Aquaponics: Physiochemical Properties monitored

There are numerous examples of projects that have a similar theme and focus. Of those that were published, all of them made use of sensors connected to microcontrollers. Some of them used Arduino, and either connected LCD screens and lights to them for output or connected them serially to computers via USB. Another example showcased a project where the students connected an Arduino and Raspberry Pi together and used the Pi to display output to a remote computer (screen).

Most of the examples, however, exclusively used a Raspberry Pi. The Pi can run Linux and offers more flexibility to write programs that are geared towards more than just the components connected on the circuit board.

The projects that used a Raspberry Pi displayed output in one of the following ways:

1. A client (computer or mobile phone) connected to the same local network as the Pi, navigates to the Pi's local IP address in their browser and views the readings from a website interface.
    - *Benefits:* Easy to view live (real-time) readings; Python can serve sensor readings directly to HTML & JavaScript
    - Drawbacks: Readings and historical data can only be viewed when connected to the same network as the Pi (i.e. user cannot view the readings remotely)
2. The Pi periodically sends readings to a remote server. The server then stores it in a database and a scripting language serves it whenever a client opens the website.
    - *Benefits:* Readings are available regardless of location
    - *Drawbacks:* Live sensor readings are not available as the IO costs are too high

In an effort to offer the benefits of both solutions, I propose the following:

We use a Raspberry Pi 3b+ with Temperature, Water Level and pH sensors. The sensors that measure NO2, NO3 and DO are very expensive – most of them start at $250 per sensor. My reasoning for picking the Pi instead of Arduino is because it will be easier to run code directly on it and communicate with a remote server without the need of a dedicated computer or extra modules. Additionally, I already own a Raspberry Pi 3b+; so its use will result in fewer costs. A Python script periodically broadcasts the sensor readings every 2/3 seconds to the MQTT broker and permanently stores the data on a remote database by calling a server-side script every 10/15 minutes.

A shared hosting service is used as a "bridge" between the sensors and the UI that displays the readings. A PHP script is called on the Pi (by Python) and includes the sensor readings in the URL parameters. The script stores the readings in the database. When a client browses to the domain, JavaScript will retrieve the data from the database using PHP. Additionally, a Websockets connection is opened to a MQTT broker (explained later in the proposal) and provides the live readings. For a hosting service, I propose shared hosting instead of a Virtual Private Server as it is cheaper and far easier to use (shared hosting is usually always managed while managed VPS is extremely expensive). GoDaddy is, in my opinion, the best option for a service due to my experience with them and their competitive pricing. They currently ask R540 for 12 months (R180 for 3 months) on a Linux server

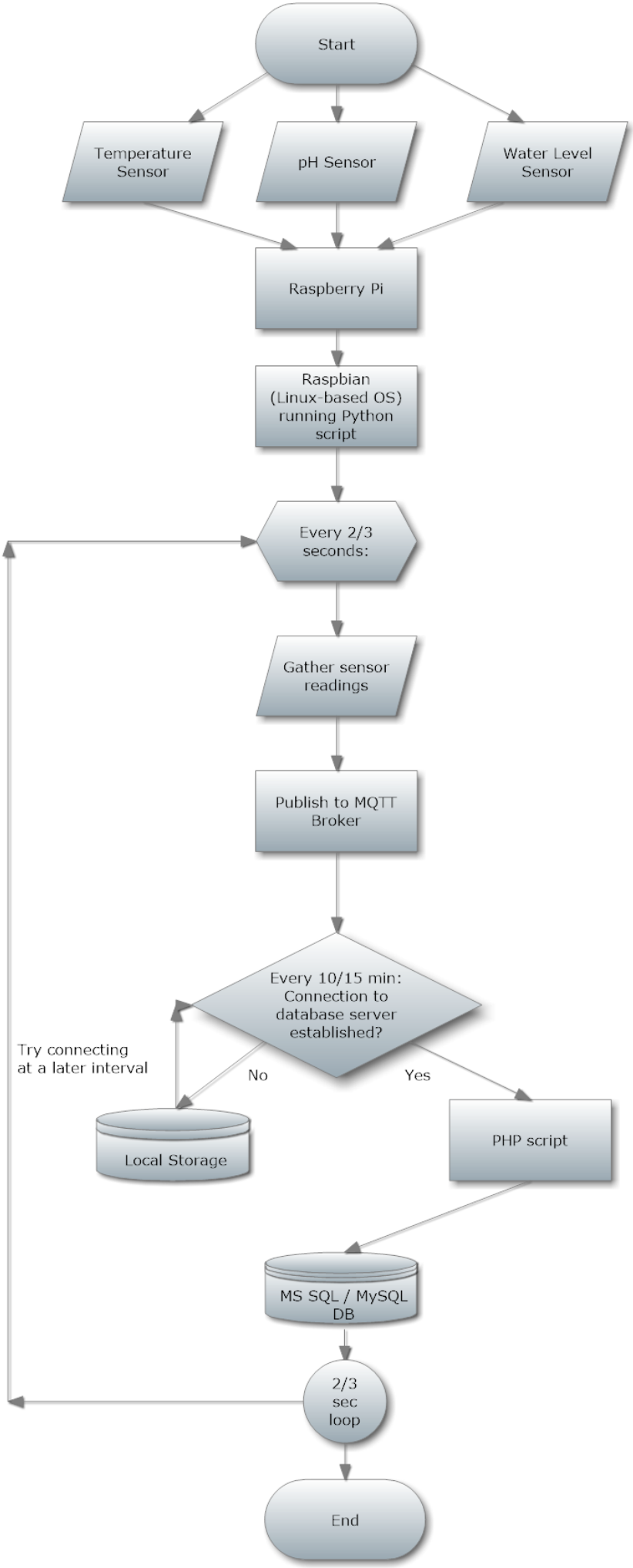
and R387 for 3 months on a Microsoft server. We will discuss which type to use based on the database we will configure.

To include real-time readings, I propose we use MQ Telemetry Report (MQTT). It is a protocol specifically designed for IoT that simplifies the transfer of messages between devices. A device can broadcast messages fixed to topics and/or subscribe to them. Broadcasters and subscribers connect to a MQTT broker to allow them to communicate in a manner similar to watching TV or listening to the radio. Note that they can both be a broadcaster and subscriber. Messages can be broadcasted indefinitely; but they are not stored on the broker's server (hence the need for a hosted database). For an MQTT Broker, I propose using CloudMQTT. They offer a $5/month plan that should be sufficed for our needs. Additionally, their service supports Websockets; which means JavaScript and Python can broadcast and subscribe directly with and to the broker in the most efficient way possible.

To summarise the solution:

- We connect Temperature, pH and water level sensors to a Raspberry Pi 3b+. Raspbian is then installed on the Pi and configured with the necessary libraries.
- A Python script runs on the Pi and publishes MQTT messages to a MQTT broker. Every 10/15 minutes, Python triggers a PHP script and includes the sensor readings in the parameters.
- A shared-hosting service provides us with a server on which we can run the PHP scripts, host a database and serve the web pages.
- When a user navigates to the specified domain, a website is displayed to them that:
    - Shows historical data in intuitive graphs
    - Opens a Websockets connection to the MQTT broker and displays live sensor readings

A flowchart summarising the process from the Pi's end:


Start
Temperature Sensor
pH Sensor
Water Level Sensor
Raspberry Pi
Raspbian (Linux-based OS) running Python script
Every 2/3 seconds:
Gather sensor readings
Publish to MQTT Broker
Every 10/15 min: Connection to database server established?
Try connecting at a later interval
No
Yes
Local Storage
PHP script
MS SQL / MySQL DB
2/3 sec loop
End

A flowchart representing what happens on the remote server when a client requests the website: