Grid Independent Water Monitoring/Management System

Anantajit Subrahmanya
Grant McKechnie
Jayanth Meka

October 18, 2023

Date	Revision	Release Notes
	Rev 01	Initial Release

Contents

1 Introduction

1.1 Problem

Current irrigation systems offer a way to set timers for scheduled watering. These solutions require a wired connection to an electrical controller and power source, which limits the locations in which irrigation systems can be installed.

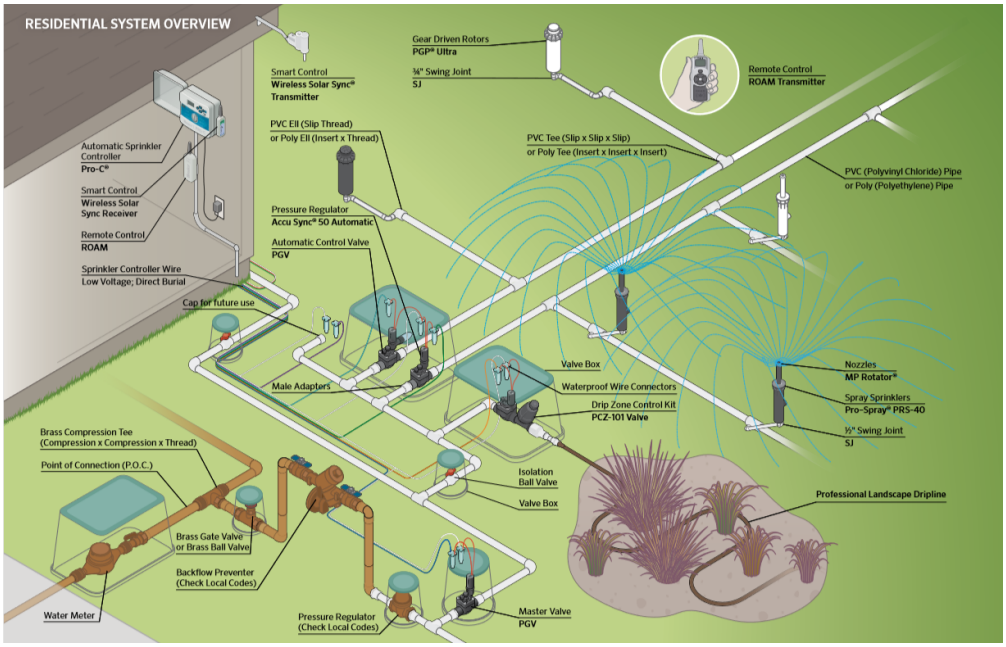

Figure 1: Typical Residential Irrigation System

As climate change strains drought-prone regions that are also subject to power outages [1], it becomes increasingly important not only to minimize total water consumption, but to also have irrigation systems that work without external power. Most systems require a central control unit that uses wires to communicate to all irrigation systems on a property. As you can see in Figure 1, this control unit is usually located within a home and is powered from a wall plug. Residents are able to set their own watering cycles and monitor water usage. This method is expensive to install and is at risk of damage that could cause the whole irrigation system to fail. The wires from the control unit are thin and usually located underground which makes them especially prone to damage from animals like moles or rats. If the irrigation system stops working, it is extremely difficult to know where the source of the failure is. There are solar powered irrigation systems on the market, but they do not completely solve the problem. Fully solar powered systems do not have a reliable source of power as solar is highly dependent on the weather.

1.2 Solution

We propose a solution to this problem in the form of a wireless self-powered irrigation system. Through the utilization of a water-powered turbine generator, we will power our

whole irrigation system which includes batteries, valve control, and watering timers. The goal of this project is to make it as self-contained as possible with an easy installation. If time permits, we would like to also modularize the design so that communication between different irrigation systems is possible. This system can still operate in the event of a power outage which is becoming increasingly common in rural areas during the dry seasons where water management is crucial. Our device will fit the form factor of existing plumbing couplings (elbows, nipples, and other joints); in other words, it will appear to be a connector between two pipes/hoses. The device will be equipped with a turbine that will generator power to charge a battery which will then power on onboard microcontroller. This microcontroller will be responsible for valve control for watering plants at a given interval. This interval will be initially set by us, but we hope to possibly incorporate soil sensing to enable real-time watering interval calculation. The microcontroller should also wirelessly communicate with a base station located nearby, allowing users to manually control the system if needed. The base station, which is powered by a wired DC source, can aggregate data from all nearby valves/sensors and present them to the user. Our solution will offer a reliable power source that is not subject to the weather, have less wires that could easily be damaged, and will be simply an attachment to existing plumbing rather than a full external system.

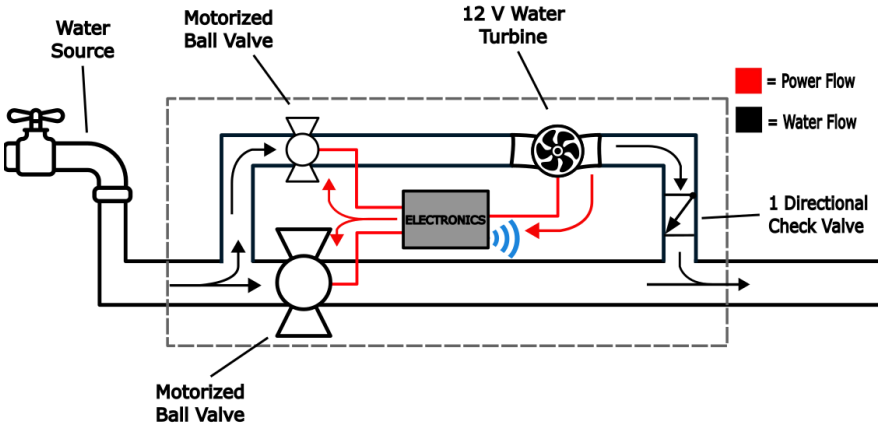

Figure 2: Visual Aid

1.3 High Level Requirements

1. The device must be able to open and close valves based on remotely-issued commands. We define an open valve as allowing at least 50% of the maximum water flow through a half-inch diameter pipe, and a closed valve as allowing ¡ 1% of the maximum water flow through a half-inch diameter pipe, for a reasonable water pressure of ¡30 gallons per minute. The opening action should occur with a maximum latency of 30 seconds from when the command is first issued by the user.

2. The device must be able to recharge the battery using water flow during water cycles. We define the act of "recharging" as the battery having more charge after a watering cycle than before the watering cycle. The assumptions from the previous high level requirements still apply.

3. The device must be able to operate for at least 48 hours on a single charge. Note that for testing this requirement during the demo, the device must also be able to measure the battery charge level. We will measure the battery drain over a 10 minute window. We can then perform a linear extrapolation to estimate the net battery life of the device from a full charge. Note that any assumptions from the previous high level requirements would still apply.

2 Design

2.1 Physical Design

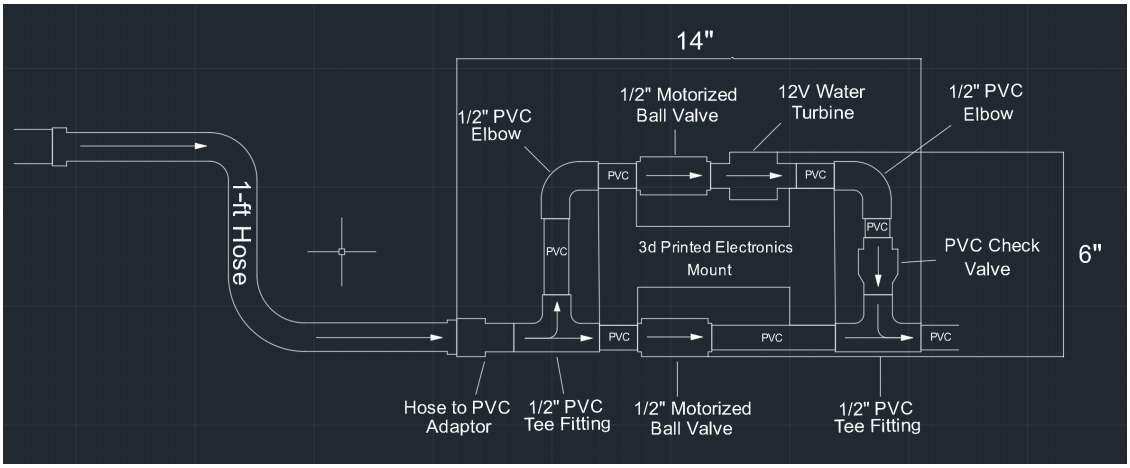

Figure 3: Physical Design Diagram

The physical frame of our self powered irrigation system will be a PVC based frame that incorporates a hose, two ball valves, a turbine, as well as our electronics. Starting from the water source, as seen in Figure 2, the Female Garden Hose (FGH) side of a 1-ft Water-Inlet Hose will be attached to the water source either directly or through the use of an adaptor, depending on the what water source is being used. The Male Garden Hose (MGH) end will be attached to 3/4" FH Thread x 1/2" Slip PVC Fitting. The other end of the fitting will have a 1/2" PVC pipe. The water flow is then split into the turbine path and the non-turbine path using a 1/2" PVC Tee Fitting. The turbine path will have a 1/2" PVC Elbow Fitting before connecting a 1/2" ball valve, which will be implemented in 1 of 2 ways(This will be explained in the valve control subsection). The output of the ball valve will be connected to the input of the 1/2" 12V DC Water Turbine Generator. This will be connected another 1/2" PVC Elbow Fitting before connecting to a 1/2" Inline Check Valve for Backflow Prevention. The check valve finally connects to another 1/2" PVC Tee Fitting, which is where the turbine path and non-turbine path are reconnected. The non-turbine path simply connects the first 1/2" PVC Tee Fitting to a 1/2" ball valve, which connects to the second 1/2" PVC Tee Fitting. All frame connections apart from the Water-Inlet Hose to the 3/4" FH Thread x 1/2" Slip PVC Fitting will be made through the use of 1/2" PVC pipes. In order to make these connections water-tight, we will be using a combination of Purple Primer and PVC Cement, which ensures that there will be no leakage. The circuitry will be held in a 4"x4"x4" 3d printed box that will clamp onto the surrounding PVC pipes. Lastly, the dimensions for the rectangular portion of the PVC frame will by 1' x 5" on the outside and 11" x 4" on the inside.

2.2 Power system with regulated turbine

Block Diagram

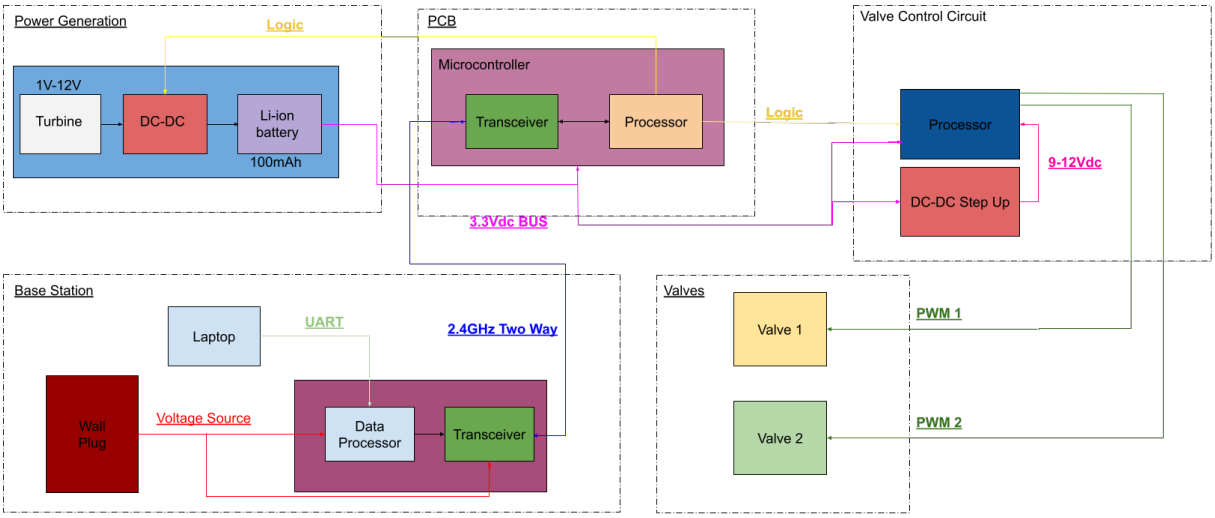

Figure 4: High Level Block Diagram

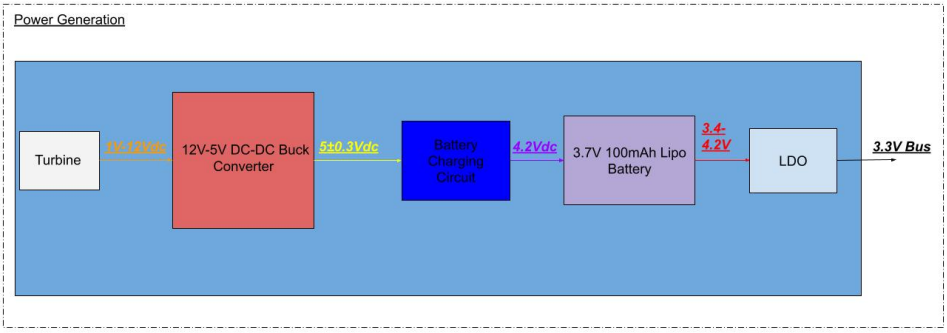

Figure 5: Zoom in Block Diagram of Power Generation Subsystem

Power Generation: This subsystem includes the water turbine generator that generates power from the water flow. The power generated from the turbine is not perfectly DC as the voltage is dependent on the flow rate, so the power must pass through a switching DC-DC converter before going to a battery charging circuit. We are still deciding which specific turbine we want to use as there are many tradeoffs. We could go with the 12V turbine that pumps out a solid 12V because of the linear regulator within the casing, or we could go with the 0-80V turbine that does not have a linear regulator inside of it. This linear regulator may cause output power problems. The linear voltage regulator keeps the voltage from surpassing 12V, and in doing so, it wastes energy in the form of heat dissipation. Let's optimistically assume that the raw, unregulated yet rectified DC output of this turbine can go up to 80V at the claimed 10W/80V = 125mA at the highest rated water pressure. The linear regulator drops the 80V down to 12V by dissipating a ton of power which causes the efficiency to be

less than 20%. For lower pressures, the amount of energy dissipated by the linear regulator would proportionately decrease, meaning the efficiency would be higher. But it still would be poor.

2.3 Power System with unregulated turbine

If we use the turbine that does not have a voltage regulator, then we could get much higher efficiencies at the cost of design complexity. We would have to design or find a proper step down DC-DC converter to output a voltage of around 4.5±0.3V to go to the battery charging circuit. This DC-DC converter would have to have good load regulation and be highly efficient (>85%), therefore we should not use a voltage divider. Ideally we would use a DC-DC Buck regulating converter. Further testing is required to figure out which solution is truly the best. A good DC-DC Buck regulating converter IC to look into would be the LMR38020FSQDDARQ1 by Texas Instruments. This device takes an input voltage range of 4.2-80V and outputs a solid 2A continuous. We would not have enough power to support this device. After further research, I have concluded that the 0-80V unregulated turbine is not a viable solution as there are no inexpensive 0-80V to 5±0.3V DC-DC regulating converters on the market that match our power constraints. We would have to design our own converter which is not feasible in the time frame given.

Because we are sticking with the 12V turbine, we would need a DC-DC buck converter to step down the unregulated 12V from the turbine to 5Vdc for the battery charging circuit. We could use this chip from TI: LM2594M-12/NOPB. A simple circuit for such conversion can be seen in Figure 5.

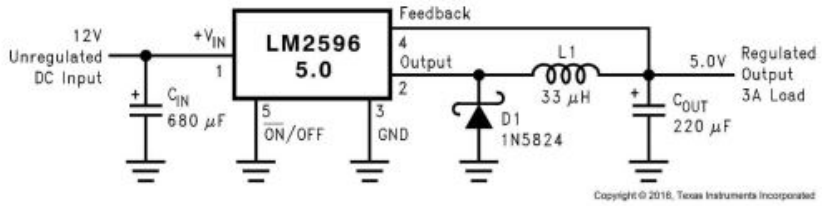

Figure 6: Unregulated 12V to 5Vdc converter

The 1N5824 Schottky Diode is not available online, so I found a replacement. The 1N5824 diode has a forward voltage of 370mV at 5A, so I found a diode that has a forward voltage of 370mV at 1A because we are only expecting less than 1A on the output of the converter. If we assume an efficiency of around 85% which lines up with the converter's datasheet, then:

$$P_{out} = 0.85 P_{in}$$

For a generous estimate, we can say that the turbine is generating 12V at around half of the current told to us by the product information. This would mean the input power should be around $12V \cdot 110mA = 1.32W$. Using the equation above, we see that the output average current of the converter should be around 224.mA with some current ripple. Therefore it is safe to assume that the current is less than 1A and we can select this Schottky Diode.

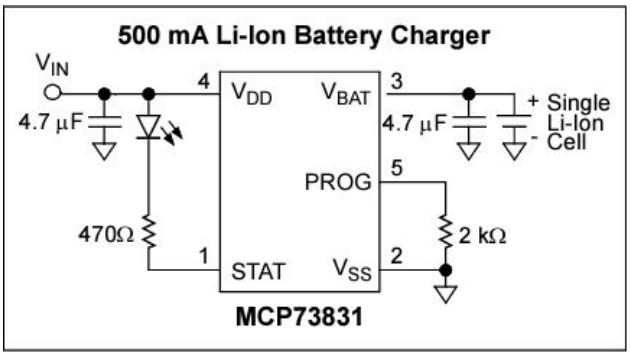

Figure 7: Simple battery charger from MCP7831 Data Sheet

The battery charging circuit is quite simple and can be done with a singular IC (MCP73831/2) and a few resistors and capacitors. This chip is great because it has high accuracy present voltage regulation (±0.75%), programmable charge current (15mA to 500mA), selectable preconditioning, and a variety of different voltage regulation options (4.2V, 4.35V, 4.4V, 4.5V) [2]. We will have to properly set the resistance values and voltage regulation options so that we supply the battery with the correct charging current curve to lengthen its lifespan and prevent damage. The battery charging circuit will have two logical outputs: "Battery Charged" and "Battery Low" which will be sent to the micro controller for further processing. We could also create a logical circuit to automatically shut off the charging circuit all together when the "Battery Charged" signal is high. The DC-DC converter could also simply get a signal from the microcontroller which allows us to control when we charge the battery. The charged battery will power the microcontroller and the valve control circuit. The battery charging circuit should supply a constant voltage of 4.2Vdc to the Lipo battery so that it can charge.

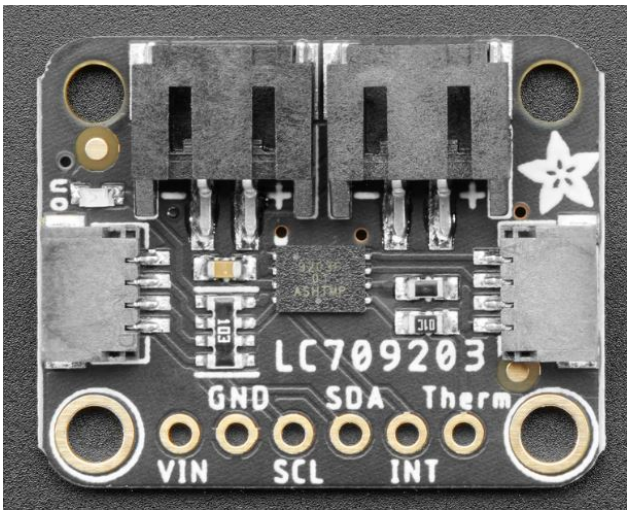

Figure 8: Battery Charge Monitor (Size of a Quarter)

We could also add a battery monitor to our charging circuit in order to more accurately

measure the battery percentage and voltage rather than just a "Battery Low" or "Battery Charged" signal. This would, however, come at the cost of increased power consumption and increased complexity. This component requires VIN from the microcontroller, GND, SCL and SDI from the microcontroller, and a STEMMA QT pin for dev board testing. Exact signals can be found from this website.

We would have to setup the communication between the battery charge monitor and the microcontroller ourselves which could be a challenge. This component is relatively inexpensive, but we do not know what the power consumption looks like. We should look into this option as it allows for more precise control over our whole system. Knowing the battery charge at any given time allows use to make decisions on whether or not to open or close valves.

We need an LDO from the battery to the 3.3Vdc bus to maintain a stable bus voltage. Because we are doing a small voltage conversion and we need a stable voltage to the microcontroller, we will need an LDO. We want to use BU33SA5WGWZ-E2 chip to be able to convert input voltages within the range of 3.3V-5.5V to an output of 3.3V.

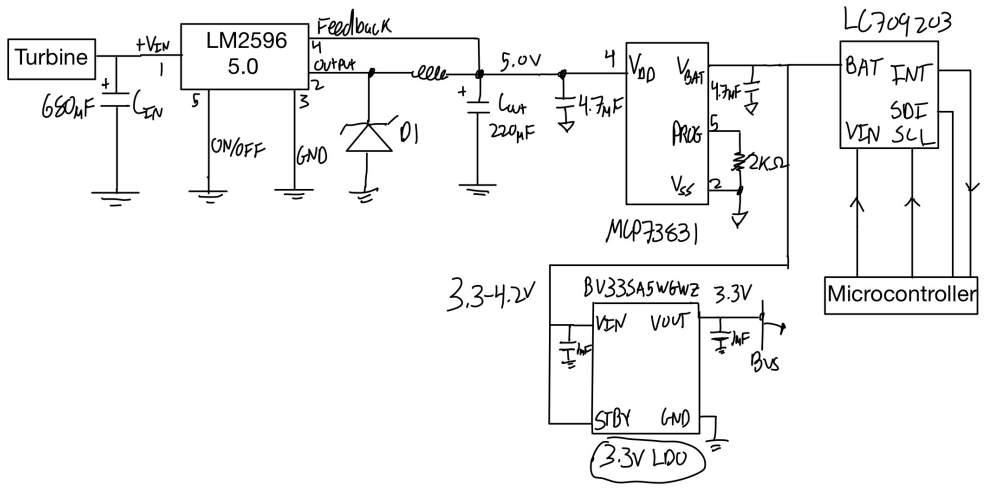

Figure 9: A first draft of the circuit outlined in Figure 4

Table 1: Power Subsystem Requirements

Requirements	**Verification**
All power conversion must be more than 85% efficient and design puts efficiency above all else.	1. Measure the input and output power at either side of every converter using a DMM and calculate efficiencies 2. Read all input and output waveforms of every converter to make sure smooth waveforms are maintained using an oscilloscope
If the battery is fully charged (4.2V ±0.1V, then stop charging	1. Place a charged battery (4.2V ±0.1V) into device right before a watering cycle 2. Monitor voltage and current to battery charging circuit with a DMM over a period of 2 minutes and make sure that battery does not get charged (voltage does not increase by more than 0.3V) 3. Measure to see if the "Battery Charged" signal is high when placing fully charged battery into device with a DMM
If the battery is going to die (3.4V ±0.1V), then close both valves	1. Place a close to dead battery into the device (3.4V ±0.1V) and observe that both valves close from an open state 2. Monitor "Low Battery" signal from battery charging circuit using a DMM and observe signal going high when dead battery is placed into device

Continued on next page

Table 1 – Continued from previous page

Requirements	**Verification**
Full battery can power all background processes for at least 5 days	1. Place a full battery (around 4.2V) into the system with valves closed and no water source 2. Observe how long the system can be powered until the "Low Battery" signal comes on 3. Record the voltage of the battery every hour and tabulate into a graph
Turbine valve shuts off when battery is fully charged.	1. Set both valves to "open" 2. Put a fully charged battery into the device 3. Observe the PWM waveform going to valve using an oscilloscope and record 4. Observe the turbine valve physically shutting

Continued on next page

Table 1 – Continued from previous page

Requirements	**Verification**
The power supply to the battery charger circuit must provide a voltage in the range of 4.7-5.5V for a current load up to 300mA when the turbine is generating power.	1. While the turbine valve is open and power is being generated from the flow of water, measure the current and voltage at the input to the battery charger circuit using a DMM to verify that the voltage is between 4.7V-5.5V through the whole watering cycle (20 minutes) 2. Take notes on the voltage and current every 30 seconds and tabulate the data into a graph 3. Note whenever the voltage or current drops below this threshold and record for how long

2.4 Microcontroller & Wireless

The Microcontroller Subsystem includes all the core software and control components for our project. This includes all wireless communication with the base station, reading data from the power subsystem, and sending digital control signals to the valve subsystem.

Table 2: Microcontroller Subsystem Requirements

Requirements	**Verification**
The device must be able to pair with a base station prior to physical installation.	1. Initiate this test with a fully-charged on-board battery. 2. Use a cable to connect a laptop to the device (disconnected from the water supply). 3. The laptop will display confirmation that the connection between the devices is successful and will initiate the key exchange process. 4. Once the key exchange is completed successfully, the base station will communicate this to the user that the device can be unplugged. This is a pass/fail test.

Continued on next page

Table 2 – Continued from previous page

Requirements	**Verification**
The device must be able to maintain a wireless connection with the base station. Furthermore, the base station should be able to send/receive data based on user input with a maximum latency of 10 seconds.	1. Start the test with the device having a fully charged battery (4.2V), and with the device pairing process completed. 2. Power on the base station (wired power source). 3. Send a 'ping' packet to the device. 4. The device should briefly enter a active mode, and send a 'ack' packet to the base station in response. 5. If the base station does not receive an 'ack' packet, it should notify the user that the device is suffering from connectivity issues. 6. For performance benchmarking, we will run this 'ping' test 100 times, and count the number of successful transactions which occured. 7. Record the number of successful transactions which occured for the final report.

Continued on next page

Table 2 – Continued from previous page

Requirements	**Verification**
The device's wireless component must be able to remain in the idle state (no valve operations, no wireless data received/-transmitted) for at least seven days time. Note that it is ok for the device to be in a fully-discharged state at the end of this 1 week period.	1. Due to the time constraints of the demo time, we will be unable to wait three days to demo this component. 2. Instead, we will measure power consumption for a smaller interval of time, and extrapolate to estimate the battery life of the device. 3. We will initiate this test over a 10 minute time window, and computing the difference in battery percentage. 4. We can then compute a $\frac{\text{minutes}}{\text{Battery \%}}$, and get our idle-time battery estimate. 5. Record this battery estimate ($100 \times \frac{\text{minutes}}{\text{Battery \%}}$) in the final report.

Continued on next page

Table 2 – Continued from previous page

Requirements	**Verification**
The device must be able to transition from idle state to active state if it receives a packet from the base station. After entering the active state for some amount of time (depending on the task given by the base station), the device returns to the idle state.	1. Initiate this test by starting the device in the idle state. 2. Then, have the base station send a wake signal to the device. 3. After sending the wake signal, initiate the ping test. 4. Note that it is ok for these steps to be combined into a single step. 5. After passing the ping test, then wait for the device to enter the idle state (we will pre-program this to happen instantly). 6. Then conduct the idle power consumption test. 7. If both tests pass, then the idle-active-idle transition requirement is met. This is a pass/fail test.

Continued on next page

Table 2 – Continued from previous page

Requirements	**Verification**
The device must be able to control valves by outputting digital signals, based on commands from the base station. Note that the valve subsystem circuitry will handle the conversion of these digital signals, and our assumption is that the valve subsystem circuitry will function as per the valve subsystem specifications.	1. Initiate this test with a nearly-fully charged battery (4.2V), and the device in the idle state. 2. Initiate a 5 minute watering cycle command from the base station. 3. The device should begin by opening the "recharge" valve. 4. Once the valve is open, the device should open the "primary watering" valve. 5. Once both valves are open, the base station should receive a confirmation that the watering cycle is active. 6. After five minutes (with no base station invervention), the device should shut off the primary watering valve within 10 seconds, then the recharge valve within 10 seconds. 7. It should then communicate that the watering cycle is completed to the base station. This is a pass/fail test.

Continued on next page

Table 2 – Continued from previous page

Requirements	**Verification**
While in the idle state, if the battery enters the "low battery" state (as defined in the Power Subsystem), the device will transmit a packet to the base station. The base station should then display this result to the user.	To conduct this test, we will need a lab setup. 1. We will spoof the battery circuitry by using a bench function generator. 2. We will initiate the device in the idle state. 3. We will then sweep from the maximum battery voltage (4.2V) down to the discharged battery voltage (3.4V) over a 10 minute time period. 4. Once the battery is below the 'low battery' threshold, the base station should receive a packet alerting that the device is low on battery. This is a pass/fail test.

All three of the design approaches are very similar from a software perspective. Before diving into the hardware details of each approach, we will discuss the overarching design of the Microcontroller Subsection.

We plan to minimize the power consumption of the wireless component of our system by minimizing the active time for the modem. However, we want to maintain a connection between the hub and the field devices at all times to ensure that we are able to continuously monitor the state of the field devices, and communicate with them without unreasonable latency. Bluetooth specifies the latency of the Bluetooth connection with a 'connection interval' - the lower the connection interval, the lower the power consumption of our device. According to Bluetooth Core Specification v5.3, the minimum connection interval is 4000ms. Note that this connection interval is not necessarily supported by any modem; it simply is an upper bound. However, all modems which are proposed support the 4 second connection interval. Also note that the master device decides the connection interval - this will not be a problem for us since we will ensure that our base station always operates with the maximum connection interval.

Once the connection is established, the device will sleep for nearly the duration of the connection interval. The device will then wake up from deep sleep, and listen for any incoming packets from the base station (time-based wake: requires timer on microcontroller). If a packet is received, the device enters active mode - this state enables valve operation, more data transmission, and begins data collection from on-board sensors (say, for the battery percentage). Note that the behavior in active mode is entirely dictated by the received packet. If no packet is received, then the device sends a 'ping' packet to the base station to keep the connection alive. We define the active time during the idle state as the time taken

for the microcontroller to deliver a packet to the base station, and get a reply confiming that transmission was successful. It is extremely important that the base station is aware if connection has been lost; there are living organisms dependent on this system, so if the valves are unable to open then the user should be notified as soon as possible.

To compute the battery life, we are using a 25mAh capacity dedicated to holding the idle state of the microcontroller - in our current design, this represents roughly 25% percent of the net battery capacity. Note that the battery voltage is matched to the microcontroller, so there will be no significant step-down or step-up conversion required (this, again, is for power efficiency reasons). We will estimate that the active time lies between 10ms and 50ms. All other parameters are lifted directly from the microcontrollers' respective datasheets.

$$B = \frac{C}{\frac{t_A}{4000} I_A + \frac{4000 - t_A}{4000} I_s} \tag{1}$$

Equation 1: Idle Battery Life Estimation: B is the battery life (h), C is the capacity of the battery (mAh), t_A is the active time (ms), I_A is the maximum active current (mA), I_s is the maximum current draw in sleep mode (mA). Assumption is battery voltage is equal to the microcontroller supply voltage.

The below summary table includes battery life estimates, for an active time ranging between 10ms and 50ms.

Approach	Active Current (mA)	Sleep Current (μA)	Battery Life (h)
Approach 1	4.9	1.5	398.525 - 1818.678
Approach 2	26	5	78.755 - 370.439
Approach 3	31	6	63.545 - 299.455

Table 3: Maximum Current Draw Comparison

2.4.1 Integrated Ultra Low Power Microcontroller/Modem

The device spends the majority of its time in the idle state. In this state, the device enters a low-power mode. The microcontroller only wakes up periodically to keep the Bluetooth connection active (the connection must remain active to have a good response latency), and to listen for any commands sent by the base station. The Bluetooth standard has a maximum connection interval of 4 seconds (note that not all modems support this large of a connection interval). For this reason, our preferred approach involves using an integrated modem/microcontroller IC which is optimized for low power. Another nice-to-have feature is the support for both BLE and Bluetooth Mesh, which would allow us to extend the functionality of our project to support mesh networking between devices if time permits (this is not part of our base functionality) with only software changes.

The specific part in question would be the nRF52833, an ARM microcontroller. There is dedicated ADC (for getting data from the power circuitry), a USB-2.0 interface (for communication and synchronizing Bluetooth keys with the base station), PWM support (for driving valve motors) along with integrated support for Bluetooth Low Energy, Bluetooth mesh, NFC, Thread and Zigbee, which gives us flexibility regarding protocol. The documentation by Nordic Semi is also good, which will help us during the development process.

During our development process, we would be using the nRF52833 DK, the corresponding development board for this SoC. To attach the part to the actual PCB, we will be using NordicSemi's reference circuitry (Figure 10). This particular reference circuit supports power from the battery of a "higher" voltage (no NFC). We prefer supplying power to VDDH because a higher battery voltage would be useful to improve the efficiency of our DC-DC conversions.

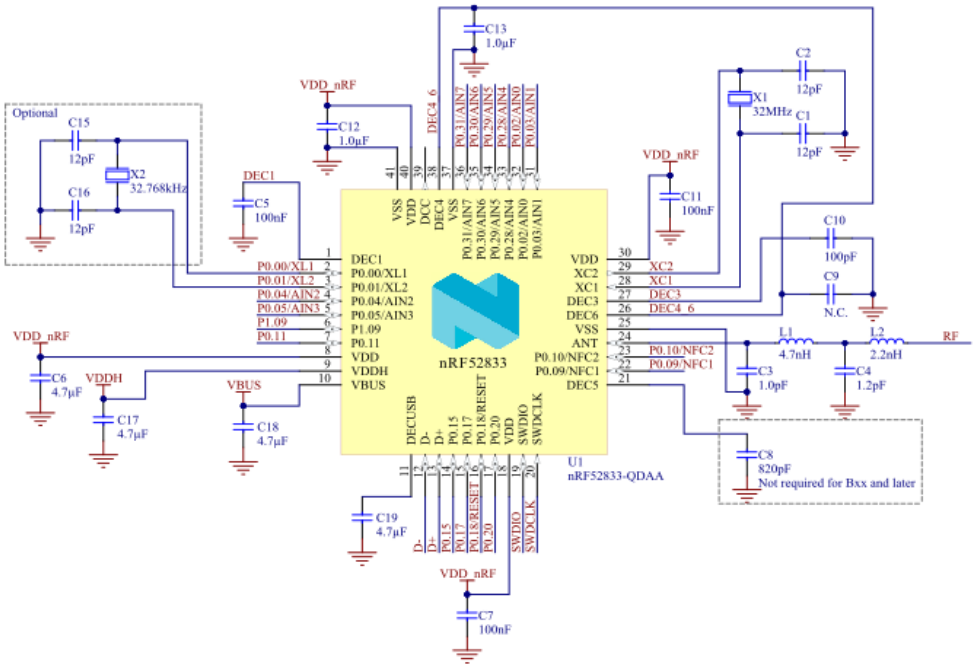

Figure 10: Nordic Semiconductor Reference Circuitry for nRF52833

The reference documentation also includes information on the specific capacitor values which should be used. The contents of Table 4 is directly pulled from Nordic's website.

The setup of the programmer of this board is slightly more involved than in other development boards – it's something we will need to design ourselves. Sparkfun previously made a breakout board for a similar nRF microcontroller (which has unfortunately been since discontinued). The guide recommends using a FTDI Basic Breakout, which provides a USB-to-serial programming interface. These correspond to the RXD, CTS, RTS and TXD pins, according to the datasheet.

We address the benefits and limitations/risks of this approach in Table 5.

2.4.2 Accessible Modem/Microcontroller SoC

To solve many of the issues with the nRF series microcontrollers, we also considered the ESP32 microcontroller created by ExpressIF. The primary advantage of ESP32 is the ease of use - there is ample documentation, and many of the SoCs support both Wi-Fi and Bluetooth. ExpressIF also provides software support for Bluetooth Mesh.

The specific part in question would be the ESP32-H2. Although there are other options, we chose the one with the lowest power consumption. The SoC also provides us many of

Table 4: nRF52833 BOM

Designator	Value	Description	Footprint
C1, C2, C15, C16	12 pF	Capacitor, NP0, ±2%	0201
C3	1.0 pF	Capacitor, NP0, ±5%	0201
C4	1.2 pF	Capacitor, NP0, ±5%	0201
C5, C7, C11	100 nF	Capacitor, X7S, ±10%	0201
C6, C19	4.7 µF	Capacitor, X7R, ±10%	0603
C9	N.C.	Not mounted	0201
C10	100 pF	Capacitor, NP0, ±5%	0201
C12, C13	1.0 µF	Capacitor, X7S, ±10%	0402
C17, C18	4.7 µF	Capacitor, X7S, ±10%	0603
L1	4.7 nH	High frequency chip inductor, ±5%	0201
L2	2.2 nH	High frequency chip inductor, ±5%	0201
U1	nRF52833-QDAA	Microcontroller/Modem	QFN-40
X1	32 MHz	Crystal SMD 1612	XTAL_1612
X2	32.768 kHz	Crystal SMD 2012	XTAL_2012

the features we need: onboard ADCs, PWM output and a Watchdog Timer. This particular SoC has a RISC-V core, although other ESP32 options have ARM options as well.

The power consumption is greater than the Nordic Semiconductor chip's maximum ratings. In active mode, the modem draws a maximum of 25mA for RX (at 3.3V), and at least 26mA for TX (with the lowest possible signal strength of -24. dBm). Between modem transmissions, we would enter a deep sleep mode (since we can wake up our system from a watchdog) with a draw of 5μA. This does present an algorithmic challenge, since the CPU and all onboard timing systems would be completely powered off, so we would need to resynchronize our systems - notably, we would be unable to use any pre-built libraries for ESP32, which takes away from many of the advantages of using the board in the first place. We have a power consumption estimate in Table 3 using 1.

We will start by prototyping on an ESP32-H2 DevKit. We have linked a specific part, but we were informed that ECE445 has a set of development boards as well. Eventually, when we move to a development board, we will need to include additional circuitry for power, flash, external clocks and RF components (antenna). Thankfully, ESP32 provides an excellent resource to help with this design component. Note that Schematic 11 was pulled directly from the hardware development guide. For this section, we have only included our own design choices (namely for parts which we must choose ourselves) - in other words, many of the parts required in the guide (for example, the oscillator must be 40MHz for the RF component to function correctly) have not been explicitly written out.

ESP32 supports either a 1.8V or 3.3V digital power supply. For this design, we chose the 3.3V option because the 1.8V variant has a peak current draw of 40mA, which may not be enough to support all of our RF needs. The external supply for 3.3V requires a 1μF filter capacitor. We will also be providing a separate supply for the on-board ADC - this is essential (according to the hardware development guide) to give the microcontroller a precise reference voltage. We will be using analog inputs for getting feedback from our turbine (for

Table 5: Pros/Cons of Approach 1

Pros	Cons
• Low power consumption: 4.9mA peak current for radio at 3.3 volts • Ample onboard resources: 512kB flash, 128kB RAM • Multiprotocol support (Zigbee, BLE, Bluetooth Mesh)	• Complex hardware configuration (ex. external oscillators) • Programming process is difficult - low level functions need manual implementation • Lack of (affordable) OTS programmer • None of the course staff have experience with nRF devices

ball-valve calibration, if time permits) and for estimating the battery levels. This circuit involves multiple capacitors and an inductor for decoupling and filtering purposes. We will also be making heavy use of the RTC for this application, since during sleep mode the ESP will depend exclusively on the RTC for wakeup between active modes. The RTC can be powered from an external LDO, and only requires a decoupling capacitor. The specific part numbers are reflected in our cost table below.

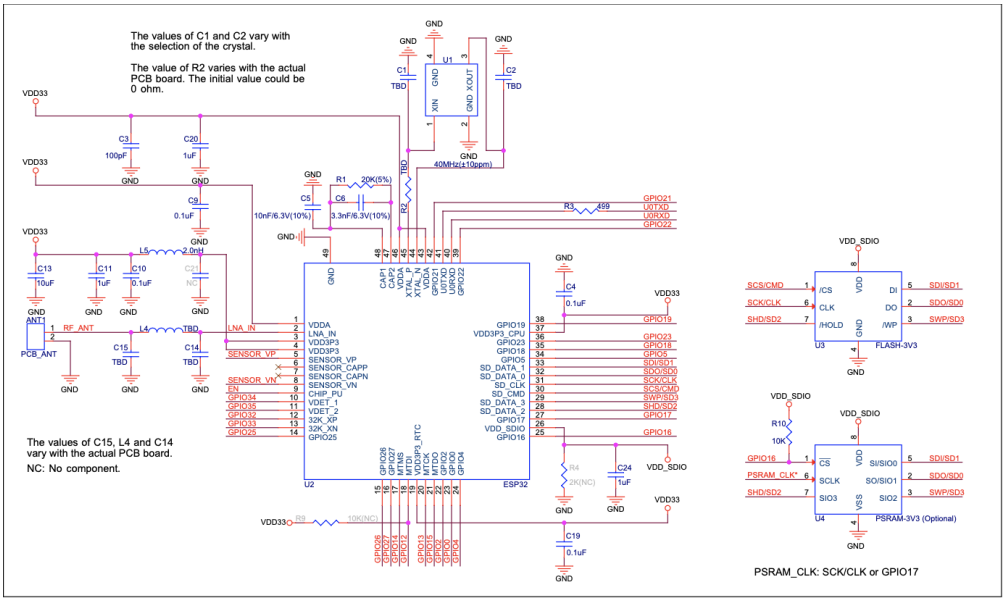

Figure 11: ExpressIF Reference Circuitry for ESP32

We have some flexibility for our flash storage choice. We do not see our application using a large amount of flash storage (for program storage or data storage) so we will only be

using the 2MB variant (W25Q16). We will not use any external RAM. We will still need to include a programmer chip - this is relatively easier, as we can just use any USB to Serial IC.

We address the overall benefits and limitations/risks of this approach in Table 4.

Table 6: Pros/Cons of Approach 2

Pros	**Cons**
• Great documentation and hardware development resources • Multiprotocol support (Zigbee, BLE, Bluetooth Mesh) • Easy to program via serial interface with CH340C	• Poor power consumption (see Table 3) • Large amount of supporting circuitry

2.4.3 [Preferred] Accessible Modem/Microcontroller Module

For the sake of brevity, we will not dive into excruciating details for this approach, since it is relatively similar to the second approach presented. The primary difference is instead of starting with the ESP32 SoC (which would require us to manually design the RF components, electric isolation, etc), we would use a pre-fabricated module. This would also let us avoid finding external storage and memory, and bypass the oscillators too.

The primary issue with this approach is that ExpressIF does not provide hard numbers for the power consumption of this module (when compared to the SOC on its own). The specifications list a marginally-higher receiver current (27mA), and we will use that to estimate the increase for the TX current of the device, as well as the sleep current.

We will be using a Sparkfun breakout for our initial testing. However, the schematic itself seems to be relatively simple, so in our second run of PCB orders, we hope that we can skip the breakout board and use the CH340C programmer chip directly. For cost analysis, we are using the more expensive (pre-assembled breakout) option to get a conservative estimate of our remaining budget.

One significant issue with this particular module is that it is newly released. As a result, Mouser does not have stock on hand (estimated October 23rd, which would significantly impact our timeline). As this is a new product, we also risk there being very limited documentation online (apart from the official docs, third party resources may not be available). Much of the advantage of using ESP32 comes from the online community, and the online community will not be able to help much with this Bluetooth module.

Why can we not just use an older product? This decision comes down to the numerical values. The minimum current draw for the H2 module (listed above) is 26mA, while the C6 series module consumes 90mA, which is around 3-4 times as much. However, we can still use the module for prototyping - in fact, this will be our working plan. This way, if the

desired module is out of stock, then we will still be able to ship our project at the end of the semester.

As this is our preferred approach, we have created a schematic for the C6 variant of our design. Note that we assume that we have a stable voltage source (since voltage regulation is handled by the power subsystem). Analog inputs from the power subsystem have not yet been marked explicitly in the figure.

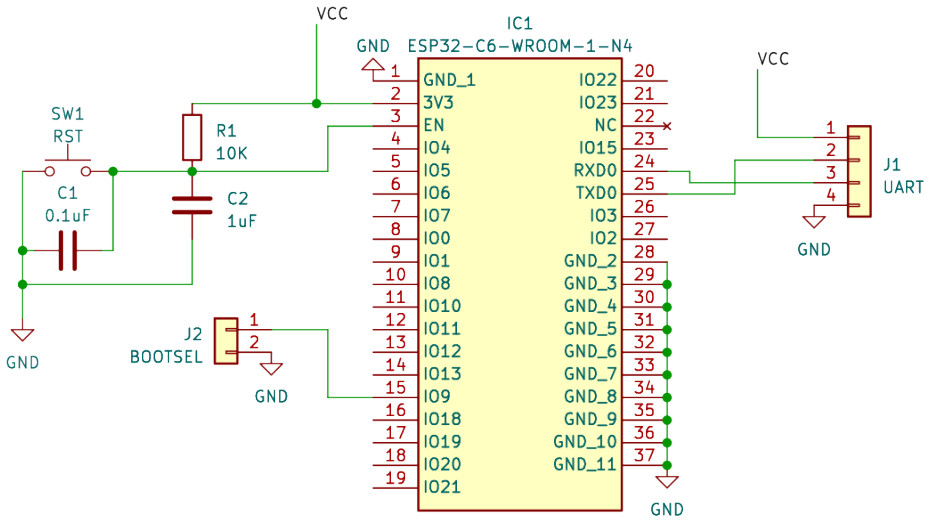

Figure 12: KiCAD schematic for ESP32 Module

Table 7: Pros/Cons of Approach 3

Pros	**Cons**
• Minimal additional circuitry required for development • Multiprotocol support (Zigbee, BLE, Bluetooth Mesh) • Easily programmable	• Worst power consumption (see Table 3) • Stock issues (arrives est. October 23rd) • New product (lack of outside resources)

2.5 Valve Subsystem

The valve subsystem takes 4 digital low current inputs of logic from the microcontroller, and an input voltage of 3.7V DC. The output of the subsystem is the independent opening and closing of two ball valves according to the control signals. In order to accomplish this, we use five components: One 12V DC Stepup Voltage Regulator (Pololu Corporation U3V16F12), two Half-Bridge Motor Drivers (Rohm Semiconductor BD6231F-E2), and two ball valves (U.S. Solid 888107092650). Since the Half-Bridge Motor Drivers require a supply voltage between 6V-32V DC and a VREF voltage between 3V-32V DC, and the two ball valves require a 12V/-12V voltage, we need a 12V DC Stepup Voltage Regulator. This voltage regulator work with an input voltage range of 1.3V- 16V and will output 12V with 90% efficiency. The voltage regulator has 3 pins, VIN, GND, and VOUT. VIN will be connected to the 3.7V DC input bus, GND is grounded, and VOUT will be the 12V output that we want from the voltage regulator. We then use the Half-Bridge Motor Drivers to provide 12V/-12V/0V depending on the logic signals. The pinout table for the motor drivers is below:

Table 1 BD6230F/BD6231F

Pin No.	Pin Name	Function
1	OUT1	Driver output
2	VCC	Power supply
3	VCC	Power supply
4	FIN	Control input (forward)
5	RIN	Control input (reverse)
6	VREF	Duty setting pin
7	OUT2	Driver output
8	GND	Ground

(Note) Use all VCC pin by the same voltage.

Figure 13: BD6231F-E2 Pinout Table

Pin No. 2, 3, and 6, which are VCC and VREF are all connected to the 12V DC output of the voltage regulator. The positive and negative sides of the ball valve are connected to Pin No, 1 and 7(OUT1 and OUT2) respectively. The two logic signals for each corresponding valve is connected to Pin No, 4 and 5 (FIN and RIN). Lastly, the GND pin is grounded. The circuit schematic for one ball valve is shown in Figure 14 below. The actual circuit will have 2 such circuits, one for each of the ball valves. Lastly, we had two potential options for the ball valves. First, we could make our own servo-controlled ball valves by attaching servos to PVC or metal ball valves. However, since we are unable to get an accurate estimate on how much torque will be needed to turn the ball valve an accurate amount using this method, we decided to use the second method. The second method was to use a motorized ball valve with a max power of 2 Watts. Since the opening/closing time is roughly 3.5 seconds, we estimated that every time one ball valve is either opened or closed, 7 Joules of energy will be expended. Since we are approximating that 70-75% of our net battery capacity, or around

75mAh/250 Joules, will be set aside for valve operations, we will have more than enough capacity to open and close both valves multiple times a day for multiple days.

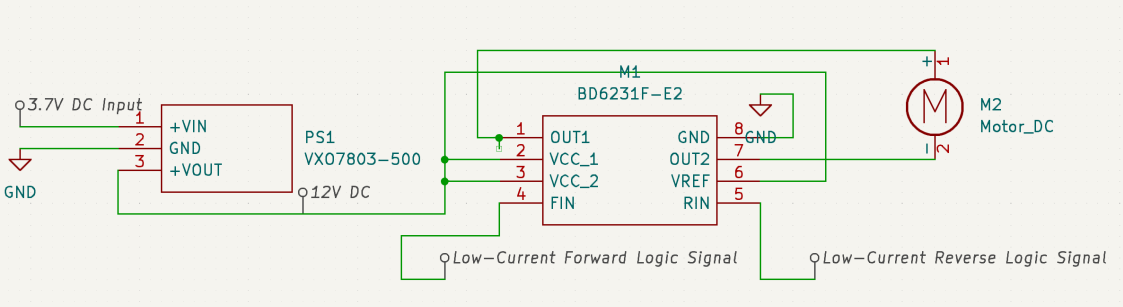

Figure 14: Valve Control Circuit Schematic

Table 8: Valve Subsystem Requirements

Requirements	**Verification**
The turbine valve must open, close, and maintain state depending on the signals transmitted to the valve subsystem.	1. First, test the turbine valve opening by setting up the system so the battery is low and starting a water cycle. We should see the turbine valve open. 2. Then, test the turbine valve closing in two situations. First test that when the battery is fully charged and the water cycle is still going, the turbine valve closes. 3. Then, test that when the battery is not fully charged but the water cycle ends, the turbine valve closes. 4. Lastly, test that if none of the above conditions are met, the turbine valve remains idle and neither opens nor closes.

Continued on next page

Table 8 – Continued from previous page

Requirements	**Verification**
The non-turbine valve must open, close, and stay idle depending on the signals transmitted to the valve subsystem.	1. First, test the non-turbine valve opening by starting a water cycle. We should see the non-turbine valve open. 2. Then, test the turbine valve closing in two situations. First test that when a water cycle ends, the non-turbine valve closes. Then, test that when the battery is low, the non-turbine valve closes. 3. Lastly, test that if none of the above conditions are met, the non-turbine valve remains idle and neither opens nor closes.
Make sure the battery consumption each time a valve opens or closes is less than 10% of the total battery capacity.	1. Test the energy consumption by opening or closing an individual valve. 2. Then check the net battery percentage before and after and to see whether less than 10% of the battery capacity was used.

3 Tolerance Analysis

The following components are copied from the above sections for the purpose of consolidation.

3.1 Power

Based on the documentation, the water turbine generator can provide a voltage of 12V at a current above 200mA. The current is heavily dependent on water flow rate and pressure. We would like to study this relationship to get a better estimate on the power generation capabilities of this turbine. If we assume the worst case scenario and say that the turbine can only produce 50mA at a voltage of 12Vdc, then the turbine can generate 0.6W of power. For a 20 minute watering cycle (which is typical for grass watering), this would mean that we could produce 720 Joules of energy ($1200 \cdot 0.6W = 360J$). The battery that we need to charge is 100mAhr at 3.7V which is 0.37Whrs ($0.1\text{Ahr} \cdot 3.7V = 0.37Whr$). This means that it would take 0.616 hours ($\frac{0.37Whr}{0.6W}$ or 37 minutes to fully charge the battery if it is initially dead. This equates to roughly 2 watering cycles.

3.2 Microcontroller

$$B = \frac{C}{\frac{t_A}{4000} I_A + \frac{4000 - t_A}{4000} I_s} \tag{2}$$

Idle Battery Life Estimation: B is the battery life (h), C is the capacity of the battery (mAh), t_A is the active time (ms), I_A is the maximum active current (mA), I_s is the maximum current draw in sleep mode (mA). Assumption is battery voltage is equal to the microcontroller supply voltage.

The below summary table includes battery life estimates, for an active time ranging between 10ms and 50ms.

Approach	Active Current (mA)	Sleep Current (μA)	Battery Life (h)
Approach 1	4.9	1.5	398.525 - 1818.678
Approach 2	26	5	78.755 - 370.439
Approach 3	31	6	63.545 - 299.455

Table 9: Maximum Current Draw Comparison: numerical values can be found in Section 2.4

3.3 Valves

The method we intend to use is a motorized ball valve with a max power of 2 Watts. Since the opening/closing time is roughly 3.5 seconds, we estimated that every time one ball

valve is either opened or closed, 7 Joules of energy will be expended.

$$\begin{aligned} Energy &= Power \cdot Time && (3)\\ &= 2W \cdot 3.5\ sec && (4)\\ &= 7J && (5) \end{aligned}$$

Since we are approximating that 70-75% of our net battery capacity, or around 75mAh/250 Joules, will be set aside for valve operations, we will have more than enough capacity to open and close both valves multiple times a day for multiple days.

$$\begin{aligned} Time &= \frac{Total\ Energy\ Capacity}{Energy\ Drained\ per\ Opening/Closing \cdot Valve\ Openings/Closings\ per\ Day}\\ &= \frac{250\ J}{7\ J \cdot 4\ Openings/Closings}\\ &\approx 9\ Days \end{aligned} \tag{6}$$

4 Cost and Schedule

4.1 Power System

Description	Part number	Cost
12V Water Turbine (w/regulator)	N/A	$12.94
100mAh 3.7V Lipo Battery	1570[3]	$10.99
Battery Charge Monitor	bq27010[4]	$2.00
12-5V Synchronous Buck Converter	LM2596[5]	$6.81
Battery Charging IC	MCP73831/2[6]	$0.76
10-4.7μF SMD Capacitors	T491A475K010AT	$3.09
10-470Ω Resistors	RC1206FR-10470RL	$0.36
10-2kΩ Resistors	CRCW04022K00FKED	$0.25
680μF SMD Capacitor	TPSE687K006R0060	$2.46
Schottky Diode Vfr = 370mV @ 1A	BAT60AE6327HTSA1	$0.44
33μH SMD Inductor	SRR1260A-330M	$1.30
220μF SMD Capacitor	TLJB227M006R0500	$0.99
IC REG LINEAR 3.3V 700MA SOT25	XC6210B332MR-G	$0.81
2 1μF Capacitors	C0402C105K9PAC7867	$0.20

4.2 Microcontroller and Supporting Circuitry

Note that we have only assembled this parts list for our 'preferred' approach. Our other approaches have additional information which could be used to estimate a price. If the cost of the programming hardware is not included, the prices for both approaches would be similar. These parts are for the H2 variant, not the prototype C6 BLE module.

Description	Part number	Cost
ESP32-H2 Microcontroller Module	ESP32-H2FH2	$1.39
SMD 10KΩ resistor	RP0402BRD0710KL	$0.41 ×4
SMD 1KΩ resistor	CPF0603B1K0E	$0.38 ×2
SMD 5KΩ resistor	CRT0805-BY-5001EAS	$0.44
SMD 0.1μF capacitor	C0603C104J1RACAUTO	$0.34 ×5
Pushbutton Switches	95C06D4GWRT	$0.62 ×2
Programmer IC Breakout	CH340C and USBC breakout	$9.95

4.3 Valve Subsystem and PVC Frame

Description	Part number	Cost
1-ft Water-Inlet Hose	WI12SSFM	$7.00
1/2 in. Slip x 3/4 in. FHT PVC Fitting	N/A	$1.70
1/2 in. PVC Schedule 40 S x S x S Tee	N/A	$0.64 ×2
1/2 in. PVC Schedule. 40 90-Degree S x S Elbow Fitting	N/A	$0.54 ×2
U.S. Solid Motorized Ball Valve- 1/2”	888107092650	$29.99 ×2
PVC Inline Check Valve for Backflow Prevention 1/2”	N/A	$6.99
1/2 in. x 24 in. PVC Sch. 40 Pipe	N/A	$1.56 ×2
PVC Primer and Regular Clear PVC Cement Combo Pack	N/A	$10.94
12V STEPUP VREG	U3V16F12	$4.95
Half-Bridge IC MOTOR DRIVER	BD6231F-E2	$2.86 ×2
Wire to Board Terminal Block	282837-2	$0.70 ×2

4.4 Overall Costs

We assume a salary of 40.00 per individual. We computed this value by rescaling the average annual salary for an EE graduate at UIUC (around $87,000) down to an hourly wage. We estimate that we will spend an average of 20 hours per week on this project. Note that the one week of break will not be billed hours, since we do not have work planned in our timeline for that week.

$$40\frac{\$}{\text{h}} \times 2.5 \times 3 \text{ partners} \times 20\frac{\text{h}}{\text{week}} \times 9\frac{\text{labor weeks}}{\text{semester}} = \$54,000$$

Description	Cost
Power Subsystem	$119.02
Microcontroller Subsystem	$17.12
Valve Subsystem	$104.16
Labor	$54,000
Grand Total	54,240.30

4.5 Timeline

We developed the below table for our timeline, understanding that it will likely changes due to our heavy reliance on the timeline for processing PCB orders for ECE445.

Week	**Milestones**
10/1/23	1. Functional draft of power electronics subsystem PCB – Grant 2. Functional draft of microcontroller (-RF) PCB – Anantajit 3. Hard numbers for performance characteristics of the turbine 4. Order components for physical design – Jay
10/8/23	1. Testing of PCBs – Anantajit & Grant 2. Refined Draft of microcontroller (+RF) PCB – Anantajit 3. Refined power PCB – Grant 4. Functional draft of valve controller PCB – Jay
10/15/23	1. Testing of Valve Controller – Jay 2. Finalized and Integrated PCB Design 3. Construct physical assembly
10/22/23	1. Software development (Field Device)
10/29/23	1. Wrap up software development (Field Device) 2. Base station (software prototype) 3. Final emergency PCB orders (just in case)

Continued on next page

Table 10 – Continued from previous page

Week	**Milestones**
11/5/23	1. Testing 2. Polishing of the base station interface
11/12/23	Testing
11/19/23	Mock Demo Prep
11/26/23	Break
12/3/23	Final Presentation

5 Discussion of Ethics and Safety

5.1 Security

Bluetooth key exchanges for pairing work over a wired connection with the base station. When the field device is connected to any base station, it will enter a "pairing mode" where it conducts the pairing process and any key exchanges over the wired connection. Secure pairing is especially important because our device is essential for plant life, and over/underwatering can cause death. Once the device is unplugged, it will automatically switch into wireless mode, where a packet is sent/received every 4 seconds to reaffirm a strong connection. Loss of connection is defined as a field device being inaccessible for more than one minute, at which point the user will be notified.

To prevent overwatering, our protocol allows the base station to only initiate watering cycles for set periods of time. For instance, if the wireless connection is lost in the middle of a watering cycle, the cycle would complete automatically after the time limit is reached. We cannot prevent underwatering (notifying the user is our best option).

The wired security component is limiting in the sense that if an intruder was able to gain physical access to the device, and they had their own base station, they would be able to become the 'master' of the field device. We are not covering this security case though, beyond alerting the user that the field device has changed to a different network (in other words, we are not taking preemptive measures to prevent this from happening).

5.2 Electrical and Water Safety

This project involves inherent safety risks due to the proximity of water and electricity. Accidental electrical shock is a potential hazard when working with the system after water has been running through it. Although the currents and voltages are low, safety precautions must be taken to avoid such incidents. Safety measures will include insulating and waterproofing components, clear warnings, and proper training for users. When charging the batteries from an external source, make sure not to overcharge for an extended period of time. The voltage of the battery should not surpass 4.2V for over 30 minutes. The battery should be checked before inserting into the device for any puffiness or deformities. The battery should be dry before placing them into the device to prevent any internal damage or rusting.

The enclosure for our microcontroller and power system must be waterproofed and tested before installation. The most problematic part of our project will be the turbine itself. Our primary goal is to fully waterproof the turbine so no water leaks from the interconnections within the system. This will ensure that no water can possibly get into the electronics. All people present for the demo must be at least 6 feet away from the device when the water is running. Once the water is turned off, we must wait at least 2 minutes before handling any electronics. We will be able to tell if there are any power supply problems based on the data sent to the base station.

References

[1] U.S. Geological Survey, "Droughts and Climate Change," *U.S. Geological Survey*, Nov. 3, 2015. [Online]. Available: https://www.usgs.gov/science/science-explorer/climate/droughts-and-climate-change

[2] Microchip, "MCP73831 Family Data Sheet," *Microchip*, [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/MCP73831-Family-Data-Sheet-DS20001984H.pdf

[3] Shenzhen PKCELL Battery Co., Ltd, "Li-Polymer Battery Technology Specification," Oct. 23, 2010.

[4] Adafruit, "Adafruit LC709203F LiPoly LiIon Fuel Gauge and Battery Monitor Pinouts," Sep. 22, 2022.

[5] Texas Instruments, "LM2594xx SIMPLE SWITCHER® Power Converter, 150-kHz, 0.5-A, Step-Down Voltage Regulator," May 2023.

[6] Microchip, "Miniature Single-Cell, Fully Integrated Li-Ion, Li-Polymer Charge Management Controllers," Feb. 28, 2020.

[7] Espressif Systems, "ESP32-H2FH2 Bluetooth Modules," 2023. [Online]. Available: https://www.mouser.com/datasheet/2/891/esp32_h2_datasheet_en-3240106.pdf

[8] Yageo, "Thin Film Resistors SMD 1/16 Wa 10KΩ," 2023. [Online]. Available: https://www.mouser.com/datasheet/2/447/PYu_RP_51_RoHS_L_0-3071169.pdf

[9] TE Connectivity-Holsworthy, "CPF0603B1K0E Resistor, 0603, 1kΩ, 1%," 2023. [Online]. Available: https://www.mouser.com/datasheet/2/418/8/ENG_DS_1773200_N2-1954016.pdf

[10] Bourns, "Resistor 5KΩ, 1%," 2023. [Online]. Available: https://www.mouser.com/datasheet/2/54/crt_as-1649073.pdf

[11] KEMET, "Capacitor 100nF, 5%," 2023. [Online]. Available: https://www.mouser.com/datasheet/2/447/KEM_C1023_X7R_AUTO_SMD-3316698.pdf

[12] Grayhill, "Non-Washable Tactile Switches," 2023. [Online]. Available: https://www.mouser.com/datasheet/2/626/grhls00979_1-2289208.pdf

[13] SparkFun, "SparkFun Serial Basic Breakout - CH340C and USB-C," 2023. [Online]. Available: https://cdn.sparkfun.com/assets/9/3/0/2/e/ch3402CDS.pdf

[14] Rohm, "DC Brush Motor Drivers," 2023. [Online]. Available: https://fscdn.rohm.com/en/products/databook/datasheet/ic/motor/dc/bd623x-e.pdf

[15] Pololu, "12V Step-Up Voltage Regulator," 2023. [Online]. Available: https://www.pololu.com/product-info-merged/4945