Smart Water Leak Shutoff Valve

Design Document

sdmay21-11
Cheng Huang
Grace Wilkins - Report Manager
Jihun Yoon - Meeting Scribe
Matthew Brandt - Backend Software Developer/ Meeting Planner
Curt Kissel - Software Developer
Wolfgang Morton - Hardware Test Engineer
Cody Juracek - Hardware Researcher Engineer

sdmay21-11@iastate.edu
sdmay21-11.sd.ece.iastate.edu

Revised: 11/15/2020 v_final

Executive Summary

Development Standards & Practices Used

- P1451-99 - Standard for Harmonization of Internet of Things (IoT) Devices and Systems
    - This standard defines a method for data sharing, interoperability, and security of messages over a network, where sensors, actuators and other devices can interoperate, regardless of underlying communication technology.
- Consumer electrons = IEEE/IEC 82079-1-2019 - IEEE/IEC International Standard for Preparation of information for use (instructions for use) of products - Part 1: Principles and general requirements.
    - This standard is about designing and integrating new technologies into a consumer's home - what safeties need to be considered, what legal regulations need to be followed, etc.
- IEEE 1667-2015 - IEEE Standard for Discovery, Authentication, and Authorization in Host Attachments of Storage Devices
    - This standard refers to discovery, authentication, and authorization protocols between storage devices and hosts over multiple transports.
- P802.16t - Standard for Air Interface for Broadband Wireless Access Systems Amendment - Fixed and Mobile Wireless Access in Narrowband Channels
    - This standard relates to the guidelines of when creating a product that produces a bandwidth signal of 5-100kHz.
- IEEE C62.36-2016 - IEEE Standard Test Methods for Surge Protectors and Protective Circuits Used in Information and Communications Technology (ICT) Circuits, and Smart Grid Data Circuits
    - This standard is about surge protectors for application on multiconductor balanced or unbalanced information and communications technology (ICT) circuits and smart grid data circuits are addressed in this standard.
- EEE 1142-2009 - IEEE Guide for the Selection, Testing, Application, and Installation of Cables having Radial-Moisture Barriers and/or Longitudinal Water Blocking
    - Detailed information relating to the design, testing, application and installation of various types of electrical cables in order to prevent the deleterious effect of moisture and chemical ingress and resultant failures in service is provided in this guide. This includes single and multi-conductor cables over a complete range of voltage ratings. Testing criteria and installation methods covered along with many technical references.

Summary of Requirements

- Shutoff Valve
- Android Application
- Control Shutoff valve with Android Application
- Wifi connectivity through Arduino
- Complete project not exceeding $250

Applicable Courses from Iowa State University Curriculum

*Note: Courses that overlap across different majors are only listed once

SE	CprE	EE
CS 309	CprE 288	EE 201
CS 228		EE 230
CS 227		EE 333
CS 319		EE 456
CS 363		
SE 329		

New Skills/Knowledge acquired that was not taught in courses

- Machine Learning
- Embedded Systems
- Establishing communication between Android app and arduino
- Getting variables from a user's system, such as their GPS location and timezone
- Some basic plumbing knowledge
- Background Processes in Android OS

Table of Contents

List of figures/tables/symbols/definitions

1 Figures

2 Tables

1 Introduction

1.1 Acknowledgment

We would like to thank Dr. Cheng Huang for being our advisor for this project. His proposal provides an applicable, hands-on opportunity to combine hardware and software components into an end product that can be utilized by many people today. We appreciate his insights and dedication to keep us on track to produce an efficient and effective product.

1.2 Problem and Project Statement

For many homeowners and property managers, water damage is a significant concern. It can be an expensive problem to fix, with one of the most frequent reasons for flood damage being leaking or burst pipes. Currently, available market options to monitor and control water flow are too expensive or lack additional functionality for most homeowners to consider purchasing. This project aims to develop a low-cost water shutoff valve with the ability to monitor water flow to prevent and protect against water damage for homeowners and property managers. The product should be able to: measure the flow of water through the waterline, connect to the internet for real-time water control and monitoring, and provide smart features such as text alerts and an automatic shutoff mode.

With this project, we hope to provide a means to prevent water damage while keeping the component cost of the product around $200. By being able to detect when abnormal water usage occurs, our product will be able to alert the homeowner and shut off the valve to prevent further damage. The shut-off valve would be an excellent preventative investment for homeowners while also being market competitive by being cheaper than other available options. We hope to deliver a complete system with internet-connected hardware that can monitor water flow, shut off the water valve, and be controlled via a user-friendly smartphone app that can also monitor the hardware and control the water shutoff valve.

1.3 Operational Environment

Our hardware components for the project will be installed by the main waterline for a building. These areas can be damp and prone to dust. The inside component will be exposed to water. The app would be expected to be used on most mobile devices in a place that has an internet connection.

1.4 Requirements

- Total hardware components should stay within a $200 goal
- Hardware should be able to be installed in a home or business setting
- Hardware should be able to monitor the flow of water through the waterline and send the data to be stored on a server
- Hardware should be able to shut off the flow of water through the waterline
- App should be able to communicate with the hardware to open and close the water valve
- App should allow users to view history of water flow
- App should allow users to monitor current water flow
- App should be responsive and easy to use
- App should allow users to list times when the valve should be automatically shut when water is detected

1.5 Intended Users and Uses

The intended users for this product are homeowners and other property owners. Water damage can be expensive to repair and can cause structural damage to buildings. This product will supply property owners with a cost-efficient way to prevent water damage and monitor water usage. Depending on the most effective solution, the product will be able to be used by people with no or limited plumbing knowledge.

Figure 1.5 - Use Case Diagram

1.6 Assumptions and Limitations

Assumptions:

- Users have access to a mobile device with internet connectivity
- Users have basic plumbing knowledge to install the valve by themselves
- Only one user can be logged into the account at once
- Users have internet connection for hardware to connect to
- Users have some basic knowledge of plumbing

Limitations:

1. Product should cost less than $250
2. Should be small enough to fit wherever the main water pipe is in the household
3. Operates at common household voltages

4. Depending on most effective solution, some knowledge of plumbing may be required

1.7 Expected End Product and Deliverables

Hardware System - internet connected water shutoff valve with waterflow connection. Multiple options available with recommendations based on the user's knowledge of plumbing. Delivered by May 2021.

Mobile App - User friendly mobile application that allows for water flow monitoring and manually shutting off water through the water valve. Delivered by May 2021.
Installation Guide - Manual for users to install the hardware system for the shutoff valve Delivered by May 2021

2 Project Plan

2.1 Task Decomposition

The following tasks will be done repeatedly until an acceptable solution is implemented:

I. Planning
   a. Assigning Roles
   b. Setting the Schedule
   c. Ordering Necessary Hardware
II. Prototyping
   a. Screen Sketches for Android Application
   b. Diagrams for Hardware/Software Integration
III. Development
   a. Android Application Programming
   b. Assembling Hardware
   c. Hardware/Software Integration
IV. Testing
   a. Android JUnit testing
   b. Android Mockito Testing
   c. Controlled Environment Hardware Testing

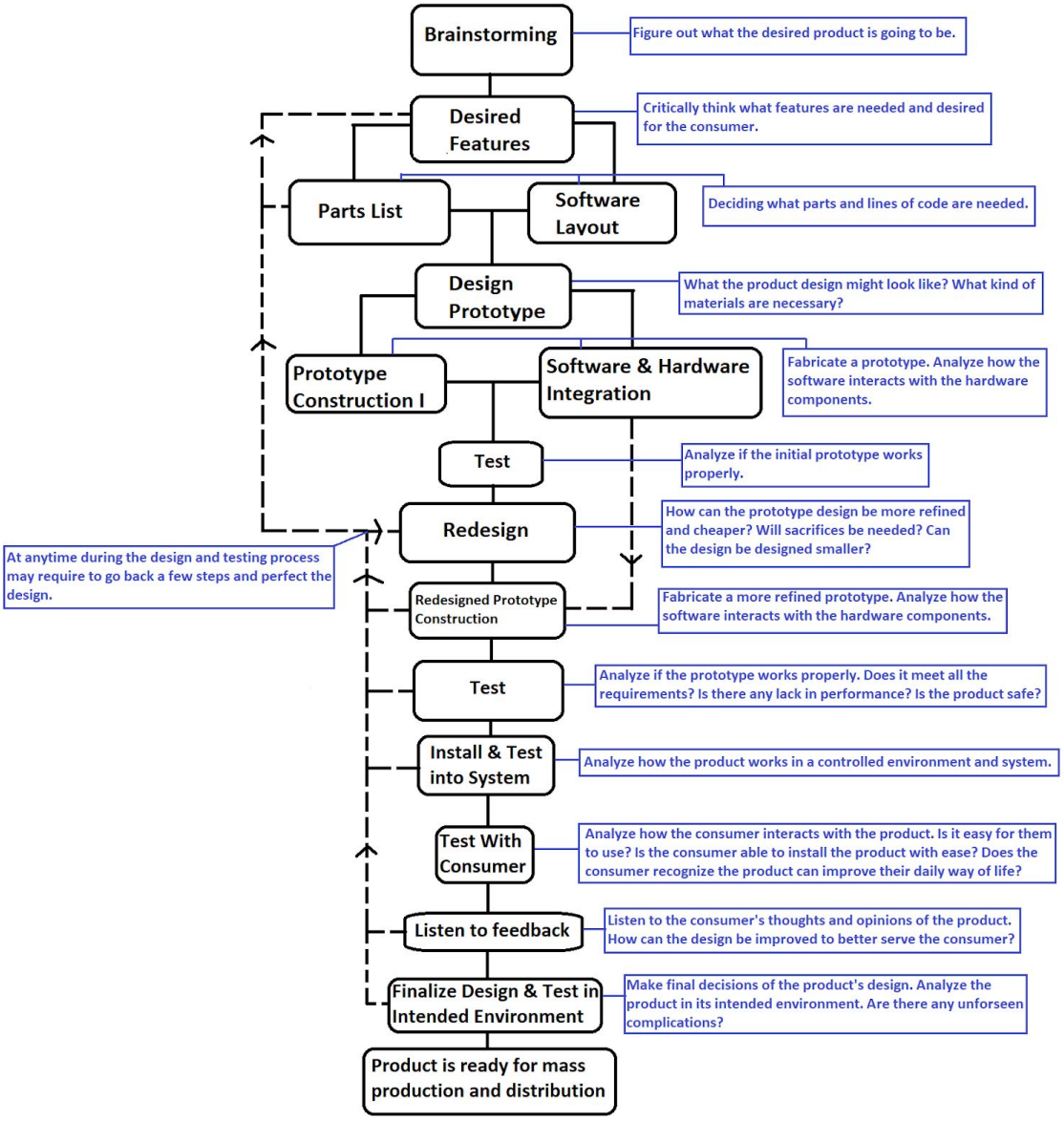


Figure 2.1 - Task Decomposition

V. User Feedback
   a. Beta Release
   b. Refactoring

2.2 Risks And Risk Management/Mitigation

Arduino Losing Internet Connection- Users will be encouraged to connect their hardware to the internet via an ethernet adapter.

App Crashing- While impossible to completely avoid app crashes, there will be thorough testing to ensure that the app functions as desired.

Software Compatibility- We will test our app across several different devices to ensure that it is compatible with all current Android devices.

Login Information Getting Compromised- We will work to encrypt a user's login credentials, as well as adding two-factor authentication.
Short Circuiting - To prevent short circuiting, the hardware will be put under multiple tests to ensure proper functionality.
Budget Cuts - There is not much that our group can do if funding is cut. In this case, we will be forced to find an even cheaper solution.
Deadline Changes - Several schedule changes will be made to allow our team to meet the deadline for this project.

2.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

- Machine learning algorithm to detect unwanted water flow will classify with 80% accuracy; the pattern recognition logic on water usage
- System sends data updates at 5 minute intervals with 90% accuracy.
- Application receives accurate data and updates charts and graphs.
- Application can communicate with hardware and hardware responds within a reasonable timeframe (500 ms).
- Working hardware prototype is developed that can halt the flow of water within a reasonable timeframe (500 ms).

2.4 Project Timeline/Schedule

- A realistic, well-planned schedule is an essential component of every well-planned project
- Most scheduling errors occur as the result of either not properly identifying all of the necessary activities (tasks and/or subtasks) or not properly estimating the amount of effort required to correctly complete the activity
- A detailed schedule is needed as a part of the plan:
    - Start with a Gantt chart showing the tasks (that you developed in 2.1) and associated subtasks versus the proposed project calendar. The Gantt chart shall be referenced and summarized in the text.
    - Annotate the Gantt chart with when each project deliverable will be delivered
- Project schedule/Gantt chart can be adapted to Agile or Waterfall development model.

How we plan for the project to be completed in two semesters:

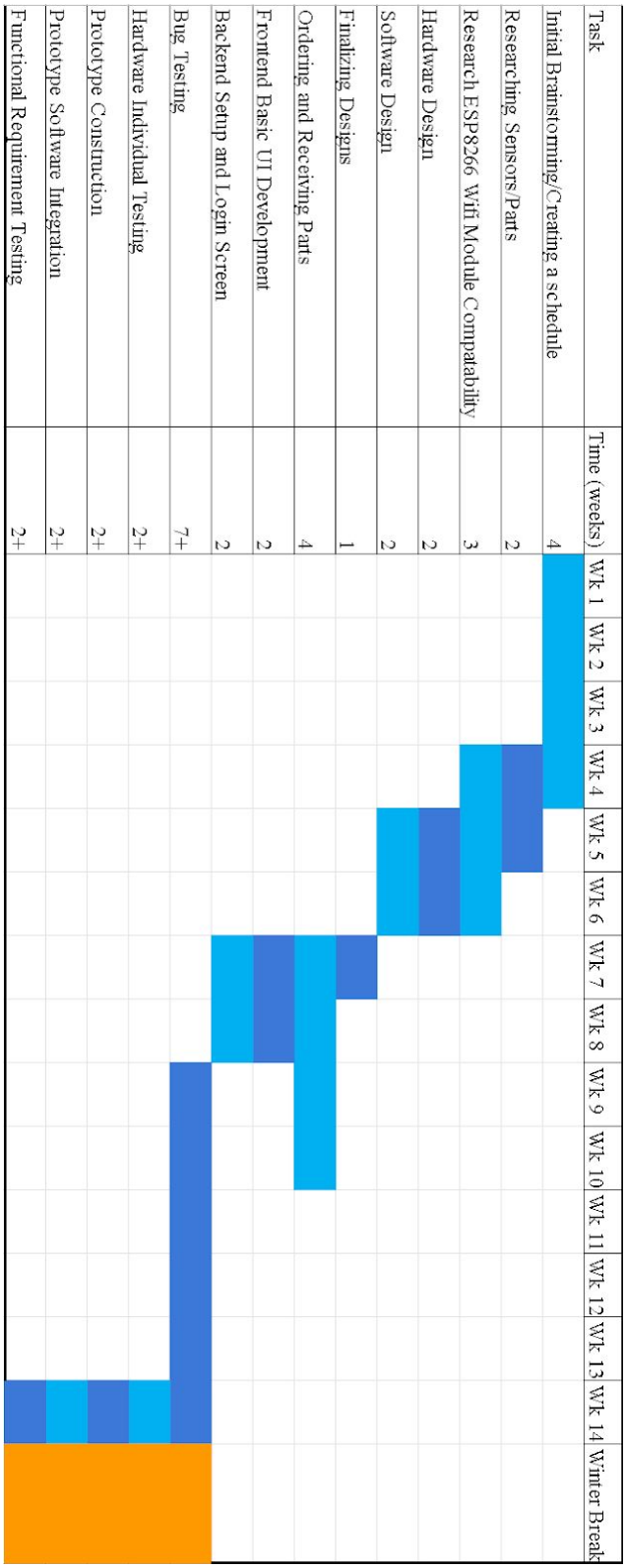


Figure 2.4 - Project Timeline (1st Semester)

2.5 Project Tracking Procedures

For this project, we will be using Git and Github to store our code and allow for multiple people to implement different features. It will also allow us to keep track of changes and to allow for multiple people to review code before finalizing updates.

We also plan on using Trello to keep people on task during our Agile sprints. This will allow us to assign cards and tasks to different people so that we can keep track of our goals for each spring. It will also allow us to monitor other people's progress so that we can know about any delays.
We will also use Discord as our means to meet as a team. This will allow us to meet electronically while allowing us to pin important conversations. It will also allow us to contact each other outside of our meetings and alert us when our team members are trying to relay important information.

2.6 Personnel Effort Requirements

Task	Effort Requirement (Person Hours)
Research Components	3
Develop Design Plan	30
Identify Use Cases and Entities	15
Develop Working Hardware Prototype	100
Develop Working Software application	100
Establish communication between hardware and software	30
Software Testing	50
Improving Prototype and Alternative Implementations	50
Feedback and Improvements	100
Total	478

Table 2.6 - Effort Requirements

The first task we had to accomplish was researching which components we needed to buy. Our project needs to be accessible to most users, so it needs to be cheap and easy to implement. We also want to see which components work the best, so we have ordered a few different components for testing.

We needed to develop a design plan which will be our course of action over the next two semesters. We will continue to revisit our plan and make changes as needed.
The third task is to develop the use cases and entities for our project. This is to identify all of the potential users for our project and all of the different components that are needed. Then, we will be

able to develop the app to be applicable to our users and to set up all of the different entities that we will need.

The next three steps are for development. The hardware team will work on building a working prototype. The software team will work on making a functioning application. Once the pieces are working correctly, we can work on establishing communication between the hardware and the software.

Over the course of the project, we will need to test the software and work out any bugs that we encounter. This will be an ongoing process.

The hardware team will need to improve on the prototype. They will also need to test which components are the most effective solution that is accessible to most users. We can then provide an analysis of which pieces are cheaper and which pieces are better depending on the plumbing knowledge of the users.

Over the course of the project, our client will be providing feedback that we will need to implement. This might cause us to need to redesign our implementation.

2.7 Other Resource Requirements

Parts we will be using:

Device	Part Name	Cost
Water flow sensor G1/2	114991173	$6.02
ultrasonic sensor	Excelity 3pcs Ultrasonic Module HC-SR04 Distance Sensor with 3pcs Mounting Bracket for Arduino	$7.99
Electric rotating motor	290-008	$19.99
Wifi module	Serial Wireless WiFi Transceiver Receiver Module	$12.98
Water ball valve, male, G1/4	Mini Ball Valve, Brass, Inline, 1-Piece, Pipe Size 1/4 in, Connection Type FNPT x MNPT	$8.65
Arduino Nano	Mini Nano V3.0 ATmega328P Microcontroller Board w/USB Cable For Arduino	$13.99

Table 2.7. Parts List

For the application, we will need various software components including, but not limited to: Android Studio, Spring Boot, Postman, MySQL, and a virtual machine to host our application. Most, if not all, of these are available to us free of charge.

2.8 Financial Requirements

Our total budget is $500. To keep the end product affordable, our goal is to keep the design below $250, if not under $150. A breakdown of the cost for each component can be viewed in the previous section.

3 Design

3.1 Previous Work And Literature

The Moen 900-002 Flo by Moen is an available market option that provides an app and a device to manually shut off the flow of water in a home. It also will record the amount of water used. While our product will function similarly to the Moen device, we are hoping to achieve the same end result at about one third of the price for the consumer.

There are other similar products, but they all cost a minimum of $500. This project will require us to learn some basic plumbing skills, and one of our group members already has that knowledge that he learned for a previous project.

3.2 Design Thinking

There are many things we had to define for this project, one of which is whom this product is for. Most homeowners and other property owners would love to have a cheap way to prevent water damage to their property. Since we are trying to make our project as widely available as possible, we had to take into consideration that most people have a lack of knowledge in regards to plumbing and piping. To make our product usable for the general public, it must be easy to implement. If it requires installation into the piping of a home, this process should be doable by most people when given detailed instructions.

When thinking about the users for this product, there were other needs we defined besides just ease of installation. One need is that the product should be cheap. There are already current solutions to this problem available, so we need to make sure that our product is competitively viable by making it a cheaper alternative. Keeping this goal in mind is what drove some of our design choices for what options were available for us. Another need that we defined is that the app should be easy to use. The features of the app should be intuitive and the actions a user can take should not be too complex. No one wants to read an instruction manual when figuring out how to work an app. If the app is too frustrating to use, users will search for a better alternative.

During the ideate phase, we came up with two possible solutions for controlling the flow of water in a property. One solution uses a sonic sensor and the other uses a waterflow sensor. Each of the solutions has its own pros and cons. The sonic sensor would be easier to implement for a user, as it can attach to the outside of a pipe. However, the sensor activates whenever the pipe vibrates, even if the vibrating is not caused by water flow. The waterflow sensor is more accurate in this regard, since it will only report data if water is sensed. However, it is more difficult to implement as it

requires access to the property's piping. In the end, we decided we would try to implement both solutions to compare and contrast the benefits of each solution.

For the software side of things, most of the ideate phase was dependent on which languages our developers were most comfortable using. For the backend, both Spring and .NET Core frameworks were considered for the app. Spring was eventually chosen for its use of java and availability of packages and plugins to streamline development.

3.3 Proposed Design

So far for the app, the backend has created an application that runs on local machines. It has a successful connection to a MySQL database and can properly store and retrieve data. The controller and service classes for the User entity have been created. The methods to create a user, get a user, get a user's username/email/phone number/password, get a list of all users, and update a user have all been tested. All of these will be used so that a user can create an account for the application. This will allow users to connect their water control device(s) to their account.

The backend still has to develop an entity for each water control device. Each device will be associated with one user. This database relationship will also need to be set up to ensure a many-to-one relationship between users and water control devices. Each device will also need to be able to contain a schedule so that certain times of the day will automatically shut off water flow if water is detected. All of this will be used to make sure that each device is registered and communicates with one account. This will also satisfy the requirement of allowing the user to create a schedule where the device will automatically stop any water flow if it is detected at certain times.

The hardware will include an arduino, the device's microcontroller, allowing the user to interact with the hardware via phone app and for it to function autonomously. In order for the user to be connected to the device, a wifi adapter will be incorporated. Currently, we are designing two devices that will achieve the same goal. The first device will include a water flow sensor that will record how long water has been flowing through the device and piping. If the sensor has been tripped for an 'x' amount of time then the water piping will get shut off, until the user tells it otherwise. Attached to this design will be a water control valve that will be switched off/on accordingly. In order to control the valve, a rotating motor must be included into the design. The control valve will also be integrated into the water piping system.

To view a visual overview of our project, refer to figure 3.72 in section 3.7 Design Option.

3.4 Technology Considerations

As far as software is concerned, we chose to use Android Studio because it supports the Java programming language. While this may limit the amount of users for the application, it is simply easier to integrate all of our ideas in Android Studio. Our software developers are more familiar with Android Studio than other app-development environments.

For the backend, we chose to use the Spring Framework since it is java based. It also provides many built-in packages which will simplify many common implementations that we will need. It is also widely used in industry, so it is good to become familiar with it for our project. It is also modular and supports dependency injection, allowing for easier testing of entities in isolation.

During our initial designs we were considering what microprocessor we should use - what would be cost effective, reasonable in size, powerful. We decided on the Arduino Nano as it has the same capabilities as a regular Arduino - it is easy to program, however it is significantly smaller in size allowing the design to be more compact. We also considered a Raspberry Pi, however that device is bigger in size and we are not quite that familiar with programming those devices.

3.5 Design Analysis

There are a few design choices. As already stated, we are planning two designs - the flow sensor being our main design and the sonic sensor our secondary. For the flow sensor we are using a specific part that will record data, and it comes with a threaded male component so it can be integrated into a piping system. Upon further analysis, however, the thread spacing is not standardized and will require a special adaptation so it can be converted to the standard spacing. We will be getting in contact with the manufacturer of the part to inquire more details about the threaded spacing. In the meantime, we will also look at other flow sensors that will not require a special adaptation.

For the sonic sensor, right now we are testing out our current sensors to see if they can detect vibrations in a flowing water pipe. If they work, then our next step would be figuring out a way to stop water flow without integrating a part into the piping system, but, rather, onto its exterior.

3.6 Development Process

We have decided on an Agile development process. This process calls for biweekly sprints and daily stand-ups. This encourages us to work early on any deadlines we set for ourselves. It also gives us many chances to communicate as a team to share any progress or challenges that we have encountered. The biweekly sprints are also conducive to our biweekly meetings with our faculty advisor and for the biweekly reports.

The Agile development process also allows us to implement any user feedback that we receive. With each sprint, we can show our progress to our client and obtain their critiques. We can then improve on anything that is not optimal or desirable.

3.7 Design Plan

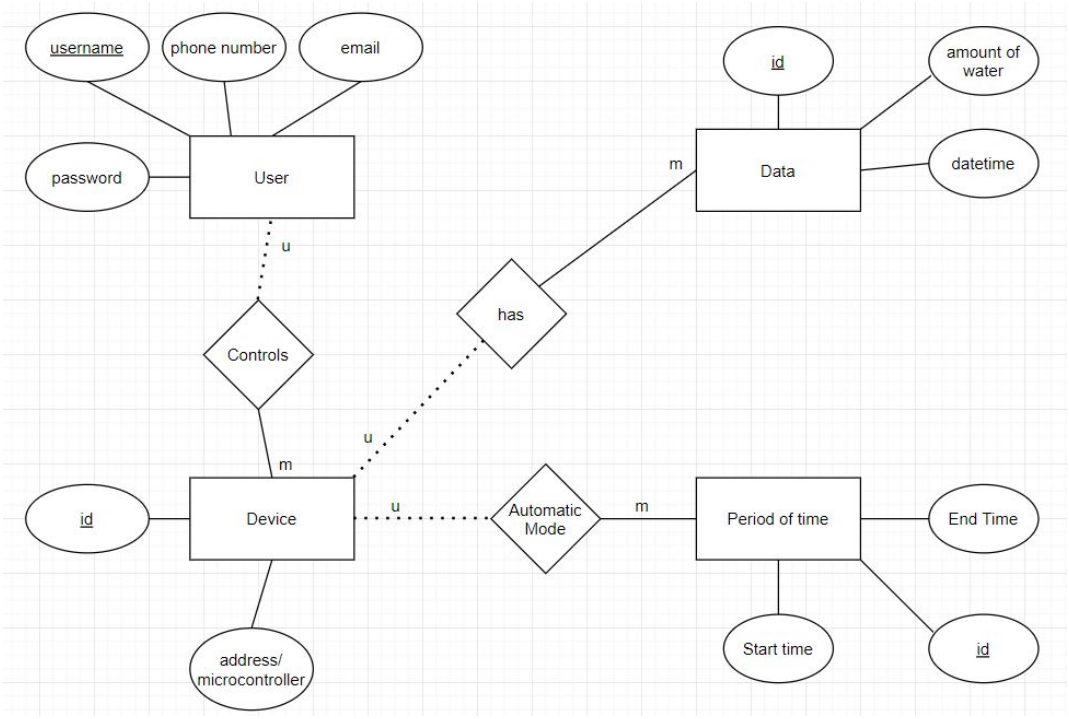


Figure 3.71 - Design Plan

Above is the relationship diagram for the database. Entities are in squares, attributes are circles, and relationships are diamonds. The primary key for each entity is underlined. The relationships are that a User can control one or more devices, and a device must have a user. A device can have multiple periods of time that it is in automatic mode. It will also have many points of data recording the amount of water flow through the pipe at a given time. This is the current design plan. More entities and attributes may be required as implementation continues.

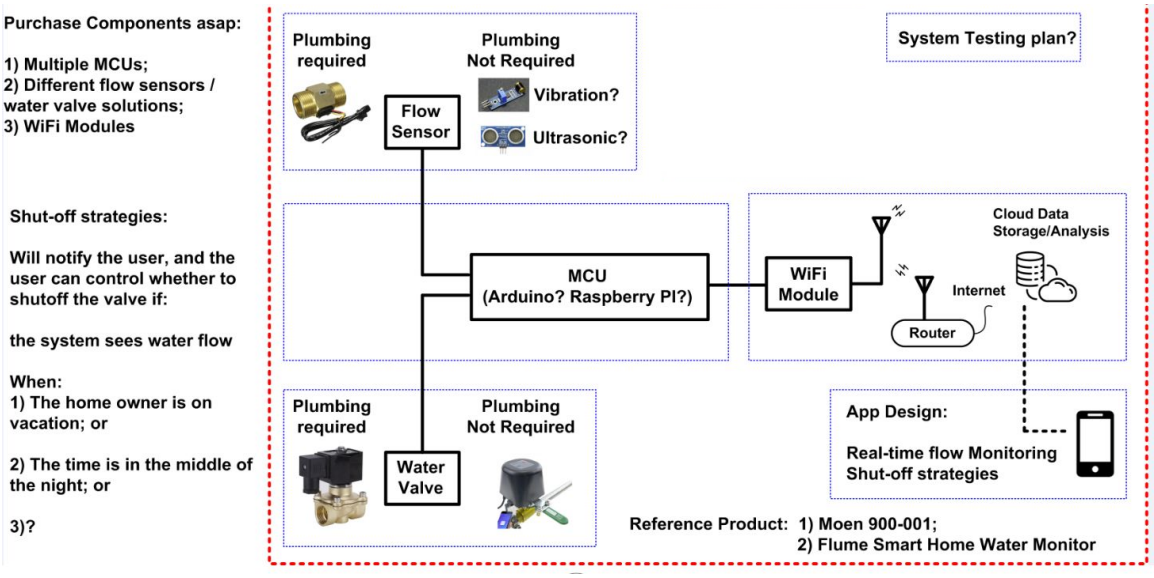


Figure 3.72 - Design Options

Our first design is the water flow sensor where it will include a sensor that will integrate into the water piping system along with a control valve. The flow sensor will be a Hall Effect sensor, where the water flow will push a hanging magnet up-horizontally creating a magnetic field that will inform the user water is flowing. When the magnet is down, the sensor will indicate zero water flow. The control valve will be a water ball valve, a device that opens and closes water flow, usually operated manually by the user, however our design will require something more automatic. We will add a rotating motor - connected to the arduino - that will be placed above the valve, and with some added programming lines the user will be able to open and close the valve with the app or automatically due to a tripped sensor.

The second design is the non-integrated sonic sensor. The idea is to create a device that will detect water flow based on the vibrations within the water pipe with the ability to open and close the valve automatically, or by the user interacting with the designed phone app. The trick is to find a part, means, that will allow a standard user to control water flow without interfacing into the water system directly, but preferably onto the exterior.

Both designs will incorporate a wifi adaptor, this will allow the user to interact with the device via the phone app. We are also including some LEDs that will visually indicate the device's current mode, if water is flowing or not.

4 Testing

On the software side, we are going to use the following testing frameworks:

Testing Framework	Purpose of Tests
JUnit	JUnit will allow us to test individual methods for correctness. This will ensure that the logic within each method is sound
Espresso	Espresso tests will be used to test the interface within the app itself. Upon clicking and entering information, it will ensure that the proper methods are called and the intended results are received.
Mockito	Mockito tests will be used to "mock" a response from the hardware component while it is still in development.

Table 4.01 - Software Testing

On the hardware side, we are looking to conduct the following methods of testing:

Testing Method	Description
Controlled environment	We will be testing our design on a homemade water piping system, to see if it works before we integrate into the real thing.
Voltage requirements for each component	Based on the devices' datasheet, we will appropriately distribute the necessary voltage to all of the components. First, we will test each component individually, then again collectively to get the voltage right.
Seeing if the sensor outputs data accurately	We will need to run a few experiments for this. The datasheet includes ways calibrating your code for it to read accurate measurements.

Table 4.02 - Hardware Testing

4.1 Unit Testing

Our application relies on properly storing the user's water usage data, as well as properly determining whether or not the water is running. Because of this, there will be a large focus on accurately receiving information from the shutoff valve, and properly storing and fetching user information.

We plan to test all entities within our database with both JUnit and Mockito tests. The mockito tests will mock responses from the database to ensure that the methods are properly handling the received data. The JUnit tests will ensure that the proper information is received from the backend. The response received from the valve itself will be tested in isolation so that the app and the valve itself are properly linked together. When the valve is on, the app must receive a signal from the valve so that it can properly record water usage.

Another big part of our project revolves around determining when the water valve is on or off. We must also determine what the water usage patterns are for our users, so that we can cater suggestions to their needs. We plan to give the users a scheduling system so that they can input necessary information, like what time they wake up or go to sleep, as well as giving them an option to input a vacation, so that the water can stay off for an extended period. We will test this extensively, and we plan to make improvements to the scheduler as we progress further.

4.2 Interface Testing

The user interface will be tested with Mockito, as well as hands on testing from our point of contact. Mockito testing will allow us to monitor the methods that are called upon input of data within the app itself. It will also allow us to see responses to inputs that are not intended so that we can cut down on app crashes.

4.3 Acceptance Testing

We plan to keep our client in the loop as we continue development of this project. Through each

iteration, we will ask for feedback from our point of contact. We plan to improve our product according to the feedback we receive until we reach a product that is able to satisfy not only the client, but also ourselves.

We plan to give the client a hands on experience with the application and the hardware for this project. We will welcome the client to attend our controlled environment tests so that he can observe the behavior of the hardware and software.

We plan to demo the functionalities of our project early and often.

We will create a prototype. Once it works by our standards, we will send it along to our point of contact for feedback.

Once controlled environment tests pass, we will move on to a real-world environment.

We will constantly get feedback and improve our product accordingly.

4.4 Results

For our current results, we are just now starting to test our components. Right now, the flow sensor male threaded component is not of standard sizing. We are currently in the process of determining the correct size so we can order a converter for it to fit in the standard threaded size that can be purchased at any hardware store.

The User entity has been tested on the web API, including all of the corresponding methods. A new user can be created without conflicting with existing usernames. The list of all users is able to be retrieved. One user can be retrieved from the database with a given username. A user's password, email, and phone number can be correctly retrieved as well. A user can also update their information as well. The data also persists through termination of the code. Error checking must still be implemented to guarantee that a valid email address and phone number are entered when creating an account.

As we continue to implement different things, we will constantly be testing our design. We will make sure each addition works in isolation, and fix any errors that we encounter. Once we are confident that the component works by itself, we can start testing it with other components that have been completed. Then we can identify any errors that occur and work to fix them.

5 Implementation

For next semester, the preliminary implementation plan is to create a prototype for the hardware component using the waterflow sensor. For the software application, the preliminary plan is to establish communication between the front end, back end, and microcontroller before implementing more complex features of the application. Once the hardware and software components have been tested in isolation, the next step for the implementation is to allow the application to communicate with the water valve to manually shut off the flow of water.

Once we are successfully able to communicate between the back end, front end, microcontroller, and water valve, we will be able to implement more complicated features for the app, such as setting the valve to automatic to automatically close if it detects water. We will also need to work on storing and retrieving waterflow data so that users can view the history of water usage.

For the hardware, once a working prototype is developed using the waterflow sensor, we plan on implementing a prototype that uses a sonic sensor. This prototype will have the same functionality, but will require less plumbing and is a potentially cheaper solution.

6 Closing Material

6.1 Conclusion

For the back end, we have a Spring application that can run on a local machine. It has a User entity set up with methods that allow for user creation, updating, and authentication. The web API for the application is working as well. The application is connected to a MySQL database, and it can successfully store and retrieve data. For the front end, we have a basic GUI set up. There is a functioning login screen as well.

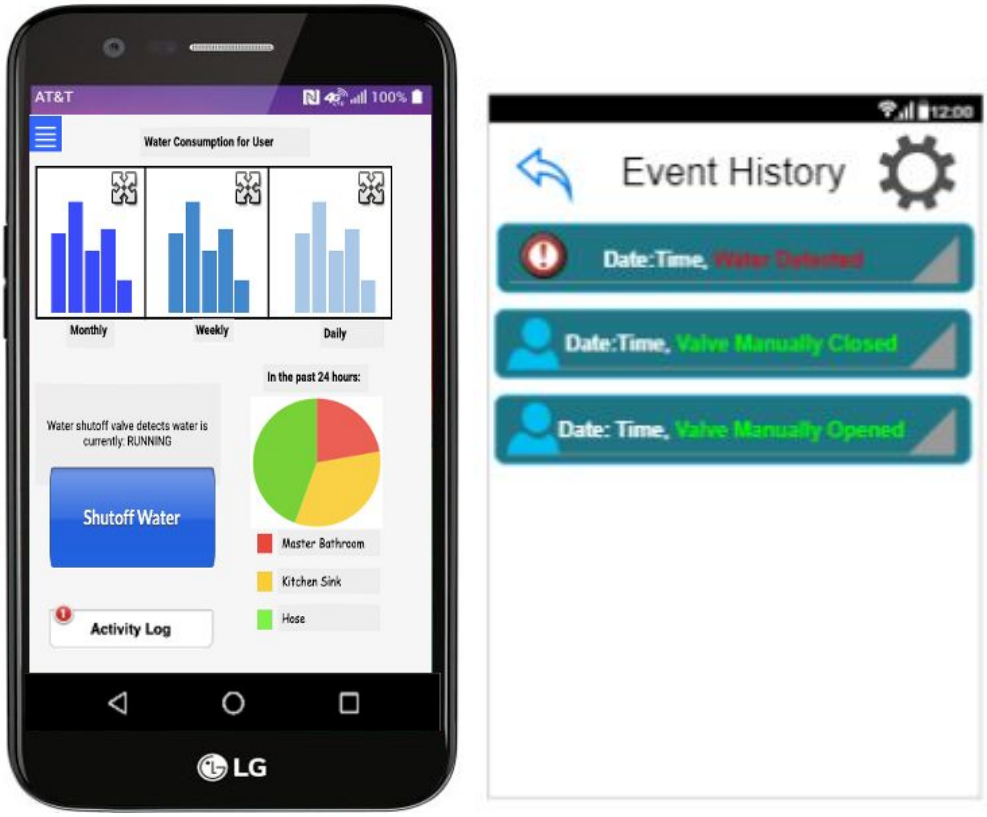
On the hardware side, we have designed and started building our plumbing prototype. By the end of the winter term we plan to have tested and completed work on the plumbing prototype. We have also begun preliminary designs for our non-plumbing prototype. By the end of the spring semester we will have our final prototype for the non-plumbing option as well as our final decision on which implementation is the best.

6.2 References

Moen Flo:
Press Release. (2020, October 27). Retrieved from Moen:
https://www.moen.com/press-room/press-releases/lennar-moen-partnership

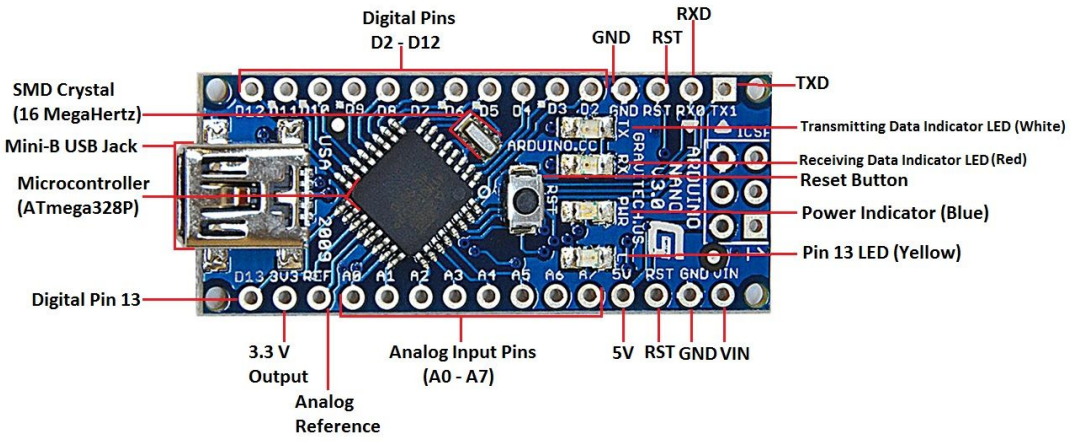
6.3 Appendices

Appendix A: Screen Sketches



Above are our designs for a home screen and an event history screen. The home screen allows the user to view the history of their water usage, and manually shut off the water valve. It also naviages to the event history screen, which shows when the water valve has been open and closed, and if it was done manually.

Appendix B: Arduino Pin Layout



The above picture of the Arduino Nano shows a visual map of the pin layout, which we will be using and referring throughout our design and testing process.

Arduino Nano Tutorial – Pinout & Schematics. (n.d.). Retrieved from Cicuits Today: https://www.circuitstoday.com/arduino-nano-tutorial-pinout-schematics

Appendix C: Parts Datasheets

Appendix C.a: Electric Open and Close Valve:

- PRODUCT DESCRIPTION
    - U.S. Solid normally open (N/O) electric solenoid valve is constructed with a durable brass body, 1/2" female threaded (NPT) connections, and nitrile rubber (NBR) seals. Suitable for use with hot or cold water, natural gas, air, very low viscosity fluids (< 20 cst) and oils, e.g.. The solenoid valves have a rating of IP65, which means they can withstand spray from water. However, if permanently installed outdoors, it is recommended that you enclose the valves in some protective housing. Not for use underwater. Customers chose this general purpose valve for: do-it-yourself projects, moderate temperature applications, water exchange for aquarium, air horn, or bleeding moisture from an air tank.
- Specifications
    - Pipe Size: 1/2"
    - Flow Aperture: 16 mm
    - Body Material: Brass
    - Gasket / Diaphragm / Seal: NBR
    - Operation Type: Semi-Direct Lift Valve
    - Operation Mode: Normally Open
    - Voltage: DC 12V ± 10%
    - Power rating: 19 W
    - Suitable Media: Air, water, oil, etc.

    - Operational pressure range: Air/Water: 0-7 Bar (0 to 101 PSI); Oil: 0-5 Bar(0 to 72 PSI)
    - Flow capacity: 4.8 gallons per minute (GPM) of water at 60 oF with a pressure drop of 1 PSI
    - Flow value: 4.8 Cv
    - Temperature, Minimum: 23 degrees Fahrenheit (-5 degrees Celsius)
    - Temperature Maximum: 176 degrees Fahrenheit (80 degrees Celsius)
- OVERHEATING WARNING: Not designed for continuous operation(not a 100% duty cycle). While energized, under normal operating conditions, solenoid valve will get HOT. The coil will reach a maximum temperature of about 176°F (80℃), which is fully complied with national standard: JB/T7352-2010. If energized continuously for more than 8 hours the coil may burn.
- WARNING: This product can expose you to chemicals including lead, which is known to the State of California to cause cancer and birth defects or other reproductive harm. For more information go to www.P65Warnings.ca.gov.
- U.S. Solid Electric Solenoid Valve- 1/2" 12V DC Solenoid Valve Brass Body Normally Open, NBR SEAL. (n.d.). Retrieved from U.S. Solid: https://ussolid.com/u-s-solid-electric-solenoid-valve-1-2-12v-dc-solenoid-valve-brass-body-normally-open-nbr-seal.html

Appendix C.b: Water Flow Sensor

    - Product description
        - Made of brass material, NPT threaded, waterproof design, prevent water ingress, heat-resistant, pressure-resistant, cold-resistant, stable and reliable.
        - Working range: 1-30L/min.
        - Maximum water pressure: 1.75MPa.
        - Rated voltage: DC5V.
        - Rated current: less than 10mA.
        - Electrical strength: AC500V, 50Hz.
- QWORK NPT Threaded 1/2" Water Flow Switch 1-30L/min, DC 5V, AC500V, SEN-HZ 21WI. (n.d.). Retrieved from Amazon: https://www.amazon.com/gp/product/B08FYKW4SH/ref=ppx_yo_dt_b_asin_title_000_so0?ie=UTF8&psc=1

Appendix C.c: Ultrasonic Sensor

- Excelity HC-SR04 Ultrasonic Moudle is to measure the distance of object by 40KHz ultrasonic sound wave. This moulde provides 2cm-400cm non-contact measurement function, the ranging accuracy can reach to 3mm. It is great for the project that need distance measurement, such as Distometer,intelligent robot navigation system,etc.
    - Product description
        - Module main technical parameters:
            - Working Voltage : 5V(DC)
            - Static current: Less than 2mA
            - Output signal: Electric frequency signal, high level 5V, low level 0V
            - Sensor angle: Not more than 15 degrees
            - Detection distance: 2cm-450cm.

- High precision: Up to 0.3cm
- Input trigger signal: 10us TTL impulse
- Echo signal : output TTL PWL signal
- Pin definition:
- VCC: connect to 5V DC
- Trig(T): control pin, connect to a standard I/O port
- Echo(R):receiving end, connect to a standard I/O port
- GND: Connect to Ground

    - Notes:
        - The module should be inserted in the circuit before been powered to avoid producing high level of misoperation; if producted, power it again.
        - Before connect to VCC terminal, the GND terminal should be connected the module first,otherwise,it will affect the normal work of the module.
    - Module Working Principle:
        - Pull the Trig pin to high level for more than 10us impulse, the module start ranging.
        - The module automatically sends eight 40KHz square wave to detect whether a signal is returned.
        - Finished ranging, If you find an object in front, Echo pin will be high level, and based on the different distance, it will take the different duration of high level.
        - Calculated the distance, the distance = (high level time * sound velocity (340M/S) / 2.
- Excelity 3pcs Ultrasonic Module HC-SR04 Distance Sensor with 3pcs Mounting Bracket for Arduino. (n.d.). Retrieved from Amazon: https://www.amazon.com/Excelity-Ultrasonic-HC-SR04-Distance-Mounting/dp/B07SC1YJ21/ref=sr_1_5?crid=38U8M6ZI6Q4WC&dchild=1&keywords=ultrasound+sensor+arduino&qid=1600399066&s=electronics&sprefix=ultrasound+sen%2Celectronics%2C159&sr=1-5