IoT Fuzzy Logic Aquaponics Monitoring and Control Hardware Real-Time System

Adnan Shaout and Spencer G. Scott
The University of Michigan - Dearborn
The Electrical and Computer Engineering Department
Dearborn, MI
shaout@umich.edu; sgscott@umich.edu

Abstract—this paper presents a design for monitoring and controlling a fish tank and growing bed in an aquaponics ecosystem. Aquaponics is a growing field in which fish and plants are grown together and mutual benefit each other. Fuzzy logic is used to evaluate the inputs and automatically provide the proper output. The system will monitor water temperature, pH, air temperature, and luminance. The system will control a light, heater, and alarm. The Arduino Uno R3 board was selected to be the hardware interface for inputs/outputs. Selecting the Arduino was based on Matlab having a support toolbox to interface with the Arduino (ATMEGA8U2-MU) microcontroller. Updating of the input value and triggering twitter alerts was all done through using the free Thingspeak server tool that connects nicely with Matlab via a toolbox.

Index Terms—Aquaculture, Aquaponics, Ecosystems, Fuzzy logic, MATLAB, Real-time systems, IoT, Arduino

I. INTRODUCTION

This paper describes a hardware design for a fuzzy logic base aquaponics monitoring and control system. Aquaponics is the coexistence of fish and plants in an indoor ecosystem for mutual benefit. The fish tank water is pumped to the plant grow bed. The plants use the nutrients of the fish waste and return clean water to the aquarium[1][2][3].

The paper aims to design an aquaponics monitoring and control system that is accurate, low maintenance, low cost and convenient. The main goal of this paper was to create a system that removes the burden of upkeep from the user. The systems parameters needed to be checked daily on startup and weekly once the system is stable. This is an area were the automatic monitoring and even control of such a system can really benefit the user.

An aquaponics ecosystem is something that can take a lot of time to properly setup and needs to be continuously monitored in the early phases. Therefore a quick, inexpensive and reliable solution was the scope of this paper. In general once an aquaponics system is stable it seems to remain stable so an expensive and precise system is not needed. Fuzzy logic was selected for this paper because of the imprecise requirements and ease of implementation.

During the concept phase an aquaponics system was evaluated for what values would be important to monitor and what equipment might be needed to keep the tank and grow-bed in the recommended range. For a basic aquaponics system the values that needed to be monitored were considered. Table 1 shows the water temperature that a healthy fish tank should stay within[10].

<table>
<thead>
<tr>
<th>Species</th>
<th>Lower Avoidance</th>
<th>Optimum</th>
<th>Upper Avoidance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atlantic Salmon</td>
<td>45</td>
<td>50-62</td>
<td>N/A</td>
</tr>
<tr>
<td>Black Crappie</td>
<td>60</td>
<td>71</td>
<td>75</td>
</tr>
<tr>
<td>Bluegills</td>
<td>58</td>
<td>69-72</td>
<td>80</td>
</tr>
<tr>
<td>Blue Catfish</td>
<td>N/A</td>
<td>77-82</td>
<td>N/A</td>
</tr>
<tr>
<td>Brook Trout</td>
<td>44</td>
<td>58</td>
<td>70</td>
</tr>
<tr>
<td>Brown Bullhead</td>
<td>65</td>
<td>78-82</td>
<td>85</td>
</tr>
<tr>
<td>Brown Trout</td>
<td>44</td>
<td>56-66</td>
<td>75</td>
</tr>
<tr>
<td>Channel Catfish</td>
<td>55</td>
<td>82-89</td>
<td>90+</td>
</tr>
<tr>
<td>Chinook Salmon</td>
<td>44</td>
<td>55</td>
<td>60</td>
</tr>
<tr>
<td>Coho Salmon</td>
<td>44</td>
<td>55</td>
<td>60</td>
</tr>
<tr>
<td>Flathead Catfish</td>
<td>81</td>
<td>85</td>
<td>90</td>
</tr>
<tr>
<td>Lake Trout</td>
<td>40</td>
<td>50-55</td>
<td>60</td>
</tr>
<tr>
<td>Largemouth Bass</td>
<td>50</td>
<td>65-75</td>
<td>85</td>
</tr>
<tr>
<td>MuskelBunge</td>
<td>55</td>
<td>63-67</td>
<td>78</td>
</tr>
<tr>
<td>Northern Pike</td>
<td>55</td>
<td>65</td>
<td>74</td>
</tr>
<tr>
<td>Pumpkansseed</td>
<td>N/A</td>
<td>81</td>
<td>N/A</td>
</tr>
<tr>
<td>Rainbow Trout</td>
<td>44</td>
<td>61</td>
<td>75</td>
</tr>
<tr>
<td>Rock Bass</td>
<td>N/A</td>
<td>70</td>
<td>N/A</td>
</tr>
<tr>
<td>Smallmouth Bass</td>
<td>60</td>
<td>65-70</td>
<td>73</td>
</tr>
<tr>
<td>Steelhead Trout</td>
<td>38</td>
<td>55-60</td>
<td>N/A</td>
</tr>
<tr>
<td>Sunfish</td>
<td>50</td>
<td>58</td>
<td>68</td>
</tr>
<tr>
<td>Walleye</td>
<td>50</td>
<td>64-70</td>
<td>76</td>
</tr>
<tr>
<td>White Bass</td>
<td>55</td>
<td>65-70</td>
<td>80</td>
</tr>
<tr>
<td>White Catfish</td>
<td>N/A</td>
<td>80-85</td>
<td>N/A</td>
</tr>
<tr>
<td>White Crappie</td>
<td>N/A</td>
<td>61</td>
<td>N/A</td>
</tr>
<tr>
<td>White Perch</td>
<td>N/A</td>
<td>89</td>
<td>N/A</td>
</tr>
<tr>
<td>Yellow Perch</td>
<td>58</td>
<td>68</td>
<td>75</td>
</tr>
</tbody>
</table>

While, Table 2 shows the optimal temperature for some of the common crops[9].

<table>
<thead>
<tr>
<th>Temperature</th>
<th>Cool Season: broccoli, cabbage, and cauliflower</th>
<th>Warm Season: tomato, pepper, squash and melons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Germination</td>
<td>40°F to 90°F, 80°F optimum</td>
<td>50°F to 100°F, 80°F optimum</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Growth</th>
<th>Daytime: 65°F - 80°F preferred 40°F minimum</th>
<th>Daytime: 86°F optimum 60°F minimum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nighttime</td>
<td>&gt;32°F, tender</td>
<td>A week below 55°F will stunt plant, reducing yields.</td>
</tr>
</tbody>
</table>
Once the idea was created different options were research to determine the feasibility of the paper. The paper was deemed difficult but feasible with the help of tools such as MATLAB (including Fuzzy Logic Toolbox) and Simulink. A state of the art literature review was performed and is shown in section III of this paper.

To have a successful aquaponics system both the fish tank and plants need to be monitored.

The fish tank will have to stay within the ideal temperature range which is around 75 degrees Fahrenheit. Additionally, the pH level of the tank should be close to 7. There are other parameters that can be measured but temperature and pH are the most crucial for the fish to survive.

The grow-bed crops have a wider range of temperatures depending on the specific crop. However, after doing more research the amount of light crops was found to be very important. Figure 1 shows some the recommended light levels for common vegetables and other common plants [8].

<table>
<thead>
<tr>
<th>Situation</th>
<th>Foot Candles</th>
<th>Crops</th>
</tr>
</thead>
<tbody>
<tr>
<td>summer full sun</td>
<td>12,000</td>
<td>outdoor crops</td>
</tr>
<tr>
<td>bright overcast</td>
<td>6,000</td>
<td>moderate light house plants</td>
</tr>
<tr>
<td>heavy overcast</td>
<td>1,000</td>
<td>low light house plants</td>
</tr>
<tr>
<td>home interior</td>
<td>50</td>
<td></td>
</tr>
</tbody>
</table>

![Figure 1. Ideal sunlight conditions for different plant types](image)

The light and temperature were chosen in this design since they are deemed most important parameters for healthy plant life. The following requirements were determined based on the importance of different parameters for the aquaponics system [5] [6]:

**FUNCTIONAL REQUIREMENTS:**

1. The system shall monitor water temperature, lumens, pH, and air temperature every 25 seconds
2. The system shall control lights, heater, and an alarm and respond within 1 minute
3. The system can be monitored from a GUI
4. The system should be simulated and tested in a virtual environment
5. The alarm will be activated if pH varies too far away from 6.9 (roughly 6.6-7.6)
6. The alarm will not sound for temperature or lumens but the control will automatically turn on the corresponding output (heater or lights respectively)

**NON-FUNCTIONAL REQUIREMENTS:**

1. MATLAB’s fuzzy logic toolbox will be used to take the inputs of the system and determine the output.
2. The system will be modeled in MATLAB using Simulink
3. The system will be simulated and tested in MATLAB with a signal generator

**II. DESIGN PHASE**

The design phase was spent finding a simplistic and reliable solution. Matlab was selected to be the interface and programming language because of the proven reliability. The Arduino Uno R3 board was selected to be the hardware interface for inputs/outputs. Selecting the Arduino was based on Matlab having a support toolbox to interface with the Arduino (ATMEGA8U2-MU) microcontroller. The Arduino was found to have drawbacks because the programming language is abstracted to the high level. Therefore, Matlab was selected to be the programming language which makes evaluating and processing data much more efficient. The high level design is shown below in Figure 2 was creating using Matlab Stateflow toolbox.

![Figure 2. High Level Design of the aquaponics system](image)

To elaborate on the above diagram the system is a firm real-time system. This is because missing a few deadlines will not cause any issues. However, If many deadlines are missed this could lead to the fish and plants dying. The timing
requirement of receiving new input sensor values every 25 seconds will be handled by a timer in Matlab. If any of the outputs meet the requirements then the respective Arduino outputs are driven high to trigger the relay shield to turn on the light or heater. The relay response time is 1ms [11] and the time to trigger an Arduino output is safely less than 1 second so the 1 minute response time is easily met. Also, the Thingspeak server update time for the alarm is almost instantaneously (less than 1 second).

Thingspeak is a free to use server tool that connects nicely with Matlab via a toolbox. Updating of the input value and triggering twitter alerts was all done in the Thingspeak server. The overall architecture is designed with scalability in mind. More sensors and controls can be added with limited extra effort.

A. Programming Phase

The programming phase took some tweaking because the MATLAB interface for Arduino and Thingspeak was not very flexible. To get around the inflexibility of these tools, special m-scripts and functions were created. The system is interrupt driven and the calling diagram in shown below in Figure 3.

![Function calling diagram for aquaponics system](image)

Figure 3. Function calling diagram for aquaponics system

Basically, the main script aquaponics_arduino.m creates a timer object which keeps the timing of the system. The timer has a timerStart function responsible for connecting to and setting up the Arduino. The timer also calls readParameters every 25 seconds. The readParameters function reads the values from the sensors and sends them to fisAquaponics to be evaluated by the fuzzy inference system. The outputs then perform the action matching the evaluation in the fuzzy system. For example if the light was evaluated to be greater than 0.5 then the Arduino would trigger digital pin 8. The relay control shield would then turn on the light output. The inputs and outputs from the system are then stored on the Thingspeak server via the tsWrite functions. Overall the system can be configured for different timing periods greater than 15 seconds (Thingspeak server restriction). Also, the number of times the read of sensor and write to the server is executed can also be configured anywhere from 0 to infinity.

Another major portion of the implementation was trying to tune the fuzzy inference system. Intuition was used to get the starting values and was tuned to achieve the required results. Figure 4 shows the fuzzy inference system (FIS) designed using the Matlab Fuzzy Logic toolbox.

![Fuzzy inference system inputs and outputs](image)

Figure 4. Fuzzy inference system inputs and outputs

Each input and output then needed to have membership functions created and tuned. Figure 5 shows the final input membership functions (linguistic values) for all input variables.

![Membership function plots](image)
Table 3 shows the goal of each of the input membership functions shown in Figure 5.

<table>
<thead>
<tr>
<th>TABLE 3. IDEAL FIS INPUT PARAMETERS</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>WATER TEMPERATURE</td>
<td>Roughly 75°F</td>
</tr>
<tr>
<td>pH</td>
<td>Between 6.2 and 7.2</td>
</tr>
<tr>
<td>AIR TEMPERATURE</td>
<td>Between 60°F and 90°F</td>
</tr>
<tr>
<td>ILLUMINANCE</td>
<td>Between 200 to 700 lux</td>
</tr>
</tbody>
</table>

As long as the values are close to the above ranges the system will be stable. Water temperature and pH are slightly more important to regularly monitor because of their ability to change quicker. Also, it is really hard to get too much illuminance but it is possible to dry out the plants.

All of the controls of the system are either on or off however this is evaluated based on fuzzy logic. If any of the values are above 0.5 then the respective output is turned on. See the membership functions for the outputs below in Figure 6 for more details.

Looking at the output membership function it is clear that the evaluations are done in a more simplistic manner. pH level is the sole controller of the alarm. Lux is the sole controller of the light. However, waterTemp and airTemp determine the heater output. The rules that link these variables are shown below in Figure 7.

In the programming phases many iteration were attempted because of some limitations with Arduino, Matlab, Simulink, and Thingspeak. Many obstacles had to be overcome because of the hardware integration with such tools. However, the
current paper implementation provides the tools and code necessary to make adding more functionality easily.

B. Test Phase

The system was relatively easy to test because it did not contain strict timing requirements. The system log from Matlab prints out the exact time the variable were updated and the Thingspeak server also keeps a log of the updated variables and their associated times. Figure 8 shows a Matlab output of the running program.

```
>>aquaponics_arduino
StartFcn executed 16-Aug-2016 00:35:09.081
TimerFcn executed 16-Aug-2016 00:35:09.081
ans =
77.4500 6.9400 60.2380 201.7052 0.3456 0.3003 0.6997
TimerFcn executed 16-Aug-2016 00:35:34.081
ans =
77.4500 6.3200 75.8857 190.0468 0.2607 0.3094 0.6997
TimerFcn executed 16-Aug-2016 00:35:59.081
ans =
77.4500 7.0600 63.1300 320.8919 0.2825 0.3026 0.6997
TimerFcn executed 16-Aug-2016 00:36:24.082
ans =
77.4500 6.5300 73.0816 360.1465 0.2607 0.3003 0.6997
TimerFcn executed 16-Aug-2016 00:36:49.081
ans =
77.4500 7.5000 69.0108 87.7196 0.2607 0.3689 0.6487
```

Figure 8. Matlab output of running aquaponics system

The Matlab output above is displaying the timing of reading the input variable and computing the output variables. The ‘ans’ variable contains the 4 inputs and 3 output variables. The ‘TimerFcn’ event which is function readParameters is executed almost exactly every 25 seconds.

```
4:37:39  0:00:25  77.45  7.65  70.76685  498.2606  0.699703
4:38:04  0:00:25  77.45  6.16  68.85357  97.99375  0.640204
4:38:29  0:00:25  77.45  7.92  60.09268  398.7097  0.699703
4:38:54  0:00:25  77.45  7.63  77.37389  87.99613  0.648432
```

Figure 9. Writes to the Thingspeak server

Figure 9 demonstrates how the timing of writing to the Thingspeak server also happens consistently every 25 seconds. Some of the fields were removed from the chart to make it more readable but the change in time is the important field to note.

The fuzzy system was also tested to ensure the inputs created the expected outputs. Unit testing was performed on the FIS created in Matlab. The testing was done in Simulink with the help of the Signal Builder, Fuzzy Logic Controller, and Scope Block as shown in Figure 10.

```
Figure 10. Simulink unit test of FIS
```

The system was tested given values that extended beyond the normal range of the inputs. Figure 11 shows the input values from the Signal Builder block that were simulated and the outputs response captured on the Scope.

```
Figure 11. Output from Simulink simulation
```

<table>
<thead>
<tr>
<th>Time</th>
<th>Δ Time</th>
<th>field1</th>
<th>field2</th>
<th>field3</th>
<th>...</th>
<th>field7</th>
</tr>
</thead>
<tbody>
<tr>
<td>4:35:59</td>
<td></td>
<td>77.45</td>
<td>7.06</td>
<td>63.1297</td>
<td>320.8919</td>
<td>0.699703</td>
</tr>
<tr>
<td>4:36:24</td>
<td>0:00:25</td>
<td>77.45</td>
<td>6.53</td>
<td>73.08158</td>
<td>360.1465</td>
<td>0.699703</td>
</tr>
<tr>
<td>4:36:49</td>
<td>0:00:25</td>
<td>77.45</td>
<td>7.5</td>
<td>69.01083</td>
<td>87.71962</td>
<td>0.648653</td>
</tr>
<tr>
<td>4:37:14</td>
<td>0:00:25</td>
<td>77.45</td>
<td>6.46</td>
<td>78.26675</td>
<td>118.5701</td>
<td>0.622229</td>
</tr>
</tbody>
</table>
The system is working as expected when comparing the inputs to the outputs. When the water temperature or air temperature is too cold the heater is on. When they are both in a good temperature range (from 4 to 6 seconds) the heater turns off. As for the alarm it will turn on if the pH is too low or too high. The alarm starts on because the pH is too low then turns off around 4.5 seconds. The alarm then turns back on at ~5.2 seconds because the pH is too high. The light follows a similar pattern to the heater. When the lux level is about 800 at 1.6 seconds into the simulation the light turns off. This method of testing was used to fine turn the FIS.

### III. STATE OF THE ART COMPARISON

A number of papers and commercial products were evaluated to compare the important benchmarks that should be considered and features as shown in Table 4. It was quite difficult to find any paper specifically on aquaponics but there were more papers regarding aquariums and greenhouses. One of the main reasons this topic was selected for a paper was the lack of hardware research done in this area.

<table>
<thead>
<tr>
<th>Reference #</th>
<th>[this paper]</th>
<th>[12]</th>
<th>[2]</th>
<th>[3]</th>
<th>[13]</th>
<th>[7]</th>
<th>[4]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost</td>
<td>$77</td>
<td>$900</td>
<td>~$80</td>
<td>~$458</td>
<td>~$292</td>
<td>~80</td>
<td>~14.9</td>
</tr>
</tbody>
</table>

**Inputs**
- Water Temp: ✔ ✔ ✔ ✔-
- pH: ✔ ✔ ✔ ✔-
- Air Temp: ✔ ✔ ✔ ✔-
- Lux: ✔ ✔ ✔ ✔-
- DO: ✔ ✔
- Water level: ✔ ✔
- Humidity: ✔ ✔ ✔
- EC/TDS: ✔

**Outputs**
- Light: ✔ ✔ ✔ ✔ ✔
- Heater: ✔ ✔ ✔ ✔
- Alarm: ✔ ✔ ✔ ✔-
- Pump: ✔ ✔ ✔ ✔-

**Features**
- IoT network: ✔ ✔ ✔ ✔-
- Real-time: ✔ ✔ ✔ ✔-
- Max # of I/O: 67 12 20 N/A 67 20 24
- Upload freq.: 25s 15m N/A N/A 10m N/A N/A

| Commercial or Research | R | C | R | R | R | R | R |

~ cost is approximated based on parts list

The commercial application outline by reference [12] is called Osmobot and is probably the closest in comparison to available features but is much more expensive than the design outlined in this paper.

The research papers [2][3] both are more focused on the entire system than just the monitoring part. Exact values were not given for cost but the parts list was, so the cost was approximated.

Research paper [13] had very similar inputs and outputs but the outputs were manually controlled not automatic. Also, the system ended up being more expensive because of the pump, webcam, and the entire supporting sensors. However, an interesting Internet of Things (IoT) connectively board was used called the WRTnode. It is only about $20 and seemed relatively easy to get up and running.

The Pi Doctor Paper [7] focuses on the monitoring of pH by using a special infrared camera. The camera monitors the normalized difference vegetation index (NDVI). Plants with a higher NDVI have a higher pH level. This is a very interesting alternative way to monitoring pH.

An IoT Greenhouse monitoring and control system is outline using a ZigBee coordinator and many nodes [4]. This gives a strategy for managing a much larger system than just a smaller indoor aquaponics paper.

### IV. HARDWARE IMPLEMENTATION

The hardware implementation efforts for this paper will be described in detail in the following sections.

#### A. Implementation of paper on a hardware platform

The paper was implemented using the Arduino UNO R3 hardware platform. The Arduino is made up of a small microcontroller with twenty input and output pins. This was a major effort to get all of the sensors interfacing properly with the board. Figure 12 shows a picture of the actual hardware and sensors board used in the paper.
The software implementation was done in MATLAB with the help of the “Arduino Support from MATLAB” which is a hardware support package add on that is free to download. This package allows connecting to the Arduino and controlling the inputs and outputs from MATLAB scripts without having to use the Arduino IDE. Here is how the Arduino can be initialized for communication:

```matlab
a = arduino('COM22', 'Uno', 'Libraries', 'PaulStoffregen/OneWire');
```

This is used in the timerStart.m MATLAB function. The parameters are as follows:
- ‘COM22’ – port number of the Arduino
- ‘Uno’ – hardware type of Arduino
- ‘Libraries’ – specifics the next parameter is a library
- ‘PaulStoffregen/OneWire’ – name of library included in paper

Here is how the water temperature input was read from the Arduino:

```matlab
data = read(sensor, addr, 9);
```

The read command is part of the OneWire library which calls the readDigitalPin command. This command converts the digital signal into a floating point value. This is used in the readParameters.m MATLAB function. The parameters are as follows:
- sensor – Arduino pin connected to the sensor
- addr – address of the OneWire device on the bus
- 9 – this is the number of data bytes to read

Here is how a write command is performed on the light and heater:

```matlab
if light_ctrl > 0.5
    writeDigitalPin(a, 'D8', 1); % write digital pin 8 high
elseif heater_ctrl > 0.5
    writeDigitalPin(a, 'D9', 1); % write digital pin 9 high
end
```

The above commands are used to set the pins D8 and/or D9 high if it is determined that one of the outputs should be on. Below in Table 5 are some other common MATLAB commands to interface with the Arduino.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>readDigitalPin</td>
<td>Read data from digital pin on Arduino hardware</td>
</tr>
<tr>
<td>writeDigitalPin</td>
<td>Write to digital pin on Arduino hardware</td>
</tr>
<tr>
<td>writePWMVoltage</td>
<td>Write digital pin PWM voltage value</td>
</tr>
<tr>
<td>writePWMDutyCycle</td>
<td>Set digital pin PWM duty cycle</td>
</tr>
<tr>
<td>playTone</td>
<td>Play tone on piezo speaker using digital pin</td>
</tr>
<tr>
<td>readVoltage</td>
<td>Read Arduino analog pin voltage</td>
</tr>
</tbody>
</table>

### B. IoT things connectivity through the Thingspeak server

This allows the system to be monitored from anywhere with an internet connection. The following internet link can be used to monitor the current values of the system:

[https://thingspeak.com/channels/140038](https://thingspeak.com/channels/140038)

Figure 13 shows a display of the monitored inputs with respect to time.

**Figure 13. Inputs of system on IoT server**

The ThingSpeak server was updated using the following MATLAB command:

```matlab
data = [wTemp pH aTemp lux light_ctrl heater_ctrl alarm_ctrl];
thingSpeakWrite(140038, data); % write data to channel
```

thingSpeakWrite has the following parameters:
- 140038 – Thingspeak server ID
- data – the array of values to be sent to the server.

This is used in the tsWrite.m MATLAB function.

### C. Alarm to send Twitter message when pH levels are out of range

When the pH levels were out of range Twitter was setup to receive a notification. This was done through the ThingSpeak server. Figure 14 shows the required options to setup a Twitter alert.

**Figure 14. Required options to setup a Twitter alert**

When the pH levels were out of range Twitter was setup to receive a notification. This was done through the ThingSpeak server. Figure 14 shows the required options to setup a Twitter alert.
The system was configured in a similar manner to react to the light and heater output. Figure 15 shows an example "tweet" that gives notification to the user on system outputs.

![Figure 15. Twitter alert](image)

**D. Setup a fish tank**

A fish tank was setup to properly test and demo the design in this paper. Figure 16 shows a picture of the fish tank used in this paper.

![Figure 16. Fish Tank](image)

**E. GUI using Google Gauges on the ThingSpeak server**

Finally, a Google Gauge visualization was implemented to display the last input to the ThingSpeak server. Figure 17 shows the mentioned Google Gauges.

![Figure 17. Google Gauges of inputs](image)

The Google Gauge setup for water temperature has the following code pieces:

**HTML**

```html
<html>
<head>
<title>Google Gauge - ThingSpeak</title>
</head>
<body>
<div id="container">
  <div id="inner">
    <div id="gauge_div"></div>
  </div>
</div>
</body>
</html>
```

**CSS**

```css
body { background-color: #ddd; }
#container { height: 100%; width: 100%; display: table; }
#inner { vertical-align: middle; display: table-cell; }
#gauge_div { width: 120px; margin: 0 auto; }
```

**JavaScript**

```javascript
var channel_id = 140038;
var api_key = 'KATI/0XFEDHOV0IM';
```
The above code can be entered into the ThingSpeak server to display real-time data of the input for water temperature every 25 seconds. The gauge will update every 15 seconds.

A similar method can be used to display the other values. To do this, the following changes should be made:

1. The variable `max_gauge_value` and `gauge_name` should be updated to the new parameter value and name respectively.

```javascript
varmax_gauge_value = 100;
var gauge_name = 'Water Temp';

var chart, charts, data;
googoloading('visualization', 1, [packages: ['gauge']]);
googoloadingCallback((int)Chart);
function displayData(point) {
  data.setValue(0, 0, gauge_name);
  data.setValue(0, 1, point);
  chart.draw(data, options);
}
function loadData() {
  var p;
  if (getJSON('https://api.thingspeak.com/channels/' + channel_id + '/feedlastkey=' + api_key)) {
    p = data.field1;
displayData(p);
  }
}
function initChart() {
  var chart = new google.visualization.Gauge(document.getElementById('gauge_div'));
data.addRows(1);
data.addColumn('number', 'Value');
data.addColumn('string', 'Label');

  options = {width: 120, height: 120, redFrom: 90, redTo: 100, yellowFrom: 75, yellowTo: 90, minorTicks: 5};

  chart.draw(data, options);
}
</script>
```

2. The variable `p` should be updated to the corresponding field number. For example “p=data.field2;” for air temperature.

```javascript
var p = data.field1;
```

3. The ranges for danger should be updated to match the input danger ranges. The `redFrom` and `redTo` is the danger range.

```javascript
options = {width: 120, height: 120, redFrom: 90, redTo: 100, yellowFrom: 75, yellowTo: 90, minorTicks: 5};
```

Other options can be configured to change the look of the gauges but the above are the only ones used in this paper.

V. CONCLUSION

The paper presents a low cost aquaponics monitoring and control system. The system will monitor water temperature, pH, air temperature, and luminance. The system will control a light, heater, and alarm. The Arduino Uno R3 board was selected to be the hardware interface for inputs/outputs. Selecting the Arduino was based on Matlab having a support toolbox to interface with the Arduino (ATMEGA8U2-MU) microcontroller. Updating of the input value and triggering twitter alerts was all done through using the free Thingspeak server tool that connects nicely with Matlab via a toolbox. The aquaponics system was demonstrated and proven to work effectively and reacting to instabilities within the aquaponics ecosystem. The design presented is completely scalable to include more or less I/O control depending on an individual’s needs.

REFERENCES

doi: 10.1109/ICCE-TW.2015.7216894


doi: 10.1109/SusTech.2014.7046233


