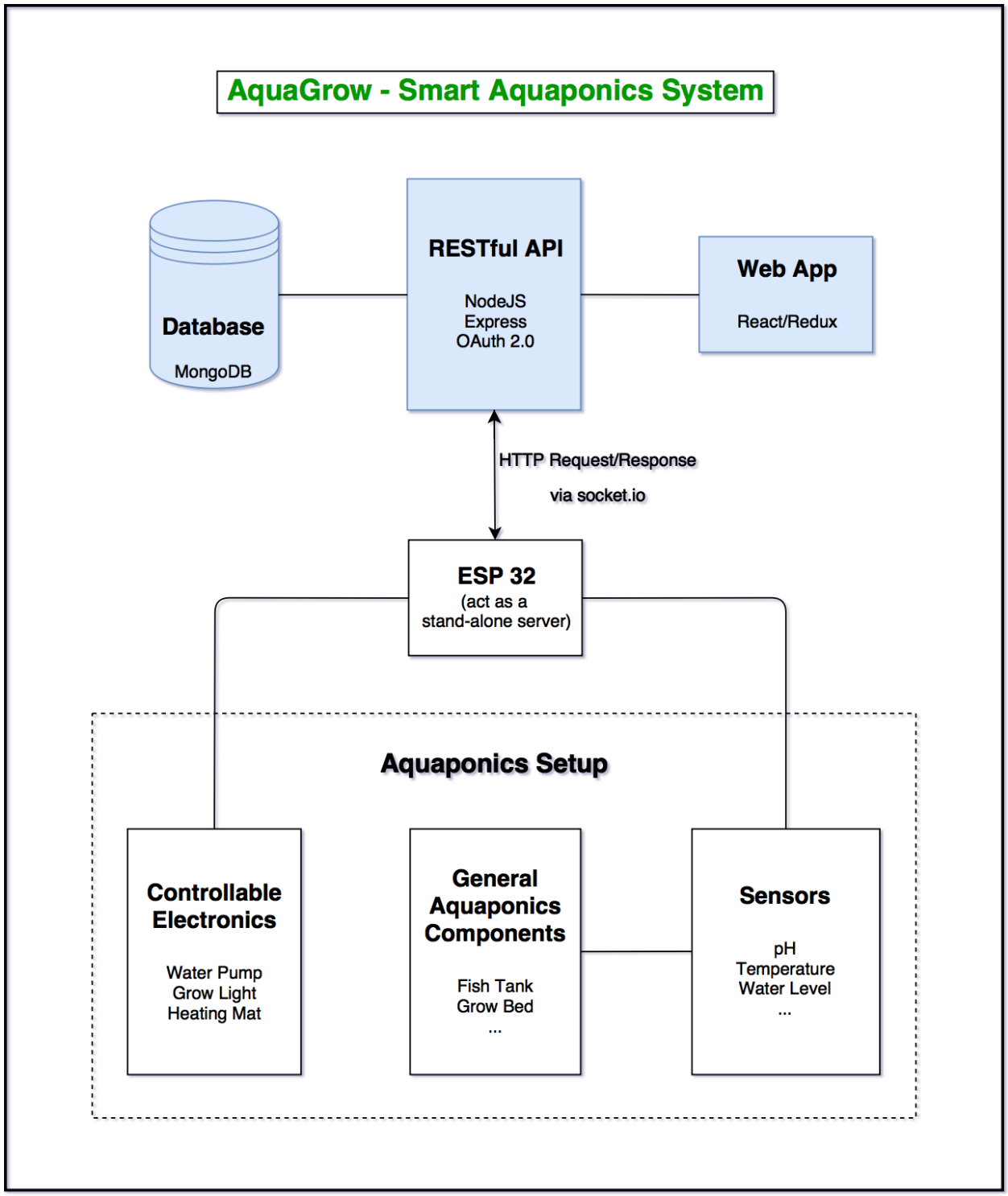

Diagram explanation:

- ESP32 directly controls the water pump, grow light and heating mat. It also directly obtains data from the sensors.
- ESP32 constantly sends HTTP POST requests containing sensor data to the API. By using socket.io, the API will then keep pushing new data to the web interface, while the web interface would process the incoming data and act accordingly (if the accepted data is out of set range, the app should send out notifications to the users)
- The web application in turn sends HTTP POST requests to ESP32 containing data regarding user's settings, such as how often the user wants to receive the sensor data or whether the user wants to turn anything on/off.
- Similarly, the API provide endpoints for the web application to send HTTP requests and communicate with the database.

Database Design:

- The database will include information regarding USDA Plant Hardiness Zones, plant presets (suggested temperature, water level, timeline, etc. for a particular plant), users' information, users' gardening portfolio, and potentially a large amount of sensor's data for the creation of graphs that help monitor real-time data and show trends/consistencies of the system.
- Since MongoDB (NoSQL database in general) is very flexible in changing the schemas of the databases, its design has not been finalized. The schemas are designed around developers' use cases.

Implementation Strategies

The implementation of the web application will focus on the use of MVC model. By using React as the front-end framework, many components will be reusable and Redux will help with managing the states of each component.

The application will also focus on having modular code base, which means that each functionality or component will be contained inside a JavaScript module.

User Interfaces

User interface will mainly consist of a dashboard. The dashboard will consist of all the real-time data from the sensors, list of the plant(s) that are being grown, and graphs that show data trend over time. Besides that, basics functionality such as signup, login, logout, settings will be provided. Detailed user interface design will be done with Sketch/Adobe XD.

Test and Integration Plan

The sensors will be tested individually first before being continuously added to the ESP32. The implementation of the hardware will be helped by Sandip Gautam, a student from Department of Electrical Engineering. While Sandip develops on the sensors + ESP32 system, the database, API and web application will be developed. Database and API endpoints will be tested by Postman. Application's responsive web interface will be tested by Sizzy.

The whole application will be tested through the use of Travis CI, a continuous integration service used to test software projects hosted through GitHub. Every time a new commit of the source code is pushed to GitHub, Travis will automatically build the project, run the tests and deploy the application.

Project Timeline

- Sun, 3/11: Finish creating main endpoints for the API. Implementation API security by authentication and authorization with OAuth protocol
- Tue, 3/13: Finalize database design, import dataset and create dummy data for API testing
- Thurs, 3/15: Finish user authentication and authorization feature
- Wed, 3/21: Finalize on UI design, start on implementing the UI (without fetching real data)
- Wed, 3/28: Finish API implementation and API testing
- Wed, 4/4: Finish draft UI, start fetching with real data from using API endpoints
- Wed, 4/11: Integrate with Travis CI for continuous integration, start deploying to Heroku
- Sun, 4/15: Start finalizing on MVP features and the application's styling
- Wed, 4/22: Continue with touching up with extra features and styling
- Sat, 4/28: Start on documentation

Bibliography

- NodeJS official documentation: https://nodejs.org/en/
- Express guide and API reference: https://expressjs.com
- Facebook's official ReactJS Docs: https://reactjs.org/docs/hello-world.html
- Redux's Documentation: https://redux.js.org
- MongoDB architecture: https://www.mongodb.com/mongodb-architecture
- MongoDB schema design: https://www.mongodb.com/blog/post/6-rules-of-thumb-for-mongodb-schema-design-part-1
- MongoDB data model examples and patterns: https://docs.mongodb.com/manual/applications/data-models/

- Travis CI documentation: https://docs.travis-ci.com
- Ant Design: https://github.com/websemantics/awesome-ant-design
- ESP32 web server: https://randomnerdtutorials.com/esp32-web-server-arduino-ide/

Minimum Viable Product

The minimum viable product will include the following:

- A well-designed MongoDB database
- A RESTful API that provides all necessary endpoints for the web interface to communicate with the database and the ESP32 microcontrollers. The API would allow developers to make any application on any platform (web, cross-platform mobile app, desktop app…). The API needs to be secured by authentication and authorization of users by OAuth protocol.
- A well-designed, mobile-responsive and dynamic web application that fulfills all the sponsor's expected features. The web interface should have 3 out of the main 4 features:
  - Monitoring: of sensors' data and system status
  - Notifications: of system failures and reminders
  - Organizing: of users' personal data and garden portfolio

  The feature that is not included in the MVP is the controlling of the water pump, grow light and heating mat. This task is highly dependent on the hardware design and implementation, which is outside of the software scope.