Water Level Proposal and Recommendation Report

Tyler Eley

Summer 2020

Intro:

Water is essential to the health and wellbeing of all living things. As such, any improvement to infrastructure related to water can have wide reaching effects. Ensuring a supply of water to animals and plants is a crucial part of animal husbandry, gardening and ranching.

A common task related to this is refilling water troughs to ensure animals and plants have plenty of water. This process can be improved by automatically informing the user when the water in the trough is in need of refills

Systems Needed for project

Microcontroller

Water Level Detection at between two levels

Transceiver to transmit from field unit to base unit

Solar Power to power field unit

IoT System

Housing

System Breakdown

Microcontroller:

Ovieview: We need a way for software to interact and control all hardware systems

System Implementation: I am using two Arduino Nanos.

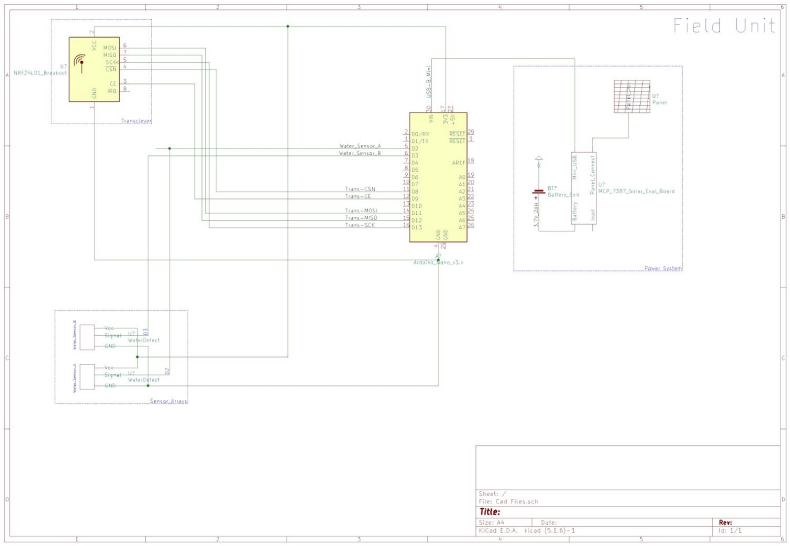


Figure 1: Field Unit

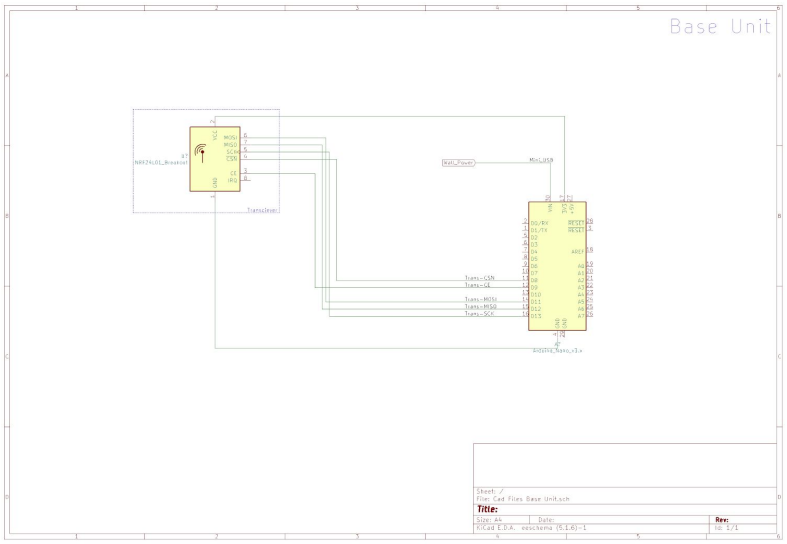


Figure 2: Base Unit

Water level detection

Overview: The design needs to detect water when the water level is in need of refill. This can be defined as when the water is below a *refill* level but above a *critical* level.

This can be achieved by using two water sensors, defined as sensor A and sensor B. We will send one of three types of signals at some regular basis as defined by the user.

The main hardware limitation is ensuring the system can operate while submerged indefinitely. The system needs to be waterproof, and corrosion proof.

System implementation:
Water Normal= Water is detected at A and B

Wn= AB

Water Low = water is detect at B and not A

Wl = A xor B

Water Critical = Water is not detected at either

Wc = $\underline{AB}$

Water Levels Truth Table				
Water Sensors at Field Unit		IntState sent to Base Unit		
A	B	Wn (Normal) [State=0]	Wl (Low) [State = 1]	Wc (Critical) [State = 2]
0	0	0	0	1
0	1	0	1	0
1	0	0	1	0
1	1	1	0	0

Figure 3: Possible Water Levels that can be detected.

Hardware Implementation and Options:

1) 2x CQRobot Water Detection Module ($9.99 ea, $19.98 total).

Specifications:

- Type: Photoelectric Level Switch
- Power Supply Voltage: DC 5V
- Output Current: 12 (mA)
- Working Temperature: -25 Degree Celsius to +105 Degree Celsius
- Low Level Output: Less Than 0.1V
- High Level Output: 3.3V or 5V (Dial Switch Control)
- Liquid Level Detection Accuracy: Plus/Minus 0.5mm
- Material: PC
- Measuring Range: NO Limit
- Life: 50,000 Hours
- Scope of Application: Water Level Control and Protection of Electrical Products
- Probe Line Length: 50cm
- Dimension: 30mm * 30mm
- Mounting hole size: 3.0mm

Pros:

- Completely Waterproof
- Corrosion Resistant
- Easily interfaced with arduino
- Probe works in any orientation

Cons:

- High price point
- Direct sunlight may cause interference (which could lead to false positives and false negatives for water detection.)
    - Can potentially be averted if a sheild is designed around the sensor or if the sensor is out of sunlight

2) 2x Rain Water Level Sensor Module Depth of Detection Liquid Surface Height for Arduino (pack of 10 for $6.19)

Specifications:

- Product Name: Water level sensor
- Working voltage: DC3-5V
- Working current: less than 20mA
- Sensor Type: Simulation
- Detection area: 40mm/ 1.57''x 16mm/0.63''
- Production process: FR4 double-sided spray tin
- Working temperature: 10 ° C - 30 ° C
- Working humidity: 10%-90% non-condensing
- Product weight: 3.5g
- Product size: 65mm/2.56'' x 20mm/0.79'' x 5mm/0.2''

Pros:

- Very cheap
- A lot of documentation

Cons:

- Mainly used for soil level detection - not designed to be submerged
- Prone to corrosion - amazon review show the leads corrode after just three weeks of nonstop contact with water.

Transceivers

Overview: There needs to be a way to wirelessly send the current state (either Wn, Wl, or Wc) from the field unit to the base unit.

For this, I propose using two NRF24L01 Wireless Transceiver Modules with one transceiver module connected to each unit.

System Implementation:

Pin on Transceiver	Wire Color	Pin on Arduino Nano
1 (ground)	Black	GND
2 (power)	Brown	3v3
3 (CE)	Red	d9
4 (CSN)	Purple	d8
5 (Required to be attached to d13)	Blue	d13
6 (Required to be attached to d11)	Yellow	d11
7 (Required to be attached to d12)	Green	d12

Figure 4: Wiring Guide

Hardware Implementation:

- 2x AIDEEPEN Transceivers ($6.99 for 2)
- Built-in 2.4Ghz antenna
- Maximum operating speeds up to 2Mbps
- Suitable for industrial control applications.
- Built-in hardware CRC error detection, Multipoint communication address control.
- Built-in voltage regulator
- 125 Channels
- Low-power 1.9 ~ 3.6V, only 1uA on Power down mode.

Software Implementation:
The field unit sends an int with a value ranging from 0 to 2 (See truth table in the Water Level Detection section).

Power Design:

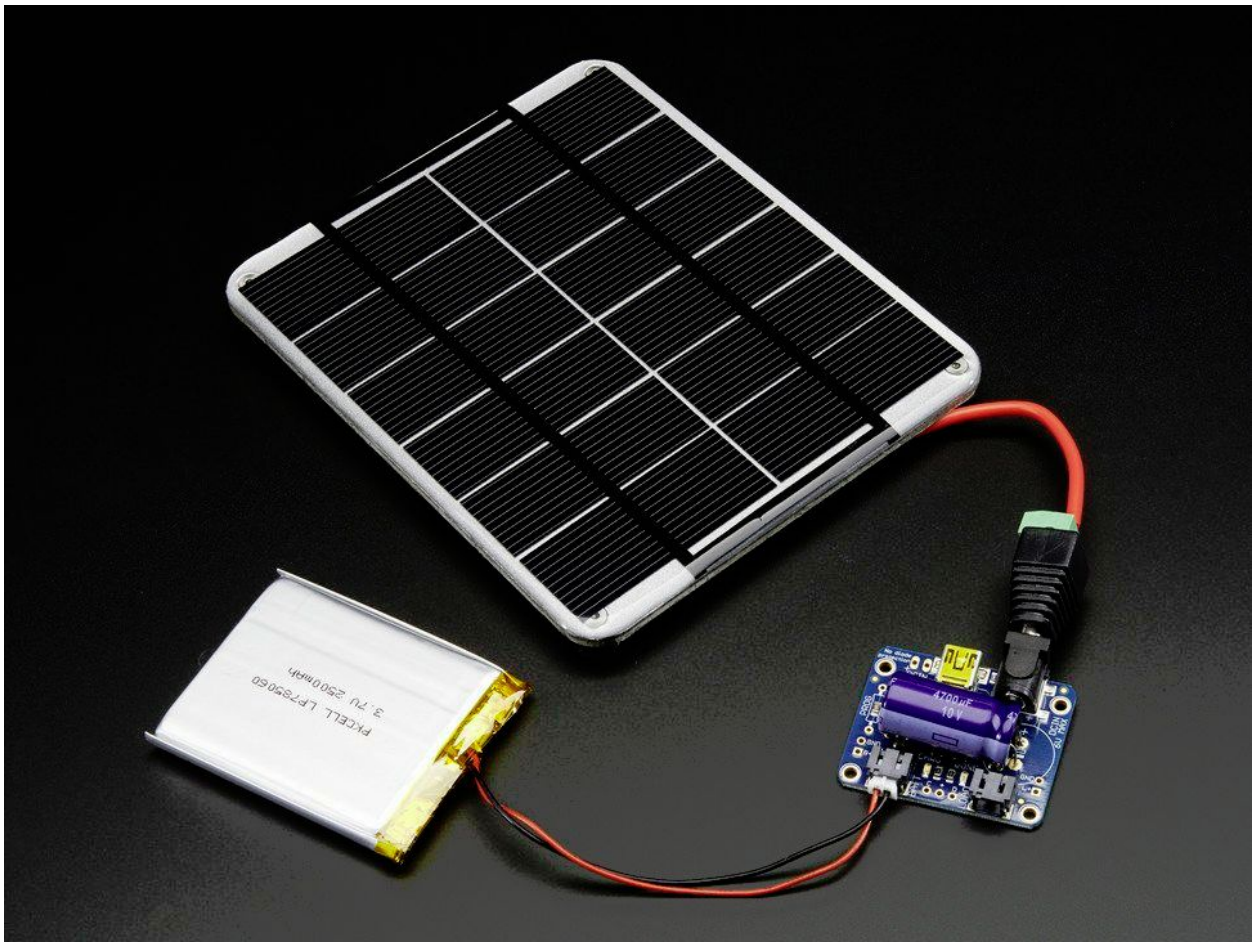


Figure 4: Adafruit Solar Charger setup

Overview:

Needs to be able to supply power to the field unit.

Max assumed power usage = 0.56 Whrs

Materials

Battery: Adafruit 3.7V 2AH [Datasheet]

Panel: MONOCRYSTALN SOLAR CELL 2.27W 7V

Charger Adapter: 3.8/1.3mm DC Jack Cable

Solar Charger: MCP 73871 Battery Charger Solar Powered Eval Board

IoT System:

Overview: IoT (Internet of Things) is a methodology for connecting personal devices to the internet. In order to do this, a Arudino Nano 33 IoT board is used, which connects to the Arduino IoT Cloud. This allows the Base Unit to connect to the internet via wifi or bluetooth.

An easy way to access the data for the device is directly through the Arduino IoT cloud website (which can be accessed on a web browser), however the Arduino IoT Cloud API enables the use of information from the Arduino IoT Cloud in apps, however I will need to sign up for Apple's Development, additionally this would no doubt be a difficult programming project for my current skills.

An even easier way to access this data is to use webhooks. Webhooks allow the arduino to send data to other websites and services. This can be used to send the data directly to google's services (for example, track states overtime with google sheets and send notifications when states change using Gmail.)

Housing:

Overview: The Field Unit will need housing that is waterproof and shockproof and durable. While it is important to protect the arduino, it is more important to protect the battery, as lithium ion batteries can catch fire or explode if not handled correctly.

Based on the datasheet for the battery used, temperature should not be a problem, as the batteries operating temperatures are 32F to 113F for charging and -4F to 160F for discharging.

Materials:

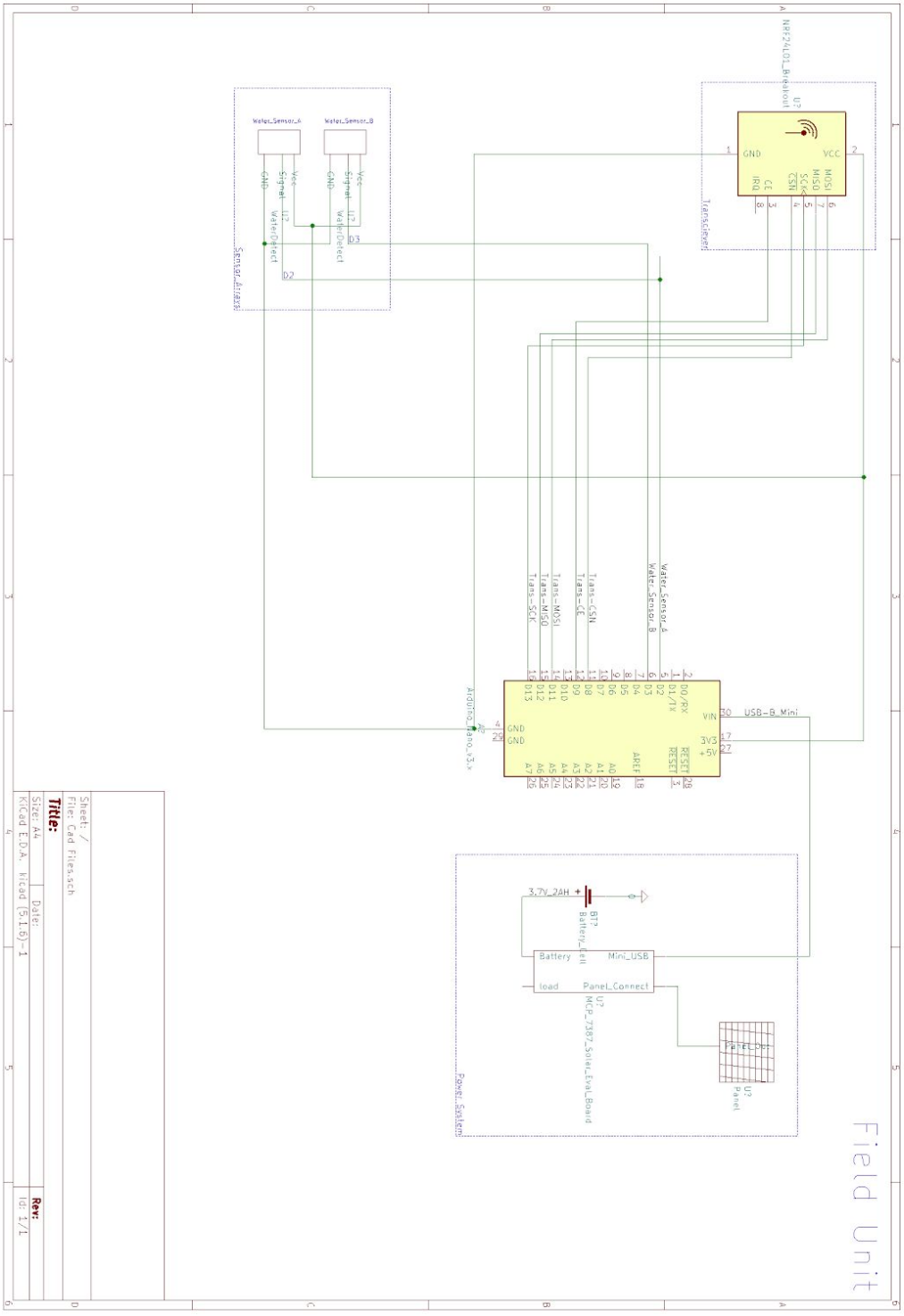
Housing: SOCKiTBOX

Appendix I - Costs and Inventory

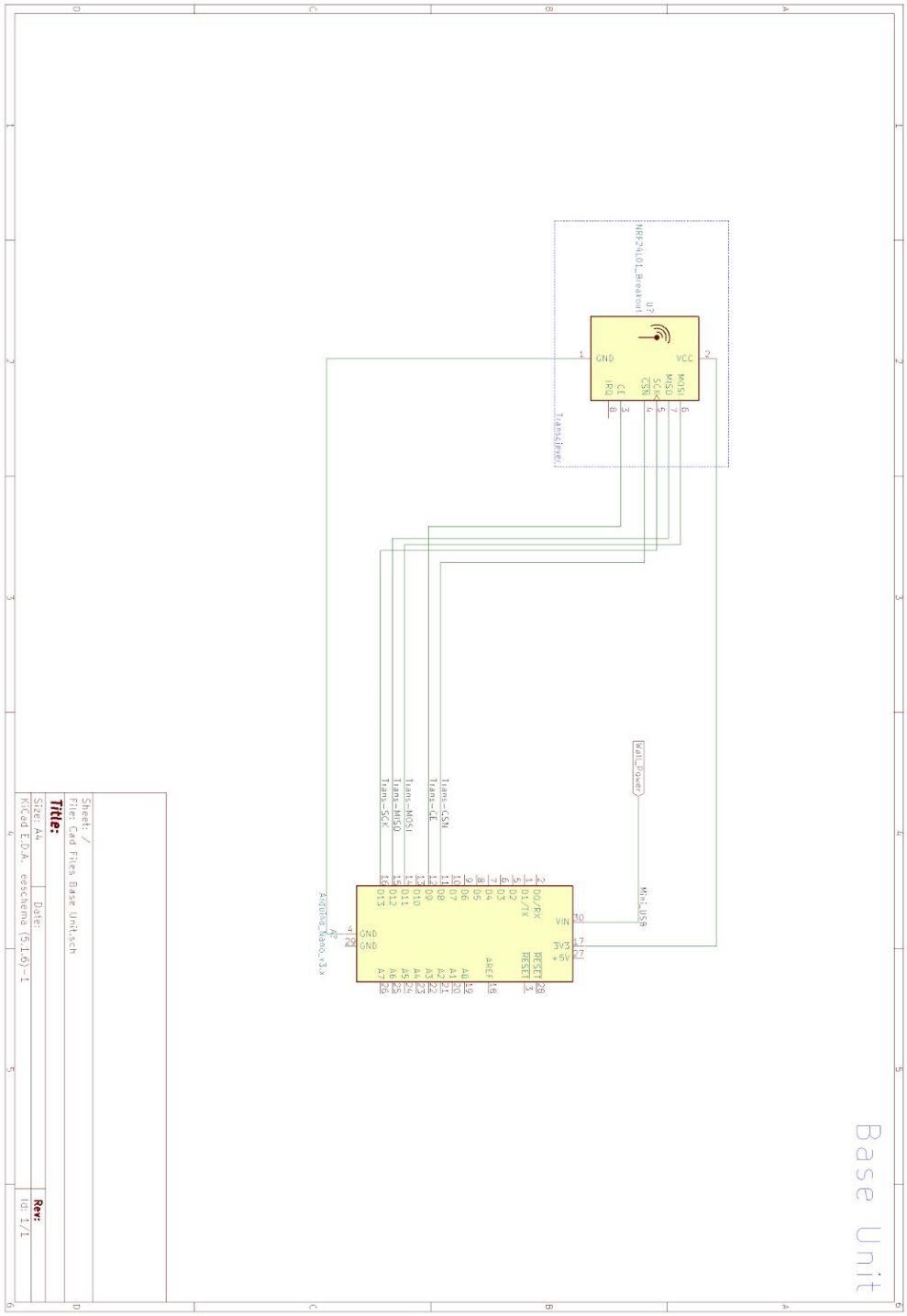
Cost and Inventory						
Item	Name	Cost	Purchased	Date Purchased	Options	Recom.
1	ELEGOO Arduino Nano x3	$19.98	Yes	May 19 2020		
2	Aideeepen Transcievers X2	$6.99	Yes	May 19 2020		
3	Jumper Cables	$5.79	Yes	May 26 2020		
4	Arduino Nano 33 IoT	$29.21	Yes	June 24 2020		
5	CQRobot Water Detectors x2	$19.98	No			
6	Generic Arduino Water Sensor X10	$6.19	No			
7	Adafruit 3.7v 2AH Battery	$12.50	No			
8	Adafruit MonoCrys Solar Cell	$29.00	No			
9	Adafruit DC Jack Adapter	$2.99	No			
10	SockitBox	$22.99	No			
11	PCB x5	$7.00	No			
12	Adafruit Solar Charger	$17.50	No			

Current Purchased Cost	$61.97
Recommended Design Cost	$173.93
Cheapest Design Cost	$160.14

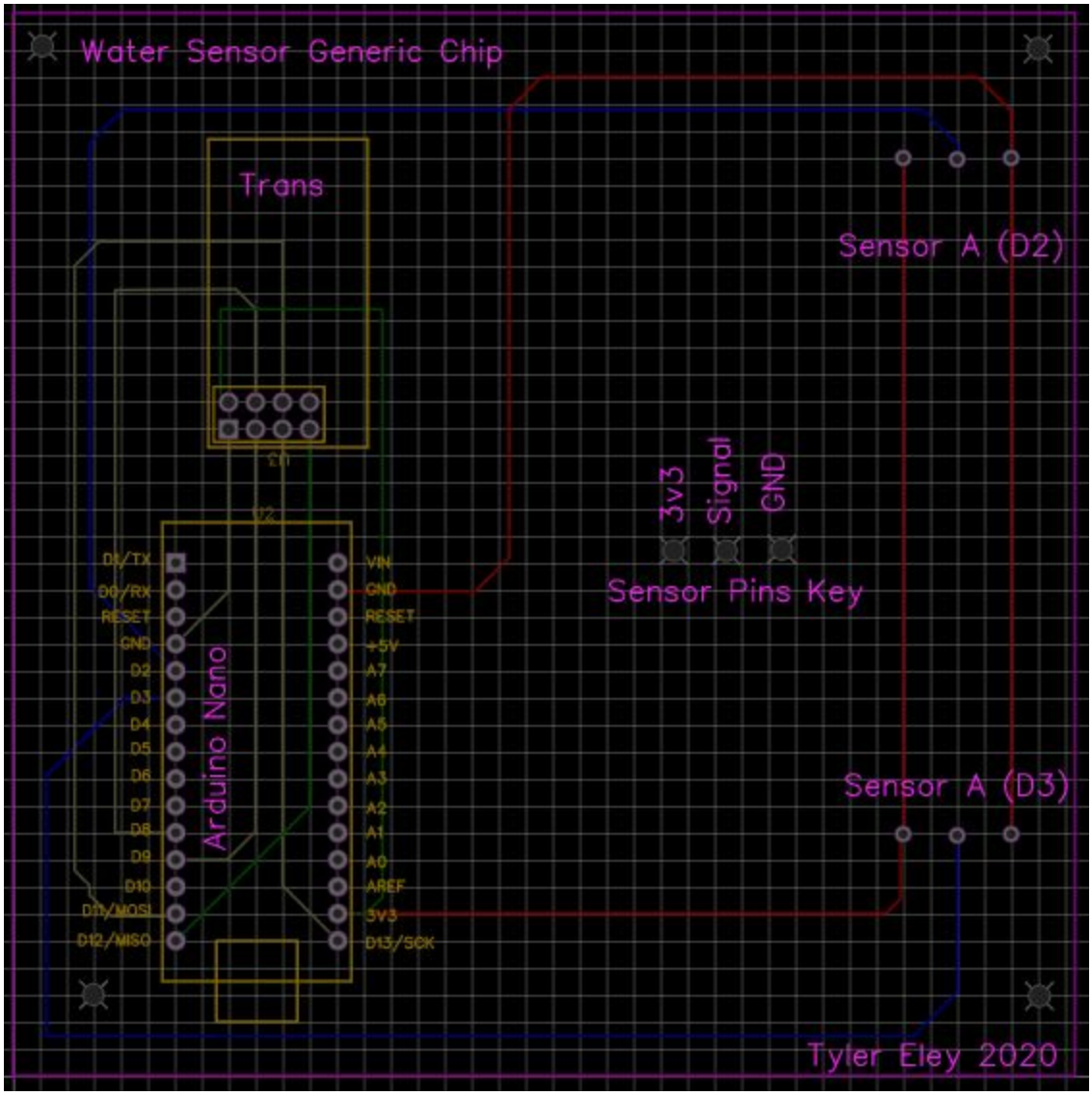
Appendix II - Field Unit



Appendix III - Base Unit Schematic

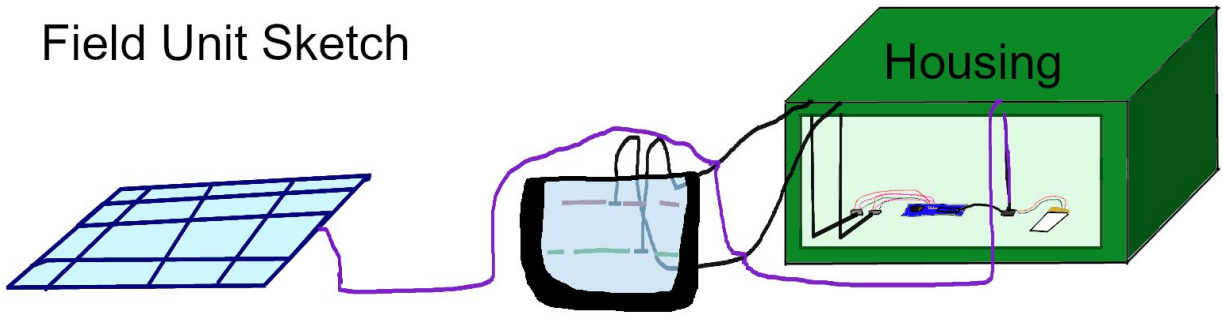


Appendix IV - PCB



While it is not strictly required, for the device to function, it is advised to use a PCB to hold everything together. This directly connects all components on a single board rather than using jumper cables to connect several boards.

Appendix V - Field Unit Sketch



Housing - Contains PCB (with arduino and transceiver), battery, battery charger

Trough - Contains two water sensors, each placed at different depths

Panel - Solar Panel

Appendix VI - Testing

Field Unit water detection Test - Success (7/17/2020)

This test demonstrates that the field unit is successfully able to detect water in all possible scenarios.

To run this test, set the boolean variable *waterTest* to 1.

Output:

When neither probe is submerged, output is 2.
When one but not both probe is submerged, output is 1.
When both probes are submerged, output is 0.

```
11:49:36.499 -> 0
11:49:37.531 -> 0
13:01:11.972 -> 2
13:01:12.993 -> 2
13:01:14.044 -> 2
13:01:15.071 -> 2
13:01:16.101 -> 2
13:01:17.126 -> 1
13:01:18.149 -> 1
13:01:19.174 -> 1
13:01:20.195 -> 1
13:01:21.221 -> 1
13:01:22.277 -> 1
13:01:23.307 -> 1
13:01:24.332 -> 0
13:01:25.359 -> 0
13:01:26.382 -> 0
13:01:27.402 -> 0
13:01:28.460 -> 0
13:01:29.482 -> 0
13:01:30.501 -> 0
13:01:31.522 -> 0
13:01:32.579 -> 0
13:01:33.606 -> 2
```