Aqua Growth
SOLUTIONS

System Architecture Proposal and Software Modeling

Team Members
Noah Jacinto
Alexander Vega
Megan Kang
Daniel Vasquez
Jeet Patel
Jaxon Brown

System Architecture Proposal

After comparing different software architectures with our team, AquaGrowth proposes a dual architecture approach including Model View Controller (MVC) and Generic Attribute Profile (GATT). The Model View Controller (MVC) is our main design pattern for the mobile application. MVC is meant to divide the program logic into three elements. The Model will manage the data and establish the logic, View represents the interface and presents the information from the model, and the Controller links the model and view by handling the input data and flow. By dividing the program logic into three elements, it allows benefits for modularity and scalability within our mobile application. Aside from the model view controller, we will pair this architecture with Generic Attribute Profile (GATT) to connect the physical device and our mobile application. The GATT architecture primarily defines how bluetooth devices transfer data between two connected devices. Data is passed as characteristics and stored in the bluetooth energy devices. At a high level overview, this architecture contains two roles: the GATT server and GATT client which reads and writes data back and forth to each other, after establishing a connection.

By integrating both MVC and GATT in our project, we can benefit from use cases in these architectures since our project depends on a hardware device and software application. The MVC will manage the data between the user interface and logic, providing a sense of modularity and division of labor inside the application. At the same time, the GATT architecture will handle the communication aspect between the physical device and mobile application, allowing for a clear two way communication to send and receive data. This protocol will allow for various sensors to send data to the mobile application. By combining MVC and GATT it allows for a clear way to structure the mobile application and provides a standardized communication between two devices.

System Architecture Justifications and Advantages

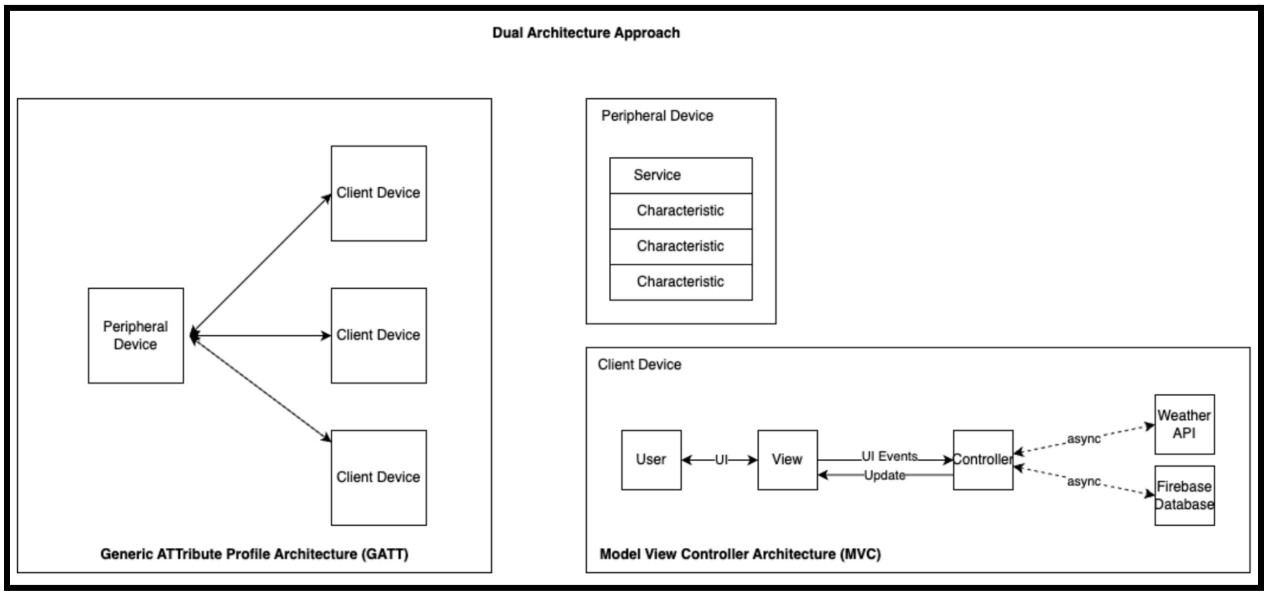
Model View Controller (MVC) allows our team to separate the entire mobile application into 3 major components (Model, View, Controller) allowing for a separation between the logic, user interface, and data. Independent development is a key process to productivity and would promote a collaborative environment because our team can split into multiple groups working on different aspects of the mobile application. Lastly, aside from separation for the mobile application our team will be able to maintain and reuse certain classes, models, and views for future use.

A few advantages of Model View Controller Architecture are Modularity and Testability. Focusing on modularity is important because our team will be working on multiple components at once, therefore capitalizing on modularity will provide independent development and scalability for each component. Aside from development, MVC allows for benefits in testability. Unit testing is a major aspect in maintaining different components and ensures each function is performing properly.

Generic Attribute Profile (GATT) will handle the bluetooth connection layer between the peripheral and client devices. Since our project will be a smart plant device that gathers plant information through sensors, the GATT architecture is a practical approach. A few justification reasons are a simple connection and structure to exchange data. The architecture offers services and characteristics which structures the exchange of data between bluetooth low energy devices. This would benefit our case of multiple plant sensors (services) and their sensor values (characteristics).

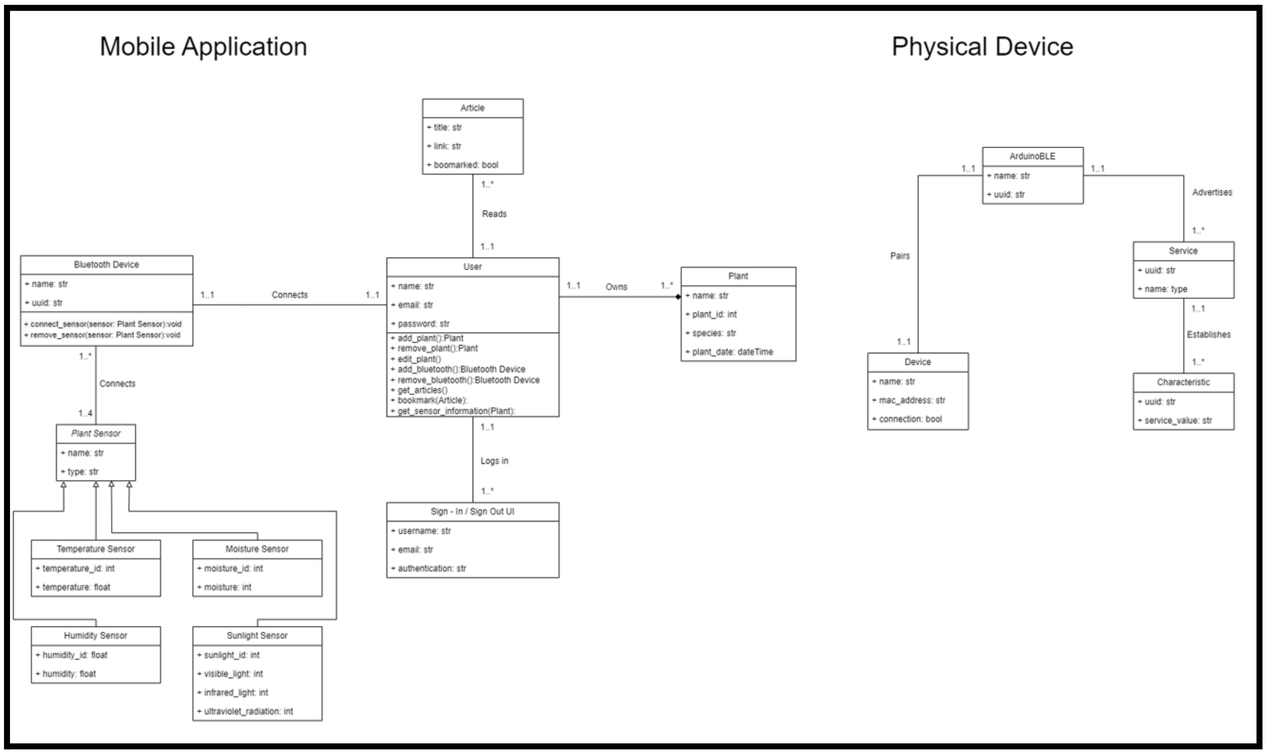
A few advantages of GATT are compatibility and resource optimization. A structured communication channel between devices ensures the connection between a peripheral and client device. This would provide a quick and seamless connection for our targeted audience. Additionally, the GATT architecture prioritizes bluetooth low energy devices, therefore it would efficiently use resources when transferring data back and forth devices.

Software Modeling



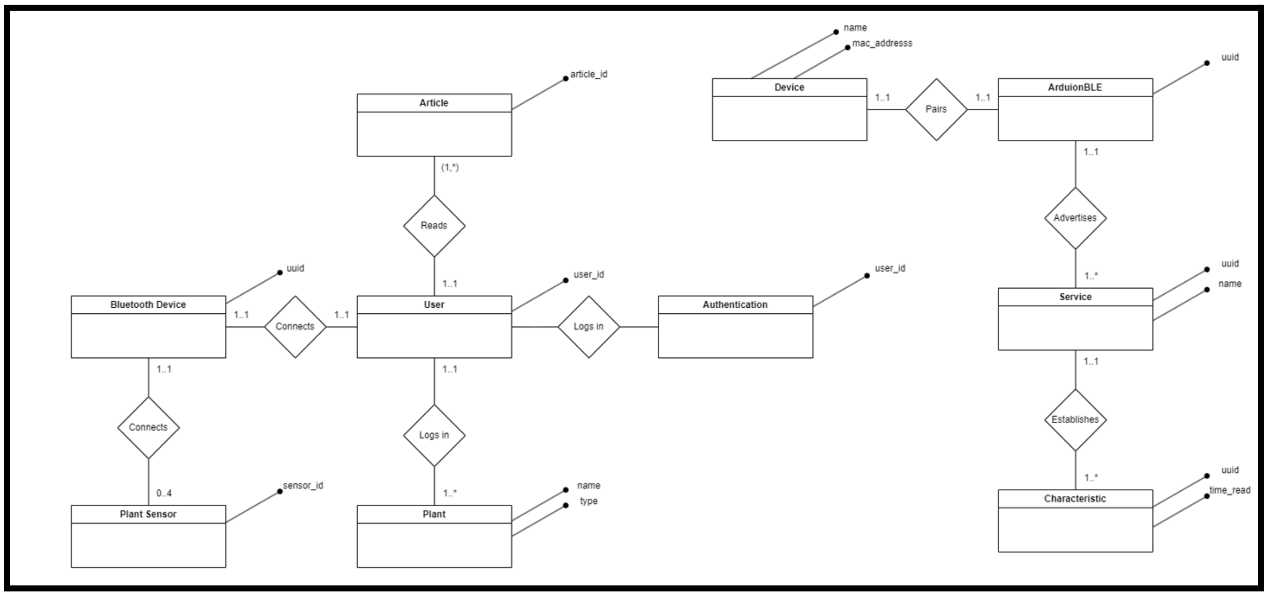
This is a high level overview of our dual architecture approach, that consists of a Model View Controller and Generic Attribute Profile Architecture. Our Generic Attribute Profile Architecture contains a peripheral and client Device, in our given context the peripheral represents the physical arduino device and each client is the iPhone. Aside from the GATT architecture, the mobile application will follow the Model View Controller to separate the logic, user interface, and data behind the scenes. This will allow us to listen to user interface events, updates, and asynchronously send requests to the weather api and firebase database.

UML Diagram



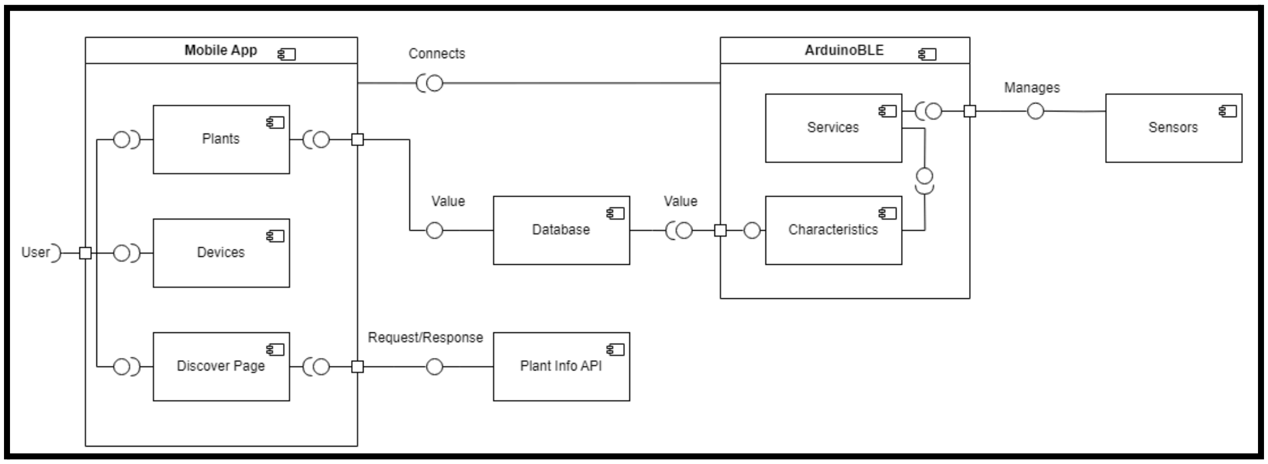
For our UML diagram, this defines our mobile application and hardware device. Focusing on the mobile application UML diagram, we separated the design by user, article, plant, and bluetooth. A user will own a plant, log in, connect to a bluetooth device, and read an article. Details of the attributes and operations are specified in the image above along with the relationships. Aside from the connections regarding the user, the Plant sensor is a general class of the sensors. A few instantiations of these general classes are Temperature Sensor, Moisture Sensor, Humidity Sensor, and Sunlight Sensor. For the physical device, we also need a UML diagram because we are coding the drivers for a client connection, service, and characteristic. The arduino will establish a connection to the device, advertise services, and the services will establish characteristics. This UML diagram defines the relationship between a physical device and mobile application for sending value information through sensors.

Entity Relationship Diagram



The entity relationship diagram is a clone of our uml diagram but converted. Following the same format, it still follows the previous diagram but defines the relationships more clearly.

Component Diagram



A component diagram illustrates the high level structure of a system and how various components interact with each other. Our system has two main subsystems, a mobile application and a hardware component (Arduino BLE). Within the mobile app subsystem, the mobile app will contain three components: plants, devices, and a discovery page. All these components require a user to interact with their functionality. In terms of the discovery page, this component requests data from an external component, Plant Info API. The hardware subsystem (the arduino) consists of two components, services and characteristics. It is important to note that the Arduino requires and manages sensors. These sensors provide a service which in turn provides a characteristic (value/data from the sensor) which is then stored in a database and sent to the mobile application.