Armoniacs: Automatized Aquaponics with Arduino

Guilherme Alcarde Gallo Christian Esteve Rothenberg

September 18, 2016

Contents

1 Introduction

1.1 Motivation

Basically, the automation purpose is to save labor and add more security, precision and velocity to tasks that formerly were made by humans or other animals.

If someone wonder about the reason for automating tasks, this person needs to pay attention about living in modern civilizations. When one walks on the sidewalk, one can see vehicles parked on the road, these are automated objects, because saves the labor of walking to distant places – or making animals to walk for us – via the electronic management of several components, such as acceleration, fuel measurement, injection, breaking with ABS technology etc. But not only cars are automated. The microwave, airplanes and elevators are automated objects which are used frequently.

Generally, an automation project is composed by a microcontroller, electronic devices and a variable number of sensors. The former is the main component, which is responsible to manage the system by reading the input from the latter, and then process what to do, so it can send signals to electronic devices to accomplish the system's goal. This processing is made with programming. A programmer can install the code inside the microcontroller, which will execute the program when turned on. So it is presupposed that the programmer will align the program with the system's intention.

1.1.1 Context

When someone builds an Aquaponics system, one needs to manage it everyday, since there is a plenty of work to do for maintaining the system work, due the dependency of living beings on the system. So it is a time wasting management, with lots of repetitive tasks, which one can be understood as periodic events. Other tasks needs precision, such as water leveling and pH balancing, that feature can be achieved with automation.

Some of these tasks can be reliably automated, together with others that involves problems that may occur during the system's growth. For example, the fish feeding is a periodic event, but the water's pH control is not a periodic event, instead it is classified as an adverse event.

Basically this work is trying to decrease human intervention in an Aquaponics system via automatic management of adverse and periodic events, in order to reduce the time wasted to take care of the system, as well as to make the management more reliable.

The figure 1 demonstrate an example of a functional Indoor Growing System applying Aquaponics.

Reasons to build an Aquaponics system A great amount of the published projects has a commercial goal: to make an efficient and small-sized system that can afford to produce organic products in a large scale.

On other hand, in the [8] there is an effort to address the sustainability aspect of the Aquaponics. This aspect stands for making a low and efficient nutrient input into

the system and making a minimal environment footprint.

Differences between Hidroponics and Aquaponics The Hidroponics is a system that uses a nutritive water to feed the plants. It is a inorganic system, where the addition of inorganic nutrients is needed and the main live component is the plant. On the other hand, the Aquaponics is a partly-organic system cite, where the fish is added to the system, and its waste, the ammonia, serves as a nutrient to the plants.

The great advantage of the Hidroponics over the Aquaponics is that the last may have some issues with human diseases, like the presence of snails with parasites in the fish tank or some water-borne disease. [18].

Figure 1: An example of a built Aquaponics system. The plant medium is above and at the right of the electronic pump. At the bottom is the fish tank. [9]

1.2 Challenges

1.2.1 Without automation

To build an Aquaponics system, one needs to be precise and cautious when one makes the design of the project, since there are lots of requirements that need to be respected in order to make the Aquaponics work. The most important requirements are strongly connected with equilibrium, because we are dealing with a biological system. For example, the proportion between the fish tank and the plant's grow bed tank are vital to optimize. If one gets the wrong proportion, the system won't work, both the fish and the plants will die precociously [12].

Another problem is to distribute the byproducts of each medium to another one, the main byproduct of the fish tank is ammonia, if the concentration of the latter

increases, the fish will die by high toxicity, hence this ammonia has to be delivered to the bacteria, who will digest this substance via the nitrogen cycle, producing nitrate.

As the final product are comestibles, there is another concern: food toxicity. One cannot use components made with materials that can cause water contamination, because the fish and the plant will be contaminated as well. So there is the need to use non-toxic elements in the system. Besides there is also a high concern with the water source, the water cannot be infected with pathogens that causes diseases, like amoeba.

Lots of research have been done about the biological aspect of the Aquaponics. In [8], the authors show high complexity problems involving mechanisms to achieve pH equilibrium for optimizing the quality of life for the fish, plants and nitro-bacteria, since each living component of the system lives well in a certain pH-Range. So there is a challenge to separate the pH level by region. Despite of the requirement R2 treats the pH level of all mediums monolithically, the dedicated controlling of each medium's pH level is not treated in this work, for the sake of simplicity.

1.2.2 With automation

In order to automate an Aquaponics system, one must take care with microelectronic components, they have to be protected from water and humidity.

Second, there is the need to use relays to add high current-drain devices, like the submersible water pump, which is essential to the system. If this advise is ignored, the microcontroller will be damaged.

1.3 Project Objectives and Scope

Design the proof of concept of a system that is capable to automatically control and optimize the water cycle and its quality of an Aquaponics system via microcontroller.

What is in the scope of this proof of concept:

- Automatized Water Cycling for exchange water between tanks
- Water pH Level Automatic Control

What is out of scope:

- Automatic Fish Feeding
- Aquaponics structural design which doesn't regard automation, such as: tanks, pipes etc
- Root Cloging prevention

2 Theoretical Background and Related Work

2.1 Background

An extensive research has been made aiming at some basic Aquaponics knowledge and previous Aquaponics automation projects. The idea for creating a period water cycle has came from [14], where a simple project is proposed and an Arduino only automates this very feature of the system. But the system design of the former project with 3 tanks and 1 second water cycle will not be applied, the [11] uses a simpler approach with only 2 tanks and a periodic water cycle, with the difference that the latter's period is much longer.

There are some projects that uses multiples microcontrollers, like [7], which uses an Arduino Mega with Raspberry PI. For the sake of simplification, this project focuses on using only one microcontroller.

2.2 Technologies

2.2.1 Arduino

Arduino is an open-source prototyping platform that provides general purpose input/output (GPIO) ports to be controlled by a programmable microprocessor [3]. The GPIO feature allows the platform to control several output like: Light-Emitting Diodes (LEDs) brightness, Motors velocity and even publish something online. Some input control examples are: read a sensor value, detect a touch on a touch-enabled screen, read an email and detect button interactions.

To program the microcontroller within the Arduino, only a USB cable and a program uploader is needed, the latter is provided by the Arduino IDE, which is an open-source software. This interface could be used for direct serial communication between Computer and the board.

There are shields which are hardware expansions to the default platform. Shields can be connected to the main Arduino board to grant new features, such as Ethernet communication.

This platform is massively used in various types of electronic projects — such as 3D printing, IoT, robotics and embedded systems — since it is an inexpensive hardware with a participative open-source community, and lots of contributed tutorials and libraries available. Besides, the Arduino IDE runs in most computers, because is compatible with Linux, MacOS and Windows.

The Arduino version used in this project is the Arduino UNO Revision 3. The table 1 shows the main characteristics from this board.

2.2.2 Sensors

The pH probe is an equipment that can be submerged into the water to measure its pH level. It is necessary for this project as a required component of R2.

It is difficult to find the pH Sensor individually, the entire probe-kit is sold. The pH probe shown in the figure 2 is originated from the DFRobot factory. To extract the

Table 1: Main Arduino UNO specifications [1]

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P)of which 0.5 KB used by bootloader
Clock Speed	16 MHz

Figure 2: pH Probe at the top, protoboard interface at the bottom left and the USB interface at the bottom right [5]

actual pH level from the input voltage, an example code was provided, the reasoning follows: take 10 measurements, sort them, take the 6 values from the center to remove noisy values, average the extract values and apply the formula 1.

$$millivolts = \frac{averageVoltageValue \times 5}{1024 \times 6} \tag{1}$$

$$pH = 3.5 \times millivolts \tag{2}$$

2.3 Related work

There are some automated Aquaponics projects available on the Internet, but most of them doesn't have a good documentation. So it has been needed to grab parts of information among every material found on Internet. One of the best sources found was from a hackaday's post from [7], which describes with decent detail how they achieved the construction of their Arduino-powered Aquaponics system.

The solution for the requirement of periodic water cycle R1 is based on the cstewart000's Instructable [14], where the objective is to make a very simple automated system, with only R1 as case use. Although this work shows the recommended time proportion between water cycling activation and deactivation of 15 minutes per hour, as well as a simple code.

Additionally, the water pH Control feature R2 is extracted from Simonson's solution [6], which presents the idea of using peristaltic pumps to regulate the pH level.

3 System Proposal

3.1 Approach

This work has been implemented with an adaptation of V-model development process [1]. The original version of this process is known as a variation of the waterfall process, with several phases that resembles the latter, but the their displacement is slightly modified, emphasising on the coding phase.

An appealing feature of the V-model is that every phase, before the coding one, has a 1:1 strong relationship with the phases that occurs after the coding, as seen in figure 3, reinforcing the mapping of the theoretical part of the project with the practical one. Moreover, in comparison with Agile models, there is much more documentation in a project using V-model, and this feature is crucial for this work.

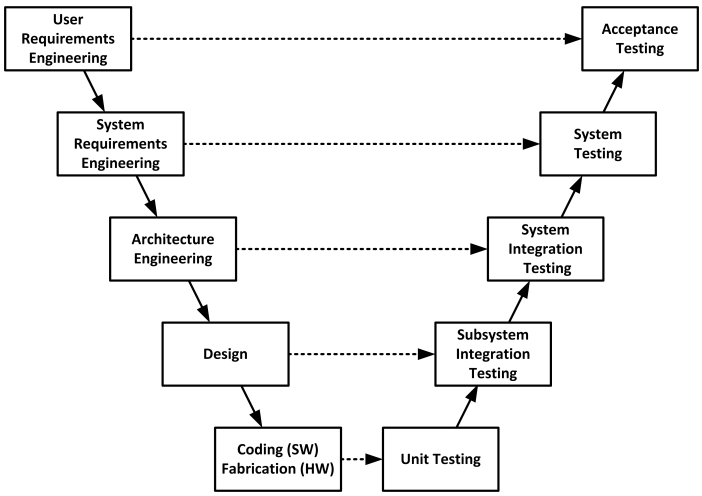

Figure 3: A V-model flow diagram. [17]

The adaptation made in this project was to divide the V-model into iterations separated by the use cases. That is, for every use case, there are the execution of every phase of V-model, as if the product would have only one use case.

[1] Described in detail here: [13]

After some research about the flaws of V-model, a variation of the V-model has been found and it shares the fundamental difference between this work's approach and V-model: the Dual V-model [4]. The shared idea is: divide and conquer V-model, by making subsystems that follows V-model, so, when these subsystems are finished, they can be integrated to create the final product.

- **Requirements Analysis** Sections 3.2 and 3.3
- **Specification / System Design** Section 3.4
- **Architectural Design / System Architecture** Section 3.5
- **Detail Design / Module Design** Section 4
- **Unit Testing** Section 5.2.2
- **Integration Testing** Section 5.2.2
- **System Testing** Section 5.2.2
- **User Acceptance Testing** Not used, because we are building a proof of concept.

In order to demonstrate the proof of concept without any physical building, there is a video with a Graphical User Interface in the section 5.2.2, which the user can simulate how the system responds to the undesired pH values in the tank. This video is made based on the IP Capture library for Processing [10], that makes feasible to render life video from a IP camera, hence it is possible to watch what happens with the Arduino in real-time when one changes a sensor value.

3.2 Use Cases

Water cycle control	**Details**
Actors:	Microcontroller
Preconditions:	• The Microcontroller must be installed in the system • A proper relay must be installed in the system • The external power supply and the water pump must be compatible
Postconditions:	The fish tank's water should be pumped to the vegetable media each quarter of an hour.

Main Success Scenario:

1. The Microcontroller executes a run cycle of 25% in a 1 hour period.
2. The circuit sends this signal to a relay.
3. The relay activates the water pump at a 15 minutes per hour rate with the adequate voltage.

Requirements:	• R1

Keep water tank's pH level in 6-7 range	**Details**
Actors:	• Microcontroller • Peristaltic Water Pumps • pH Sensors
Preconditions:	• The Microcontroller, pH sensors and the peristaltic pumps must be installed in the system • A proper relay must be installed in the system OR a power transistor to supply 12V DC • The external power supply and the water pump must be compatible
Postconditions:	The water tank's pH level is between 6 and 7.

Main Success Scenario:

1. The Microcontroller checks the pH sensor.
2. The pH output is between 6 and 7.

Extensions:

2.a Lower pH values in output:

1. The circuit sends this signal to start the high pH relay.
2. The relay distributes the needed current to the high pH's peristaltic pump.
3. High pH water is blended with the water tank's, until the latter gets into the normal pH range.
4. The circuit sends this signal to stop the high pH relay.

2.a Higher pH values in output:

1. The circuit sends this signal to start the low pH relay.
2. The relay distributes the needed current to the low pH's peristaltic pump.
3. High pH water is blended with the water tank's, until the latter gets into the normal pH range.
4. The circuit sends this signal to stop the low pH relay.

Special Requirements: • R2

3.2.1 Summary table

Table 4: Requirements table

Use Case	Requirement	References
Optimize Water Cycle	R1	
Keep pH Level optimum	R2	[15]

3.3 Requirements

R1 Optimize the water cycle period between fish tank and plants medium.

R2 Keep the pH Level around the 6-7 range.

3.4 System Design

The figure 4 shows a high level diagram to demonstrate how the automation can be implemented in the Aquaponics. Each node has an input and output, indicated by an incoming and outgoing arrows, respectfully.

The components are from the original Aquaponics system, which are surveyed by the sensors. The latter captures dedicated data from the medium, like temperature or water level and emits an electronic signal to the microcontroller.

The latter is the main responsible for the automation, it is where the logic is stored, so the input signal are interpreted by the installed program and another electronic signal is sent to the components, this signal could be directed or indirect, the former is when the output current is enough to the target component to work, otherwise the latter is used via a relay, that is connected with an external power supply, and behaves as an signal amplifier.

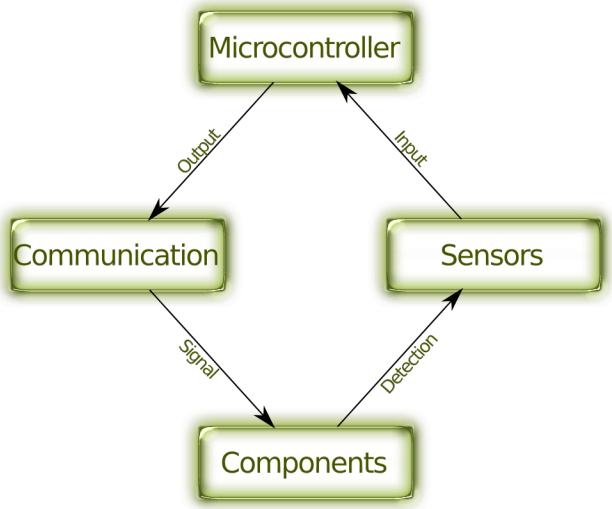

Figure 4: A simple diagram to represent a high level design of this project

3.5 System Architecture

Figure 3.5 illustrates the low-level design. Based on the figure 4, the figure 5 depicts a model to automatize the water cycle control from the requirement R1. Now one can see where to connect each terminal and every important component is presented.

Each wire color has a meaning. Blue represents the ground voltage, red is high voltage and yellow is low voltage. The submersible water pump is located inside the fish tank and normally is turned off. When the periodic signal comes from the microcontroller's digital output, it is amplified by the relay switching, which in turn makes the power supply to deliver high voltage to the water pump, the latter works until the relay is switched again by the microcontroller's signal downtime.

3.6 Project Decisions

There are a lot of authors [8] [4] [12] that have had experience with Aquaponics automation. Most of them chooses Arduino or Raspberry Pi as the main microcontroller.

Two great reasons for the Arduino's usage are: This work's author already has an Arduino UNO, but not a Raspberry PI. Besides the main reference of this work, the

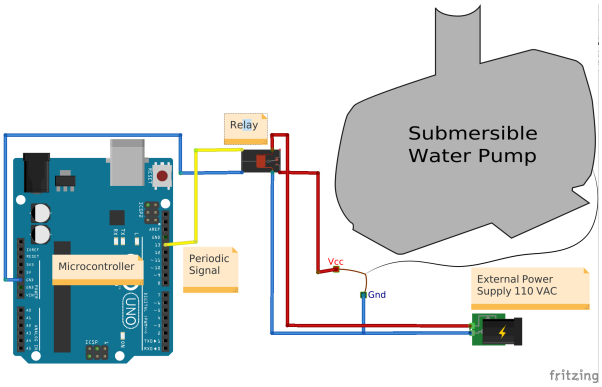

Figure 5: This diagram shows how the microcontroller interacts with the water pump

[11], whose author has many years of experience with Aquaponics and its automation and he uses the Arduino as microcontroller.

The chosen components have been based in the references.

4 Implementation

4.1 Periodic Water Cycling Code

```
/*  Emits a periodic signal one quarter per hour */

// Which pin is connected with the water pump?
const int waterPumpPIN = 4;

// Time intervals
// 15 min = 60e3 ms * 15 = 9e5 ms
const int fifteenMin = 900000;
const int fourtyFiveMin = 2700000;

void setup(){
  // Sets the waterPumpPIN as the output of the periodic signal
  pinMode(waterPumpPIN, OUTPUT);
}

void loop(){
  digitalWrite(waterPumpPIN, HIGH); // sets the signal to logical 1 ON
  delay(fifteenMin); // stay on high voltage for 15 minutes
  digitalWrite(waterPumpPIN, LOW); // sets the signal to logical 0 OFF
  delay(fourtyFiveMin); // stay on low voltage for 45 minutes
}
```

Listing 1: Water Cycle First Code

4.1.1 Reviewed Code

A program that uses delay in the main loop to make timed actions is error-prone, since the delay will be belated by the overhead of other operations located inside the main loop. Moreover the Arduino will not detect any interruption during a delay, so we need to use delays only when it is really necessary.

To make a better code and to improve the scalability of the automation for more use cases that needs to be precisely timed, one needs to review the code to find another way to do periodic actions. We used [2] as a reference to remove the delay-based code from water cycling's code 1. Given that Arduino only offers a functions that retrieve time in order of milliseconds, the listing 2 uses the millis() function.

```
1
 /*  Emits a periodic signal one quarter per hour without using delay */
3
// Which pin is connected with the water pump?
5 const int waterPumpPIN = 4;

7 // State of the signal
int waterPumpState = LOW;
9
// Using unsigned variable to support more data
11 unsigned long previousMillis = 0;

13 // Time intervals
// 15 min = 60 s * 15 = 9e3 s
15 const int fifteenMin = 9000;
unsigned long previousSeconds = 0;
17 // Milliseconds in one second
const int second = 1000;
19 // Seconds in one hour
const int hour = 3600;
21
void setup(){
23   // Sets the waterPumpPIN as the output of the periodic signal
   pinMode(waterPumpPIN, OUTPUT);
25 }

27 void loop(){
   // Get current time in milliseconds
29   unsigned long currentMillis = millis();

31   // If a second passes
   if (currentMillis - previousMillis >= second) {
33     // Store the last timestamp for the last second
     previousMillis = currentMillis;
35     // Increment the number of seconds passed
     ++previousSeconds;
37     // Reset each hour
     previousSeconds %= hour;
39   }

41   // For 15 minutes do
   if ( previousSeconds <= fifteenMin ) {
```

```
43      digitalWrite(waterPumpPIN, HIGH); // sets the signal to logical 1 ON
    }
45  // For other 45 minutes do
    else {
47      digitalWrite(waterPumpPIN, LOW); // sets the signal to logical 0 OFF
    }
49
}
```

Listing 2: Water Cycle Code without Delays

Note that we are storing the timestamps in a 32-bit variable at line 11, so the overflow will occur in almost 50 days, because the variable storage limit is $(2^{32} - 1)$ ms.

$$limit = FFFF \qquad limit + 1 = 0000 \tag{3}$$

The overflow phenomenon is showed in (3), when 4 bytes can't afford to represent larger numbers anymore, because all bits are set to 1. If someone adds one to this limit, the value turns to zero, due to the 9th bit, or the carry, is ignored.

But there is a fancy property with this overflow that soothes this worry: the subtraction of two timestamps will remain correct even if there is a overflow in those numbers, given that the variable is unsigned.

$$time_A = limit - 3\ milliseconds = FFFB \tag{4}$$

$$time_B = 10\ milliseconds = 000A = limit + B \tag{5}$$

$$time_B - time_A = 000A - FFFB = 000E = 14\ milliseconds \tag{6}$$

For example, $time_A$ (4) is a timestamp that records a time 3 milliseconds past the overflow, $time_B$ (5) records a time with 10 milliseconds after overflow, or 11 milliseconds after the limit, so there is a difference of 14 milliseconds between $time_B$ and $time_A$. The equation (6) shows that the result is still correct even after an overflow.

4.2 Finite State Machine

When a program counts time like the listing 11, with many conditionals waiting for some event, such as a time limit, this program could be interpreted as an implementation of a finite state machine, or FSM.

This representation will be important to design the next requirements, since we could use the time management FSM to interact with more FSMs. So this project will be event-driven, resembling what happens in a real-world environment.

4.3 pH Level Control

If the water tank's pH level is under 6, the high pH water will be pumped precisely, until the former gets the pH between 6 and 7. Or else, if the pH reaches levels greater than 7, deposit the low pH water in to the tank.

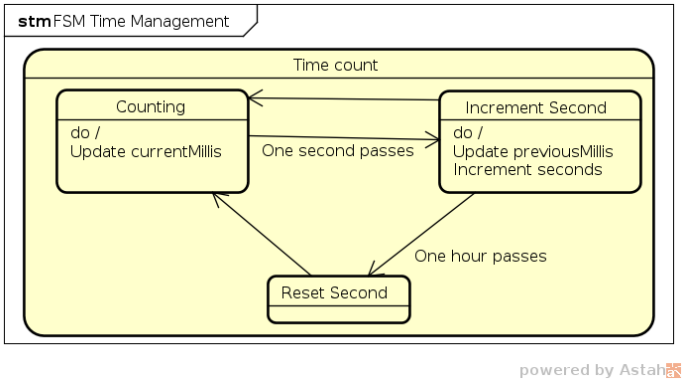

Figure 6: A FSM shows how the time counting from periodic water cycle works

```
/*  Keeps pH level around 6.5 */

// Which pin is connected with the pH Sensor?
const int pHsensorPIN = 5;
// Which pins are connected with the pH peristaltic pumps?
const int high_phPIN = 6;
const int low_phPIN = 7;

// pH Limits
const int lowestPH = 6;
const int highestPH = 7;

void setup() {
    // Sets the pHsensorPIN as the output
    pinMode(pHsensorPIN, OUTPUT);
}

void loop() {
    pHRawValue = analogRead (pHsensorPIN);
    // Transform raw value into the actual one
    pH = process(pHRawValue);

    if ( pH < lowestPH ) {
        // Turning on the high pH pump
        digitalWrite ( high_phPIN, HIGH );
        // Guaranteeing that the low pH pump is off
        digitalWrite ( low_phPIN, LOW );
    }
    else if ( pH < highestPH ) {
        // Turning on the high pH pump
        digitalWrite ( high_phPIN, LOW );
        // Guaranteeing that the low pH pump is off
        digitalWrite ( low_phPIN, HIGH );
    }
}

```

Listing 3: pH Control First Code

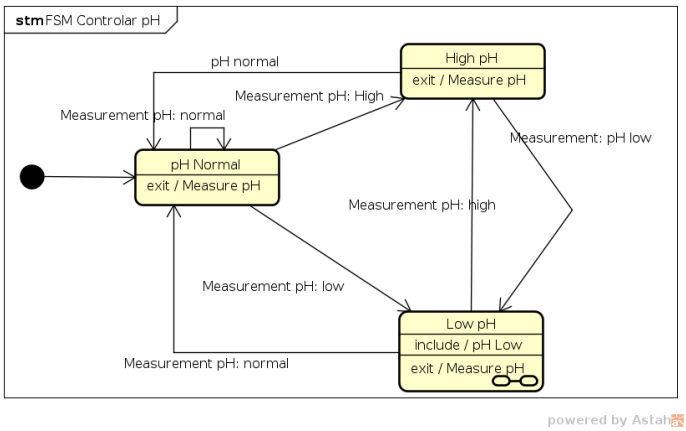

Figure 7: A FSM to represent how the system keeps the pH at secure levels

5 Results and Evaluation

5.1 The Circuit Setup

To setup the test environment, four LED have been replacing by the water cycling and pH pumps. It is important to pay attention at the maximum current draw from Arduino, as well as the 20 milliamperes limit for each LED, hence there are three 270Ω resistors in serial with each LED, as represented in the circuit diagram of figure 8.

5.2 A Graphical User Interface

For the sake of making the unit tests to be shown in a written report, a simple Graphical User Interface (GUI) has been implemented using Processing. Processing is a simple programming language which is integrated with OpenGL, Java and Serial communication. So it is a reliable choice to display and transmit Serial data from/to Arduino within the computer screen. But, instead of being a monitoring-only GUI, the Processing provides built-in functions to make mouse interactions easily, hence the tests could be made faster. The GUI program converses with a special Arduino program, which is a merge of the code from 2 and 3.

5.2.1 A Simple Serial Communication Protocol

To make the Arduino and Processing exchange messages and understand those exact meaning intended, a simple protocol has been made allowing a simple handshake and several simple commands, which can be interpreted as text messages.

The figure 10 is a sequence diagram which describes in detail how the communication is made. Note that the messages derived from 1 are regarded to the connection phases, and the messages derived from 2 groups every command related to pH Control.

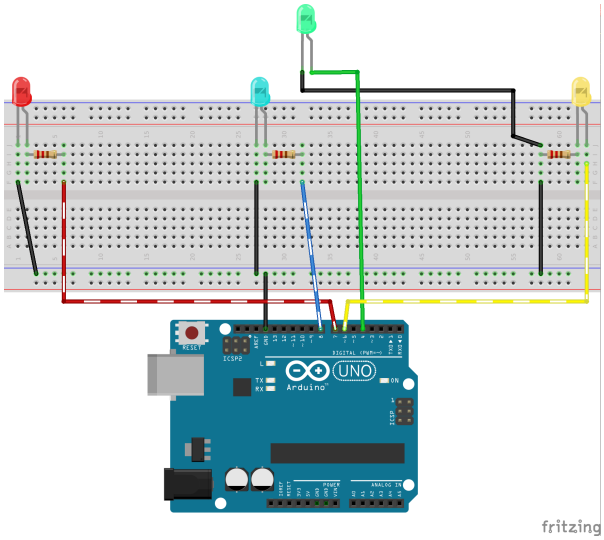

Figure 8: The circuit diagram used for testing the system. It respects the pump pins convention, where the pumps are replaced by LED for the sake of simplify the test

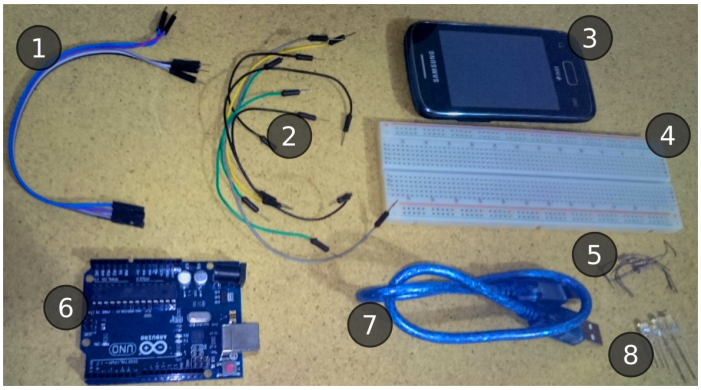

Figure 9: The components used in the experiment. 1) Male-female jumper cables to physically separate different FSM LED; 2) Normal breadboard cables; 3) A mobile phone to record the real-time video; 4) A breadboard; 5) Four 270Ω resistors; 6) An Arduino Uno Revision 3 programming platform; 7) USB cable for serial communication and power supply; 8) Four LED with contrasting colors: Red, Green, Blue and Yellow;

A screenshot of the serial monitor has been made to show a real example of the communication, as follows in the figure 11, one can see that the pH of 7.5 was requested and the response is composed by 14 messages updating the pH value for each second, with an extra one to finish the response packet and let Arduino to receive new pH

Table 5: Arduino–Processing Commands

Command	**Source**	**Meaning**
A	Arduino	Try to establish contact with a client
ACK	Arduino	Handshake is done
C	Arduino	The water cycling pump has been toggled
p<float>	Processing	Send a hypothetical pH value of <float> to simulate how the system responds to it
r	Processing	Establish Contact or Retry Establishment

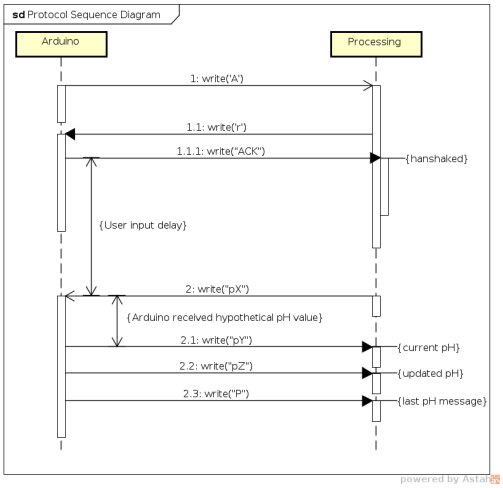

Figure 10: A sequence diagram depicting a connection between Arduino and Processing powered GUI followed by a pH Control request

values. This relative high number of updates are expected, since there the pH variation is simulated like as described in section 5.2.2.

Error Handling A classic retry connection option is available to make the interface more usable, when the user has forgotten to connect the Arduino to the USB port, for example.

IP Camera Support With the usage of the IP Capture library [10] and an Android App, the author's camera turned into a IP Camera which can be displayed in real-time within the GUI.

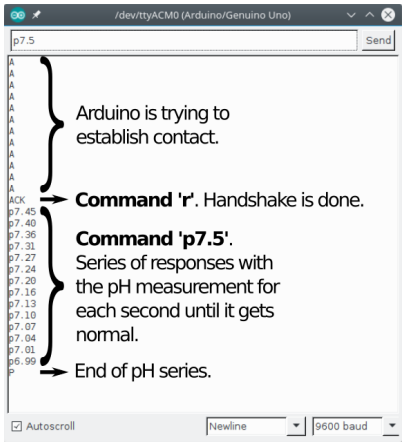

Figure 11: The Arduino IDE Serial Monitor showing raw serial data

Table 6: Circuit LED turned on representation

Green	Water is being cycled by a pump
Red	pH is below the minimum desired value
Blue	pH is in the normal range
Yellow	pH is above the minimum desired value

5.2.2 A Video Overview

A screencast has been uploaded at Google Drive,[2] where the LED lights represent the FSM's states as described in table 6. Their location is showed in figure 12.

Concerning the pH variation during time – when two solution with different pH are blended –, a simple and rough approximation has been made, based on the expectation that the pH will vary lesser and lesser during time, as the tank solution's pH becomes closer to the pH of the controlling pH tanks.

Moreover, to still being concise, the water cycling parameters have been reduced: one hour is 12 seconds and a quarter of a hour is 3 seconds.

The water cycle counter increments when the pump turns off, in other words, when the LED is turned off in the simulation. The pH level terminology follows what is expected to be a desirable pH range in R2.

[2] https://drive.google.com/file/d/0B4QCTJgfjhCbc3FqTmVNSEE1U2c/view?usp=sharing

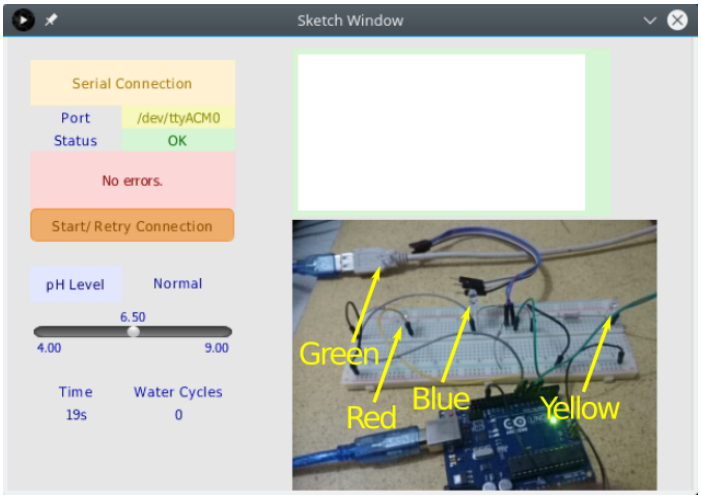

Figure 12: A screenshot of the GUI with all four LED highlighted

6 Conclusion

As the author of this work did not have the enough experience with microelectronics, because he had been doing the software-biased modality of his degree, he had some difficulties relating the schematics designs, how to use relays and other advanced components.

There are three main contributions provenient from this work: simple automatization of a relevant system with inexpansive components, a documented project based on variations of a classic development model and engineering abstractions, and a simple GUI to detail what happens with each component when the automatization happens.

The latter contribution serves as the high-level demonstration of the POC, evincing that the use cases are being respected by simulating the sensor's value in real-time.

In the interface's aspect, due to the communication protocol, it is feasible to extend this system to respond to the new use cases, such as automatic fish feeding, solar illumation simulation, temperature control and root cloging prevention. The modification needed in this case is just new commands to treat new sensors inputs and outputs.

References

[1] *Arduino - ArduinoBoardUno*. URL: https://www.arduino.cc/en/Main/ArduinoBoardUno (visited on 06/30/2016).

[2] *Arduino - BlinkWithoutDelay*. URL: https://www.arduino.cc/en/Tutorial/BlinkWithoutDelay (visited on 05/30/2016).

[3] *Arduino - Introduction*. URL: https://www.arduino.cc/en/Guide/Introduction (visited on 06/30/2016).

[4] John O Clark. "System of systems engineering and family of systems engineering from a standards, V-model, and dual-V model perspective". In: *Systems Conference, 2009 3rd Annual IEEE*. IEEE. 2009, pp. 381–387.

[5] DFRobot. *Analog pH Meter Pro*. URL: http://www.dfrobot.com/index.php?route=product/product&product_id=1110 (visited on 07/01/2016).

[6] Elder Austin Simonson. *Hyduino - Automated Hydroponics with an Arduino*. URL: http://www.instructables.com/id/Hyduino-Automated-Hydroponics-with-an-Arduino/ (visited on 07/01/2016).

[7] Gareth Coleman. *aquaPionics - Hackaday.io*. aquaPionics. 2014. URL: https://hackaday.io/project/2190-aquapionics (visited on 02/20/2016).

[8] Simon Goddek et al. "Challenges of sustainable and commercial aquaponics". In: *Sustainability* 7.4 (2015), pp. 4199–4224.

[9] Harry Goldstein. "The Indoor Farm". In: *IEEE Spectrum* 50.6 (2013), pp. 59–63.

[10] *IPCapture*. URL: http://www.stefanobaldan.com/projects/ipcapture/ (visited on 07/05/2016).

[11] Rik Kretzinger. *Build an Aquaponic Garden with Arduino — Gardening | Make:* ed. by Rik Kretzinger. 2015. URL: http://makezine.com/projects/aquaponic-garden/ (visited on 03/10/2016).

[12] M. U. Leatherbury. "VEGILAB and aquaponics indoor growing system". In: *Technologies for Sustainability (SusTech), 2014 IEEE Conference on*. July 2014, pp. 135–139. DOI: 10.1109/SusTech.2014.7046233.

[13] Sonali Mathur and Shaily Malik. "Advancements in the V-Model". In: *International Journal of Computer Applications* 1.12 (2010).

[14] *Simple Arduino Controlled Aquaponic System*. URL: http://www.instructables.com/id/Simple-Arduino-Controlled-Aquaponic-System (visited on 05/19/2016).

[15] *Tables and Charts*. URL: http://ibcofaquaponics.com/information/tables-and-charts/ (visited on 05/02/2016).

[16] Desair Tom. *LaTeX template for use cases*. Apr. 2012. URL: http://www.tomdesair.com/blog/2012/04/latex-template-for-use-cases/.

[17] *Using V Models for Testing*. URL: http://insights.sei.cmu.edu/sei_blog/2013/11/using-v-models-for-testing.html (visited on 06/23/2016).

[18] Geoff Wilson. "Greenhouse aquaponics proves superior to inorganic hydroponics". In: *Aquaponics Journal* 39.4 (2005), pp. 14–17.