PROJECT ARTEFACT REPORT

Prepared by
Philp Williams
s222574652

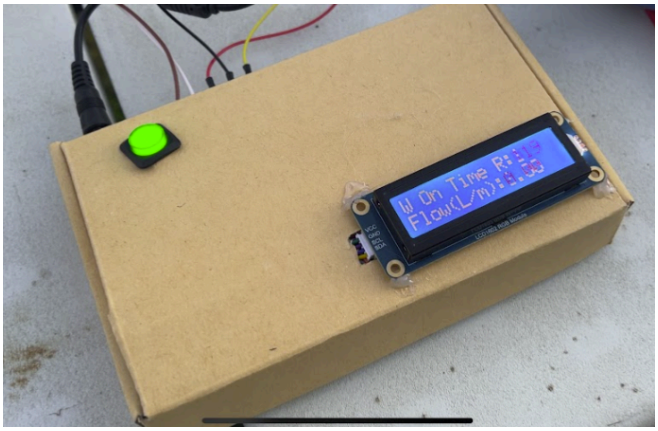


TABLE OF CONTENTS

INTRODUCTION

Welcome to this Project Artefact Report for SIT730 - Embedded Systems Development. Over the following pages, we will explore in depth the EWS system (Emergency Water Shutoff) and how it meets the requirements laid out as part of the final project assessment for the class.

The core objective was to develop an embedded system to solve a real problem; along with the purpose of solving this problem, the system should include several embedded systems using a range of sensors that work together to achieve the desired outcome.

This report will demonstrate how the EWS system meets these requirements along with:

- Project overview covering the problem in detail and how the system can solve it
- Interdevice communication via MQTT and peripheral communication with I2C
- Development of robust fault-tolerant system.
- Real-time response in the system via interrupts
- A responsive system that manages input from multiple endpoints.
- Robustness, with individual components able to perform their primary functions if the WiFi or MQTT systems are offline.

1. PROJECT OVERVIEW

Background / The Problem

The idea behind the EWS system is the result of a flood I had in January 2024, where a flexi hose in our ensuite burst, flooding my master bedroom and study. Luckily, my wife was at home, discovering the water within half an hour, according to the usage data from our smart water meter, and was able to turn off the main water supply, saving the rest of the house. Sadly, this is not the first such event we have had over our house's 14-year life, with one other major flood from a fridge water filter and four external leaks from cracked pipes that did not cause significant damage but wasted a lot of water.

Side Note: *Please consider this in your own home as many people are not aware the steel braiding on these flexi hoses can deteriorate over time, and you should inspect them every six months for any bulges indicating an imminent break. The functional lifespan is 5-10 years, so it is a good idea to change them out as preventive maintenance along with washing machines*

and dishwasher hoses. These flexi hoses *have become popular in the last 20-30 years, mainly used on mixer taps.*

Fig. 1 – Images showing resulting damage from flood and Bust flexi hose

Unfortunately, this scenario is not unique and happens to thousands of Australians yearly; Water Damage accounts for almost a quarter (24%) of home claims. A Press Release from QBE outlines these as the leading cause for escape of liquid QBE claims in the home:

- **Burst or blocked pipes,** which account for 46 per cent of water damage, (…)
- **Damaged roofs**, which account for 27.2 per cent of water damage instances.
- **Old or worn-out plumbing**, accounting for 27.2 per cent of instances.
- **Overflowing baths,** sinks and showers, responsible for 13.4 per cent.
- **Clogged drains** accounted for a further 13.4 per cent.
- **Flexi hoses,** at 8.2 per cent of instances. *[1]*

The same QBE report states that while 53% of respondents said their first response to water damage would be to turn off the water main, 21% of Australians do not know where their water main is, and 18% do not know how to turn it off [1]

Also, mitigating rapid water escape by turning off the water main assumes:

1. Someone is home to turn the mains water off
2. The escape of liquid is noticed in a timely manner.

This is why a system like the EWS could be of assistance. While it won't stop a flood from occurring, a well-calibration system can detect when a flood-style event is in progress and turn off the water supply to limit the damage.

Problem Statement

Water damage from scenarios like burst pipes or overflowing bathtubs can cause extensive and costly damage to homes. Quick Action is essential to minimise this damage, but if the occupant is not home, the entire property may become flooded, necessitating extensive restoration and repairs. An automated system that monitors water usage and localised leak sensors to shut off the water supply quickly can significantly reduce the risk and impact of such incidents.

Requirements/Key Features

The Requirmen is the development of an embedded system that can detect when a water event is occurring. This is done through two core mechanisms:

- Flow monitoring: If water has been flowing for a long duration beyond what is typical for a household, the default is 20 minutes, and the supply of water will be turned off once the threshold is exceeded.
- Leak Detection: Remote Leak Detectors are placed in strategic locations; if they sense water, a signal is sent to turn off the water supply.

Additionally, the system will have a UI interface and logging system so the status and alerts can be reviewed remotely, along with controlling the system.

These are the Key Features of each component within the EWS ecosystem:

- Main Unit (EWS)
    - Ability to turn on/off water supply
    - Monitor the duration of water usage, i.e. How long a Tap has been turned on.
    - Turn off the supply of water once a threshold has been reached.
        - The practical default limit is 20 mins, but 2 mins is used in the prototype testing.
    - Report the current state via an I2C display
        - WiFi and MQTT connection active
        - Water on/off
        - Water usage
    - Real-time local controls for turning water on/off
        - Using Interrupts for immediate response.
    - Report state to a remote system via WiFi and MQTT
    - Accept commands from remote systems
    - Fault Tolerant – Able to perform primary monitoring tasks if the network is down
- Remote Water Detector
    - Detect when water is present

    - Indicator and audible alarm to show when water is detected
    - Report the current state via WiFi and MQTT
    - Send a command to the main unit to turn the water off when water is detected
    - Fault Tolerant
        - Able to perform primary monitoring tasks if the network is down
        - Use Watchdog Timer to restart the device in the event of a software issue.
- Remote System
    - Provide GUI for displaying and controlling water on/off state
    - Report on Water Detector state
    - Log data on the device states (Main Unti water on/off, Leak detector wet/dry)
    - Provide alerts on water detected.
- Water System Simulation
    - Pump System to simulate water flowing in a home
    - Have different fixtures to allow the monitoring and measurement of different water endpoints.

Existing work

Similar systems already exist, but most are targeted at the North American market. However, there is one comparable product in the Australian market called Aqua Trip[2], and it shares many of the same capabilities of the EWS system like flow monitoring, water shutoff and optional leak detectors. I believe Aquar Trip may have a lock on the Australian market to do a patent for A fluid flow monitor [3] in their name; this is due to expire in 2026.

The core features that set the EWS apart are our system's online monitoring and control, as Aqua Trip is a fully offline system.

2. DESIGN PRINCIPLES

These are the core embedded system design principles incorporated into the EWS system and how they are implemented.

Component Base Approach.

The components that make up the minimum viable product (MVP) prototype for the EWS system were selected as a combination to meet the technical requirements of the solution along with what was easily obtainable or already on hand.

See Prototype Architecture – Hardware Architecture for a detailed hardware breakdown.

As a result of the component selection based on availability and what was on hand, several sub-optimal design choices would need to be addressed in future versions, while acceptable for a proof of concept (POC); this is a small list of potential improvements.

- Alternative Communication participle, i.e., alternative to WiFi for leak sensors to EWS, direct communication, avoids potential issues with the user network on a critical infrastructure.
- Alternative Soilnoid – the unit used in the prototype also had a restrictive flow rate while it is normally closed, so while we consider it fail-safe, which is desirable, it would also have a higher energy usage and increased chance of component failure.
- Addition of battery systems for emergency use.

Real-Time

While we are not using a full real-time OS with multi-threading, we are using interrupts as part of two functions for real-time response.

- The flow sensor is attached to a hardware interrupt pin, so each pulse from the sensor is captured in real-time by the Nano to calculate the flow rate of the water.
- Push button interrupts – the push button is configured with a pull-up resistor on a falling trigger; when the button is pushed, we directly call the function to close or open the solenoid.

Fault Tolerance

Fault Tolerance has been considered within the design of the system in several ways

- Flow sensor disconnect – should the system lose the flow sensor, flow monitoring will become unavailable; however, a secondary mechanism like the leak sensor can still trigger the switch off of the water supply.
- Both the EWS and Leak Detector are designed to work in standalone mode should the WiFi or MQTT broker go offline. The EWS can turn off the supply based on flow monitoring, and the Leak Detector can at least trigger an audible alarm along with an indicator should water be detected.

Robustness

While all due care has been taken in developing the hardware and software, unintended events can still occur. To help address unexpected events, a 16-second watchdog timer (WDT) has been implemented on the Leak Detector. Should the program hang, the WDT will restart the Nano.

A 16-second timeout was required as the MQTT function has a 10-second timeout.

While this function would also be desirable on the EWS, the implementation interfered with the flow sensor readings, so a rewrite of the flow sensor library or WDT would be required.

Responsiveness

The EWS system is designed to be responsive, with state changes reflected across the system, typically in milliseconds. From a user's perspective, pressing a button on the UI and seeing the EWS turn the solenoid on/off is perceived as instantaneous.

User Centric Design

While only a prototype, we have focused on the user interface, and we use simple indicators to communicate the status to the user, including:

- EWS
  - A single button is used to turn the water supply off/on, with an indicator showing when the supply is on.
  - Local Display output showing status, Water off/on, Current Water Flow Rate and Time Remaining until water is turned off.
- Leak Detector
  - Green LED to show the device is online
  - Red LED and Audible Alarm to indicate water is detected
- Home Assistant - Using the well-established Home Assistant UI, we make it easy for end users to:
  - View the current state of the EWS and Leak Detectors
  - Turn the EWS on/off
  - View the state history of EWS and Leak Detectors
  - Receive Push notifications on the alert of water detected by the Leak Detectors.

3. PROTOTYPE ARCHITECTURE

Hardware Architecture

EWS Main Unit

The main EWS unit is comprised of the following components

- Arduino Nano 33 IoT
  - Primary Brain of the system and network connectivity
- I2C 16-line display
  - Display and provide the current state
- Button with LED
  - Used to Toggle water on/off, and LED indicates if water is off/on

- 5V/3.3V Power Supply (6-9V Input)
  - Provides 5V for Nano, Relay Control, Flow Sensor, Reference for Logic Level Converter
  - Provides 3.3V for Display, Reference for Logic Level Converter, and Button Pull Up resistor
- Buck Converter
  - converting from 12V to 8V for the 5V/3.3V Power Supply
- 12V Relay
  - For turning the solenoid on/off
- ¾ Inch Flow Sensor
  - Uses a hall sensor to measure the flow of water is connected to an interrupt pin, so each rotation is registered and the speed of the water can be calculated
- ¾ Inch 12V solenoid
  - Used to turn the Water supply on/off
  - It is a normally closed model, i.e. it will fail safely should the system shut down.

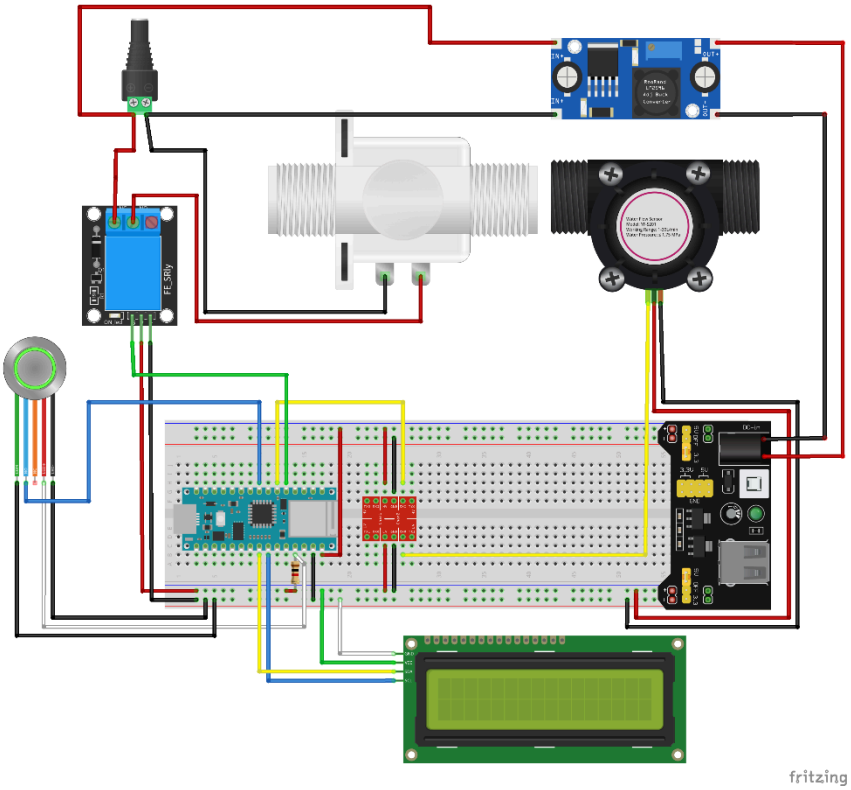


Fig. 2 - Main Unit Wiring Diagram

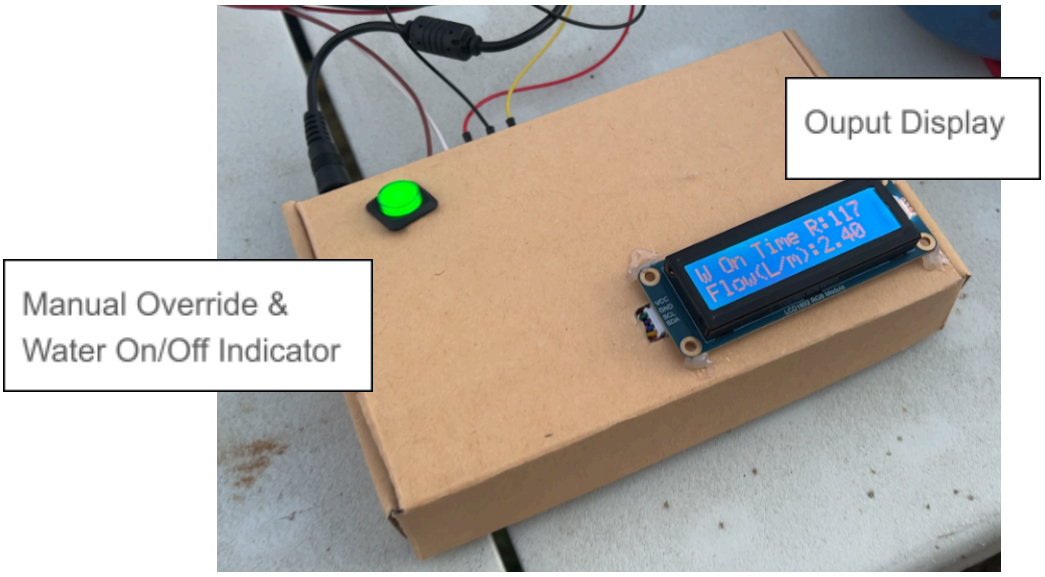


Fig. 3 - Main Unit exterior

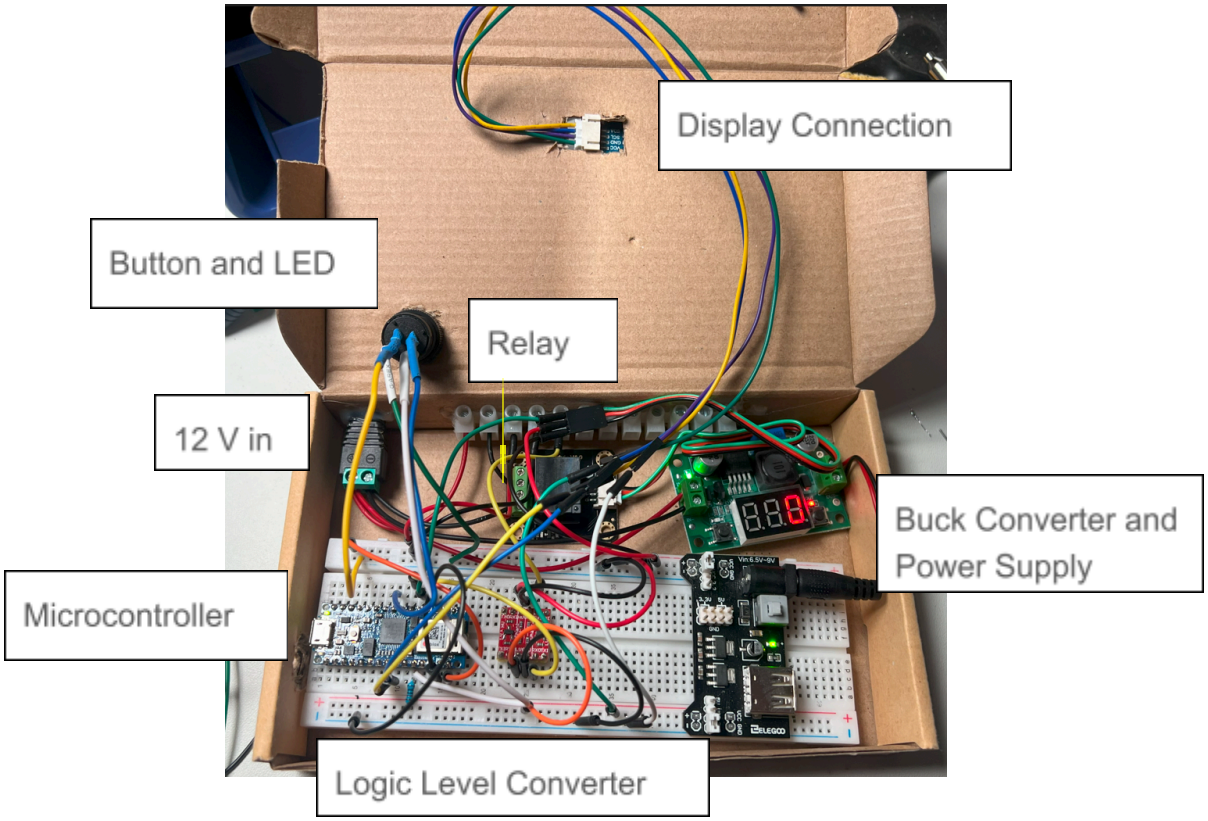


Fig. 4 - EWS Main Unit Internals

Leak Detector

The Leak Detector's primary components and functions.

- Arduino Nano 33 IoT
  - Primary Brain of the system and network connectivity
- 2 x LEDs with resistors
  - Act indicator lights for Connectivity Status and water alerts
- Piezo buzzer
  - Alerts with 1000Hrz tone in 250ms intervals when water is detected
- pn2222 Transistor
  - Used to trigger the piezo buzzer to avoid overdrawing the current on the Nanos GPIO.
- Water Sensors
  - Used to detect water.

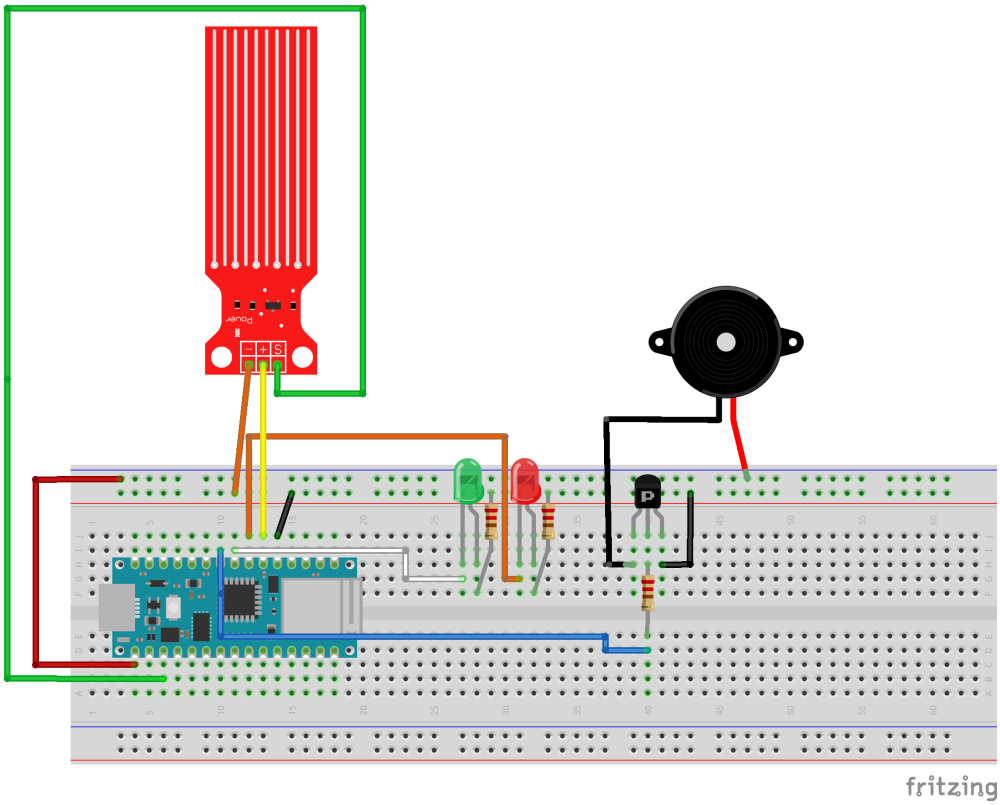


Fig. 5 - Leak Detector Wiring Layout

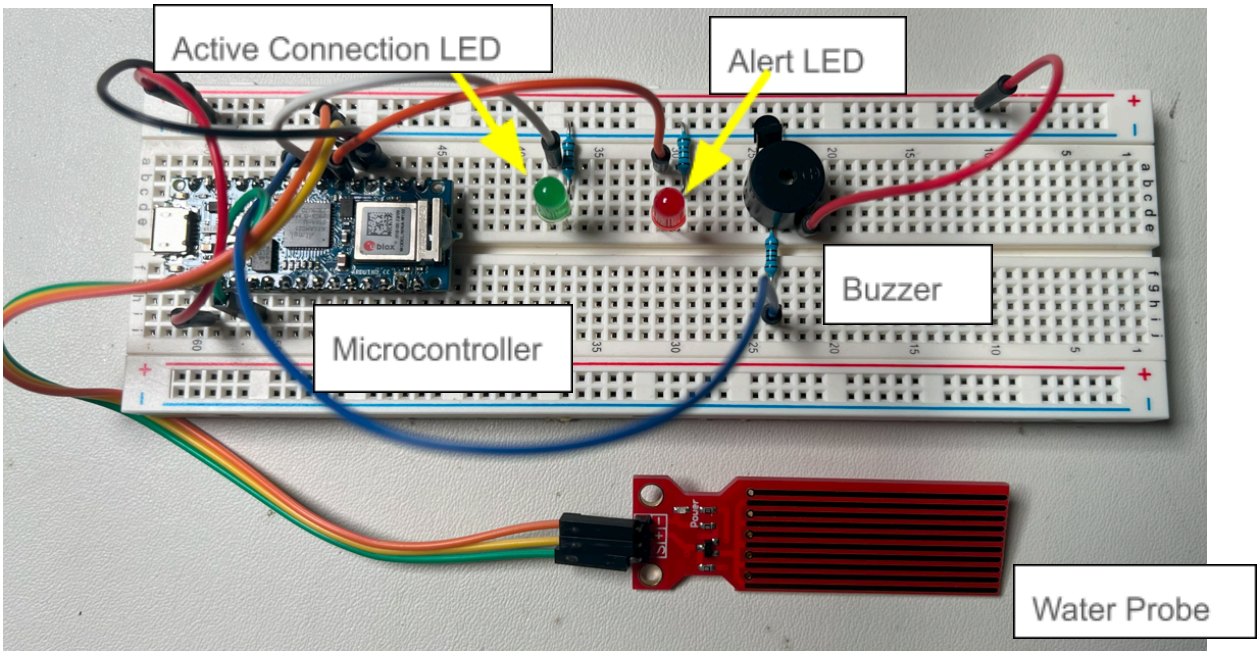


Fig. 6 – Leak Detector Physical Layout

Server

- Raspberry Pi 4, Power Supply, Suitable Case
- Micro SD card
    - Loaded with Home Assistant Image

Network Arectacture

The prototype has been put in a separate LAN with a dedicated router for testing.

The Home Assistant and Maoqueto MQTT broker are running on Raspberry Pi and are directly connected to the router with a static IP. All other components are connected via WiFi using DHCP for addressing; see Fig. 7 - Network Layout.

Optionally, this system can be placed on the internet using a Nuba Cass subscription to make Home Assitant accessible outside the local LAN.

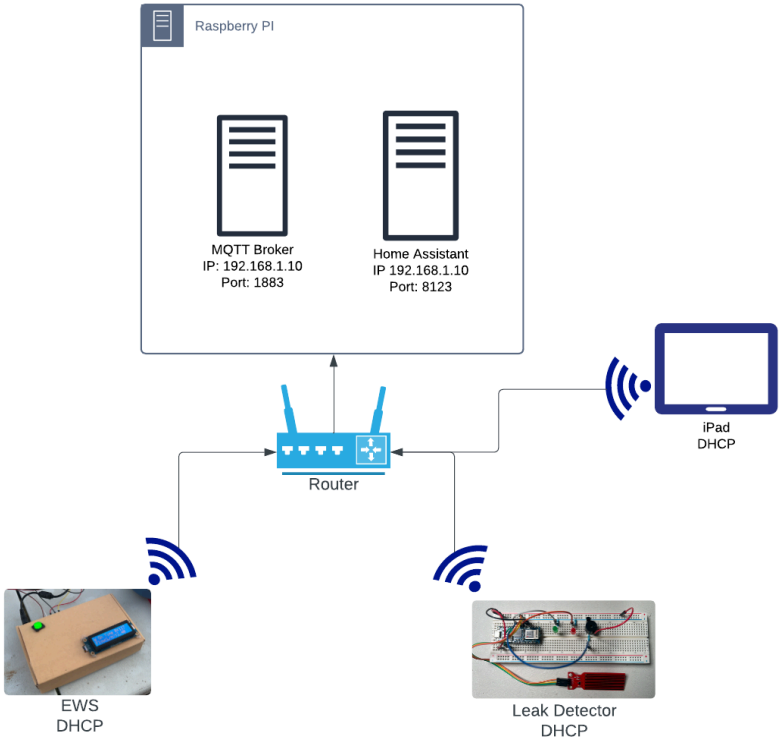


Fig. 7 - Network Layout

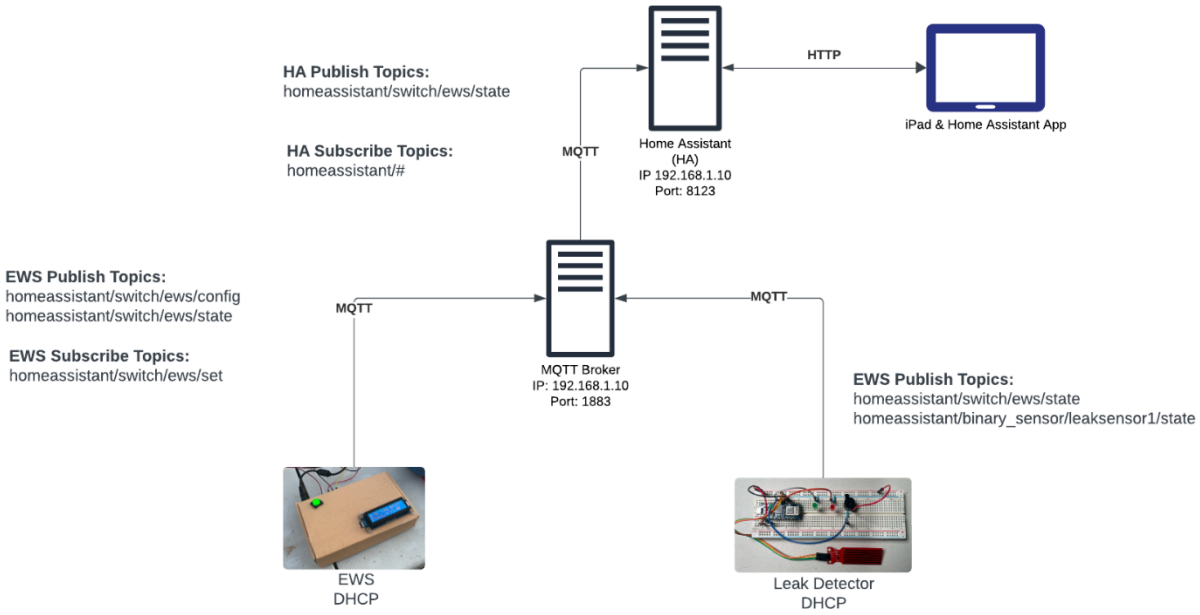


Fig. 8 - Logical connection layout and MQTT Topics

4. GITHUB CODE

Link to the GitHub Repository

https://github.com/DaGoblin/SIT730-Task11.1

5. TESTING APPROACH

For the development of the EWS system, we took a unit testing approach where each component and sensor was built and functionally tested to ensure it worked as expected, mainly via trial and error. These code units were then integrated into the larger code base and functionally tested as a whole system.

There is a single code test function for the WDT timer where we trigger an infinite loop to verify that the WDT works as expected.

Core Functions have been tested methodically.

- Test Automatic Shutoff at 2 minutes
- Test Shot Off with Leak sensor
- Test UI Interaction with EWS
- Test Data Reporting for EWS and leak detector
- Testing Button Response (see button debounce issue)
- Test Multiple Triggers and interactions

This is roughly the development and test timeline.

- Build test rig for plumbing.
    - I discovered that the original pump could not provide enough pressure for the system to work.
- Operate the system by connecting the Solinoid directly to 12v
- Adjust the system until no leaks are present

- Develop and test code for the Flow Sensor
- Test the operation of the Relay and Solenoid via the Micro Controller
- Build main EWS prototype and Integrate Flow sensor and Relay code
    - Discovered that the flow sensor needs to be connected to an interrupt pin (pin 4 is not an interrupt pin on the Nano 33 IoT)
- Build code for the button and interrupt
- Button reliability issue found in the
    - Integration testing found an issue where the button could register multiple presses, and the relay actuating could also register as a press. A debounce timer of 250ms was added for button presses and relay changes.
- Build Remote Leak Sensor
- Unit test locally with LED
- Setup MQTT on EWS via trial and error
- Setup Leaksensor MQTT alerts
- Integration Testing system works as expected
- Setup and tested Automation for notifications in Home Assistant
- Added display to EWS and tested output functions
- Full system integration and function Tests
- **Live Demo**
- Leak Detector added Connected LED and Routine
- Leak Detector added a piezo buzzer
- Updated Button interrupt to call water on/off function directly
    - This led to stability issues with the EWS locking up periodically.
    - In the second implementation, only the Relay and LED are turned off/on during the interrupt call; other functions like serial output and MQTT updates take place during the relay status change in the main loop.

- Improved WiFi and MQTT functions to no block the system from running, so there is a fallback mode if WiFi or MQTT broker is offline.
- Implemented Watchdog Timer and test routine on Leak sensor
    - EWS WDT and flow sensors interfere with each other.

6. USER MANUAL

This guide will explain in detail how to build your own EWS system. I recommend reading all sections before purchasing your hardware or starting assembly, as you may wish to make changes or skip parts like the pump test rig.

Home Assistant and MQTT broker setup

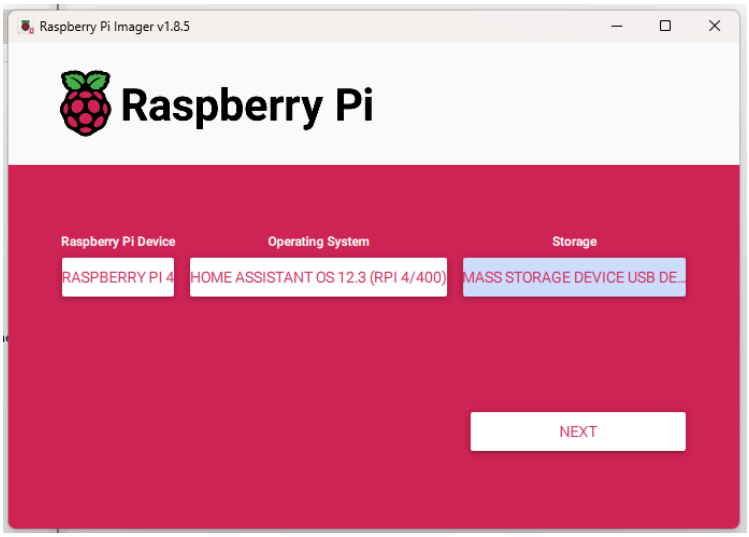
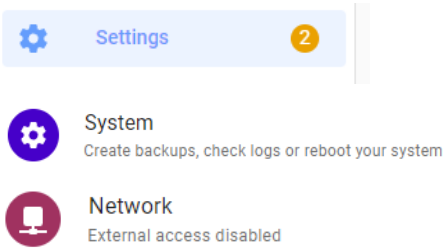
Hardware Required:

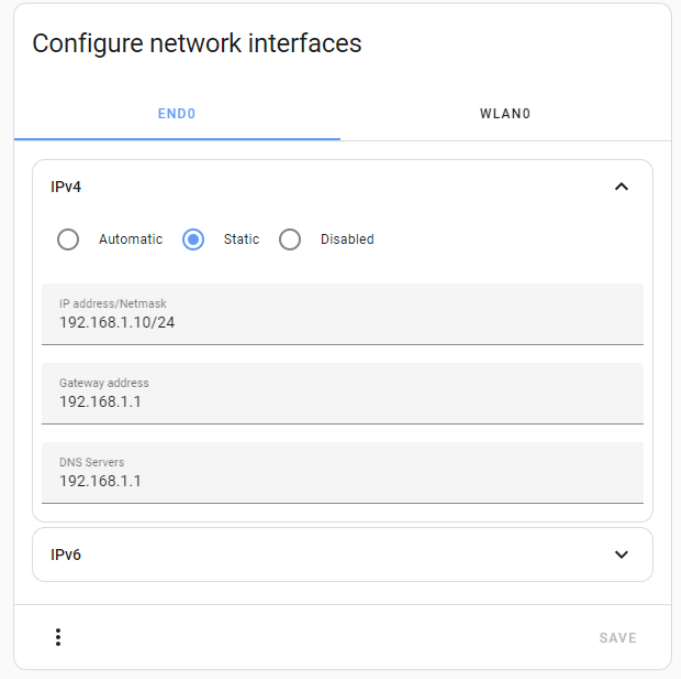
- Raspberry Pi 4 or 5
- Micro SD Card
- Pi Power Supply
- PC with Micro SD card reader

Setup Process:

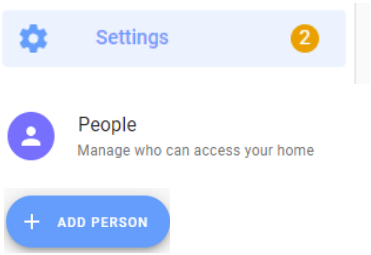
This guide will be at a higher level on how to set up Home Assistant (HA); detailed instructions are available at https://www.home-assistant.io/installation/raspberrypi for HA installation.

Step	Instruction
1.	Download the Raspberry Pi Imager from https://www.raspberrypi.com/software/ and install it
2.	In the Raspberry Pi Imager ● Select your Pi Version ● Select HA as your operating system ○ Other specific-purpose OS > Home assistants and home automation > Home Assistant. ● Select your SD card as the storage target

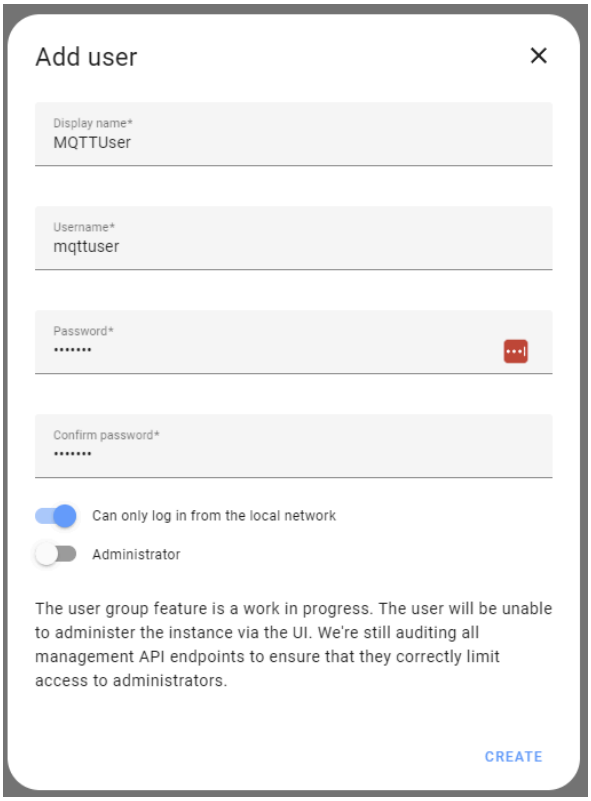
	• Click next to write the image to the SD card 
3.	Insert the SD card into your Raspberry Pi and connect the Pi to your network via an ethernet cable. Plug in the Pi to power it on.
4.	After the Pi has booted using a computer on the same network in your browser, go to homeassistant.local:8123 or check the settings on your router to find the HA IP address and go there directly. http://xxx.xxx.xxx.xxx:8123
5.	Click Create My Smart Home and wait for the process to complete
6.	Create your main admin user, for the prototype, we just used Student, Student.
7.	Continue following the prompts until you hit finish.
8.	Set a static IP address. Use something in the subnet for your current network. We used 192.168.1.10, but this will depend on your local network. • Select Settings • System • Network • Enter your network details under IPv4 and click save (you could also set up a WiFi here) 



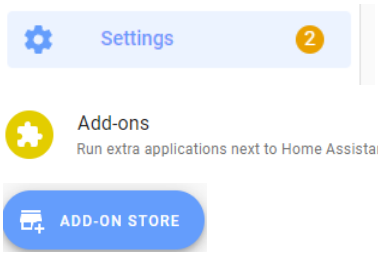
9. Setup a new User for MQTT



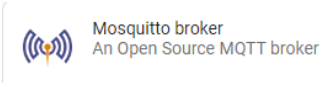
- Select Settings
- People
- + ADD PERSON
- Select can log in
- Enter name "MQTTUser"
- Password (we used Student)
- Click Create



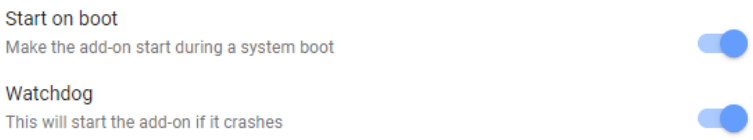
10. Setup Mosquitto broker



- Select Settings
- Add-ons
- ADD-ON STORE



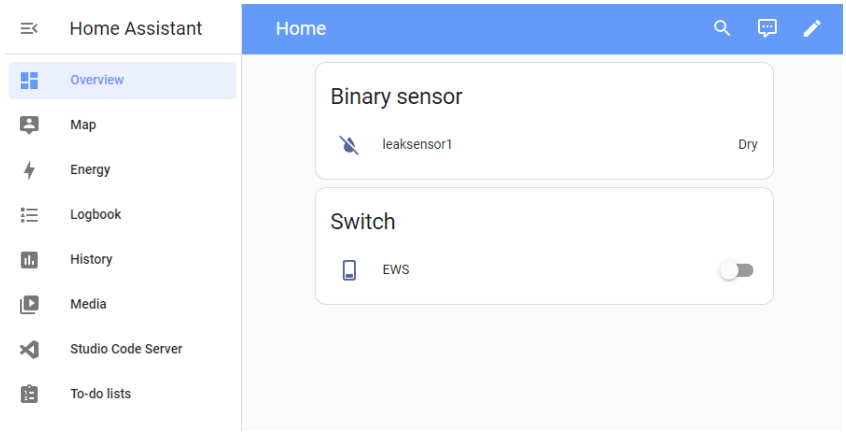
- Find or search for Mosquitto broker
- Click Install
- Once installed, make sure Start on boot and Watchdog are turned on



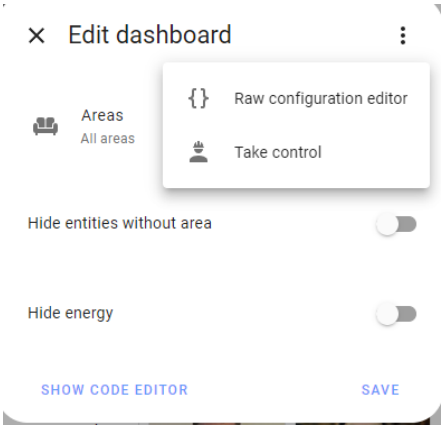
- Click start if it hasn't started automatically.

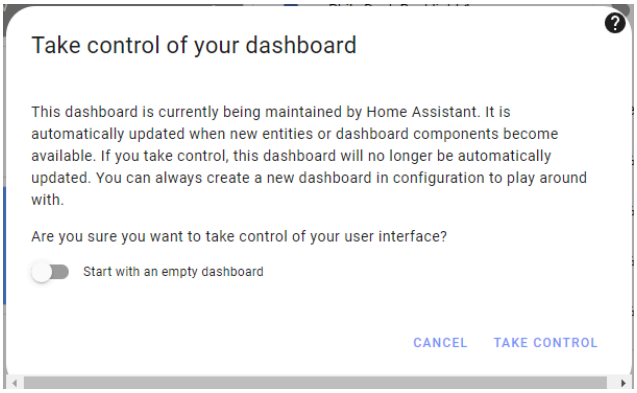
11.	This concludes the basic HA setup; the remaining steps will need to wait until the EWS and Leak Detector are online.

Configurer EWS and Leak Detector

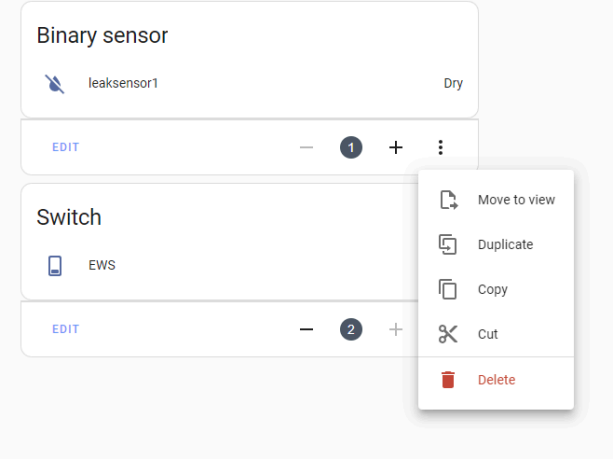
1.	Once the EWS is Online and connected to the MQTT broker, this should happen automatically if the steps have been followed or if you are using a different IP address than 192.168.1.10; update the broker IP address in the EWS and leak detector.
2.	Click on the Overview Dashboard your sensors should be shown with their current status. Note that there may be other devices that HA automatically added during setup. Optionally you can remove these. 
3.	You can press the on/off toggle for the EWS and view the leak sensor status here.

Optional: Remove other entities from dashboard

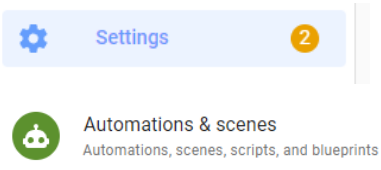
1.	• Click the pencil icon in the top right • On the edit dashboard menu, press the three-dot menu and select take control.  • Click Take Control

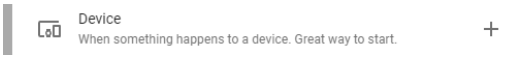
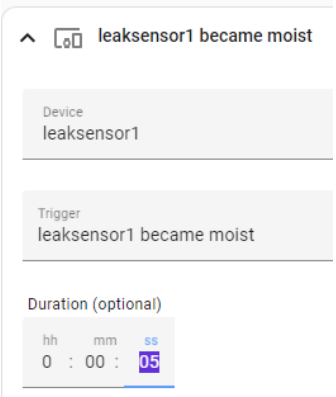
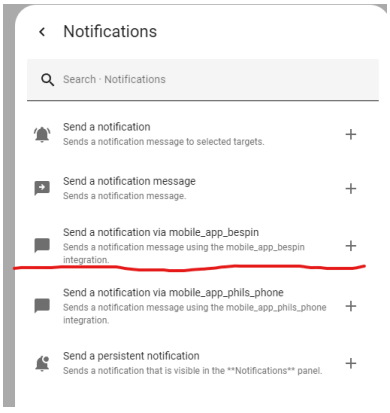


- While in the edit mode, click the 3 dots on anything you don't want and click delete.

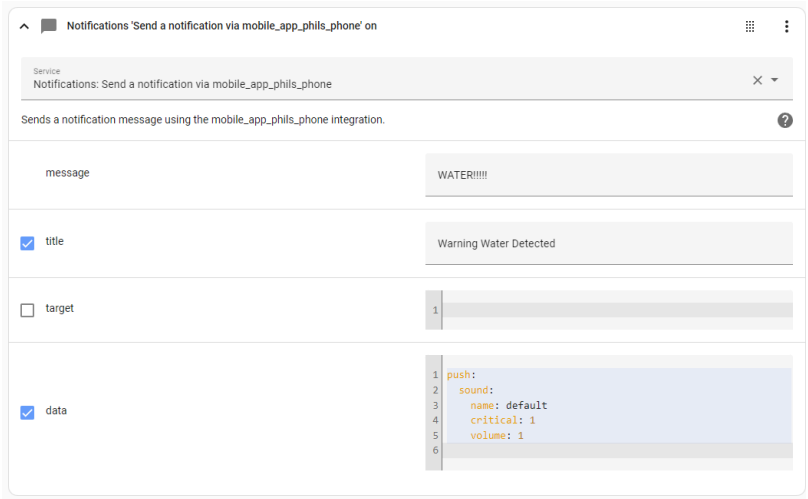


Optional Setup Notification Alert

1.	If you have an iOS device, you can set push notification alerts.
2.	Download and install the Home Assistant app from the App Store.
3.	Open the app, it should detect your home assistant instance; enter your username and password. You may need to give the app permission to push notifications in the IOS settings.
4.	Open Automations 1. In HA Select Settings 2. Automation & Scenes 

	3. + CREATE AUTOMATION  4. Select Create new automation
5.	1. Click + Add Trigger 2. Select Device  3. Pick the leak sensor from the menu 4. Set a duration of 5 sec (wait for the status to be moist for 5 seconds before triggering.  5. Under Then do click+ Add Action 6. Select Notification 7. Select the name of your iOS device  8. Enter Message Water!!!!! 9. Title Warning Water Detected 10. Under data add the following push: sound: name: default critical: 1

volume: 1



11. And then Click save
12. Test by putting the leak sensor in water, you should receive a notification.

Setup EWS

This guide assumes you are familiar with the Arduino platform and Arduino IDE

Required Hardware:

- Arduino Nano 33 IoT
- G3/4 Water Flow Sensor (Seeed Studio)
- 12V Solenoid Valve - 3/4" BSP
- Duinotech Arduino Compatible Logic Level Converter Module
- Breadboard Power Supply 5V/3.3V
- 9V to Barrel Jack Adapter (cut off as we are only using the jack)
- LCD1602 RGB Module, 16x2 Characters LCD, RGB Backlight, 3.3V/5V, I2C Bus
- LM2596 Adjustable Voltage Regulator 4.0-40V to 1.25-37V DC
- Breadboard
- Narva Electrical Terminal Connector Strip (Optional)
- Universal Adapter Power Supply 30W AU 2A,100-240V to 3V,4.5V,5V,6V,7.5V,9V,12V,Adjustable
- 16mm Illuminated Pushbutton - Green Momentary
- 1000k resistor
- Jumper Wires

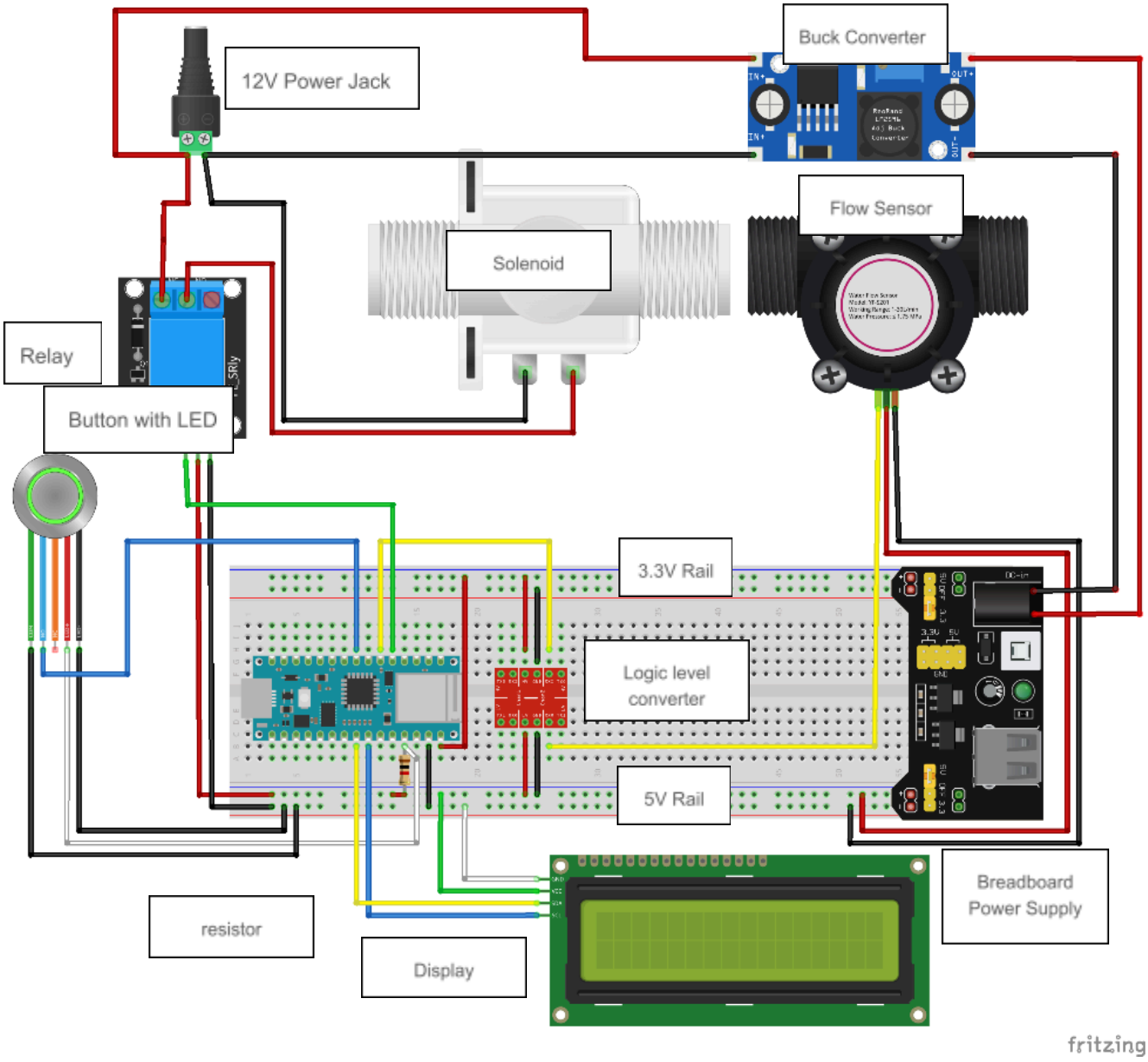


Fig. 9 - EWS fritzing diagram

Assemble the EWS hardware as per Fig. 9, paying close attention to what connects to the 5v and 3.3v Rails; before connecting the barrel jack to the buck converter, set the voltage; see details below. I used 8V as the breadboard power supply accepts 6-9V

The logic level converted HV side needs to reference the 5V power rail, and the LV side needs to reference the 3.3V power rail.

The Relay can be on the positive or negative side of the solenoid, and the solenoid can be wired either way as it does not have a direction. You should use the NO (Normally Open) and COM on the Relay for the connection.

Pins for Nano 33

D2	Relay
D3	Flow Sensor (this must be an interrupt pin, don't use 4)
D5	Led
D21	Button
SDA	To Display SDA and SCL pins
SCL	

Set Buck Converter

Press the right-hand button so the out LED is lit; use a screwdriver to turn the potentiometer counterclockwise to step down the voltage until it hits 8V

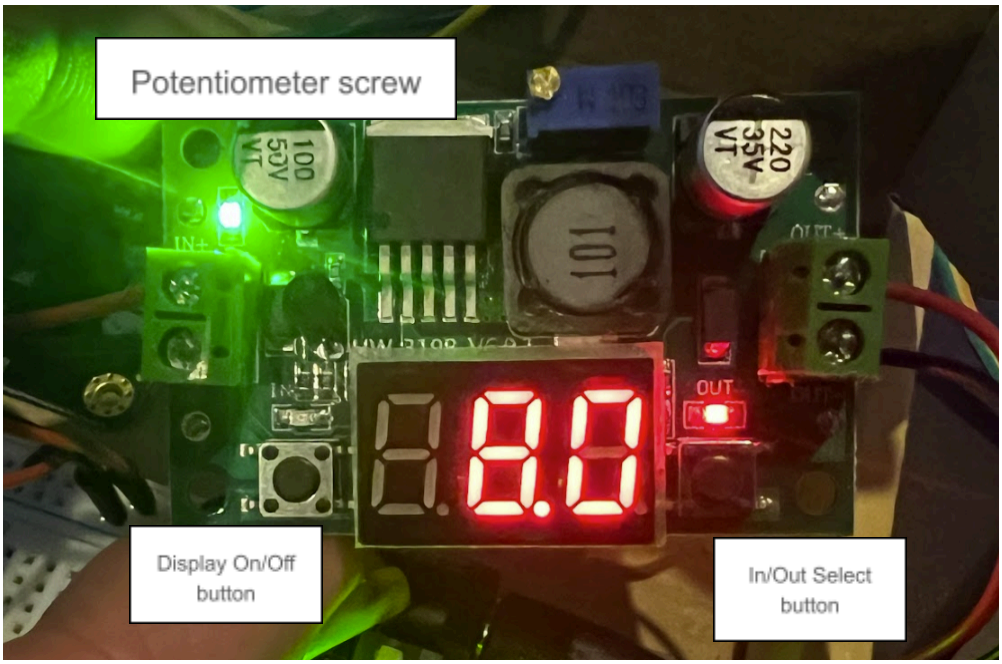


Fig. 10 - Buck Converter

The final unit should look similar to Fig. 11

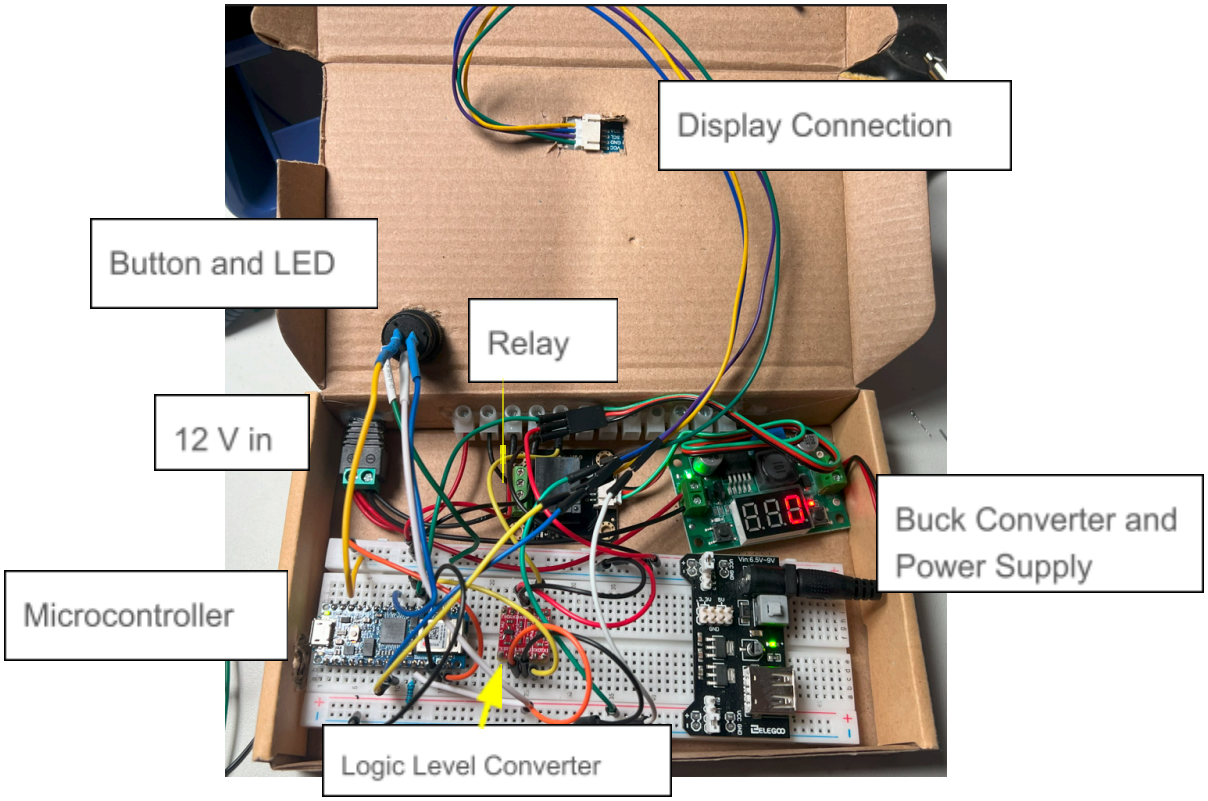


Fig. 11 - EWS Main Unit Internals

Once Assembled, download the EWS code from GitHub, Create an **arduino_secrets.h** file and add your wif and MQTT details

```
#define SECRET_SSID "MyWifi"
#define SECRET_PASS "WiFiPass"
#define SECRET_MQTT_USER "MQTTUser"
#define SECRET_MQTT_PASS "student"
```

Note if you are using a different IP address for HA than the default (192.168.1.10) you will need to update the broker IP on line 27.

Make sure you install these libraries:

- SafeString by Matthew Ford
- FlowSensor by hafidhh

- ArduinoMqttClient by Arduino
- WiFiNINA by Arduino
- Waveshare_LCD1602_RGB.h manually download and install from the manufacturer: https://files.waveshare.com/upload/5/5b/LCD1602-RGB-Module-demo.zip

Push the code to the nano 33 IoT using the Arduino IDE. You should see the unit connected to the WiFi and MQTT server from the display output you can also test if the Relay is working by pressing the button; the LED should light up, and the display should also update to show the water is turned on. Also, make sure the 12V power supply is connected, it is not always necessary to test the Relay. Further testing will require the pump assembly or other water supply.

Leak Detector Setup

Required Hardware

- Arduino Nano 33 IoT
- 3 x Resistors 1000k
- 2 x Led, (Green, Red)
- Piezo Buzzer
- Simple Water Detection Sensor with Digital Output
- 1 x NPN Bipolar Transistors (PN2222)

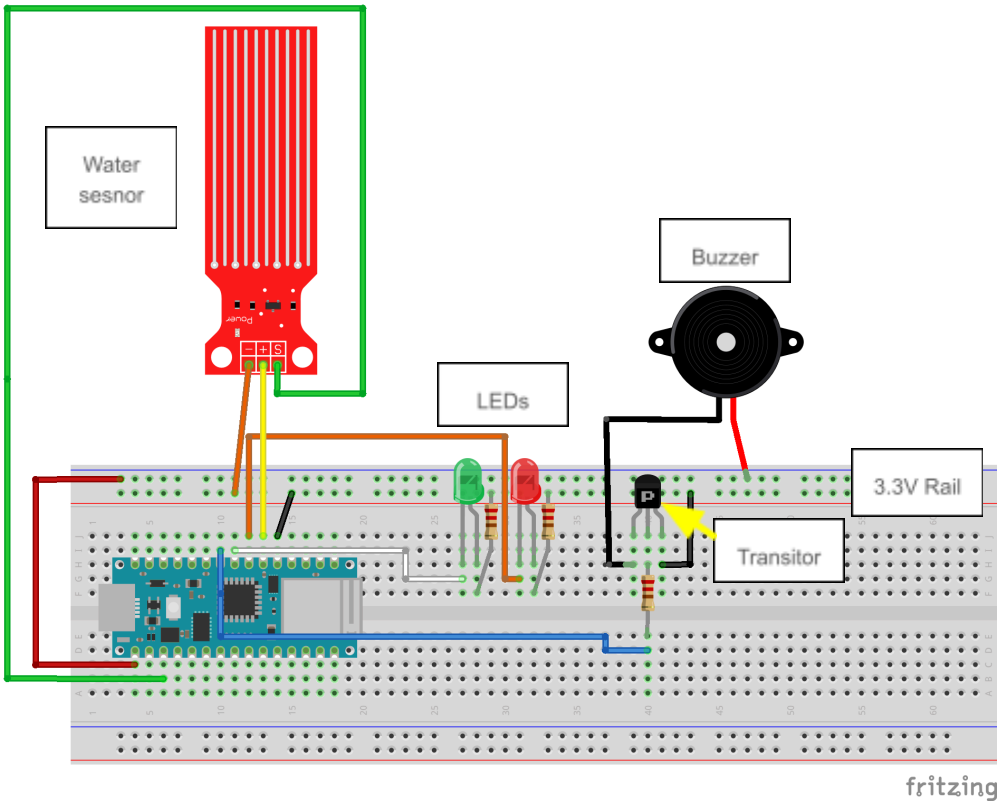


Fig. 12 - Leak Detector wiring layout

Assemble the leak detector as per Fig. 12; once done, it should look similar to Fig. 13. Note on the Transistor, the middle pin connects back to the Nano; the outside pins should be used between the ground pin on the buzzer and ground.

Connect the pins as per the following:

Pins for Nano 33

D2	Sensor Power
D3	Alert LED
D4	Connection LED
D5	Buzzer
A1	Sensor data Pin

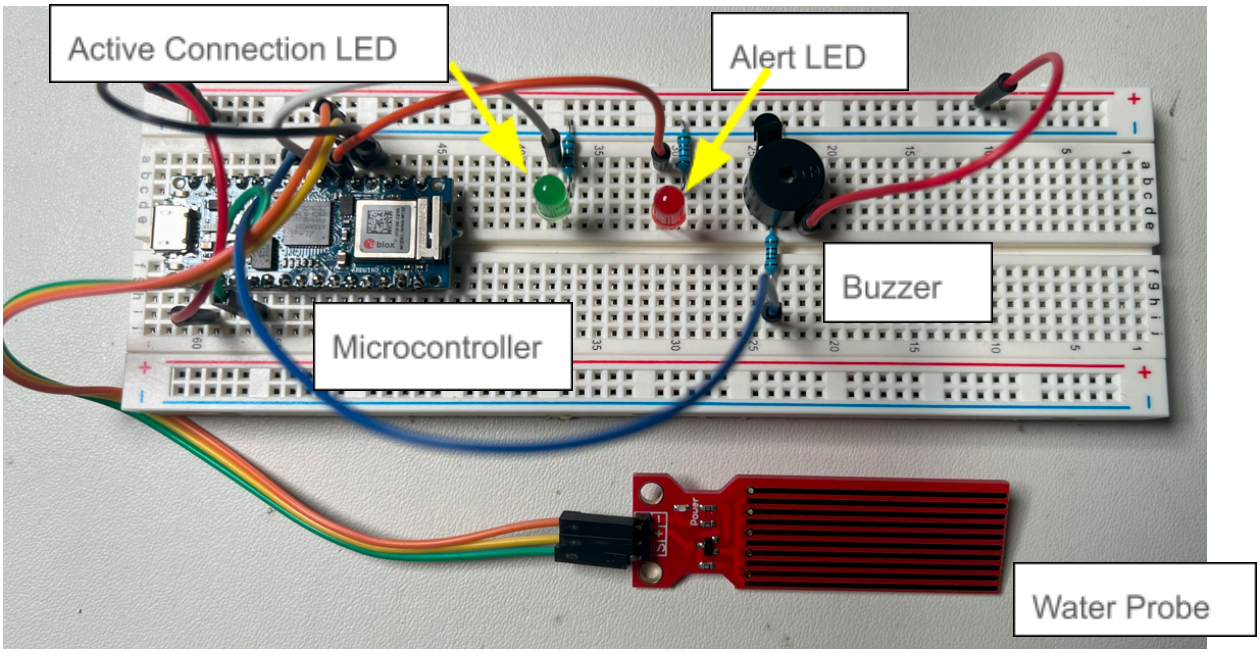


Fig. 13 – Leak Detector Physical Layout

Once Assembled, download the Leak Detector Code from GitHub, Create an **arduino_secrets.h** file and add your WiFi and MQTT details

```
#define SECRET_SSID "MyWifi"
#define SECRET_PASS "WiFiPass"
#define SECRET_MQTT_USER "MQTTUser"
#define SECRET_MQTT_PASS "student"
```

Note if you are using a different IP address for HA than the default (192.168.1.10) you will need to update the broker IP on line 24.

Make sure you install these libraries:

- SafeString by Matthew Ford
- FlowSensor by hafidhh
- ArduinoMqttClient by Arduino
- WiFiNINA by Arduino

Upload the code to the Nano, and leave the Nano plugged into the computer or other USB power supply. If your sensor is alerting continuously, look at changing the trigger threshold on line 61. Theoretically, the sensor should read 0 when there is no water, but I found it tends to read about 20 and sometimes spikes when moving, likely due to poor connections.

```
const int waterTriggerLvl = 70;
```

Look at continuing the setup in Home Assistant.

Water System

The water system method is somewhat optional and can be simulated using a house connected to a tap on your home, but it would potentially waste a large amount of water.

To build a comparable system to mine but simplified, these are the core components you would need you could potentially use a different pump, but it needs enough pressure to open the solenoid.

Basic Hardware:

- 2 x Philmac 3/4" BSP Thread Pipe Socket (used to connect the solenoid and flow sensor)
- Ozito 350W Dirty Water Submersible Water Pump
- 19mm Hose
- Toro 19mm Tail x 25mm BSP Male Poly Director
- Philmac 1-1/2" BSP Thread Pipe Socket
- Philmac 1-1/2" x 1" BSP Thread Pipe Bush (this might be incorrect; double check before purchase)
- Toro 19mm Tail x 20mm BSP Male Director - Single
- All Set 100L Grey And Green Heavy Duty Storage Container With Flat Lid
- 2 x Kinetic 17-32mm 304 Stainless Steel Hose Clamp

With these components, you can interface the pump, the solenoid and the flow valve; any additional plumbing fixtures can be added at your discretion. Remember to use Teflon tape to help avoid leaks. See Fig. 14 and Fig. 15 for connection order.

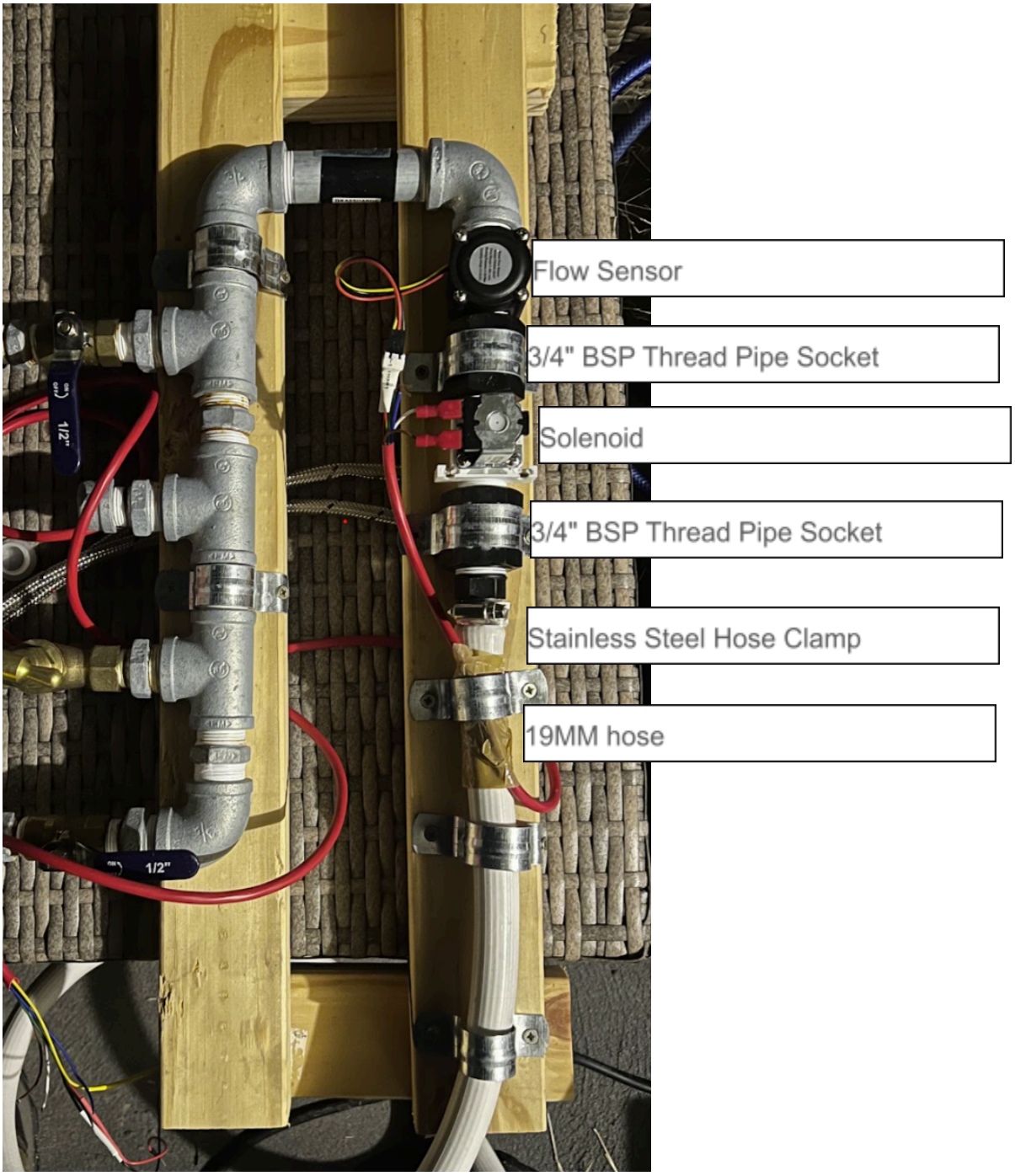


Fig. 14 - Plumbing connections

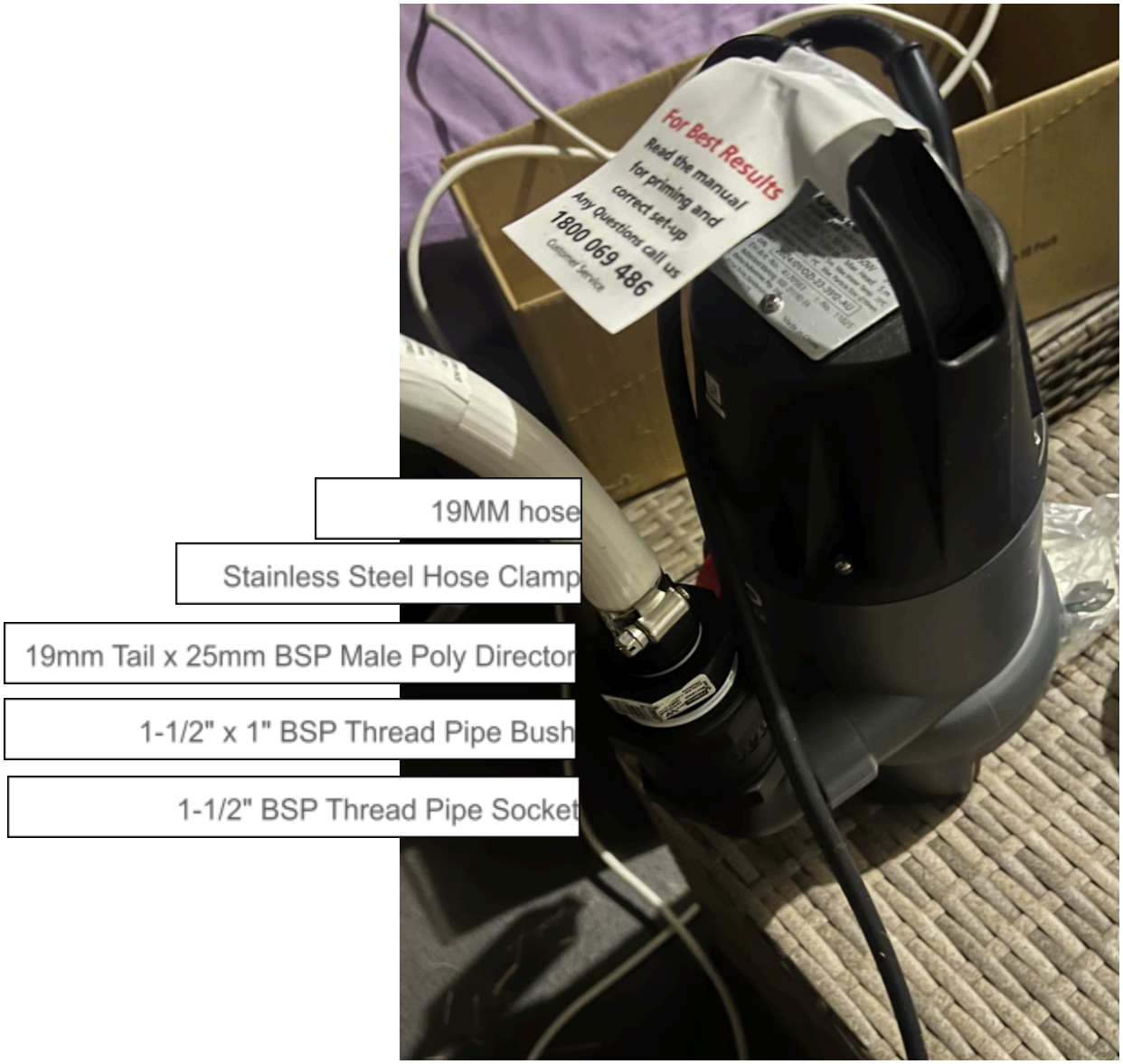


Fig. 15 - Water Pump Connections

7. VIDEO LINKS

Link to YouTube Demo Video: https://youtu.be/g38VAFEXvCA

Link to Detailed Software Breakdown: https://youtu.be/Y4pMxJxQces

8. CONCLUSIONS

At the end of the day, I achieved what I set out to do, which was to develop a system that could monitor the flow of water and, when necessary, react to turn off the water supply. Overall I'm proud of what I achieved, although I wanted to get more out of the project, like building an internally hosted webpage on the Nanos for making configuration changes; apparently, you can access the ESP32s flash storage for extra space, but there is only so much time. I do feel I achieved all the technical outcomes required to achieve the highest marks, and integrating into a smart home platform was a good idea as I can leverage its UI, reporting, automations and notifications that would have required my own app.

If I were doing the project over I would simply the plumbing system and just use one tub to cycle the water, maybe one fixture. I put far too much time, effort and money into that part, which had little to do with the embedded itself. The elaborate setup was because the original intent was to model the data and see if I could detect the difference between different fixtures based on water flow. The core development had a few curve balls, like the button not toggling correctly or triggering when I turned it off/on from the remote system. Adding a debounce timer helped a lot; otherwise, I think capacitors are needed to help smooth the power out; there were half a dozen other little issues getting everything working. I would have liked to have the WDT working on the EWS, but interference with the flow sensor (caused NAN values) is an issue I may have to pick apart the libraries to see if I can overcome it.

Overall, the experience was great, and I think I have a new passion for building embedded systems.

REFERENCES

[1] QBE. (2021, 21 Sep 2021) "Water damage insurance claims remain high despite more Aussies working from home." QBE. [Website]. Available: https://www.qbe.com/au/media-centre/press-releases/water-damage-claims-remain-high

[2] "AquaTrip." [Website]. Available: https://www.aquatrip.com.au/

[3] A. S. Meikle, "A fluid flow monitor," Australia Patent Appl. 2006257712, 14-June-2006, 2006. [Online]. Available: https://ipsearch.ipaustralia.gov.au/patents/2006257712