Optimizing the Automatic Release of Water for Lawn Irrigation with Household Rainwater Harvesting

Optimizing the Automatic Release of Water for Lawn Irrigation with Household Rainwater Harvesting

A senior design project report submitted in partial fulfillment of the requirements for the degree of Bachelor of Science in Engineering Sciences at Harvard University

Billy Beauregard

S.B. Degree Candidate in Engineering Sciences, Environmental Science & Engineering Track

Advisor: Professor Ken Thomas

Thesis Reader: Professor Kaighin McColl

Harvard University School of Engineering and Applied Sciences
Cambridge, MA
March 31, 2023

Table of Contents

Acknowledgements

I would like to express my deepest appreciation to my advisor, Ken Thomas, for his mentorship and guidance throughout this project. His technical expertise and moral support have proved to be invaluable and integral to the success of this project. I would also like to express my gratitude to my thesis reader, Prof. Kaighin McColl, for his advice and feedback that have aided in the completion of this project.

I am also extremely grateful for the structure and support of the entire ES 100 faculty and staff, and especially for my section leader, Bryan Yoon, for providing unparalleled guidance and reassurance that has made this project a successful and unique learning experience. Thank you also to the staff at the SEAS Active Learning Labs for being an incredible support system.

Finally, I would like to sincerely thank my family and friends for their unwavering support throughout my academic journey. Their belief in me, the laughs we've shared, and the conversations we've had have kept my motivation high, and this project would not have been possible without their continued emotional support.

Abstract

The two key precipitation trends in Massachusetts are higher-intensity storms and longer dry periods between storms, resulting in both higher flooding risks and greater drought stresses. Widespread rainwater harvesting on the household scale can alleviate both of these issues, by capturing significant amounts of rainfall and serving as a supplementary water source for lawn irrigation. As such, an engineering solution is needed to encourage the widespread use of rainwater harvesting. This project involves the design, implementation, testing, and evaluation of an algorithm that controls the automatic release of water from a lawn irrigation system that integrates a household's piped, treated water supply with the household's rainwater harvesting supply. This algorithm utilizes historical and forecasted weather data from OpenWeatherMap APIs to increase water-use efficiency. 90 simulations were run for each of 4 locations in Massachusetts using weather data from the past 10 years and various combinations of lawn size and rainwater harvesting tank size at a household. While results highlight the complexity of making lawn irrigation more efficient, they also suggest potential cost savings for consumers on the order of $1,000 dollars per household per year. The relatively simple algorithm developed in this project serves as a starting point for the improvement of lawn irrigation technology and can be expanded upon for added precision, efficiency, and cost savings.

List of Figures

List of Tables

1 Introduction

1.1 Project Summary

The impacts of climate change on precipitation trends are widespread. In particular, Eastern North America is very likely to experience increased precipitation, both in total annual depths and in extreme precipitation events [1]. Confidence in these projections is higher in northern regions [2], such that Massachusetts can almost certainly expect to experience these trends in the coming years. Increased precipitation will put further stress on drainage systems, especially in places with significant amounts of impervious surfaces.

Widespread use of rainwater harvesting (RWH) would alleviate some of this stress on drainage systems and would help with water conservation, as water collected through RWH at the household level can be used as a supplementary water source for lawn watering, a water-intensive process. Currently available RWH systems range from simple cisterns to more complex automated-release systems, but even the more complex systems lack integration with a household's primary, piped water supply. Designing a system that integrates a household's RWH water supply with its primary, piped water supply and automatically dispenses water from the appropriate source (RWH or piped water) for lawn watering would make widespread RWH at the household level more feasible. Designing such a system is the goal of this project. The automatic dispensing of water is based on an algorithm that considers the size of the household's lawn, the rainwater level/volume in the RWH tank, the weather forecast in the location of the system, and existing soil moisture of the surrounding lawn. This automatic system should not only minimize the effort required by users to water their lawns, but the algorithm should also optimize water use for lawn watering at the household, thereby resulting in water cost benefits for consumers.

1.2 Background and Motivation

Changes in precipitation trends due to climate change have been observed across the world. Particularly, significant increases in mean precipitation and extreme precipitation events have been seen in Eastern North America [2][3]. In the Northeast United States, annual precipitation rates have increased by over 1 inch per decade since the late 1800s, and additionally, there was an observed increase in amount of rainfall during extreme events by over 70% between 1958 and 2012 [4][5]. These precipitation trends are likely to continue in the Northeast United States as global temperatures continue to rise [2]-[5], and, in Massachusetts, increases in the intensity of precipitation events is the main driver of these projections [4]. These projections raise concerns for the ability of stormwater drainage systems in Massachusetts to manage increasingly heavy rain events, especially in areas containing significant impervious coverage. Undersized stormwater systems result in more runoff over impervious surfaces and, therefore, increased flooding and negative impacts on water quality.

Despite projections for increased mean annual precipitation and more extreme rain events, there are also concerns for drought conditions becoming more frequent in Massachusetts. Projections indicate that precipitation events during Massachusetts summers will have increased intensity but will also be separated by longer periods of consecutive dry days [4]-[6]. Therefore, not only are there concerns for stormwater drainage systems and flooding, but also for increased dry periods in the summer months. As of August 16, 2022, 157 municipalities in Massachusetts had implemented mandatory water restrictions [7], and similar restrictions are likely to be seen in future summers if trends continue.

Making upgrades to stormwater drainage systems to increase capacity and implementing water restrictions when necessary would both help alleviate these issues at hand. Additionally, green stormwater infrastructure projects, a subset of stormwater improvement projects that take a more sustainable approach to addressing increased precipitation volumes, have become more prevalent in cities, such as an involved program in Philadelphia [8][9]. However, stormwater improvement projects can be expensive and long processes, and water restrictions likely would not be well-received by the general public, so getting other solutions in motion would be helpful. Widespread use of rainwater harvesting (RWH) on the household scale is one strategy that could impact both challenges suggested by precipitation projections. On one hand, the water volume collected with RWH would reduce the amount of water entering the stormwater drainage systems during a given rain event, and on the other, the water collected via RWH could be used as a supplementary water source for lawn watering, a water-intensive activity, during the longer dry periods in between summer storm events. As will be discussed in Section 2, current household RWH systems are not optimized to address both challenges arising from precipitation trends, and therefore, a new approach to RWH must be developed.

1.3 Project Goals and Problem Statement

This project seeks to design a RWH system that makes widespread RWH at the household level in Massachusetts more feasible, with the overarching idea that widespread RWH will help to address the precipitation trends in Massachusetts. In working towards this goal, there are some non-technical factors to consider. For one, there must be buy in from the public, such that adoption of the RWH system is socially accepted. This likely requires a RWH system that is low-effort and easy-to-use. Additionally, ensuring that the RWH system would allow for reductions in water use and therefore monetary savings for consumers would help to get public buy-in. Overall, the proposed solution should satisfy the technical goals (alleviate drainage system stresses and providing a supplementary household water source for irrigation), but should also maximize the social and economic benefits to gain consumer buy-in. Thus, this project offers a solution to the following problem: how can household RWH systems be optimized to serve as a supplementary water supply for lawn irrigation while minimizing the effort required by users to encourage widespread RWH?

The focus of this project is on Massachusetts specifically, partly because the precipitation trends and projections described above are especially strong for the Northeast United States, and partly because I have lived my whole life in Massachusetts and will begin my career in the state, too. As such, I have a strong interest in improving water systems in Massachusetts and have pursued one potential solution through this project.

1.4 Potential Users and Stakeholders

The targeted end user for this project solution is single-family homeowners in Massachusetts, and specifically, those with lawns on their property. These homeowners may feel that consistently maintaining their lawn is challenging, either because of the cost of using so much water each day or the effort required to water their lawn daily – or both. This project could address both of these challenges: RWH would reduce water costs by reducing the amount of piped water used for irrigation, and an automatic-release, fully integrated RWH system would allow for minimal effort by the end users.

Additionally, the design focus for this project does not include a RWH tank, but rather this project has a focus on the automatic release of water for irrigation and the integration of a RWH system with a household's primary piped water supply. In this sense, the product of this project will reside in a larger system, and thus a RWH cistern/tank would also be considered an end user for this project, with the ultimate end user being homeowners.

2 Existing Solutions and Previous Work

2.1 Existing Solutions

At the most basic level, a household RWH system consists of a downspout from roof gutters connected to a cistern storage tank, typically with an overflow and outlet pipe, as seen in Joseph Taborek's US patent [10]. This project seeks to integrate a RWH cistern like Taborek's with a household's piped water supply, and to provide the additional functionality of automatic water release for lawn irrigation. There have been some efforts to automate the release of water from RWH cisterns, such as John Larrison's now expired US patent for an "Automated rainwater collection system controller", which utilizes electrical communication between multiple pumps, valves, and pipes [11]. This project uses similar concepts for the integration of the RWH cistern with a household's piped water supply and determining which water supply to pull from at a given time.

In addition to existing solutions for RWH specifically, there is existing work related to automated lawn irrigation systems that this project seeks to build on. Although there is significant previous work with both RWH systems and automated irrigation systems, the key needs that must be addressed are integration and increased efficiency for irrigation. That is, a RWH collection system must be integrated with a household's primary, piped water supply, and the automatic release of water from this integrated system needs to be optimized for lawn irrigation. Table 2.1 below summarizes existing work that will help to inform this project.

Table 2.1: Existing Work Informing the Design

Patent	Summary
US 8,881,756 B1: "System for harvesting rainwater" [10]	Simple RWH system. A collection tank with a downspout from a roof as the inlet, an overflow pipe, and an outlet pipe.
US 9,633,532 B1: "Automated rainwater collection system controller" (expired) [11]	Electrical communication with pumps, valves, and pipes for automated control of a RWH system
US 6,850,819 B1: "Irrigation control system" (expired) [12]	Irrigation control system that utilizes rainfall data and moisture content to determine watering schedule.
US 10,225,997 B1: "Smart sprinkler system and method" [13]	Automated sprinkler system to prevent over-watering. Utilizes rainfall information via radar data to inform watering schedule.

2.2 Previous Work and Engineering Background

I have not conducted any previous work that is directly related to this project. Table 2.2 shows the engineering classes that serve as a foundation for this project.

Table 2.2: Engineering Courses Serving as a Foundation for this Project

Course	Summary
ES 91hfr / ES 105hfr: Humanitarian Design Projects	During this course, students perform work for Harvard's chapter of Engineers Without Borders. I have taken this course several times and have been a part of EWB since my sophomore year, and I was a Project Lead for the team during my junior and senior years. The EWB project I have been a part of is for designing a water distribution system for a small community in the Dominican Republic, so this course serves as a strong foundation in water systems in general. Additionally, this course has given me exposure to various softwares (Civil 3D, Revit, EPANET, etc.)
ES 123: Intro. to Fluid Mechanics & Transport Processes	Some topics covered in this course help with the design of the piping aspect of the RWH system (i.e water flow through the pipes).
ES 96: Engineering Problem Solving & Design Project	During my ES 96 project, I gained exposure to working with Arduinos, which is useful for the automation / electrical control aspect of my project.
PHYS 113: Electronics for Physicists	I took this course during the Fall 2022 semester. This was a laboratory course and gave me further exposure to working with circuits and Arduinos.

3 Design Independent Technical Specifications

Table 3.1 displays a summary of each design-independent technical specification for this project, and the following subsections describe, in further detail, the given specifications, the method for measurement, and the justification for each.

Table 3.1: Design Independent Technical Specifications

Specification	Value	Measure	Justification
Average total water released for irrigation per week	< 1" (25.4 mm)	Simulating irrigation during watering season based on previous years' rainfall & soil moisture data	Recommendations for lawn watering from EPA and municipalities in MA are for 1" (25.4 mm) per week [14][15][16]
RWH supply's share of total water released	> 60%	Simulating irrigation during watering season based on previous years' rainfall and soil moisture data	Maximum share from RWH is ~86%, based on 2014-2020 rainfall data [17]. Factor of 1.5 yields ~60% as a reasonable value.
Total outdoor water use from piped water supply for a typical single-family household per watering season	Avg. < 12,000 gal (45,425 L)	Simulations for multiple household types (different lawn, tank sizes) based on previous years' rainfall & soil moisture data	Based on total outdoor water demand for Massachusetts Water Resources Authority [18], scaled down to household water use.
Rainwater Harvesting Tank Size Compatibility	250 – 5000 gal (946 – 18.927 L)	Check if tank size input to system allows this range	Typical RWH tank sizes (above ground on small end, underground storage on large end) [19]
Lawn Size Compatibility	1500 – 22,000 sq. ft. (139 – 2044 m^2)	Check if lawn size input to system allows this range	Typical lawn sizes in Boston determine minimum, average lawn sizes in MA with 1.5x factor determine maximum [20][21]

3.1 Average Water Released for Irrigation Per Week

The United States Environmental Protection Agency (EPA) recommends that a household's landscape will typically require one inch of water per week [14]. Municipalities in Massachusetts share this recommendation [15][16]. This one-inch requirement includes rainfall, so assuming that rainfall will account for part of this one inch during at least some weeks, a solution for this project should release less than the required one inch, on average, in order to accomplish efficient water use.

To measure whether this specification is satisfied by this project, simulations were run over the course of a watering season using previous years' rainfall and weather data, based on the automated water release system that was developed, as described in Section 5.1. The total volume of water released from the system over the course of the simulated watering season was obtained, and a weekly average depth was calculated based on the number of weeks in the watering season and the area of the lawn that the simulation is run with as an input.

3.2 Rainwater Harvesting Supply's Share of Total Water Released

The solution developed in this project should result in water cost reductions for users. As such, the developed system aims to supply some share of the water released for irrigation using the RWH supply, in order to offset some of the household's piped water use.

The Massachusetts Department of Conservation and Recreation (Mass DCR) maintains monthly average precipitation data for the drought regions of Massachusetts [17]. Using the Mass DCR average precipitation data from 2014-2020, and assuming a total rainfall needed for sufficient water supply based on the 1" (25.4 mm) per week assumption, the maximum share of water for irrigation from a RWH supply was calculated for each month of the watering season (April – October). Overall, for the watering season, the maximum share from RWH is 86%. Using a 1.5 contingency factor to account for the fact that some households' RWH tanks may not be large enough to hold sufficient water to supply this maximum possible share, 60% emerges as a reasonable value for this specification.

To measure this specification for the project, the simulations described in Sections 3.1 and 5.1 were run, and the percentage share of water released from the RWH system was obtained.

3.3 Total Outdoor Water Use from Piped Water Supply for Typical Single-Family Household Per Watering Season

This specification further quantifies the water cost reductions that this project offers, as the determined value is based on the total outdoor water demand for the Massachusetts Water Resources Authority (MWRA) [18]. The MWRA provides the full water supply to 29 municipalities in the Boston Metropolitan area, and thus its data offers a reliable picture of water demand in Massachusetts. The average outdoor water demand of 17 million gallons per day (MGD) over the last 20 years from MWRA was scaled down based on several assumptions in order to obtain a value for the outdoor water use for a typical single-family household in Massachusetts over the course of a watering season. Assumptions and data used in this scale-down calculation include:

- The approximate number of households in MWRA's full-service area, from the MWRA website [22]
- The percentage of single-family households in the MWRA service area, based on US Census data for Essex, Middlesex, Norfolk, and Suffolk counties [23]
- An assumption that 70% of single-family households in the MWRA service area have a lawn, and that they water that lawn – approximately 35% of the full-service flow share is from Boston, so the assumption that some homes would not have a lawn is reasonable since Boston is a densely populated city
- An assumption that 17% of water demand is lost to leaks in water main piping [24]

The result of the scale-down calculations was a value of approximately 12,000 gallons (45,425 L) of outdoor water use per household per watering season. A system is successful under this specification if it releases less than 12,000 gallons of water from the piped water supply over the course of a watering season for a typical household in the MWRA full-service area. To measure for this specification, the simulations described in Sections 3.1 and 5.1 were run for a variety of lawn area and tank size inputs that are reasonable for the MWRA full-service area, and the water released from the piped water supply over the course of the watering season was tracked across each of the simulations.

3.4 Rainwater Harvesting Tank Size Compatibility

The system developed in this project should be compatible with a wide range of RWH tanks in order to fulfill the requirement of encouraging widespread household RWH. The range of sizes for this specification is based on a manufacturer's available rainwater tank sizes [19]. The low end of the range, 250 gallons (946 L), is intended for a smaller rainwater tank above ground, with above ground tanks ranging up to about 1000 gallons (3,785 L). The high end of the range accounts for compatibility with underground storage tanks. Some tank sizes on the lower end of this 250–5000-gallon (946 – 18,927 L) range may not be sufficient to meet some of the other specifications related to piped water use reductions. However, this range is kept intentionally wide to ensure that widespread RWH can be encouraged by the project solution.

Measurement of this specification was completed by determining whether the developed system has an input for tank size, and whether the system can take inputs on either end of the specified range.

3.5 Lawn Size Compatibility

The system developed should also be compatible with a range of lawn sizes representative of lawns in Massachusetts. The median lot size in Boston for single-family homes is approximately 4900 square feet (455 m^2) [20], suggesting a typical lawn size in Boston of around 3000 square feet (279 m^2). To account for the smaller lawns in Boston, a lower bound of 1500 square feet (139 m^2) was determined. The average lawn size across Massachusetts is 14,520 square feet (1,349 m^2) [21], and a factor of 1.5 to account for the larger lawns yields an upper bound of 22,000 square feet (2,044 m^2). Thus, to encourage widespread RWH, the system developed for this project should be able to operate with this wide range of lawn sizes, 1500—22,000 square feet (139 – 2,044 m^2).

Measurement of this specification was completed in the same fashion as described in Section 3.4 to determine whether the developed system can take lawn size inputs on either end of the specified range.

4 Design Approach

4.1 Approach Overview

The design approach for this project is geared towards a system that automatically releases water for lawn irrigation at a household from either the household's piped, treated water supply or from the household's RWH tank supply. The ultimate goal of this system is to integrate a household's RWH supply with the primary, piped water supply and to optimize the release of water from this integrated system for irrigation. There are three main components to the overall design of this system:

- An automated water-release algorithm
- An Arduino-based electrical system
- A mechanical valve system

The design is centered around the development of an algorithm that determines whether a household's lawn needs to be watered at a given time, and if so, how much water should be released, and which water supply should be used. The electrical system component utilizes sensors and internet connectivity to provide data as parameters for the algorithm. Additionally, the electrical system communicates with the mechanical valve system to open or close valves to the two water supplies (piped or RWH) based on the output of the water-release algorithm.

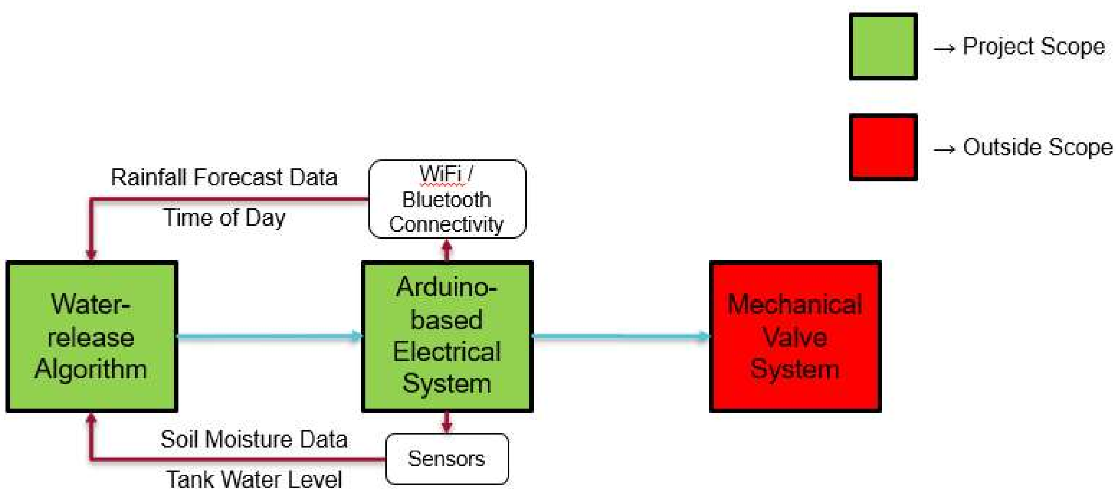


Figure 4.1: Block diagram showing the three design components. The water-release algorithm and the Arduino-based electrical system are within the project scope. The mechanical valve system is outside the project scope.

The key design components and primary scope of this project are the algorithm and the electrical system. The algorithm is the most innovative and new approach to the problem at hand, and the electrical system is needed to provide the necessary data for the algorithm to run. Due to time and budget constraints, the mechanical valve system is outside the scope of this project, even though it is part of the overall system solution.

Figure 4.1 displays a block diagram showing the three design components and how they interact with each other.

4.2 Design Components and Design Dependent Technical Specifications

Table 4.1 summarizes the design-dependent technical specifications for this project. The following subsections provide details for the two design components in the scope of the project: the water-release algorithm and the Arduino-based electrical system.

Table 4.1: Design Dependent Technical Specifications

Specification	Value	Justification
Arduino Board with Wi-Fi or Bluetooth Connectivity	Arduino MKR WIFI 1010	Electrical system must have internet connectivity to obtain rainfall forecast data and time of day in real-time
API to Obtain Weather Data	OpenWeatherMap APIs for Historical Weather Data and 5-day Weather Forecast Data	Various weather data are required as inputs and parameters to the algorithm
Minimum Nominal Pipe Size*	½”	Massachusetts code 248 CMR 10 requires ½” minimum for hose connections [25]
Device to Protect Against Backflow*	N/A	Required by Massachusetts code 248 CMR 10 because system will connect potable water (piped system from water main) with non-potable water (RWH supply) [25]

* Part of the mechanical system and thus outside the project scope

4.2.1 Component 1: Water-Release Algorithm

4.2.1.1 Overview: A Mass-Balance Framework

The overall design of the water-release algorithm is centered around a mass-balance framework. The soil water content of a household’s lawn is monitored in comparison to the soil’s available water capacity (AWC), which is the maximum amount of water stored in the soil that can be extracted by the grass roots. A minimum soil water content threshold was set, using a management allowable depletion (MAD) of 50% of the AWC, and irrigation is triggered when the soil water content in the lawn dips below this threshold. Once irrigation is triggered, the algorithm uses a mass-balance framework to determine how much water must be released from the system in order to bring the soil water content back to field capacity (FC), which is 100% of the AWC. This framework takes into account the potential for rainfall to supply some of this water, thus optimizing the amount of water that is released from this system. A visualization of the water content available to plants in soil is shown in Figure 4.2. Various inputs and data parameters used in the algorithm are described in further detail in Section 4.2.1.2.

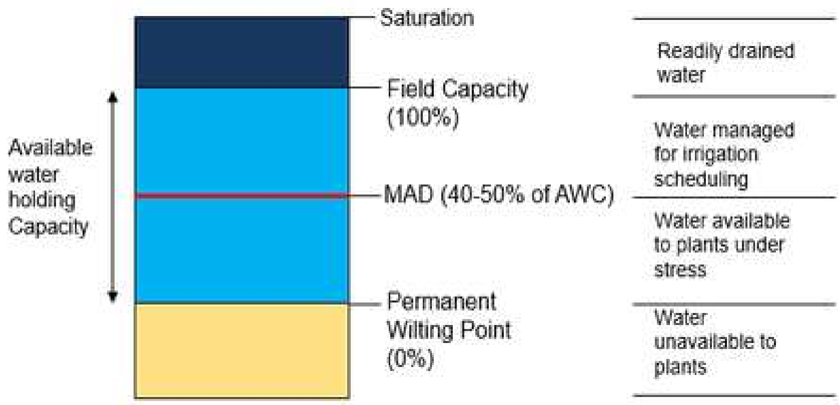


Figure 4.2: A visualization of the water available to plants in a typical column of soil [26]. The available water capacity (AWC) refers to the maximum amount of water available to be extracted by plants; field capacity (FC) refers to a soil water content at 100% of the AWC; and the management allowable depletion (MAD) is a value set by irrigation managers that refers to the maximum amount of water allowed to be taken up by crops before irrigation is triggered.

4.2.1.2 Inputs and Parameters

There are two required inputs for the water-release algorithm that serve to calibrate the system to an individual household. These two parameters are:

- The size of the RWH tank at the household
- The size of the lawn at the household

Additionally, the algorithm includes four key parameters that are captured in real-time. These parameters include past weather data, a soil moisture calculation, weather forecast data, and the water level in the RWH tank. The following subsections describe these parameters in further detail, including how these parameters are obtained.

4.2.1.2.1 Past Weather Data

Weather data from previous days is obtained using OpenWeatherMap's History API [27]. The OpenWeatherMap API contains free options, which is useful for maintaining a low cost for consumers and is one of the most widely used APIs for obtaining weather data.

This API is used to obtain temperature, relative humidity, wind speed, and precipitation data for use in the water-release algorithm. This past weather data is used in an estimation of the soil water content in the household's lawn at a given time. The method for this estimation calculation is described in Section 4.2.1.4.

4.2.1.2.2 Soil Moisture Calculation

Arduino-compatible soil moisture sensors are low-cost, but they output a relative soil moisture value, and the accuracy of calibrating these sensors to a known source can vary. More reliable soil moisture sensors exist, but they are more expensive, which would increase costs for consumers and thus are counterintuitive to the overall goal of designing a system that is widely accessible and encourages widespread RWH use. As such, because of the inability to find a reasonable balance between cost and reliability, a soil moisture sensor was considered but not utilized in the final design of the algorithm. Instead, a method to calculate the estimated soil water content of the lawn is used, as described in Section 4.2.1.4

4.2.1.2.3 Weather Forecast Data

Weather forecast data is acquired using OpenWeatherMap's 5 Day / 3 Hour Forecast API, which provides 5-day weather forecasts for any location in 3-hour timesteps [28]. Precipitation data obtained from this API is used to determine the potential for rainfall to supply water to the lawn, which aids in determining how much water needs to be released from the irrigation system at a given time.

4.2.1.2.4 Water Level in RWH Tank

The volume of water in the household's RWH tank is required by the system to determine which water source to release water from when irrigation is triggered. A pressure sensor within the RWH tank is likely the best strategy for acquiring this parameter. However, since the mechanical component of the system is outside the scope of this project, this project does not include a physical sensor. Instead, for testing the algorithm, the water volume in the tank is estimated using precipitation data and the amount of water released from the RWH water supply. To calculate the amount of water collected by the RWH tank based on precipitation volume, Equation 4.1 is used, adapted from [29]:

$$V_{supply} = A \times P \times C, \tag{4.1}$$

where V_{supply} is the amount of rainfall collected, A is the collection surface (roof) area, P is the precipitation depth, and C is the runoff coefficient. The algorithm assumes:

- a roof area of 1500 square feet (139 square meters) based on Google Earth measurements of various single-family households in Massachusetts, and
- a runoff coefficient of 0.90, which is the value for asphalt roof and is the more conservative value to use [29].

4.2.1.3 Soil Water Content Threshold

A key aspect of the algorithm design is determining a minimum threshold for the soil water content in a household's lawn that triggers irrigation. This threshold was determined using Equation 4.2, consisting of the product of the available water capacity per unit depth of the soil, the management allowable depletion as a fraction of the AWC, and the root depth of the crop of interest.

$$WC_{min} = AWC \times MAD \times root\ depth \tag{4.2}$$

For grass as the crop, the recommended MAD is 50% of the AWC [30].

Because water movement within soil and irrigation decisions are highly dependent on the specific soil and grass type in question, assumptions have been made in this respect to determine the appropriate AWC and root depth values. The following subsections describe the soil and grass type assumptions, the selected values for AWC and root depth, and the resulting soil water content threshold.

4.2.1.3.1 Soil Type Assumptions and AWC Value Selection

To determine the appropriate soil type to assume for this system, a qualitative analysis of the US Department of Agriculture soil maps in Massachusetts [31] was performed. Based on this qualitative analysis, the following four soil series were determined to be representative of common soils in Massachusetts:

- Merrimac series – sandy loam
- Hollis series – fine sandy loam
- Paxton series – course sandy loam
- Canton series – fine sandy loam

As such, the algorithm assumes a sandy loam for the soil type. For sandy loams, the typical AWC range is 1.3–1.6 in/ft (108–133 mm/m) [32], so an AWC of 1.45 in/ft (121 mm/m) was assumed as part of the algorithm design.

4.2.1.3.2 Grass Type Assumption and Root Depth Value Selection

Common grass types used in Massachusetts lawns are cool season grasses, including Kentucky bluegrass, perennial ryegrass, tall fescue, and fine fescues [33]. Of these common types, Kentucky bluegrass is the most widely used and therefore serves as the assumption for grass type. Kentucky bluegrass roots are most highly concentrated in the upper 10" (254 mm) of soil [34], and thus an assumption for a 10" root depth was used to calculate the soil water content threshold, such that the system maintains sufficient soil moisture in the most concentrated root section. Figure 4.3 displays a typical root concentration profile for Kentucky bluegrass.

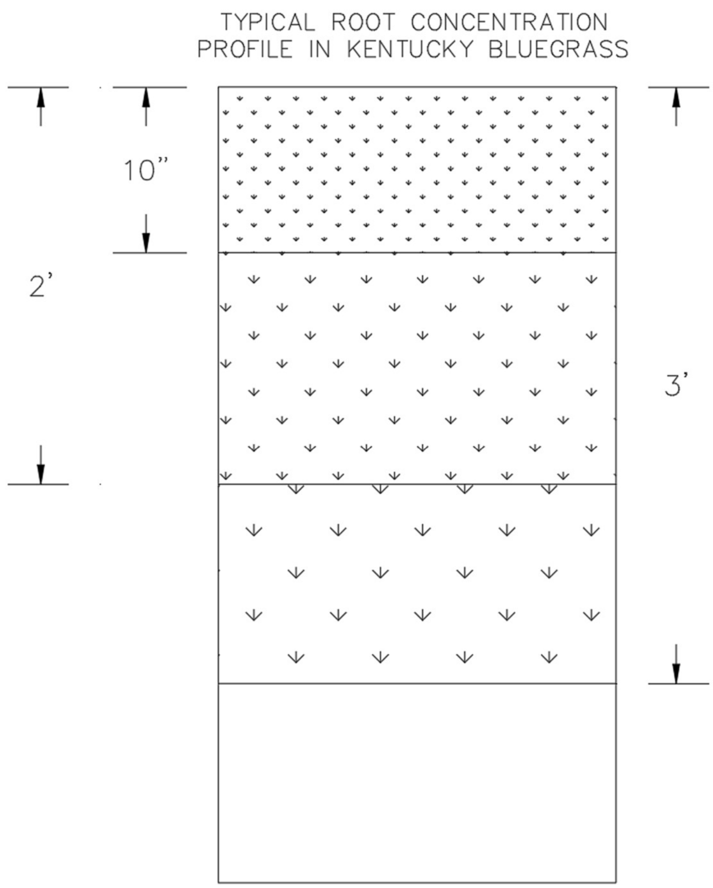


Figure 4.3: The typical root concentration profile for Kentucky bluegrass, created using information from [34]. The highest concentration of roots is in the upper 10" (254 mm) of soil; the majority of roots are located within the upper 2' (0.61 m) of soil; and some roots reach up to 3' (0.91 m) in depth.

4.2.1.3.3 Resulting Soil Water Content Threshold

With these soil and grass type assumptions, and the corresponding values for AWC and root depth, Equation 4.2 was used to determine the resulting soil water content threshold:

$$WC_{min} = 0.60\ in.\ (15.24\ mm)$$

When the soil water content drops below this minimum value, irrigation is triggered by the algorithm.

4.2.1.4 Obtaining Soil Water Content Value

Each run of the algorithm requires a value for the soil water content that can be compared to the water content threshold to determine whether irrigation should be triggered. As discussed in Section 4.2.1.2.2, low-cost soil moisture sensors are inconsistent with respect to accuracy, and thus a method to calculate the soil water content using various data parameters is used instead.

The method to calculate the soil water content follows a mass-balance framework, considering inputs and outputs to the soil water content. Water inputs to the soil considered for this method include infiltration from precipitation (F_P) and infiltration from irrigated water (F_I), and water losses considered are evaporation and transpiration, encompassed in one evapotranspiration value (ET_c). Thus, the water content at a given time (WC_t) is calculated using Equation 4.3:

$$WC_t = WC_{t-1} + F_P + F_I - ET_c, \tag{4.3}$$

where WC_{t-1} is the previous soil moisture value and all parameters are in units of water depth.

Another potential water input to the soil is infiltration from groundwater, however this method ignores that input because with a focus on solely the upper 10" of soil, infiltration from groundwater is negligible.

The following subsections outline the methods used for calculating infiltration into the soil and evapotranspiration rates.

4.2.1.4.1 Infiltration Calculation Method

To calculate the amount of water infiltrated into the soil from precipitation and irrigated water, an adaptation of the Natural Resources Conservation Services (NRCS) Method for calculating rainfall excess is used [35]. The NRCS Method indicates that infiltrated water is given by Equation 4.4:

$$F = \frac{(P-0.2S)S}{P+0.8S}, \tag{4.4}$$

where F is the depth of water infiltrated into the soil, P is the precipitation depth, S is the total surface storage, and all units are in millimeters. The total surface storage is given by Equation 4.5:

$$S = \frac{25400}{CN} - 254 \tag{4.5}$$

where CN is the runoff curve number. Assuming grass as the crop and Hydrological Group A as the soil type, the runoff curve number is 65 [36], which yields:

$$S = 136.77\ mm.$$

Substituting this value into Equation 4.4 results in the infiltration calculation used for this algorithm, Equation 4.6:

$$F = \frac{(P - 27.35)\ \times\ 136.77}{P\ +\ 109.42} \tag{4.6}$$

This NRCS method is intended for determining the water infiltrated from precipitation, but this same method is also used for determining the water infiltrated from irrigation, since there is no standard method of calculating infiltration from irrigation.

4.2.1.4.2 Evapotranspiration Rate Calculation Method

There are several methods available for obtaining evapotranspiration rates. Among these are:

- Temperature Method (Blaney-Criddle)
- Energy Method (Penman-Monteith)
- Radiation Method
- Evaporation Pan Method

Of these methods, the energy method is most accurate for irrigation scheduling on a daily basis [37]. Additionally, there are monthly average evapotranspiration estimates available for Boston, MA and Worcester, MA that could be used [38]. However, this system is intended to be more precise, in both time and space, when estimating the soil water content. As such, the algorithm utilizes the Food and Agriculture Organization's (FAO) Penman-Monteith method to calculate the evapotranspiration rate for a given day.

Equation 4.7 is the FAO Penman-Monteith Equation [39]:

$$ET_0 = \frac{0.408\Delta(R_n - G) + \gamma \frac{900}{T + 273} u_2(e_s - e_a)}{\Delta + \gamma(1 + 0.34u_2)} \tag{4.7}$$

where ET_0 is the evapotranspiration rate [mm day^{-1}], Δ is the slope of the vapor pressure curve [kPa °C^{-1}], R_n is the net radiation at the crop surface [MJ m^{-2} day^{-1}], G is the soil heat flux density [MJ m^{-2} day^{-1}], γ is the psychrometric constant [kPa °C^{-1}], T is the mean daily air temperature [°C], u_2 is the wind speed at 2 meters height [m s^{-1}], e_s is the saturation vapor pressure [kPa], and e_a is the actual vapor pressure [kPa].

Wind speed and temperature data are readily available via the OpenWeatherMap API and thus those measurements are used directly in the equation. Detailed calculation procedures contained in [40] are used in determining the remaining parameters' values, since measured values are not readily available. A summary of these procedures and relevant assumptions are as follows.

On a daily time scale, the soil heat flux density is negligible, so $G = 0$ is assumed in the algorithm.

The psychrometric constant is given by Equation 4.8:

$$\gamma = a_{psy}P \tag{4.8}$$

where a_{psy} is a coefficient dependent on the psychrometer being used and P is atmospheric pressure [kPa]. Asmann type psychrometers are the most used, so a value of $a_{psy} = 0.000662$ is assumed.

Saturation vapor pressure and the slope of the vapor pressure curve are both estimated using temperature data obtained from the API. Specifically, saturation vapor pressure for a given temperature ($e^o(T)$) is given by Equation 4.9:

$$e^o(T) = 0.6108 \exp\left(\frac{17.27T}{T + 237.3}\right) \tag{4.9}$$

where T is the air temperature [°C]. The saturation vapor pressure used in the Penman-Monteith Equation (e_s) is the mean between the saturation vapor pressure at the maximum temperature (T_{max}) and the minimum temperature (T_{min}) in the given day, as shown in Equation 4.10:

$$e_s = \frac{e^o(T_{max}) + e^o(T_{min})}{2} \tag{4.10}$$

The slope of the vapor pressure curve is calculated with Equation 4.11:

$$\Delta = \frac{4098\, e^o(T_{mean})}{(T_{mean} + 237.3)^2} \tag{4.11}$$

where T_{mean} is the mean air temperature on the given day.

The actual vapor pressure (e_a) is calculated from temperature and relative humidity data obtained from the API, following Equation 4.12:

$$e_a = \frac{e^o(T_{min})\frac{RH_{max}}{100} + e^o(T_{max})\frac{RH_{min}}{100}}{2} \tag{4.12}$$

where RH_{max} and RH_{mi} are the maximum and minimum relative humidity [%] for the given day.

Net radiation is estimated with extensive calculations using time, location, and temperature data obtained from the API. Details for these calculations can be found in [40].

4.2.1.5 Determining the Amount of Water Needed from Irrigation

Once the soil water content is below the minimum threshold, the algorithm must determine the appropriate amount of water to release from the system in order to bring the soil moisture back to field capacity. The algorithm takes into account the predicted rainfall in this mass-balance calculation. As such, the first step after the soil water content dips below the threshold is to use the API to query for the projected rainfall in the next three days. In doing so, the algorithm only considers rainfall projections with at least a 60% chance of occurring, as anything less than this is considered unreliable.

After the rainfall projections are obtained, the algorithm calculates the amount of water to be released from the system using Equation 4.13:

$$I = (FC - WC - F_P) + SF \tag{4.13}$$

where I is the amount of water to be released for irrigation, FC is the field capacity of the soil, WC is the soil water content at the time of irrigation, F_P is the expected amount of water infiltrated into the soil from projected rainfall, and SF is a safety factor to account for irrigated water that does not infiltrate into the soil. Both F_P and SF are calculated using the NRCS Method for infiltration discussed in Section 4.2.1.4.1.

These calculations are in units of depth of water (e.g., inches or millimeters), and the lawn size input is used to convert this depth into a volume of water (e.g., gallons or cubic meters).

4.2.1.6 Determining Which Water Source to Use

To determine which water source to open once irrigation is triggered, the volume of water needed for irrigation is compared to the volume of water in the household's RWH tank at the given time. If the volume of water needed is less than that in the RWH tank, then all the water for irrigation can be supplied from the RWH source. Otherwise, the algorithm instructs the system to release water from the RWH supply until the tank is at a critically low level, and then release the remaining water needed from the household's piped water supply. This "critically low level" in the RWH tank is reached when the water level in the tank reaches 2" (50.8 mm). This threshold will ensure that some water remains in the tank, such that the pump in the tank never runs dry, as running a pump in the absence of water can damage the pump.

4.2.1.7 Algorithm Flowchart Visualization

Figure 4.4 displays a flowchart outlining the key steps in the water-release algorithm design. These steps include:

1. Determining the water content in the soil
2. If the water content is below the threshold, querying for project rainfall in the next three days
3. Calculating the amount of water needed for irrigation release
4. Determining which water supply to open when irrigation is triggered

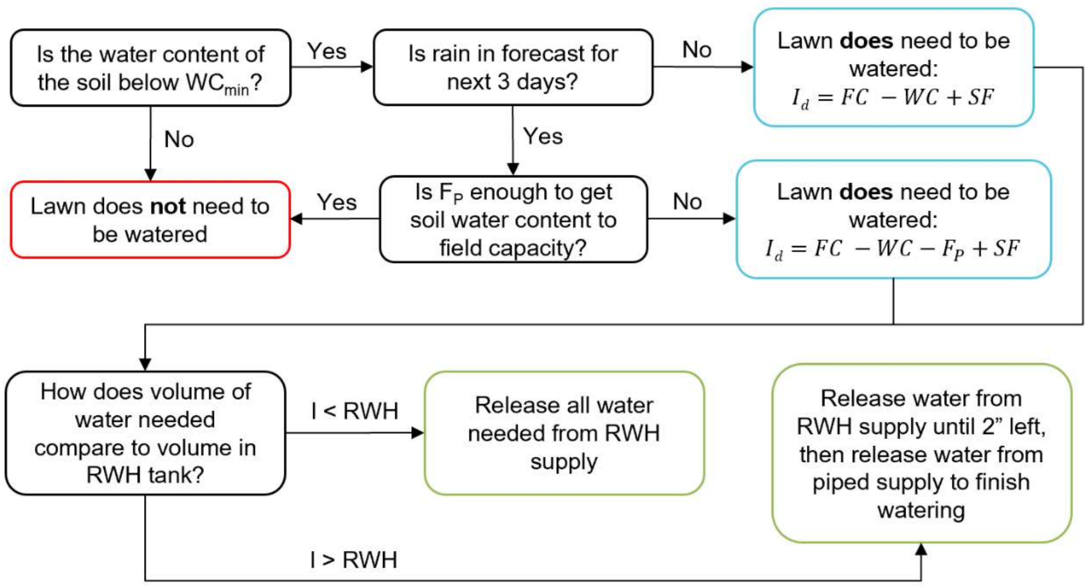


Figure 4.4: A flowchart outlining the main logic steps in the water-release algorithm design. In the equations, I_d is the depth of water to be released for irrigation, I is the volume of water to be released for irrigation, FC is the field capacity of the soil, WC is the soil water content, F_P is the water infiltrated into the soil from projected rainfall, and SF is a safety factor to account for the fact that not all irrigated water will infiltrate into the soil.

4.2.2 Component 2: Arduino-based Electrical System

The electrical system component of this project has two main functions: 1) obtain weather data from an API, and 2) send digital signals to control the valves on the two water sources (RWH and piped water). One key constraint to the electrical system arises from the first function: an Arduino board with Wi-Fi connection is required, such that obtaining weather data from the OpenWeather API is possible. To satisfy this constraint, the Arduino MKR WIFI 1010 was selected as the main component of the electrical system.

To satisfy the second main function, the system outputs a high or low logic signal on two of the Arduino's digital output pins to control whether the solenoid valves are open or closed. Upon integration of this electrical system with actual solenoid valves, additional components such as op-amps would likely be required, since the Arduino's supply voltage alone may not be sufficient, depending on the solenoid valves selected. Since the mechanical system is outside the scope of this project, these high/low signal outputs from the digital pins are a sufficient indicator. Figure 4.5 shows a rendering of the electrical system diagram, in which the connection to the solenoid valves ignores any additional components needed and is thus a conceptual design for visualization purposes only.

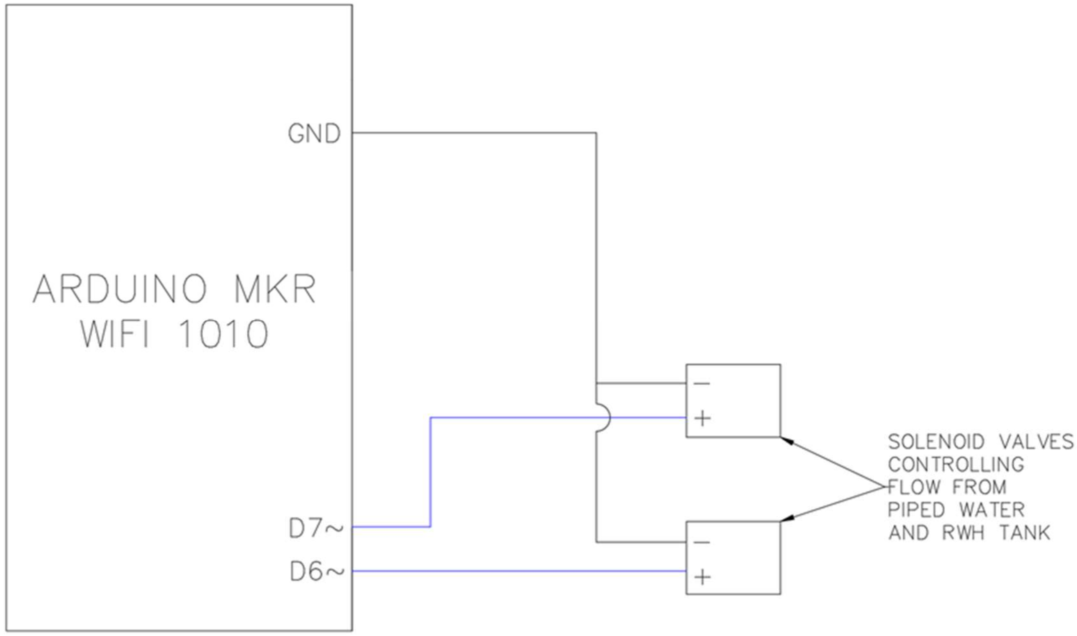


Figure 4.5: Diagram of the electrical system component design. The connection to the solenoid valves is a conceptual design only, as additional components may be needed to amplify the signal from the Arduino, depending on the selected solenoid valves.

5 Testing and Evaluation

5.1 Simulations

5.1.1 Setup Details

To evaluate the success of this project, testing of the automatic water-release algorithm was necessary. The algorithm could not be tested in real-time due to time constraints – that is, it would take at least a full watering season (April – October) to acquire meaningful data in real-time. Instead, testing of the algorithm involved running simulations using previous years' weather data. Simulations were run with multiple scenarios for each of the past ten years (2013-2022), varying the lawn size, RWH tank size, and location inputs. Small, medium, and large lawn and tank size inputs were tested, as indicated in Table 5.1, and the following locations were tested to encompass different areas of Massachusetts:

- Boston, MA
- Worcester, MA
- Plymouth, MA
- Salem, MA

For each simulation, the following initial conditions were set for the first day of the watering season (April 1st):

- The RWH tank was assumed to be full.
- The soil water content of the lawn was assumed to be at field capacity.

Table 5.1: Lawn Size and RWH Tank Size Inputs for Simulations for Testing

Size	**Lawn, ft^2 (m^2)**	**RWH Tank, gal (L)**
Small	2,000 (232)	250 (946)
Medium	10,000 (929)	1,000 (3,785)
Large	20,000 (1,858)	5,000 (18,927)

Thus, there are nine scenarios for a given year and location corresponding to the various combinations of lawn and tank size, as shown in Table 5.2. For each simulation run over the course of a watering season, the total amount of water released from the RWH tank supply, and the total water released from the piped, treated water supply were tracked. From these totals, values for the technical specifications outlined in Section 3 were calculated for each simulation as follows:

- The average total water released for irrigation per week was calculated by summing the totals from the two water sources and dividing by the number of weeks in the watering season (30.57 weeks from April 1st to October 31st).

- The RWH supply's share of the total water released was calculated by dividing the total water released from the RWH tank by the sum of the total water released from each water source.
- The total outdoor water use from the piped water supply is simply the tracked total from the simulation.

The raw data of the totals and the processed data, including the calculated specification values, for each simulation can be found in Appendix A.

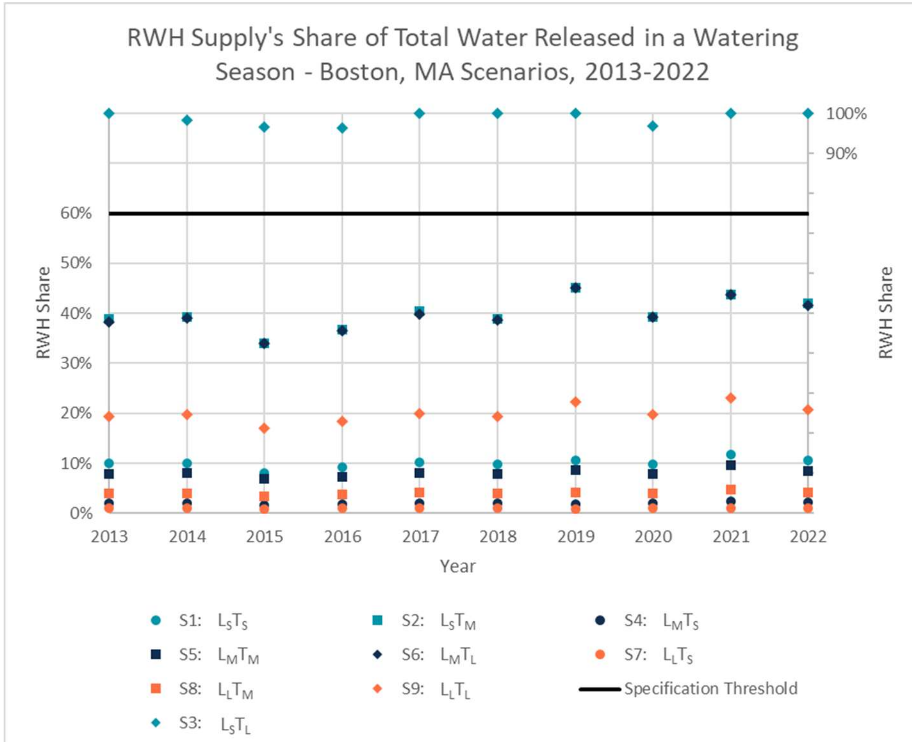
Simulations for the year 2022 were tested with the APIs in the Arduino-based code, found in Appendix C, to prove that the Arduino-based system was able to run successfully. The remaining simulations were run in Python, with the code in Appendix D, using downloaded bulk history weather data from OpenWeatherMap, as the historical weather API is only able to access data from the previous year.

Table 5.2: Nine Scenarios for the Simulations for a Given Year and Location Based on Lawn and RWH Tank Size Variations

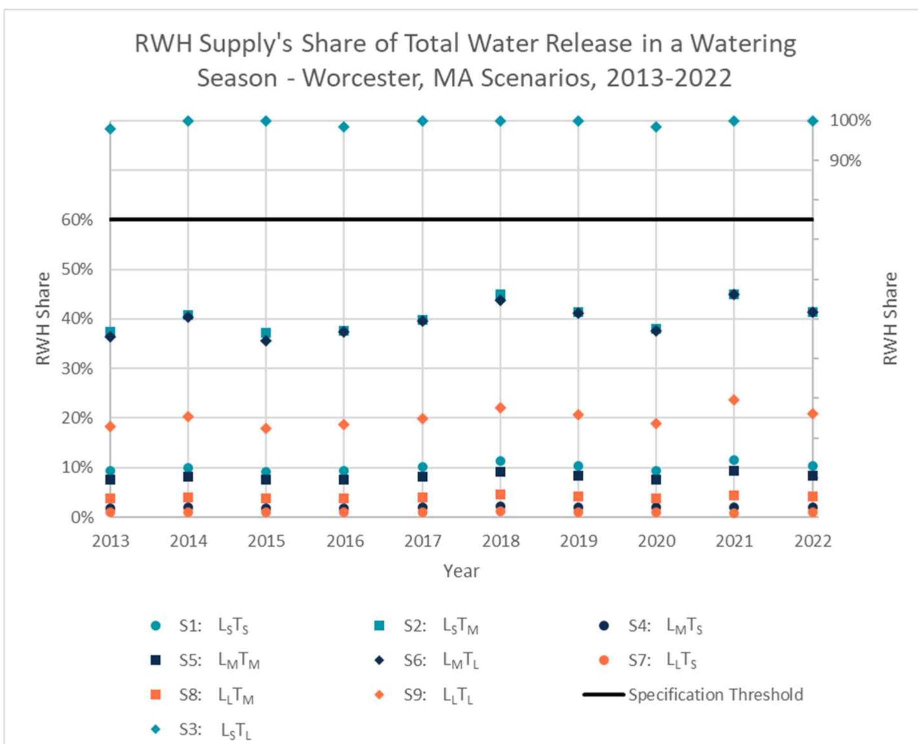
Year X, Location Y		
Scenario	**Lawn Size**	**RWH Tank Size**
1	Small	Small
2	Small	Medium
3	Small	Large
4	Medium	Small
5	Medium	Medium
6	Medium	Large
7	Large	Small
8	Large	Medium
9	Large	Large

5.1.2 Results

For each of the four locations tested, a scatter plot was compiled for each of the three technical specifications, displaying the results for the nine scenarios across the ten simulated years. Figure 5.1 displays the results for the second technical specification – the RWH supply's share of the total water released – for both the Boston location and the Worcester location. The results vary slightly across different locations, as seen in Figure 5.1, but the largest variations are a result of the differing lawn and tank sizes. As such, this section presents results mainly for the Boston location, and a complete set of scatter plots for each location can be found in Appendix B.



(a)



(b)

Figure 5.1: Simulations results for each of the nine scenarios for the RWH supply's share of the total water released in a watering season for both (a) the Boston location and (b) the Worcester location. Scenarios with the same lawn size have the same icon color, and scenarios with the same tank size have the same icon shape.

There was also considerable variation across the nine scenarios in the total outdoor water use from the piped water supply in a watering season, as shown in Figure 5.2. In general, Scenario 3, which used the small lawn size and large tank size, performed the best (largest RWH supply share and smallest piped water use), while Scenario 7, which used the large lawn size and small tank size, performed the worst (smallest RWH supply share and greatest piped water use). As expected, larger lawn sizes generally resulted in a smaller RWH share and greater piped water use, and larger tank sizes generally resulted in a larger RWH share and less piped water use. Additionally, lawn size had a greater impact on the specification results and the overall water use for each simulation than tank size.

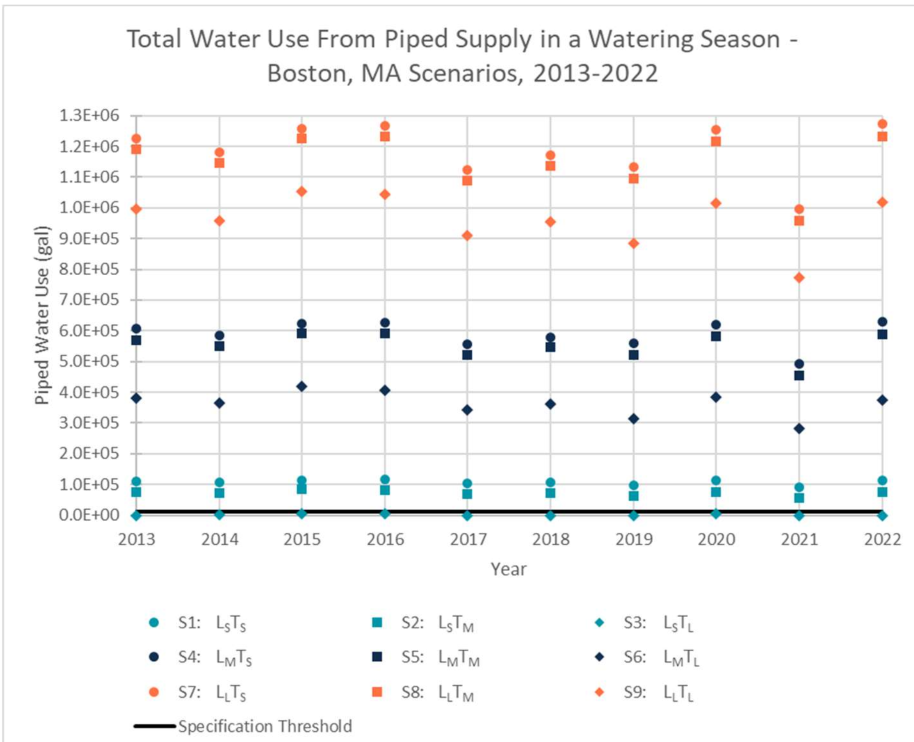


Figure 5.2: Simulations results for each of the nine scenarios for the total outdoor water use from the piped water supply in a watering season for Boston, MA.

There was considerably less variation between scenarios in the average depth of water released for irrigation per week, as displayed in Figure 5.3. This tighter spread stems from the fact that water depth normalizes for lawn size, and thus variations due to lawn size are eliminated. Across the ten simulated years, the average irrigation per week for Boston fell within the range of 2.6 to 3.4 inches (66.1 to 86.4 mm).

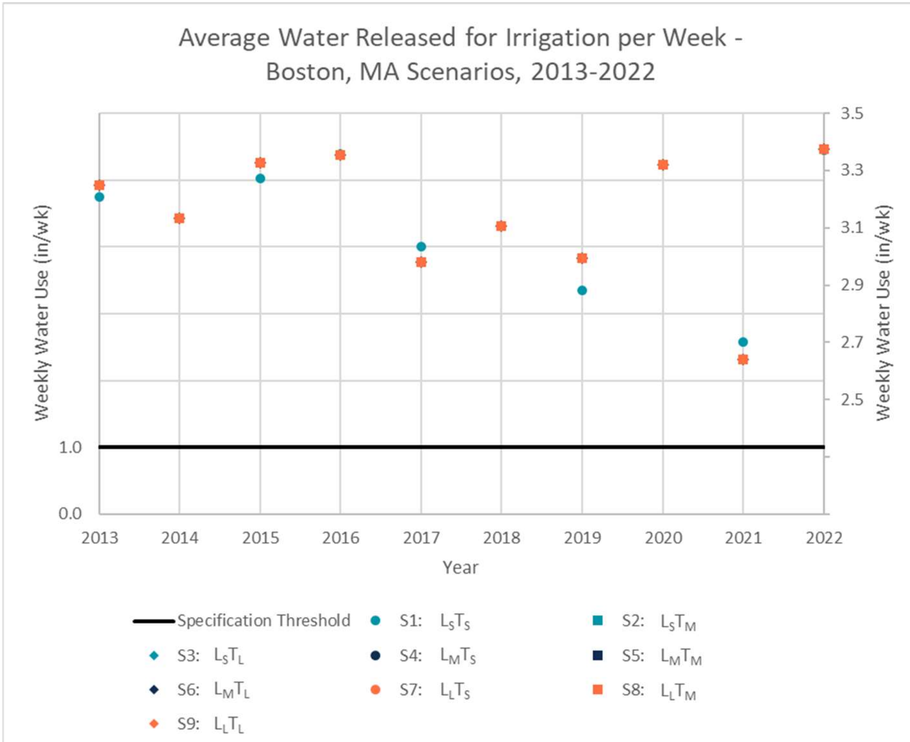


Figure 5.3: Simulations results for each of the nine scenarios for the average depth of water released for irrigation per week for Boston, MA.

In the scatter plots for each of the three technical specifications, slight variations are observed across the simulated years due to differences in weather conditions, including rainfall, temperature, and humidity, among other parameters. As such, considering the time-averaged results for each scenario allows for an easier comparison between the scenarios. The average values across the ten simulated years for each technical specification and each scenario are shown in Table 5.3.

Table 5.3: Technical Specification Results for Each Scenario Averaged Over 2013 – 2022 in Boston, MA

Scenario	[Spec 1] Avg. Water Per Week (in [mm])	[Spec 2] RWH Supply Share (%)	[Spec 3] Outdoor Water Use - Piped (gal [L])
1: L_ST_S	3.14 [79.76]	10%	107,713 [407,693]
2: L_ST_M	3.15 [80.01]	40%	72,361 [273,886]
3: L_ST_L	3.15 [80.01]	99%	1,474 [5,579]
4: L_MT_S	3.15 [80.01]	2%	588,046 [2,225,755]
5: L_MT_M	3.15 [80.01]	8%	551,800 [2,088,565]
6: L_MT_L	3.15 [80.01]	40%	363,155 [1,374,543]
7: L_LT_S	3.15 [80.01]	1%	1,188,192 [4,497,308]
8: L_LT_M	3.15 [80.01]	4%	1,151,892 [4,359,912]
9: L_LT_L	3.15 [80.01]	20%	961,110 [3,637,800]

With these averages, the lack of variation between scenarios in the average water depth per week is even more apparent. Additionally, it's clear to see that if lawn size is fixed, increasing tank size results in greater RWH share and a lower volume of water from the piped supply. For example, Scenario 3 has a greater RWH share and a lower piped water use than Scenarios 1 and 2.

5.2 Electrical Output Testing

In addition to simulations, testing was required to determine whether the Arduino-based system could successfully output a high or low logic signal in response to the water-release algorithm. In a deployable system, these outputs would come from two of the Arduino's digital output pins to control solenoid valves, as described in Section 4.2.2. For testing purposes, the built-in RGB LED on the MKR WIFI 1010 was used to display this output. During simulations, the Arduino turned the LED on green to indicate water being released from the RWH tank supply and turned the LED on red to indicate water being released from the piped water supply. Otherwise, the Arduino held the LED off. Turning the LED on and off involves sending a high or low logic signal to the red, blue, and green digital pins on the Arduino, and thus this method of testing accurately shows the system's ability to control solenoid valves with the same high/low logic. This testing is more qualitative than quantitative. For all of the simulations, the LED output worked as expected, and thus the electrical output functionality of the system was deemed successful.

5.3 Evaluation of Technical Specifications

The following subsections discuss whether the project has met each of the five technical specifications described in Section 3 based on the simulation results presented in Section 5.1.2 and Appendices A and B.

5.3.1 Average Water Released for Irrigation Per Week

Figure 5.3 and Table 5.3 show that all scenarios had a greater average irrigation depth per week for the ten simulated years in the Boston location, and this result was consistent across the other three locations, too. Thus, this project's water-release algorithm fails to meet the technical specification of having an average of less than 1" (25.4 mm) of irrigated water per week over the course of a watering season.

Failure to meet this specification may indicate that the water-release algorithm is flawed; however, although the algorithm could be developed further to add complexity and accuracy, the algorithm as-is is based on reasonable assumptions and irrigation methods from reputable sources. Thus, it is likely that this failure instead highlights how water-intensive of a process lawn watering is in general.

5.3.2 Rainwater Harvesting Supply's Share of Total Water Released

Figure 5.1 and Table 5.3 indicate that only Scenario 3 had greater than a 60% RWH share over the course of a watering season for the ten simulated years for Boston and Worcester, and this was also the case for Salem and Plymouth. As such, the algorithm largely fails to meet this technical specification.

Failure to meet this technical specification indicates a general limitation of household RWH as it relates to the given lawn and tank size at a household. The success of Scenario 3 shows that using the largest tank size (5000 gal) allowed for nearly 100% of irrigation from RWH use for the smallest lawn size (2000 ft^2). Thus, reaching even the 60% threshold from this specification with a larger lawn size would require even larger tank sizes. These larger tanks would likely necessitate an underground storage system, which isn't feasible to have at most homes due to spatial and financial constraints. As such, failing to meet this specification is not the fault of the designed system, but rather represents a global constraint on the overarching problem.

5.3.3 Total Outdoor Water Use from Piped Water Supply Per Watering Season

Figure 5.2 and Table 5.3 show that only Scenario 3's total piped water use for irrigation was below the threshold set by this specification for Boston, and this result was consistent for Plymouth, Salem, and Worcester. Thus, the algorithm largely fails to meet this technical specification.

Similar to the first specification, failure to meet this specification highlights lawn watering as a very water-intensive process.

5.3.4 Rainwater Harvesting Tank Size Compatibility

Testing of the water-release algorithm with the Arduino-based electrical system indicated that the system could handle RWH tank size inputs throughout the range documented in Section 3, and thus the system successfully meets this technical specification.

5.3.5 Lawn Size Compatibility

Testing of the water-release algorithm with the Arduino-based electrical system indicated that the system could handle lawn size inputs throughout the range documented in Section 3, and thus the system successfully meets this technical specification.

6 Budget

Table 6.1: Itemized Budget

Item	Example Source	Cost
Arduino MKR WIFI 1010	Active Learning Labs	$0
Historical Weather Data & Weather Forecast APIs	Open Weather Map – Free Student Package	$0
Bulk History Weather Data	Open Weather Map	4 locations x $10 per location = $40
TOTAL		**$40**

7 Conclusions

7.1 Impact

Despite failing to meet three of the technical specifications, the system designed in this project is a valuable starting point for making irrigation technology more efficient and up to date. Irrigation is one area where outdated practices and methods are still used, as water is relatively cheap and thus increasing efficiency of irrigation has not been a major focus. However, increased efficiency of irrigation technology has the potential to provide homeowners with significant cost savings, especially if updated technology is paired with rainwater harvesting.

Table 7.1 shows the potential cost savings for each of the nine simulated scenarios for the Boston location based on the results averaged over 2013 to 2022 and assuming a cost of $10 per 1000 gallons of water [41]. The cost savings are calculated from the product of the cost of water per gallon and the water released from the household's RWH tank supply during a watering season. Even the worst-performing scenario, Scenario 7, managed to save an average of 11,511 gal (43,568 L) with the RWH water supply, resulting in over $100 in cost savings, which is not a negligible amount of money.

Table 7.1: Potential Cost Savings for Each Scenario Using the Average RWH Water Use for the Ten Simulated Years for the Boston Location, Assuming a $10 / 1000 gal Cost of Water [41]

Scenario	Average Total RWH Water Use (gal [L])	Potential Cost Savings
1: L_ST_S	11,943 [45205]	$119.43
2: L_ST_M	47,876 [181210]	$478.76
3: L_ST_L	124,586 [471557]	$1,245.86
4: L_MT_S	11,838 [44808]	$118.38
5: L_MT_M	48,286 [182,764]	$482.86
6: L_MT_L	238,042 [900,988]	$2,380.42
7: L_LT_S	11,511 [43,568]	$115.11
8: L_LT_M	48,013 [181,730]	$480.13
9: L_LT_L	239,855 [907,850]	$2,398.55

7.2 Future Work

Although the algorithm developed in this project is a valuable start towards increasing the efficiency of irrigation technology, there is considerable room for improvement and added complexity. One potential point of improvement relates to the method used for calculating infiltration from irrigated water. The NRCS method for infiltration was used, as described in Section 4.2.1.4.1. However, this method was developed for calculating infiltration from precipitation, and thus future work could determine a more accurate method for infiltration calculations from irrigated water specifically.

Additionally, future work could be dedicated to adding complexity to the water-release algorithm. For one, potentially introducing soil moisture sensors to work in parallel with the method for calculating the estimated soil water content or including a network of sensors throughout a household's lawn, especially for larger sized lawns. This network of sensors could be used to inform lawn irrigation in different sectors of the lawn, as some parts of the lawn may dry more quickly than others. Also, complexity could be added by considering how the water needs of grass change over the course of the watering season, perhaps by introducing a dynamically changing water content threshold instead of maintaining a static threshold throughout the year.

Finally, future work could include testing this water-release algorithm in different locations outside Massachusetts, perhaps in places with more yearly rainfall such as the southern United States. This could yield better results, as more precipitation has the potential to offset more irrigation, leading to less water use.

Once the water-release algorithm is in satisfactory form, the next major step would be to design and build the mechanical valve and piping system that would allow for the water-release algorithm to be deployed in actual households. This would result in the most meaningful testing and evaluation, as quantitative results could be coupled with qualitative observations (e.g., is the lawn healthy, is there any pooling of water, etc.). Ultimately, this project represents a small, yet meaningful, piece of the difficult and complex problem of bringing higher efficiency to irrigation technology with the goal of being able to modify end-user behavior.

References

[1] P.A. Arias, et. al., "2021: Technical Summary," *Climate Change 2021: The Physical Science Basis. Contribution of Working Group I to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change*, Cambridge University Press, Cambridge, United Kingdom and New York, NY, USA, pp. 33-144, doi:10.1017/9781009157896.002. [Online]. Available: https://www.ipcc.ch/report/ar6/wg1/downloads/report/IPCC_AR6_WGI_TS.pdf. [Accessed: 08-Sep-2022]

[2] R. Ranasinghe, et. al., "2021: Climate Change Information for Regional Impact and for Risk Assessment," *Climate Change 2021: The Physical Science Basis. Contribution of Working Group I to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change*, Cambridge University Press, Cambridge, United Kingdom and New York, NY, USA, pp. 1767-1926, doi:10.1017/9781009157896.014. [Online]. Available: https://www.ipcc.ch/report/ar6/wg1/downloads/report/IPCC_AR6_WGI_Chapter12.pdf. [Accessed: 08-Sep-2022]

[3] S.I. Seneviratne, et. al., "2021: Weather and Climate Extreme Events in a Changing Climate," *Climate Change 2021: The Physical Science Basis. Contribution of Working Group I to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change*, Cambridge University Press, Cambridge, United Kingdom and New York, NY, USA, pp. 1767-1926, doi:10.1017/9781009157896.013. [Online]. Available: https://www.ipcc.ch/report/ar6/wg1/downloads/report/IPCC_AR6_WGI_Chapter11.pdf. [Accessed: 08-Sep-2022]

[4] "Precipitation changes," *Massachusetts Wildlife Climate Action Tool*. [Online]. Available: https://climateactiontool.org/content/precipitation-changes. [Accessed: 08-Sep-2022]

[5] "Climate impacts in the Northeast," *EPA*, 22-Dec-2016. [Online]. Available: https://19january2017snapshot.epa.gov/climate-impacts/climate-impacts-northeast_.html. [Accessed: 08-Sep-2022]

[6] J. Runkle, K. Kunkel, et. al., "State Climate Summaries 2022 150-MA," *NOAA National Centers for Environmental Information*. [Online]. Available: https://statesummaries.ncics.org/downloads/Massachusetts-StateClimateSummary2022.pdf. [Accessed 08-Sep-2022]

[7] "Outdoor water use restrictions for cities, towns, and golf courses," *Mass.gov*. [Online]. Available: https://www.mass.gov/info-details/outdoor-water-use-restrictions-for-cities-towns-and-golf-courses. [Accessed 08-Sep-2022].

[8] "Green Stormwater Infrastructure Planning & Design Manual," *Philadelphia Water Department*, Jan-2021. [Online]. Available: https://water.phila.gov/pool/files/gsi-planning-and-design-manual.pdf. [Accessed 08-Sep-2022].

[9] J. Fitzgerald and J. Laufer, "Governing green stormwater infrastructure: the Philadelphia experience," *Local Environment*, vol. 22, no. 2, pp. 256-268, Jun 2016. doi: 10.1080/13549839.2016.1191063. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/13549839.2016.1191063?scroll=top&needAccess=true. [Accessed 16-Oct-2022].

[10] System for harvesting rainwater, by J. Taborek. (2014, Nov. 11). U.S. Patent 8 881 756 B1. [Online]. Available: https://patents.google.com/patent/US8881756B1/en. [Accessed 08-Sep-2022].

[11] Automated rainwater collection system controller, by J. Larrison. (2017, Apr. 25). U.S. Patent 9 633 532 B1. [Online]. Available: https://patents.google.com/patent/US9633532B1/en?oq=US+9%2c633%2c532+B1. [Accessed 08-Sep-2022].

[12] Irrigation control system, by J. Townsend. (2005, Feb. 1). U.S. Patent 6 850 819 B1. [Online]. Available: https://patents.google.com/patent/US6850819B1/en?oq=U.S.+Pat.+No.+6%2c850%2c819. [Accessed 08-Sep-2022].

[13] Smart sprinkler system and method, by M Levine and L. Dickens. (2019, Mar. 12). U.S. Patent 10 225 997 B1. [Online]. Available: https://patents.google.com/patent/US10225997B1/en?oq=10225997. [Accessed 08-Sep-2022].

[14] "Watering Tips," *EPA*. [Online]. Available: https://www.epa.gov/watersense/watering-tips. [Accessed 16-Oct-2022].

[15] "Watering Efficiently," *Needham Public Works*. [Online]. Available: https://www.needhamma.gov/3724/Watering-Efficiently. [Accessed 08-Sep-2022].

[16] "How Much to Water Your Lawn," *North Reading, Massachusetts*. [Online]. Available: https://www.northreadingma.gov/water-division/pages/how-much-water-your-lawn. [Accessed 08-Sep-2022].

[17] "Precipitation Data: Average Precipitation Statistics," *Mass.gov*. [Online]. Available: https://www.mass.gov/info-details/precipitation-data. [Accessed 16-Oct-2022].

[18] "Report on 2021 Water Use Trends and Reservoir Status," *Massachusetts Water Resources Authority*, 16-Feb-2022. [Online]. Available:

https://www.mwra.com/monthly/wscac/staffsummaries/2022/021722-trends.pdf. [Accessed 16-Oct-2022].

[19] "Underground Rainwater Tanks," *The Tank Depot*. [Online]. Available: https://www.tank-depot.com/p-4267/underground-rainwater-tanks. [Accessed 16-Oct-2022].

[20] "Residential Land Use In Boston," *City of Boston*, Feb-2004. [Online]. Available: https://www.bostonplans.org/getattachment/62f2b1bf-c8b3-434b-89e4-ea8f5c8508b4. [Accessed 16-October-2022].

[21] "More Than Just a Yard," *Mass.gov*. [Online]. Available: https://www.mass.gov/doc/more-than-just-a-yard-ecological-landscaping-tools-1/download. [Accessed 16-Oct-2022].

[22] "About MWRA," *Massachusetts Water Resources Authority*. [Online]. Available: https://www.mwra.com/02org/html/whatis.htm. [Accessed 16-Oct-2022].

[23] "2021: ACS 1-Year Estimates Subject Tables. S1101 | Households and Families," *United States Census Bureau*. [Online]. Available: https://data.census.gov/cedsci/table?q=Families%20and%20Household%20Characteristics%20in%20massachusetts&g=0500000US25005,25009,25017,25021,25023,25025_310XX00US14460&y=2021&tid=ACSST1Y2021.S1101. [Accessed 16-Oct-2022].

[24] A. Rupiper, et. al., "Untapped potential: leak reduction is the most cost-effective urban water management tool," *Environ. Res. Lett.*, vol. 17, no. 3, 24-Feb-2022. [Online]. Available: https://iopscience.iop.org/article/10.1088/1748-9326/ac54cb. [Accessed 16-Oct-2022].

[25] "Massachusetts Code 248 CMR 10: Uniform State Plumbing Code," *Mass.gov*. [Online]. Available: https://www.mass.gov/doc/248-cmr-1000-uniform-state-plumbing-code-0/download#:~:text=248%20CMR%2010.00%20is%20founded,and%20adequately%20maintained%20plumbing%20systems. [Accessed 16-Oct-2022].

[26] "Basics of irrigation scheduling," *University of Minnesota Extension*. [Online]. Available: https://extension.umn.edu/irrigation/basics-irrigation-scheduling. [Accessed 30-Oct-2022].

[27] "History API," *OpenWeather*. [Online]. Available: https://openweathermap.org/history. [Accessed 21-Nov-2022].

[28] "5 day weather forecast," *OpenWeather*. [Online]. Available: https://openweathermap.org/forecast5. [Accessed 21-Nov-2022].

[29] C. Novak, G. Van Giesen, K. DeBusk, *Designing Rainwater Harvesting Systems: Integrating Rainwater Harvesting into Building Systems*. Hoboken, New Jersey: John Wiley & Sons, Inc., 2014, p. 84.

[30] "Crops," in *National Engineering Handbook, Part 652: Irrigation Guide*, United States Department of Agriculture, 1997, ch. 3. [Online]. Available: https://directives.sc.egov.usda.gov/OpenNonWebContent.aspx?content=17837.wba. [Accessed 30-Oct-2022].

[31] "Web Soil Survey," United States Department of Agriculture. [Online]. Available: https://websoilsurvey.sc.egov.usda.gov/App/WebSoilSurvey.aspx. [Accessed 30-Oct-2022].

[32] "Soils," in *National Engineering Handbook, Part 652: Irrigation Guide*, United States Department of Agriculture, 1997, ch. 2. [Online]. Available: https://directives.sc.egov.usda.gov/OpenNonWebContent.aspx?content=17837.wba. [Accessed 30-Oct-2022].

[33] "Selection of Grasses," *UMass Center for Agriculture, Food, and the Environment*. [Online]. Available: https://ag.umass.edu/turf/fact-sheets/selection-of-grasses. [Accessed 15-Nov-2022].

[34] "Kentucky Bluegrass," *University of Maryland Center for Environmental Science*. [Online]. Available: https://www.umces.edu/sites/default/files/Kentucky-bluegrass-summary.pdf. [Accessed 15-Nov-2022].

[35] V. Novotny, *Water Quality: Diffuse Pollution and Watershed Management*. Hoboken, New Jersey: John Wiley & Sons, Inc., 2003, pp. 162-164.

[36] "Abstraction," *Engenious Systems, Inc.* [Online]. Available: https://www.engenious.com/Reference/Abstraction. [Accessed 12-Feb-2023].

[37] "Water Requirements," in *National Engineering Handbook, Part 652: Irrigation Guide*, United States Department of Agriculture, 1997, ch. 4. [Online]. Available: https://directives.sc.egov.usda.gov/OpenNonWebContent.aspx?content=17837.wba. [Accessed 30-Oct-2022].

[38] "Potential Evapotranspiration for Selected Locations," *Northeast Regional Climate Center*. [Online]. Available: http://www.nrcc.cornell.edu/wxstation/pet/pet.html. [Accessed 15-Nov-2022].

[39] R. Allen, et. al., "FAO Penman-Monteith equation," in *Crop evapotranspiration – Guidelines for computing crop water requirements – FAO Irrigation and drainage paper 56*, Rome, 1998, ch. 2. [Online]. Available: https://www.fao.org/3/x0490e/x0490e06.htm#chapter%202%20%20%20fao%20penman%20monteith%20equation. [Accessed 21-Nov-2022].

[40] R. Allen, et. al., "Meteorological data," in *Crop evapotranspiration – Guidelines for computing crop water requirements – FAO Irrigation and drainage paper 56*, Rome,

1998, ch. 3. [Online]. Available: https://www.fao.org/3/x0490e/x0490e07.htm#chapter%203%20%20%20meteorological%20data. [Accessed 21-Nov-2022].

[41] "Current Water and Sewer Rates," *Boston Water and Sewer Commission*. [Online]. Available: https://www.bwsc.org/residential-customers/rates. [Accessed 29-Mar-2023].

APPENDIX A: RAW AND PROCESSED DATA FROM SIMULATIONS

Table A.1: Raw Data from Boston, MA Simulations

*** ALL VALUES IN GALLONS**	Year	RWH	Piped Water	Days Watered
Scenario 1	2013	12250	109956	56
	2014	12000	107453	54
	2015	10000	114733	54
	2016	11750	116211	54
	2017	11750	103942	51
	2018	11500	106808	51
	2019	11527	98300	53
	2020	12500	113987	58
	2021	12122	90770	55
	2022	13500	114968	55
Scenario 2	2013	48172	75660	57
	2014	46880	72521	54
	2015	43000	83834	55
	2016	47000	80897	54
	2017	46000	67582	50
	2018	46000	72347	51
	2019	51423	62643	57
	2020	49696	76825	58
	2021	43909	56678	54
	2022	54000	74620	55
Scenario 3	2013	123832	0	57
	2014	117499	1902	54
	2015	122408	4426	55
	2016	123390	4508	54
	2017	113582	0	50
	2018	118347	0	51
	2019	114066	0	57
	2020	122618	3903	58
	2021	100587	0	54
	2022	128620	0	55
Scenario 4	2013	12250	606911	57
	2014	12000	585005	54
	2015	10750	623421	55
	2016	11750	627737	54
	2017	11500	556410	50
	2018	11500	580235	51
	2019	10634	559695	57
	2020	12500	620107	58
	2021	11594	491340	54
	2022	13500	629601	55

Scenario 5	2013	49000	570161	57
	2014	48000	549005	54
	2015	43000	591171	55
	2016	47000	592487	54
	2017	46000	521910	50
	2018	46000	545735	51
	2019	49634	520695	57
	2020	50000	582607	58
	2021	47801	455133	54
	2022	54000	589101	55
Scenario 6	2013	237055	382106	57
	2014	232817	364188	54
	2015	215000	419171	55
	2016	233845	405642	54
	2017	226409	341502	50
	2018	228845	362889	51
	2019	257113	313217	57
	2020	248480	384127	58
	2021	219545	283389	54
	2022	267777	375323	55
Scenario 7	2013	12250	1226072	57
	2014	12000	1182009	54
	2015	10750	1257592	55
	2016	11750	1267224	54
	2017	11500	1124321	50
	2018	11500	1171970	51
	2019	8269	1132390	57
	2020	12500	1252713	58
	2021	10938	994930	54
	2022	13500	1272701	55
Scenario 8	2013	49000	1189322	57
	2014	48000	1146009	54
	2015	43000	1225342	55
	2016	47000	1231974	54
	2017	46000	1089821	50
	2018	46000	1137470	51
	2019	47269	1093390	57
	2020	50000	1215213	58
	2021	47688	958180	54
	2022	54000	1232201	55
Scenario 9	2013	240905	997417	57
	2014	236143	957866	54
	2015	215000	1053342	55

	2016	233845	1045129	54
	2017	226409	909412	50
	2018	228845	954624	51
	2019	255269	885390	57
	2020	250000	1015213	58
	2021	231588	774279	54
	2022	267777	1018424	55

Table A.2: Processed Data from Boston, MA Simulations

	Year	Lawn Size (sf)	Total Water Released (gal)	Total Water Released (in)	[Spec 1] Avg. Water Per Week (in)	[Spec 2] RWH Supply Share (%)	[Spec 3] Outdoor Water Use - Piped (gal)
S1: L_ST_S	2013	2000	122206	98	3.2	10%	109956
	2014	2000	119453	96	3.1	10%	107453
	2015	2000	124733	100	3.3	8%	114733
	2016	2000	127961	103	3.4	9%	116211
	2017	2000	115692	93	3.0	10%	103942
	2018	2000	118308	95	3.1	10%	106808
	2019	2000	109827	88	2.9	10%	98300
	2020	2000	126487	101	3.3	10%	113987
	2021	2000	102892	83	2.7	12%	90770
	2022	2000	128468	103	3.4	11%	114968
S2: L_ST_M	2013	2000	123832	99	3.2	39%	75660
	2014	2000	119401	96	3.1	39%	72521
	2015	2000	126834	102	3.3	34%	83834
	2016	2000	127897	103	3.4	37%	80897
	2017	2000	113582	91	3.0	40%	67582
	2018	2000	118347	95	3.1	39%	72347
	2019	2000	114066	91	3.0	45%	62643
	2020	2000	126521	101	3.3	39%	76825
	2021	2000	100587	81	2.6	44%	56678
	2022	2000	128620	103	3.4	42%	74620
S3: L_ST_L	2013	2000	123832	99	3.2	100%	0
	2014	2000	119401	96	3.1	98%	1902
	2015	2000	126834	102	3.3	97%	4426
	2016	2000	127897	103	3.4	96%	4508
	2017	2000	113582	91	3.0	100%	0
	2018	2000	118347	95	3.1	100%	0
	2019	2000	114066	91	3.0	100%	0
	2020	2000	126521	101	3.3	97%	3903
	2021	2000	100587	81	2.6	100%	0

	2022	2000	128620	103	3.4	100%	0
S4: L_MT_S	2013	10000	619161	99	3.2	2%	606911
	2014	10000	597005	96	3.1	2%	585005
	2015	10000	634171	102	3.3	2%	623421
	2016	10000	639487	103	3.4	2%	627737
	2017	10000	567910	91	3.0	2%	556410
	2018	10000	591735	95	3.1	2%	580235
	2019	10000	570330	91	3.0	2%	559695
	2020	10000	632607	101	3.3	2%	620107
	2021	10000	502934	81	2.6	2%	491340
	2022	10000	643101	103	3.4	2%	629601
S5: L_MT_M	2013	10000	619161	99	3.2	8%	570161
	2014	10000	597005	96	3.1	8%	549005
	2015	10000	634171	102	3.3	7%	591171
	2016	10000	639487	103	3.4	7%	592487
	2017	10000	567910	91	3.0	8%	521910
	2018	10000	591735	95	3.1	8%	545735
	2019	10000	570330	91	3.0	9%	520695
	2020	10000	632607	101	3.3	8%	582607
	2021	10000	502934	81	2.6	10%	455133
	2022	10000	643101	103	3.4	8%	589101
S6: L_MT_L	2013	10000	619161	99	3.2	38%	382106
	2014	10000	597005	96	3.1	39%	364188
	2015	10000	634171	102	3.3	34%	419171
	2016	10000	639487	103	3.4	37%	405642
	2017	10000	567910	91	3.0	40%	341502
	2018	10000	591735	95	3.1	39%	362889
	2019	10000	570330	91	3.0	45%	313217
	2020	10000	632607	101	3.3	39%	384127
	2021	10000	502934	81	2.6	44%	283389
	2022	10000	643101	103	3.4	42%	375323
S7: L_LT_S	2013	20000	1238322	99	3.2	1%	1226072
	2014	20000	1194009	96	3.1	1%	1182009
	2015	20000	1268342	102	3.3	1%	1257592
	2016	20000	1278974	103	3.4	1%	1267224
	2017	20000	1135821	91	3.0	1%	1124321
	2018	20000	1183470	95	3.1	1%	1171970
	2019	20000	1140659	91	3.0	1%	1132390
	2020	20000	1265213	101	3.3	1%	1252713
	2021	20000	1005867	81	2.6	1%	994930
	2022	20000	1286201	103	3.4	1%	1272701
S8: L_LT_M	2013	20000	1238322	99	3.2	4%	1189322
	2014	20000	1194009	96	3.1	4%	1146009

	2015	20000	1268342	102	3.3	3%	1225342
	2016	20000	1278974	103	3.4	4%	1231974
	2017	20000	1135821	91	3.0	4%	1089821
	2018	20000	1183470	95	3.1	4%	1137470
	2019	20000	1140659	91	3.0	4%	1093390
	2020	20000	1265213	101	3.3	4%	1215213
	2021	20000	1005867	81	2.6	5%	958180
	2022	20000	1286201	103	3.4	4%	1232201
S9: $L_L T_L$	2013	20000	1238322	99	3.2	19%	997417
	2014	20000	1194009	96	3.1	20%	957866
	2015	20000	1268342	102	3.3	17%	1053342
	2016	20000	1278974	103	3.4	18%	1045129
	2017	20000	1135821	91	3.0	20%	909412
	2018	20000	1183470	95	3.1	19%	954624
	2019	20000	1140659	91	3.0	22%	885390
	2020	20000	1265213	101	3.3	20%	1015213
	2021	20000	1005867	81	2.6	23%	774279
	2022	20000	1286201	103	3.4	21%	1018424

Table A.3: Raw Data from Plymouth, MA Simulations

**** ALL VALUES IN GALLONS***	**Year**	**RWH**	**Piped Water**	**Days Watered**
Scenario 1	2013	10500	95192	46
	2014	11000	101245	51
	2015	9250	86930	43
	2016	10750	103342	49
	2017	11000	96415	47
	2018	11000	100194	48
	2019	11000	91370	46
	2020	10750	99431	48
	2021	10412	80179	45
	2022	11500	94369	49
Scenario 2	2013	42000	62985	45
	2014	42832	67778	50
	2015	38000	58043	42
	2016	43000	70980	49
	2017	44000	63415	47
	2018	44000	67186	48
	2019	45000	58960	47
	2020	42000	66106	47
	2021	39663	50995	45
	2022	42000	63559	48

Scenario 3	2013	104985	0	45
	2014	108059	2551	50
	2015	96043	0	42
	2016	111882	2098	49
	2017	107415	0	47
	2018	109320	1865	48
	2019	103960	0	47
	2020	106118	1988	47
	2021	90658	0	45
	2022	105559	0	48
Scenario 4	2013	10500	514423	45
	2014	10750	542300	50
	2015	9500	470715	42
	2016	10750	559151	49
	2017	11000	526077	47
	2018	11000	544928	48
	2019	11250	508552	47
	2020	10500	530030	47
	2021	10500	442791	45
	2022	10500	517294	48
Scenario 5	2013	42000	482923	45
	2014	43000	510050	50
	2015	38000	442215	42
	2016	43000	526901	49
	2017	44000	493077	47
	2018	44000	511928	48
	2019	45000	474802	47
	2020	42000	498530	47
	2021	41992	411299	45
	2022	42000	485794	48
Scenario 6	2013	206622	318301	45
	2014	210569	342481	50
	2015	182691	297525	42
	2016	212991	356910	49
	2017	218759	318319	47
	2018	212904	343024	48
	2019	220768	299034	47
	2020	208204	332326	47
	2021	198314	254977	45
	2022	210000	317794	48
Scenario 7	2013	10500	1039346	45
	2014	10750	1095351	50
	2015	9500	950931	42

	2016	10750	1129052	49
	2017	11000	1063155	47
	2018	11000	1100856	48
	2019	11250	1028354	47
	2020	10500	1070560	47
	2021	10500	896083	45
	2022	10500	1045088	48
Scenario 8	2013	42000	1007846	45
	2014	43000	1063101	50
	2015	38000	922431	42
	2016	43000	1096802	49
	2017	44000	1030155	47
	2018	44000	1067856	48
	2019	45000	994604	47
	2020	42000	1039060	47
	2021	42000	864583	45
	2022	42000	1013588	48
Scenario 9	2013	206622	843224	45
	2014	211409	894692	50
	2015	182691	777740	42
	2016	212991	926811	49
	2017	218759	855396	47
	2018	212904	898952	48
	2019	220768	818836	47
	2020	208204	872856	47
	2021	201628	704954	45
	2022	210000	845588	48

Table A.4: Processed Data from Plymouth, MA Simulations

	Year	Lawn Size (sf)	Total Water Released (gal)	Total Water Released (in)	[Spec 1] Avg. Water Per Week (in)	[Spec 2] RWH Supply Share (%)	[Spec 3] Outdoor Water Use - Piped (gal)
S1: L_ST_S	2013	2000	105692	85	2.8	10%	95192
	2014	2000	112245	90	2.9	10%	101245
	2015	2000	96180	77	2.5	10%	86930
	2016	2000	114092	92	3.0	9%	103342
	2017	2000	107415	86	2.8	10%	96415
	2018	2000	111194	89	2.9	10%	100194
	2019	2000	102370	82	2.7	11%	91370
	2020	2000	110181	88	2.9	10%	99431
	2021	2000	90591	73	2.4	11%	80179

	2022	2000	105869	85	2.8	11%	94369
S2: L_ST_M	2013	2000	104985	84	2.8	40%	62985
	2014	2000	110610	89	2.9	39%	67778
	2015	2000	96043	77	2.5	40%	58043
	2016	2000	113980	91	3.0	38%	70980
	2017	2000	107415	86	2.8	41%	63415
	2018	2000	111186	89	2.9	40%	67186
	2019	2000	103960	83	2.7	43%	58960
	2020	2000	108106	87	2.8	39%	66106
	2021	2000	90658	73	2.4	44%	50995
	2022	2000	105559	85	2.8	40%	63559
S3: L_ST_L	2013	2000	104985	84	2.8	100%	0
	2014	2000	110610	89	2.9	98%	2551
	2015	2000	96043	77	2.5	100%	0
	2016	2000	113980	91	3.0	98%	2098
	2017	2000	107415	86	2.8	100%	0
	2018	2000	111186	89	2.9	98%	1865
	2019	2000	103960	83	2.7	100%	0
	2020	2000	108106	87	2.8	98%	1988
	2021	2000	90658	73	2.4	100%	0
	2022	2000	105559	85	2.8	100%	0
S4: L_MT_S	2013	10000	524923	84	2.8	2%	514423
	2014	10000	553050	89	2.9	2%	542300
	2015	10000	480215	77	2.5	2%	470715
	2016	10000	569901	91	3.0	2%	559151
	2017	10000	537077	86	2.8	2%	526077
	2018	10000	555928	89	2.9	2%	544928
	2019	10000	519802	83	2.7	2%	508552
	2020	10000	540530	87	2.8	2%	530030
	2021	10000	453291	73	2.4	2%	442791
	2022	10000	527794	85	2.8	2%	517294
S5: L_MT_M	2013	10000	524923	84	2.8	8%	482923
	2014	10000	553050	89	2.9	8%	510050
	2015	10000	480215	77	2.5	8%	442215
	2016	10000	569901	91	3.0	8%	526901
	2017	10000	537077	86	2.8	8%	493077
	2018	10000	555928	89	2.9	8%	511928
	2019	10000	519802	83	2.7	9%	474802
	2020	10000	540530	87	2.8	8%	498530
	2021	10000	453291	73	2.4	9%	411299
	2022	10000	527794	85	2.8	8%	485794
S6: L_MT_L	2013	10000	524923	84	2.8	39%	318301
	2014	10000	553050	89	2.9	38%	342481

	2015	10000	480215	77	2.5	38%	297525
	2016	10000	569901	91	3.0	37%	356910
	2017	10000	537077	86	2.8	41%	318319
	2018	10000	555928	89	2.9	38%	343024
	2019	10000	519802	83	2.7	42%	299034
	2020	10000	540530	87	2.8	39%	332326
	2021	10000	453291	73	2.4	44%	254977
	2022	10000	527794	85	2.8	40%	317794
S7: L_LT_S	2013	20000	1049846	84	2.8	1%	1039346
	2014	20000	1106101	89	2.9	1%	1095351
	2015	20000	960431	77	2.5	1%	950931
	2016	20000	1139802	91	3.0	1%	1129052
	2017	20000	1074155	86	2.8	1%	1063155
	2018	20000	1111856	89	2.9	1%	1100856
	2019	20000	1039604	83	2.7	1%	1028354
	2020	20000	1081060	87	2.8	1%	1070560
	2021	20000	906583	73	2.4	1%	896083
	2022	20000	1055588	85	2.8	1%	1045088
S8: L_LT_M	2013	20000	1049846	84	2.8	4%	1007846
	2014	20000	1106101	89	2.9	4%	1063101
	2015	20000	960431	77	2.5	4%	922431
	2016	20000	1139802	91	3.0	4%	1096802
	2017	20000	1074155	86	2.8	4%	1030155
	2018	20000	1111856	89	2.9	4%	1067856
	2019	20000	1039604	83	2.7	4%	994604
	2020	20000	1081060	87	2.8	4%	1039060
	2021	20000	906583	73	2.4	5%	864583
	2022	20000	1055588	85	2.8	4%	1013588
S9: L_LT_L	2013	20000	1049846	84	2.8	20%	843224
	2014	20000	1106101	89	2.9	19%	894692
	2015	20000	960431	77	2.5	19%	777740
	2016	20000	1139802	91	3.0	19%	926811
	2017	20000	1074155	86	2.8	20%	855396
	2018	20000	1111856	89	2.9	19%	898952
	2019	20000	1039604	83	2.7	21%	818836
	2020	20000	1081060	87	2.8	19%	872856
	2021	20000	906583	73	2.4	22%	704954
	2022	20000	1055588	85	2.8	20%	845588

Table A.5: Raw Data from Salem, MA Simulations

*ALL VALUES IN GALLONS	Year	RWH	Piped Water	Days Watered
Scenario 1	2013	11250	104939	52
	2014	11750	107636	54
	2015	10000	88864	44
	2016	11250	110734	53
	2017	11500	104201	51
	2018	11500	104798	52
	2019	12750	100019	53
	2020	11750	109945	53
	2021	11500	91101	50
	2022	12500	113989	55
Scenario 2	2013	45248	73028	53
	2014	45658	73925	54
	2015	39792	59528	45
	2016	43000	79588	53
	2017	43000	72688	50
	2018	46000	70789	52
	2019	51970	62307	54
	2020	46000	73997	52
	2021	44504	56417	49
	2022	51000	77771	56
Scenario 3	2013	116067	2209	53
	2014	116946	2637	54
	2015	96784	2537	45
	2016	122140	447	53
	2017	115688	0	50
	2018	116789	0	52
	2019	114276	0	54
	2020	115494	4503	52
	2021	100921	0	49
	2022	128202	569	56
Scenario 4	2013	11500	579883	53
	2014	11750	586163	54
	2015	10000	486603	45
	2016	10750	602188	53
	2017	10750	567690	50
	2018	11500	572444	52
	2019	13000	558381	54
	2020	11500	588487	52
	2021	11250	493354	49
	2022	12750	631106	56

Scenario 5	2013	46000	545383	53
	2014	47000	550913	54
	2015	40000	456603	45
	2016	43000	569938	53
	2017	43000	535440	50
	2018	46000	537944	52
	2019	52000	519381	54
	2020	46000	553987	52
	2021	45000	459604	49
	2022	51000	592856	56
Scenario 6	2013	225300	366083	53
	2014	227349	370564	54
	2015	190839	305764	45
	2016	215000	397938	53
	2017	212991	365450	50
	2018	227691	356254	52
	2019	258053	313328	54
	2020	227564	372424	52
	2021	222391	282212	49
	2022	255000	388856	56
Scenario 7	2013	11500	1171266	53
	2014	11750	1184076	54
	2015	10000	983206	45
	2016	10750	1215126	53
	2017	10750	1146131	50
	2018	11500	1156389	52
	2019	13000	1129762	54
	2020	11500	1188474	52
	2021	11250	997957	49
	2022	12750	1274962	56
Scenario 8	2013	46000	1136766	53
	2014	47000	1148826	54
	2015	40000	953206	45
	2016	43000	1182876	53
	2017	43000	1113881	50
	2018	46000	1121889	52
	2019	52000	1090762	54
	2020	46000	1153974	52
	2021	45000	964207	49
	2022	51000	1236712	56
Scenario 9	2013	229059	953707	53
	2014	230638	965188	54
	2015	191877	801329	45

	2016	215000	1010876	53
	2017	212991	943890	50
	2018	227691	940198	52
	2019	258204	884557	54
	2020	227564	972411	52
	2021	224059	785148	49
	2022	255000	1032712	56

Table A.6: Processed Data from Salem, MA Simulations

	Year	Lawn Size (sf)	Total Water Released (gal)	Total Water Released (in)	[Spec 1] Avg. Water Per Week (in)	[Spec 2] RWH Supply Share (%)	[Spec 3] Outdoor Water Use - Piped (gal)
S1: L_ST_S	2013	2000	116189	93	3.0	10%	104939
	2014	2000	119386	96	3.1	10%	107636
	2015	2000	98864	79	2.6	10%	88864
	2016	2000	121984	98	3.2	9%	110734
	2017	2000	115701	93	3.0	10%	104201
	2018	2000	116298	93	3.1	10%	104798
	2019	2000	112769	90	3.0	11%	100019
	2020	2000	121695	98	3.2	10%	109945
	2021	2000	102601	82	2.7	11%	91101
	2022	2000	126489	101	3.3	10%	113989
S2: L_ST_M	2013	2000	118277	95	3.1	38%	73028
	2014	2000	119583	96	3.1	38%	73925
	2015	2000	99321	80	2.6	40%	59528
	2016	2000	122588	98	3.2	35%	79588
	2017	2000	115688	93	3.0	37%	72688
	2018	2000	116789	94	3.1	39%	70789
	2019	2000	114276	92	3.0	45%	62307
	2020	2000	119997	96	3.1	38%	73997
	2021	2000	100921	81	2.6	44%	56417
	2022	2000	128771	103	3.4	40%	77771
S3: L_ST_L	2013	2000	118277	95	3.1	98%	2209
	2014	2000	119583	96	3.1	98%	2637
	2015	2000	99321	80	2.6	97%	2537
	2016	2000	122588	98	3.2	100%	447
	2017	2000	115688	93	3.0	100%	0
	2018	2000	116789	94	3.1	100%	0
	2019	2000	114276	92	3.0	100%	0
	2020	2000	119997	96	3.1	96%	4503
	2021	2000	100921	81	2.6	100%	0

	2022	2000	128771	103	3.4	100%	569
S4: L_MT_S	2013	10000	591383	95	3.1	2%	579883
	2014	10000	597913	96	3.1	2%	586163
	2015	10000	496603	80	2.6	2%	486603
	2016	10000	612938	98	3.2	2%	602188
	2017	10000	578440	93	3.0	2%	567690
	2018	10000	583944	94	3.1	2%	572444
	2019	10000	571381	92	3.0	2%	558381
	2020	10000	599987	96	3.1	2%	588487
	2021	10000	504604	81	2.6	2%	493354
	2022	10000	643856	103	3.4	2%	631106
S5: L_MT_M	2013	10000	591383	95	3.1	8%	545383
	2014	10000	597913	96	3.1	8%	550913
	2015	10000	496603	80	2.6	8%	456603
	2016	10000	612938	98	3.2	7%	569938
	2017	10000	578440	93	3.0	7%	535440
	2018	10000	583944	94	3.1	8%	537944
	2019	10000	571381	92	3.0	9%	519381
	2020	10000	599987	96	3.1	8%	553987
	2021	10000	504604	81	2.6	9%	459604
	2022	10000	643856	103	3.4	8%	592856
S6: L_MT_L	2013	10000	591383	95	3.1	38%	366083
	2014	10000	597913	96	3.1	38%	370564
	2015	10000	496603	80	2.6	38%	305764
	2016	10000	612938	98	3.2	35%	397938
	2017	10000	578440	93	3.0	37%	365450
	2018	10000	583944	94	3.1	39%	356254
	2019	10000	571381	92	3.0	45%	313328
	2020	10000	599987	96	3.1	38%	372424
	2021	10000	504604	81	2.6	44%	282212
	2022	10000	643856	103	3.4	40%	388856
S7: L_LT_S	2013	20000	1182766	95	3.1	1%	1171266
	2014	20000	1195826	96	3.1	1%	1184076
	2015	20000	993206	80	2.6	1%	983206
	2016	20000	1225876	98	3.2	1%	1215126
	2017	20000	1156881	93	3.0	1%	1146131
	2018	20000	1167889	94	3.1	1%	1156389
	2019	20000	1142762	92	3.0	1%	1129762
	2020	20000	1199974	96	3.1	1%	1188474
	2021	20000	1009207	81	2.6	1%	997957
	2022	20000	1287712	103	3.4	1%	1274962
S8: L_LT_M	2013	20000	1182766	95	3.1	4%	1136766
	2014	20000	1195826	96	3.1	4%	1148826

	2015	20000	993206	80	2.6	4%	953206
	2016	20000	1225876	98	3.2	4%	1182876
	2017	20000	1156881	93	3.0	4%	1113881
	2018	20000	1167889	94	3.1	4%	1121889
	2019	20000	1142762	92	3.0	5%	1090762
	2020	20000	1199974	96	3.1	4%	1153974
	2021	20000	1009207	81	2.6	4%	964207
	2022	20000	1287712	103	3.4	4%	1236712
S9: L_LT_L	2013	20000	1182766	95	3.1	19%	953707
	2014	20000	1195826	96	3.1	19%	965188
	2015	20000	993206	80	2.6	19%	801329
	2016	20000	1225876	98	3.2	18%	1010876
	2017	20000	1156881	93	3.0	18%	943890
	2018	20000	1167889	94	3.1	19%	940198
	2019	20000	1142762	92	3.0	23%	884557
	2020	20000	1199974	96	3.1	19%	972411
	2021	20000	1009207	81	2.6	22%	785148
	2022	20000	1287712	103	3.4	20%	1032712

Table A.7: Raw Data from Worcester, MA Simulations

**** ALL VALUES IN GALLONS***	**Year**	**RWH**	**Piped Water**	**Days Watered**
Scenario 1	2013	11000	106383	52
	2014	12000	108600	54
	2015	12000	118255	58
	2016	12000	115917	56
	2017	12000	106904	53
	2018	12750	100350	54
	2019	11250	97375	52
	2020	11000	106696	53
	2021	11045	84132	52
	2022	12250	105195	53
Scenario 2	2013	44000	73742	52
	2014	50000	72614	55
	2015	47641	80790	57
	2016	47988	79511	56
	2017	47291	71677	53
	2018	51620	63037	55
	2019	45000	63633	52
	2020	45000	73690	54
	2021	40912	50233	48
	2022	49482	70357	54

Scenario 3	2013	115449	2292	52
	2014	122614	0	55
	2015	128200	231	57
	2016	125419	2081	56
	2017	118969	0	53
	2018	114657	0	55
	2019	108529	104	52
	2020	116814	1876	54
	2021	91144	0	48
	2022	119839	0	54
Scenario 4	2013	11000	577708	52
	2014	12500	600571	55
	2015	12000	630151	57
	2016	12000	625499	56
	2017	12000	582843	53
	2018	13000	560286	55
	2019	11250	531914	52
	2020	11250	582200	54
	2021	8824	446898	48
	2022	12500	586694	54
Scenario 5	2013	44000	544708	52
	2014	50000	563071	55
	2015	48000	594151	57
	2016	48000	589499	56
	2017	48000	546843	53
	2018	52000	521286	55
	2019	45000	498164	52
	2020	45000	548450	54
	2021	42574	413148	48
	2022	50000	549194	54
Scenario 6	2013	214313	374395	52
	2014	247564	365508	55
	2015	228370	413780	57
	2016	238360	399139	56
	2017	235089	359754	53
	2018	251346	321940	55
	2019	224059	319105	52
	2020	223418	370032	54
	2021	204558	251165	48
	2022	247408	351786	54
Scenario 7	2013	11000	1166416	52
	2014	12500	1213642	55
	2015	12000	1272301	57

	2016	12000	1262999	56
	2017	12000	1177686	53
	2018	13000	1133572	55
	2019	11250	1075079	52
	2020	11250	1175650	54
	2021	6399	905046	48
	2022	12500	1185889	54
Scenario 8	2013	44000	1133416	52
	2014	50000	1176142	55
	2015	48000	1236301	57
	2016	48000	1226999	56
	2017	48000	1141686	53
	2018	52000	1094572	55
	2019	45000	1041329	52
	2020	45000	1141900	54
	2021	40149	871296	48
	2022	50000	1148389	54
Scenario 9	2013	214313	963103	52
	2014	247564	978579	55
	2015	230167	1054134	57
	2016	238418	1036581	56
	2017	236545	953141	53
	2018	253245	893327	55
	2019	224059	862270	52
	2020	223418	963482	54
	2021	214901	696544	48
	2022	249816	948573	54

Table A.8: Processed Data from Worcester, MA Simulations

	Year	**Lawn Size (sf)**	**Total Water Released (gal)**	**Total Water Released (in)**	**[Spec 1] Avg. Water Per Week (in)**	**[Spec 2] RWH Supply Share (%)**	**[Spec 3] Outdoor Water Use - Piped (gal)**
S1: $L_S T_S$	2013	2000	117383	94	3.1	9%	106383
	2014	2000	120600	97	3.2	10%	108600
	2015	2000	130255	104	3.4	9%	118255
	2016	2000	127917	103	3.4	9%	115917
	2017	2000	118904	95	3.1	10%	106904
	2018	2000	113100	91	3.0	11%	100350
	2019	2000	108625	87	2.9	10%	97375
	2020	2000	117696	94	3.1	9%	106696
	2021	2000	95177	76	2.5	12%	84132

	2022	2000	117445	94	3.1	10%	105195
S2: L_ST_M	2013	2000	117742	94	3.1	37%	73742
	2014	2000	122614	98	3.2	41%	72614
	2015	2000	128430	103	3.4	37%	80790
	2016	2000	127500	102	3.3	38%	79511
	2017	2000	118969	95	3.1	40%	71677
	2018	2000	114657	92	3.0	45%	63037
	2019	2000	108633	87	2.9	41%	63633
	2020	2000	118690	95	3.1	38%	73690
	2021	2000	91144	73	2.4	45%	50233
	2022	2000	119839	96	3.1	41%	70357
S3: L_ST_L	2013	2000	117742	94	3.1	98%	2292
	2014	2000	122614	98	3.2	100%	0
	2015	2000	128430	103	3.4	100%	231
	2016	2000	127500	102	3.3	98%	2081
	2017	2000	118969	95	3.1	100%	0
	2018	2000	114657	92	3.0	100%	0
	2019	2000	108633	87	2.9	100%	104
	2020	2000	118690	95	3.1	98%	1876
	2021	2000	91144	73	2.4	100%	0
	2022	2000	119839	96	3.1	100%	0
S4: L_MT_S	2013	10000	588708	94	3.1	2%	577708
	2014	10000	613071	98	3.2	2%	600571
	2015	10000	642151	103	3.4	2%	630151
	2016	10000	637499	102	3.3	2%	625499
	2017	10000	594843	95	3.1	2%	582843
	2018	10000	573286	92	3.0	2%	560286
	2019	10000	543164	87	2.9	2%	531914
	2020	10000	593450	95	3.1	2%	582200
	2021	10000	455722	73	2.4	2%	446898
	2022	10000	599194	96	3.1	2%	586694
S5: L_MT_M	2013	10000	588708	94	3.1	7%	544708
	2014	10000	613071	98	3.2	8%	563071
	2015	10000	642151	103	3.4	7%	594151
	2016	10000	637499	102	3.3	8%	589499
	2017	10000	594843	95	3.1	8%	546843
	2018	10000	573286	92	3.0	9%	521286
	2019	10000	543164	87	2.9	8%	498164
	2020	10000	593450	95	3.1	8%	548450
	2021	10000	455722	73	2.4	9%	413148
	2022	10000	599194	96	3.1	8%	549194
S6: L_MT_L	2013	10000	588708	94	3.1	36%	374395
	2014	10000	613071	98	3.2	40%	365508

	2015	10000	642151	103	3.4	36%	413780
	2016	10000	637499	102	3.3	37%	399139
	2017	10000	594843	95	3.1	40%	359754
	2018	10000	573286	92	3.0	44%	321940
	2019	10000	543164	87	2.9	41%	319105
	2020	10000	593450	95	3.1	38%	370032
	2021	10000	455722	73	2.4	45%	251165
	2022	10000	599194	96	3.1	41%	351786
S7: L_LT_S	2013	20000	1177416	94	3.1	1%	1166416
	2014	20000	1226142	98	3.2	1%	1213642
	2015	20000	1284301	103	3.4	1%	1272301
	2016	20000	1274999	102	3.3	1%	1262999
	2017	20000	1189686	95	3.1	1%	1177686
	2018	20000	1146572	92	3.0	1%	1133572
	2019	20000	1086329	87	2.9	1%	1075079
	2020	20000	1186900	95	3.1	1%	1175650
	2021	20000	911445	73	2.4	1%	905046
	2022	20000	1198389	96	3.1	1%	1185889
S8: L_LT_M	2013	20000	1177416	94	3.1	4%	1133416
	2014	20000	1226142	98	3.2	4%	1176142
	2015	20000	1284301	103	3.4	4%	1236301
	2016	20000	1274999	102	3.3	4%	1226999
	2017	20000	1189686	95	3.1	4%	1141686
	2018	20000	1146572	92	3.0	5%	1094572
	2019	20000	1086329	87	2.9	4%	1041329
	2020	20000	1186900	95	3.1	4%	1141900
	2021	20000	911445	73	2.4	4%	871296
	2022	20000	1198389	96	3.1	4%	1148389
S9: L_LT_L	2013	20000	1177416	94	3.1	18%	963103
	2014	20000	1226142	98	3.2	20%	978579
	2015	20000	1284301	103	3.4	18%	1054134
	2016	20000	1274999	102	3.3	19%	1036581
	2017	20000	1189686	95	3.1	20%	953141
	2018	20000	1146572	92	3.0	22%	893327
	2019	20000	1086329	87	2.9	21%	862270
	2020	20000	1186900	95	3.1	19%	963482
	2021	20000	911445	73	2.4	24%	696544
	2022	20000	1198389	96	3.1	21%	948573

APPENDIX B: GRAPH OUTPUTS FROM SIMULATIONS FOR EACH TECHNICAL SPECIFICATION AND LOCATION

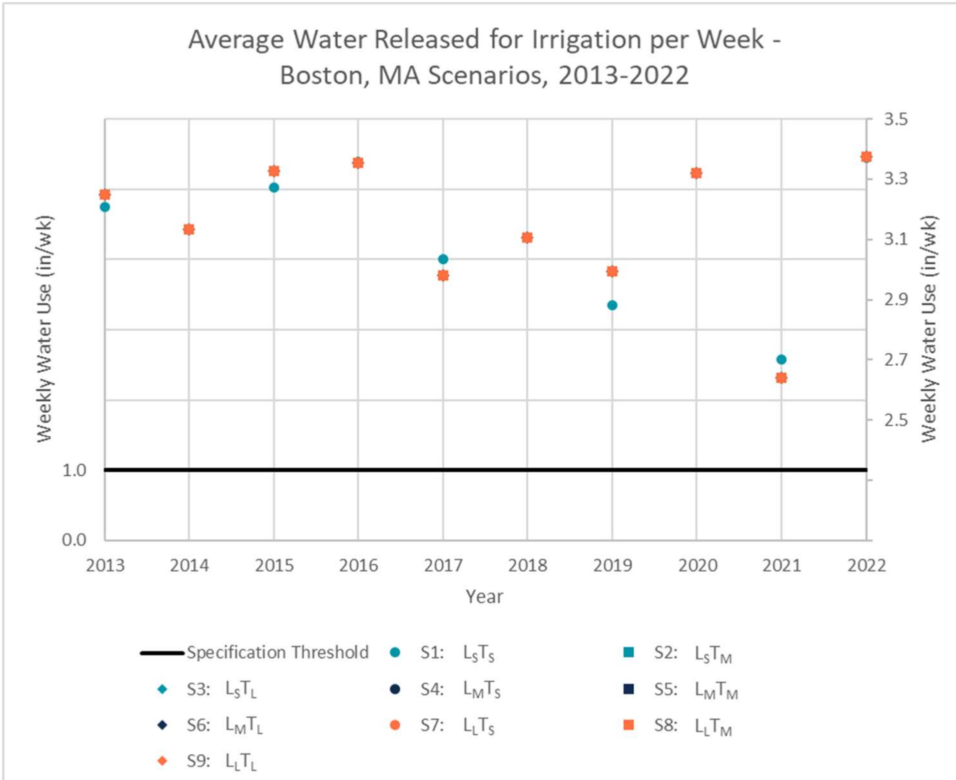


Figure B.1: Average Depth of Water Released for Irrigation per Week for Boston, MA Simulations

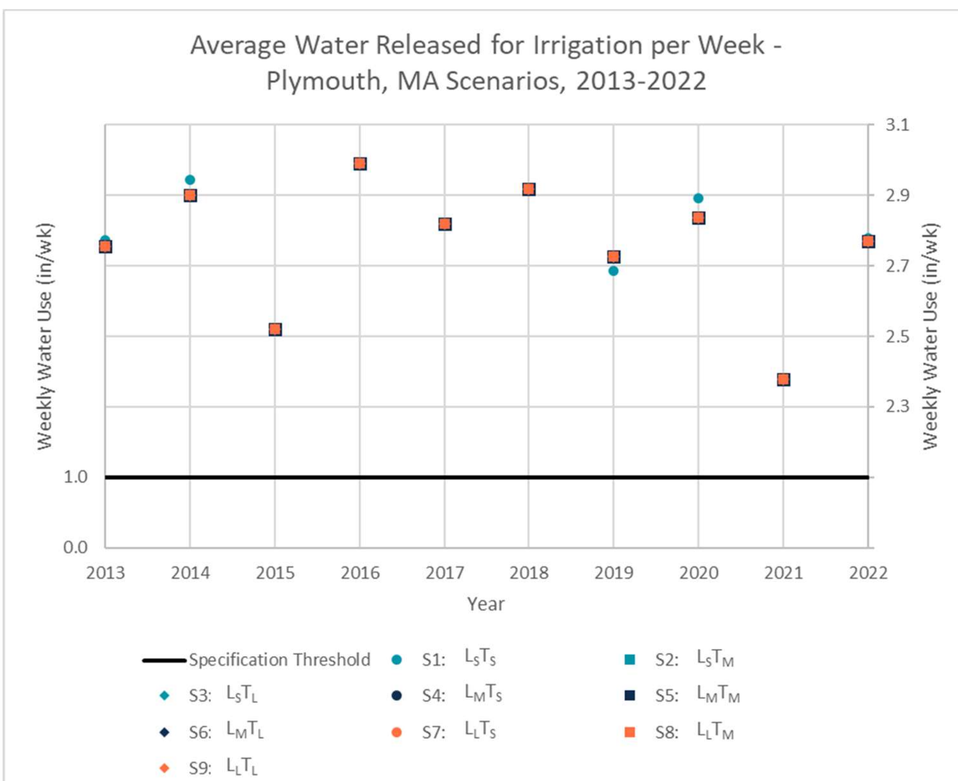


Figure B.2: Average Depth of Water Released for Irrigation per Week for Plymouth, MA Simulations

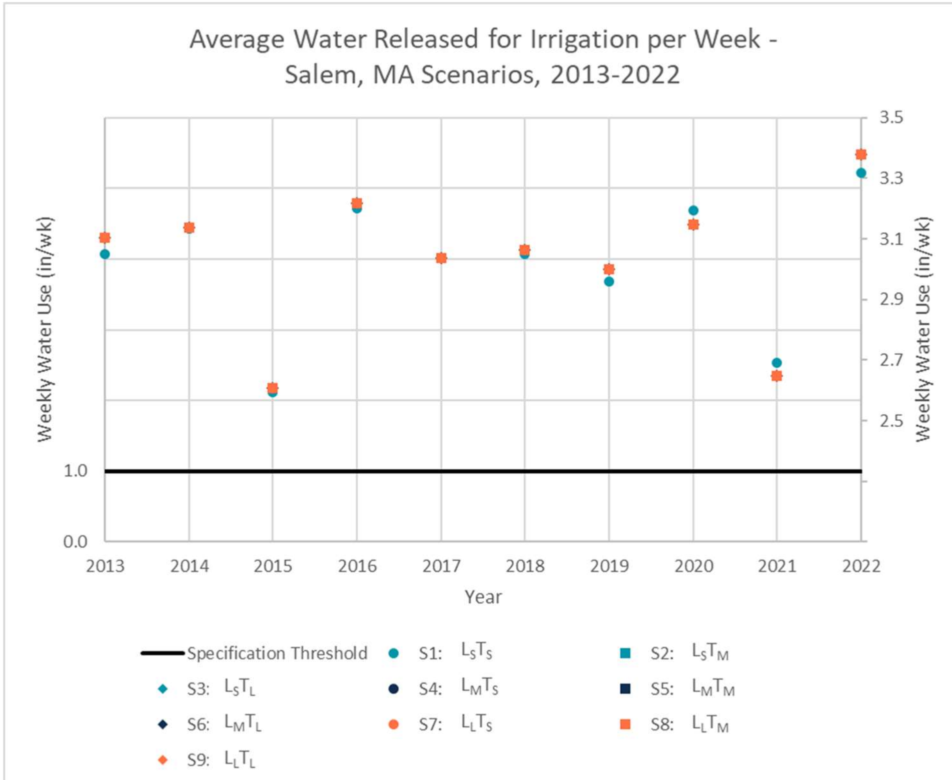


Figure B.3: Average Depth of Water Released for Irrigation per Week for Salem, MA Simulations

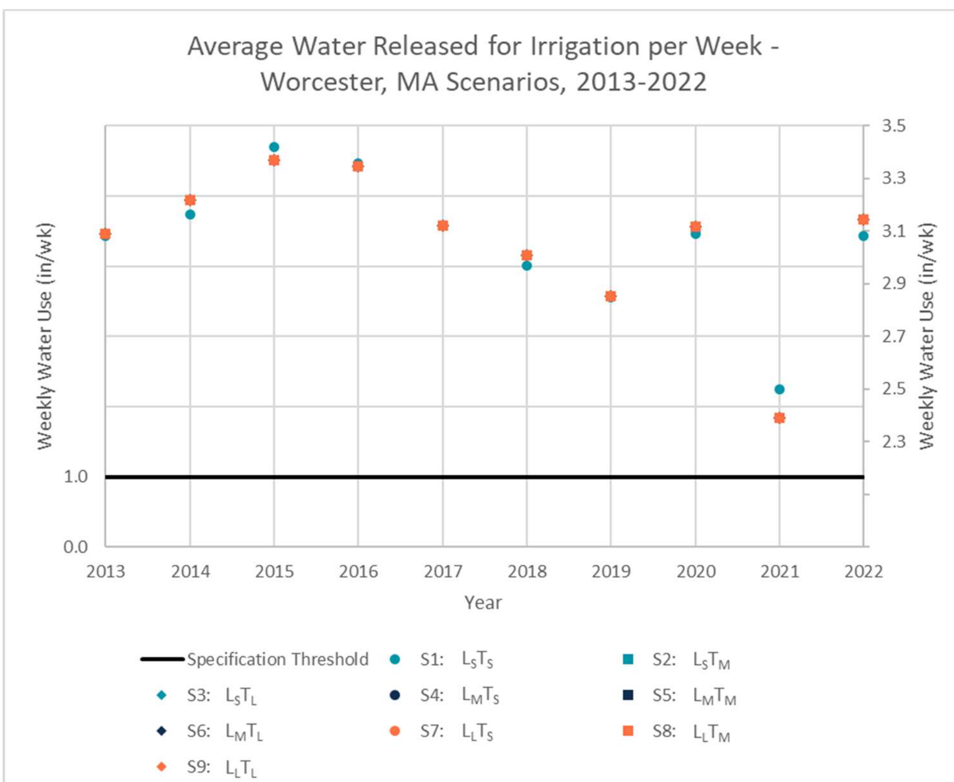


Figure B.4: Average Depth of Water Released for Irrigation per Week for Worcester, MA Simulations

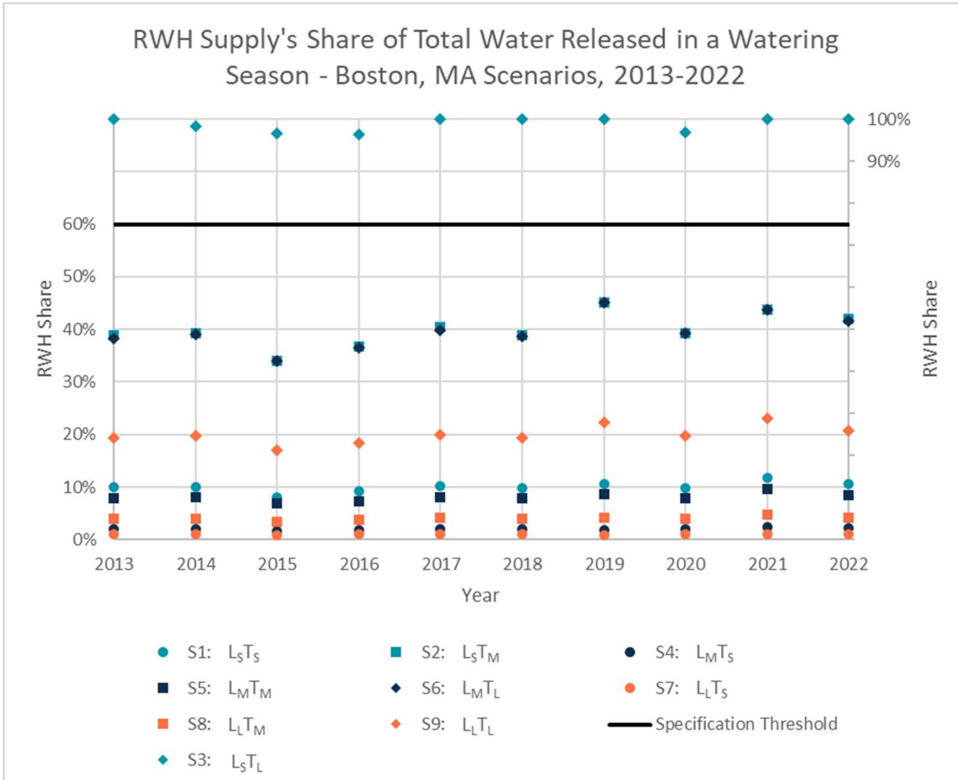


Figure B.5: RWH Supply's Share of Total Water Released for Boston, MA Simulations

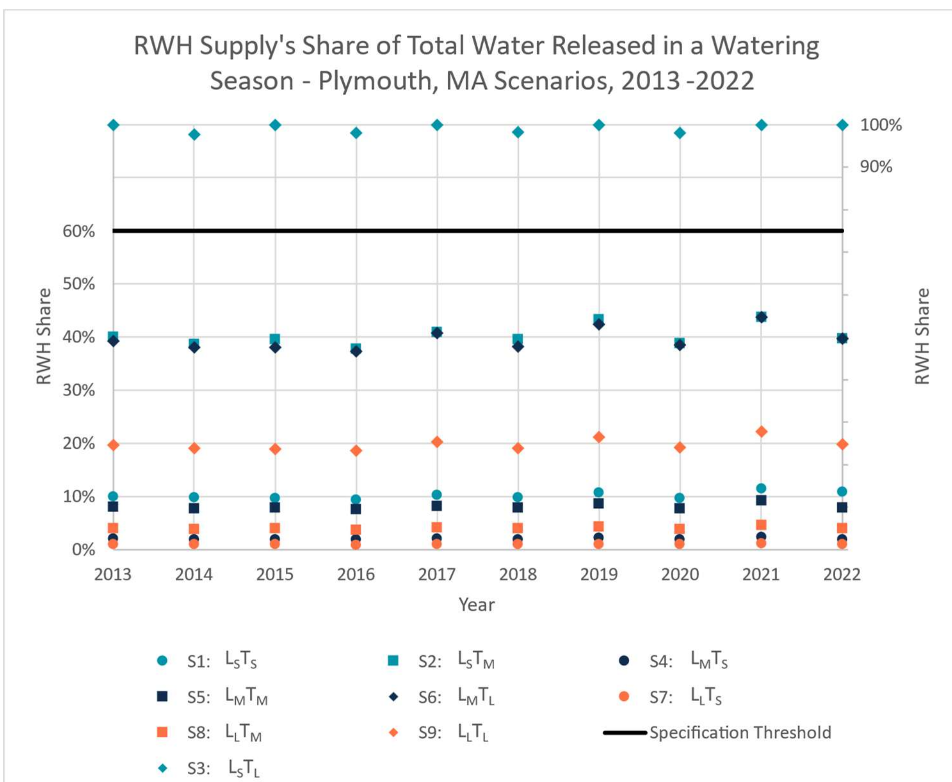


Figure B.6: RWH Supply's Share of Total Water Released for Plymouth, MA Simulations

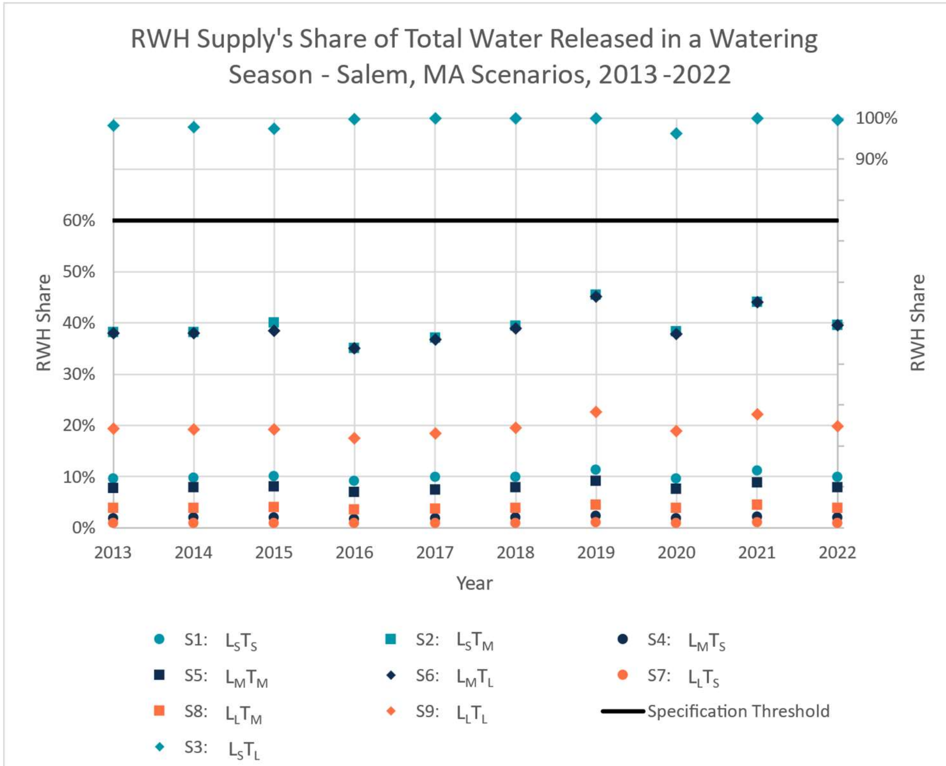


Figure B.7: RWH Supply's Share of Total Water Released for Salem, MA Simulations

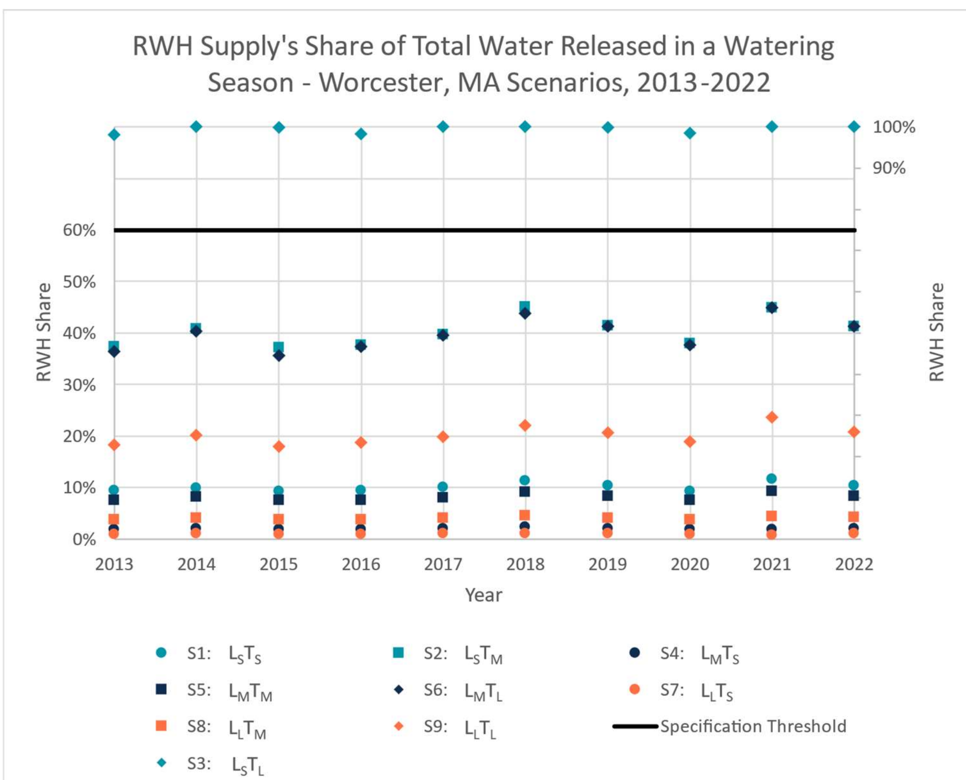


Figure B.8: RWH Supply's Share of Total Water Released for Worcester, MA Simulations

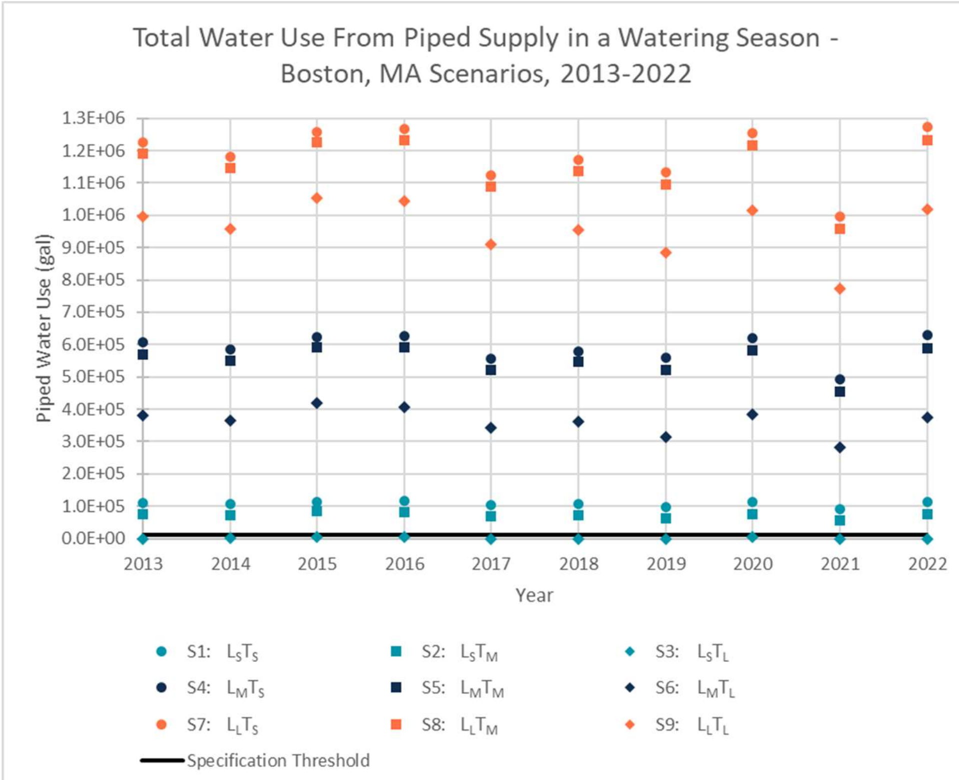


Figure B.9: Total Water Use from Piped Supply for Boston, MA Simulations

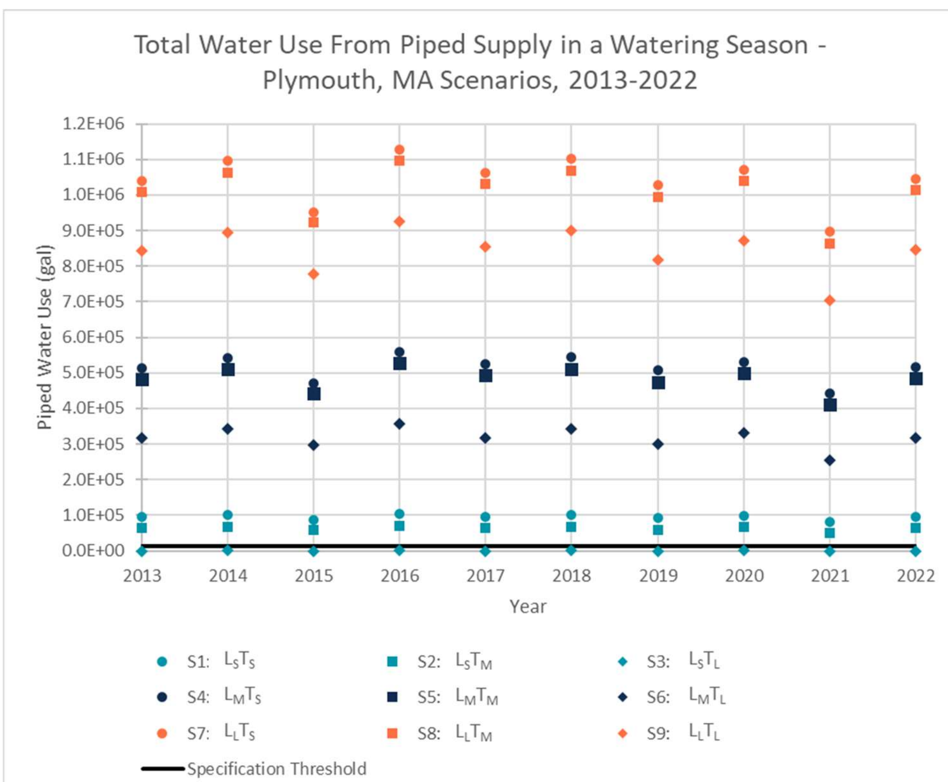


Figure B.10: Total Water Use from Piped Supply for Plymouth, MA Simulations

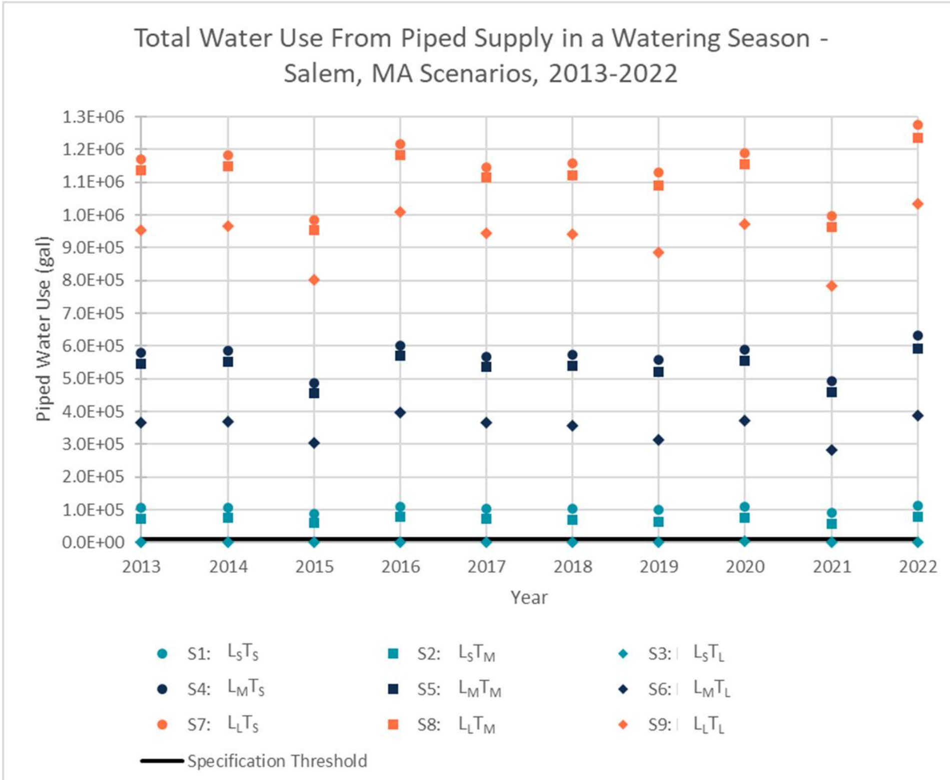


Figure B.11: Total Water Use from Piped Supply for Salem, MA Simulations

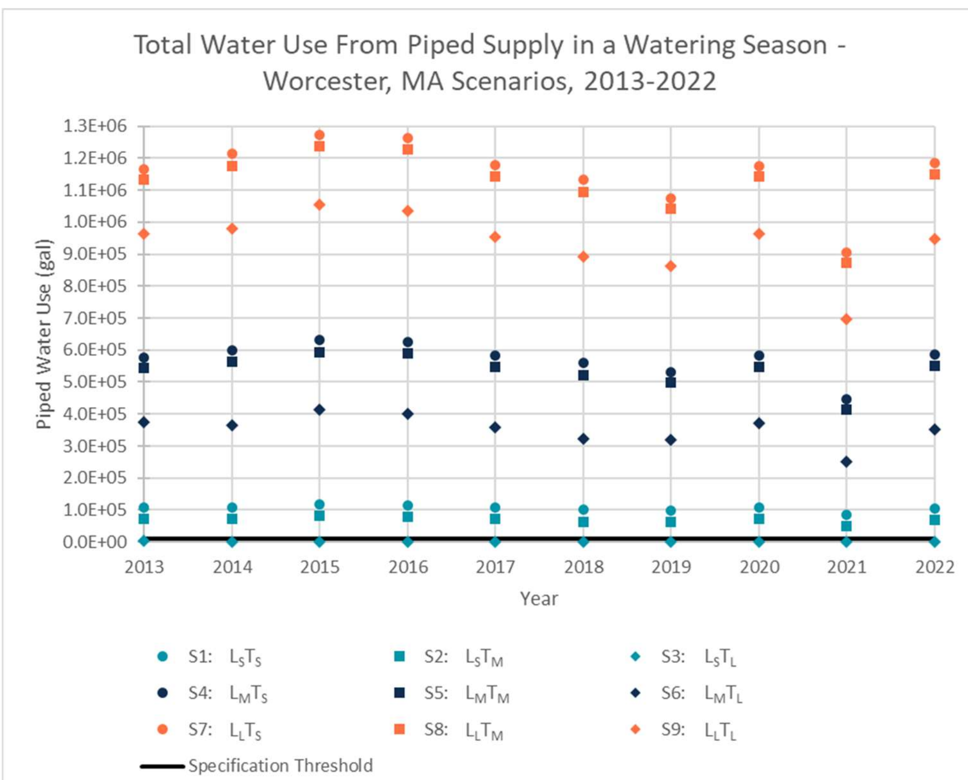


Figure B.12: Total Water Use from Piped Supply for Worcester, MA Simulations

APPENDIX C: ARDUINO CODE FOR WATER-RELEASE ALGORITHM

This Appendix contains the code written in the Arduino IDE for the water-release algorithm developed in this project. The code consists of 6 separate files, and thus the code copied below is organized by file.

Main Arduino file: ES_100.ino

```
#include <WiFiNINA.h>
#include <utility/wifi_drv.h> // needed for accessing LED on board
#include <ArduinoHttpClient.h>
#include <ArduinoJson.h>
#include "arduino_secrets.h"
#include "FAO_P-M_functions.h"
#include "API_dataFunctions.h"
/*
  Sketch generated by the Arduino IoT Cloud Thing "Untitled"
  https://create.arduino.cc/cloud/things/c917b7bf-9c96-450c-b30e-b887b7768bdc

  Arduino IoT Cloud Variables description

  The following variables are automatically generated and updated when changes
are made to the Thing

  - No variables have been created, add cloud variables on the Thing Setup page
    to see them declared here

  Variables which are marked as READ/WRITE in the Cloud Thing will also have
functions
  which are called when their values are changed from the Dashboard.
  These functions are generated with the Thing and added at the end of this
sketch.
*/

#include "thingProperties.h"

/* GLOBAL PARAMETERS */
// Variables Related to Water Content Threshold
float awc = 1.45; // Available Water Capacity, in/ft.
float mad = 0.5; // Management Allowable Depletion, b/t 0 and 1.
float root_depth = 10; // Root Depth of crop, in.
float FC; // field capacity
float WC; // soil water content at a given time [in]
float WC_min; // WC threshold

// Variables for FAO Equation / ET Calcs
float e_s; // See ET Calcs
```

```
float delta; // See ET Calcs
float e_a; // See ET Calcs
float R_n; // See ET Calcs
int day = 91; // Day of Year -- start at April 1st -- 91 on normal year, 92 on
leap year
int last_day = 304; // Last Day of Watering Season, October 31st --  304 on
normal year, 305 on leap year
float lat_deg = 42.360082;
float ET; // See ET Calcs

// Variables for Infiltration Calcs
float CN = 65; // Runoff Curve Number
float S = 25400/CN - 254; // surface storage (mm)
float F_P; // Infiltration from Precipitation -- calculated later
float F_I; // Infiltration from Irrigation -- calculated later
float F_P_predicted; // Expected Infiltration from Forecasted Rain -- calculated
later
float F_I_desired; // Desired infiltration from irrigation -- calculated later
float predicted_rain; // Rainfall in Next 3 Days -- taken from API Call

// Variables for Irrigation Depth & Volume
float I_depth; // irrigation depth for a given day [in]
float I_vol; // irrigation volume for a given day [gal]

// USER INPUTS
float lawnSize = 20000; // Size of Lawn [square feet]
float tankSize = 5000; // Size of Tank [gallons]
float roofArea = 1500; // Area of Roof [square feet]

// Variables for RWH Tank
float waterInTank = tankSize; // Volume of Water in RWH Tank, assume full at
start [gallons]
float C_roof = 0.90; // runoff coefficient for roof

// Variables to Collect Water Use Data During Simulations
float rwhWaterUse = 0;
float pipedWaterUse = 0;
float totalWaterUse = 0;
int waterDays = 0;

// OpenWeatherMap API Key
String apiKey = "3501dd0601d2417040fe846b7bfa9331";

// Initialize String for API url
String url = "";
```

```
// Timestamps for API Calls
int startTimeHistorical = 1648728000; // Historical API Call starts at March
31st, 8am -- this is for the year 2022
int startTimeForecast = startTimeHistorical + 86400; // Forecast API Call starts
at April 1st, 8am -- 24 hours (86400 seconds) after Historical Start Time

// City Codes for OpenWeatherMap API
int cityCode = 4930956; // Boston, MA

// OpenWeatherMap endpoint
const char *host = "history.openweathermap.org";
const int httpPort = 80;

WiFiClient wifiClient;
HttpClient client = HttpClient(wifiClient, host, httpPort);


void setup() {
  // Initialize serial and wait for port to open:
  Serial.begin(9600);
  // This delay gives the chance to wait for a Serial Monitor without blocking if
none is found
  delay(1500);

  // Defined in thingProperties.h
  initProperties();

  // Connect to Arduino IoT Cloud
  ArduinoCloud.begin(ArduinoIoTPreferredConnection, false); // 'false' disables
the Watchdog Timer (WDT)

  /*
     The following function allows you to obtain more information
     related to the state of network and IoT Cloud connection and errors
     the higher number the more granular information you'll get.
     The default is 0 (only errors).
     Maximum is 4
 */
  setDebugMessageLevel(2);
  ArduinoCloud.printDebugInfo();

  /* CALCULATE THE SOIL WATER CONTENT THRESHOLD */
  WC_min = threshold_calc(awc, mad, root_depth); // [inches]
  Serial.print(F("WC Threshold: "));
```

```
  Serial.println(WC_min);

  /* ASSUME SOIL WATER CONTENT IS AT FIELD CAPACITY TO START */
  FC = awc * root_depth/12; // Field Capacity of soil [inches]
  WC = FC; // [inches]
  Serial.print(F("Initial WC: "));
  Serial.println(WC);

  /* SETUP LED ON ARDUINO BOARD */
  WiFiDrv::pinMode(25, OUTPUT); // RED will indicate water being released from
RWH Tank
  WiFiDrv::pinMode(26, OUTPUT); // GREEN will indicate water being released from
Piped Supply
  WiFiDrv::pinMode(27, OUTPUT); // BLUE
  WiFiDrv::digitalWrite(25, LOW); // Initialize Red as OFF
  WiFiDrv::digitalWrite(26, LOW); // Initialize Green as OFF
  WiFiDrv::digitalWrite(27, LOW); // Initialize Blue as OFF
}

void loop() {
  ArduinoCloud.update();
  // Your code here
  /* IF END OF SIMULATION, RETURN */
  if (day > last_day) {
    return;
  }
  Serial.print(F("Beginning of Loop. Day: "));
  Serial.println(day);
  /* BEGINNING OF DAY (8am) */
  /* CALL HISTORICAL API AND COMPILE ALL NECESSARY DATA */
  // Getting 6 hrs at a time from API, so need to call 4 times to get data for
past 24 hrs
  for (int i=0; i<4; i++) {
    // Build the URL for the API call
/data/2.5/weather?q=Boston,us&APPID=3501dd0601d2417040fe846b7bfa9331
    url = String("/data/2.5/history/city?id=") + cityCode + "&type=hour&start=" +
startTimeHistorical + "&cnt=" + cnt + "&appid=" + apiKey;
    // Make API Call and Get JSON doc
    String response = getResponseFromAPI(client, url);
    DynamicJsonDocument doc(3072);
    DeserializationError error = deserializeJson(doc, response);
    if (error) {
      Serial.print(F("Deserialization failed: "));
      Serial.println(error.c_str());
      return;
```

```
    }
    response = "";
    // If it's the first set of 6 hrs, set each data value as normal
    if (i == 0) {
      T_max = get_T_max(doc);
      T_min = get_T_min(doc);
      T_mean = get_T_mean(doc);
      rel_hum_max = get_rel_hum_max(doc);
      rel_hum_min = get_rel_hum_min(doc);
      u_mean = get_wind_mean(doc);
      rain_total = get_rain_sum(doc);
    }
    // If it's the 2nd-4th set of 6 hrs, update the data values as necessary
    else {
      if (get_T_max(doc) > T_max) {
        T_max = get_T_max(doc);
      }
      if (get_T_min(doc) < T_min) {
        T_min = get_T_min(doc);
      }
      T_mean = T_mean*i/(i+1) + get_T_mean(doc)*1/(i+1);
      if (get_rel_hum_max(doc) > rel_hum_max) {
        rel_hum_max = get_rel_hum_max(doc);
      }
      if (get_rel_hum_min(doc) < rel_hum_min) {
        rel_hum_min = get_rel_hum_min(doc);
      }
      u_mean = u_mean*i/(i+1) + get_wind_mean(doc)*1/(i+1);
      rain_total += get_rain_sum(doc);
    }
    // At end of loop, update Start Time for the API Call
    startTimeHistorical += (3600 * cnt);
    Serial.println(F("Historical Start Time Updated"));
  }
  Serial.println(F("Historical Data Acquired"));

  /* CALCULATE VOLUME OF WATER IN RWH TANK USING YESTERDAY'S PRECIPITATION */
  waterInTank += (roofArea * rain_total*25.4 * C_roof * 0.623); // [gal]
  // Make Sure Water Level Isn't Greater Than Tank Size
  if (waterInTank > tankSize) {
    waterInTank = tankSize;
  }
  Serial.println(F("Water in Tank Updated"));

  /* SOIL MOISTURE CALCULATIONS */
```

```
  // Step 1: ET Calc
  e_s = e_s_calc((T_max - 273.15), (T_min - 273.15)); // sat. pressure calc --
convert T_max, T_min to Celsius from K
  delta = delta_calc(T_mean - 273.15); // slope of sat. pressure curve calc --
convert T_mean to Celsius from K
  e_a = e_a_calc(rel_hum_max, rel_hum_min, (T_max - 273.15), (T_min - 273.15));
// actual vapor pressure calc -- convert T_max, T_min to Celsius from K
  R_n = radiation_calc(day, lat_deg, (T_max - 273.15), (T_min - 273.15), e_a); //
net radiation calc -- convert T_max, T_min to Celsius from K
  ET = FAO_ET_calc(R_n, (T_mean - 273.15), u_mean, e_s, e_a, delta); // ET Calc
[in/day] -- convert T_mean to Celsius from K
  Serial.print(F("ET Calculated: "));
  Serial.println(ET);

  // Step 2: Precipitation Infiltration Calc
  if (rain_total > (0.2*S)) {
    F_P = (((rain_total - 0.2*S)*S) / (rain_total + 0.8*S)) / 25.4; // converted
to [inches]
  }
  else {
    F_P = 0;
  }
  Serial.print(F("Precipitation Infiltration Calculated: "));
  Serial.println(F_P);

  // Step 3: Irrigation Infiltration Calc
  if ((I_depth*25.4) > (0.2*S)) {
    F_I = (((I_depth*25.4 - 0.2*S)*S) / (I_depth*25.4 + 0.8*S)) / 25.4; //
converted to [inches]
  }
  else {
    F_I = 0;
  }
  Serial.print(F("Irrigation Infiltration Calculated: "));
  Serial.println(F_I);

  // Step 4: Soil Moisture Calc
  WC = WC + F_P + F_I - ET;
  // Make Sure Moisture Doesn't Exceed Field Capacity or Goes Negative
  if (WC > FC) {
    WC = FC;
  }
  if (WC < 0) {
    WC = 0;
  }
```

```
  Serial.print(F("Soil Moisture Calculated: "));
  Serial.println(WC);

  /* CHECK WATER CONTENT AGAINST THRESHOLD */
  if (WC < WC_min) {
    Serial.println(F("Water Content Below Treshold"));
    waterDays ++;
    /* CALCULATE PREDICTED INFILTRATION FROM RAINFALL IN NEXT 3 DAYS (TODAY,
TOMORROW, NEXT DAY) */
    // Step 1: Make API Call to Query for Rainfall in Next 3 Days
    // Getting 6 hrs at a time from API, so need to call 12 times to get data for
next 72 hrs
    for (int i=0; i<12; i++) {
      // Build the URL for the API call
      url = String("/data/2.5/history/city?id=") + cityCode + "&type=hour&start="
+ startTimeForecast + "&cnt=" + cnt + "&appid=" + apiKey;
      // Make API Call and Get JSON doc
      String response = getResponseFromAPI(client, url);
      DynamicJsonDocument doc(3072);
      DeserializationError error = deserializeJson(doc, response);
      if (error) {
        Serial.print(F("Deserialization failed: "));
        Serial.println(error.c_str());
        i -= 1; // This will make the for loop repeat with the current i value
        ArduinoCloud.update(); // should help reconnect to the Cloud if
connection is lost (this is why the API Call would fail)
      }
      else {
        response = "";
        // If it's the first set of 6 hrs, set initial rainfall total
        if (i == 0) {
          predicted_rain = get_rain_sum(doc);
        }
        // If it's the 2nd-12th set of 6 hrs, update the rainfall total
        else {
          predicted_rain += get_rain_sum(doc);
        }
        // At end of loop, update Start Time for the API Call
        startTimeForecast += (3600 * cnt);
        Serial.print(i);
        Serial.println(F(" out of 12 forecast APIs called"));
      }

    }
    Serial.println(F("Forecast Data Collected"));
```

```
    // Step 2: Calculate Expected Infiltration from Precipitation
    if (predicted_rain > (0.2*S)) {
      F_P_predicted = (((predicted_rain - 0.2*S)*S) / (predicted_rain + 0.8*S)) /
25.4; // converted to [inches]
    }
    else {
      F_P_predicted = 0;
    }
    /* CALCULATE DEPTH OF IRRIGATION NEEDED */
    F_I_desired = FC - WC - F_P_predicted; // amount of infiltration needed from
irrigation
    I_depth = ((S*(0.2*S + 0.8*(F_I_desired*25.4))) / (S - (F_I_desired*25.4))) /
25.4; // depth of irrigation to release [inches]

    /* CALCULATE VOLUME OF IRRIGATION NEEDED USING LAWN SIZE */
    I_vol = ((I_depth/12) * lawnSize) * 7.481; // volume of irrigation to release
[gallons]
    Serial.println(F("Irrigation Depth and Volume Calculated"));

    /* RELEASE WATER FROM SYSTEM -- FIRST FROM RWH TANK, THEN FROM PIPED WATER
SUPPLY */
    // if sufficient water in tank, release all water from RWH
    if (I_vol <= waterInTank) {
      // Turn Green LED on for 3 seconds to indicate RWH supply is being used
      WiFiDrv::digitalWrite(26, HIGH); // Green ON
      delay(3000); // Wait 3 seconds
      WiFiDrv::digitalWrite(26, LOW); // Green OFF
      // Update RWH Water Use Total
      rwhWaterUse += I_vol;
      // Update Total Water Use
      totalWaterUse += I_vol;
      // Update Water Volume in Tank
      waterInTank -= I_vol;
    }
    // if insufficient water in tank BUT tank has some water, empty RWH tank and
then finish watering with piped supply
    else if ((I_vol > waterInTank) && (waterInTank > 0)) {
      // Turn Green LED on for 3 seconds to indicate RWH supply is being used
      WiFiDrv::digitalWrite(26, HIGH); // Green ON
      delay(3000); // Wait 3 seconds
      WiFiDrv::digitalWrite(26, LOW); // Green OFF
      // Update RWH Water Use Total
      rwhWaterUse += waterInTank;
      // Update Piped Water Use Total
```

```
      pipedWaterUse += (I_vol - waterInTank);
      // Update Total Water Use
      totalWaterUse += I_vol;
      // Update Water Volume in Tank -- tank empty
      waterInTank = 0;
      // Turn Red LED on for 3 seconds to indicate piped supply is being used
      WiFiDrv::digitalWrite(25, HIGH); // Red ON
      delay(3000); // Wait 3 seconds
      WiFiDrv::digitalWrite(25, LOW); // Red OFF
    }
    // else (RWH tank is completely empty), release all water from piped supply
    else {
      // Turn Red LED on for 3 seconds to indicate piped supply is being used
      WiFiDrv::digitalWrite(25, HIGH); // Red ON
      delay(3000); // Wait 3 seconds
      WiFiDrv::digitalWrite(25, LOW); // Red OFF
      // Update Piped Water Use Total
      pipedWaterUse += I_vol;
      // Update Total Water Use
      totalWaterUse += I_vol;
    }
  }

  // If Water Content is Above the Threshold, then set the Irrigation Depth and
Volume for the Day to Zero
  else {
    I_depth = 0;
    I_vol = 0;
  }

  /* UPDATE DATE FOR THE NEXT LOOP */
  // Go to Next Day
  day ++;
  Serial.println(F("Day Updated"));
  // If end of watering season, print out message and turn Blue LED light on to
signal end of simulation */
  if (day > last_day) {
    Serial.println(F("Simulation Complete"));
    Serial.print(F("Total RWH Use: "));
    Serial.println(rwhWaterUse);
    Serial.print(F("Total Piped Water Use: "));
    Serial.println(pipedWaterUse);
    Serial.print(F("Total Water Use: "));
    Serial.println(totalWaterUse);
    Serial.print(F("Total Days Watered: "));
```

```
    Serial.println(waterDays);
    WiFiDrv::digitalWrite(27, HIGH); // Blue Light ON
    //delay(10000); // keep on for 10 seconds
    //WiFiDrv::digitalWrite(27, LOW); // Blue Light OFF
  }
  // Update Start Time for Forecast API Call (24 hours ahead of Historical Start
Time)
  startTimeForecast = startTimeHistorical + 86400;
  Serial.println(F("Forecast Start Time Updated"));

  delay(1000);

}
```

Helper File for Functions Related to API Calls: API_dataFunctions.h

```
// Variables for Getting Data from API Call
float rain_total = 0; // [mm]
float T_max; // [K]
float T_min; // [K]
float T_mean; // [K]
int rel_hum_max; // [%]
int rel_hum_min; // [%]
float u_mean; // Avg Wind Speed [m/s]
int cnt = 6;

/* FUNCTION: CALL API FOR 6-HR STEP AND RETURN RESPONSE */
String getResponseFromAPI(HttpClient client, String url) {
  // Make the API call
  client.beginRequest();
  client.get(url);
  client.endRequest();

  // Read status code and body of the response
  int statusCode = client.responseStatusCode();
  String response = client.responseBody();
  //Serial.print(F("Status code: "));
  //Serial.println(statusCode);
  //Serial.println(F("Response Found"));
  return response;
}
```

```
/* FUNCTION: GET MAX TEMP FROM cnt-HR API CALL */
float get_T_max(DynamicJsonDocument doc) {
  float temp;
  float T_max1;
  for (int i=0; i<cnt; i++) {
    temp = doc["list"][i]["main"]["temp"]; // [Kelvin]
    if (i == 0) {
      T_max1 = temp;
    }
    else {
      if (temp > T_max1) {
        T_max1 = temp;
      }
    }
  }
  return T_max1; // [K]
}

/* FUNCTION: GET MIN TEMP FROM cnt-HR API CALL */
float get_T_min(DynamicJsonDocument doc) {
  float temp;
  float T_min1;
  for (int i=0; i<cnt; i++) {
    temp = doc["list"][i]["main"]["temp"]; // [Kelvin]
    if (i == 0) {
      T_min1 = temp;
    }
    else {
      if (temp < T_min1) {
        T_min1 = temp;
      }
    }
  }
  return T_min1; // [K]
}

/* FUNCTION: GET MEAN TEMP FROM cnt-HR API CALL */
float get_T_mean(DynamicJsonDocument doc) {
  float total = 0;
  float temp;
  float T_mean1;
  for (int i=0; i<cnt; i++) {
    temp = doc["list"][i]["main"]["temp"]; // [Kelvin]
    total += temp;
```

```
  }
  T_mean1 = total / cnt;
  return T_mean1; // [K]
}

/* FUNCTION: GET MAX RELATIVE HUMIDITY FROM cnt-HR API CALL */
int get_rel_hum_max(DynamicJsonDocument doc) {
  int rel_hum;
  int rel_hum_max1;
  for (int i=0; i<cnt; i++) {
    rel_hum = doc["list"][i]["main"]["humidity"]; // [%]
    if (i == 0) {
      rel_hum_max1 = rel_hum;
    }
    else {
      if (rel_hum > rel_hum_max1) {
        rel_hum_max1 = rel_hum;
      }
    }
  }
  return rel_hum_max1; // [%]
}

/* FUNCTION: GET MIN RELATIVE HUMIDITY FROM cnt-HR API CALL */
int get_rel_hum_min(DynamicJsonDocument doc) {
  int rel_hum;
  int rel_hum_min1;
  for (int i=0; i<cnt; i++) {
    rel_hum = doc["list"][i]["main"]["humidity"]; // [%]
    if (i == 0) {
      rel_hum_min1 = rel_hum;
    }
    else {
      if (rel_hum < rel_hum_min1) {
        rel_hum_min1 = rel_hum;
      }
    }
  }
  return rel_hum_min1; // [%]
}

/* FUNCTION: GET MEAN WIND SPEED FROM cnt-HR API CALL */
float get_wind_mean(DynamicJsonDocument doc) {
  float total = 0;
  float wind;
```

```
  float u_mean1;
  for (int i=0; i<cnt; i++) {
    wind = doc["list"][i]["wind"]["speed"]; // [m/s]
    total += wind;
  }
  u_mean1 = total / cnt;
  return u_mean1; // [m/s]
}

/* FUNCTION: GET TOTAL RAINFALL DEPTH FROM cnt-HR API CALL */
float get_rain_sum(DynamicJsonDocument doc) {
  float rain;
  float rain_total1 = 0;
  for (int i=0; i<cnt; i++) {
    if (doc["list"][i]["rain"]["1h"]) {
      rain = doc["list"][i]["rain"]["1h"]; // [mm]
    }
    else {
      rain = 0;
    }
    rain_total1 += rain;
  }
  return rain_total1; // [mm]
}
```

Helper File for Functions Related to Penman-Monteith Equation: FAO_P-M_functions.h

```
// Function to Calculate Water Content Threshold based on inputs
float threshold_calc(float awc, float mad, float depth) {
  float threshold = awc * mad * depth/12;
  return threshold;
}

// FAO Penman-Monteith Equation for Calculating Evapotranspiration Rate, using
these inputs:
  // R_n; Net radiation [MJ/m2/day]
  // T; Mean daily air temp. [Celsius]
  // u; Wind speed at 2m height [m/s]
  // e_s; Saturation vapor pressure [kPa]
  // e_a; Actual vapor pressure [kPa]
  // delta; Slope vapor pressure curve [kPa/Celsius]
```

```
float FAO_ET_calc(float R_n, float T, float u, float e_s, float e_a, float delta)
{
  float ET; // ET rate [mm/day]
  float G = 0; // Soil heat flux density [MJ/m2/day]
  float gamma = 0.000662 * 101.3; // psychrometric constant [kPa/Celsius] --
assume Asmann type psychrometer and atmos. pressure at sea level

  ET = (0.408 * delta * (R_n - G) + gamma * 900/(T+273) * u * (e_s - e_a)) /
(delta + gamma * (1 + 0.34*u));
  float ET_inches = ET / 25.4; // ET value in [in/day]
  return ET_inches;
}

// Saturation Vapor Pressure for a given temperature
float e_s0(float temp) {
  float e_s0 = 0.6108 * exp(17.27 * temp / (temp + 237.3));
  return e_s0;
}

// Calculate Saturation Vapor Pressure for FAO P-M Equation
float e_s_calc(float T_max, float T_min) {
  float e_s0_maxT = e_s0(T_max);
  float e_s0_minT = e_s0(T_min);
  float e_s = (e_s0_maxT + e_s0_minT) / 2;
  return e_s;
}

// Calculate Slope of Vapor Pressure Curve (delta) for FAO P-M Equation, using
mean temp
float delta_calc(float T_mean) {
  float e_s0_meanT = e_s0(T_mean);
  float delta = 4098 * e_s0_meanT / sq(T_mean + 237.3);
  return delta;
}

// Calculate Actual Vapor Pressure for FAO P-M Equation, using relative humidity
float e_a_calc(float rel_hum_max, float rel_hum_min, float T_max, float T_min) {
  float e_s0_maxT = e_s0(T_max);
  float e_s0_minT = e_s0(T_min);
  float e_a = (e_s0_minT * rel_hum_max / 100 + e_s0_maxT * rel_hum_min / 100) /
2;
  return e_a;
}
```

```
// Calculate Net Radiation for FAO P-M Equation -- T_max, T_min both in Celsius -
- e_a is actual vapor pressure
float radiation_calc(float day, float lat_deg, float T_max, float T_min, float
e_a) {
  // Extraterrestrial Radiation
  float G_sc = 0.0820; // solar constant [MJ/m2/min]
  float lat_rad = PI / 180 * lat_deg; // latitude in radians
  float d_r = 1 + 0.033 * cos(2 * PI * day / 365); // inverse relative distance
Earth-Sun
  float solar_decl = 0.409 * sin(2 * PI * day / 365 - 1.39); // solar declination
[rad]
  float omega_s = acos(-tan(lat_rad) * tan(solar_decl)); // sunset hour angle
[rad]
  float R_a = 24 * 60 / PI * G_sc * d_r * (omega_s * sin(lat_rad) *
sin(solar_decl) + cos(lat_rad) * cos(solar_decl) * sin(omega_s)); //
Extraterrestrial Radiation (MJ/m2/day)

  // Daylight Hours
  float N = 24 * omega_s / PI;

  // Solar Radiation
  float k_rs = 0.19; // adjustment coefficient [Celsius^-0.5]
  float R_s = k_rs * sqrt(T_max - T_min) * R_a; // Solar Radiation

  // Clear-Sky Solar Radiation
  float R_so = 0.75 * R_a;

  // Net Solar / Shortwave Radiation
  float albedo = 0.23; // albedo for grass
  float R_ns = (1 - albedo) * R_s;

  // Net Longwave Radiation
  float sb_const = 4.903 * pow(10, -9); // Stefan-Boltzmann constant
[MJ/K4/m2/day]
  float T_maxK = T_max + 273.16; // max temp in Kelvin
  float T_minK = T_min + 273.16; // min temp in Kelvin
  float R_nl = sb_const * ((pow(T_maxK, 4) + pow(T_minK, 4)) / 2) * (0.34 - 0.14
* sqrt(e_a)) * (1.35 * R_s / R_so - 0.35);

  // Net Radiation [MJ/m2/day]
  float R_n = R_ns - R_nl;
  return R_n;
}
```

Helper File Containing WIFI Credentials: arduino_secrets.h

```
#define SECRET_SSID "Harvard University" // name of network
#define SECRET_USER "bbeauregard@college.harvard.edu" // eg, x@seas.harvard.edu
#define SECRET_PASS "" // leave this blank
```

JSON File Created by Arduino Cloud: sketch.json

```
{
  "cpu": {
    "fqbn": "arduino:samd:mkrwifi1010",
    "name": "Arduino MKR WiFi 1010",
    "type": "serial"
  },
  "secrets": [],
  "included_libs": []
}
```

Helper File Created by Arduino Cloud: thingProperties.h

```
// Code generated by Arduino IoT Cloud, DO NOT EDIT.

#include <ArduinoIoTCloud.h>
#include <Arduino_ConnectionHandler.h>

const char THING_ID[] = "ES 100"; // your Thing ID (!)

const char SSID[] = SECRET_SSID;    // Network SSID (name)
const char USER[] = SECRET_USER;
const char PASS[] = SECRET_PASS;

void initProperties(){


}
WiFiConnectionHandler ArduinoIoTPreferredConnection(SSID, PASS);
```

APPENDIX D: PYTHON CODE FOR SIMULATIONS

```
# Import Libraries
import math
import numpy as np
import json
from csv import writer

# DEFINE FAO P-M FUNCTIONS

# Function to Calculate Water Content Threshold based on inputs
def threshold_calc(awc, mad, depth):
    threshold = awc * mad * depth/12
    return threshold

# FAO Penman-Monteith Equation for Calculating Evapotranspiration Rate, using
these inputs:
  # R_n; Net radiation [MJ/m2/day]
  # T; Mean daily air temp. [Celsius]
  # u; Wind speed at 2m height [m/s]
  # e_s; Saturation vapor pressure [kPa]
  # e_a; Actual vapor pressure [kPa]
  # delta; Slope vapor pressure curve [kPa/Celsius]
def FAO_ET_calc(R_n, T, u, e_s, e_a, delta):
    G = 0 # Soil heat flux density [MJ/m2/day]
    gamma = 0.000662 * 101.3 # psychrometric constant [kPa/Celsius] -- assume
Asmann type psychrometer and atmos. pressure at sea level
    ET = (0.408 * delta * (R_n - G) + gamma * 900/(T+273) * u * (e_s - e_a)) /
(delta + gamma * (1 + 0.34*u))
    ET_inches = ET / 25.4 # ET value in [in/day]
    return ET_inches

# Saturation Vapor Pressure for a given temperature
def e_s0(temp):
    e_s0 = 0.6108 * math.exp(17.27 * temp / (temp + 237.3))
    return e_s0

# Calculate Saturation Vapor Pressure for FAO P-M Equation
def e_s_calc(T_max, T_min):
    e_s0_maxT = e_s0(T_max)
    e_s0_minT = e_s0(T_min)
    e_s = (e_s0_maxT + e_s0_minT) / 2
    return e_s

# Calculate Slope of Vapor Pressure Curve (delta) for FAO P-M Equation, using
mean temp
def delta_calc(T_mean):
    e_s0_meanT = e_s0(T_mean)
    delta = 4098 * e_s0_meanT / pow(T_mean + 237.3, 2)
    return delta

# Calculate Actual Vapor Pressure for FAO P-M Equation, using relative humidity
def e_a_calc(rel_hum_max, rel_hum_min, T_max, T_min):
    e_s0_maxT = e_s0(T_max)
```

```
    e_s0_minT = e_s0(T_min)
    e_a = (e_s0_minT * rel_hum_max / 100 + e_s0_maxT * rel_hum_min / 100) / 2
    return e_a

# Calculate Net Radiation for FAO P-M Equation -- T_max, T_min both in Celsius --
e_a is actual vapor pressure
def radiation_calc(day, lat_deg, T_max, T_min, e_a):
    # Extraterrestrial Radiation
    G_sc = 0.0820 # solar constant [MJ/m2/min]
    lat_rad = math.pi / 180 * lat_deg # latitude in radians
    d_r = 1 + 0.033 * math.cos(2 * math.pi * day / 365) # inverse relative
distance Earth-Sun
    solar_decl = 0.409 * math.sin(2 * math.pi * day / 365 - 1.39) # solar
declination [rad]
    omega_s = math.acos(-math.tan(lat_rad) * math.tan(solar_decl)) # sunset hour
angle [rad]
    R_a = 24 * 60 / math.pi * G_sc * d_r * (omega_s * math.sin(lat_rad) *
math.sin(solar_decl) + math.cos(lat_rad) * math.cos(solar_decl) *
math.sin(omega_s)) # Extraterrestrial Radiation (MJ/m2/day)

    # Daylight Hours
    N = 24 * omega_s / math.pi

    # Solar Radiation
    k_rs = 0.19 # adjustment coefficient [Celsius^-0.5]
    R_s = k_rs * math.sqrt(T_max - T_min) * R_a # Solar Radiation

    # Clear-Sky Solar Radiation
    R_so = 0.75 * R_a

    # Net Solar / Shortwave Radiation
    albedo = 0.23 # albedo for grass
    R_ns = (1 - albedo) * R_s

    # Net Longwave Radiation
    sb_const = 4.903 * pow(10, -9) # Stefan-Boltzmann constant [MJ/K4/m2/day]
    T_maxK = T_max + 273.16 # max temp in Kelvin
    T_minK = T_min + 273.16 # min temp in Kelvin
    R_nl = sb_const * ((pow(T_maxK, 4) + pow(T_minK, 4)) / 2) * (0.34 - 0.14 *
math.sqrt(e_a)) * (1.35 * R_s / R_so - 0.35)

    # Net Radiation [MJ/m2/day]
    R_n = R_ns - R_nl
    return R_n

# DEFINE DATA COLLECTION FUNCTIONS FOR JSON FILE
cnt = 24

# GET MAX TEMP FOR cnt-HR RANGE FROM JSON FILE
def get_T_max(doc, time):
    # find the index of the element with the correct start time
    for num, x in enumerate(doc):
```

```
        if x["dt"] == time:
            startnum = num
            break

    T_max1 = 0
    for i in np.arange(startnum, startnum + cnt):
        temp = doc[i]["main"]["temp"] # [Kelvin]
        if i == 0:
            T_max1 = temp
        else:
            if temp > T_max1:
                T_max1 = temp

    return T_max1 # [K]

# GET MIN TEMP FOR cnt-HR RANGE FROM JSON FILE
def get_T_min(doc, time):
    # find the index of the element with the correct start time
    for num, x in enumerate(doc):
        if x["dt"] == time:
            startnum = num
            break

    T_min1 = 1000
    for i in np.arange(startnum, startnum + cnt):
        temp = doc[i]["main"]["temp"] # [Kelvin]
        if i == 0:
            T_min1 = temp
        else:
            if temp < T_min1:
                T_min1 = temp

    return T_min1 # [K]

# GET MEAN TEMP FOR cnt-HR RANGE FROM JSON RANGE
def get_T_mean(doc, time):
    # find the index of the element with the correct start time
    for num, x in enumerate(doc):
        if x["dt"] == time:
            startnum = num
            break

    total = 0
    for i in np.arange(startnum, startnum + cnt):
        temp = doc[i]["main"]["temp"] # [Kelvin]
        total += temp

    T_mean1 = total / cnt
    return T_mean1 # [K]

# GET MAX RELATIVE HUMIDITY FOR cnt-HR RANGE FROM JSON FILE
def get_rel_hum_max(doc, time):
```

```
    # find the index of the element with the correct start time
    for num, x in enumerate(doc):
        if x["dt"] == time:
            startnum = num
            break

    rel_hum_max1 = 0
    for i in np.arange(startnum, startnum + cnt):
        rel_hum = doc[i]["main"]["humidity"] # [%]
        if i == 0:
            rel_hum_max1 = rel_hum
        else:
            if rel_hum > rel_hum_max1:
                rel_hum_max1 = rel_hum

    return rel_hum_max1 # [%]

# GET MIN RELATIVE HUMIDITY FOR cnt-HR RANGE FROM JSON FILE
def get_rel_hum_min(doc, time):
    # find the index of the element with the correct start time
    for num, x in enumerate(doc):
        if x["dt"] == time:
            startnum = num
            break

    rel_hum_min1 = 1000
    for i in np.arange(startnum, startnum + cnt):
        rel_hum = doc[i]["main"]["humidity"] # [%]
        if i == 0:
            rel_hum_min1 = rel_hum
        else:
            if rel_hum < rel_hum_min1:
                rel_hum_min1 = rel_hum

    return rel_hum_min1 # [%]

# GET MEAN WIND SPEED FOR cnt-HR RANGE FROM JSON FILE
def get_wind_mean(doc, time):
    # find the index of the element with the correct start time
    for num, x in enumerate(doc):
        if x["dt"] == time:
            startnum = num
            break

    total = 0
    for i in np.arange(startnum, startnum + cnt):
        wind = doc[i]["wind"]["speed"] # [m/s]
        total += wind

    u_mean1 = total / cnt
    return u_mean1 # [m/s]
```

```
# GET TOTAL RAINFALL DEPTH FOR cnt-HR RANGE FROM JSON FILE
def get_rain_sum(doc, time):
    # find the index of the element with the correct start time
    for num, x in enumerate(doc):
        if x["dt"] == time:
            startnum = num
            break

    rain_total1 = 0
    for i in np.arange(startnum, startnum + cnt):
        if "rain" in doc[i].keys():
            if "1h" in doc[i]["rain"].keys():
                rain = doc[i]["rain"]["1h"] # [mm]
            else:
                rain = doc[i]["rain"]["3h"] / 3 # [mm]
        else:
            rain = 0
        rain_total1 += rain

    return rain_total1 # [mm]

# DEFINE VARIABLES

# Variables Related to Water Content Threshold
awc = 1.45 # Available Water Capacity, in/ft.
mad = 0.5 # Management Allowable Depletion, b/t 0 and 1.
root_depth = 10 # Root Depth of crop, in.
FC = awc * root_depth/12 # Field Capacity of Soil [inches]
WC = FC # Water Content of Soil [inches]
WC_min = threshold_calc(awc, mad, root_depth) # Water Content Threshold [inches]

# Variables for FAO Equation / ET Calcs
first_day = 91 # Day of Year -- start at April 1st -- 91 on normal year, 92 on
leap year
last_day = 304 # Last Day of Watering Season, October 31st -- 304 on normal year,
305 on leap year
days = np.arange(first_day, last_day + 1)

#lat_deg = 42.360082 # decimal degree latitude of Boston
#lat_deg  = 42.262593 # decimal degree latitude of Worcester
#lat_deg = 42.519747 # decimal degree latitude of Salem
lat_deg = 41.958446 # decimal degree latitude of Plymouth

# Variables for Infiltration Calcs
CN = 65 # Runoff Curve Number
S = 25400/CN - 254 # Surface Storage [mm]
rain_yesterday = 0.0 # Rainfall from Previous Day, initially zero [mm]

# Variables for Irrigation Depth & Volume
I_depth = 0.0
I_vol = 0.0
```

```
# USER INPUTS
lawnSizes = [2000, 10000, 20000] # Size of Lawn [square feet]
tankSizes = [250, 1000, 5000] # Size of Tank [gallons]
roofArea = 1500 # Area of Roof [square feet]

# Define Scenarios
scenarios = {
    'Scenario 1':{'lawn':lawnSizes[0], 'tank':tankSizes[0]},
    'Scenario 2':{'lawn':lawnSizes[0], 'tank':tankSizes[1]},
    'Scenario 3':{'lawn':lawnSizes[0], 'tank':tankSizes[2]},
    'Scenario 4':{'lawn':lawnSizes[1], 'tank':tankSizes[0]},
    'Scenario 5':{'lawn':lawnSizes[1], 'tank':tankSizes[1]},
    'Scenario 6':{'lawn':lawnSizes[1], 'tank':tankSizes[2]},
    'Scenario 7':{'lawn':lawnSizes[2], 'tank':tankSizes[0]},
    'Scenario 8':{'lawn':lawnSizes[2], 'tank':tankSizes[1]},
    'Scenario 9':{'lawn':lawnSizes[2], 'tank':tankSizes[2]}
}

# Define Years
years = np.arange(2013, 2023)

# Variables for Rainwater Harvesting Tank
C_roof = 0.90 # runoff coefficient for roof

# Variables to Collect Water Use Data During Simulations
rwhWaterUses = 0.0
pipedWaterUse = 0.0
totalWaterUse = 0.0
waterDays = 0

# Start Timestamps for Weather Data
startTimesHistorical = [1364731200, 1396267200, 1427803200, 1459425600,
1490961600, 1522497600, 1554033600, 1585656000, 1617192000 ,1648728000] #
Historical API Call starts at March 31st, 8am -- this is for the year 2022

# Read in JSON File
with open(r'C:\Users\bbeau\OneDrive\Documents\ES 100\History Bulk
Data\PlymouthHistoryBulk2013_present.json', 'r') as datafile:
    doc = json.load(datafile)


# THE ALGORITHM ITSELF

# Loop Through All Days in the Watering Season
for y, year in enumerate(years):
    print(year)
    row_to_add = [year]
    if year == 2016 or year == 2020:
        first_day = 92 # Day of Year -- start at April 1st -- 91 on normal year,
92 on leap year
        last_day = 305 # Last Day of Watering Season, October 31st -- 304 on
normal year, 305 on leap year
```

```
        days = np.arange(first_day, last_day + 1)
    else:
        first_day = 91 # Day of Year -- start at April 1st -- 91 on normal year,
92 on leap year
        last_day = 304 # Last Day of Watering Season, October 31st -- 304 on
normal year, 305 on leap year
        days = np.arange(first_day, last_day + 1)


    for scenario in scenarios:
        # FIRST RESET ALL THE NECESSARY VARIABLES
        # Lawn and Tank Size Variables
        lawnSize = scenarios[scenario]['lawn']
        tankSize = scenarios[scenario]['tank']
        waterInTank = tankSize # Volume of Water in RWH Tank, assume full at
start [gallons]
        rain_yesterday = 0.0 # Rainfall from Previous Day, initially zero [mm]
        # Variables to Collect Water Use Data During Simulations
        rwhWaterUse = 0.0
        pipedWaterUse = 0.0
        totalWaterUse = 0.0
        waterDays = 0
        # Variables for Irrigation Depth & Volume
        I_depth = 0.0
        I_vol = 0.0
        # Variables for Start Times
        startTimeHistorical = startTimesHistorical[y]
        startTimeForecast = startTimeHistorical + 86400
        print(scenario)
        print("Start Time: ", startTimeHistorical)


        for day in days:
            # BEGINNING OF DAY (8am)
            # COMPILE NECESSARY DATA FOR PREVIOUS DAY FROM JSON FILE
            T_max = get_T_max(doc, startTimeHistorical)
            T_min = get_T_min(doc, startTimeHistorical)
            T_mean = get_T_mean(doc, startTimeHistorical)
            rel_hum_max = get_rel_hum_max(doc, startTimeHistorical)
            rel_hum_min = get_rel_hum_min(doc, startTimeHistorical)
            u_mean = get_wind_mean(doc, startTimeHistorical)
            rain_total = get_rain_sum(doc, startTimeHistorical)
            # Update Start Time for Historical Data
            startTimeHistorical += (3600 * cnt)

            # CALCULATE VOLUME OF WATER IN RWH TANK USING YESTERDAY'S
PRECIPITATION
            waterInTank += (roofArea * rain_total*25.4 * C_roof * 0.623) # [gal]
            # Make Sure Water level Isn't Greater Than Tank Size
            if waterInTank > tankSize:
                waterInTank = tankSize
```

```
            # SOIL MOISTURE CALCULATIONS
            # Step 1: ET Calc
            e_s = e_s_calc((T_max - 273.15), (T_min - 273.15)) # sat. pressure
calc -- convert T_max, T_min to Celsius from K
            delta = delta_calc(T_mean - 273.15) # slope of sat. pressure curve
calc -- convert T_mean to Celsius from K
            e_a = e_a_calc(rel_hum_max, rel_hum_min, (T_max - 273.15), (T_min -
273.15)) # actual vapor pressure calc -- convert T_max, T_min to Celsius from K
            R_n = radiation_calc(day, lat_deg, (T_max - 273.15), (T_min -
273.15), e_a) # net radiation calc -- convert T_max, T_min to Celsius from K
            ET = FAO_ET_calc(R_n, (T_mean - 273.15), u_mean, e_s, e_a, delta) #
ET Calc [in/day] -- convert T_mean to Celsius from K

            # Step 2: Precipitation Infiltration Calc
            if rain_total > (0.2*S):
                F_P = (((rain_total - 0.2*S)*S) / (rain_total + 0.8*S)) / 25.4 #
converted to [inches]
            else:
                F_P = 0

            # Step 3: Irrigation Infiltration Calc
            if (I_depth*25.4) > (0.2*S):
                F_I = (((I_depth*25.4 - 0.2*S)*S) / (I_depth*25.4 + 0.8*S)) /
25.4 # converted to [inches]
            else:
                F_I = 0

            # Step 4: Soil Moisture Calc
            WC = WC + F_P + F_I - ET
            # Make Sure Moisture Doesn't Exceed Field Capacity or Go Negative
            if WC > FC:
                WC = FC
            if WC < 0:
                WC = 0

            # CHECK WATER CONTENT AGAINST THRESHOLD
            if WC < WC_min:
                waterDays += 1
                # CALCULATE PREDICTED INFILTRATION FROM RAINFALL IN NEXT
                # Step 1: Compile Rainfall in Next 3 Days from JSON File
                # Need to Call get_rain_sum() 3 times
                for i in range(3):
                    # for first 24-hrs, set initial rainfall total
                    if i == 0:
                        predicted_rain = get_rain_sum(doc, startTimeForecast)
                    # otherwise, add to the total rainfall
                    else:
                        predicted_rain += get_rain_sum(doc, startTimeForecast)
                    # At end of loop, update Start Time
                    startTimeForecast += (3600 * cnt)

                # Step 2: Calculate Expected Infiltration from Precipitation
```

```
                if predicted_rain > (0.2*S):
                    F_P_predicted = (((predicted_rain - 0.2*S)*S) /
(predicted_rain + 0.8*S)) / 25.4 # converted to [inches]
                else:
                    F_P_predicted = 0

                # CALCULATE DEPTH OF IRRIGATION NEEDED
                F_I_desired = FC - WC - F_P_predicted # amount of infiltration
needed from irrigation
                I_depth = ((S*(0.2*S + 0.8*(F_I_desired*25.4))) / (S -
(F_I_desired*25.4))) / 25.4 # depth of irrigation to release [inches]

                # CALCULATE VOLUME OF IRRIGATION NEEDED USING LAWN SIZE
                I_vol = ((I_depth/12) * lawnSize) * 7.481 # volume of irrigation
to release [gallons]

                # RELEASE WATER FROM SYSTEM -- FIRST FROM RWH TANK, THEN FROM
PIPED WATER SUPPLY
                # if sufficient water in tank, release all water from RWH
                if I_vol <= waterInTank:
                    # Update RWH Water Use Total
                    rwhWaterUse += I_vol
                    # Update Total Water Use
                    totalWaterUse += I_vol
                    # Update Water Volume in Tank
                    waterInTank -= I_vol
                # if insufficient water in tank BUT tank has some water, empty
RWH tank and then finish watering with piped supply
                elif (I_vol > waterInTank) and (waterInTank > 0):
                    # Update RWH Water Use Total
                    rwhWaterUse += waterInTank
                    # Update Piped Water Use Total
                    pipedWaterUse += (I_vol - waterInTank)
                    # Update Total Water Use
                    totalWaterUse += I_vol
                    # Update Water Volume in Tank -- tank empty
                    waterInTank = 0
                # else (RWH tank is completely empty), release all water from
piped supply
                else:
                    # Update Piped Water Use Total
                    pipedWaterUse += I_vol
                    # Update Total Water Use
                    totalWaterUse += I_vol
            # If Water Content is Above the Threshold, then set the Irrigation
Depth and Volume for the Day to Zero
            else:
                I_depth = 0
                I_vol = 0

            # UPDATE START TIME FOR FORECAST API CALL (24 HOURS AHEAD OF
HISTORICAL START TIME
```

```
            startTimeForecast = startTimeHistorical + 86400


        # Print Out Info for the Simulation
        print(scenario)

        # Append Data for the Scenario to the row_to_add List
        row_to_add.append(rwhWaterUse)
        row_to_add.append(pipedWaterUse)
        row_to_add.append(waterDays)

    # At End of Year, Append Row to the CSV File
    with open(r'C:\Users\bbeau\OneDrive\Documents\ES 100\Plymouth Compiled
Data.csv', 'a') as csvfile:
        writer_object = writer(csvfile)
        writer_object.writerow(row_to_add)
```