

ELE3305

Design Report

Student Name: Kate Bowater

Student Number: U1019160

Due Date: 15 April 2024

Executive Summary

This report entails the design of an automated control system that utilizes Internet of Things (IoT) for use in aquaponics or aquariums. Its purpose is to help the user monitor the environment of the tanks and alleviate stress as well as save time on upkeep and maintenance.

The automated system measures three sensor values: water temperature, dissolved oxygen (DO) and pH levels. The ideal range for these values is as follows:

- DO:
 - 80-100%
- pH:
 - 6.5-8
- Water temperature:
 - 18-30°C

Values outside of these ranges will require corrective action to maintain a healthy environment for the fish and plants in the aquaponics system. Information from these sensors is sent via MQTT to The Things Network (TTN) and forwarded to subscribed nodes. The nodes include OpenPLC, Node-Red, and a web-based human machine interface (HMI).

OpenPLC is used to automatically control the maintenance components of the aquaponics system. For low DO, the system will turn on an air pump to add more oxygen to the water and turn off when the DO is high enough. For water temperature, a heater will turn on in low temperatures while a water pump will turn on for high temperatures which increases water flow to cool down the tank. The pH however is more complex; changes in water temperature and DO can directly affect this aspect, therefore the system will require user intervention to determine the problem.

Node-Red is software that is used to create flows to process information and apply actions. It will receive information from the sensors via the MQTT, and functions written in Javascript are applied to decode and scale the data to obtain the actual values of pH, temperature and DO. Once the actual values are obtained, they can then be used in the flows. The first flow is to apply conditions which will change the inputs in the OpenPLC circuit based on the ideal ranges specified above and thus the activity of the outputs to maintain stability of the tanks' environment. The next flow is to push the data to the dashboard on the HMI for the user to see. The final flow is to upload the sensor information to the mapper API.

The HMI is part of the Node-Red software. This will be set up on a device, such as a tablet, in kiosk mode which will allow the user to view the sensor information but not make changes to the software or device settings. The HMI will also give the user the option to activate the air pump and water pump if they require.

Regarding protocols, Modbus TCP/IP is used to transmit the information between Node-Red and OpenPLC. It is a lightweight protocol and useful for small data packets being sent over IoT such as in this system. The nodes are to be connected to a local area network (LAN) and the network configuration for all the nodes is outlined in the table below:

Node	IP address / Subnet
OpenPLC	192.168.0.2 /24
Node-Red	192.168.0.3 /24
HMI	192.168.0.4 /24

The ICT risks associated with this setup are primarily concerned with unauthorized access to the system, theft of information transmitted over the network, and tampering of data and system settings. By not having appropriate authentication, authorization and encryption methods, vulnerabilities in the aquaponics system can be exploited which can lead to failure in the upkeep of the tanks, potentially leading to environmental failure and fish or plant death. Mitigation strategies include use of TLS certificates, API keys, and AES keys. Using strong passwords for login credentials is vital for protection, and password hashes are useful for preventing use of stolen login information. Keeping the system up to date with patches is another mitigation strategy that will reduce vulnerabilities and software bugs.

Lastly, after conceptualisation and testing of the system, several aspects for improvement have been identified. When uploading information to the mapper API, it was worth noting that the flow does not check for updated or valid values. If no number is sent, then it will update with NaN. This can be addressed using a conditional node in the flow to upload old values alongside new values if they are sent at separate intervals. The functionality of this system for an aquaponics setup is also very basic and does not entail all aspects required for a complete setup, such as water flow rate, tank levels, and light levels. However, it is believed that this system can be easily adapted to include more sensors and scaled up for a greater number of tanks.

Table of Contents

Executive Summary.....	2
Application Scenario.....	5
Specifications.....	5
Sensors.....	5
Control	5
Power Supply	6
System Overview	6
Network	6
Nodes	7
Data Decoding and Scaling.....	8
Dissolved Oxygen	9
pH	9
Water Temperature	10
ICT Security Risks.....	10
Authentication	10
Authorization.....	11
Encryption	11
Patching and Other Recommendations	11
Conceptualisation	Error! Bookmark not defined.
Network Configuration.....	12
OpenPLC	12
Node-Red	14
Improvements.....	19
References	19

Application Scenario

Aquaponics is defined as a closed system where fish and plants survive and grow together by providing each other with nutrients. In this system, rather than filtering out waste from the tanks, the waste from fish is consumed by bacteria and converted into nitrates which plants use for nourishment [1] [2]. The thriving plants then release oxygen into the water which circulates back to the fish. In this system, the delicate balance of water pH, temperature and oxygen levels are vital for a happy ecosystem [3], and so requires constant monitoring to ensure fish and plants are obtaining adequate nutrients and not stressed out.

Implementing automated monitoring and maintenance to the aquaponics system using Internet of Things (IoT) will take the strain off the owner of the aquaponics system to ensure all these elements are measured and addressed in a timely manner, reducing the risk of plant and fish illness or death caused by lack of supervision [1] [4].

Specifications

Sensors

The details of the sensors required to monitor the aquaponics system include the following:

Table 1: Sensor information

What is Measured	Ideal Range	Units
Dissolved Oxygen (DO)	5-8 80-100	mg/L %
pH	6.5-8	-
Water Temperature	18-30	°C

It should be noted that the ideal range for DO in water is 5-8 mg/L [1] [2], but this can change depending on water temperature and salinity. Hence an alternative measurement is through percentage with an ideal range of 80-100%. In this case, less than 80% will risk the health of the fish and plants, while greater than 100% is likely to create algae blooms [12].

Control

To control the aforementioned factors of the tanks, a PLC program can provide automation of required equipment. For dissolved oxygen, an air pump will add more O₂ to the water and turn off when the required level is reached.

The water temperature is controlled using a heater to increase the temperature, and when the water temperature is too high then a water pump will activate to increase flow through the tank to cause temperature to decrease.

The pH level is a complex factor closely tied to oxygen and temperature levels. It is assumed in a well-balanced system that the pH level will be self-maintained and should not fall outside the ideal range, but if it does it requires direct intervention from the user due to complex associations to other variables, so only an alert will be sent to the user.

These actions are all outlined in Table 2 below. If any values outside of the ideal range mentioned in Table 1 are detected, the PLC will activate to maintain balance of the aquaponics system. It should be noted that the provided actions are not exclusive (as temperature, DO and pH can directly affect each other), but are the only ones provided to maintain simplicity for the conceptual design of the system.

Table 2: Control of sensor readings

Sensor	Problem	Required Action
DO	< 80%	Turn on air pump
	> 100%	Turn off air pump
pH	< 6.5	Notify user
	> 8	Notify user
Water Temp	< 18 °C	Turn on heating system
	> 30 °C	Turn on water pump

Power Supply

For the system to run over Internet of Things (IoT) the software can be implemented on a small device such as Raspberry Pi. This can be connected to a small power supply either via mains or utilise solar. It could also incorporate battery back up to ensure upkeep of the tanks in the event of power loss. The simplicity of this aquaponics system can then be scaled up to meet larger demands including aquariums and farming [2].

System Overview

Network

The information obtained from the sensors in Table 1 are sent via LoRaWAN (long range wide area network) to The Things Network (TTN). As LoRaWAN networks use ALOHA based protocols to avoid collision, the sensors don't need specific gateways and messages are sent through all gateways within range. In this way, the sensor information can be sent to a message broker via the network server [13].

The message broker, in this case Mosquitto, uses MQTT protocol to take these messages as subscriptions. MQTT is appropriate for this situation because of the simple, small amounts of data being sent from the tank's sensors which do not require large headers or bandwidth. HTTP is another protocol that could have been used for this scenario, but MQTT is low cost, reliable in unstable connections, and provides scalability unlike HTTP which requires more resources to run [6]. The request-response model of HTTP is also a disadvantage compared to MQTT which can send messages without direct authentication to the nodes.

Using TTN, the message broker then sends the message through a local network to the subscribed nodes: Node-Red, OpenPLC, and a web-based human machine interface (HMI). Figure 1 shows the intended network and connected devices.

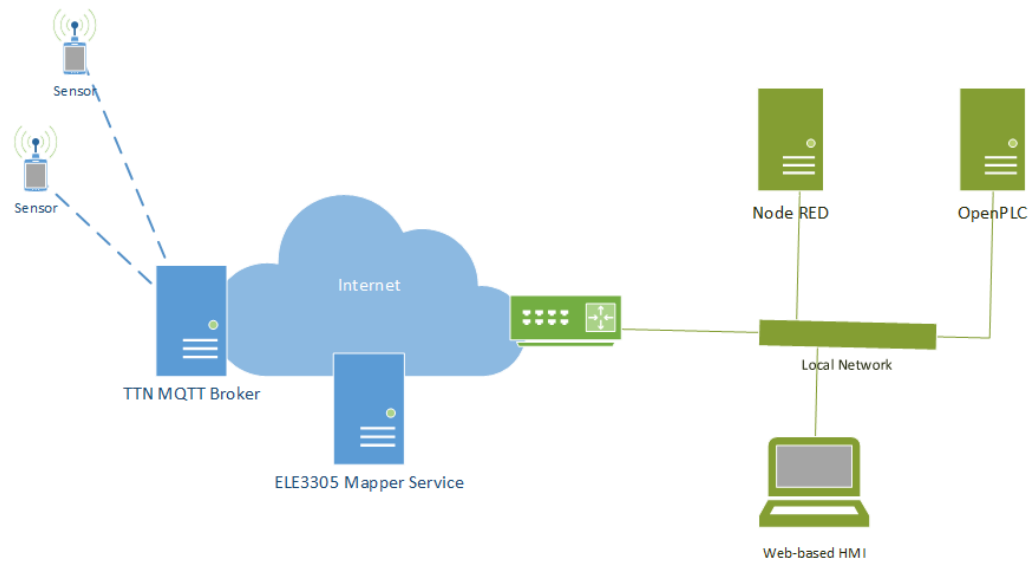


Figure 1: Network layout for aquaponics system

The protocol to connect the sensors and devices to OpenPLC is Modbus TCP/IP. Using TCP/IP allows the data to be carried over ethernet as it is already a widely used protocol and provides compatibility with preexisting ethernet infrastructure [10]. As the data being transferred is only small, Modbus is also suitable for this application due to its limited packet transfer size of 260 bytes [11].

Nodes

As seen in Figure 1, the nodes provided are Node-Red, OpenPLC and a web-based HMI. Node-Red is used to implement the IoT network. This creates flows for the decoding and scaling of the received information before forwarding to OpenPLC and the HMI. Node-Red has a built in feature to create a HMI in the form of a dashboard for the user to interact with where they can view alerts and controls of the system.

The human-machine interface (HMI) will be implemented on a device set up in kiosk mode. This will only allow access to Node-Red's user interface (UI) and will display information on the pH, DO and water temperature of the tanks and provide the controls to the user. This can be implemented on a tablet device for convenience so the user can easily monitor the aquaponics system.

Use of Node-Red is appropriate as it is easy to set up with visual nodes for flow creation. This means other technicians can easily access and interpret the layout to make changes for the user if needed. Connection to other nodes on the IoT network will be easy to manage using the flows, and the built-in UI dashboard is a necessary feature for the user and is conveniently provided in Node-Red's features.

OpenPLC can be used to implement the circuit that will provide automated control to the aquaponics system. It should follow the requirements of the system as laid out in Table 2 under specifications. Once a circuit is uploaded to this software, the connection to Node-Red via Modbus will allow the Node-Red flows to control the system.

The actions in Table 2 can be implemented with the pseudocode below. For DO and water temperature, the pumps will activate automatically but the design will also allow the user to turn on or off the air pump and water pump as needed. The water heating does not have the same functionality because if the heater is left on and reaches the high temperature threshold then it is simply wasting power and risking the health of the fish and plants.

Pseudocode for aquaponics system:

```
If dissolved_oxygen is low OR airpump_button_on:
  - Turn on air pump
Else If dissolved_oxygen is high OR airpump_button_off:
  - Turn off air pump

If pH_level is high OR low:
  - Turn on alert

If water_temp is low:
  - Turn on heater
Else turn off heater

If water_temp is high OR waterpump_button_on:
  - Turn on water pump
Else if water_temp is not high OR water_temp is low OR waterpump_button_off:
  - Turn off water pump
```

Data Decoding and Scaling

The information sent by the sensors must be decoded and scaled before being used in the system. Each data packet contains 14 bytes. The raw information contained in each packet is encoded in base 64 and can be decoded in Node-Red using a code snippet written in Javascript as follows:

```
// Separating data in base64 to individual bytes (hexadecimal)
var hexbytes = Buffer.from(msg.payload.uplink_message.frm_payload, 'base64');
msg.payload = hexbytes; //overwriting the payload with the hex values
return msg;
```

This takes all the information from the data packet and converts it to hexadecimal values that correspond to the bytes. After this, the values can be isolated and scaled for the requirements of the system.

Dissolved Oxygen

The value for DO is contained in bytes 0 and 1. It is understood that the range in these bytes is 0-5000. While the ideal range for DO is from 80-100%, a maximum range to 120% will be sufficient for determining excessively high values. It is then a simple matter of scaling through the following equation:

$$DO = \frac{value}{\frac{5000}{120}} = value * \frac{120}{5000}$$

The following code obtains the transmitted value and scales it to the required 0-120% which can then be used in the system.

```
const bytes = msg.payload; // holding all the bytes of the payload
let DO_data = ((bytes[0]<<8)+bytes[1]);
// the byte needs to be shifted because low byte of transmission is the highest
byte of value
// the range is 0-5000
let DO_value = DO_data / (5000/120); //scaling to the DO range 0-120%
msg.payload = DO_value;
return msg;
```

pH

The pH value is obtained from bytes 6 and 7. It is understood that this value has a range of 0-100 and was multiplied by 10 for transmission, so first it is divided by 10 before scaling into the pH range 0-14 using the following equation:

$$pH = \frac{Data}{(\frac{100}{14})} = Data * \frac{14}{100}$$

The following code determines the transmitted value and scales it to the pH level using the equation above.

```
const bytes = msg.payload; // holding the bytes of the payload
const ph_data = ((bytes[6]<<8)+bytes[7]); //bytes 6 and 7 hold the pH value
let ph_value = ph_data /10; // the transmission was multiplied by 10, so need to
divide by 10 for true value
let ph_info = ph_value / (100/14); // scaling the data to the range 0-14 for pH
level
msg.payload = ph_info;
return msg;
```

Water Temperature

Water temperature is contained in bytes 8 and 9. It is noted that the value was multiplied by 100 for transmission. After dividing by 100, the temperature value is in Kelvin. This is then converted to Celsius by the following equation:

$$\text{Degrees Celcius} = \text{Kelvin} - 273.15$$

The following code was used to decode and evaluate the temperature from the bytes:

```
const bytes = msg.payload; // holding all bytes from the payload
const temperature = ((bytes[8]<<8)+bytes[9]); //bytes 8 and 9 hold the
temperature value
let temp_kelvin = temperature/100; // temperature is transmitted in kelvin*100.
So needs to be divided by 100
let temp_C = temp_kelvin - 273.15; // converting from kelvin to celcius
msg.payload = temp_C; // overwrite the msg to display temp in celcius
return msg;
```

ICT Security Risks

With regards to MQTT and the IoT network that the aquaponics system utilizes, security threats typically derive from the following three aspects: authentication, authorization, and package encryption [7]. Attackers can take advantage of weaknesses in these aspects to cause problems in the system, steal data, or perform DDOS attacks [6] [7] [8]. Modbus TCP is also vulnerable to DDOS attacks as it was originally designed for serial connections which had no security features and remains an unsecured networking protocol [9].

While the sensors of the aquaponics system do not transmit sensitive data (i.e. there is no information that can be used for identity theft or fraud), interference with the data packets can be used to obtain login information which would then allow an unauthorized user to tamper with the information from the nodes. Damage to the tanks' environment can occur as a result of changing inputs and outputs on the PLC, sending misleading data to the user, or blocking the system from performing its intended routines.

Authentication

Without appropriate authentication, information from the nodes and sensors can be intercepted and stolen when being sent across the network and through the internet. Recommended mitigation strategies include using secure socket layer (SSL) or transport layer security (TLS) certificates for authentication [7]. Documentation for TTN states that it already supports authentication using API keys, OAuth access tokens and session cookies [16]. The Node-Red editor on the other hand is not secured by default and requires configuration to incorporate security settings. It uses HTTP but can be changed to use HTTPS certificates within the settings file. It also supports API authentication via username and password credentialing for admin access only, as well as setting user permissions to restrict access to the flows [17].

Regarding OpenPLC, the threat involves access to the program to upload unwanted programs to tamper with the state of the controls. As with Node-Red, OpenPLC uses only HTTP and the login credentials are sent in plaintext which exhibits an easy exploit [18]. The software does not provide any inherent security features so mitigation strategies could include setting up TLS certificates, changing the default username and password to something more secure, and isolation of the software to reduce the risk of attack via other nodes.

Authorization

TTN also supports authorization using Rights which are assigned to the API keys used in authentication [16]. As mentioned above, OpenPLC requires log in credentials which must be changed from the default of 'openplc' for higher security.

Unauthorized access to physical devices, such as the devices containing the HMI or the nodes, can be mitigated by using strong passwords for logins, revoking privileges to the software, or in the case of the HMI for the user, setting up biometric authentication such as fingerprint or facial recognition, if the device supports it.

As the system is not anticipated to be used for web browsing at all, the risk of downloading malware is minimal, however the risk still exists should someone else gain access to the logins by packet sniffers or an injection attack to do so [7] [18]. An antivirus software is not recommended due to limitations in processing capabilities of the small devices used for the IoT network [6] [8]. In lieu of this, it is recommended to ensure that authorization for each node is sufficiently secure so that if a node is compromised then malicious operations cannot be easily transferred to other nodes [6].

Encryption

Encryption will help prevent stolen packets from being used or tampered with. The data packets sent via LoRaWAN are typically encrypted using advanced encryption standard, AES-128 symmetric keys [13]. Hashing passwords further protects authentication methods and makes them more difficult to decipher [7], and this is already included in Node-Red software [17]. Securing messages through MQTT via Mosquitto can be achieved by configuring SSL/TLS end-to-end AES encryption [19].

Patching and Other Recommendations

One final method of security is to ensure software is up to date and firmware patches are regularly applied. Out of date software can exhibit vulnerabilities that have been previously exploited, and applying updates can mitigate this risk as well as eliminating bugs that may cause issues in the system. For devices using IoT, connectivity may not always be assured so downloading the required updates can prove difficult and time consuming [8]. One suggestion is to provide an IT maintenance schedule for the user's system to check for issues and apply updates as necessary. Another suggestion is to have an additional device part of the network that will download the software updates separately, then push them to the devices once completed. Educating the user on cyber security and signs of potential interference can also help address the consequences of attacks.

Proof of Concept

Network Configuration

For the design of this system, the nodes need to have designated IP addresses to communicate to others on the local network. For simulation of the system using a virtual operating system, the loopback address of 127.0.0.1 can be used with different ports to access the different nodes.

In setting up the physical system for the aquaponics setting, the local area network (LAN) requires dedicated IP addresses and subnet. A subnet of 255.255.255.0 will allow up to 254 usable hosts which is sufficient for the needs of the aquaponics setup at present considering the available IP addresses are from 192.0.0.0 to 223.255.255.255 [14] [15]. The nodes could then have any IP address in this range, and the table below provides the designations for the LAN discussed:

Table 3: Designated node IP addresses

Node	IP address
OpenPLC	192.168.0.2
Node-Red	192.168.0.3
HMI	192.168.0.4

OpenPLC

Based on the pseudocode described earlier in this report, the circuit for automated control of the aquaponics system was created using ladder logic in the OpenPLC editor:

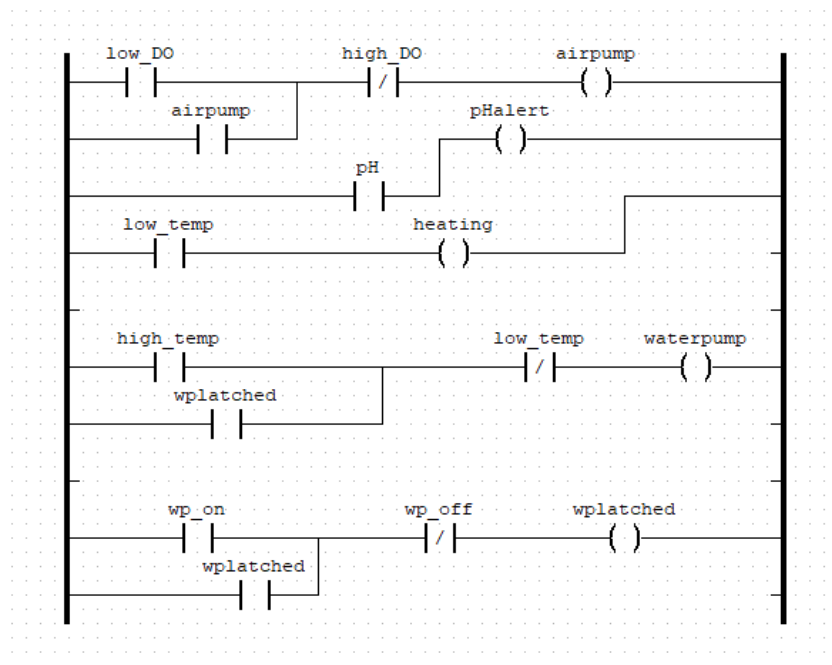
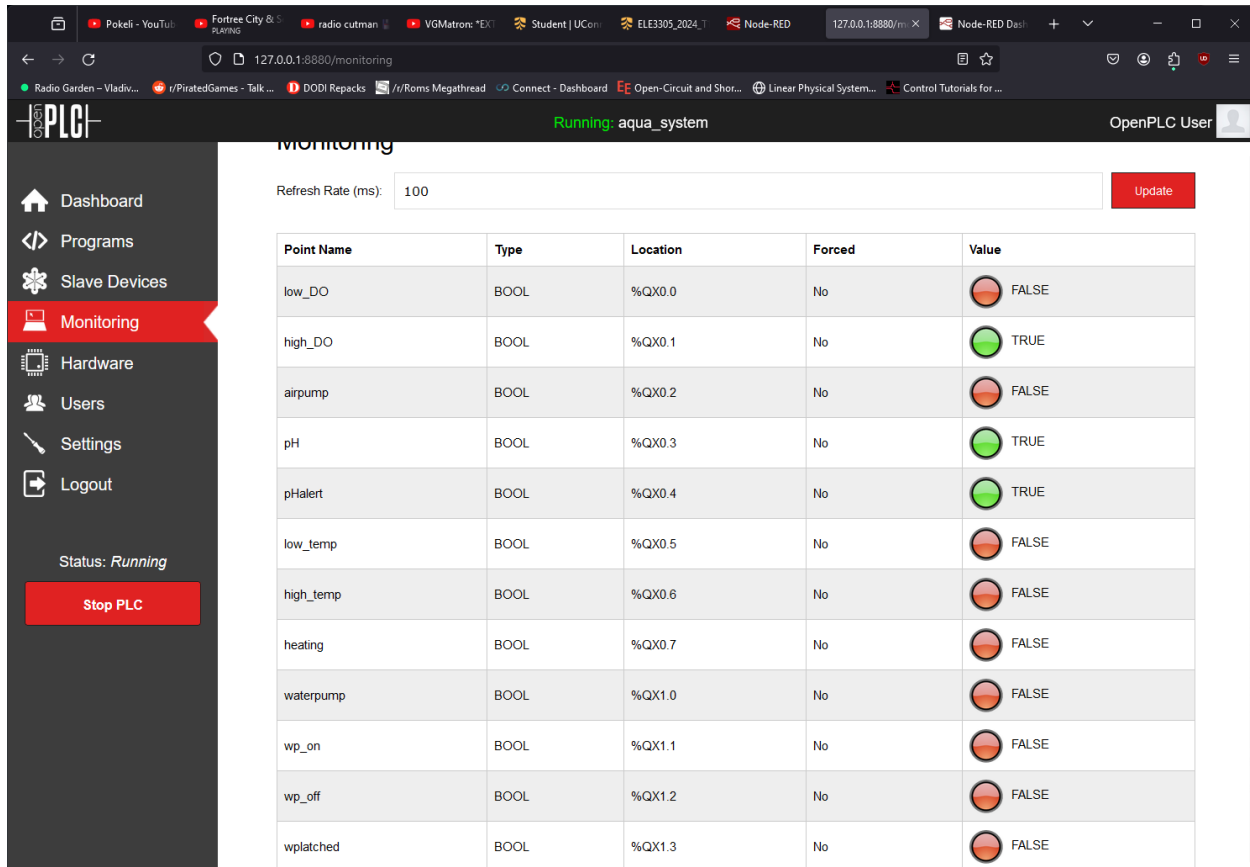


Figure 2: Screenshot of circuit in ladder logic

The inputs and outputs of this circuit are set up as BOOL values, i.e., they are only on or off states. The following categorizes all the variable names into inputs and outputs.

- Inputs
 - Low_DO
 - High_DO
 - pH
 - low_temp
 - high_temp
- Outputs
 - Airpump
 - pHalert
 - heating
 - waterpump

The variables wp_on, wp_off and wplatched are used as a latch in the circuit to ensure the water pump stays on if the user enables it via the HMI. The air pump has the same functionality, but this is achieved by changing the high and low DO values instead. The inputs are read from the sensors to control the output coils. Some of the outputs are also used as inputs to latch certain controls when a button is pressed. Figure 3 below shows the program uploaded and running in OpenPLC.



The screenshot shows the OpenPLC Monitoring interface. On the left is a sidebar with navigation options: Dashboard, Programs, Slave Devices, Monitoring (highlighted), Hardware, Users, Settings, and Logout. Below the sidebar, it shows 'Status: Running' and a 'Stop PLC' button. The main area displays the 'Monitoring' section for the 'aqua_system' program. It includes a 'Refresh Rate (ms): 100' input and an 'Update' button. Below this is a table with 5 columns: Point Name, Type, Location, Forced, and Value. The table lists 14 points, all of which are BOOL type and not forced. The 'Value' column shows the current state of each point, represented by a colored circle (green for TRUE, red for FALSE) and the text value.

Point Name	Type	Location	Forced	Value
low_DO	BOOL	%QX0.0	No	FALSE
high_DO	BOOL	%QX0.1	No	TRUE
airpump	BOOL	%QX0.2	No	FALSE
pH	BOOL	%QX0.3	No	TRUE
pHalert	BOOL	%QX0.4	No	TRUE
low_temp	BOOL	%QX0.5	No	FALSE
high_temp	BOOL	%QX0.6	No	FALSE
heating	BOOL	%QX0.7	No	FALSE
waterpump	BOOL	%QX1.0	No	FALSE
wp_on	BOOL	%QX1.1	No	FALSE
wp_off	BOOL	%QX1.2	No	FALSE
wplatched	BOOL	%QX1.3	No	FALSE

Figure 3: The status of inputs and outputs in OpenPLC

Node-Red

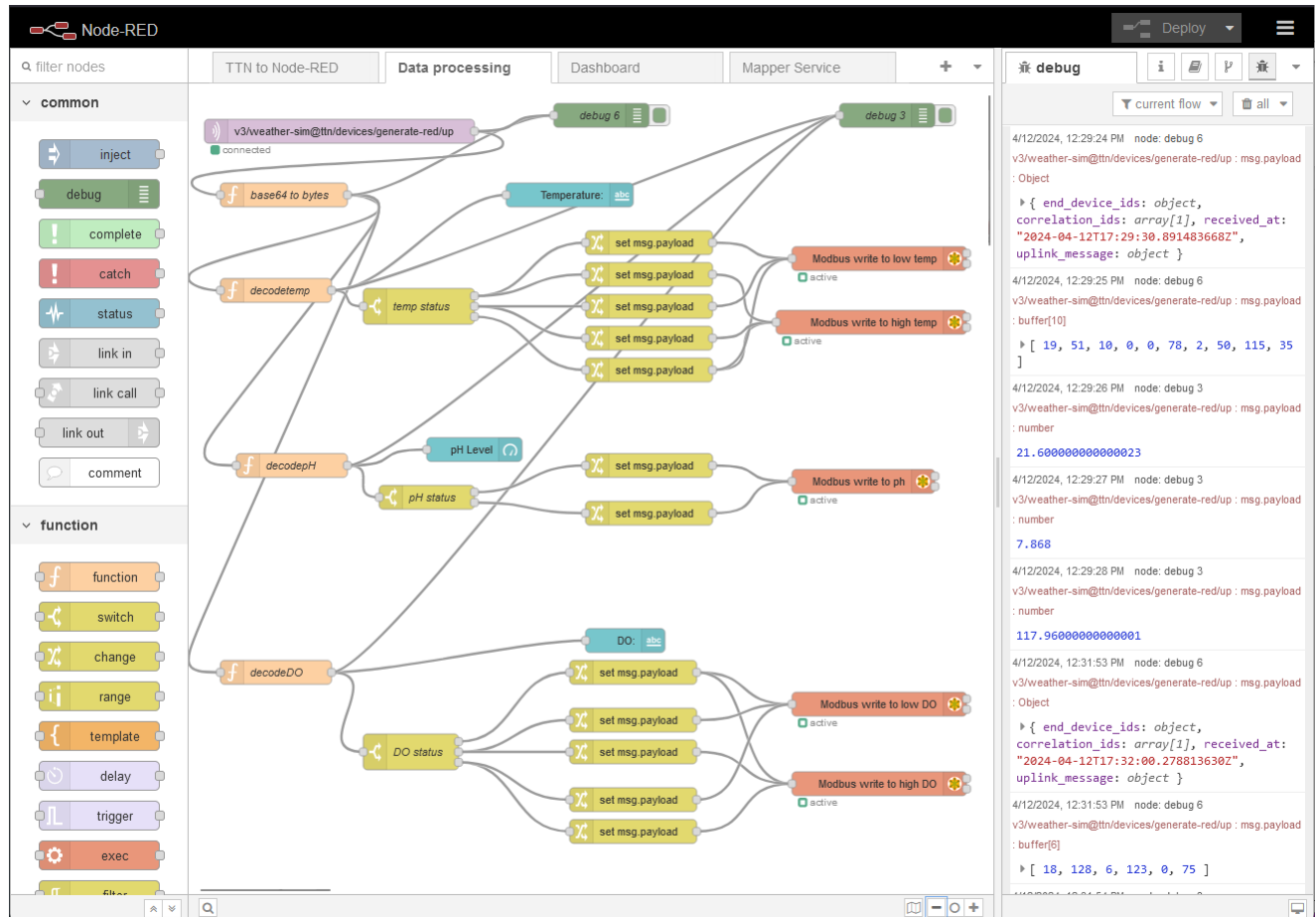
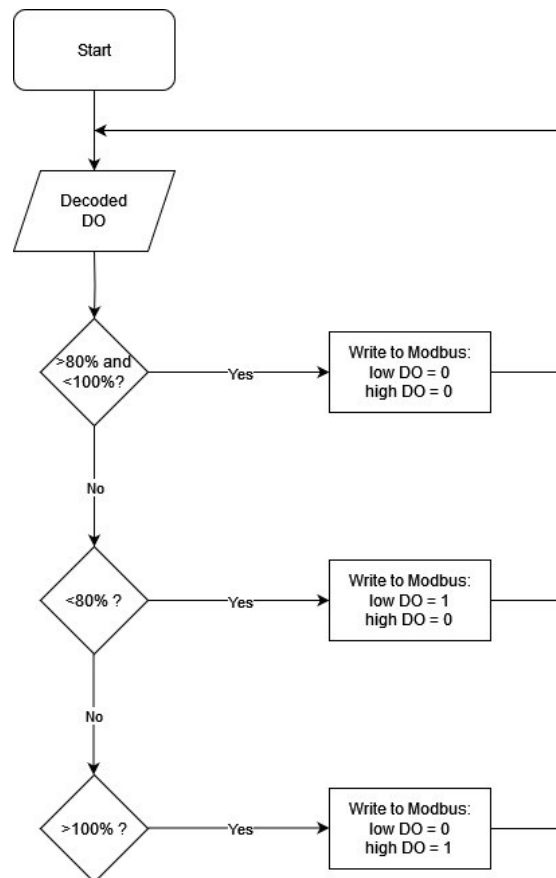
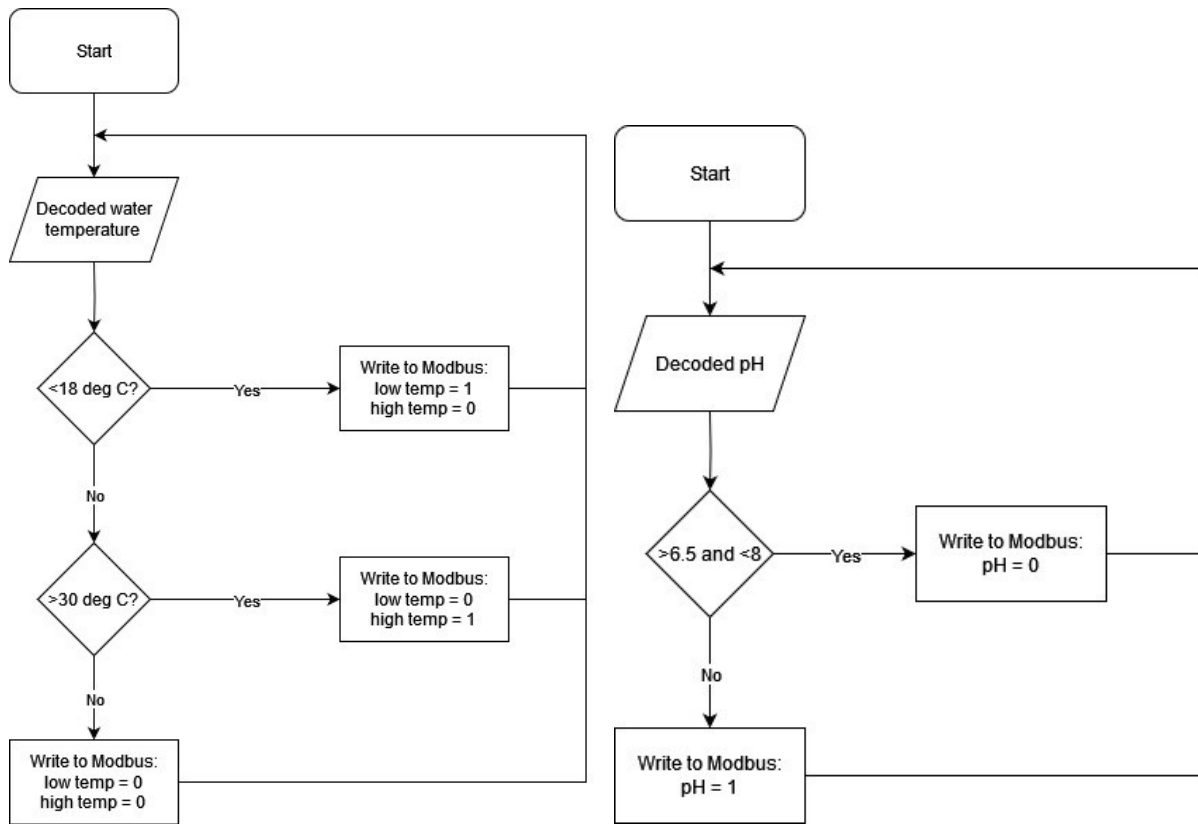


Figure 4: The flow for data processing in Node-Red

An MQTT node in Node-Red is used to subscribe to the broker to receive the sensor information. This is visible in Figure 4 on the top left as the starting point of the flow. The information from here is converted from base64 to hexadecimal via the “base64 to bytes” node and scaled after that to the required pH, DO or temperature value – the code for achieving all this was written in the section on Data Decoding and Scaling. After this, the pseudocode can be referred to again as it is further implemented via switch nodes (as if-statements) which then writes through Modbus to change the status of the inputs in OpenPLC and therefore the automated outputs. The flowcharts below show this process for the water temperature, pH level, and DO which can be converted directly into Node-Red.



After decoding and scaling, the data is also pushed through to the dashboard to be displayed to the user. Reading the values on the Modbus is used to change the dashboard features, such as the LEDs, and to ensure the dashboard displays information correctly. The flow in Figure 5 shows the nodes used and the buttons implemented for user control.

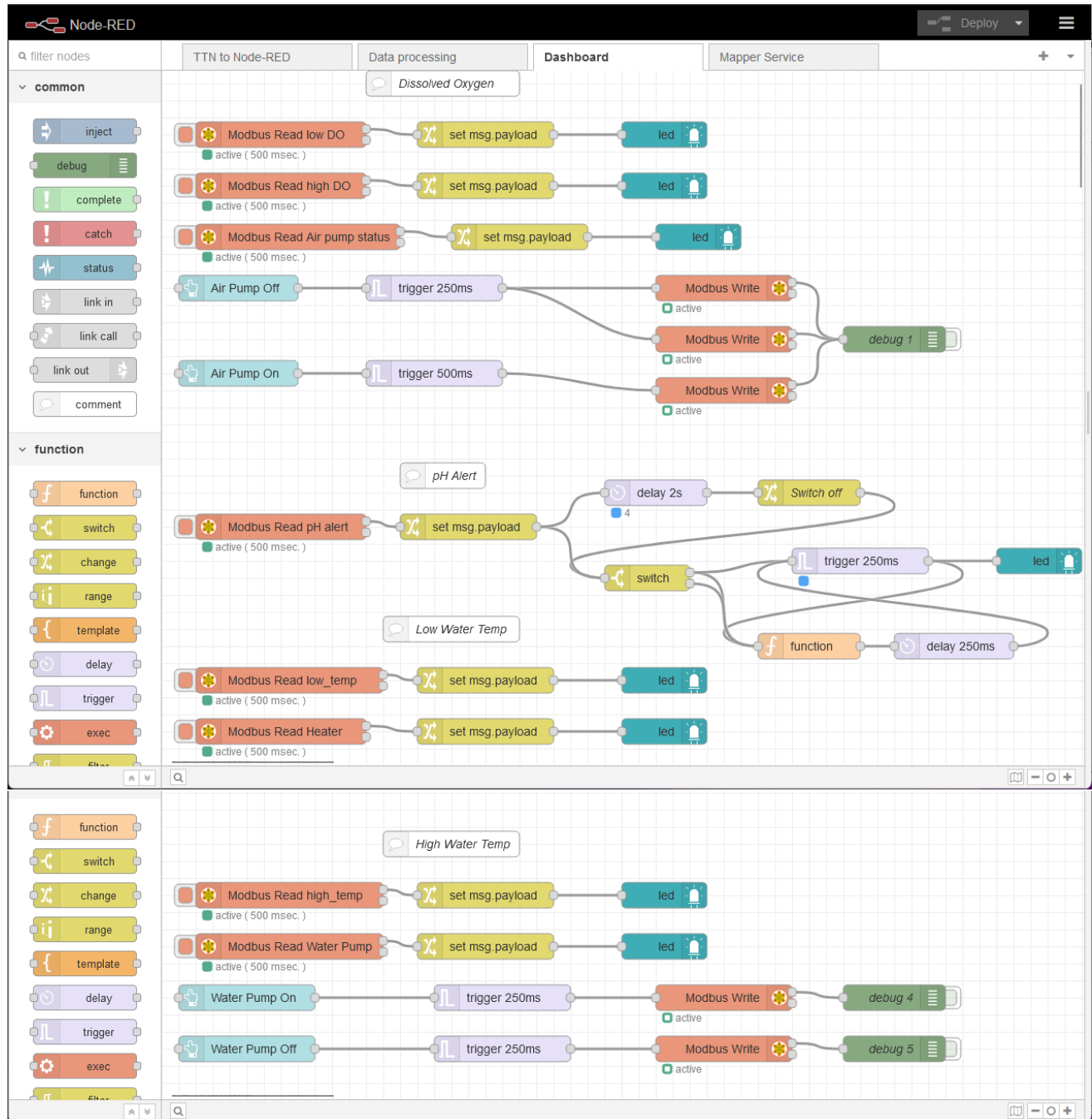


Figure 5: The flow for configuring the dashboard in Node-Red

The images below show an example of the HMI in use where DO is too low and thus the air pump has been turned on. In Figure 6, the pH level is sitting at 5.21 which is too low for the aquaponics system, and in Figure 7 the pH is too high at 8.6. A flashing LED is present to alert the user to the check the tanks. The water temperature is within the acceptable range in both images, but the buttons to turn the pump on or off are still available to the user.

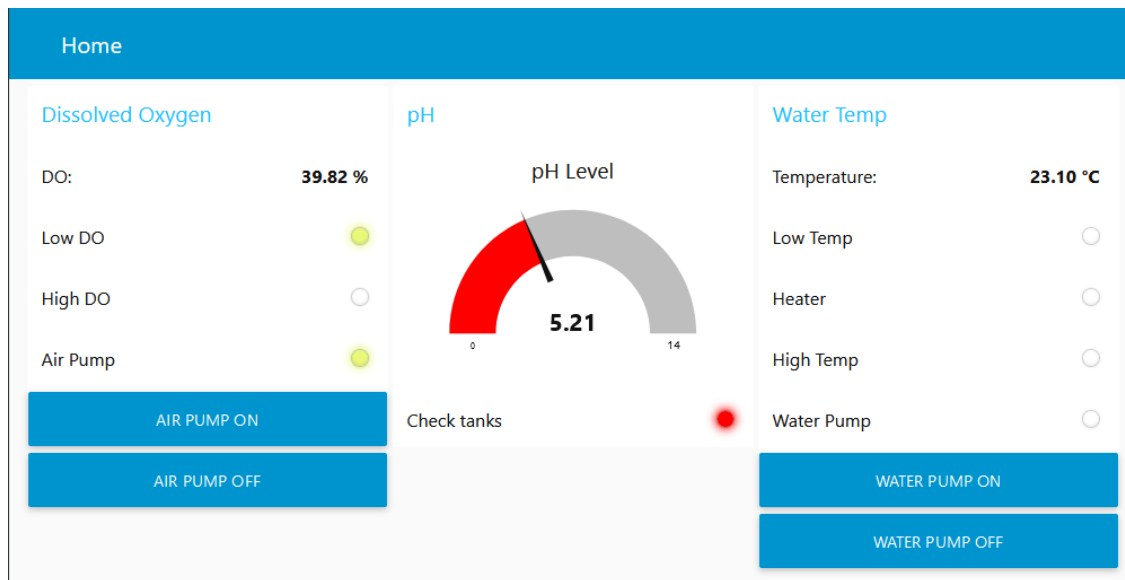


Figure 6: HMI Dashboard with low pH and DO with the air pump on

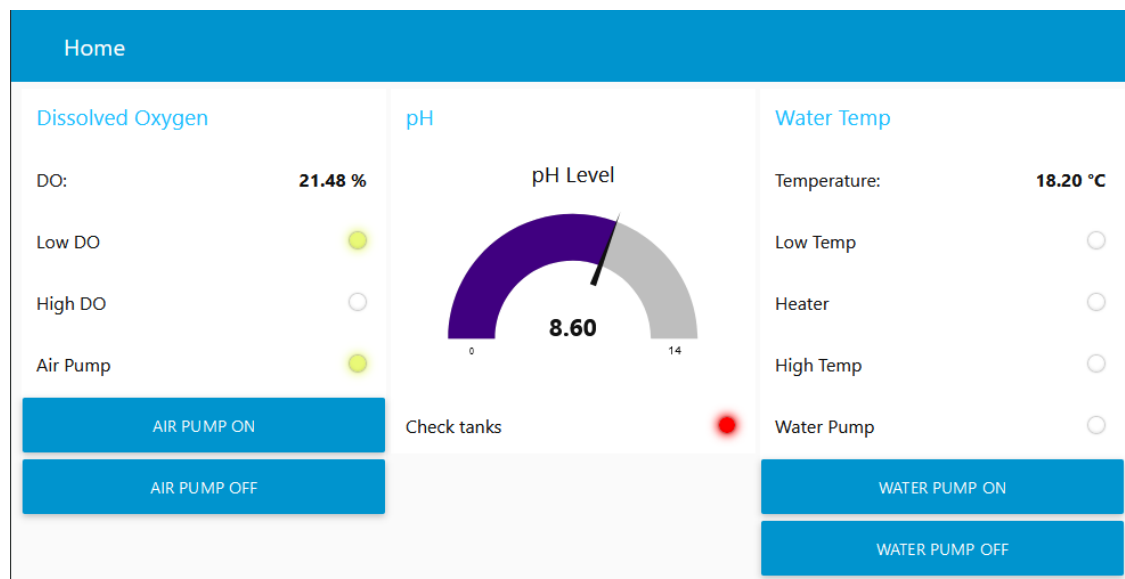


Figure 7: HMI Dashboard with high pH, low DO and normal water temp

The final flow in Node-Red is to push the data to the mapper API. This utilises the decoded DO, water temperature and pH values which are combined into an array. The values in the array can

then be referred to when uploading information to the mapper service. The field names include the following:

- studentid = 61019160
- nickname = KHB
- temp = {{{temp}}}
- hum = {{{pH}}}
- rain = {{{DO}}}
- message = aquaponics system

The URL to push this data through to the mapper service is:

<https://mapper.kist.ws/upload?studentid=61019160&nickname=KHB&temp={{temp}}&hum={{pH}}&rain={{DO}}&message=aquaponics%20system&valid>

Figure 8 shows the created flow with the debug messages on the right-hand side showing the values available in the array under the heading “payload”. The values have been formatted to include units.

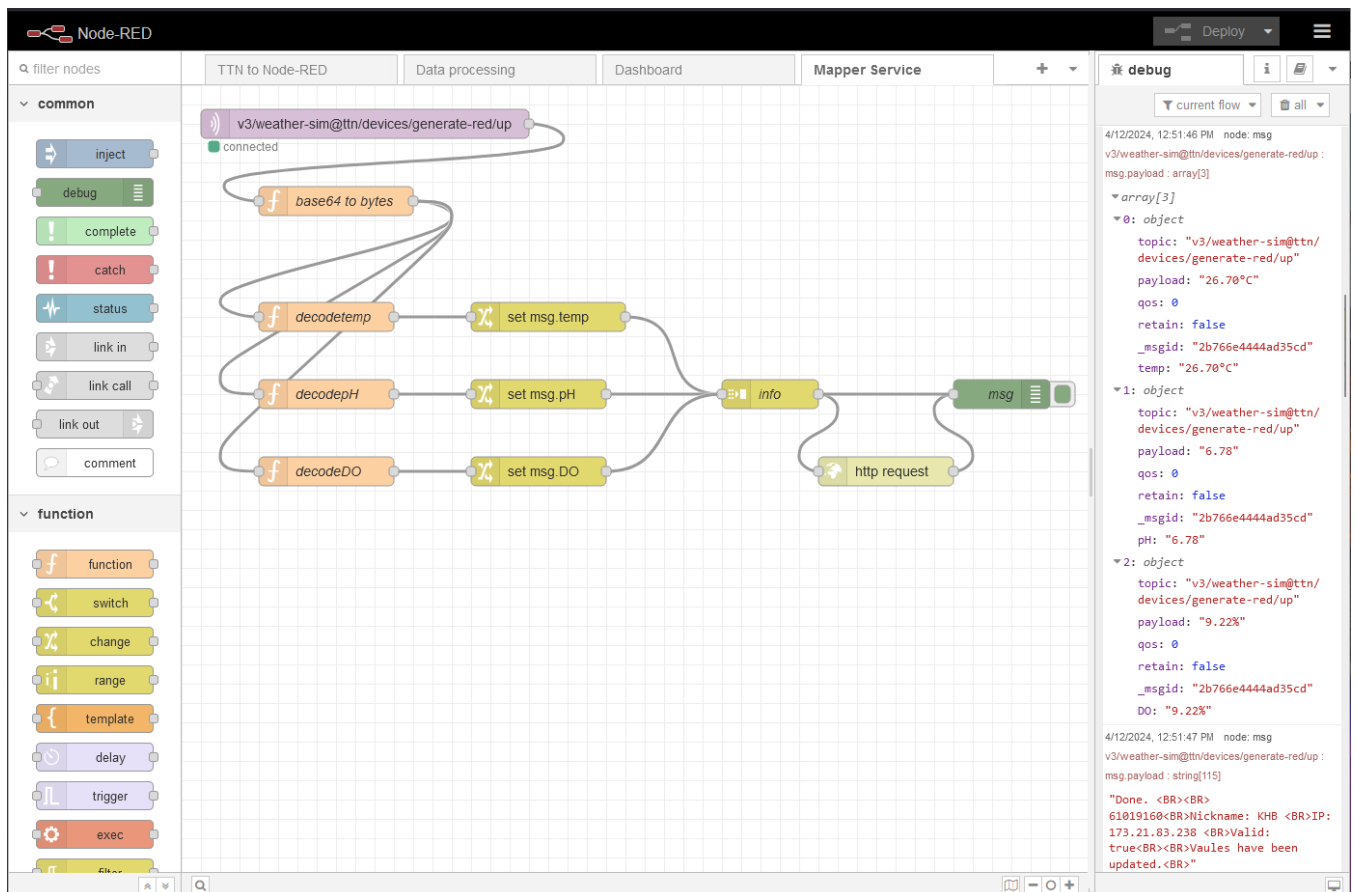


Figure 8: Node-Red flow for uploading data to mapper API

Improvements

After conceptualisation and testing, several improvements have been identified for this system. At present, the functionality is very basic for an aquaponics system and the data provided for the user is limited. It would be ideal to incorporate further information such as water levels of the tanks, flow rate of the water pump, and light or UV levels which pertain to the livelihood of the plants and changes in waste levels [2] [3] [5]. It would also be ideal to send notifications directly to the user's phone when the readings of the tanks are outside their ideal range rather than the user relying solely on the dashboard for updates which would allow for more prompt addressing of any issues [1] [4].

When using the generated test data in Node-Red, all three values are sent at the same, whereas with variable data the information comes through at separate intervals. To improve this flow, there will need to be a node to check for what is sent and only overwrite the field if a new value is sent through while maintaining the old values that haven't been updated. Uploading information to the mapper API could also be improved by creating a condition to validate the information being pushed through. At present all information is uploaded regardless of if it is valid or not, e.g. negative numbers for pH can be uploaded, and any values that are NaN are also uploaded.

Despite these issues, it is believed that this system can be easily changed and adapted to suit the user should they require more sensors and can be easily scaled to incorporate more tanks in the complete setup.

References

- [1] M.M.M. Mahmoud, R. Darwish and A.M. Bassiuny, "Development of an economic smart aquaponic system based on IoT," *J. Eng. Res.*, Aug 2023, doi: <https://doi.org/10.1016/j.jer.2023.08.024>.
- [2] M. F. Taha et al., "Recent Advances of Smart Systems and Internet of Things (IoT) for Aquaponics Automation: A Comprehensive Overview," *Chemosensors*, vol. 10, no. 8, Aug 2022, doi: <https://doi.org/10.3390/chemosensors10080303>
- [3] S. Fu, W. Xing, J. Wu, J. Chen and S. Liu, "Research and design of an intelligent fish tank system," *PLoS ONE*, vol. 18, no. 5, May 2023, doi: <https://doi.org/10.1371/journal.pone.0285105>.
- [4] F. A. Z. Shaikh and U. Bhaskarwar, "Smart Aquarium using IoT," *Int. J. Res. Appl. Sci. Eng. Tech. (IJRASET)*, vol. 10, no. 3, Mar 2022. [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4089695
- [5] C.-H. Chiung, Y.-C. W, J.-X. Zhang and Y.-H. Chen, "IoT-Based Fish Farm Water Quality Monitoring System," *Sensors*, vol. 22, no. 17, Sep 2022, doi: <https://doi.org/10.3390/s22176700>.
- [6] A. Zamifroiu et al. "IoT Communication Security Issues for Companies: Challenges, Protocols and The Web of Data," *Proc. 14th Int. Conf. Bus. Exc. 2020*, vol. 14, no. 1, pp. 1109-1120, doi: 10.2478/picbe-2020-0104.

- [7] B. Russell and D. Van Duren, "Practical Internet of Things Security," Packt Publishing, 2016.
- [8] M. Hunain et al. "Preventing MQTT Vulnerabilities Using IoT-Enabled Intrusion Detection System," *Sensors*, vol. 22, no. 2, doi: 10.3390/s22020567.
- [9] E. Gamess, B. Smith and G. Francia III, "Performance Evaluation Of Modbus TCP In Normal Operation And Under A Distributed Denial Of Service Attack," *Int. J. Comp. Net. Comm. (IJCNC)*, vol. 12, no. 2, doi: 10.5121/ijcnc.2020.12201.
- [10] *Introduction To Modbus TCP/IP*, Acromag, Wixom, MI, USA, 2005.
- [11] *Modbus Application Protocol Specification*, v1.1b3, Modbus, Andover, MA, USA, 2012.
- [12] Fundamentals of Environmental Measurements. "Dissolved Oxygen." [fondriest.com, https://www.fondriest.com/environmental-measurements/parameters/water-quality/dissolved-oxygen/](https://www.fondriest.com/environmental-measurements/parameters/water-quality/dissolved-oxygen/) (accessed Apr. 2, 2024).
- [13] The Things Industries. "LoRaWAN." [thethingsindustries.com, https://www.thethingsindustries.com/docs/getting-started/lorawan-basics/](https://www.thethingsindustries.com/docs/getting-started/lorawan-basics/) (accessed Apr. 6, 2024).
- [14] Cisco. "Understand Host and Subnet Quantities." [cisco.com, https://www.cisco.com/c/en/us/support/docs/ip/routing-information-protocol-rip/13790-8.html](https://www.cisco.com/c/en/us/support/docs/ip/routing-information-protocol-rip/13790-8.html) (accessed Apr. 8 2024).
- [15] Cisco. "Configure IP Addresses and Unique Subnets for New Users". [cisco.com, https://www.cisco.com/c/en/us/support/docs/ip/routing-information-protocol-rip/13788-3.html](https://www.cisco.com/c/en/us/support/docs/ip/routing-information-protocol-rip/13788-3.html) (accessed Apr. 8, 2024).
- [16] The Things Industries. "Authentication." [thethingsindustries.com, https://www.thethingsindustries.com/docs/api/concepts/auth/](https://www.thethingsindustries.com/docs/api/concepts/auth/) (accessed Apr. 9, 2024).
- [17] Node-Red. "Securing Node-RED." [nodered.org, https://nodered.org/docs/user-guide/runtime/securing-node-red](https://nodered.org/docs/user-guide/runtime/securing-node-red) (accessed Apr. 10, 2024).
- [18] W. Alsabbagh, C. Kim and P. Langendörfer, "Good Night, and Good Luck: A Control Logic Injection Attack on OpenPLC," doi: 10.13140/RG.2.2.32913.20321.
- [19] mosquito. "mosquitto-tls man page." [mosquitto.org, https://mosquitto.org/man/mosquitto-tls-7.html](https://mosquitto.org/man/mosquitto-tls-7.html) (accessed Apr. 10, 2024).