

A LOW-COST WATER MANAGEMENT SYSTEM USING IOT FOR FRESHWATER FISH FARMS

Alexander Joseph Johansen

IoT Capstone Project

MicroMasters® Program in Internet of Things (IoT)

**Curtin University (CurtinX) via edX.org**

**2020**

# TABLE OF CONTENTS

<b>List of Figures</b> .....	<b>2</b>
<b>List of Tables</b> .....	<b>2</b>
<b>Introduction</b> .....	<b>2</b>
Learning Outcomes .....	3
What problem is being solved? .....	3
Summary .....	3
<b>Literature Review and Market Research</b> .....	<b>3</b>
Literature Review .....	3
Market Research .....	4
Summary .....	6
<b>Design Methodologies</b> .....	<b>7</b>
Requirements.....	7
Specifications .....	7
Evaluation of alternative solutions.....	8
Project Plan .....	12
Prototype progress.....	13
Summary .....	17
<b>Evaluation and testing</b> .....	<b>17</b>
Evaluation of progress.....	17
<b>Conclusion</b> .....	<b>17</b>
Future work.....	18
<b>References</b> .....	<b>19</b>

List of Figures

Figure 1: NH3 testing .....9

Figure 2: pH testing.....10

Figure 3: digital meter.....10

Figure 4: Node-Red MQTT communication setup .....14

Figure 5: Node-RED programming.....15

Figure 6: Node-RED dashbaord via browser .....16

Figure 7: Simple network diagram of the solution .....17

List of Tables

[Table 1: Industry analysis](#).....5

[Table 2: Competitor analysis](#).....6

## Introduction

### Learning Outcomes

The student wanted to gain more understanding of the IoT systems and to understand how to design an IoT. Student has been introduced to the MQTT protocol and ZigBee protocol (IEEE 802.14.5) along with how to configure microcontrollers with sensors and actuators. Understanding of different data types, different communication technologies for IoT, cloud and fog have been understood and applied in this work. Student has also been introduced to; how to design & implement a secure IoT system.

### What problem is being solved?

Since Thailand is in the middle of a transformation to Thailand 4.0 [1], new technologies are being introduced to all sectors; traditional farming → smart farming, traditional services → high value services and from buying technologies to making technologies. In 2000, a total area of 96,145 hectares was freshwater aquaculture farms in Thailand [2]. A low-tech fish farm requires little knowledge to set up but managing the right level of water quality requires more. To maintain a good quality water for fish needs to have a pH value between 7.5-8.5, dissolved oxygen value above 3 mg pr. liter, ammonia and nitrite less than 0,3 mg pr. liter and Oxidation-Reduction Potential or O.R.P., generally a O.R.P. reading of between +125 to -200 mV for freshwater systems and ponds. Now a day, farmers depend on the chemical tester kits for measure the quality of the water. This is a time-consuming procedure. The IoT system that has been developed, can tackle the problem of the time and resources to manage the water quality for fish farms and for saving the water resources, the system supports having a water treatment pond so after the water is treated, farmers can reuse the water. The water is considered treated when their have acceptable readings from the sensors. Farmers also need to reserve the water for the dry seasons and needs to monitor the level of the water in the pond closely in the rainy seasons for not flooding the fishpond.

### Summary

When looking into the solutions of the IoT system that has been developed, there are 3 main problems that are being solved. The water quality, secondly, water resource saving and lastly, forecasting water resource.

## Literature Review and Market Research

### Literature Review

The Internet of Things a new shift in the IT area. The phrase “Internet of Things” or IoT is coined from two words. The first word “Internet” and the second word is “Things”. The Internet is a network of networks that are interconnected to form a routable network on different mediums. While the Things refers to any object or person that can be distinguishable by the real world. We normally think of “things” as electronic devices and do not count other physical objects. In the IoT world, we can connect “Everything” to the Internet.

IoT has drawn attention for IT researchers to research on how we can improve the business processes with this technology. Generally, smart farming is a farming management concept using modern technology to increase the quantity and quality of agricultural products. Farmers in the 21st century have access to GPS, soil scanning, data management, and Internet of Things technologies. By precisely measuring variations within a field and adapting the strategy accordingly, farmers can greatly increase the effectiveness of pesticides and fertilizers, and use them more selectively.

Similarly, using Smart Farming techniques, farmers can better monitor the needs of individual animals and adjust their nutrition correspondingly, thereby preventing disease and enhancing herd health [3].

In aquaculture, that is a lagging area of technology compared to other areas such as agriculture. So, it is important to solve the problems that are in this area with the support of technology. Encinas et al. [4] presents the implementation of a prototype and a proof of concept about a remote monitoring system applying the concept of IoT among others technologies addressed to aquaculture water quality. The system is low cost, low power consumption, scalable, versatile, distributed, mobile and accurate but required that the system to be always connected to the Internet to push data to the cloud. Niswar et al. [5] proposed a IoT system for monitoring a crab farm but the system lacks the automatic water quality correction by cycling the water. Dupont et al. [6] detailed the following for water quality parameters;

“The fisheries management relies totally on the water quality monitoring. Fish diseases are very frequent and impact directly the harvesting yield. A low water quality can also impact the fish growth and delay the harvest. The optimum fish production is totally dependent on the physical, chemical and biological qualities of water no matter the type of facility. Therefore, water quality is the key to succeed a good fishery management. It is determined by variables such as temperature, turbidity, carbon dioxide, pH, alkalinity, ammonia, nitrite, nitrate, etc. Amongst them, the most critical are temperature, dissolved oxygen and pH.

Optimum temperature is dependent of the fish species, but as fish are cold blooded animals, it is vital that the temperature is controlled and maintained in the correct range. And even in the correct range, higher temperature increases the rate of bio-chemical activity of the microbiota and so increase the oxygen demand. To limit disease and oxygen consumption, temperature has to be finely regulated.

Optimum dissolved oxygen should always be above 5 ppm. Fish needs enough oxygen in the water to survive, otherwise they stay at the surface to catch up more oxygen, have slower metabolism and grow slower, and ultimately can die of lack of oxygen. It is even a bigger problem for aquatic organism to obtain sufficient oxygen than for terrestrial ones, due to low solubility of oxygen in water.

Optimum pH for fish life is between 7 and 8.5, ideal for biological productivity, otherwise fishes can become stressed in water, again slowing down their growth. Many other parameters may be also monitored, but they generally directly influence the 3 main parameters mentioned above. Monitoring and controlling these parameters are therefore the basis for a good water quality. In addition, real-time monitor will provide faster reaction time.”

## Market Research

To identify the problem area in our business and understand the needs of existing customers and why they chose our service over competitors, the student has done the industry and competitor analysis.

## 1. Industry analysis

Aspect	Analysis
<b>Political</b>	Thailand is in a middle of a transformation to Thailand 4.0 which includes using more technology in every sector; ex. traditional farming --> smart farming. Since using the technology to solve current and new challenges is in the focus of the Kingdom, the solution developed is a bull's-eye.
<b>Economic</b>	Currently, the global economic is quite unstable, the business might have problems with the exchange rates and inflation in the near future. The price of products might get a lot higher since all of the materials are being ordered from China.
<b>Social</b>	Thailand is prepared to enter into a new era, Thailand 4.0. People are already introduced to new technology. There might be some challenges to the fish farmers if we can't show them the benefits of the system but the student personally thinks that new fish farmers that want to enter this business will use the system to manage their farms.
<b>Technology</b>	The internet access for pushing the data to the cloud might be a problem for some of the rural farms. A 4G/5G-router might be used to fill that gap, but the cost will get higher.
<b>Industry</b>	Smart Farming is trending in Thailand; to enter into this industry, we have to provide an easy to system to operate and very importantly, it must be a cheap system and work efficiently.

## 2. Competitor analysis

Competitor	Est. date	Product	Market share (%)	Value to customers	Strengths	Weaknesses
<b>Endress+Hause r (Thailand) Ltd.</b>	2000	Netilion Smart System for Aquaculture - can measure these parameters: * Dissolved oxygen * Temperature * Ammonium * pH / Nitrate	10%	Quality / convenience	A solution in one "box". Extremely easy to setup by an app on cellphone.	Expensive, must pay subscription for using the application.
<b>Siemens (Global)</b>	1847	A solution to aquaculture industry. This is a very high-end complete solution to run high-tech fish farms.	1%	Quality / service	Systematic and standardized solution for running fish farms.	Awfully expensive, normal fish farmers cannot afford the system.
<b>NECTEC (Thai government)</b>	1986	They might start developing the system by themselves since NECTEC is a statutory government organization.	30%	Quality / price	Have good knowledge and resources to "copy" our system.	Might takes time if they are not prioritizing the development of the product.
<b>eFishery (Indonesia)</b>	2018	Indonesian fishtech startup "eFishery" has been launching a pilot project to introduce its smart fish feeder technology.	20%	Service	They have already introduced their product (fish feeder) and might copy and introduce their customers.	Takes time to develop.

### Summary

For maintaining a good level of water quality, multiple sensors were used for real-time monitoring so the farmers can act instantly when the quality of the water is decreasing. The system itself contains

multiple sensors to collect data that can measure these parameters: Dissolved oxygen, Temperature, Ammonium, pH / Nitrate. An automatic Water Quality Correction mechanism will be provided to maintain the water quality by cycling the water. The system will track the water usage in the farm and along with the public forecast data, we can forecast the volume that is needed in the future.

Since this is a low-cost system. Most of the farmers interested in smart "fish"-farming can afford the system. When compared to the competitors; our solution is much cheaper than what it is out there. so, we are expecting a big market share in the beginning when entering to this market and might decrease when NECTEC or eFishery start introducing same product/solution.

## Design Methodologies

### Requirements

To complete this project comprehensively, there are many parameters to be measured. Due to time and resources constraints (because of Corona virus pandemic, main sensors could not be shipped from China), only certain parameters will be measured. The device must:

- Be able to monitor the water quality (pH, DO, NH<sub>3</sub>/NH<sub>4</sub>) constantly by emulating sensors with Arduino.
- Basic Automatic water quality correction function.
- Capture and store data from water level sensor from own forecasting for water management.
- Capture and store data from sensors to local database for analysis with ML
- Be able to measure pH, DO, NH<sub>3</sub>/NH<sub>4</sub> to determine if the quality of the water is in acceptable level. If not, alert the user and/or pump in good water into the system.
- Basic User Interface on PC/mobile phone
- Program RF modules on microcontrollers to be able to communicate with ZigBee in API mode

### Specifications

- **Required devices to be connected:**
  - **sensor:** DO, pH, NH<sub>3</sub>/NH<sub>4</sub>, water level sensors for measuring the water quality and the amount of water in every pond in the system.
  - **actuator:** 220V relay for switching the water pump on and off.
  - **intermediate devices:** Microcontrollers (ATMEL's ATMEGA328P) with sensors mentioned earlier along with a RF module (ZigBee), aerial, sink node (acts like a gateway to TCP/IP network to send/receive data to/from MQTT broker) and SBC (Single Board Computer with IoT middleware and MQTT broker installed).
- **IoT communication protocol:** Short range IoT protocol that supports forwarding packets on the device itself. The choice of IoT protocol is IEEE 802.14.5 or ZigBee standard.
- **Data:**
  - **type generated:** Volume, limited data range and will be partly processed in the fog. Basic analysis locally in the fog and push analyzed data to the cloud for further processing.
  - **storage solution:** local storage (SQL database).
- **Programming required:**



- **device function:** Programmed to collect all sensors data for analysis of the water quality and trigger the automatic water quality correction function.
  - **networking requirement:** Sensor-nodes can route packets to each other without a wireless access point. The wireless network will require a sink-node/gateway which will act like a bridge to the LAN.
  - **data management:** Algorithms are needed to identify the data for water quality from sensor-data and activate the water-pump when the quality of the water is out of range along with storing data in SQL database locally.
- **Security and Privacy:**
    - **device:** Need to be near the fishponds. Physical security is needed.
    - **network:** Needs to only transmit to own farm. Use some form of encryption to overcome the problem of network eavesdropping.
    - **data:** Backup of historical data to a remote site (cloud). Algorithm to detect data-errors for system's operation accuracy.

#### Evaluation of alternative solutions

Basically, a traditional chemical testing kits can be used to test the quality of the water. This is a very time-consuming process. The user needs to collect a sample of the water to be tested and then mix the chemical solution to it and compare the color of that water with the chemicals in it. If you want to sample 3 parameters, then you must do the testing 3 times. For example, to get the values of DO, pH, NH<sub>3</sub>/NH<sub>4</sub> then you must do 3 separate tests. Fig. 1 and fig. 2 show the process of testing to get NH<sub>3</sub> and pH values.



Figure 1:  $\text{NH}_3$  testing



Figure 2: pH testing

Another way to test the water quality is to use the digital water testers/meters. Generally, if one needs to measure multiple parameters, multiple testers are being needed. Digital meters are more expensive than using chemicals to test the water. Readings are on a limited LCD-screen attached to the device. Data from readings cannot be exported externally, users will have to log the reading by themselves. Figure 3 is an example of a digital meter for measuring pH & temperature.



Figure 3: digital meter

Compared to the solution that has been developed, the new and novel system which can display the values of measurement of parameters in real-time via a web browser, both on mobile-phone and PC. Instant alert on mobile phone via email and IM clients when the water quality is out of range, is possible with this new novel system. The system that has been developed also have an “automatic water quality correction” function which could cycle in fresh & clean water into the main pond automatically. Readings are stored automatically in the database for processing and historical data can be reviewed afterwards.



## Prototype progress

The development of the prototype has been grouped into the following:

### 1. Installation and configuration of the server in the fog

The base OS on this server was Armbian, Linux for ARM development boards. Since Raspberry pi boards were not available in the lab, Orange Pi PC+ board was used. After Armbian was installed, the system was up and running. Network was provided to the board via an ethernet cable connected to the lab's router.

### 2. Installation and configuration of IoT middleware & MQTT broker

For the middleware, Node-RED was installed via a script. Node-RED is a programming tool for wiring together hardware devices, APIs and online services in new and interesting ways. It provides a browser-based editor that makes it easy to wire together flows using the wide range of nodes in the palette that can be deployed to its runtime in a single-click. Mosquitto, an open source (EPL/EDL licensed) message broker that implements the MQTT protocol versions 5.0, 3.1.1 and 3.1, was installed. Mosquitto is lightweight and is suitable for use on all devices from low power single board computers to full servers.

### 3. Installation and configuration of the coordinator node

The coordinator node was an Arduino UNO board with an Xbee RF module installed on an Xbee shield along with an Ethernet module to support wired TCP/IP network connection. There were 2 modes that could be configured on Xbee: AT mode or API mode. To form a working PAN (Personal Area Network) with unique addresses to send and receive packets to/from Xbee modules, API mode was chosen. In API mode, we can uniquely send and receive from both the coordinator and others XBees in the network. A firmware update process with XCTU software was performed to install the newest firmware with correct functionality for the coordinator node. API escaped (API 2) was set. In API 2 mode, the length field does not include any escape character in the frame and the checksum is calculated with non-escaped data.

### 4. Installation and configuration of wireless sensor nodes

The router node was programmed with the XCTU software to upgrade to the latest firmware and to set the correct function of the Zigbee hardware. This node also uses Xbee shield along with the Arduino UNO micro controller development board. To simulate the sensors that could not be shipped from supplier in China, 4 buttons were hooked up with Arduino to increase / decrease the value of the emulated sensors (D.O. and pH). For temperature, DHT11 was used.

### 5. Programming the microcontrollers to communicate with the MQTT broker

#### 5.1 Programming the coordinator node

The coordinator node was programmed to publish 3 values to the MQTT broker that it received from the wireless sensor node. The values were being published every 10 seconds. The coordinator node was getting its IP from the DHCP server in the lab's router via ethernet cable. After IP connectivity, the node then made connection to MQTT broker with "ALX" as client id. After it has made the connection with the MQTT broker, Arduino then PUBLISH and SUBSCRIBE to the topics of interest. Instructions for activating the relay (water pump) was programmed. The function is triggered by a message that was being sent from the MQTT broker that the Arduino is subscribed to. (see Appendix A)

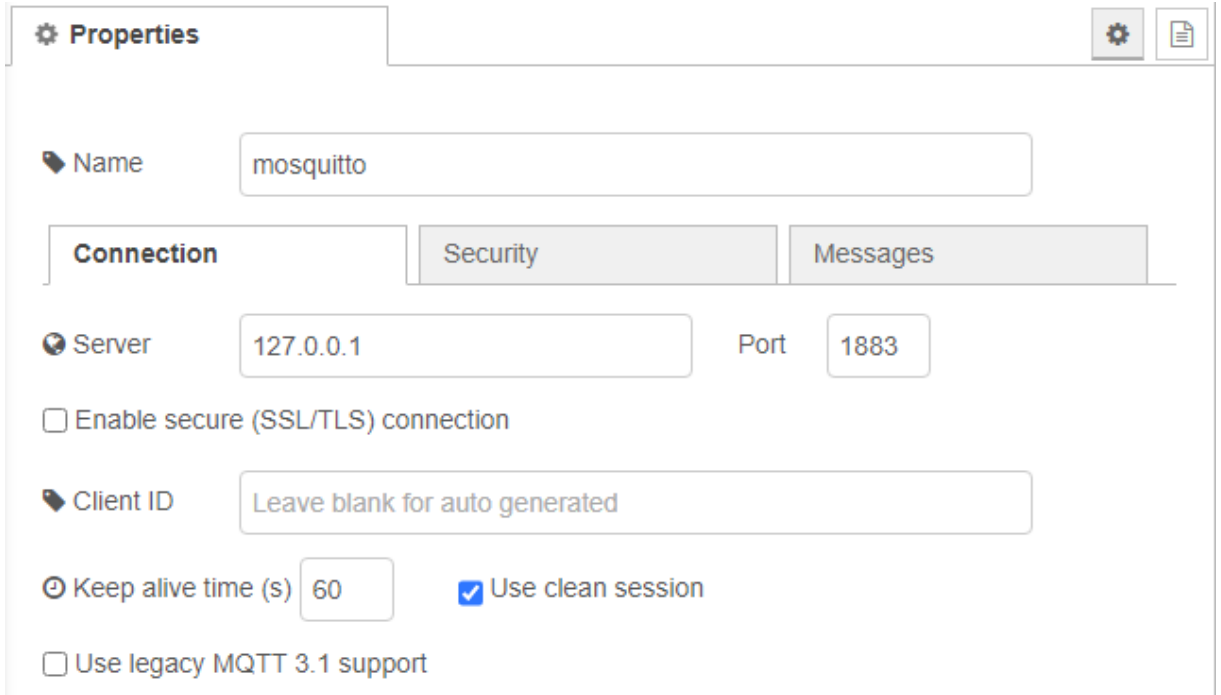
#### 5.2 Programming the wireless sensor nodes

The wireless sensor node (router node) was programmed to send a ZigBee packet by creating a payload of 3 values collected from sensors. The ZigBee then sent those packets that were

created every 10 seconds to the coordinator by specifying the address in the setAddress64 function, "txRequest.setAddress64(0x0013a20041a7b0d3)". (see Appendix B)

## 6. Programming Node-Red to communicate with the MQTT broker

Node-Red was configured to communicate with the MQTT broker that was installed on the same server as Node-Red.



The image shows the 'Properties' panel for an MQTT broker node in Node-Red. The 'Name' field is set to 'mosquitto'. The 'Connection' tab is active, showing the 'Server' as '127.0.0.1' and the 'Port' as '1883'. There are three tabs: 'Connection', 'Security', and 'Messages'. Under 'Connection', there is a checkbox for 'Enable secure (SSL/TLS) connection' which is unchecked. The 'Client ID' field is set to 'Leave blank for auto generated'. There is a radio button for 'Keep alive time (s)' set to '60' and a checked checkbox for 'Use clean session'. At the bottom, there is an unchecked checkbox for 'Use legacy MQTT 3.1 support'.

Figure 4: Node-Red MQTT communication setup

## 7. Programming in IoT middleware; Node-RED

Since this system was trying to push all the system's intelligence into the fog, programming in Node-RED was a must. Pushing the intelligence to the microcontroller would be a bad idea, the coordinator node's sketch used 24770 bytes (76%) of program storage space. Global variables used 1202 bytes (58%) of dynamic memory, leaving 846 bytes for local variables. Maximum is 2048 bytes. One of the requirements was to have a dashboard to visualize the data from sensors, a dashboard module was installed on Node-RED. Programming Node-RED was straightforward. Nodes were connected to MQTT broker to subscribe to the topics that were sent from the coordinator node and visualized them on the dashboard which was accessible from a web browser (fig. 6).

Programming instructions:

- Temperature  
Data for temperature was directly visualized to the dashboard by utilizing the "chart node" in Node-RED (fig. 5).
- pH

2 flows were created for pH. First one for visualizing the values and the second one for alerting the user when the pH value is out of range (fig. 5).

- D.O.  
2 flows were created for D.O. First one for visualizing the DO values and the second one for activating the pump. A custom function was created to PUBLISH the correct data so Coordinator node could react correctly when received the message. (see Appendix C)
- Address64  
Address (LSB) of remote wireless sensor node was directly visualized to the dashboard (fig. 5).

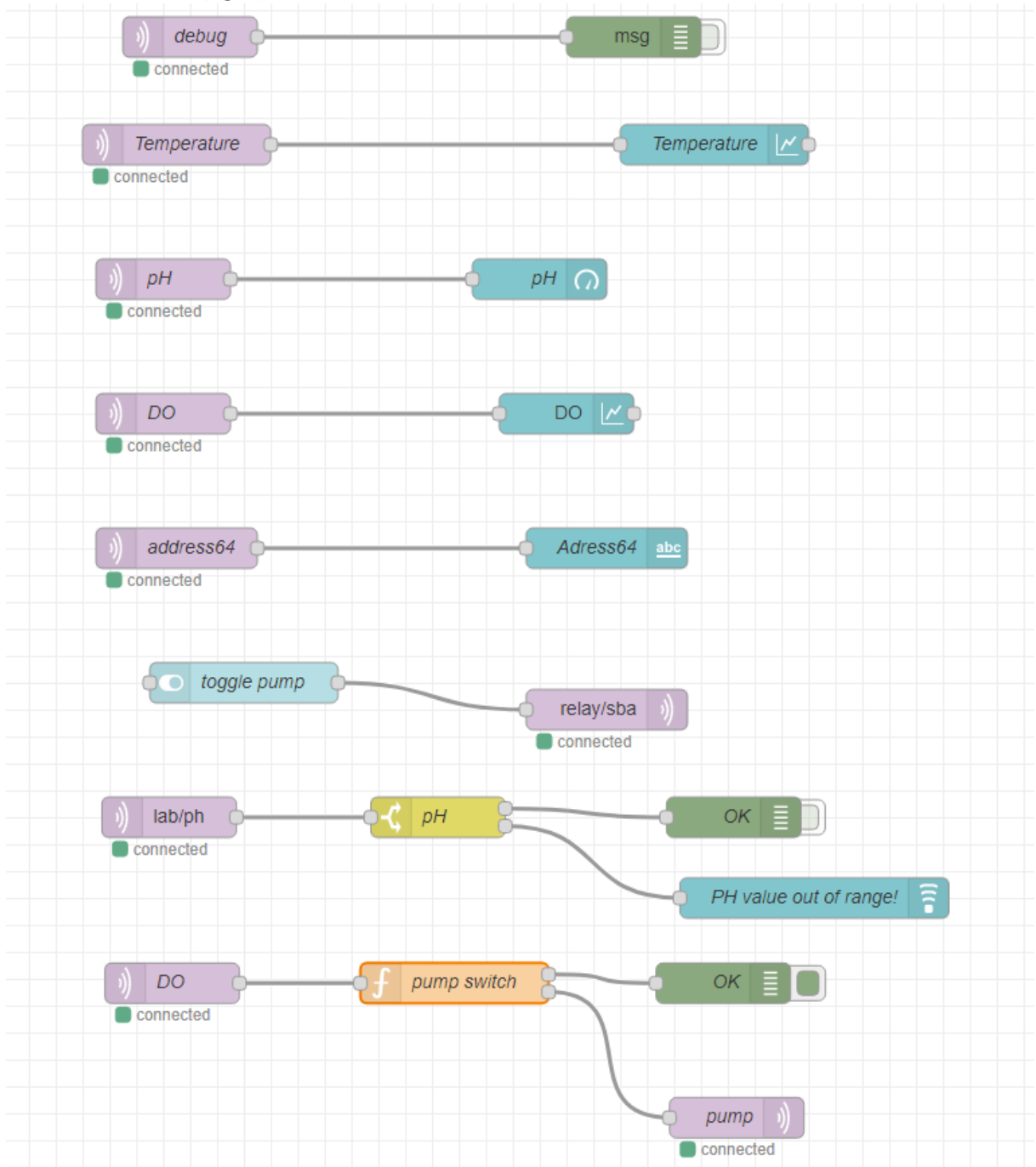
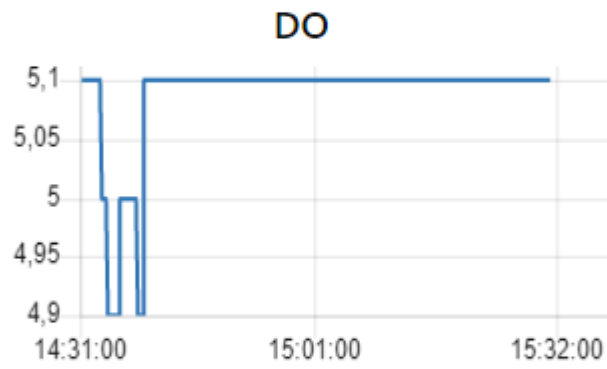
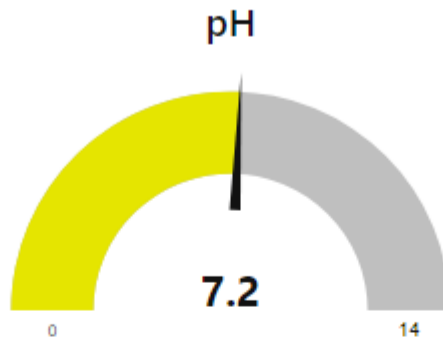


Figure 5: Node-RED programming



### Sensors



Pond ID: **b0b9**

switch

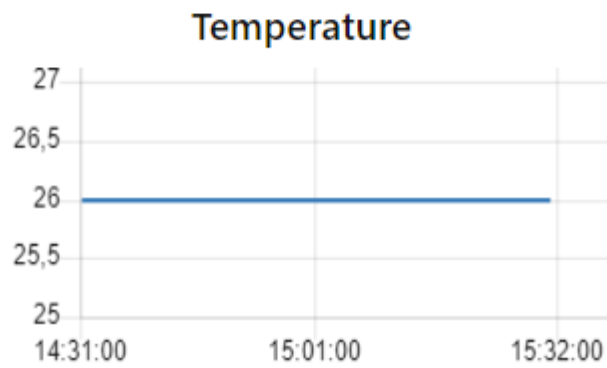


Figure 6: Node-RED dashbaord via browser

## Summary

The system has been developed to the specifications mentioned earlier in this section. The system filled the gaps that were left from previous research. The storage of the sensor's data was being stored locally in the fog and would have possibility to be pushed to the cloud for further analysis if we have had more time to develop the system. A simple network diagram of the system is detailed in figure 7.

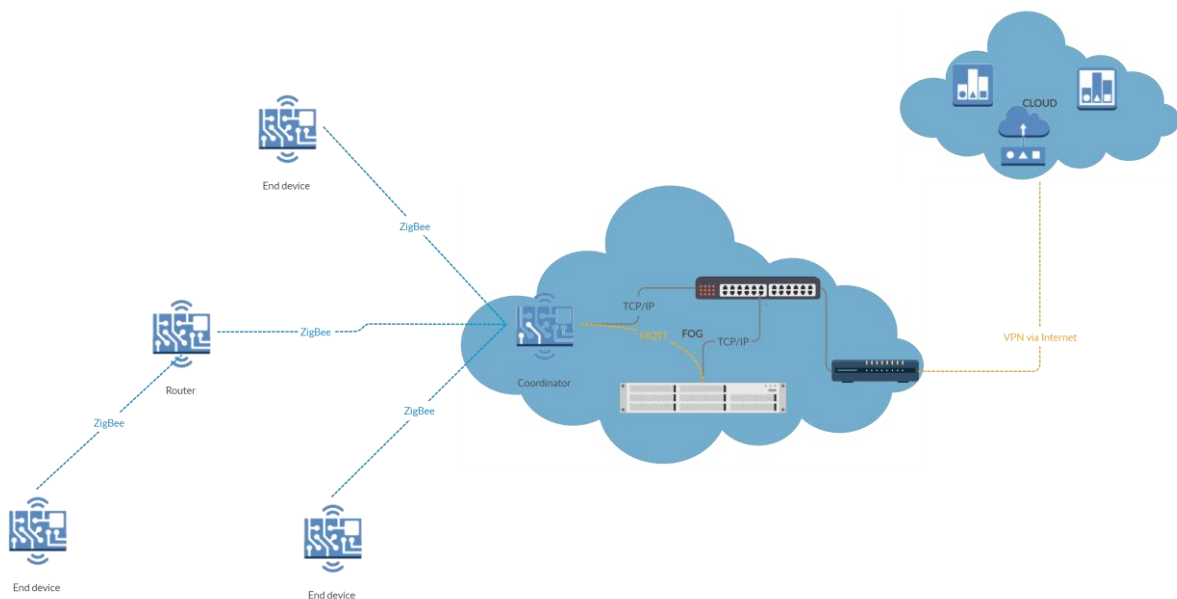


Figure 7: Simple network diagram of the solution

## Evaluation and testing

### Evaluation of progress

The system has been developed to function as intended. With the limit of time and budget, this project could have been finalized better. Arduino UNO has only 1 serial interface which has to be shared with communication with XBee module, this was making trouble for debugging the code on the microcontroller. Arduino Leonardo should have been used. SQL integration was not programmed in Node-RED since the database design must have been in place first. Remote relay via ZigBee was not programmed since the lag of time and debugging the code was exceedingly difficult.

## Conclusion

This work has given the student knowledge for how to implement an IoT system with Arduino and ZigBee. The prototype is functional but requires more testing before it can be commercialized.

### Future work

The system should be tested in real environment with real sensors. SQL integration can be added to the system by using a Node-RED node. Forwarding of packets on the router node could be tested more. Commercializing the system will need more testing to be covered. RF modules could be replaced with 802.11b standard to simplify the setup but this will require a wireless access point available at the site.

## References

- [1] R. T. E. W. D.C., "What is Thailand 4.0?" [Online]. Available: <https://thaiembdc.org/thailand-4-0-2/>.
- [2] P. Edwards, *OVERVIEW OF SMALL-SCALE FRESHWATER AQUACULTURE IN THAILAND*. 2003.
- [3] "Smart Farming is key for the future of agriculture." 2017, [Online]. Available: <https://www.schuttelaar-partners.com/news/2017/smart-farming-is-key-for-the-future-of-agriculture>.
- [4] C. Encinas, E. Ruiz, J. Cortez, and A. Espinoza, "Design and implementation of a distributed IoT system for the monitoring of water quality in aquaculture," in *2017 Wireless Telecommunications Symposium (WTS)*, Apr. 2017, pp. 1–7, doi: 10.1109/WTS.2017.7943540.
- [5] M. Niswar *et al.*, "IoT-based Water Quality Monitoring System for Soft-Shell Crab Farming," in *2018 IEEE International Conference on Internet of Things and Intelligence System (IOTAIS)*, Nov. 2018, pp. 6–9, doi: 10.1109/IOTAIS.2018.8600828.
- [6] Charlotte DUPONT, "Low-Cost IoT Solutions for Fish Farmers in Africa," 2018, [Online]. Available: [http://cpham.perso.univ-pau.fr/Paper/ISTAfrica2018\\_Paper\\_ref\\_52.pdf](http://cpham.perso.univ-pau.fr/Paper/ISTAfrica2018_Paper_ref_52.pdf).

## Appendix A

```
#include <SPI.h>

#include <PubSubClient.h>
#include "binary.h"
#include <SPI.h>

#include <Ethernet.h>

#include <dht11.h>
#include <XBee.h>

XBeeWithCallbacks xbee;

dht11 DHT11;

#define DHT11PIN 7
#define CLIENT_ID "alx"
#define ledPin 9
#define relayPin 8

EthernetClient ethClient;
PubSubClient mqttClient;

bool statusKD = HIGH;
bool statusBD = HIGH;
bool statusGD = HIGH;
bool relaystate = LOW;
bool pir = LOW;
bool startsend = HIGH;// flag for sending at startup
int lichtstatus; //contains LDR reading
const int lamp = 4;
String ip = "";

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
//IPAddress ip(192,168,0,177);//modifying according your own IP
IPAddress server(192, 168, 0, 21);
// Initialize the Ethernet server library
// with the IP address and port you want to use
// (port 80 is default for HTTP):
//EthernetServer server(80);

// Callback function header
//void callback(char* topic, byte* payload, unsigned int length);
//PubSubClient client(server, 1883, callback, ethClient);

void setup()
{
  pinMode(ledPin, OUTPUT);

  Serial.begin(9600);
  //Ethernet.begin(mac, ip);
  //server.begin();
  xbee.begin(Serial);
  delay(1);

  // Setup callbacks
```

```

xbee.onZBRxResponse(processRxPacket);

    if (Ethernet.begin(mac) == 0) {
        Serial.println(F("Unable to configure Ethernet using DHCP"));
        for (;;)
    }
    ip = String (Ethernet.localIP()[0]);
    ip = ip + ".";
    ip = ip + String (Ethernet.localIP()[1]);
    ip = ip + ".";
    ip = ip + String (Ethernet.localIP()[2]);
    ip = ip + ".";
    ip = ip + String (Ethernet.localIP()[3]);
    //Serial.println(ip);
    Serial.println(F("Ethernet configured via DHCP"));
    Serial.print("IP address: ");
    Serial.println(Ethernet.localIP());
    Serial.println();

    // setup mqtt client
    mqttClient.setClient(ethClient);
    mqttClient.setServer(server, 1883); // or local broker
    Serial.println(F("MQTT client configured"));
    mqttClient.setCallback(callback);

    Serial.println("Connecting to the MQTT broker...");
    mqttClient.connect(CLIENT_ID);
    mqttClient.publish("home/br/nb/ip", ip.c_str());
    mqttClient.subscribe("lab/sba");

}

void processRxPacket(ZBRxResponse& rx, uintptr_t) {
    /*DebugSerial.print(F("Received packet from "));
    */
    //printHex(Serial, rx.getRemoteAddress64());
    /*
    DebugSerial.println();
    DebugSerial.print(F("Payload: "));
    DebugSerial.write(rx.getData(), rx.getDataLength());
    DebugSerial.println(); */
    //digitalWrite(ledPin, HIGH);
    Buffer b(rx.getData(), rx.getDataLength());
    XBeeAddress64 addr = rx.getRemoteAddress64();
    uint8_t type = b.remove<uint8_t>();

    if (type == 1 && b.len() == 12) {
        /* DebugSerial.print(F("DHT packet received from "));
        printHex(DebugSerial, rx.getRemoteAddress64());
        DebugSerial.println();
        DebugSerial.print(F("Temperature: "));
        DebugSerial.println(b.remove<float>());
        DebugSerial.print(F("Humidity: "));
        DebugSerial.println(b.remove<float>());*/
        if (mqttClient.connected()) {

            int nHex = addr.getLsb();
            char pHexStr[100];
            sprintf(pHexStr, "%x", nHex);

```

```

static char xTemp[7];
dtostrf(b.remove<float>(), 6, 2, xTemp);
static char yTemp[7];
dtostrf(b.remove<float>(), 6, 2, yTemp);
static char zTemp[7];
dtostrf(b.remove<float>(), 6, 2, zTemp);

mqttClient.publish("lab/ph", xTemp);
mqttClient.publish("lab/do", yTemp);
mqttClient.publish("lab/temperature", zTemp);
mqttClient.publish("lab/address64", pHexStr);
mqttClient.subscribe("relay/sba");

}
else {reconnect();}
return;
}

}

void loop()
{
/*Serial.println();

int chk = DHT11.read(DHT11PIN);

Serial.print("Humidity (%): ");
Serial.println((float)DHT11.humidity, 2);
static char humidityTemp[7];
dtostrf(DHT11.humidity, 6, 2, humidityTemp);
Serial.print("Temperature (C): ");
Serial.println((float)DHT11.temperature, 2);
static char temperatureTemp[7];
dtostrf(DHT11.temperature, 6, 2, temperatureTemp);
if (mqttClient.connected()) {
mqttClient.publish("room/temperature", temperatureTemp);
mqttClient.publish("room/humidity", humidityTemp);
}
else {reconnect();}
delay(3000);*/
xbee.loop();
mqttClient.loop();
//mqttClient.subscribe("lab/sba");
delay(5000);
//digitalWrite(ledPin, LOW);
//delay(5000);
}

/* void callback(String topic, byte* message, unsigned int length) {
Serial.print("Message arrived on topic: ");
Serial.print(topic);
Serial.print(". Message: ");
String messageTemp;

for (int i = 0; i < length; i++) {

```

```

        Serial.print((char)message[i]);
        messageTemp += (char)message[i];
    }
    Serial.println();

    // Feel free to add more if statements to control more GPIOs with MQTT

    // If a message is received on the topic room/lamp, you check if the
    message is either on or off. Turns the lamp GPIO according to the message
    if(topic=="room/lamp"){
        Serial.print("Changing Room lamp to ");
        if(messageTemp == "on"){
            digitalWrite(lamp, HIGH);
            Serial.print("On");
        }
        else if(messageTemp == "off"){
            digitalWrite(lamp, LOW);
            Serial.print("Off");
        }
    }
    Serial.println();
}
*/

void reconnect()
{
    // Loop until we're reconnected
    while (!mqttClient.connected())
    {
        Serial.print("MQTT reconnect...");

        // Attempt to connect
        if (mqttClient.connect(CLIENT_ID))
        {
            Serial.println("connected!");
            //mqttClient.subscribe("lab/sba");
        }
        else
        {
            Serial.print("failed, rc=");
            Serial.print(mqttClient.state());
            Serial.println(" try again in 5 s");

            // Wait 5 seconds before retrying
            delay(5000);
        }
    }
}

void callback(char* topic, byte* payload, unsigned int length) {
    char msgBuffer[20];
    // I am only using one ascii character as command, so do not need to take
    an entire word as payload
    // However, if you want to send full word commands, uncomment the next
    line and use for string comparison
    // payload[length]='\0';// terminate string with 0
    //String strPayload = String((char*)payload); // convert to string
    // Serial.println(strPayload);
}

```



```

//mqttClient.publish("lab/nb", "Payload received");

Serial.print("Message arrived [");
Serial.print(topic);
Serial.print("] "); //MQTT_BROKER
for (int i = 0; i < length; i++) {
    Serial.print((char)payload[i]);
}
Serial.println();
Serial.println(payload[0]);
String mystring(payload[0]);
// Examine only the first character of the message
if (payload[0] == 49) // Message "1" in ASCII (turn output
ON)
{
    //digitalWrite(LED_BUILTIN, HIGH); //
    //digitalWrite(relayPin, HIGH);
    digitalWrite(ledPin, HIGH);
} else if (payload[0] == 48) // Message "0" in ASCII (turn output
OFF)
{
    //digitalWrite(relayPin, LOW); //
    //digitalWrite(LED_BUILTIN, LOW);
    digitalWrite(ledPin, LOW);
} else if (payload[0] == 50)
{
    mqttClient.publish("home/br/nb/ip", ip.c_str()); // publish IP nr
} else {
    Serial.println("Unknown value");
//    mqttClient.publish("lab/nb", "Syntax Error");
    mqttClient.publish("lab/nb", "Unknown value");
}
}

/*void callback(char* topic, byte* payload, unsigned int length) {
    Serial.print("Message arrived [");
    Serial.print(topic);
    Serial.print("] ");
    for (int i=0;i<length;i++) {
        Serial.print((char)payload[i]);
    }
    Serial.println();

    mqttClient.publish("lab/nb", "Payload received");
}*/

//print any message received for subscribed topic
/*void callback(char* topic, byte* payload, unsigned int length) {

    mqttClient.publish("lab/nb", "Payload received");
    digitalWrite(ledPin, HIGH);
    delay(5000);
    Serial.print("Message arrived [");
    Serial.print(topic);
    Serial.print("] ");
    for (int i=0;i<length;i++) {
        Serial.print((char)payload[i]);
    }
    Serial.println();
}*/

```

## Appendix B

```
#include <XBee.h>
#include <dht11.h>
#include "binary.h"

XBeeWithCallbacks xbee;

dht11 DHT11;

#define DHT11PIN 8

//XBee xbee = XBee();
float ph = 7.5;
float DO = 5.0;

void setup() {
    // put your setup code here, to run once:

    Serial.begin(9600);
    xbee.setSerial(Serial);
    delay(1);

    pinMode(13, INPUT_PULLUP);
    pinMode(12, INPUT_PULLUP);
    pinMode(11, INPUT_PULLUP);
    pinMode(10, INPUT_PULLUP);
}

void sendPacket(float x, float y, float z) {
    // Prepare the Zigbee Transmit Request API packet
    ZBTxRequest txRequest;
    txRequest.setAddress64(0x0013a20041a7b0d3);

    // Allocate 13 payload bytes: 1 type byte plus three floats of 4 bytes
    each
    AllocBuffer<13> packet;

    // Packet type, pH, DO, temperature
    packet.append<uint8_t>(1);
    packet.append<float>(x);
    packet.append<float>(y);
    packet.append<float>(z);

    txRequest.setPayload(packet.head, packet.len());

    // And send it
    xbee.send(txRequest);
}

unsigned long last_tx_time = 0;

void loop() {
    xbee.loop();
}
```

```

int chk = DHT11.read(DHT11PIN);
int sensor_pH_higher = digitalRead(13);
int sensor_pH_lower = digitalRead(12);
int sensor_DO_higher = digitalRead(11);
int sensor_DO_lower = digitalRead(10);

if (sensor_pH_higher == LOW) {
    ph=ph+0.1;
}
if (sensor_pH_lower == LOW) {
    ph=ph-0.1;
}

if (sensor_DO_higher == LOW) {
    DO=DO+0.1;
}
if (sensor_DO_lower == LOW) {
    DO=DO-0.1;
}
delay(2500);

Serial.println(ph);
Serial.println(DO);
Serial.print("Temperature (C): ");
Serial.println((float)DHT11.temperature, 2);
float temp = ((float)DHT11.temperature);
Serial.println(temp);
// Send a packet every 10 seconds
if (millis() - last_tx_time > 10000) {
    sendPacket(ph,DO,temp);
    last_tx_time = millis();
}
}

```

## Appendix C

```
var timeStarted=global.get('timeStarted') || 0;
var cycling=global.get('cycling') || 0;
var now = Date.now();

if (msg.payload < 5 && cycling == 0) {
  //start cycling process
  msg.payload = '1' + ' loop1';
  global.set('cycling',1);
  // return [null,msg];
}

if (msg.payload < 5 && cycling == 1) {
  //start cycling process
  msg.payload = '1'+ ' loop2';
  //global.set('cycling',0);
  //return [null,msg];
}

if (msg.payload >= 5 && cycling == 1) {
  //start cycling process
  msg.payload = '1'+ ' loop3';
  // global.set('cycling',1);
  //return [null,msg];
}

if (msg.payload >= 5 && cycling == 0) {
  //start cycling process
  msg.payload = '0'+ ' loop4';
  // global.set('cycling',0);
  //return [null,msg];
}

if (cycling)
{
  if (now - timeStarted > 30000) {
    // you can just write function's name
    //timeStarted = timeStarted || now;
    timeStarted = now;
    global.set('timeStarted',now);
    global.set('cycling',0);
  }
  else {global.set('cycling',1);}
}
//msg.payload = msg.payload +
//              ' The time is ' +
//              now + ' ' +
//              timeStarted + '.';
//
//msg.payload = '0';
return [msg,msg];
```