Independent Moisture-Sensitive Automatic Watering System

by
Jessica Sherbon

Senior Project

ELECTRICAL ENGINEERING DEPARTMENT

California Polytechnic State University
San Luis Obispo

June 2012

© 2012 Jessica Sherbon

Table of Contents

Section

Appendices

List of Tables

List of Figures

Acknowledgements

I would like to give thanks for the guidance and dedication of my family, friends, and instructors who have helped me through the journey of my education. In particular, thanks to my project advisor Bridget Benson and the rest of the Electrical Engineering Department faculty and staff for their support and advice during my time at Cal Poly.

Abstract

With the availability of home improvement supplies, plant owners can quickly and easily set up an automatic system that waters their gardens based on a specified schedule. However, most of these systems require an electric outlet or water faucet making them unusable in locations where installed water and power lines do not exist, such as apartment balconies. This project defines a small-scale automated watering system that runs independently of installed water and power lines. The system takes water from a reservoir, such as a tank or bucket, and supplies that water to individual plants via a drip system. The design runs on battery power and incorporates maximum water efficiency by measuring soil moisture before watering and protecting against dry operation.

I. Introduction

Proper irrigation has long been integral to maintaining healthy plants. From large-scale agriculture to pots on a windowsill, plants growing in a controlled environment need a plan to receive the right amount of water. Many plant owners with small patio gardens or indoor potted plants water by hand and must check their plants regularly to determine when watering is required. With the busy lives of modern-day individuals and the variety of non-native plants preferred in the home and garden, it becomes even harder to keep those plants alive. This moisture-sensitive automatic watering system provides a solution by keeping plant watering on a strict schedule. The system is optimized for outdoor use, but may also be used indoors since it operates with its own water tank and battery. Very few existing systems provide this level of flexibility.

II. Background

In the last fifty or more years, controlled growing environments have used electricity to increase crop quality and yield [1]. For example, artificial light in greenhouses gives growers control over how much “sunlight” plants get and temperature sensors allow for simulation of warmer or cooler environments. More recently, independently powered electric pumps have been employed to move water to where it’s needed, and some of these systems may be solar powered [2]. Some watering systems will even include wireless sensors [3]-[4].

For plant owners concerned about soil erosion and excess water runoff, drip systems significantly increase water conservation by delivering water only to the roots of the plants that need it [5]. With all this development in plant care, one might expect that multiple systems meeting the requirements of this project already exist. Surprisingly, Claber’s “Oasis” is the only widely available product similar to this project [6]. An indoor watering system intended for vacation use, Oasis runs water continuously through a loop of supplied tubing to allow watering of individually potted plants. Oasis should not be used outdoors, does not have environmental sensors, and can only supply water to plants arranged in a circle. The moisture-sensitive automatic watering system

takes small-scale home watering options one step further by incorporating these features and more.

III. Requirements

Depending on user settings for watering frequency and the size of the supplied reservoir, the system should water for two weeks to a month without being refilled. See Table I, on the next page, for further detailing of requirements and specifications.

Table I: Project Requirements and Specifications

Marketing Requirements	Engineering Specifications	Justification
1, 2	Provides water pressure between 10 and 30 psi from an un-pressurized water source.	Drip system tubing can handle these pressures without breaking and emitters require this range to deliver water correctly [7].
1, 3	Interfaces with ½" diameter hose for drip line.	Common hose diameters for drip line include ½" and ¼", with ½" providing the backbone of the irrigation system [5].
3, 4, 5	Can sense changes in soil moisture with a sensitivity of at least 0.07 Water Fractional Value (WFV) [8]-[10].	The system is intended for outdoor use, so external watering, such as from rain, needs to be considered and measured with sufficient sensitivity for maximum water conservation.
3, 5	Incorporates at least three watering settings that a user may manually select.	Allows for watering different plant types that require wet or dry conditions.
6	System supplies sufficient water for 10-15 potted plants in a 5-8m^2 area [11].	Number of pots/plants estimated given the space proposed (balcony) and the water pressure supplied.
1, 6	Maintains sufficient water pressure for drip system emitter use through at least 4 m of drip line.	Based on the number of plants and considering losses in water pressure as plants are watered by the system.
7	Design for 12V supplied power.	Convenient battery power sufficient for most small-scale systems.
8	Project costs no more than $150.	This cost limit ensures an affordable product and allows for the maximum monetary support from the Electrical Engineering department.

Marketing Requirements

1. Interface with standard drip system structure.
2. Independent of local water main.
3. Maximize water conservation.
4. Reliable in an outdoor environment.
5. Includes multiple watering settings.
6. Supports a small garden.
7. Independent of grid power.
8. Low cost.

The requirements and specifications table format derives from [12], Chapter 3.

IV. Functional Decomposition

This section details the functional systems required to complete the automatic watering system. See Fig. 1 below for the Level 0, or black box, diagram, which defines the most basic inputs and outputs of the system. The system primarily moves water and interprets inputs from its environment. Table II explains each of the system's primary inputs and outputs in greater detail.

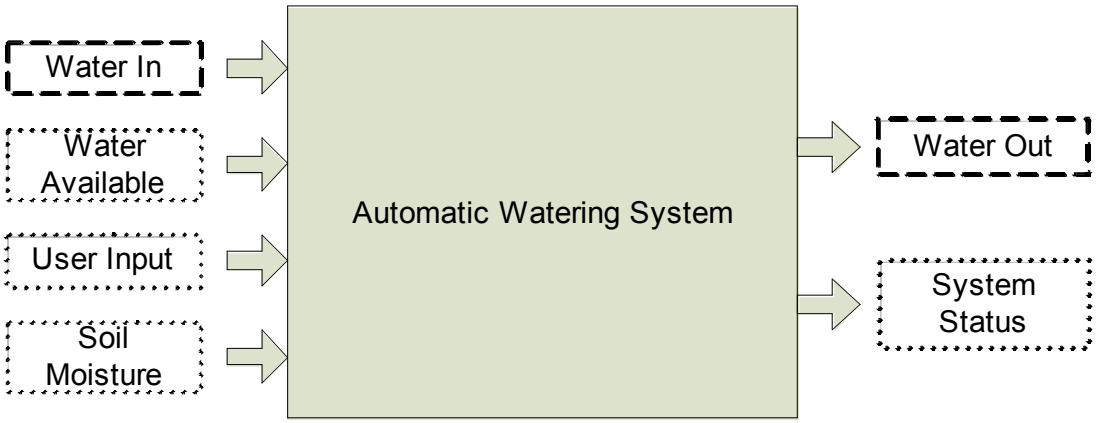

Figure 1: Level 0 System Diagram

Table II: Level 0 Diagram, Inputs and Outputs

Name	Explanation
System Function	Based on a user- or sensor-defined schedule, transports water from a tank to a drip system, in order to water individual plants.
Water In	Water drawn from some source, such as a water tank.
Water Available	A measurement of water level or water availability from the source.
User Input	Watering settings determined by the user.
Soil Moisture	Measurement of sample soil water content.
Water Out	Pressurized water output to the drip system lines.
System Status	Indicators of the current system settings.

The Level 1 diagram, in Fig. 2, shows the next level of detail for internally required components of the watering system.

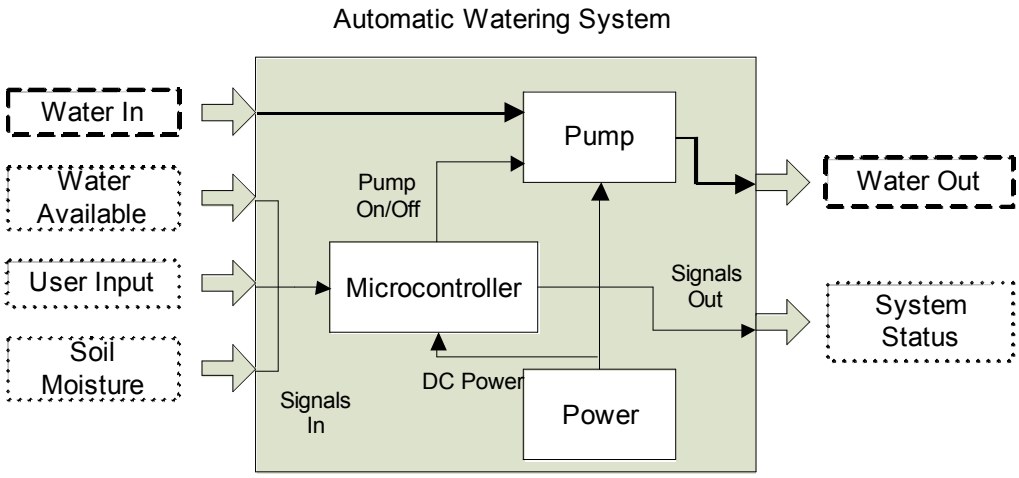

Figure 2: Level 1 System Diagram

Table III describes the internal signals and components of the Level 1 diagram.

Table III: Level 1 Signals and Components

Name	Explanation
Microcontroller *(sub-system)*	Interprets input data in order to run the system and display status.
Power *(sub-system)*	The power supply of the system, and any regulation it requires.
Pump *(sub-system)*	A method for moving water from the tank to the drip system.
Signals In	Describes all data coming into the system to be handled by the microcontroller.
DC Power	The power required for the microcontroller and pump.
Signals Out	Whatever data signals needed to output system status.
Pump On/Off	Signal to turn the water pump on and off.

V. Design

Design began with careful consideration of the project constraints. The complete project has many parts that must work together smoothly and simultaneously for the entire project to function.

A. Providing Power

Power is the first concern, and 12 Volt batteries became the obvious choice after reviewing the design requirements. Batteries fit the requirement for a system off the electrical grid, and 12 Volts standardize the system to a commonly used and adaptable value. These batteries also handle higher amperages required by the larger system components, such as a pump, and become easily accessible in the form of car or motorcycle batteries. Experience with the Cal Poly Amateur Radio Club and the usefulness of 12 Volt batteries in an emergency communications environment influenced this decision, and the club kindly lent a battery for project tests.

B. Water Delivery

The second consideration is moving water from the reservoir to the plants. The project can accomplish this movement with either a gravity or pump driven system. A gravity driven system depends on placement of the water reservoir to develop pressure in the lines. In order for water to flow to the plants, the reservoir sits or hangs above the plant level. However, this

arrangement cannot guarantee the optimal pressure for reliable drip system performance and may not fit within the constraints of the user's space. Therefore, the more dependable solution is a pump driven system. A pump driven solution gives freedom to the reservoir placement and, with the right pump, can supply reliable water pressure to the drip system. Proper support for a pump driven system results in the water delivery path shown in Fig. 3.

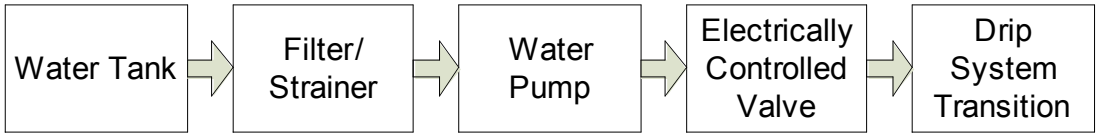

Figure 3: Diagram of water transportation system for interface with electrical control.

The "Irrigation Pump Tutorial" contains useful information for selecting between pump types [7]. According to this resource, some pumps need priming in order to function, which requires a submersible pump in the reservoir to pump water into the primary pump. That arrangement results in a two-pump system - one pump in the tank to get water out and one pump to actually deliver water to the plants. To avoid the extra investment in and hassle of integrating a second pump, the pump for this project self-primes. Further research brought attention to Recreational Vehicle (RV) pumps, which run on 12 Volts and supply water pressures near the range required for drip systems. The Shurflo 105-003 Series Water Pump, shown in Fig. 4, provides

30 psi with automatic shutoff, self-primes up to 2 ½ feet of head, incorporates dry-run protection, and draws a maximum current of 3 Amperes.

Supporting the pump systems, the electrically controlled valve fine-tunes watering control and prevents leaks due to siphoning. Similarly to the pump selection, project constraints quickly narrow the options for electrically controlled valve. Ehcotech International's Electric Solenoid Valve fit the bill with a default closed valve position and 12 Volts required for activation. The filter blocks particulate matter that could damage the pump, and the drip system transition allows for easy interface with drip system tubing. Fig. 4 also shows both of these more generic items, found at the local hardware store.

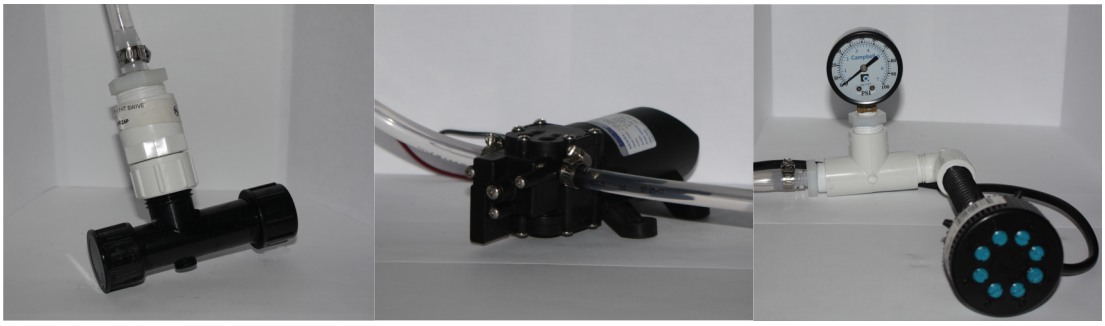

Figure 4: Water system components - Filter, pump, and delivery to ¼ inch drip system tubing with a pressure gauge for testing.

C. Control System

Next, the control system (or microcontroller) is the heart of the project, and must drive all system functions. While fans of PIC microcontrollers make a compelling argument for the PIC's usefulness and compact size, the Arduino

Uno (Fig. 5) integrates most easily, with the most flexibility, and with the most relation to existing programming experience. The Arduino coding environment derives from C, still one of the most prominent higher level coding languages thanks to its popularity over many previous years.

Further, the Arduino has the benefit of having a large user base, having become very popular with hobbyists. Therefore, it has access to a number of extensions or add-ons already developed for it, some of which are called "shields," that fit the profile of the Arduino Uno and attach to it without hassle [13]. Some of these extensions include displays, which would provide a flexible user interface for the project. For example, connecting a Liquid Crystal Display (LCD) to the project displays a variety of information in real time, rather than connecting the project to a computer for settings review and data recovery. The flexibility of this display also opens design options to additional features, such as a user menu and the ability to set the time of day. Adding a standard keypad to this combination allows users to directly manage available features. Dr. Liu's PHI Panel Backpack, pictured in Fig. 5, fits all these criteria and more: an LCD and keypad combination that has its own library to manage serial communication with the Arduino Uno [14].

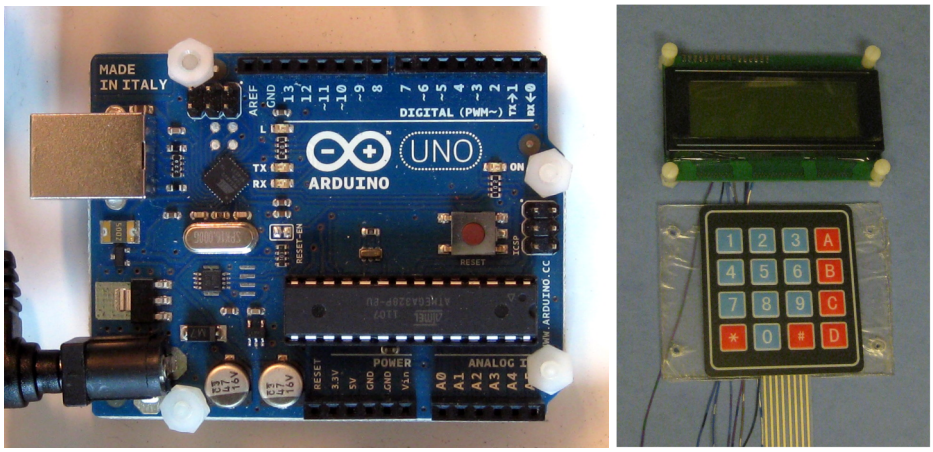

Figure 5: Arduino Uno, and PHI Panel Backpack with keypad.

D. Sensor Array

The sensor array measures the soil moisture and allows the system to determine whether or not it is appropriate to water. For cost concerns this array primarily consists of simple resistance-based moisture sensors. With a temperature sensor for calibration, these moisture sensors have a wide enough range to meet project requirements. Find suggestions for building one of these sensors at Gardenbot [15]. A similar approach suffices for a water level sensor. The water level sensor simply measures for a closed or open circuit, and so does not require the same level of finesse as the moisture sensor. Optionally, a light sensor could be added to the array as an extra feature. Fig. 6 shows the resulting choices.

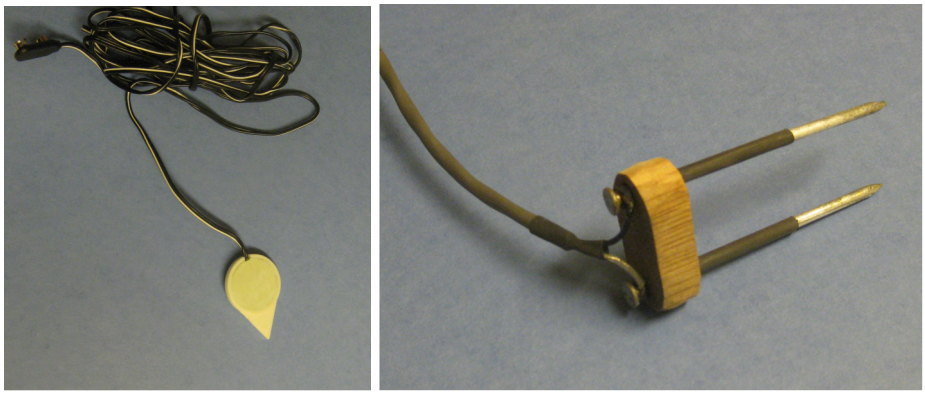

Figure 6: Moisture sensors - ceramic sensor and simple moisture sensor.

VI. Test Plans

Testing for the mechanical portions of the system are fairly straightforward: apply power and the pump runs, the valve switches. Testing the electrical control for these portions took a few more steps, first testing digital I/O, then switching through the transistors needed to interface with 12 Volts, and finally switching the relays with the Arduino. Similarly, components of the sensor array test individually first, then as part of the whole array with data collection managed by the Arduino. This entire process finishes with tests of the whole system. The table below lays out these steps in more detail.

Table IV: Test Plan Details

Items Tested	Test Method	Expected Results
Power *(battery)*	Multimeter, for voltage and current	The battery supplies 12V and the required amperage.
Pump	Observation, pressure gauge	Pump moves water from the tank to the drip system with a maximum of 30 psi.
Valve	Observation	The valve switches open upon application of 12 Volts, remains closed otherwise.
Microcontroller Pins	Driving an LED	Arduino I/O pins display behavior concurrent with programmed code.
Control of Relays	Arduino connected to relays	Relays switch while hooked up to a power supply.
Control of Pump and Valve	Observation	The pump runs and valve switches under system control.
Moisture Sensors	Multimeter, for resistance	Sensor shows a measurable change between water presence and absence.
Sensor Input	Observation, setting up wet and dry conditions	Reasonable readings can be taken from the sensor array, and translated into guidelines for the watering cycle.

VII. Development and Construction

Component purchases followed the overall power and cost constraints determined in the planning and design phase. Once the key components were ordered and received, they had to be made to interface with each other correctly.

A. Water Connections

Despite efforts made in the design phase to choose components that would interface well, many parts of the water delivery system have very different connectors. The filter had female hose thread (FHT) which needed to convert to ¾ inch barb for the pump hose. After the pump, the rest of the system uses various diameters of pipe threading – a standard for irrigation system pipes. Fig. 7 shows the whole assembly of components and adaptors after the pump.

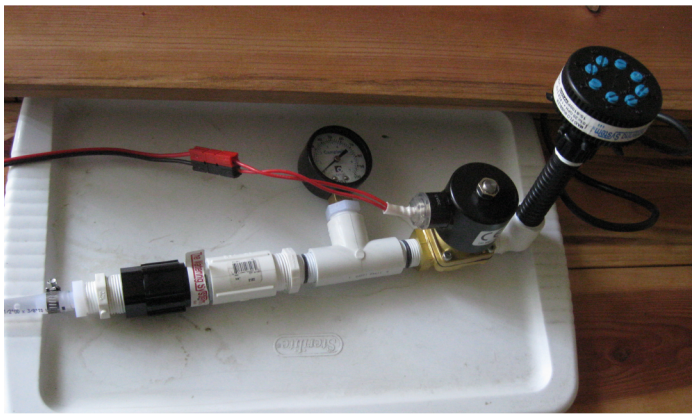

Figure 7: Complete water delivery system from pump output to drip system transition.

B. Microcontroller Interface

The Arduino Uno can only source 40mA via its digital I/O pins, so it must have electrical relays to protect it from the much higher current draw of the pump and valve. Fig. 8 shows the power harness and the relays, which manage 12V and up to 15A on each normally open contact, as well as the transistors interfacing from the Arduino's 5V output to the 12V needed by the relays.

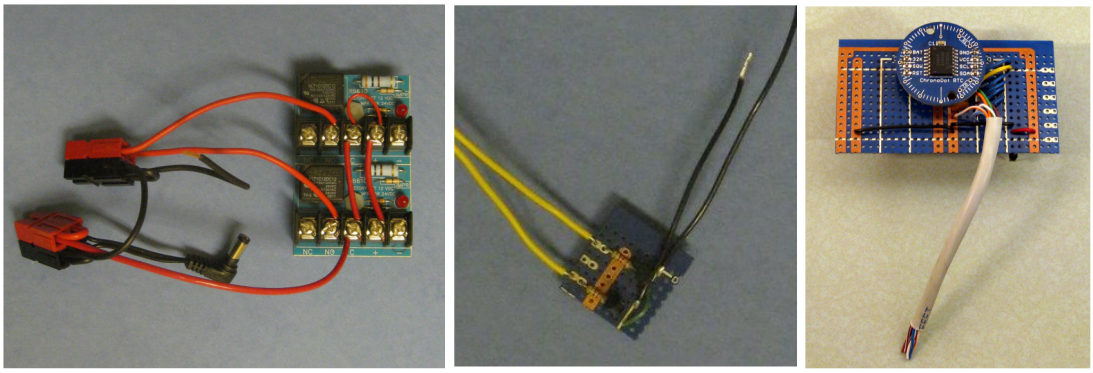

Figure 8: Relays with power harness, transistors interfacing Arduino Uno with relays, and sensor shield with ChronoDot.

The Arduino Uno interfaces with moisture and temperature sensors via a homemade shield, shown last in Fig. 8, which easily and reliably connects the sensor array and the components needed for calibration and current protection to the Arduino Uno. The ChronoDot incorporated into the shield helps the system keep the correct time of day after removing external power. Code developed for the Uno helped with component analysis and with

determining the best combination of components for sensing moisture accurately. See Appendix C for system code.

VIII. Integration and Test Results

Following the methods laid out in the test plans, each component of the project was tested for individual functionality before being integrated into the whole project. This approach drastically simplified troubleshooting and gave a baseline performance for each part and subsystem for later reference.

A. Water Delivery System

The first test arrangement, shown in Fig. 9, tests water flow and pressure through the pump. Without the electrical valve to control water flow, the drip system transition leaked a little bit when under pressure.

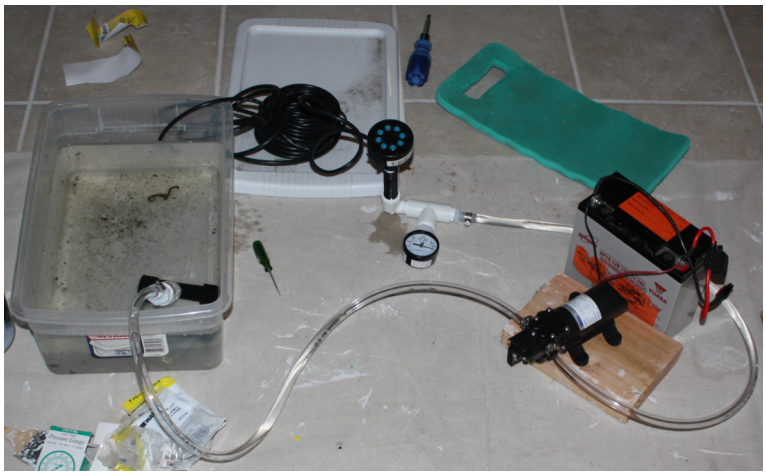

Figure 9: Basic water transportation system in testing, maintaining line pressure of 30 psi.

Other than that small leak, this version of the water transportation system tested just as expected. The addition of a pressure gauge to the system

proves that the pump maintains 30 psi in the line, stopping automatically when it reaches that pressure and starting up again whenever the pressure drops below about 20 psi. The rate at which the pump cycles on and off depends on water flow allowed through the drip system transition. With this in mind, a pressurized tank or pressure regulator may be needed in the final system to smooth water output and reduce stress on the pump.

B. Electrical Control System

The final version of the control system fits into a chassis for a completely enclosed project box. Fig. 10 shows the control system components ready for installation on the right, and then the power harness, relays, and Arduino installed in the bottom of the sheet metal chassis designed for the project.

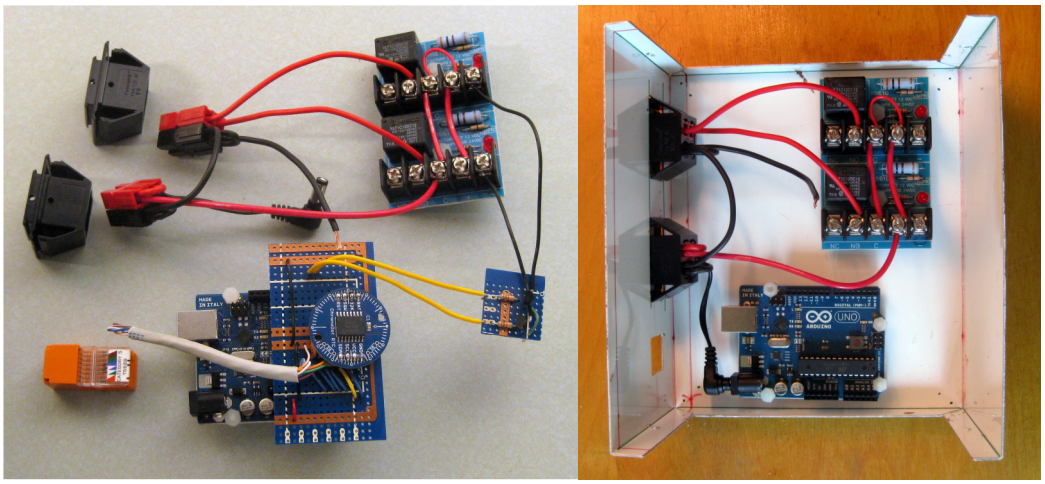

Figure 10: Electrical control system before chassis installation and key components partially installed.

C. Sensor Array Setup

Resistors calibrate each sensor separately and also provide current protection for the Arduino. Table V shows average performance of each moisture sensor, and the resulting readings seen by the Arduino.

Table V: Moisture Sensor Performance

Sensor	Submerged (Tap Water)	Range in Dry to Wet Soil	Calibration Resistor	Measured Voltage Range
Ceramic	150 kΩ	566 kΩ – 150 kΩ	360 kΩ	3.056 V - 1.470 V
Simple	4-5 kΩ	159.3 kΩ – 10.6 kΩ	82 kΩ	3.300 V – 0.572 V

While the ceramic moisture sensor measures across a more consistent medium by bringing water into the ceramic, it suffers from hysteresis. Since the ceramic sensor soaks up water, it takes minutes or hours to first absorb water and then to dry out again. The simple moisture sensor changes resistance based on its medium, and with bare metal exposed to the soil a short could cause false readings. To mitigate this last concern, in the prototype the ceramic moisture sensor tests plant soil moisture while the simple moisture sensor measures the reservoir water level.

D. Final Installation

The sheet metal chassis holds electrical components and display as show in Fig. 11. Though a watertight chassis would be optimal to protect the system from all outdoor weather, time and cost constraints led to this simpler housing option, which will survive mild San Luis Obispo conditions.

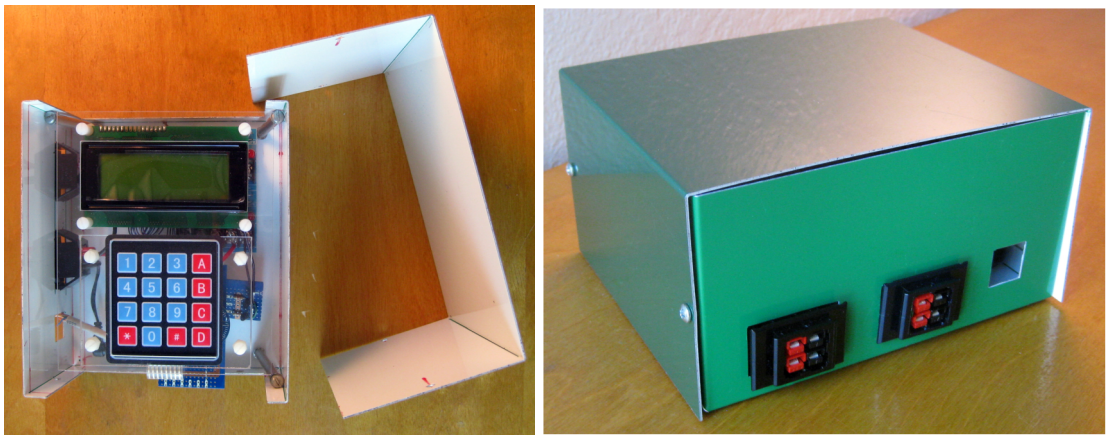

Figure 11: Electrical components in the chassis and the chassis closed.

Below, Fig. 12 presents the shelf designed for the system and plants. Though the bulk and weight of the pump, valve, and battery add up so that the system becomes difficult to move, the arrangement suffices for this stationary installation.

Figure 12: Tiered redwood plant shelf designed and built for installation of the Independent Moisture Sensitive Automatic Watering System.

An arrangement of the prototype installation appears on the title page of this report.

IX. Conclusion

The Independent Moisture Sensitive Automatic Watering System has proven its design by going above and beyond the expectations of the project criteria. As laid out in the requirements and specifications, the system checks for sensor inputs, determines whether to run a watering cycle or not, and successfully operates the pump and valve assembly in order to move water. The system has plenty of potential for further extension with possible additions including solar panels, Ethernet capability for system updates and data taking, and more. The designing, building, and testing processes of this project summarize the skills taught in Electrical Engineering at Cal Poly. Embedded and control systems, programming, transistors for relay control, circuit design, power distribution, and soldering all make an appearance, turning this system into a well-rounded Electrical Engineering project.

References

[1] Procter, F.M.; Weir, J.A.C.; , "Electricity in horticulture," *Electrical Engineers, Proceedings of the Institution of* , vol.120, no.10, pp.1133-1164, October 1973 doi: 10.1049/piee.1973.0240
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5251815&isnumber=5250926

[2] Bigger, J.; Stokes, K.; , "PV-powered water pumping as a utility service: a utility-customer-industry demonstration," *Photovoltaic Specialists Conference, 1993., Conference Record of the Twenty Third IEEE* , vol., no., pp.1252-1257, 10-14 May 1993 doi: 10.1109/PVSC.1993.346941
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=346941&isnumber=8044

[3] Juang, J.-N.; Ekong, D.U.; Carlson, C.; Longsdorf, W.; Miller, M.; , "A Computer-Based Soil Moisture Monitoring and Hydrating System," *System Theory, 2007. SSST '07. Thirty-Ninth Southeastern Symposium on* , vol., no., pp.142-144, 4-6 March 2007 doi: 10.1109/SSST.2007.352335
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4160821&isnumber=4160781

[4] Fazackerley, S.; Lawrence, R.; , "Reducing turfgrass water consumption using sensor nodes and an adaptive irrigation controller," *Sensors Applications Symposium (SAS), 2010 IEEE* , vol., no., pp.90-94, 23-25 Feb. 2010 doi: 10.1109/SAS.2010.5439386
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5439386&isnumber=5439372

[5] “Drip Irrigation”, *Home Depot*, 2011. [Online] Available: http://www.homedepot.com/webapp/wcs/stores/servlet/ContentView?pn=Drip&langId=-1&storeId=10051&catalogId=10053 [Accessed Oct. 28, 2011].

[6] “Drip Irrigation Kits”, *Claber*, 2011. [Online] Available: http://www.claberinc.com/products/irrigation.asp?sz=34&fm=66 [Accessed Nov. 1, 2011].

[7] Stryker, J., “Irrigation Pump Tutorial”. [Online] Available: http://www.irrigationtutorials.com/pump.htm

[8] “Hydra Probe II Soil Sensor”, Stevens Water Monitoring Systems, Inc. [Online] Available: http://www.stevenswater.com/catalog/products/soil_sensors/datasheet/hydraprobeiidatasheetnewweb.pdf [Accessed Nov 6, 2011]

[9] “Soil Terminology”, *SoilSensor.com*, 2011. [Online] Available: http://www.soilsensor.com/terminology.aspx#W [Accessed Nov 6, 2011]

[10] “Field Estimation of Soil Water Content”, 2008. [Online] Available: http://www-pub.iaea.org/mtcd/publications/pdf/tcs-30_web.pdf [Accessed Nov 11, 2011]

[11] “Appendix 12: Minimum Residential Apartment Standards”, District Plan, City of Auckland, 2011, p. 3 [Online] Available: http://www.aucklandcity.govt.nz/council/documents/central/pdfs/appendix12.pdf [Accessed Nov. 14, 2011]

[12] R. Ford and C. Coulston, *Design for Electrical and Computer Engineers*, McGraw-Hill, 2007, p. 37

[13] “Shield”, *Arduino*, 2012, [Online] Available: http://www.arduino.cc/playground/Main/SimilarBoards#goShie [Accessed Feb 21, 2012]

[14] Dr. Lui, “phi-panel”, *Luidr’s Blog*, ,[Online] Available: http://liudr.wordpress.com/gadget/phi-panel/ [Accessed Mar 24, 2012]

[15] Frueh, A., “The Soil Moisture Sensor”, Gardenbot, [Online] Available: http://gardenbot.org/howTo/soilMoisture/ [Accessed Dec 9, 2011]

Appendices

A. Senior Project Analysis

Independent Moisture Sensitive Watering System

Student: Jessica Sherbon Advisor: Bridget Benson

Summary of Functional Requirements

This project designs a small-scale automated watering system for home use that runs independently of installed water and power lines. The system takes water from a reservoir, such as a tank or bucket, and supplies that water to individual plants via a drip system. The design runs on battery power and incorporates maximum water efficiency by measuring soil moisture before watering and protecting against dry operation.

Primary Constraints

Problems anticipated before the design phase included pump control and pressure regulation. Power requirements and the method of pump control depend heavily on the pump used, as does water pressure through the system. Correct determination of moisture in the soil and water in the supply also proved difficult due to the limited availability of highly accurate moisture sensors within the project budget. While these limitations found workarounds

in the design, optimizing the system requires more time and specialized components.

Economic

This project depends on parts made by existing companies, thus incorporating manufacturing capital. Further, the handling, packing, and transportation of these materials utilize human capital. Materials required for production, such as plastics and metals, negatively impact global natural capital, while use of the system benefits local natural capital. Fig. 3 displays the project timeline from research and design to prototype development to project completion.

	Task Name	Duration	Start	Finish	Pre
1	**Project Length**	**181 days**	**Thu 9/8/11**	**Thu 5/17/12**	
2	**Planning and Design**	**93 days**	**Thu 9/8/11**	**Mon 1/16/12**	
3	Research	70 days	Thu 9/8/11	Wed 12/14/11	
4	EE 460 Requirements	60 days	Mon 9/19/11	Fri 12/9/11	
5	Design	33 days	Fri 11/18/11	Tue 1/3/12	
6	Order and Receive Parts	9 days	Wed 1/4/12	Mon 1/16/12	5
7	**Build and Test**	**71 days**	**Tue 1/17/12**	**Tue 4/24/12**	
8	Implement Design	14 days	Tue 1/17/12	Fri 2/3/12	6
9	First Tests	10 days	Tue 2/7/12	Mon 2/20/12	8
10	Adjust Design and Rebuild	12 days	Tue 2/21/12	Wed 3/7/12	9
11	Second Tests	10 days	Thu 3/8/12	Wed 3/21/12	10
12	Final Adjustments	10 days	Thu 3/22/12	Wed 4/4/12	11
13	Final Tests	14 days	Thu 4/5/12	Tue 4/24/12	12
14	**Written Report**	**179 days**	**Mon 9/12/11**	**Thu 5/17/12**	
15	Structure	30 days	Mon 9/12/11	Fri 10/21/11	
16	Content	166 days	Mon 9/19/11	Mon 5/7/12	
17	Figures and Graphs	85 days	Tue 1/10/12	Mon 5/7/12	
18	EE 463 Progress Report	14 days	Tue 2/21/12	Fri 3/9/12	
19	Touchups and Formatting	14 days	Wed 4/25/12	Mon 5/14/12	13
20	EE 464 Report	7 days	Wed 5/9/12	Thu 5/17/12	

Figure 13: Project Gantt Chart

Personal learning from system design and implementation make up the greatest portion of individual input to the project. Funding for this project comes primarily from material donations and out-of-pocket expenses, with assistance from the Electrical Engineering Department up to $150.00. Table IV has a breakdown of cost estimates for components detailed in the Level 1 system diagram (Fig. 2). Time and cost estimates for Fig. 3 and Table IV determined using Equation 1, below, adapted from equations (1) and (6) in [1].

$$Total = \frac{optimistic + 4(realistic) + pessimistic}{6} \quad (1)$$

Table VI: Project Cost Estimates

Item	Min. Cost	Max. Cost	Most Likely Cost	Explanation	Predicted Cost
Microcontroller	$2	$50	$30	Looked at Arduino and PIC microcontrollers [2-3].	$28.67
Pump	$10	$200	$35	From a brief search for small electric pumps.	$58.33
Moisture Sensors	$8	$50	$20	Adjusted for two sensors.	$23.00
Drip Line	$10	$25	$15	Prices for tubing and some junctions [4].	$15.80
Water Tank	$5	$22	$10	Considering a simple plastic tub.	$11.17
Electrical Wire	$3.60	$4.60	$4	Using 10 feet at $0.40/foot for the most likely cost.	$4.03
Total Material Cost	-	-	-	Sum of material costs.	$141.00
Labor	-	-	-	Approximately 2 worker-months*$100,000/12 months [1].	$16,667.00
Total Cost	-	-	-	Final sum.	$16,808.00

Commercial Manufacturing

While there are no plans for large scale manufacturing at this time, a model may be designed for consumer use. Large-scale production of the system would also tie back into economics by providing jobs for individuals. Large-scale production implies a design suitable for an assembly line and hundreds of units available per year. Comparing prices to similar products, each unit should cost the consumer between $50.00 and $100.00 to be competitive [2]. The user needs only supply water and batteries on a regular basis, with one-time startup costs of the drip system components and a tank or bucket to hold the water. The user also supplies the plants.

Environmental

To be resilient in a natural outdoor environment the watering system requires non-biodegradable parts, such as plastics and metal. Easily recyclable versions of these components may not be commercially available. In this case, large-scale manufacturing has negative environmental impacts as natural resources are diverted to production of the product. These effects may be reduced if design adjustments make the watering system parts reusable or recyclable.

Manufacturability

As a system compatible with existing irrigation techniques, parts should use standard sizes and be easily available. This requirement applies to the interfaces between battery, pump, and drip system tubing in particular. For this project, construction will be accomplished primarily by hand, largely because the project needs only one final prototype. Large-scale manufacturing may require a complete re-envisioning of the system design in order to accommodate more efficient assembly-line techniques.

Sustainability

Regarding product life, the system design must support and protect itself from its environment so that the system remains viable. For example, water cannot enter into electrical components in order to avoid short circuits and corrosion. Similarly, the water tank, pump, and other connections should not leak in order to maintain the goal of maximum water conservation. Considering other aspects of sustainability, this watering system promotes a healthier environment for individual users by providing greenery in the home. Widespread use of this or similar systems could even improve the local environment by reducing the carbon footprint of individual households.

Ethical

If this project were sold as a commercial product, ethical benefits include empowerment and promotion of individual responsibility in successfully caring for a plant or garden. The garden can then become an outlet for self-care by supporting aesthetics and personal wellness. In contrast, reallocating resources from the natural environment to private homes for personal benefit may be considered ethically detrimental. Minimizing the impacts of product production mitigates this disadvantage. With the implementation of re-usable or recyclable components and the project emphasis on water conservation, this project remains an ethically responsible choice for plant maintenance. Referring to the IEEE code of ethics, full disclosure of the beneficial and detrimental impacts of the project maintains “ethical and professional conduct” [6]. Further, the project design incorporates the highest regard for public safety, as discussed next.

Health and Safety

In case of misuse or systems failure, this project may cause harm to individuals, pets, or property. Electric shock may hurt people and pets if they come into contact with the power supply. Leaks from the water tank, pump, or drip system may damage property if the system seals fail. Luckily, the small scale of the watering system reduces these dangers since the system only

requires low voltages and a small, carefully controlled source of water as opposed to the electrical grid and water mains. As discussed in the sections on sustainability and ethics, an automatic watering system may also promote wellness by helping maintain greenery in or near the home. This may extend to healthy eating if the garden becomes a source of fresh produce or culinary seasonings.

Social and Political

The watering system enables homeowners/renters to eat more healthily via their own gardens, even in an environment with limited gardening resources. This ideal ties in with the push for green living seen in some consumer choices over the last decade, and can promote both mental and physical health [7].

Development

Completion of this project requires a study of water flow and watering systems. Since Electrical Engineering at Cal Poly does not require fluid dynamics, this section demanded some of the most learning. As such, it took considerable research to develop an effective design, mostly determining which parts available on the market were applicable to the project. The project also requires an understanding of embedded systems and microcontrollers,

as well as programming experience with them. The sources cited below represent the beginning of this research and learning.

Analysis References and Literature Search

[1] R. Ford and C. Coulston, *Design for Electrical and Computer Engineers*, McGraw-Hill, 2007, p. 37

[2] *Arduino*, 2011. [Online] Available: http://www.arduino.cc/ [Accessed Nov 20, 2011].

[3] *Microchip*, 2011. [Online] Available: http://www.microchip.com/ [Accessed Nov 20, 2011].

[4] "Drip Irrigation", *Home Depot*, 2011. [Online] Available: http://www.homedepot.com/webapp/wcs/stores/servlet/ContentView?pn=Drip&langId=-1&storeId=10051&catalogId=10053 [Accessed Oct. 28, 2011].

[5] "Claber Oasis", *Amazon*, 2011. [Online] Available: http://www.amazon.com/s/?ie=UTF8&keywords=claber+oasis&tag=googhydr-20&index=aps&hvadid=4274282335&ref=pd_sl_7e7klbxvgx_b [Accessed Nov 1, 2011].

[6] "IEEE Code of Ethics", IEEE, 2012. [Online] Available: http://www.ieee.org/about/corporate/governance/p7-8.html [Accessed Jan 3, 2012].

[7] Page, M., "Gardening as a therapeutic intervention in mental health", *Nursing Times*, 2008, 104: 45, 28-30. [Online] Available: http://www.nursingtimes.net/nursing-practice-clinical-research/gardening-as-a-therapeutic-intervention-in-mental-health/1921374.article [Accessed Jun 6, 2012].

[8] *IEEE Std 1233, 1998 Edition*, p. 4 (10/36), DOI: 10.1109/IEEESTD.1998.88826

[9] "Drip Irrigation Kits", *Claber*, 2011. [Online] Available: http://www.claberinc.com/products/irrigation.asp?sz=34&fm=66 [Accessed Nov. 1, 2011].

[10] "Hydra Probe II Soil Sensor", Stevens Water Monitoring Systems, Inc. [Online] Available: http://www.stevenswater.com/catalog/products/soil_sensors/datasheet/hydraprobeiidatasheetnewweb.pdf [Accessed Nov 6, 2011].

[11] "Soil Terminology", *SoilSensor.com*, 2011. [Online] Available: http://www.soilsensor.com/terminology.aspx#W [Accessed Nov 6, 2011]

[12] "Appendix 12: Minimum Residential Apartment Standards", District Plan, City of Auckland, 2011, p. 3 [Online] Available: http://www.aucklandcity.govt.nz/council/documents/central/pdfs/appendix12.pdf [Accessed Nov. 14, 2011].

[13] *Sure to Grow*, 2008. [Online] Available: http://www.SureToGrow.com/ [Accessed Nov. 1, 2011].

[14] M. McKinley, *Scotts Sprinklers and Watering Systems*, Meredith Books, 2005.

[15] "Field Estimation of Soil Water Content", 2008. [Online] Available: http://www-pub.iaea.org/mtcd/publications/pdf/tcs-30_web.pdf [Accessed Nov 11, 2011].

[16] Sanz, M.; Sanz, J.F.; Botero, D.; Navarro, M.; Val, F.J.; Melero, J.J.; Sallan, J.; Llombart, A.; , "Optimal integration of renewable energies in a pumping station for irrigation," *IECON 02 [Industrial Electronics Society, IEEE 2002 28th Annual Conference of the]* , vol.4, no., pp. 3332- 3337 vol.4, 5-8 Nov. 2002 doi: 10.1109/IECON.2002.1182932 [Online]

Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1182932&isnumber=26552 [Accessed Dec 10, 2011]

[17] Liu Xiaochu; Ling Jingpeng; Yao Li; Wu Hualong; Tao Jianhua; , "Engineering quality control of solar-powered intelligent water-saving irrigation," *Informatics in Control, Automation and Robotics (CAR), 2010 2nd International Asia Conference on* , vol.3, no., pp.254-257, 6-7 March 2010 doi: 10.1109/CAR.2010.5456679 [Online] Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5456679&isnumber=5456636 [Accessed Dec 10, 2011]

B. Electrical Diagrams

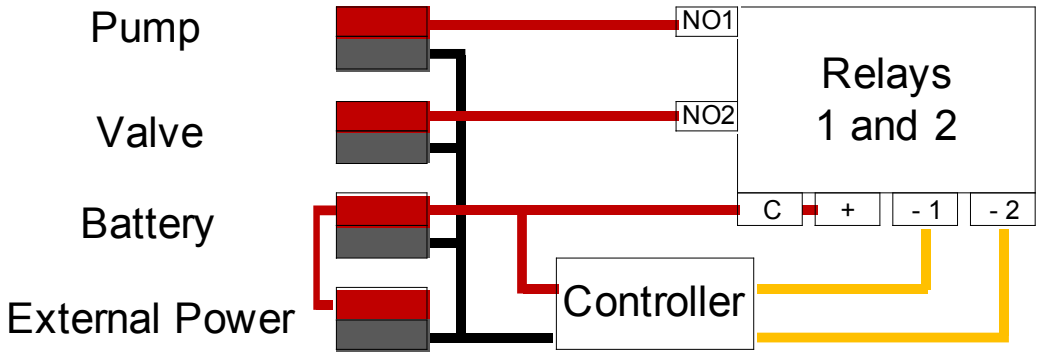

Figure 14: Electrical Power Connections

C. System Code

```
/*  Written by Jessica Sherbon, last updated 06/14/2012
With reference to and sections from Dr. Lui's Phi Panel demo code and Evil Mad
Scientist's Bulbdial clock code.
This program tests the combined functions of sensors, watering cycle, and user
interface for the Independent Moisture Sensitive Automatic Watering System. */

#include <avr/pgmspace.h> //Libraries
#include <SoftwareSerial.h> //For communication with Phi Panel display
#include <Wire.h>  //For I2C communication to ChronoDot (RTC)

//End of month lookup
PROGMEM const byte eom_lut[] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

int brightness = 170; //For Phi Panel backlight
int second = 0;
byte secNow, minNow, hrNow;
byte monthNow, dateNow, yearNow;
byte VCRmode;  // In VCR mode, the clock blinks until the time has been set.
byte ExtRTC;  // Indicates if a Chronodot was found
char msg[80]; // This is a buffer for the output string.
char main_menu_text[]="Main menu\n1.General Settings\n2.Water Cycle~";
char clk_menu_text[]="Settings menu\n1.Set Time\n2.Set Date~";

byte menuMode = 0;
char response = 0;
byte updateDate = 0;

//Defining digital pins
#define PUMP_PWR 6 // to switch water pump on/off
#define VALVE_PWR 7 // to switch solenoid water valve on/off
#define SENSOR_PWR 8 // to power sensor array

//Defining analog pins
#define WATER_LVL_SENSOR A0   // to read water level
#define SOIL_MOIST_SENSOR A1 // to read moisture sensor
#define TEMP_SENSOR A2   // to read temperature sensor
#define LIGHT_SENSOR A3   // to read light sensor (optional)

//Phi Panel Tx on pin 2, Rx on pin 3
SoftwareSerial sSerial(2,3);
```

```
//sensor and water cycle variables
int water_lvl, soil_moist, temp_F, light_in = 0;
int water_ok = 0;

void setup()
{
  //Initialize digital pins
  pinMode(PUMP_PWR, OUTPUT);
  pinMode(VALVE_PWR, OUTPUT);
  pinMode(SENSOR_PWR, OUTPUT);

  //Initialize analog pins
  pinMode(WATER_LVL_SENSOR, INPUT);
  pinMode(SOIL_MOIST_SENSOR, INPUT);
  pinMode(TEMP_SENSOR, INPUT);
  pinMode(LIGHT_SENSOR, INPUT);

  //Serial communication (display)
  sSerial.begin(19200);
  delay(100);
  sSerial.print("\f\e[0m~"); // Clear screen
  sprintf(msg,"\eQ%d~/e[21~", brightness); //Change LCD backlight brightness and
turn off scrolling
  sSerial.print(msg);

  //clock setup -> default: midnight 3/11/88
  secNow = 0;
  hrNow = 0;
  minNow = 0;
  dateNow = 11;
  monthNow = 3;
  yearNow = 88;

  VCRmode = 1;  //Time is not yet set

  Wire.begin();  //Set up I2C on A4 and A5 pins for Chronodot communication
  ExtRTC = 0;

  // Check if RTC is availiable, and, if so, use it to set the time
  ExtRTC = RTCgetTime(); // If no RTC is found, no attempt should be made to use it
thereafter

  if(ExtRTC)     // If time is already set from the RTC...
    VCRmode = 0;
```

```
  // initialize Timer1
  noInterrupts();          //Disable interrupts
  TCCR1A = 0;              //Set Timert 1 control A & B for CTC (Clear Timer on
Compare) and 1024 prescale
  TCCR1B = (1 << WGM12) | (1 << CS12) | (1 << CS10);
  TCNT1  = 0;              //Clear count
  OCR1A = 15625;           //Set compare match register to 16 MHz / 1024 for 1 Hz
rate
  TIMSK1 |= (1 << OCIE1A); //Timer/Counter1 Output CompareA Match Interrupt
Enable
  interrupts();            //Enable interrupts
}

void loop()
{
  if(sSerial.available()) {
    response = sSerial.read();
    if(response == '\n') {
      menuMode = 1;
      main_menu();
      menuMode = 0;
    }
  }
  WaterCycle();
}

void WaterCycle()
{
  SensorArray();

  if(water_ok = 1) {
    digitalWrite(VALVE_PWR, HIGH); //turns valve pin on - valve opens
    delay(1000);
    digitalWrite(PUMP_PWR, HIGH); //turns on pump after waiting 1 second
    delay(5000);
    digitalWrite(PUMP_PWR, LOW); //turn off pump after waiting 5 seconds
    delay(7000);
    digitalWrite(VALVE_PWR, LOW); // turn off valve after waiting another 7 seconds
    delay(6000); // wait 6 seconds
  }
}

void SensorArray()
{
  digitalWrite(SENSOR_PWR, HIGH); //turns sensor power on
  delay(1); //wait 1ms
```

```
  water_lvl = analogRead(WATER_LVL_SENSOR);

  int sensorValue, temp_raw = 0;
  for(int i = 0; i < 5; i++) {           // for greater accuracy, average moisture sensor input
    sensorValue += analogRead(SOIL_MOIST_SENSOR);
    delay(1); // wait 1ms
  }
  soil_moist = sensorValue/5;
  if(soil_moist > 510) {
     water_ok = 1; //allow water cycle to run
  }else{
     water_ok = 0;
  }
  temp_raw = analogRead(TEMP_SENSOR);
  // value is degrees * 10, then -50 corrects for calibration
  temp_F = (int)( 5000.0 * ((float)sensorValue / 1023.0) ) - 50;
  light_in = analogRead(LIGHT_SENSOR);
  delay(5); // wait 5ms
  digitalWrite(SENSOR_PWR, LOW); //turns sensor power off
}

ISR(TIMER1_COMPA_vect) // Timer compare interrupt service routine, must not
take more than 1 second
{
  if(menuMode == 0) {  //Don't display if in menu mode

    if(!(VCRmode && (secNow & 1))) {
      // Update clock
      sprintf(msg, "\e[1;1H~%02d/%02d/%02d  ", monthNow, dateNow, yearNow); //
Reset graphic style, row 1 column 1, format mm/dd/yy
      sSerial.print(msg); // Output mo/da/yr
      sprintf(msg, "%02d:%02d:%02d  ", hrNow, minNow, secNow); // Format
hh:mm:ss
      sSerial.print(msg); // Output hh:mm:ss
      sSerial.print("\nPress * for Menu"); //user instructions
    }
    // VCR blink mode
    else {
      sSerial.print("\f"); // Clear screen
    }
  }
  time_update();
}
```

```
void time_update()
{ // Update time
  if(secNow++ >= 59) {
    secNow = 0;
    if(minNow++ >= 59) {
      minNow = 0;
      if(hrNow++ >= 23) {
        hrNow = 0;
        if(dateNow++ >= eom_lut[monthNow - 1]) {
          if( (monthNow != 2) || (dateNow != 29) || yearNow % 4 != 0) {
            dateNow = 1;
            if(monthNow++ >= 12) {
              monthNow = 1;
              yearNow++;
            }
          }
        }
      }
    }
  }
}

void main_menu()
{
  sSerial.print("\eG"); // Start long message
  sSerial.print(main_menu_text); // Print menu as long message

  while(!sSerial.available());
  char response = sSerial.read();
  if(response == '1') clk_menu();
  if(response == '2') WaterCycle();
  sSerial.print("\f"); // Clear screen
}

void clk_menu()
{
  sSerial.print("\eG"); // Start long message
  sSerial.print(clk_menu_text); // Print menu as long message

  while(!sSerial.available());
  char response = sSerial.read();
  if(response == '1') setClock();
  if(response == '2') setDate();
  sSerial.print("\f"); // Clear screen
}
```

```
void setClock()
{
  sSerial.print("\fEnter Time HHMM:\n");
  unsigned long acc = 0;

  while(1) {
    if(sSerial.available()) {
      char response = sSerial.read();
      if((response>='0') && (response<='9')) {
        sSerial.print(response);
        acc = (acc * 10) + (response & 0x0f);
      }

      if(response == '\n') {
        acc %= 10000;  //Accept only last four numbers
        hrNow = acc / 100;
        minNow = acc % 100;
        secNow = 55;
        sprintf(msg, "\n%02d:%02d", hrNow, minNow);
        sSerial.print(msg);
        RTCsetTime(hrNow, minNow, secNow);
        VCRmode = 0;
        delay(2000);
        break;
      }
    }
  }
}

void setDate()
{
  sSerial.print("\fEnter Date MMDDYY:\n");
  unsigned long acc = 0;

  while(1) {
    if(sSerial.available()) {
      char response = sSerial.read();
      if((response>='0') && (response<='9')) {
        sSerial.print(response);
        acc = (acc * 10) + (response & 0x0f);
      }

      if(response == '\n') {
        acc %= 1000000;  //Accept only last six numbers
        monthNow = acc / 10000;
        dateNow = (acc % 10000) / 100;
```

```
        yearNow = acc % 100;
        sprintf(msg, "\n%02d/%02d/%02d", monthNow, dateNow, yearNow);
        sSerial.print(msg);
        RTCsetDate(dateNow, monthNow, yearNow);
        delay(2000);
        break;
      }
    }
  }
}

// Chronodot Interface Functions - lifted from Bulbdial code
void RTCsetTime(byte hourIn, byte minuteIn, byte secondIn)
{
  Wire.beginTransmission(104); // 104 is DS3231 device address
  Wire.write((byte)0); // start at register 0

  byte ts = secondIn / 10;
  byte os = secondIn - ts*10;
  byte ss = (ts << 4) + os;

  Wire.write(ss); //Send seconds as BCD

  byte tm = minuteIn /10;
  byte om = minuteIn - tm*10;
  byte sm = (tm << 4 ) | om;

  Wire.write(sm); //Send minutes as BCD

  byte th = hourIn /10;
  byte oh = hourIn - th*10;
  byte sh = (th << 4 ) | oh;

  Wire.write(sh); //Send hours as BCD

  Wire.endTransmission();

}

void RTCsetDate(byte dateIn, byte monthIn, byte yearIn)
{
  Wire.beginTransmission(104); // 104 is DS3231 device address
  Wire.write((byte)4); // start at register 4

  byte ts = dateIn / 10;
  byte os = dateIn - ts*10;
```

```
  byte ss = (ts << 4) + os;

  Wire.write(ss); //Send date as BCD

  byte tm = monthIn /10;
  byte om = monthIn - tm*10;
  byte sm = (tm << 4 ) | om;

  Wire.write(sm); //Send month as BCD

  byte th = yearIn /10;
  byte oh = yearIn - th*10;
  byte sh = (th << 4 ) | oh;

  Wire.write(sh); //Send year as BCD

  Wire.endTransmission();
}

byte RTCgetTime()
{ // Read out time and date from RTC module, if present
  // send request to receive data starting at register 0

  byte status = 0;
  Wire.beginTransmission(104); // 104 is DS3231 device address
  Wire.write((byte)0); // start at register 0
  Wire.endTransmission();
  Wire.requestFrom(104, 7); // request seven bytes (seconds, minutes, hours, day,
date, month, year)

  int seconds, minutes, hours, day, date, month, year;

  while(Wire.available())
  {
    status = 1;
    seconds = Wire.read(); // get seconds
    minutes = Wire.read(); // get minutes
    hours = Wire.read();   // get hours
    day = Wire.read();  // get day of week, though not retained for this application
    date = Wire.read(); // get day of month
    month = Wire.read(); // get month
    year = Wire.read();   // get year
  }
```

```
  if (status) {
    seconds = (((seconds & 0b11110000)>>4)*10 + (seconds & 0b00001111)); //
convert BCD to decimal
    minutes = (((minutes & 0b11110000)>>4)*10 + (minutes & 0b00001111)); //
convert BCD to decimal
    hours = (((hours & 0b00110000)>>4)*10 + (hours & 0b00001111)); // convert BCD
to decimal (assume 24 hour mode)
    date = (((date & 0b11110000)>>4)*10 + (date & 0b00001111)); // convert BCD to
decimal
    month = (((month & 0b11110000)>>4)*10 + (month & 0b00001111)); // convert
BCD to decimal
    year = (((year & 0b00110000)>>4)*10 + (year & 0b00001111)); // convert BCD to
decimal

    secNow = seconds;
    minNow = minutes;
    hrNow = hours;
    dateNow = date;
    monthNow = month;
    yearNow = year;
  }

  return status;
}
```