EE/MKT/FASC Collaborative Project

Spring 2012

Irrigation Controller for Improved Water Conservation

Product Overview

The project this semester is a residential or commercial irrigation controller that incorporates design features that will hopefully lead to reduced water consumption. The water conservation goal is achieved by two methods: sensors that help determine the amount of watering that needs to be done, and an improved user interface that makes it easier for users to adjust water usage.

Most low cost existing sprinkler controllers run as an open-loop system with no feedback to adjust the amount of water used based on recent rainfall, temperature, humidity, etc. This leads to water being wasted when the sprinklers activate when the ground is already sufficiently wet. The ability to adjust the water usage is a feature usually only found on higher priced models. A goal for this project is to incorporate this feature in a model in the price range most homeowners find acceptable.

A common problem with many irrigation controllers is that users find them difficult or inconvenient to make changes to the programming. For example, an owner notices late one night that it's raining and realizes that the sprinklers are scheduled to run early the next morning. Most users would be unwilling to go outside in the rain to adjust the controller to not run the sprinklers. The product being developed here should have a way to make adjustments to the program that doesn't require going to the controller box and pressing switches.

Another reason people give for not making adjustments to their sprinkler controller when the weather changes is that they don't know how it works. The design team should give careful consideration to the design of the user interface to make it easy for people to make adjustments.

Product Features

The requirements stated below for the basic product can be implemented in a variety of ways. It is up to the design team to decide how to implement the features. Design trade-offs in areas such as reliability, manufacturability, ease of use, cost, etc., should also be analyzed to determine which is the best way to design the product.

The main requirement for this irrigation controller is to include features that reduce water consumption. Two are mentioned above, sensing environmental conditions and easier programming, but teams are encouraged to add others if their market research indicates that these are technically feasible and would make the product more appealing to consumers. However the team must remember that each added feature will add to the product's cost so adding lots of interesting features may force the price above what many consumers would be willing to pay. Some possible design options can be found by studying the features that other manufacturers have included in similar devices.

Environmental Sensor

At least one type of environmental condition must be measured and used by the controller for determining how much to water, or if at all. The conditions might include rainfall, humidity, temperature, moisture content of the ground, etc. The sensor(s) can be connected directly to the controller or can communicate

over a wireless link, whichever the product team feels is best. The sensor information can be analyzed over as long a time as the team feels is appropriate. For example if the sensor is detecting rainfall now, but there hasn't been any for the last two weeks, the controller may still decide to run the sprinklers for a short time.

Irrigation Capability

Many commercial sprinkler controllers can handle 6, 8, 12, or more control valves or "zones". The cost of the unit goes up as the number of valves it can handle increases. For this product, the number of zones the controller can handle should be determined by the market research and estimates of the cost of the device.

For however many zones your product is designed to support, the controller must have the ability to program each valve go on at least three times in a 24-hour period. The controller needs to be able to keep track of the time of day much like a digital clock. It should also know what day of the week it is and allow programming that is different for each day of the week or that repeats after some number of days. Additional programming flexibility and features can be added at the discretion of the product design team.

Inputs

The controller should be programmable locally with some sort of user interface. The design team should select the input controls based on the type and amount of information that needs to be stored and their market research as to what consumers what to use. Some options are:

- Buttons
- Dials
- Keypad
- Remote control device
- Interface to computer

The product should be easy for non-technical users to program and use.

The ability to also program the controller from a remote site should also be considered. A remote site does not have to be far away but could simply be another control interface inside the house.

Outputs

The device needs to have some sort of output in order to communicate its status and conditions to the user. The output display should be some sort of display that can be seen in the dark. The design team must decide what information needs to be displayed. The output also needs to provide the user with any pertinent information when the device is being programmed.

The product also needs to provide an output for each of the control valves.

Power Outages

The controller must include the ability to retain all its programming settings if the power to the device is interrupted. A user should not have to reprogram the device if it has lost power for a few hours. It should also include the capability to maintain the correct time in its internal clock through power outages. It is not necessary that it run the sprinklers if the power is out.

Technical Requirements

The technical details of the product can be implanted in a variety of ways. In all projects involving microcontrollers there are ways to build the project that require more hardware and less software, and vice-versa. The design team needs to decide whether to implement the various parts of the design in hardware or software.

In addition to the baseline requirements, students are expected to think of other features that would add value to the product and if possible add some of these features to the project. Feature that cannot be added to the prototype due to time constraints can be described in the final report as features that should be considered for inclusion if and when the device goes into production.

Microcontroller I/O

One of the more important constraints on any project using microcontrollers is the number and type of available I/O lines the microcontroller has. Various input and output devices will all require using these lines and it is very easy to develop a design the exceeds the number available. This constraint has a significant effect on the decision as to what type of input and output devices to use. Some types of input and output devices require more I/O lines to the micro than others.

User Interface

An important part of the project is the design of the user interface. In the case of this project the user interface must reflect the number irrigation zones the product is capable of controlling, but the prototype hardware is only required to actually interface to two control valves. In other words, your product may say that it can handle 12 zones, and the buttons and display may make it look like it's handling 12 zones, but only two zones need to be capable of working to the extent of controlling a valve.

Control Valves

For this project it is required that the device be able to control two standard 24 Volt AC irrigation control valves. The 24VAC power source is not a part of the project, but your design must incorporate something that allows it to control the valves by switching the 24VAC power from an external power source to the valve on and off. This can be done in a variety of ways (relays, triacs, etc.) and your team will have to decide on a method that works best. The EE 459Lx lab has a testing device with two sprinkler control valves. Teams will be provided with hardware to connect their device to the test board.

Timing

Most commercial sprinkler controllers keep track of time by counting the cycles of the 120 Volt, 60Hz AC power line. In the interest of safety, we do not want to have any 120 Volt power connected to our project boards. Projects teams should use some other method to keep track of time:

- a real-time clock (RTC) chip and oscillator
- use the microcontroller's internal timers to generate interrupts
- an oscillator (e.g. NE555 timer) on the project board

For the purposes of this prototype, accuracy is not very important so any method that makes the clock run with less than about a 5% error is acceptable.

When testing the device it would be very useful to have the option to make it run much faster than normal. For example, the design might include a switch that makes the internal clock advance one minute for every second of actual time, or one day for every minute, etc.

Memory

The requirement that the programming setting must be saved through power outages can be achieved by using some sort of non-volatile memory such as an EEPROM or the device can be provided with long-term battery backup.

Printed Circuit Board

In order for the FASC students to make a physical "looks-like" prototype of the product they will need information from the EE students on how big the internal components will have to be. The team is required to do a rough paper design of printed circuit board showing that all the necessary components and connectors will fit on a board of that size. The EE team should discuss with the rest of the team the desired shape of the board. For example, the design of the product might require a long rectangular PC board rather than a square one.

It is not required to design or fabricate an actual PC board, or to do the layout of the conductor traces on the board. A full-size 2-D drawing showing how the board might look with all the components in place is sufficient. You can also base your PCB design on using smaller components than we use for the prototypes. For example, we use a 28-pin DIP version of the microcontroller for the prototypes but a production design would likely use smaller surface mount versions of the same chip.

Don't Go Overboard

Teams must remember that they are responsible for coming up with a design that they can build during the time scheduled. Be careful to avoid "mission creep" where more and more features are added to the design until it becomes too bloated to be constructed. Do not allow optional features to get in the way of designing and constructing the baseline project.

Milestones

Each team is required to complete certain fundamental parts of the project and demonstrate that they have them working by specified dates. The purpose of this is to make sure no team gets too far along in the semester without having some very basic parts of the project working.

1. Program the microcontroller: Install the power connector, the oscillator and the microcontroller on the board. Demonstrate that you can program the microcontroller to do something such as make one of the I/O lines oscillate between the high and low states. You can show that it's working by using the oscilloscopes in the lab.
2. Input devices: Put a switch or some other input device on the board and interface it to the microcontroller. Demonstrate that you can sense the state of the switch. This could be by installing an LED on the board and making it go on and off.
3. Output device: Put your selected output device on the board and make it do something to show that your microcontroller can work with it. Your display can be a very useful part of your debugging since once it is working you can use it to output information about what is happening with the rest of the system.
4. Real-Time Clock: Decide on how you will implement the time-of-day clock function needed for the project and get it working.
5. Remote Connection: Show that you can receive data from one or more remote sensors either using a wired or wireless connection.

Specifications

The following are some specifications as to how the project is to be constructed.

1. Most of the project circuitry must be built on a single prototyping board. These are provided in the class and are approximately 6 by 9 inches. If teams choose to build their project on some other board they must first get permission from the instructor.
2. All signal and power connections to the logic board must be made using connectors and wiring that make for a reliable connection. For example, using clip leads to attach to a pin of an IC socket should not be used for anything more than the very first test of the board. A person trying to use your board should be able to see immediately where the power and signals are connected. They shouldn't have to be told to "use a jumper to connect the +5 volts to this pin over here."
3. The main part of the design should be based on a Freescale MC68HC908JL16 or Atmel ATmega168A microcontroller. Projects may use multiple microcontrollers if needed. The use of a different microcontroller must be approved by the instructor. A secondary board used in the product can use other microcontrollers as specified by the team.
4. A microcontroller that needs to be removed from the prototyping board for reprogramming should be installed in a zero-insertion force (ZIF) socket in order to make it easy to remove. When planning where to mount the various IC sockets on the board, make sure to leave extra room for the ZIF socket since it is longer and wider than a normal socket.