iSprinkle: when education, innovation and application meet

Carlos Adrian Gomez, Michael Soltys [(1)], Adam Sędziwy [(2)]

(1) Department of Computer Science, California State University Channel Islands,
One University Drive, Camarillo, CA 93012, USA
(2) AGH University of Science and Technology, al. Mickiewicza 30, 30-059 Kraków, Poland

---

Abstract
This paper presents a senior undergraduate project which consists in the implementation of a Raspberry Pi based sprinkler system. The outcome is ingenious and innovative for several reasons: it is a low cost product, but of high quality and versatility (arguably better and cheaper than most household products on the market); it is well suited for usage in draught stricken regions (such as Southern California); it interfaces automatically with weather stations on the Internet, and adapts its sprinkling according to forecasts. For the student this was an opportunity to apply a wide range of technologies, APIs, and to work with watering regulations. The project was both pedagogically rich and intellectually challenging.

Keywords: iSprinkle, water management, draught, open source, Raspberry Pi

I. Introduction

The aim of this paper is to report on the design and implementation of a weather forecast data driven sprinkler timer for home usage, as well as to expound on the pedagogical value of the exercise. The work provides enough details and references so that the reader can replicate the final product, and thus obtain a working sprinkler timer, as well as reports on the insights for instructors who may wish to repeat this project and draw learning value for the student.

The end result is a functioning prototype, but, more importantly, the project has tremendous pedagogical value as it combines advanced programming, introduction to embedded systems and controllers, application, as well as aspects of sustainability and California State law. Moreover, it opens prospects for future development for commercial purposes. This last aspect strongly motivates a student toward creating a high quality solution.

iSprinkle is a Raspberry Pi-powered irrigation controller which allows the user to set an initial, default irrigation schedule for a sprinkler system using a web interface. The crucial feature of the project is that iSprinkle can connect a weather forecast provider to obtain predicted local weather conditions and to adjust the base watering schedule as-needed. By doing so, iSprinkle is able to irrigate more efficiently compared to a system with a fixed schedule (most household systems) in terms of water usage; by programmatically modifying the user's watering schedule, iSprinkle increases/decreases the irrigation time that the schedule dictates depending on data that it receives from a weather API. iSprinkle hopes to make it easier for homeowners to conserve water by automating adjustments to their irrigation schedule.

II. Motivation

The student was easily motivated by the real-life application of the project. The Southern California draught is a complex problem that affects everyone. This project allows the student to make a contribution to the solution; this in itself has the effect of placing the project beyond “academia.” From the point of view of the instructor, the project is an excellent opportunity for the student to put into action concepts and techniques learned throughout the student’s undergraduate years: software design, programming, embedded systems, human-computer interaction, and how these technologies connect with social issues and the law. The students’ work is both conceptual, and highly technical.

II.1 Problem analysis

The current draught in California has become so severe that the state and local governments have begun regulating water consumption, imposing sanctions and even passing legislation in order to curb water usage. On average, the statewide ratio for water usage is about 50% environmental, 10% urban, and 40% agricultural (Mount et al, 2014).

Irrigation is one of the widest uses of water nationwide, accounting for more than 60% of water withdrawals in our state (Maupin et al, 2010). Fortunately, the amount of water used in both urban and agricultural irrigation has been reduced through a variety of measures, including a strong trend in the use of precision irrigation techniques (e.g., drip irrigation) (Hanson, 2007). However, research suggests our current unprecedented drought is only expected to get worse; for this reason, it is imperative to be proactive and reduce our water consumption further (Cook et al, 2015).

In a random survey of single-family water customers sponsored by the California Department of Water Resources, results showed that 87% of homes appeared to be irrigating with only 54% doing so in excess (DeOreo, 2015). However, the surveyors also mentioned that most water customers were irrigating at or below average levels. The survey found that 62% of excess usage occurred on 18% of all irrigating lots, leading the surveyors to conclude that "the majority of savings from outdoor use will be found from around 15% of the customers." For this reason, they suggested that a solution to reduce outdoor water usage should be focused on those households which over-irrigate, so that households who are irrigating at appropriate levels are not affected. Of the survey respondents, only 4% were said to be using weather-based irrigation controllers (or WBIC), despite some municipalities offering rebates towards commercially available products. The low levels of adoption surrounding WBIC's is especially concerning given the potential savings; it is estimated that a WaterSense-labeled irrigation controller, or one that meets the EPA's requirements for watering without doing so in excess, can save the average home almost 9,000 gallons (approx. 34,000 liters) of water per year (US EPA, 2017). The EPA estimates that if every US home replaced their sprinkler timer with a WaterSense labeled controller, the potential savings "could save $435 million in water costs and 120 billion gallons (approx. 454 billion liters) of water across the country" (US EPA, 2017).

Over the past 15 years, there have been numerous studies intended to evaluate the reduction of water usage of WBIC's, compared to traditional timers, when retrofitted at an over-irrigating household. Most studies suggest that "savings of 40-50 gallons (approx. 150-190 liters) per household per day, or roughly 10% of total use can be expected from a residential WBIC retrofit program assuming such programs target high water users" (Western Policy Research, 2014)

The aforementioned studies differed in the criterion for targeting over irrigators, citing difficulty in devising a methodology that was effective. However, as WBIC's become more commonplace and more households in general begin to adopt the technology, we can be sure that at least a percentage of these will be over irrigating households and will reap the benefits of "smart" irrigation.

II.2 Pedagogical value

Smart water irrigation is a well-defined topic, and the motivation is clear to the student. The possibility of wielding current low-cost technologies to solve an important problem is very attractive. Also:

1. The student carries out the system development process beginning from the requirement analysis stage through integration of particular logical and physical project's components, up to the final stage of tests and completing the end-user documentation. It should be remarked that the system is a composition of software and hardware parts, so this gives the student the opportunity to work with both abstract concepts (algorithms and methods) and technical issues (assembling physical components and deploying software on them).
2. Preparing a solution fulfilling the requirements such as low cost, usability, etc., is truly a real-life exercise. The student must document the process in order so that the solution can be replicated (and indeed it has been, at all stages, by the instructor and others).
3. The solution has significant market potential if the student decides to develop it. In fact, it would have been an interesting extension of the project if the student provided a "take-to-market" plan.

III. iSprinkle

This section contains an overview of the system. As it is beyond the scope of this article we do not describe in depth the solution details (we will make the full technical solution available online). Instead we present the main components of iSprinkle. Its logical diagram is shown in Figure 1. It consists of three layers representing a user, iSprinkle and the irrigation devices.

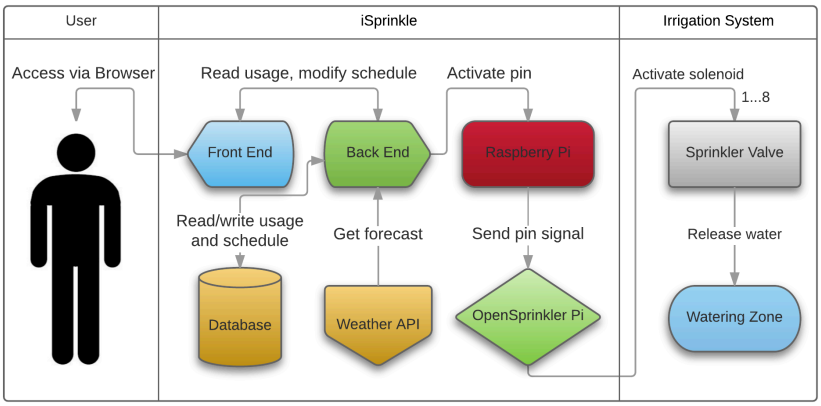

Figure 1. Diagram of the project.

III.1 Hardware components

One of the most important assumptions about iSprinkle was that its cost should be minimized as much as possible. The hardware configuration applied in the project, including Raspberry Pi 2 unit (RPi) as its core element, required about $150 which is an acceptable price for such a solution. Figure 2 shows the pictures of hardware components and Table 1 contains an itemized list of their prices. Note that we do not include the sprinkler system itself, that is the valves, the pipes, the sprinklers, etc., as iSprinkle is only intended to be a replacement for the sprinkler scheduling system.

(a) Raspberry Pi 2 (b) Open Sprinkler Pi

(c) MicroSD card (d) Edimax WiFi (e) 24V Transformer

Figure 2. Pictures of hardware components

Component	Price
Raspberry Pi 2 (RPi)	$39.99
OpenSprinkler Pi (OSPi)	$77.99
Samsung 16Gb MicroSD card (SD)	$7
Edimax USB WiFi adapter (Edimax)	$8.50
24V Transformer (T)	$12.97
Total:	**$146.45**

Table 1. Itemized hardware components

The next step after collecting all hardware items is establishing a software environment consisting of an operating system for the RPi and the necessary software updates, customizations and components. Next, the Raspbian environment had to be configured appropriately.

III.2 Software components

Once all the hardware components are out of the box, we must start by downloading and installing an **operating system** for the RPi. We use Raspbian, a Debian Linux-based OS. In order to keep Raspbian updated with the latest software patches, we need to update the system's package list and then upgrade relevant items. Before setting up iSprinkle, one of the most critical parts of configuration is the RPi's timezone settings; they must be configured to local time so that the watering schedule start times are accurate.

Other configuration options include the ability to remotely access the Raspbian's desktop environment via Virtual Network Computing (VNC), disabling the desktop environment altogether, and modifying the amount of memory allotted to the Graphical Processing Unit (GPU).

The server-side software of iSprinkle is written solely in Python, a versatile scripting language which has been adopted by many learning institutions for teaching programming to beginners. Python is an excellent choice for new and experienced programmers due to its ecosystem. Because Python is open-source, there is a wide variety of freely-available learning materials as well as online communities providing support.

Although Python's standard library is already quite extensive, offering features such as built-in support for networking and interfacing with the underlying OS (Python Software Foundation, 2017a), there are also thousands of third-party libraries available via PyPI, the Python Package Index (Python Software Foundation, 2017b). Due to Python's open nature, users will find it easy to modify and extend iSprinkle's existing codebase.

III.3 Performance

iSprinkle users are able to create, read, update, and delete watering times for stations on the schedule page exposed by a web browser-based GUI. The entire weekly schedule can be seen at a glance, with one station per row and every weekday as a column. Each station can have multiple start times in a day.

When iSprinkle executes, a job scheduler provided by the Advanced Python Scheduler module is instantiated (Grönholm, 2017). As the watering schedule is loaded into memory from disk, new jobs are added to the job scheduler for each station. Each job can be thought of as a tuple which contains the watering function, start time, and watering function arguments, where the latter consists of the station number and watering duration. The job scheduler runs in a separate thread from the main application and executes the watering function when a start time for a job is reached.

When the start time for a job is reached (i.e. a station is due to start watering), iSprinkle retrieves weather data from the past month. The average temperature, along with the current temperature and desired watering duration, are used to produce an optimized watering duration which is then used in place of the user's original value.

IV. Learning Experience

Although this project involves both hardware and software, the learning potential leans greatly on the latter. We will expand on the learning value of the project from the initial setup to the functioning prototype.

IV.1 Web Development

The user-interface for iSprinkle combines HTML, CSS, as well as JavaScript. Bootstrap, a popular framework for developing web sites, is used to easily create stylish pages, many of which can contain components such as buttons, forms, and icons, and are responsive (i.e. adaptable to the viewer's screen size and platform) (Otto,

Thornton, 2017c). Bootstrap provides a gentle introduction to web development in that it makes it easy to create pages which follow a convention rather than spending time on configuration; however, because the documentation is so extensive, users will easily be able to learn about HTML and CSS for both structuring and styling websites, as well as JavaScript, for adding interactivity. However, because Bootstrap only provides structure, style, and a limited set of dynamic features for the user interface, another component is required.

Angular, a client-side JavaScript framework maintained by Google, is used to "extend the vocabulary" of HTML by adding templating, bi-directional data binding, and scope to the traditional static HTML page (Google, 2017d). Users will learn to use "scopes", in the traditional computer science sense, as they declare variables and use logic to add even more dynamic features to web pages. In addition, Angular makes it easy to start learning about asynchronous programming, such as when making requests to a server, be it local (such as iSprinkle's backend) or remote (such as an external API), without affecting the user's experience by waiting for the reply. Finally, JavaScript itself has seen enormous changes within the last decade; countless web frameworks have been built upon it, making it an enormously versatile language to learn.

IV.2 Software Engineering

The design of a system such as iSprinkle requires a holistic approach that is very different from most class assignments. The former usually span a few files that are to be turned in within a week or two, making it difficult to implement a system with many “moving parts.” However, iSprinkle's functionality is divided between the front-end and backend, both of which need to communicate so that the user's requests are fulfilled. Designing such a system requires taking into consideration many aspects; from major decisions such as selecting a backend language to use, to minutiae such as the date and time formats to use across the backend and front end to maintain consistency.

V. Conclusion

This article presented a senior capstone project aimed at preparing a low cost, open source-based sprinkler timer capable of performance adjustment on the basis of weather forecast data being gathered in the background. The project covered assembling both hardware and software components. The latter ones were based on open source solutions and technologies: the Raspbian operating system, Python-based components, Angular, Bootstrap and others. The project required the student to develop skills in several areas: assembling together all hardware items; installing and setting up the OS and the software environment; programming using advanced web technologies. As mentioned in Section III, the basic assumption was reducing the cost of the solution. Meeting this requirement and relying on the open software, make iSprinkle to be easily replicable. This fact together with the environmental context, i.e. the common water shortages in California, open good prospects for commercial application of iSprinkle.

References

United States Environmental Protection Agency. (2017). *WaterSense Labeled Irrigation Controllers.* WaterSense. US EPA. http://bit.ly/2i7pSeH. (Accessed on 06/01/2017).

COOK B. I., AULT T. R., SMERDON J. E. (2015). *Unprecedented 21st century drought risk in the American Southwest and Central Plains*. Climatology, February 2015.

DEOREO W. B., MAYER P. W., MARTIEN L., HAYDEN M., FUNK A., KRAMER-DUFFIELD M., DAVIS R. (2015). *California Single Family Water Use Efficiency Study*. Aquacraft. http://bit.ly/2qDJ0Wo (Accessed on 06/01/2017).

Python Software Foundation. (2017a) . The Python Standard Library – Python 3.5.2 documentation. http://bit.ly/2rnuNdm, September 2016. (Accessed on 06/01/2017).

Python Software Foundation. (2017b). *PyPI - the Python Package Index*. https://pypi.python.org/pypi, 2016. (Accessed on 06/01/2017).

OTTO M., THORNTON J. (2017c). *Bootstrap – The World’s Most Popular Mobile-First And Responsive Front-End Framework*. http://getbootstrap.com/. (Accessed on 06/01/2017).

Google. (2017d). AngularJS. A Superheroic JavaScript MVW Framework. https://angularjs.org/. (Accessed on 06/01/2017).

HANSON B. (2007). *Irrigation Of Agricultural Crops In California*. Technical Report, University of California Davis.

MAUPIN M. A., KENNY J. F., HUTSON S. S., LOVELACE J. K., BARBER N. L., LINSEY K. S. (2010). *Estimated Use Of Water In The United States In 2010*. Circular 1405, United States Geological Survey.

MOUNT J., FREEMAN E., LUND J. (2014). *Water Use In California.* Technical Report. Public Policy Institute of California.

Western Policy Research. (2014). *Weather Based Irrigation Controllers*. http://bit.ly/2qDSSeY. (Accessed on 06/01/2017).

Grönholm A. (2017). *Advanced Python Scheduler* – APScheduler 3.3.1 Documentation. http://bit.ly/2rg8bwW. (Accessed on 06/01/2017).