Building a Smart Irrigation System to Optimize Turfgrass Irrigation with Recycled Water

Casey Levitt[1, 2], Graduate Student Mentor: Max Weiss[2, 3], Faculty Advisor: Junko Munakata-Marr, Ph.D[2, 3]
[1]Johns Hopkins University, [2]ReNUWIt, [3]Colorado School of Mines

Introduction and Background

Turfgrass is America's largest irrigated crop, occupying around 2% of the land of the continental US and requiring substantial amounts of irrigation water (Milesi et al., 2005). Smart irrigation systems have been developed to prevent overwatering of turfgrass and reduce water waste. Such systems take real-time soil moisture data and terminate irrigation once the moisture has reached a sufficient threshold (EPA, 2013). However, even more can be done to reduce the burden of turfgrass irrigation on potable water supplies. This project explores the possibility of irrigating turfgrass with recycled water.

Several water quality challenges are associated with using recycled water for irrigation. Treated sewage effluent contains nutrients and dissolved salts, both of which have detrimental effects on plant health (Bauder et al., 2014). Turfgrass has the potential to thrive with recycled water irrigation because it absorbs excess nutrients and can tolerate higher levels of soil salinity (Harivandi, 1999). However, despite the relatively high saline tolerance of turfgrass, salt can accumulate in the root zone over time and exceed concentrations safe for turfgrass (Harivandi, 2011). Therefore, recycled water irrigation requires that soil saline concentrations be monitored. This project aims to develop a proof-of-concept smart irrigation system that irrigates turfgrass with recycled water safely by tracking both soil salinity and soil moisture.

Methods

This smart irrigation system has four main components: the in-ground sensors, the irrigation valve controller, the basestation, and the server (Figure 1). The sensor units (or "nodes") are responsible for taking soil moisture and salinity readings and sending them to the basestation. The irrigation valve controller executes commands to start or stop watering. The basestation is responsible for facilitating communication between the different components. The server is the central component of the system, helping to orchestrate operations. Sensor readings are routed through the basestation to the server. The server uses the readings to generate irrigation commands, which are routed through the basestation to the valve controller.

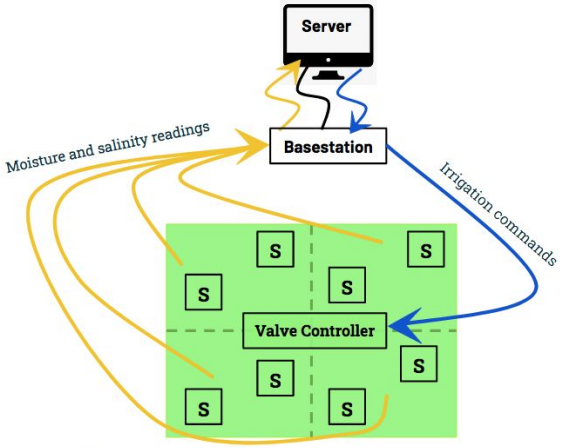

Figure 1. Components and communication schematic

Each of these components have actively running software. The server runs Python code on a standard laptop, while the other components run on Arduino microcontrollers. Each Arduino has an XBee radio module mounted on it, connected by a shield. The XBee module enables wireless communication.

The system was originally deployed in August of 2019, and a few major problems arose. The first observed problem was unreliable communication between the server, the basestation, and the valve controller. Irrigation commands sent from the server (and routed through the basestation) did not seem to be received by the valve controller. The approach taken to solve this problem was thorough testing and debugging of this communication line. To test that the controller was receiving and executing commands, the irrigation valves were modeled with LEDs on a breadboard; an LED on indicated that the valve controller turned on an irrigation valve.

The second observed problem was a battery power issue. The nodes are powered by 12V 9Ah sealed lead acid (SLA) batteries, each with a 5 Wh solar panel attached. When the system was deployed, the solar batteries proved to be incapable of supplying the nodes with enough power to keep them running indefinitely. It became clear that keeping the nodes fully powered at all times was not feasible for the solar batteries. The approach taken to solve this problem was research and testing to enable the nodes to power-down when not actively taking readings.

Results and Discussion

Thorough testing of the server-basestation-controller communication line revealed that the communication failure occurred when multiple commands were sent from the server too rapidly. The basestation code was written with the assumption that commands would be spaced out. As a result, the basestation would read in the entirety of the data on the communication port as one command, even when multiple commands had piled up. Minor changes to the basestation code resolved this issue, and the valve controller showed no further problems receiving and responding to irrigation commands.

The issue of powering down the nodes proved more complex. The nodes need to send readings once per hour throughout the day and every minute during the watering period, which lasts for 15 minutes and occurs once per day. The nodes can therefore be powered down for the majority of every hour. Each "node" is really made up of 3 pieces of hardware, all of which use power: the Toro sensor, the XBee radio module, and the Arduino microcontroller. The goal was for all 3 components to power down together.

Previous work enabled the moisture sensors to turn off completely when they are not needed. Power from the solar battery is routed through the Arduino to the sensor. When the node has finished taking and sending readings, the Arduino shuts off the current flow to the sensor using a transistor.

The XBee S2C radio modules have built-in functionality that allows them to "sleep," or shut off, and "wake up" after a set period of time. I successfully reconfigured the XBee modules to sleep for 50-minute periods. This also required changes in the basestation code regarding communication with the nodes, as the nodes are unable to communicate when the XBee radios are asleep.

The Arduino microcontroller also has built-in sleep modes. However, unlike the XBee radios, the Arduino was not designed to be able to sleep for long periods and wake up with a timer. The internal timer that wakes the Arduino from sleep (called the "watchdog timer") has a maximum value of only 8 seconds. It is possible to run these 8 second sleep intervals in a loop, effectively achieving longer periods of sleep (the loop would iterate 375 times to reach 50 minutes). However, the watchdog timer is very inaccurate, and it is inconsistent between Arduino boards (Figure 2). To ensure that all of the Arduinos are awake when readings are needed, I decreased the number of sleep iterations to 320. Unfortunately, this means several of the Arduinos may wake up early and use unnecessary power.

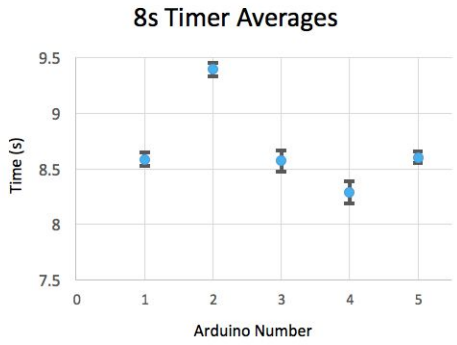

Figure 2. Arduino Uno Watchdog Timer Testing

	Awake	Asleep
Sensor	35mA	0mA
XBee	15mA	0mA
Arduino	45mA	30mA
Total	**95mA**	**30mA**

Figure 3. Current draw measurements for node components

As can be seen in Figure 3, configuring all of the node components to sleep simultaneously reduces the nodes' current draw from 95mA to 30mA, which is a decrease of almost 68%. We anticipate that by enabling the nodes to sleep for the majority of every hour, the solar batteries will be able to power the nodes for much longer periods of time, if not indefinitely.

Conclusion and Next steps

Based on initial testing, it seems that the major issues preventing this system from succeeding as a proof of concept have been resolved. The next step for this project is the full redeployment of the system. Barring any further issues, the system will autonomously irrigate the test plot and collect data on the soil moisture, soil salinity, and amount of water applied. This data will be used to make a more final determination on the feasibility of turfgrass irrigation with recycled water.

References

Milesi, C., Elvidge, C. D., Dietz, J. B., Tuttle, B. T., Nemani, R. R., & Running, S. W. (2005). A Strategy for Mapping and Modeling the Ecological Effects of US Lawns. *International Society for Photogrammetry and Remote Sensing, XXXVI-8/W27*, 1-6.

United States Environmental Protection Agency. (2013). *WaterSense Notice Of Intent (NOI) to Develop a Draft Specification for Soil Moisture-Based Control Technologies.* https://www.epa.gov/sites/production/files/2017-01/documents/ws-products-noi-sms.pdf

Harivandi, M. A. (2011). Purple Gold; A Contemporary View of Recycled Water Irrigation. United States Golf Association Green Section Record, 49 (45), 4-7.

Bauder, T. A., Waskom, R. M., Sutherland, P. L., & Davis, J. G. (2014). *Irrigation Water Quality Criteria. Colorado State University Extension, Crop Series Irrigation* (0.506), 1-4.

Harivandi, M. A. (1999). Interpreting Turfgrass Irrigation Water Test Results. *University of California Division of Agriculture and Natural Resources* (8009), 1-9. https://doi.org/10.3733/ucanr.8009