Smart Water Irrigation System

Civil and Environmental Engineering 186:
Design of Cyber-physical System
Professor: Scott Moura
GSI: Eric Burger

by Siyu Chen, Nicolas Fatras, Haoran Su

Abstract

The smart water irrigation system developed by our team is an adaptive plants and crops irrigation system. The purposes of our smart water irrigation system are to provide a water delivering schedule to the crops to ensure all the crops have enough water for their healthy growth, to reduce the amount of water wasted in irrigation, and to minimize the economic cost for the users. Our system takes in real time data of the water content of the plant as input argument, combines it with other parameters such as water cost schedule and precipitation on the crop field, runs the designed linear optimization system periodically and outputs the most efficient amount of water the plants need, which is translated by a specific actuation time of the water pumps. The linear optimization system, which is essentially the brain of our system, is able to make decisions for the users when to distribute water into the crops fields and how much water should be delivered. Given the number of factors to take into account and the different crop requirements to take into account for each type of plant, this problem because much too complex to solve through simple management methods and has to be supported by automated systems such as the one provided by our group. In the droughty California nowadays, utilizing our smart water irrigation system not only supports the environmental sustainability of the regional area, but also significantly lowers the expense of water usage for the farmers.

Introduction

Motivation and Background

California agriculture's revenue occupies 12% of the total revenue of agriculture of United States (1). At the same time, the agriculture in California consumes 80% of water usage of the state (1), not including some other water activities such as groundwater extraction which may potentially increase this figure to a higher level. The drought in California has lasted for four years, which already causes huge economic loss to this market. Farmers have generally changed the types of crops which consumes less water to maintain their business. While if there is not an ultimate solution to increase the irrigation systems in California now, there is going to be a catastrophe on California's agriculture market.

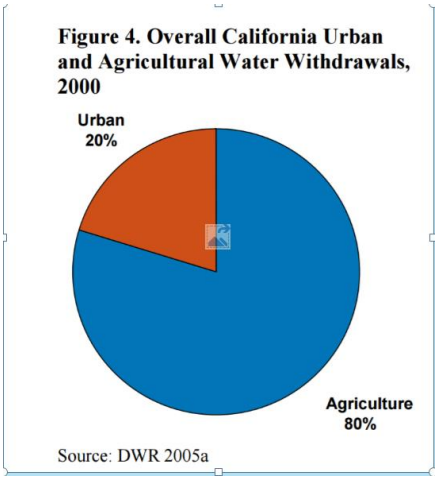


FIgure 1: Distribution of water consumption in California

Therefore, our team, all as future civil engineers, developed this smart water irrigation system to save the water sustainability of the regional area, to maintain the crop fields environmental friendly by preventing soil and earth from getting flooded or dried, and, most importantly, to save economic cost of water usages for the farmers and for the whole market. While several different environmental factors play a key role in agricultural productivity, this project focuses purely on the water consumption aspects of agriculture. The project is a scaled version, as much in size as in complexity, of what agricultural estates experience on a daily basis.

The three of us are all taking the course CE191-analysis of civil engineering systems, and all of us totally understand the importance of a linear optimization system in every civil engineering system. Particularly, water system is a very crucial system for human society and the ongoing water shortage is the global and prevalent. Therefore, we decided to develop a linear optimization as the core of the smart irrigation system.

With water being such a crucial focus of Californian agriculture nowadays, several different water economization techniques and procedures have been put into practice already. Drip system irrigation is becoming more and more generalized and drought resistant crops are starting to replace too heavy water-consuming plants. However, following a study from the Pacific Institute, the method enabling the largest amount of water saving is through the installation of smart irrigation systems, as shown in the graph below (from a study of the Pacific Institute):

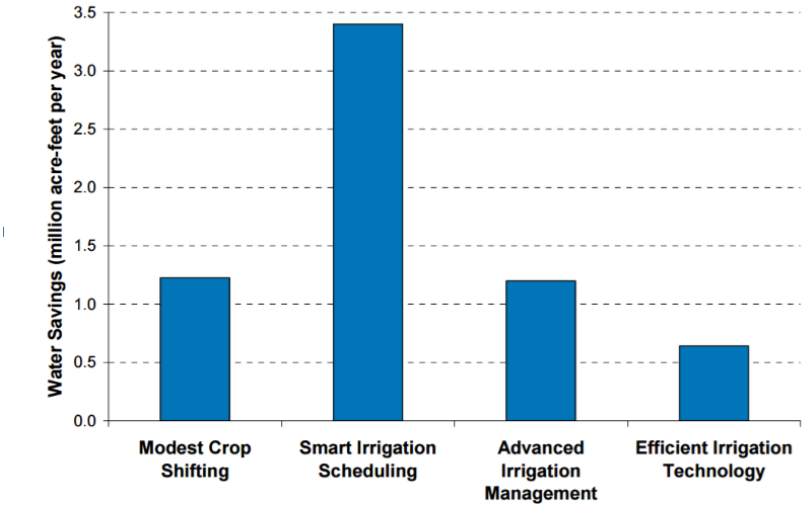


Figure 2: Water savings by scenario

The modest crop shifting is able to save approximately 1.25 million acre-feet per year, the advanced irrigation management is able to save about the same amount of water and the efficient irrigation technology will also save around half a million acre-feet per year. However, the smart irrigation scheduling, which is essentially the concept smart water irrigation system is

adopting, can save more the total of all these alternative irrigation methods. This is also the reason our staff would like to increase the irrigation system efficiency using the smart irrigation scheduling instead of other irrigation methods or concepts.

Focus of the study

The focus of the smart water irrigation system, using the linear optimization designed, is to provide a comprehensive water delivering schedule which could both ensure that plants could have enough water for their growth and costs the minimum money for the water usage. The focus of the study is also trying to make all the information visual to the user and take in user defined input as new parameters into the linear optimization system.

Before diving into the principles of optimization theory, it is important to understand the principles on which we built our model. While the system is trying to use water wisely, it is not trying to reduce the amount of water used by the crops. Plants have specific water requirements in order to grow efficiently and in a productive way. Reducing water consumption by cutting productivity is opposite to the farmer's interest and shouldn't even be an eventuality. Hence we focus on delivering water optimally in the sense that it will cost the farmer the least amount while avoiding any unnecessary loss of water.
The interesting theoretical assumption behind our model is that the price of water varies with the amount consumed. At peaks consumption times, the price is higher while water is cheapest at troughs. This is an analogy to the electricity supply network where cost varies directly with usage due to the input of multiple components of the energy mix such as renewable energy. In our case, cost of water is mainly decided by the farmers themselves since they represent 80% of the state's water consumption. Hence when they all water their crops at the same time, the cost of water is going to be the most expensive, which is quite inconvenient for the farmers.

In our model, prices are based on the average price of 0.0012*10^-3 dollars per cm^3, which is common in California. This rises up to 0.0018*10^-3 $ at peaks and goes down to 0.0008*10^-3$ at troughs. While this price variation is fixed for each day within our model, the next step would be to have a dynamic price function which adjusts to the consumption changes of the farmers. While this whole model might seem at first like an unrealistic theoretical assumption, important hidden costs lie behind this variation in consumption. Most importantly, high water demands at specific times require large infrastructure with large dams, bigger canals and important water storage facilities to cope with this important variability.

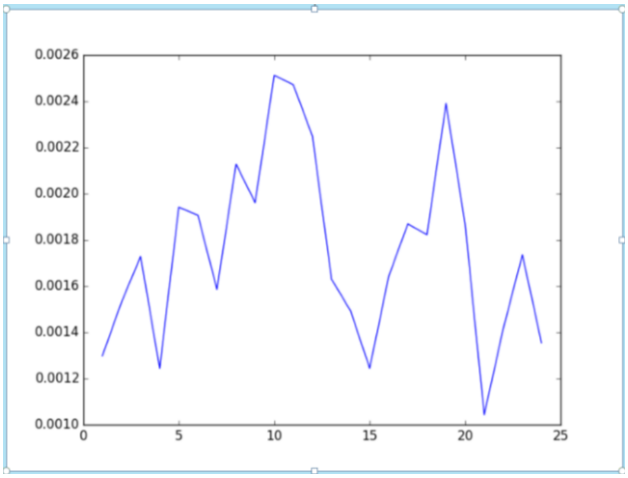
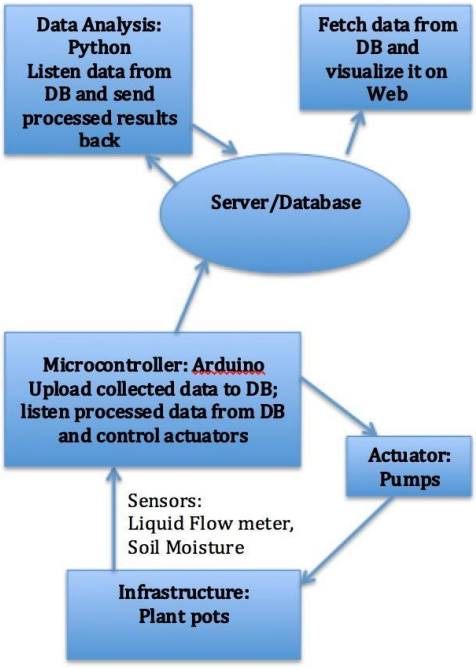


Figure 3: Price variation of water throughout the day

Hence, if farmers adopt this optimization system, benefits are on both sides. The system should help the farmer use water at cheapest times, and hence save a considerable amount of money, while the water supplier benefits from this decrease in variability. Indeed, if our program incites to consume at troughs while avoiding peaks, the general consumption trend should be evened out and make the water supply more uniform over the day.

Additionally, if the watering constraints are satisfied, the productivity of the crops will increase as the crops are in favorable growing conditions, while less water will be wasted by over-saturation and water percolation. The plant/water ratio is increased through this increase in productivity, which in the end is how a lot of water can be saved.

Figure 4: Schematic of CPS:



The architecture of our cyber-physical system starts from the infrastructure, which is the crop plants. In the crop field, there are two types of sensors: the soil moisture sensors and the liquid flow meters, which will be elaborated in the Hardware Components. The soil moisture sensors generally tells the current water content of the crops, and the liquid flow meters will tell the amount of water has been delivered. An additional advantage of the flow meter is that it enables to detect leakages in the irrigation network and consequently fix the problem quickly. The data collected by these two kinds of sensors will be uploaded to the database, then the linear optimization system will run the model to give the optimized solution and send the data back to the database. Now, a web visualization model will be established and all the data from the database will be presented as plot on the dashboard. Users are also able to change some of the parameters, such as the soil profile, to modify the linear optimization constraints. The database, now containing the optimized solution, will send data back to the actuator, which is the pump in the crop field, to distribute and deliver water as expected.

Technical description

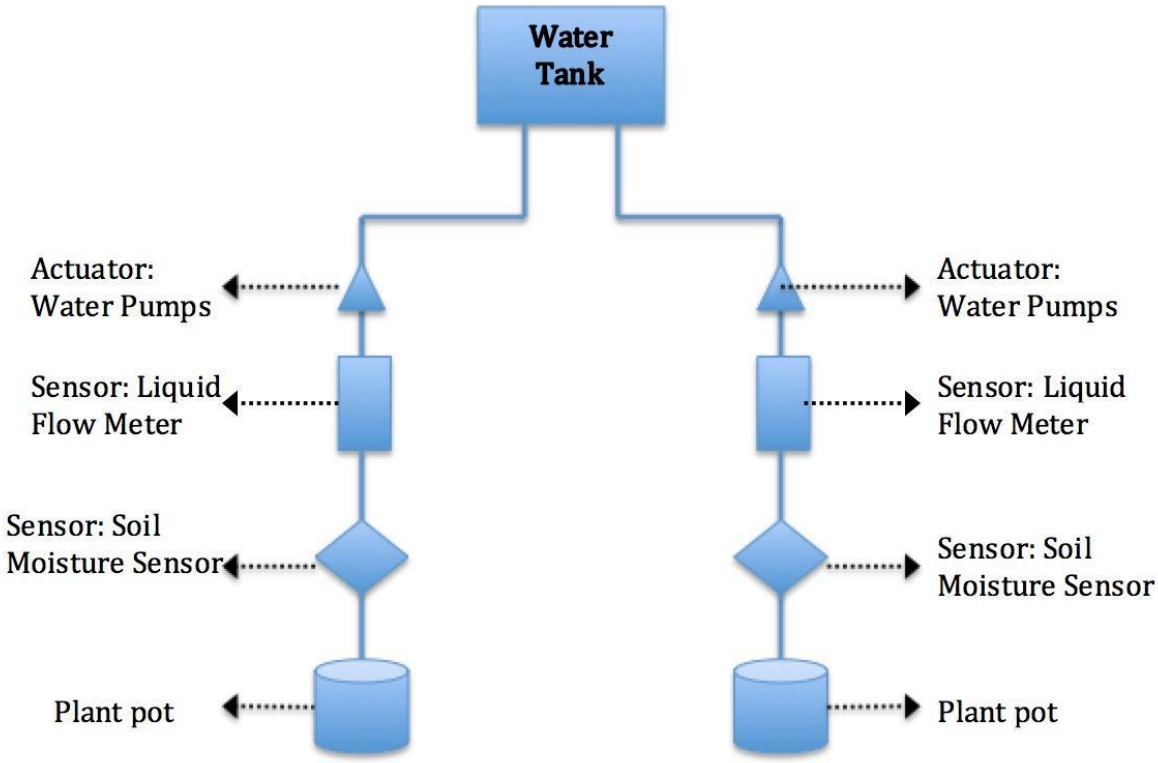


Figure 5: Hardware Components:

- **2 Arduino Uno**: Arduino is our microcontroller, which can receive the data collected by sensors and send signals to control the actuators. It's also connected to computer, which can send data and receive data from computer.
- **2 Liquid Flow Meters:** Liquid Flow Meter sits in line with our water line, and uses a pinwheel sensor to measure how much liquid has moved through it. The sensor comes with three wires: red (5-24VDC power), black (ground) and yellow (digital pin). By counting the pulses from the output of the sensor, we can easily track fluid movement: each pulse is approximately 2.25 milliliters.
- **2 SparkFun Soil Moisture Sensor:** Soil moisture sensor has two large exposed pads which function as probes for the sensor, together acting as a variable resistor. The sensor has three wires: read (3.3V-5V power), black (ground) and yellow (analog pin). The more water that is in the soil means the better the conductivity between the pads will be and will result in a lower resistance, and a higher SIG out.

By calibrating, we find that when the soil is saturated the water percent is around 40% and the soil moisture data is 880; when the soil is drought enough that plant cannot absorb water from it, the water percent is around 20% and the soil moisture data is 525. To calculate how much percent of water can be absorbed by plants, we use the following formula:

$$percent = \frac{sensorreading - 525}{880 - 525}$$

Then using measured plants depth data (cm), soil area data (cm^2) and total plant available water data (cm/m) (tpaw gets from UC Davis soil web), we can calculate what current water amount can be absorbed by plants is. The formula is:

$$plantwater = percent * area * depth * tpaw/100$$

- **2 Mini DC12V Micro Brushless Water Oil Pump 240L/H 5W:** Water pump is used to pump water from water container. Because it requires 12V power and Arduino can only provide 5V power, we use a relay and a 12 VDC adapter. We connect water pump to a 12 VDC adapter first and then connect the adapter to relay. The relay is connected to the normal AC outlet. This can prove water pumps enough power.
  Water pump is also our actuator. We control it through controlling relay. The relay can receive digital signal and has 2 pins: red (digital pin) and black (ground). We connect the relay to Arduino so Arduino can control the relay and the water pump.
- **2 Plant pots:** This is our infrastructure. One pot contains grass; another one contains a follower called impatiens. Both of them have Soil Moisture Sensors inserted so we can know what the current moisture content is.
- **2 Laser Cut Cases:** One laser cut case is used to contain two liquid flow meter sensors. Its dimension is 23cm*20cm*8cm (D*W*H). Another one is used to contain two Arduinos. Its dimension is 25cm*25cm*10cm (D*W*H). These cases can make our system look like cleaner.

Software code outline:

- Arduino Code:
  - PumpCon.ino: It can receive the data collected by one Liquid Flow Meter sensor and one Soil Moisture sensor. It also can receive the data sent from Serial in order to control one water pump. Because we use two Arduinos, we just connect both to the same laptop and upload the Arduino codes to each of them respectively by changing ports.
- CloudLPApp: The Cloud Optimization Solver is written by Eric Burger, which is a web application capable solving simple Linear Program. We interact with the app according to the RESTful Web API below.
- DBNanoServer: This nanoserver is also written by Eric Burger, which is core of software part. Its main function we use is to store data into database or select data from database.
- Python files:
  - updefaultdata.py: The purpose of this script is to post default data into database. Default data includes: ideal water consumption of plants during normal day data, grass daily water demand over 60 days data and impatiens water demand over 60 days data.
  - Listenandsend.py: The purpose of this script is to listen the raw data sent from two Arduinos and store it in the database. It also listens to the processed results from database and sends them to two Arduinos to control two water pumps. Its skeleton is the same as the script in Lab 5. The main functions are:

        - Senddata: send the grass moisture, impatiens moisture, cumulative water provided to grass and cumulative water provided to impatiens data to database.
        - ListenData: listen the processed results from database. The processed results are water amount needed to provide to grass in the next one hour data and water amount needed to provided to impatiens in the next one data.
        - Processreult: calculate how much time should water pumps keep running. It uses water amount divided by water flow rate.
    - Listenandprocess.py: The purpose of this script is to listen the raw data stored in database and use it to run optimization model. After optimization model done, it will send the results back to database to store. Its skeleton is also same with the script in Lab 5. The main functions are:
        - Listen: listen the grass moisture data, impatiens moisture data, water consumption percent during normal day data, grass water demand data, impatiens water demand data from data base.
        - WeatherRequest: get next 4 hours' precipitation amount from http://www.yr.no/place/United_States/California/West_Berkeley/hour_by_hour.html website.
        - Optimization: our optimization model function. It uses inputs, like grass and impatiens moisture data, water consumption percent data, grass and impatiens water demand data to build an optimization model. Then it sends the built up matrices to Cloud Solver API to solve it.
        - SendData: send the cost of irrigation, water needed to provide to grass and water needed to provide to impatiens data to database.
    - Casestudy.py: The purpose of this script is to compare the total cost of normal irrigation way and the total cost of our smart irrigation system to irrigate 1 acre grass one day.
- Web Visualization files: Most of them are same with files in Lab 5. But we did some changes to JavaScript file.
    - nanodashboard.html: it creates our web page based on Bootstrap Dashboard. This file is almost same with the one in Lab5. But we add one more sidebar called Parameters, which has a link to UC Davis SoilWeb and can let users to input parameters like plant depth, soil area and total plant available water amount.
    - nanodashboard2.js: it is based on dashboard.js in Lab 5. The first main function of it is to load page content when you click on different sections of the web page. The second main function is to make plots according to the data in database. The last one is to post the user input from a form to the nanoserver and then the nanoserver will store it in the database. The main functions are:
        - loadContent: load all page content containers, including Overview, Parameters, Files, Plots and Export.
        - loadData: load all data from the NanoServer API.
        - parseNetworkData: parse the network data and load into content containers.
        - loadStreamIco: load a stream icon in data-select div
        - loadStreamPlot: load a stream plot in content div.
        - reloadPlotAndExport: reload the Plot and Export displays by retrieving most recent data from db server.

Optimization part:

To run the optimization code, we call a web API through python which solves the program for us according to the objective function and the constraints we pass to us. This requires that we use a linear program, with a linear objective function and linear constraints.
For the rest of the explanation, we will refer to the variables shown below to set up our cost function and constraints:

Symbol	Meaning
x	Decision variable, which is the unknown we are running this program for. Its value depends on the objective and the constraints. The first 4 elements represent the amount of water in cm^3 delivered each hour to the first plant, the next four the amount delivered to the second type of plant and so on
C	Cost coefficients (price in dollars per cm^3, varying with time.
W	Vector of rain prediction in next hours
S	Water need of the plant initially specified by the user and updated by the adaptive code
k	Soil coefficient representing capacity of soil to retain usable water
M	Actual moisture content (in cm^3) of the soil measured by soil moisture sensors
A,h	Dimensions of the agricultural surface considered (area and depth to the roots)
Tpaw (Total plant available water)	Ratio of maximum extractable volume of water per volume of soil

Figure 6: tables of variables used in optimization program

While the concept of the cost function has all of the subtleties mentioned above, expressing it mathematically is pretty straight forward. A vector of cost coefficients C, representing the cost per cm^3 of water for each hour, is multiplied by the objective function x which represents the volume of water in cm^3 supplied to the plant for each hour. The optimization code is run every four hours, for the reasons explained below once the constraints have been presented. This means that the length of x is 4 times the number of different plant types grown, which in our case is 2. So c and x are both 8*1 vectors.

Setting up the constraints is more challenging for several reasons. First of all, satisfying the plants needs require constraints in irrigation time and volume which can be hard to express correctly in a mathematical way. For example for grass, to avoid water waste through percolation, the minimum value of the decision variable vector has to be at least half the maximum value in the decision variable. This is to avoid that the program simply decides to apply all of the water at the cheapest hour, which the soil obviously wouldn't be able to handle, and would create a lot of water waste.

However constraints relating the minimum and the maximum are not linear in nature, which can't be solved by the web API. It is thus necessary to translate this into linear inequalities. One solution we opted for would be to have all variables for that plant be bounded in this way: 0.2 <xg(i)<0.4 for i from 1 to 4. While this is an example, many other requirements had to be translated from the physical to the mathematical world. The fully set up linear program is finally shown below, for the two chosen plants, grass and impatiens:

Objective function:

Min $c^T \cdot$ x

s.t W+kM+$\sum x_i + x_g \geq$Si + Sg

ximin ≥ 0.5 ximax

xgmax ≥ 0.7 sum(xggrass)

xgmin≥0.1 xgmax

It is to be noted that while the two different types of plants have different watering requirements and hence separate actuation systems, a single optimization program is run for all the plants together, so that water consumption is optimized over the farm as a whole, which is what makes this problem interesting since solving this problem empirically becomes too complex otherwise.

The first inequality constraint of the above equation implies that the total amount of water available to the plant for four hours has to meet the plant's requirements S for that time interval. The available water is comprised of several factors:

- W is the volume of water acquired from precipitation. This information is acquired using a GET request to a weather forecast website showing precipitation in mm per hour. This information has to be multiplied by the area of the field considered, inputted by the user on the visual interface, to get a water amount in volume.
- M is the volume of water present in soil, to which a coefficient k is applied depending on the useful water withholding capacity of the soil. Again, this information is obtained from

the web visualization input. The method used to determine the amount M in volume units is explained in the section on hardware specifications. If the value of M measured is higher than the needs of that plant at that time, the optimization code will just output a value of 0 for the amount of water to supply.

- $\sum x_g + x_i$, which corresponds to the total amount of water supplied by the pumps over the optimization time interval considered.

Here xi represents the part of the decision variable for the impatiens flower and xg represents the part of the decision variable for the grass. In addition to these water volume constraints, time constraints are also applied through the graphs of water consumption over time shown below:

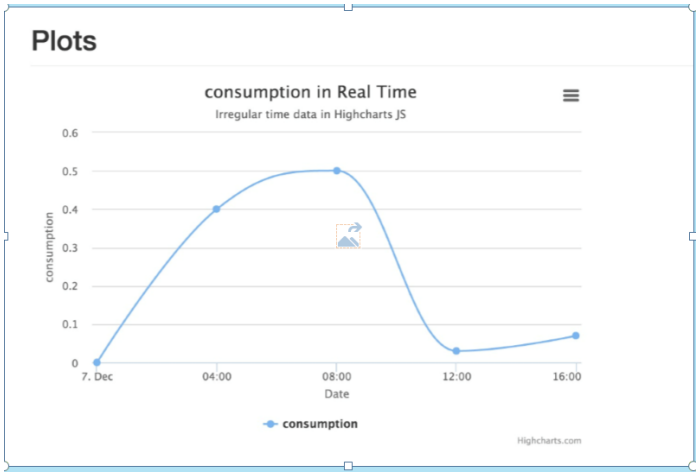


Figure 7: Plot of ideal water consumption of plants during normal day

Plants will tend to be extracting water at higher rates during specific times of the day. Plants also need to have a certain quantity of water over several days and have a water application pattern alternating between highs and lows. An example can be shown in the graph below:

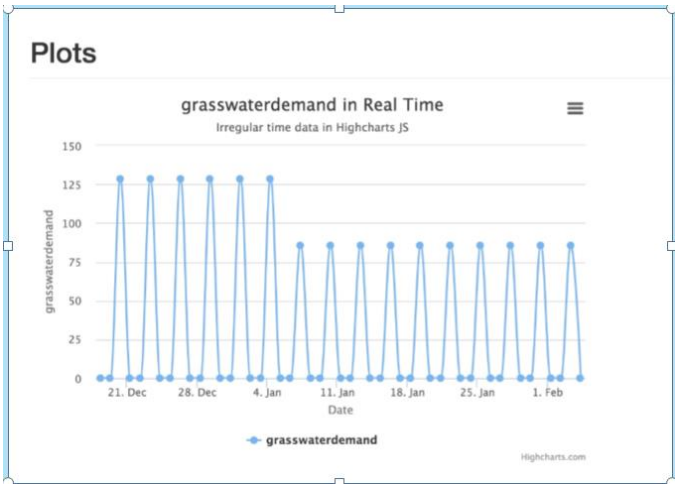


Figure 8: Variation of water needs of grass over several days

Setting up all of these constraints is where the human interaction with the cyber-physical system becomes crucial. Each plant will require different values and parameters. One could say however that this can be too complex and time consuming for the user. Moreover, he could not know exactly what amount of water his plants need to grow optimally. Our system is trying to account for this by having adaptive constraints depending on the information sent back from the moisture sensor. When looking at the graph above, grass has a 3 day cyclic irrigation pattern, while impatiens' irrigation cycle is every 48 hours. Hence for each plant, the moisture from 72 (or 48 depending on the plant) hours ago will be compared to the present moisture. If the difference is too big, adjustments to the water needs of the plant are made, as shown below:

$$\textbf{If } M_{t+72h} \geq 1.1 * M_t * \frac{S_{t+72}}{S_t} \quad \cup \quad M_t \geq 0.8 * M_{sat}$$
$$\Rightarrow \quad \mathbf{S \text{ -> } S/1.2}$$

$$\textbf{If } M_{t+72h} \leq 0.9 * M_t * \frac{S_{t+72}}{S_t}$$
$$\Rightarrow \quad \mathbf{S \text{ -> } S*1.2}$$

Finally, let us go back to reason why the optimization code is running every four hour, instead of simply once for the whole day. The amount applied varies depending on the values read by the moisture sensor. In a 4 hour time interval, enough changes have had the time to occur to affect the moisture of the soil. This means that the value of M measured previously is not valid anymore for the x values at later times and a new measurement of M has to be made. Factors affecting soil moisture can be water runoff or evapotranspiration due to the plant itself and the sun. Indirectly, solar radiation can hence be determined by comparing rates of soil moisture change for a given time interval. The soil moisture sensor is hence the only indicator of the "well-being" of the plant, and we send moisture values every second to the database.

Web visualization:

While the optimization system is running automatically in the background, the user would probably like to supervise his irrigation system to then be able to implement decisions accordingly. Indeed, a farmer probably would like to know at what point of the day he is consuming the most amount of water, how much money he is comparatively saving and whether his irrigation network suffers from leakages or any technical problem. He could then decide to grow plants which avoid these peaks in consumption and consequently cost him less to grow. In general, it is important to keep in mind that while the human is not involved in the actuation, he is the primary decision maker, thus the system should integrate the human aspect as good as possible to work efficiently. A user interface has hence been setup using html and

Javascript in a Bootstrap framework, of which a screenshot can be seen below:

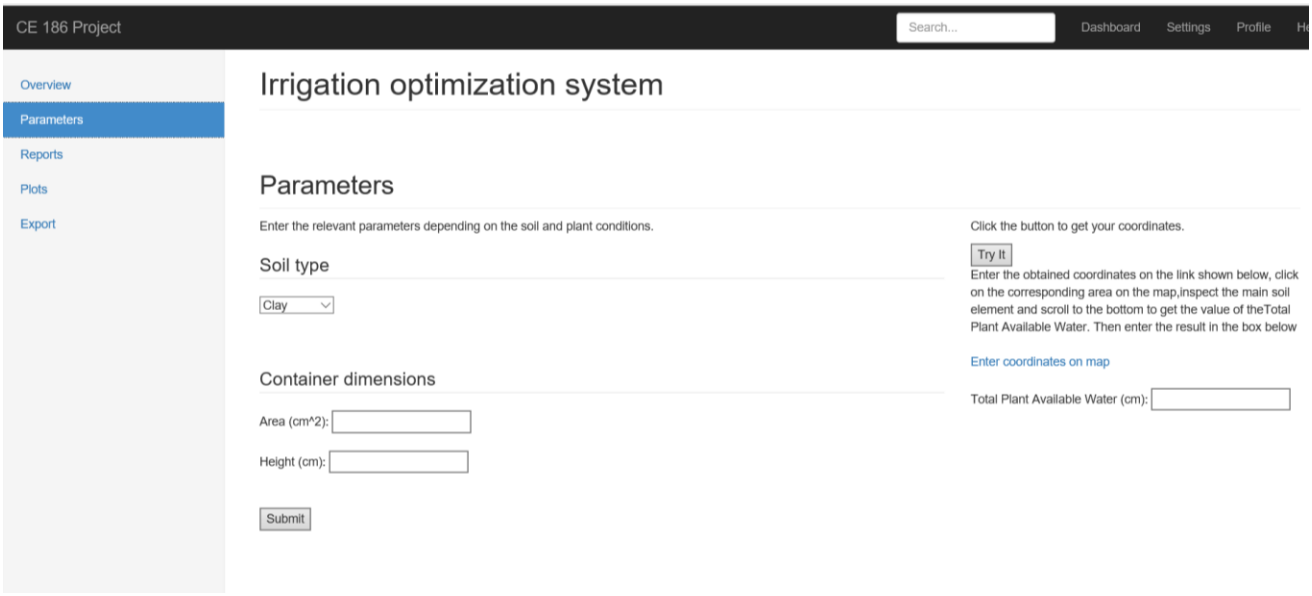


FIgure 9: Parameters input page used to set up part of the constraints

While the Parameters section is mostly useful to set up the initial constraints, the plots sections shows the evolution of total water consumption, actuation times of the pump for the different plants and other variables of interest. The information of the form setup with html in the parameters section is then sent to the database to be analyzed by the Python code. In our code, information is sent with an AJAX jquery command in the Javascript code of the web page, and posted on the local url with the key “process”. This data is then accessed by the Python script nanoserver.py through the “process” key with a GET request and then passed further to the SQLite database by connecting to it, opening a new table and populating the rows with the values just received.

To help the user during the setup of constraints, a useful feature figures on the right hand side. When clicking on the “Try it” button, the user gets his current coordinates. He can then enter these in the link provided below, which will take him to the UC Davis SoilWeb website at the coordinates he entered. There information on the dominant type of soil is available with parameters of interest such as the total plant available water which he can then input into the form.

Hence the visualization page is communicating both ways between the system and the user. This makes it a crucial part of the system as a whole, especially since it is the only visual indicator of the efficiency of the system, apart from the state of growth of the plants.

Presentation and analysis of the data collected

The following plots are plots generated by JavaScript according to the data in database. Because we have run our system several times during the showcase, there is a smooth line without any points on it in the middle of all plots. We can just ignore it and focus on the two ends. In reality, if our system keeps running it will not produce such data discontinuity.

- Grass moisture content data: This is the raw soil moisture data collected by soil moisture sensor in grass pots. Recent data is plotted in the following graph. The left end data values are around 826. The right end data values are around 843. The soil moisture sensor reading for saturated water content is about 880 so we can say that the grass pot water content on right end is close to the saturation, which is due to the fact that the plants were repetitively irrigated during the exposition at a rate far beyond their intake capacity. If the code had been left to run a lot longer, the adaptive system comparing values from 3 days ago to the current values would have diminished the supply constraints in order to account for this oversaturation. We can also see that the moisture doesn't keep increasing up to complete saturation because the supply constraints were satisfied by the moisture alone so that no extra water was needed to be provided, hence not increasing the moisture any further.

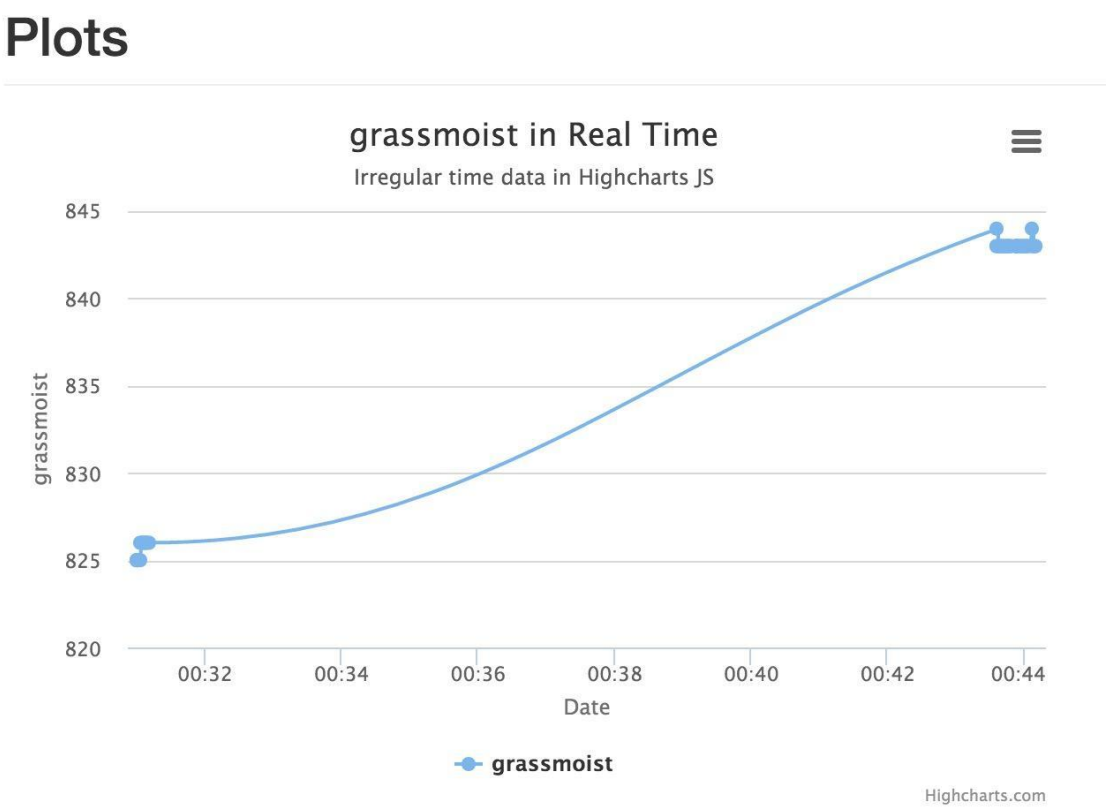


Figure 10: Evolution of grass moisture over time

- Cumulative water volume provided to grass data: This is the raw data sent from the liquid flow meter sensor, which sits in line with the water pipe connected to the grass pot. For the left end we can see that the cumulative water amount provided to grass is around 0.03 liters. For the right end we can see that the cumulative water amount

provided to grass is around 0.02 liters. The reason that there is still water provided to grass when the grass moisture content is close to saturation is because on showcase we want to show demo to people so we didn't run Listenandprocee.py, which means we didn't run optimization model and let actuation part code listen to historical processed results of optimization model so there was water provided to grass even though there was enough water for grass.

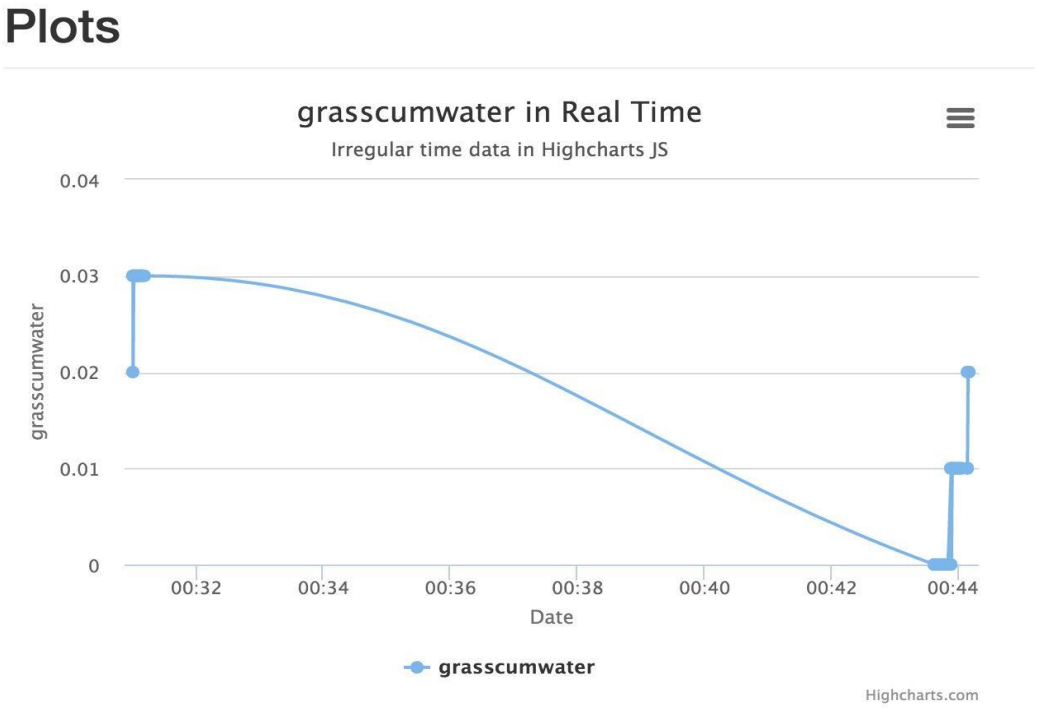


Figure 11: Evolution of cumulative water provided to grass over time

- Impatiens water content data: This is raw soil moisture sensor data of impatiens. Same with grass moisture data, the impatiens moisture content is also close to saturation.

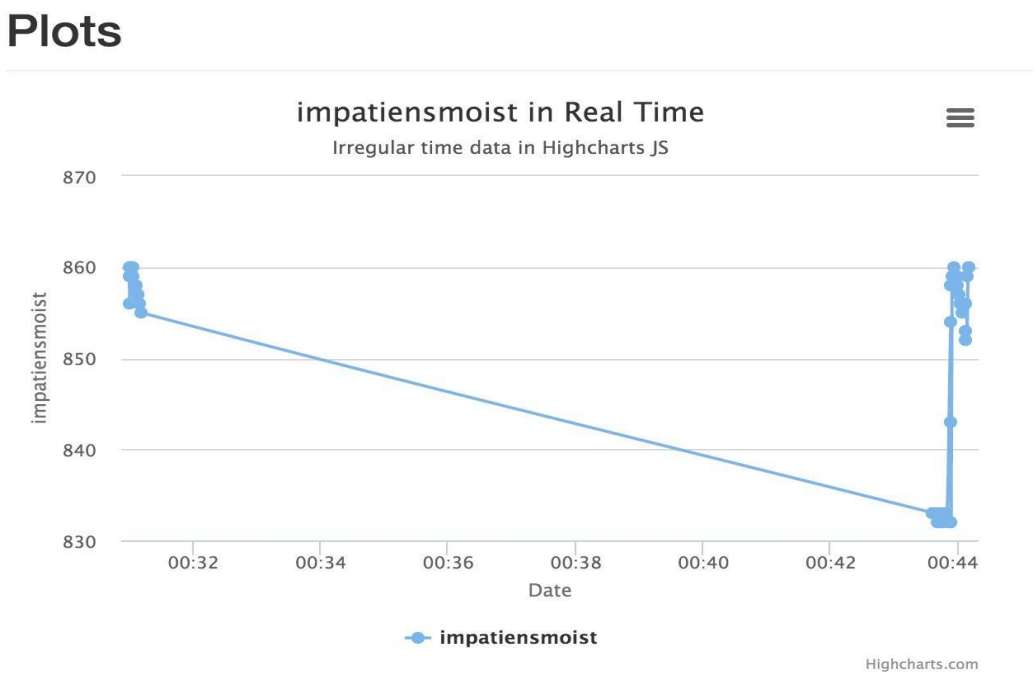


Figure 12: Evolution of impatiens moisture over time

- Cumulative water volume provided to impatiens data: This is the cumulative water volume data recorded from liquid flow meter sensor. Same with cumulative water volume provided to grass, the cumulative water is increasing because we use historical optimization results to control actuators.

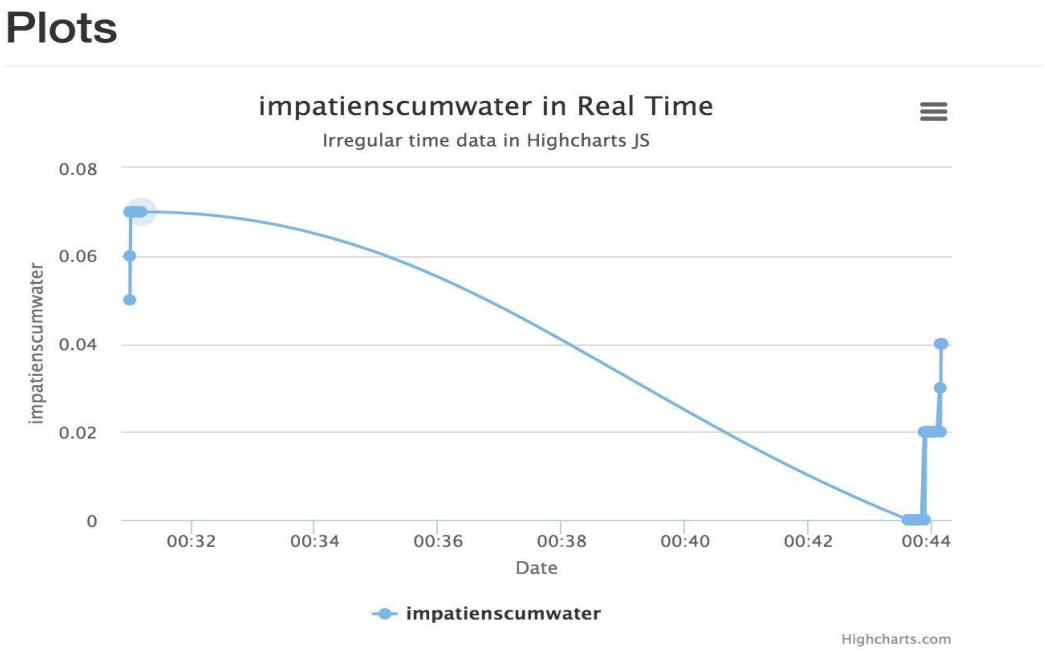


Figure 13: Evolution of cumulative water provided to impatiens over time

- Precipitation data: This is the next 4 hour precipitation amount forecast data get from weather forecast website. As we can see, the precipitation forecast amount is always 0 when our system is running.

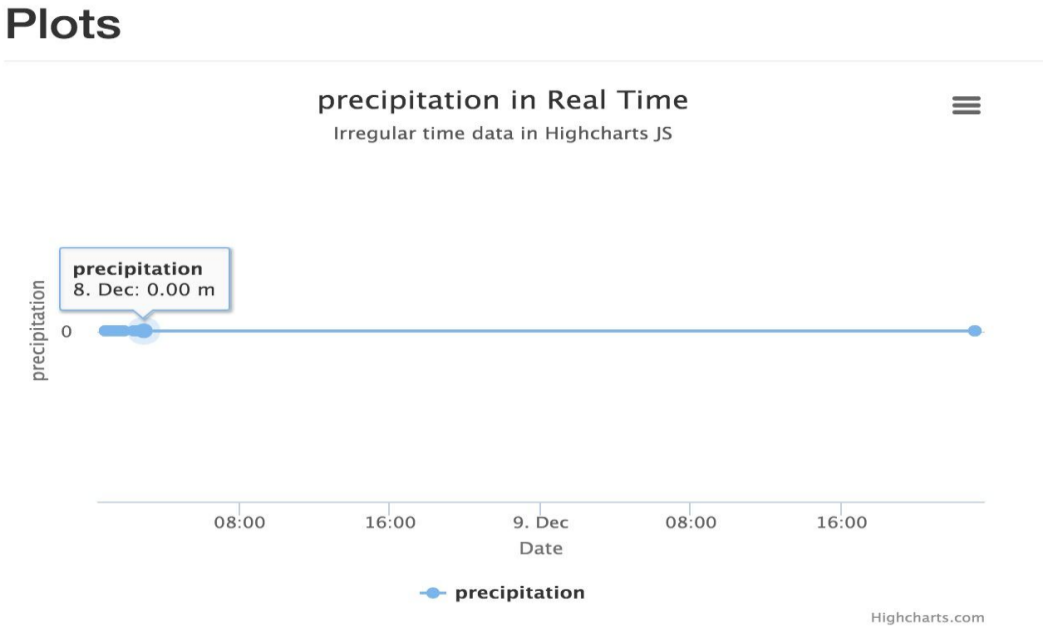


Figure 14: Evolution of grass moisture over time

- Water provided to grass: This is one of optimization model results. It represents how much water we need to provide to grass in the next one hour. Since we didn't run optimization model on December 9 (showcase day), the following plot is just based on December 8 data. Combined with grass moisture data on December 8, we can find that the optimization model decides to provide some water to grass when the moisture of grass pot is around 825. Since our optimization model gives us 4 water volume results for next four hours one time, we can see there are four points at the end of plot. There is a peak because the water cost at that hour is low so we want to provide most of water to grass at that time. The trough shows that water cost at that hour is high and optimization model decides to provide less water. The reason why we don't provide all the water when the water cost is low because we don't want our plants overcommit water in a small time range.

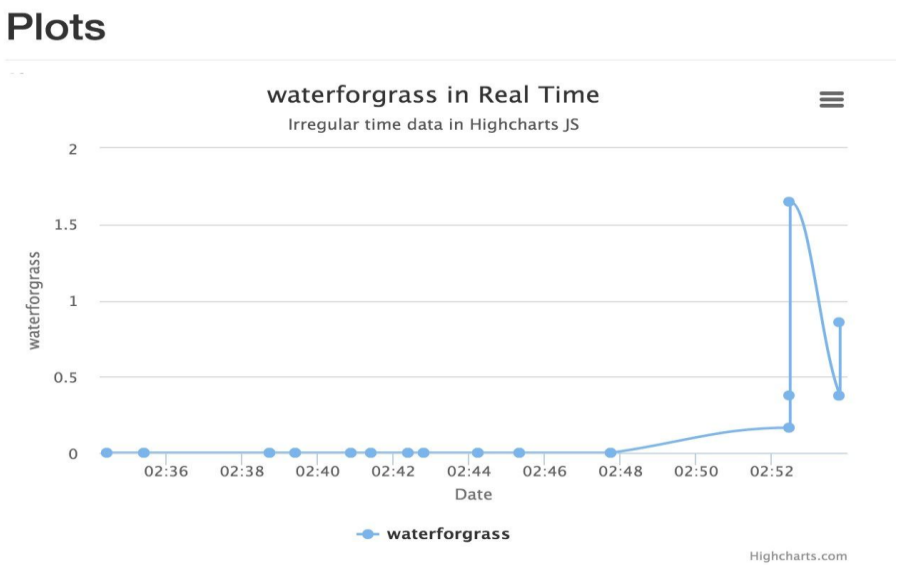


Figure 15: Evolution with time of water applied to grass

- Water provided to impatiens: This is one of optimization model results. It represents how much water we need to provide to impatiens in the next one hour. Same to water for grass data, this plot is plotted based on data on December 8. Similar reasons, the peak represents water cost at that hour is low so we want to provide more water; the trough represents water cost at that hour is high so we want to provide as less as possible.

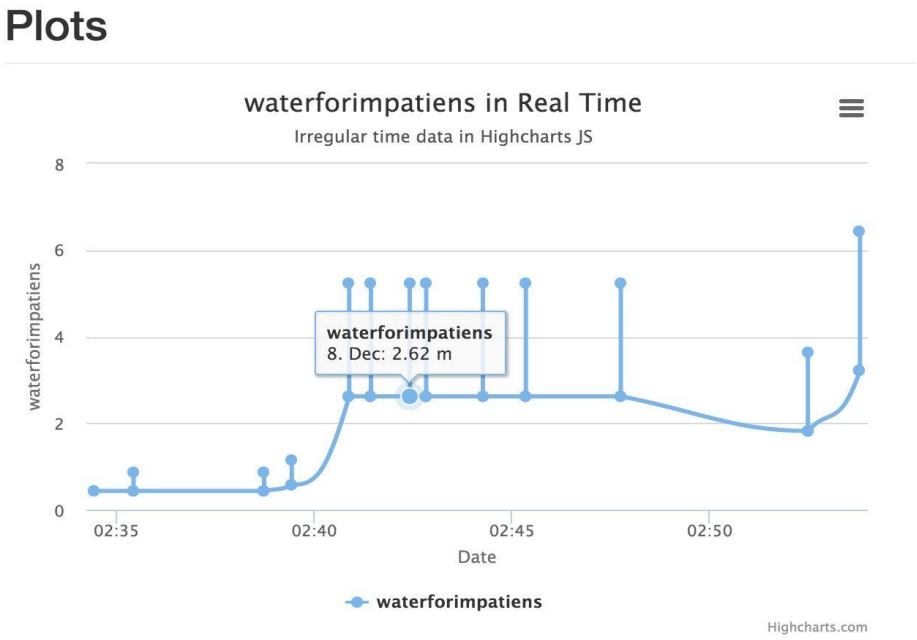


Figure 16: Evolution with time of water applied to impatiens

- Cost: The cost is the total water cost prediction for grass and impatiens in next 4 hours. This is also plotted based on data on December 8 but it also contains historical data. We once deleted data table of water for grass and water for impatiens. We can just focus on the end part because the time for that part is consistent with water for grass

and water for impatiens. As we can see, the cost is stable around 02:30 because at

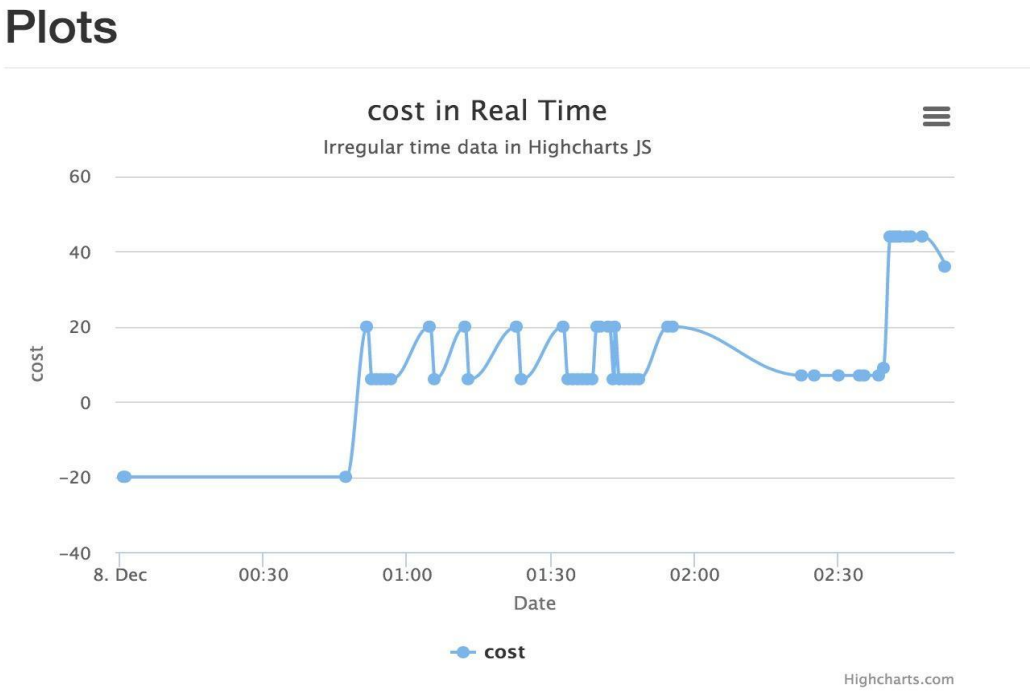


Figure 17: Evolution of money spent each hour (in dollars)

Case Study

To see the difference by using the smart water irrigation and using the normal way of irrigation, which is watering the plants twice a day, a quantitative comparison analysis is conducted to help see the advantages of the smart water irrigation system.
Some of the parameters, necessary for comparison, are assumed by us in order to give a clear implication:
Area: 30,000,000 square centimeters
Depth: 100 cm
Tpaw: 0.2, which is the ratio maximum extractable water by unit volume of soil is 0.2
Plant one day water demand: 128.52 cubic centimeters
water amount in the morning: 85.68 cubic centimeters
water amount in the evening: 42.84 cubic centimeters
We are also using the given cost as the cost schedule. Refer to figure 2 for detailed information on the variation of water cost over the day.

The results:
total cost: 1621.34870542 dollars
Optimal cost: [504.4103656941631, 530.2124393645449, 0.0, 0.0]
The optimal cost shows the cost for each hour over the four hour optimization program
Cost difference: 586.725900359 dollars

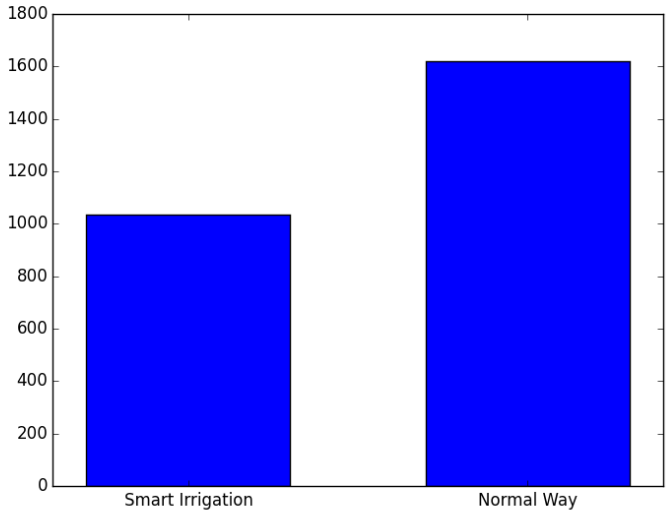


Figure 18: Comparison of optimized model with empirical irrigation technique

After running the designed optimization system,it is easy to find out the optimal cost would be 1034 dollars by using the smart water irrigation system. Using the normal way of irrigation costs 1621 dollars. The difference in cost in 586 dollars per day, which means that using the smart water irrigation system saves the user around 30% of the cost, which is a quite significant percentage. If the smart water irrigation systems will be applied to a larger scale, for example, to all of the farms in the United States, the amount of water saved and the cost saved would definitely be titanic.

Discussion

As presented in the introduction, several options exist to implement changes on the water consumption of crops. As we opted for the smart irrigation option in order to satisfy a cyber-physical system, some plant requirements were neglected in order to simplify the system and make it possible to express it in a mathematical way. The limitation of this is that our system relies heavily on human knowledge to set up the constraints required to satisfy plant growth conditions. Ideally, information sent from sensors should be enough to determine the amount of water to provide to the plant, without any need to specify conditions. However, programming such a situation involves a deep understanding of the interaction of the plant with its environment during its growth process. This is a whole specialised field in its own and is not the focus of this study. Ideally, a more comprehensive information library on plant requirements would be necessary to have a system which really improves productivity. However having an integrated adaptive system is still necessary to adapt to the local particularities.

One interesting further step which could be taken with this cyber-physical system is to have several users in the same area use this optimization program. A variation with time could be

seen of the cost of water as each crop has different requirements throughout their stages of growth. It would be interesting to know if having a great number of users would smooth out the consumption over the day or on the opposite increase peak consumptions.

Summary:

Throughout this report, it can be seen that the concept of cyber-physical systems implies an interaction between different components in the physical and the cyber worlds, with different means of communication and data processing. The emphasis on system really implies an interconnected network, which can only be understood through a well-defined mathematical representation of it. This can be applied to different fields to adapt them to the harsh requirements imposed by environmental conditions and economic competitivity. The use of our system in the irrigation context has been proven to be relevant as it clearly helps different actors involved in it. While it cuts prices for farmers and makes sure water wastage is reduced, it also redistributes water consumption throughout the day and enables to reduce the important infrastructure requirements implied by peaks in consumption.

Relevant Literature

Open Garden - Hydroponics & Garden Plants Monitoring for Arduino
https://www.cooking-hacks.com/documentation/tutorials/open-garden-hydroponics-irrigation-system-sensors-plant-monitoring#step1

Reference:

(1) Pacific Institute: More with less, agricultural water conservation and efficiency in California, by Cooley, Christian-Smith, Gleick
http://www.pacinst.org/wp-content/uploads/sites/21/2013/02/more_with_less3.pdf