ECE 445

SENIOR DESIGN LABORATORY

FINAL REPORT

Wireless Irrigation System

Team #17

GRANT MCKECHNIE
(granttm2@illinois.edu)
JAYANTH MEKA
(jmeka2@illinois.edu)
ANANTAJIT SUBRAHMANYA
(as85@illinois.edu)

TA: Abhisheka Mathur Sekar
Professor: Olga Mironenko

November 26th 2023

Abstract

Traditional irrigation systems for residential properties severely limit the ease of installation of additional valves. Power and control wires are damage prone, with backyard pests, which in turn deal with increasing costs. Our solution is a Wireless Valve, powered by water flow and controlled over a Bluetooth connection. By the end of the semester, our goal was to control valve actuation over the wireless connection, recharge the battery using a water turbine inside of our Wireless Valve, and have the whole system last for two days on a single charge, all while giving the user constant updates with the current state of the system. Unfortunately, due to many inefficiencies in our design, many of which were caused by accidents during testing, we were unable to hit either of our efficiency requirements. However, our end device was able to demonstrate, as a proof of concept, that it is indeed possible to water plants using a completely wireless solution.

Contents

1 Introduction

1.1 Problem

Residential irrigation systems are essential for suburban greenery, from watering lawns to flower gardens. In traditional irrigation systems, a central hub controls the distribution of water to various watering zones around a property. Solenoid valves are then strategically placed to control water flow to each zone and connect to the central hub through a network of wired connections. By setting watering schedules on the central hub, users can control the volume and frequency of water required for each watering zone on a property.

However, the wired nature of the traditional model severely limits the ease of installation for additional valves. Power and control wires are damage-prone, with backyard pests, such as gophers, rats, and occasionally stray cats, biting and scratching the frail connection wires. In cases where the installation team makes a creditable effort to protect the wires by shielding the wires using a steel conduit, this adds significant costs to the system which discourages the homeowner from installing more outlet valves than are strictly necessary. This has led to the widespread adoption of zone-based watering, where a single valve is in charge of watering a large group of plants. Zone-based watering is becoming increasingly problematic: not every plant needs the same amount of water, and this reduction in control leads to some plants being either over or under-watered. Not only does this lead to water wastage, but it can be detrimental to the health of plant life as well.

1.2 Solution

Our proposed solution to this problem was a completely wireless irrigation system, which would improve the convenience and cost of valve installations. We eliminate the need for wired power by using a rechargeable battery on the valve, and we get rid of the wired communications by establishing a Bluetooth connection between each valve and the base station (acting as the central hub).

A major emphasis of this project was to make it as self-contained as possible with easy installation and maintenance. For this reason, our device fits the form factor of existing plumbing couplings (elbows, nipples, and other joints); this way, installation is as easy as connecting one end of our valve to a standard garden hose. Instead of requiring users to recharge the battery themselves, which would detract from the user experience, the device is equipped with a turbine that can generate power to charge the battery used to power all other parts of the valve. On the control side, our design uses a microcontroller responsible for all valve control, which could potentially be adapted to factor in sensor data such as soil moisture, sunlight, or temperature to ensure each plant receives the optimal amount of water required. The microcontroller also wirelessly communicates with the base station located nearby, allowing users to continuously monitor the state of the system.

Our solution, the Wireless Valve, offers a reliable power source that is not subject to the weather, has no outer wiring that can easily be damaged, and attaches to existing plumbing, making it easy to install and use.

1.3 High Level Design

The Wireless Valve has two water flow lines, which split off from the water source and rejoin at the output, which is what would connect to the plants. The Watering Line is an unimpeded connection from the source to the output, and its primary purpose is to provide a stable high-volume flow of water to the plants. The Recharge Line, on the other hand, has a fluid turbine which, as water flows through it, recharges the battery. By using internal ball valves to control which line the water flows through, the Wireless Valve is capable of providing high water pressure at its output while also being able to recharge itself when necessary.

On the electronics side, we divided our project into two highly interconnected subsystems, as seen in Figure 1.4: the power subsystem and control subsystem.

At the most general level, the power subsystem is responsible for providing consistent power to all parts of the project with a high efficiency of voltage conversion. If we split the subsystem down into its constituent parts, there is a turbine that produces the power, a buck converter that steps down the turbine voltage, a battery charging circuit, and a linear & low-dropout (LDO) regulator for the microcontroller 3.3V bus. The control subsystem deals with all control/monitoring elements in our system using the microcontroller, including driving the internal ball valves open and closed, monitoring the charge state of the battery, and communicating wirelessly with the base station. In our final design, the Power Subsystem ensures that the Control Subsystem is supplied with a reliable source of energy, and the Control Subsystem harnesses this energy to enable the watering capabilities of the valve itself.

1.4 High Level Requirements

To provide quantitative measures for the performance of our project, we devised the following three high-level requirements (quoted verbatim from the design document):

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 $\leq 1\%$ 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.

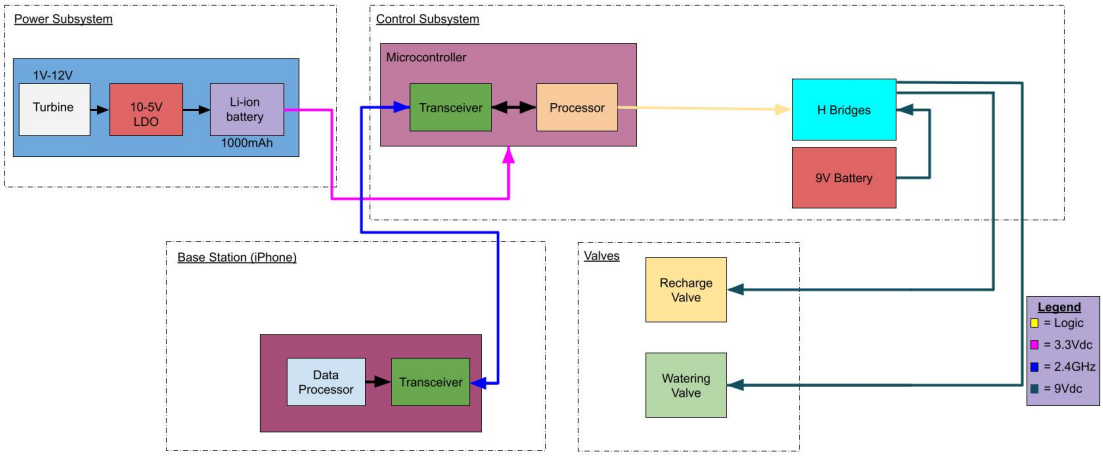

Figure 1: Final Block Diagram which provides a high-level overview of the internals of the three primary subsystems (Power and Control) of each Wireless Valve, along with the interactions between the subsystems and peripheral devices.

2 Design Procedure

2.1 Physical Design

2.2 Power Subsystem

2.2.1 Turbine

Based on the documentation, the water turbine generator can provide a voltage of 12V at a current above 200mA [1]. The current is heavily dependent on water flow rate and pressure. 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 ($1200 \cdot 0.6W = 720J$). The battery that we need to charge is 1000mAhr at 3.7V which is 3.7Whrs (1Ahr $\cdot\ 3.7V = 3.7Whr$). This means that it would take 6.16 hours ($\frac{3.7Whr}{0.6W}$ to fully charge the battery. This equates to roughly 19 watering cycles assuming no other operation in the system.

2.2.2 Buck Converter

The original buck converter from the design document was tested, but we figured out that it was the wrong chip for our application. The buck converter required much more power than we anticipated, and we accidentally burned it in the process of testing. The reason why the original converter burned may have also been due to the lack of proper capacitor and inductor selection or even shorting from poor soldering. Because there is high frequency switching with these buck converters, proper capacitor selection is essential. If a capacitor is selected with a high ESR at the switching frequency, the capacitor acts like a resistor rather than a capacitor which causes heat losses. Our final iteration utilized a much lower power buck converter which was more desirable for the objective of our project. The buck converter that was selected is the LMR50410-Q1 Simple Switcher. The input can range from 4-36V and it can support a load up to 1A [2]. For this application, the input was 12V and the output had to be 5V for the battery charging circuit (Fig:6).

In order to obtain the correct input-output relationship and ensure that we stay within the current limits of the device, we had to properly size and pick out the correct components. Most importantly, we needed to size and select the inductor and the resistors properly. According to the documentation,

$$R_{FBT} = \frac{V_{OUT} - V_{REF}}{V_{REF}} \cdot R_{FBB}$$

In this case, we wanted V_{OUT} to be 5V and V_{REF} was by default 1V. The documentation stated that R_{FBT} should be between $10k$ and $100k$. Selection of R_{FBT} was also based on the resistors that were available on Mouser. After a few iterations of searching Mouser, and evaluating the above equation, R_{FBT} was selected to be 44.2k. R_{FBB} therefore had to be 11k.

The next component that we had to select was the inductor. It is essential to pick the correct inductor to ensure proper operation of the converter. The design equation was given in the documentation [2] as

$$L_{min} = \frac{V_{in,max} - V_{OUT}}{I_{OUT} \cdot K_{ind}} \cdot \frac{V_{OUT}}{V_{in,max} \cdot f_{sw}}$$

Where $V_{in,max} = 12.3V$, $V_{OUT} = 5V$, $K_{ind} = 0.4$, $f_{sw} = 2.1MHz$, and $I_{OUT} = 0.67A$. Solving for L_{min}, we obtain $5.16\mu H$. The documentation recommended an inductor capable of up to 1.5A RMS and a saturation current of 2.5A for safety. The inductor selected for our project has an inductance of $5.6\mu H$, a maximum RMS current of 6.7A, and a saturation current of 6.9A. These values were obtained at a test frequency of 7.96MHz which is above the switching frequency. All inductor properties fit the specification.

The output capacitor is also an essential component to ensure minimal voltage ripple on the output. The selected output capacitor needed to have an equivalent series resistance less than 75m with a capacitance above $4.76\mu F$ and a voltage rating of above 5V. After searching, the output capacitor that we chose was a X7R ceramic capacitor with a capacitance pf $10\mu F$, a voltage rating of 10V, and an equivalent series resistance of only 10m.

The documentation essentially selected all other components for us, it was our job to simply find them on Mouser. When we soldered on all of the parts onto our final PCB, we started by testing each module separately before connecting all parts together. The first section that we tested was the buck converter and the charge management IC. The rest of the circuit was isolated during these tests to minimize any possible damages. Instead of the turbine, we utilized the bench DC supply for the 12V DC input. As soon as we turned on the bench, the current limit was hit and the voltage was only around 1.6V. This raised big red flags because this condition is indicative of a short somewhere in the circuit. Sure enough, the battery charge management IC was extremely hot and even started smoking. We immediately turned off the voltage supply and analyzed the problematic chip. It looked like two of the pins were shorted to each other due to some poor soldering which in turn caused the chip to short. We then removed the shorted chip and tried testing the input output relationship of just the buck converter. When we turned on the voltage supply again, the same condition occurred: 1.6V and 1A. This meant that when the charge management IC shorted, it also caused a short in the buck converter and damaged it past a point of no return. We did not have enough time to order a new PCB and new parts, so we improvised a solution. After doing research into the supply shop's inventory, we found a 10V-5V LDO [3] that we could use instead of the buck converter. The input was 12V which is outside of the input range for the LDO, but we read the documentation and found out that the chip has its own internal thermal regulation and power limiters. This chip has low power consumption, and the current draw is adequate for our application. The final circuit schematic with selected components can be seen here Fig: 5.

2.2.3 Battery Charging Circuit

The battery charging solution that we initially selected was 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 ($\pm 0.75\%$), programmable charge current (15mA to 500mA), selectable preconditioning, and a variety of different voltage regulation options (4.2V, 4.35V, 4.4V, 4.5V) [4].In all of our designs we needed the same battery charging chip, but during final testing, we accidentally fried the chip. We needed a replacement for the battery charging chip which took creative problem solving. Thankfully, the rechargeable lithium ion batteries that we bought came with USB 3.0 charging modules that took in 5V and outputted the proper current and voltage to charge the battery. We took apart the casing of this USB charger, did research into the pin layout of the circuit, and soldered jumper wires onto the input and output ports. After soldering the LDO [3] onto the backside of the board, we fed the output of the LDO into the input of the new battery charge circuit. The output of the battery charge circuit went to the battery.

2.2.4 Battery Voltage Measurement

Originally we wanted a chip that would measure the battery voltage and send that information to the microcontroller over SPI. This chip could also measure the health of the battery and relay a variety of different statuses to the microcontroller. We chose not to do this because of the added complexity. The chip had to be programmed in a very particular manner and had a strict set of minimum and maximum voltage and current values that were too restrictive for our application. Instead, we created a voltage divider with two large resistors in order to step down the battery voltage into the acceptable voltage range of the GPIO pins on the microcontroller. With this solution, we ensured that we would not go past the maximum GPIO voltage of 3.3V. In our case, we simply set R_{top1} equal to R_{bot1} equal to 1M This solution does use more power, but it is much less expensive, and easier to troubleshoot. This circuit theoretically draws between 8.82μW and 5.78μW using the formula below and the knowledge that a full battery is 4.2V and an empty battery is 3.4V

$$P_{loss} = \frac{V_{battery}^2}{R_{top1} + R_{bot1}}$$

The schematic is shown here: Fig: 3.

2.2.5 LDO

We stuck with the same LDO for all iterations. The LDO that we selected (XC6210B332MR-G) has a very low dropout of only 100mV @ 200mA and a low power consumption of 35μA [5]. This component is essential because it serves as the voltage regulator between the battery (range of 3.2-4.2V) and the microcontroller power ($\approx$ 3.3V). The voltage into the microcontroller cannot exceed 3.3V and should not get any lower than 3.0 volts for proper operation [6].

2.3 Control Subsystem

2.3.1 Microcontroller Unit

The choice of Microcontroller Unit (MCU) for this project was crucial. This is because to enable the Wireless Valve to have a reasonable battery life, battery drain due to wireless communication on the MCU needed to be at a minimum. We specifically optimized for the idle power consumption of the MCU, which we defined as the power used to maintain a Bluetooth Low Energy (BLE) connection with the base station. Additionally, to aid with development time, choosing an MCU which was easy to solder onto the board and easy to program would determine the time available to work on other parts of the project. To quantitatively evaluate the power consumption of various MCU, we devised a method which would give a rough estimate of the battery life of a MCU. During idle, the MCU operates in two states: the modem-on state and the RTC-only state. During the modem-on state, the MCU sends a single packet to the base station to ensure that the BLE connection is kept active. This is the relatively higher power consuming state, so minimizing the time spent in the modem-on state is important. The majority of the idle time is spent in the RTC-only state. In this state, the MCU powers down the primary processor and modem for some time, dropping the current draw of the MCU significantly. Equation 1 gives an estimate for the number of hours, H, that a particular MCU would last on a battery with capacity C.t_A is the time spent in the modem-on state (ms) for every 4 seconds. I_A and I_S represent the current (mA) drawn by the MCU in the modem-on and RTC-only states, respectively.

$$H = \frac{C}{\frac{t_A}{4000} I_A + \frac{4000 - t_A}{4000} I_S} \tag{1}$$

The three primary candidates we compared were the Nordic Semiconductor nRF52833, ExpressIf ESP32C6-MINI-1 [6], and the ExpressIf ESP32C6-MINI-U1 [6]. The MINI-1's main attraction was convenience - the software programming interface was well documented, the pins were large enough to solder and the wireless antenna was already present on the module for use, all at the cost of the power consumption. The nRF52833 on the other hand was extremely efficient, but had very small pins and required us to design a PCB antenna ourselves. The MINI-U1 was a middle ground; although it did not come with a built-in PCB antenna, it shared many of the interfaces with the MINI-1 module. A summary of the power consumption metrics are available in Table 2.3.1.

In the end, we decided to use the ESP32C6-MINI-1 module because the ease of development was a high priority, given the strict timelines we had to meet this semester. In addition to wireless connectivity using BLE 5.0, the ESP32C6-MINI-1 also provided us many of the features we need: a 12-bit onboard ADC and PWM output for speed control of the motors. This particular SoC has a RISC-V architecture.

The MCU we chose 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 would not be enough to support all of our RF needs. The external supply for 3.3V

Approach	Active Current (mA)	Sleep Current (μA)	Battery Life (h)
nRF52833	4.9	1.5	398.525 - 1818.678
ESP32C6-MINI-1	26	5	78.755 - 370.439
ESP32C6-MINI-U1	31	6	63.545 - 299.455

Table 1: Power consumption and battery life estimates for various MCUs considered, based on calculations using Eq. 1

requires a 1μF filter capacitor. We used these analog inputs for estimating the battery levels. This circuit involves multiple capacitors and an inductor for decoupling and filtering purposes. The full schematic for the peripheral circuitry of the MCU can be seen in Figure 2.3.4.

2.3.2 Valves and Gate Drivers

The method we intend to use is a motorized ball valve with a max power of 2 Watts [7]. 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. This is a very high estimate, but serves as a good upper

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

Since we are approximating that 70-75% of our net battery capacity, or around 750mAh/2500 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.

$$\begin{aligned} Time &= \frac{Total\ Energy\ Capacity}{Energy\ Drained\ per\ Opening/Closing \cdot Valve\ Openings/Closings\ per\ Day}\\ &= \frac{2500\ J}{7\ J \cdot 4\ Openings/Closings}\\ &\approx 90\ Days \end{aligned} \quad (5)$$

The valve subsystem takes 4 digital low current inputs of logic from the microcontroller, and an input voltage of 12V 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 four components: two Half-Bridge Motor Drivers (Rohm Semiconductor BD6231F-E2) and two ball valves [7]. We use the Half-Bridge Motor Drivers to provide 12V/-12V/0V depending on the logic signals. 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. The completed valve control schematic can be seen here Fig: 9.

2.3.3 12V Step Up

Because 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. While going through the design process, the location and the layout of the boost converter changed. We originally were thinking about a breakout board for the boost converter at the output of the LDO because we could ensure that the converter worked before incorporating it into the final circuit and the input would be a stable voltage. We went against this because we wanted freedom in design and having the boost converter at the output of the LDO could cause the LDO to draw too much current which would reduce its regulating capabilities. Instead, we designed our own components for a low power boost converter based on our desired inputs and outputs. The boost converter that we obtained accepts a wide range of input voltages (2.65V - 5.5V) and outputs a solid 12Vdc output. This output voltage is programmed through the use of an external resistor divider.

$$R_1 = (\frac{V_{OUT}}{V_{REF}} - 1) \cdot R_2 \tag{6}$$

Because boost converter outputs a stable 800 mV as VRef, and we want VOut to be 12V, we find that the combination of R1 = 1.5 MOhms and R2 = 107 kOhms satisfies Equation 6. R2 is chosen to be so low as the data sheet advised to use a value of R2 lower than 150 kOhms for better immunity against noise injection. For the capacitor selection, the data sheet advised to use any capacitor larger than one uF as the input capacitor.

$$C_{Out} = (\frac{I_{OUT} \cdot D_{MAX}}{f_{SW}} \cdot V_{RIPPLE}) \tag{7}$$

Using Equation 7, we are able to find the output capacitor needed to be 0.1 uF in order to maintain loop stability.

$$I_{L(DC)} = (\frac{V_{OUT} \cdot I_{OUT}}{V_{IN} \cdot Mu}) \tag{8}$$

The last component that needed to be chosen was our inductor. Using Equation 8, we found that we needed a 4.7uH inductor. This was also partly due to the data sheet stating that for a good design margin, an inductor with a 30 percent tolerance should be used [8]. A completed schematic of all selected components for the boost converter can be seen here Fig: 8.

The boost converter circuit worked perfectly when we first soldered it onto the final board. It was not until we tried to perform edge case testing where the boost converter got fried and went out with a phenomenal amount of smoke the day before our demo. In order to test the edge cases of the battery charge, we had to utilize the DC bench supply to simulate the battery voltage. First we set the bench supply to 3.7V and observed the measured voltage from the phone and saw that they agreed. We next slowly decrease the bench voltage in decrements of 0.1V until the valves closed. We observed that the valves closed when the voltage was around 30% of its maximum voltage. This behaviour was expected. We then foolishly moved to another bench, plugged in the DC power source, and pressed the "OUTPUT" button without knowing the preset voltage. We then tried probing the output of the boost converter, but the chip started smoking without stop. We immediately shut off the voltage source and checked what the set voltage was. The voltage of the bench was previously set to 9V which was almost 3x the voltage that we were testing. This over voltage destroyed the boost converter and then when we tried measuring the output voltage, we briefly shorted the pins which caused the chip to fry. This tragedy occurred very close to our demo and thus we did not have enough time to order parts and solder a new board. When the boost chip fried, it also completely destroyed the adjacent traces thus shorting parts of the board together. We had to manually cut the trace to isolate the destruction before moving onto figuring out a solution. Our solution to this problem was simply replacing the boost converter with a 9V battery. The valves require 9-24V DC in order to operate, and a full 9V battery is around 9.5V so it should work. We carefully soldered the 9V battery onto the board and tested the valve actuation. We were successfully able to open and close the valves, albeit at a slightly slower speed.

2.3.4 Physical Design

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 (Fig:4). 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.

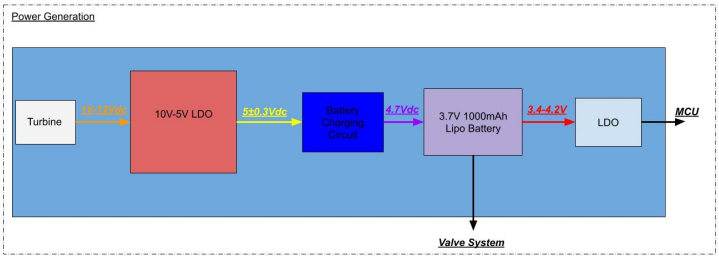

Figure 2: Final Power Subsystem Block Diagram which shows specific components within final power generation system build

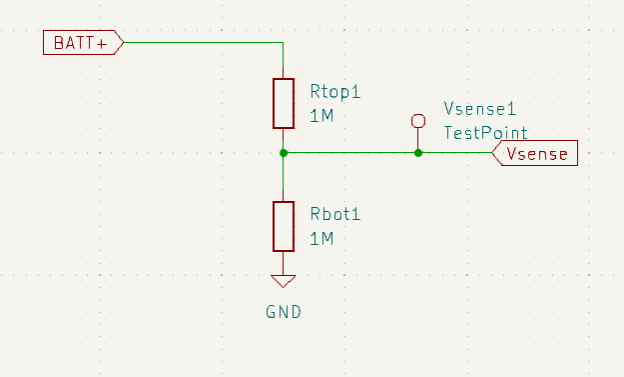

Figure 3: Voltage sensing schematic with selected resistances

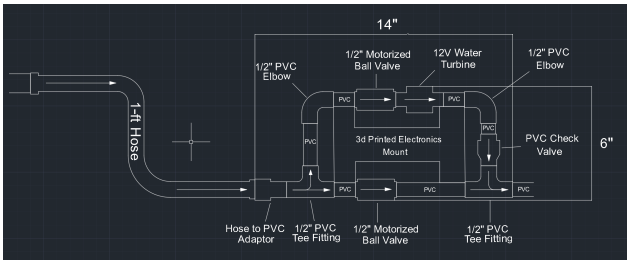

Figure 4: Physical Design Diagram

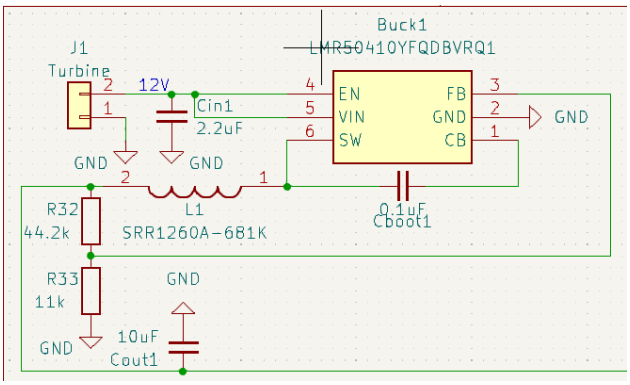

Figure 5: Schematic of buck converter circuit with selected values

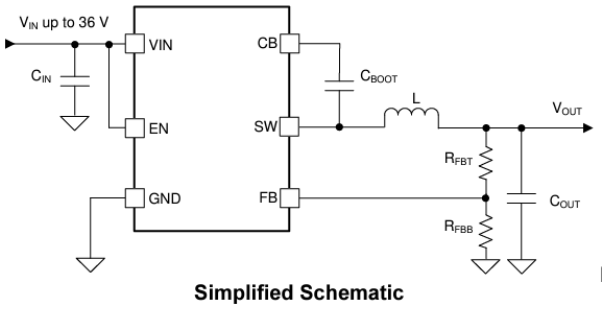

Figure 6: Buck Converter circuit with general values which will be specified through design process

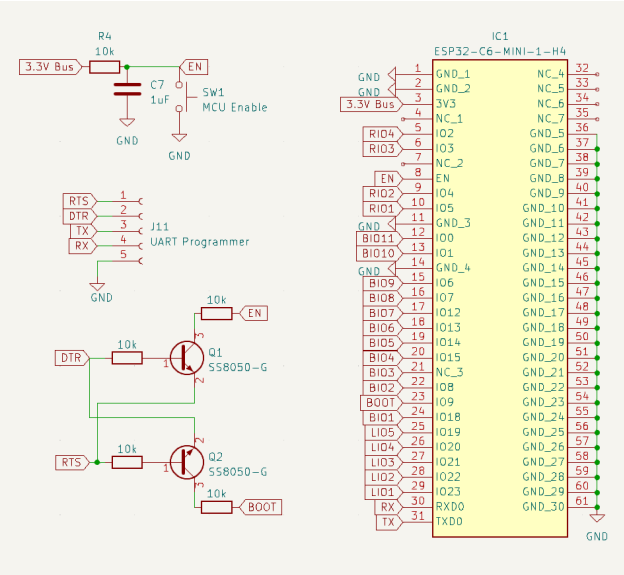

Figure 7: Microcontroller Peripheral Schematics: includes a delay circuit for driving enable pin of MCU during boot, along with circuits for controlling the boot mode during programming over the UART interface.

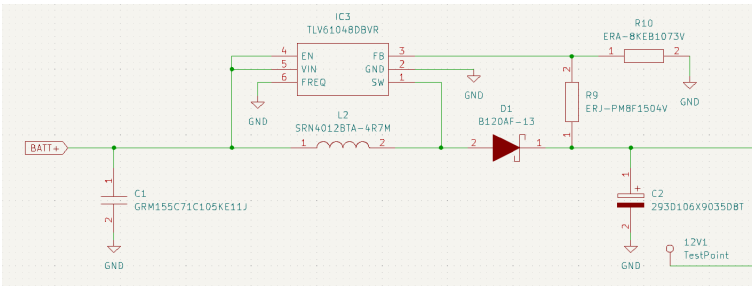

Figure 8: Boost converter circuit with input as the battery and the output being a solid 12V

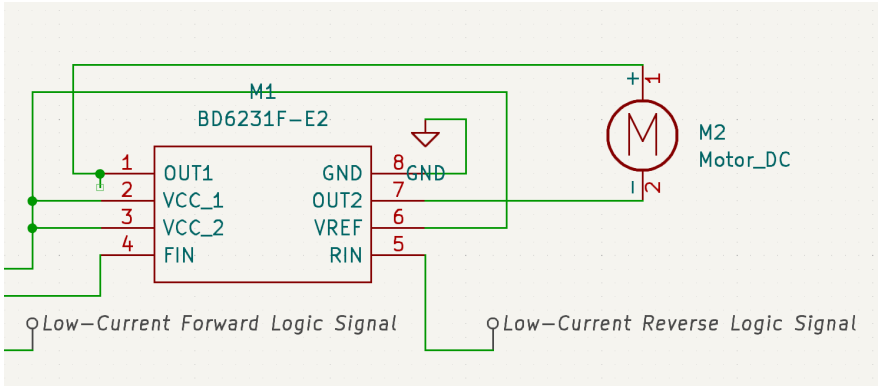

Figure 9: Valve Control Circuit Schematic with selected chips and input of 12Vdc from boost converter circuit (Fig:8)

3 Verification

Before performing any system level tests, we first performed component level tests to validate the operation of each chip. This was done by using the bench dc supply as a source, changing the input voltages, and measuring the voltages at different parts in the circuit. This is very standard, but an essential first step to validating overall system operation.

3.0.1 Power R&V

The purpose of the following tests was to understand the power demand of various parts of the project. The first test that was run was to characterize the behaviour of the valves.

1. Connect bench voltage source +25V and COM to a bread board and set the voltage to +12V
2. Connect the positive lead of the valve to the positive voltage lead of the bench source and the negative lead of the valve to the negative lead of the bench source
3. Press the ”OUTPUT” button and time how long it takes for the valve to stop making noise after it starts making noise
4. At the same time, record the current that the valve draws from the supply while it is actuating
5. Turn off the bench voltage source and flip the positive and negative leads
6. Repeat steps 3-4
7. Repeat steps 2-5 three times
8. Repeat steps 2-6 with the other valve
9. Tabulate data and average data

The average amount of power that a valve consumed can be found by multiplying the average current with the voltage. The average energy can be found by multiplying the average power with the average time Fig: 10.

The first requirement that we set for the power subsystem was very unrealistic, so our final design did not meet it. We wanted ”All power conversion must be more than 85% efficient and design puts efficiency above all else.” This was a lofty goal, but it did have a chance of being met if all went to plan. Because the buck converter and the battery charging chip got fried, we had to use a low efficiency LDO which automatically made us fail this requirement. The efficiency of the 10-5V LDO was around 50% at the operating condition based on the data sheet [[3]].

We were able to hit our second requirement of stopping the charging process if the battery

voltage exceeds 4.2± 0.1V. This was evident when we plugged in a 4.1V bench supply and observed the recharge valve closing.

The third requirement for the power subsystem was to close the valves if the battery voltage was under a threshold of 3.4$\pm 0.1V$. We verified this requirement by using the bench supply to slowly decrease the voltage from 3.7V on the battery terminal down to 3.5V and observing both valves closing.

In order to verify our 4th requirement: "Full battery can power all background processes for at least 5 days," we performed the below test:

1. Connect microcontroller to PCB and connect to it through the base station
2. Measure the initial voltage of a battery and make sure that it is within 3.7-4.2V
3. Plug in the battery and start a timer for 1 hour
4. After the hour passes, record the voltage of the battery
5. Calculate the change in voltage over the change in time and multiply by (4.2V-3.4V) to obtain the approximate time that a full battery can passively power the system

The number that is calculated from step 5 is not truly accurate due to the non linearities of a true battery discharge profile. Treat this calculated number as a generous estimate Fig: 11.

In our case, we had an initial battery voltage of 3.92V and a final voltage of 3.79V which represents a voltage decrease of 0.13$\frac{V}{hr}$. Using linear extrapolation, it would take around 6.15 hours for the battery to discharge from 4.2V to 3.4V. This is much less than our requirement of 5 days. We failed this requirement due to the power inefficiencies within the system that are isolated to the 10-5V LDO and the devboard. The 10-5V LDO has a much lower efficiency than the proposed buck converter ($\leq 50\%$ vs $\geq 75\%$) and the devboard consumes more than 4x the power that the selected integrated microcontroller consumes.

The sixth requirement ("Turbine valve shuts off when battery is fully charged") was very similar to the second requirement and was verfied with the same procedure.

Due to the power inefficiencies within our final system, our seventh requirement: "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" failed verification. In fact, the battery voltage decreased while the turbine was on. Even though this requirement failed, we tried to quantify the battery charging capabilities with the test below:

1. Disconnect/turn off the microcontroller
2. Measure initial battery voltage with a voltmeter
3. Put a 12V bench dc source on the terminals of the Turbine input on the PCB

4. ”OUTPUT” and start a timer at the same time for 3 minutes
5. During the charging time, measure the voltage of the battery charger to ensure that the voltage does not exceed 5V
6. After 3 minutes, turn off bench supply and measure the voltage of the battery with a voltmeter
7. Tabulate data and linearize the voltage increase over time to predict how long it will take to charge the battery from 3.4V to 4.2V

The initial battery voltage was around 3.8V and the final battery voltage was 3.9V after charging for 3 minutes. This represents an approximate linear charging time of 24 minutes from 3.4V (dead) to 4.2V (full). The data is shown here: Fig: 12.

The next test was to understand the system's passive power draw.

1. Set DC bench supply to 3.7V and turn off output
2. Disconnect battery and replace with bench voltage supply
3. Turn on bench supply
4. Record the voltage and current draw after 3 minutes

We observed a passive draw of 129mA at 3.7V which is a passive battery power draw of 0.4773W. This test validated that our final system draws much more power than expected due to the addition of the devboard and the 10-5V LDO that replaces our buck converter. The theoretical net battery power if we assume a 50% efficiency of the 10-5V LDO and a turbine power of 0.6W (12V $\cdot$ 50mA = 0.6W) is given by

$$0.5 \cdot 0.6W - 0.4773W = -0.177W \tag{9}$$

This is indicative of net power flowing out of the battery when the turbine is on and the system is passive. This means that the battery cannot be charged by the turbine alone.

3.0.2 Valve R&V

The most important requirment for the valves is that an opening and closing of both valves should not consume more than 10% of the battery. In order to verify this, we ran first analyzed the energy consumption of each valve from running the valve test (Fig: 10). Once we validated that the valves took around 2J per actuation, we then ran the below test:

1. Measure the initial battery voltage
2. Plug in battery
3. Send an ”open” signal from phone to MCU over BLE. Wait 10 seconds.
4. Send a ”close” signal from phone to MCU over BLE. Wait 10 seconds.

5. Unplug battery and record voltage
6. Validate that difference in before and after test battery voltage is less than 10% of total battery votlage range.

When we ran our test, we had an initial battery voltage of 3.87V and a final battery voltage of 3.85V. This voltage difference was clearly less than 10% of the total battery voltage range of 0.8V which means that we were able to verify this requirement successfully.

Valve Testing	Recharge Valve		Watering Valve	
Voltage	+12V	-12V	+12V	-12V
Avg time (s)	4	4	3.8	3.84
Avg Current (mA)	40	42	40	40
Avg Power (mW)	480	480	476	480
Avg Energy (J)	1.92	1.92	1.81	1.845

Figure 10: Valve test results

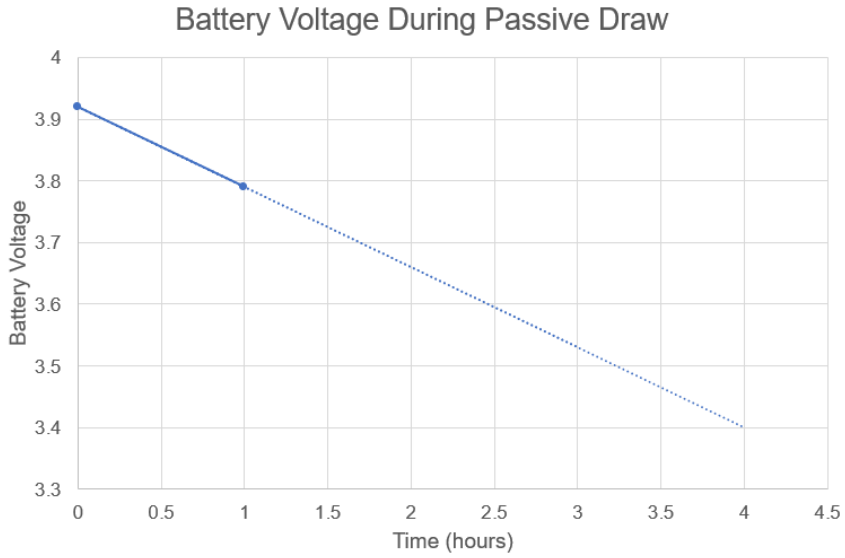

Figure 11: Battery discharge test results

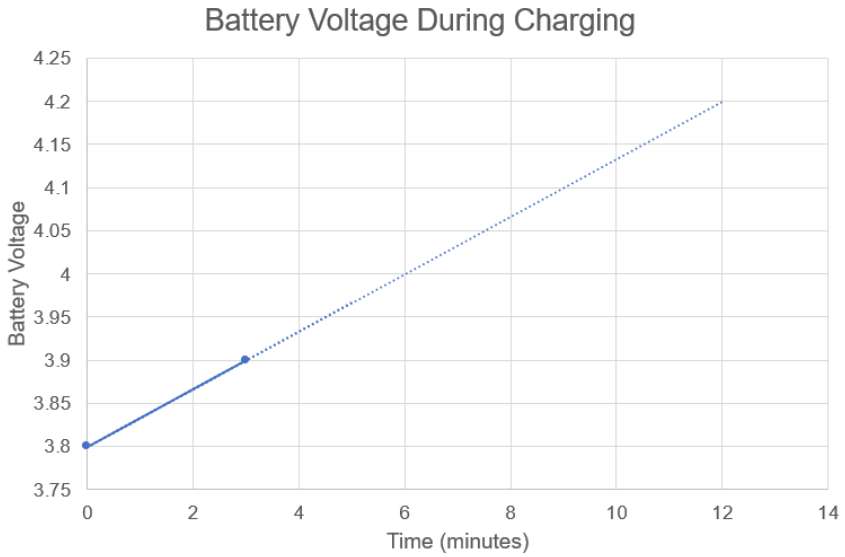

Figure 12: Valve test results

4 Costs

4.1 Power System

Description	Part number	Cost
12V Water Turbine (w/regulator)	N/A	$12.94
1000mAh 3.7V Lipo Battery	1570	$13.99
Battery Charge Monitor	bq27010[4]	$2.00
12-5V Synchronous Buck Converter	LMR50410YFQDBVRQ1[5]	$6.81
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

Description	Part number	Cost
ESP32-C6 Microcontroller Module	ESP32-C6-MINI-1	$2.57
SMD 10KΩ resistor	RP0402BRD0710KL	$0.41 $\times 4$
SMD 1KΩ resistor	CPF0603B1K0E	$0.38 $\times 2$
SMD 5KΩ resistor	CRT0805-BY-5001EAS	$0.44
SMD 0.1μF capacitor	C0603C104J1RACAUTO	$0.34 $\times 5$
Pushbutton Switches	95C06D4GWRT	$0.62 $\times 2$
Programmer IC Breakout	CH340C breakout	BORROWED
Development Board	ESP32-S3 Dev Module	BORROWED

4.3 Physical Design

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 $\times 2$
1/2 in. PVC Schedule. 40 90-Degree S x S Elbow Fitting	N/A	$0.54 $\times 2$
U.S. Solid Motorized Ball Valve- 1/2″	888107092650	$29.99 $\times 2$
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 $\times 2$
PVC Primer and Regular Clear PVC Cement Combo Pack	N/A	$10.94

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 spent an average of 15-20 hours per week on this project. For this estimate we are going to use an upper bound.

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

Description	Cost
Power Subsystem	$36.25
Control Subsystem	$17.12
Physical Design	$104.16
Labor	$54,000
Grand Total	54,157.53

4.5 Schedule

Week	Milestones
10/1/23	**Grant:** Spent this entire week working on designing the first version of the power PCB, and revising the design document. Benchmarked turbines which were purchased out of pocket. **Anantajit:** Researched various BLE modules and solutions. **Jay:** Developed a draft of the physical design with concrete part listing.
10/8/23	**Grant:** Researched and designed LDO circuitry, and finalized the design of the power PCB with help from the PCB review. Prepared a list of questions to ask Jason during office hours. **Anantajit:** Continued to work on the ESP32 module (finalized MCU by this point). Prepared a list of questions to ask Jason during office hours. **Jay:** Decided gate driver method, compiled materials needed for physical designs.
10/15/23	**Grant:** Attended Jason's office hours, and used it to submit the first round PCB order. Grant was in charge of combining the three PCB designs this round. **Anantajit:** Attended Jason's office hours, and used it to finalize MCU PCB design. Handed off PCB to Grant for combining. Finalized parts for all subsystems except for valve control components. Ordered the last of the first round of parts to go on the first round PCB. **Jay:** Put together physical frame.
10/22/23	**Grant:** Soldered the first round power PCB, discovered many issues. Researched new components and handed off to Anantajit to order. **Anantajit:** Cleaned up the PCB design. Anantajit was in charge of assembling this PCB order. **Jay:** Started and finished CAD Model for PCB encasing.
10/29/23	**Grant:** Revamped the entire power PCB for the final order. Incorporated Jay's circuit for controlling the valves for the first time. Also added the resistors for battery voltage sensing. **Anantajit:** Finished designing MCU final supporting circuit, with breakouts. Removed all nonfunctional circuitry with stable ones. **Jay:** Tested valves for functionality.

Continued on next page

Table 2 – Continued from previous page

Week	Milestones
11/5/23	**Grant:** Tested the components on the board and discovered new issues with the battery charging IC. Devised potential backup solutions which were eventually implemented. **Anantajit:** Finalized MCU PCB design after testing. Implemented basic software tests on development board. **Jay:** Tested H-bridges for functionality. Resized physical frame to fit PCB.
11/12/23	**Grant:** Additional testing during this week, including soldering all of power subsystem and half of the valve subsystem, prepared for handing off the board to Anantajit for software development. **Anantajit:** Soldered all of microcontroller and parts of valve subsystem pcb. Completed the ”heartbeat” test to verify correct board layout for MCU.
11/19/23	Fall Break
11/26/23	**Grant:** Finishing up the project. During testing, accidentally fried the boost circuit for the valve system after working on our project from 6:00pm - 4:30am. **Anantajit:** Finishing up the project. Discovered that MCU was fried in transit during Fall Break, so shifted setup to dev board. Worked with Grant on all nighter; after boost was fried, implemented workaround. **Jay:** Present for testing.
12/3/23	Final Presentation/Demo

5 Conclusions

At the time of the final demo, this project was able to actuate valves based on control signals from a phone through a wireless connection, read live battery charge and voltage from a phone charge a battery, close valves based on the state of the battery, and last around 6.15 hours off of a fully charged battery. We did face many hurdles on our journey to the final demoed project which were explored in the design section of this document.

5.1 Future improvements

In its current state, the project is not waterproof from external sources like rain and other irrigation systems. The first future improvement would be redesigning the PCB housing to be 100% waterproof from all sources. Next, we would like to properly integrate the microcontroller onto the PCB. This would decrease the passive power draw from the battery and increase the overall efficiency of power flow thoughout the whole system. In the current state, this project cannot charge a battery because the load draws more power than the charger can supply. This was evident when we left our system on for 10 minutes with a 12Vdc bench supply and saw that the battery only increased in voltage by less than 0.05V. The dev board requires at least 4x the power that the selected microcontroller requires which could be causing the battery to not charge. This goes back to basics, if the power demand from the load is higher than the supply, then the battery cannot charge. Next we would like to use the buck and boost converters that we selected in order to increase the power conversion efficiencies. The LDO that we selected is less than 50% efficient while the buck converter would be around 80% efficient. The boost converter is essential for the system to be fully integrated because the 9V battery will eventually die over time. Down the line, we would like to see communication between multiple valves in order to setup a full irrigation network. We would also like to see smart monitoring from either sensors or the internet to control watering cycles. This would prevent water waste if it was rainy (valves would stay closed).

5.2 Ethics

There are not many ethical considerations to our project. The main one is related to the first IEEE code of ethics: ”1. to hold paramount, the safety, health, and welfare of the public, to strive to comply with ethical design and sustainable development practices,..., and to disclose promptly factors that might endanger the public or the environment;” [[9]]. Our device must be sustainable and should not waste water unnecessarily. A few of our proposed tests will make sure that if the battery gets too low and water is flowing, the system will go into an emergency state that immediately shuts all valves, even if we are in the middle of a watering cycle. We must include a warning in our final product about the potential for our project to waste water in order to uphold the IEEE code of ethics because it is a real possibility. An error could randomly occur in the microcontroller to valve communication that can cause the valves to stay open indefinitely which will waste precious water resources. 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, we made sure not to overcharge for an extended period of time. The voltage of the battery should not surpass 4.2V. This will prevent any possibility of the battery puffing and exploding which is a possibility.

References

[1] ”Yosoo”. ““Yosoo DC Water Turbine Generator”,” ”Yosoo”. (Jan. 2018).

[2] T. Instruments, *”LMR50410-Q1 SIMPLE SWITCHER® 4-V to 36-V, 1-A Buck Converter in SOT-23-6 Package”*, Revised Edition, Texas Instruments, Nov. 2020.

[3] T. Instruments, *”TPS763xx-Q1 Low-Power, 150-mA, Low-Dropout Linear Regulators”*, Revised Edition, Texas Instruments, Mar. 2016.

[4] Microchip, *”MCP73831 Family Data Sheet”*, Revised Edition, Microchip, Dec. 2018.

[5] ”TOREX SEMICONDUCTOR LTD.”, *”XC6210 Series”*, 1st Edition, ”TOREX SEMICONDUCTOR LTD.” [Online]. Available: https://product.torexsemi.com/system/files/series/xc6210.pdf (visited on 09/26/2023).

[6] M. Electronics, *”ESP32-H2 Datasheet”*, 6th Edition, Mouser Electronics, Aug. 2023.

[7] U. Solid, *”Manual-U.S. Solid USS-MSV Motorized Ball Valves”*, U.S. Solid, 1200 E.55th Unit C, Cleveland, OH, 44103.

[8] T. Instruments, *”TLV61048 14-V Output Voltage Non-synchronous Boost Converter in SOT-23 package”*, Revised Edition, Texas Instruments, Jul. 2019.

[9] IEEE. ““IEEE Code of Ethics”.” (2016), [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html (visited on 02/08/2020).

Appendix A Requirements and Verifications

Table 3: 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

Continued on next page

Table 3 – Continued from previous page

Requirements	**Verification**
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
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 3 – 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

A.1 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 4: 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 4 – 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 4 – 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.
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 4 – 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 4 – 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.

Table 5: 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.
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.

Continued on next page

Table 5 – Continued from previous page

Requirements	**Verification**
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.