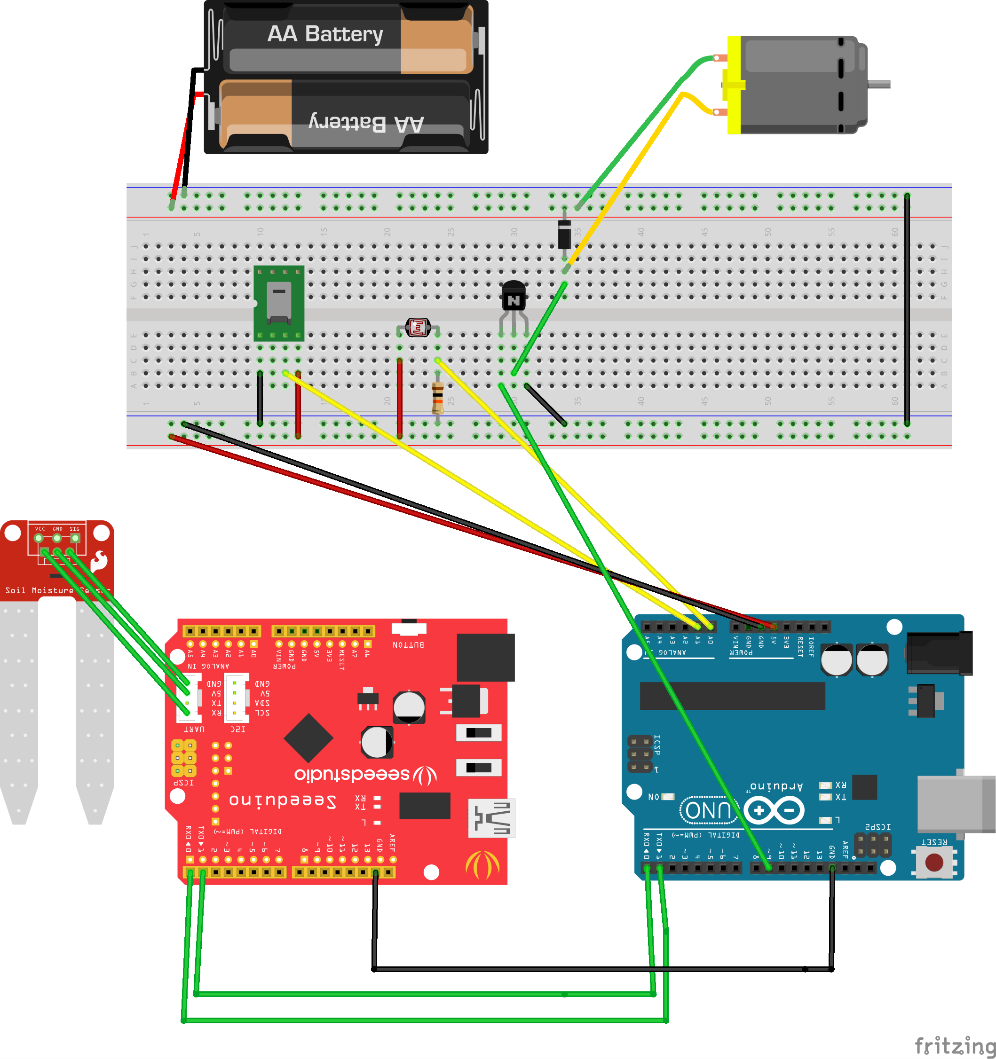
**Smart Irrigation System**

**Description of the system:**

The smart irrigation system created for this assignment is a prototype system designed to show the potential implementation of a system that will allow users to monitor specific sections of their garden or small agriculture setup and using an MQTT communication network to create an IOT network

**Hardware:**

**Planned Circuit Diagram:**



**Description and Concept:**

The main idea behind the smart irrigation system was to use the Arduino as the processing unit to multiple sensors, acting as a central ‘hub’ to process data that comes through. To that end, we used the Arduino Uno Genuino and Seed Lotus v1, the Lotus is designed to work with a specialised Moisture sensor called the Grove Moisture Sensor, which can read the water content in soil. The Lotus board communicates via serial communication with the Arduino main board to transfer the moisture data over for processing. This achieved by connecting the two boards on their Tx and Rx ports, crossing the wires so that the Rx of the Uno is connected to the Tx of the lotus and vice versa. The ground wire is required for this communication as the Uno is powering the Seed board, in order to dispose of any excess power being run through the system.

The rest of the sensors connect to the Uno via a bread board network. Firstly, on the left-hand side of the breadboard in the image is the DHT11 temperature-humidity sensor. This sensor records both the temperature and the humidity in the air and returns the value of temperature in degrees Celsius and the humidity as a percentage. The DHT11 provides analogue data and therefore is connected to pin 1 on the Uno. Next to this sensor is the Arduino light sensor which is simply wired into the breadboard to receive power and grounded by a resistor and the data input wire is connected to analogue pin 0. Finally, in this initial circuit idea, there is a motor attached to the other side of the breadboard, intended to simulate a potential sprinkler or other irrigation technology that could be controlled via the system.

The diagram is missing one final feature - the Xbee shields attached to the Uno boards. These enable wireless communication between devices, in our case a laptop and the arduinos. This feature is essentially in allowing communication between device and to implement the IOT notion behind the prototype and provides a degree of mobility that is essential to make the prototype feasible.

**Software:**

**Processing GUI Dashboard:**

In order to get direct information from the Arduinos for testing and development processes, we developed an Admin Dashboard, providing direct visualisation of the information provided in the form of graphs and allowing data trails to be followed through a view history function that holds. The Dashboard is also essential in providing the basis for the MQTT network. As the Arduinos have size limits on the amount of data being held, we free them of memory restraints by passing this over to the device that is running the dashboard. This processing dashboard then does the major heavy lifting in the MQTT delivery system, connecting to the broker and sending appropriate message to topics required by the web app. It also allows the web app to communicate back to the Arduinos by acting as a middle man and subscribing to required MQTT topics.

**Web Application:**

The website prototype is the final part of the IoT system, emulating a prototype of an application that may be hosted on a website or on devices such as smart phones. It is the customer-side of the project, and the final stage of the MQTT network. In order to incorporate the MQTT system into the website, a websocket based web application had to be developed, taking advantage of Eclipse Paho Javascript *(Eclipse.org, 2018)* extension to incorporate a seamless MQTT network through a broker hosted on a separate device (to simulate a ‘hub’ server for a company). By subscribing to the topics that the admin Dashboard posts to, it receives real-time data updates on the sensor data, which then can be interpreted into required actions for the user as well as warnings. In the early prototype, we start with a simple water sprinkler system, but it is open to extensions such as fertilisation sprinklers, automated harvesters or greenhouse controllers.

While an online websocket may be used, we instead used the Eclipse Mosquitto system *(Eclipse Mosquitto, 2018)* to implement a websocket supported local broker, allowing us finer control and to monitor connections. A customized broker also allows the option for password protected connections and SSL enabled data transfer to handle the common security risks that come from IOT connections based on MQTT.

The parameters to determine the best time to prompt the user to enable the sprinkler system or take their own action are based on research information. In a finalised system, you may have the option for the user to select presents depending on the plant or conditions required. In our example we based the data on the best conditions for the growth of daffodils. MQTT connections are also what enables the message for the sprinklers to be activated to the admin dashboard, which then pushes the instruction to the X-bee serial to pass the Arduino devices.

**Development Processes and Tools:**

Version control was established through a Git style repository hosted on BitBucket. We chose a Git system of version control as it is the industry standard for maintaining code and version control. Bitbucket itself was chosen especially due to the group being well versed in its use and its simple, friendly UI. It also hosts the integrated SourceTree application to allow the easiest and fastest version control system.

For our project management we relied on the website “Trello” to develop a quasi-agile development system in which tasks could be assign and keep a record of tasks to be completed, who they were assigned to and the deadlines for each task. This involved member moving the cards when they were working on, or had completed their given task, allowing the other group member to see what was being completed and when to better plan their own individual workloads. For example if a member was waiting for a specific section to be completed they could see if that section was currently in progress.

Documentation was also managed using docblockr, an Atom-based package that allows for the production of professional documentation (Javadoc style) within the Atom IDE *(Atom, 2018).* The reason to use this instead of the traditional Doxygen or Javadoc is due to the fact that Processing while being Java-based in syntax actually has some intrinsic differences that can sometimes flag errors. The HTML and CSS code was simply commented regularly as no advanced documentation systems exist for these mark-up and styling languages.

**Personal Contribution to the Project Development:**

During the initial discussions of the project development, we initially decided to split the work into the 3 distinct sections above, the hardware development, the Admin Dashboard and the Web Application. We would then come back together to work as a team once the structures for these sections were in place.

It was decided that I would be responsible for the web application design and development due to my experience with basic webdesign, javascript and php, though these responsibilities would later broaden into developing the websocket and MQTT broker network. Because much of my work was network based, it gave me a chance to develop the tools independent of the hardware and allowed me to establish a very individual contribution to the project. I researched into MQTT websockets and discovered the various pitfalls of using this relatively new technology, including the lack of windows-based executable version of Eclipse Mosquitto, forcing us to later use a OSX device to act as the broker instead.

Once I discovered the basic implementation of Eclipse Paho to connect to a Mosquitto Broker via pair programming along side Jordan, I was able to then properly develop the website, using Javascript to provide real-time client-side data transfer that could negate the need for the databases and php.

The final stretch of development was developing a basic HTML and CSS structure to house the required javascript, as well as implementing the parameters required to provide the user with the warnings and buttons required. I was also involved in ensuring that the web application could also communicate back with the processing dashboard and that both systems would work in sync, to avoid issues of different data sets being processed. I was also in charge of documentation, integrating the DocBlockr system to provide professional documentation for the project.

**Critical Reflection and Evaluation:**

**Hardware Reflection:**

There were however several problems during the project lifecycle related to hardware. A lack of knowledge of sensors and hardware lead to stumbling blocks that could have been overcome with better research. For instance buying a sensor whose library was only compatible with windows when a team member was using exclusive an OS device. Also lack of foresight for things such as power units for the Arduino shields meant while functionality was there, the end demonstration lacked the same effect.

There were also significant reductions in functionality due to an issue with the system to emulate a sprinkler. Due to some hardware malfunctions, the motor originally planned had to replaced with an LED to simulate an external system, as this is just a simulation of another piece of hardware this did not affect the functionality of the system. This is as well as more hardware compatibility issues, as we were only able to use one of the two test sensors contained the Seed Lotus and moisture sensor. Meaning both arduinos were not fully ‘independent’ of each other as intended.

**Software Reflection:**

There were adjustments made due to some software limitations encountered during development. For instance due to firewall limitations on networks, we could not use an online ‘cloud’ MQTT broker, which while with security issues would have allowed for MQTT communication to occur regardless of whether a broker device was ‘online’. However while this may have been a downside, discovery of SSL and security issues meant that a locally run broker was a much more safer and testable device, without relaying on external APIs for stability.

Also a lack of research into the limitations of serials meant that we ended up designing a system with an overreliance on the feature, which often was inconvenient and unyielding, essentially leading to the activation of the LED ‘sprinkler’ to be demonstrable more on the dashboard console than on the actual physical device. This is a major failing in the intended design of the system and could have been avoided if other languages or techniques were explored.

**Overall Evaluation:**

When considering the main objective of the project, to provide an IoT network based on an MQTT messaging system that allowed for wireless connections between devices, I believed we succeeded. All communication through devices were completely independent of each other, using a mixture of Xbees and MQTT to provide information between each service. This successful integration of IoT also left an easy way to extend and add to the system, as the basis is a light-weight responsive network that consists simply of a Processing file and a Website template. Data was analysed and used effectively and provided tangible information on the system at play.

Given more resources and time there was potential expand the functionality and usefulness of the system due to it’s simple and easy structure. It could also become more integrated with server side systems developed with PHP and SQL to provide data that is persist rather than just session based.

I believe this potential and overall fulfilment of the original objective as per the project spec *(APPENDIX A)*, meant the assignment fulfilled what it set out to do, even if it was not in the most efficient and portable manner. More research and time on the project would have certainly allowed it to reach its full potential.

*(Word Count: 1991)*

# **References**

*Eclipse.org. (2018). Eclipse Paho - MQTT and MQTT-SN software. [online] Available at: https://www.eclipse.org/paho/clients/js/ [Accessed 22 Mar. 2018].*

*Eclipse Mosquitto. (2018). [online] Available at: https://projects.eclipse.org/projects/technology.mosquitto [Accessed 22 Mar. 2018].*

*Atom. (2018). docblockr. [online] Available at: https://atom.io/packages/docblockr [Accessed 22 Mar. 2018].*

APPENDIX A

**PROJECT SPECIFICATION - Project (SEGM) 2017/18**

|  |  |
| --- | --- |
| **Students:** | **Maria Khan, Aijaz Ahmen, Jordan Richards** |
| **Date:** | **05/02/2018** |
| **Supervisor:** |  |
| **Degree Course:** | **Computer Science** |
| **Title of Project:** | **Developing network of soil sensor stations and sprinklers to provide an internet of things farming irrigation system.** |

#### Elaboration

|  |
| --- |
| For both home and industrial use soil monitoring and plant irrigation is an extremely important role for establishing crops and flowers for use. Due to weather changes, time consumption and a lack of detailed data, it can often be difficult to assess the best time to water or feed your vegetation to get the best result.  The project is aimed at using the internet of things in order to create a device that can report multiple factors to determine the best time to water plants and achieve maximum yield. These include rainfall, sunlight and soil chemical data. All this information was passed to a device, which would then provide the user the means to active a sprinkler system to immediately tend to the issues. This also involves communication between the sensors and the sprinklers so that the user can set an automated system that manages to water based on the sensor data with no input from the user. |

#### Project Aims

|  |
| --- |
| * Merge multiple sensors onto an Arduino board in order to process multiple data types to provide an overview of crop conditions. * Use a data logger to handle large packets of data in real time. * Develop a web-based site to act as the ‘app’ for the user to use for the internet of things. * Create an interface between the sensors and the motor in order to network the devices together. |

#### Project deliverable(s)

|  |
| --- |
| We will deliver a chip system that will be able to be used on an actual plotted plant. It will display data as well as give the user the option to set up the automatic sprinkler system or manually activate. The sprinklers will be represented by a motor that will communicate with the chipset. |