White Paper

WP107

Distributed Control of Resource-Limited Systems

***by* Larry Mittag,**
Lead Consultant,
Mittag Enterprises

Introduction

Microprocessors have made tremendous strides towards increasing the capabilities of a wide range of devices that are not traditionally associated with computers. These embedded computer systems have allowed complexity to be transferred into software rather than complex and inflexible dedicated hardware solutions. This approach has proven so successful that microprocessor-based designs have become the default for any number of consumer and industrial devices that must contend with any reasonable degree of complexity.

But increased computing power is only one of the tools available to systems designers. Standalone microprocessor-based systems can also take advantage of communications and the proliferation of cheap sensors to enable system-wide optimizations that can provide real benefits on an even larger scale.

This white paper explores this combination of computing power, communications, and I/O capability as it applies to a specific class of problems. The emphasis will be on cost-effective solutions for relatively simple devices. The solutions discussed do not require expensive and complex 32-bit processors and full operating systems. Instead, they are designed around inexpensive but very capable RabbitCore modules and software from Z-World.

Statement of the Problem

Standalone embedded systems excel at solving purely local problems, but they are generally unable to optimize the use of resources that are shared. The lack of a global context prevents these systems from taking anything but a local approach to optimization. This works very well as long as the common resource is available in abundance, but that abundance holds a cost in terms of the efficiency, scalability, and cost-effectiveness of the overall system.

The specific example we will be using to illustrate this class of problems is that of an irrigation system for a large farm. The resource that we will optimize the use of is the water supply to that irrigation system, or more precisely the water pressure. If too many outlets come on at once, the system pressure will drop and none of the outlets will be able to deliver water efficiently. If too few outlets are active at once, the watering time may extend beyond the optimum time of day to deliver the water effectively.

This problem is a classic trade-off between local and global interests. Each local area has responsibility for getting its crops watered, but that can only happen if they all the areas cooperate globally. On the other hand, if any particular local area bends to the global interest too much, then the crops in that area will suffer.

This problem is simple enough that we can discuss it within the limited confines of this paper, but there is enough complexity to allow for multiple design approaches. We will discuss several of these approaches in detail and show how they apply to the general problem of coordinating access to centralized resources.

Solution #1: Centralized Control

The most obvious approach to coordinating access to centralized resources is to control that access from a central point. This is simple and intuitive enough that we will use it as the baseline implementation, but as we will see it starts to run into real difficulties as the problem scales up.

The Traditional Approach

The traditional approach is quite simple, and it is in many ways the traditional way of handling problems such as this. One controller is used and programmed in terms of zones, with each zone being hard-coded in terms of duration of watering time. Typically these zones are processed sequentially at one or more start times. These start times can be set in terms of which days of the week they activate and what times during those days they trigger.

In this approach the problem of limited water pressure is handled simply by not letting more than one zone operate at a time. The assumption is that each of the zones is designed within the flow constraints of the available water source.

Sample Implementation

This system is illustrated in Figure 1 below. The single microprocessor-based controller activates a number of relays that trigger valves. Each valve feeds water to a watering zone. The system can be thought of as a state diagram, with a relatively small and inflexible number of states. The possible stimuli to the system are as follows.

1. Start watering a zone (comes either from a timer or a direct user input)
2. Stop watering (again, from a timer or a user input)

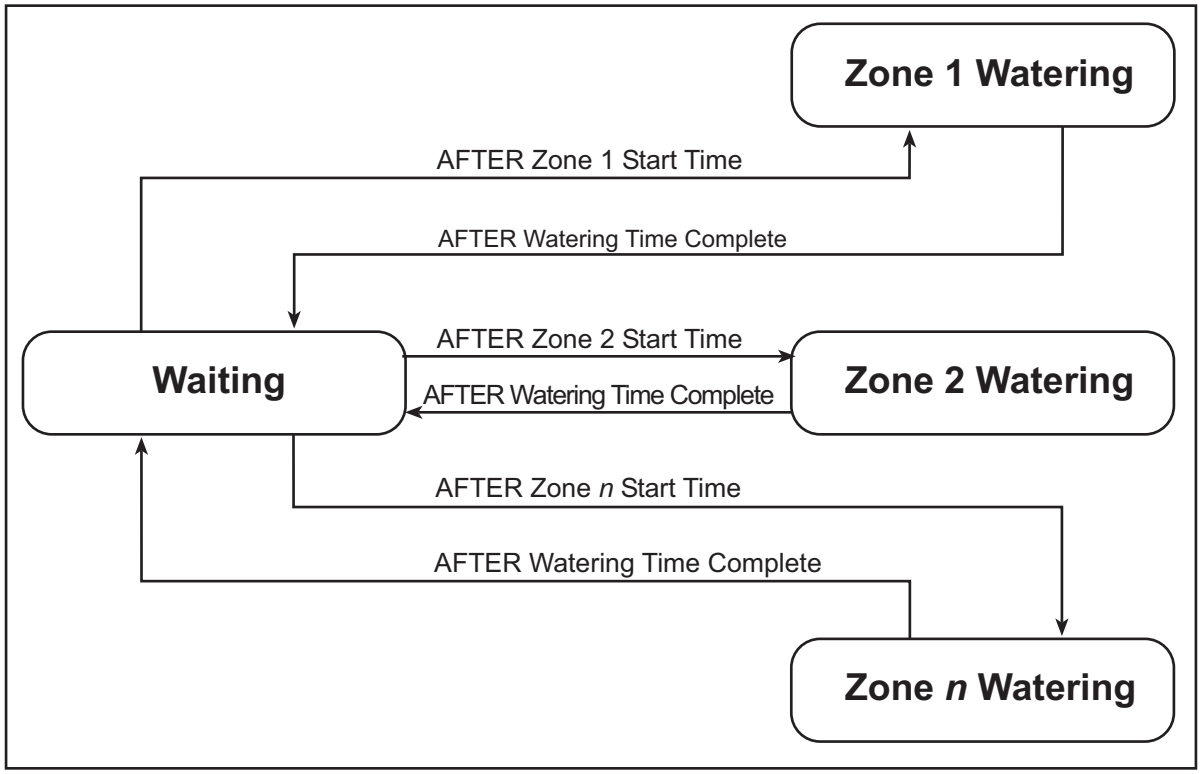

Figure 1. Traditional Solution

The states that the system can be in are also very simple. These are the states.

1. Watering a single zone
2. Waiting

Granted, there is some complexity built into the user interface that comes into play while actually programming the device, but let's set that aside for the moment. Our major concern is what the device will be doing during the other 99% of the time.

Practically speaking, it is unlikely that we would be building a custom version of this controller. They can be purchased at any number of retail stores for less than $50 today and put together into a working system by a typical handyman.

Strengths of This Approach

As mentioned above, this setup is very simple. Any number of home watering systems can be set up with these controllers, as are many commercial installations. In a very real sense, this is the accepted “right” way to solve this problem. As long as the size of the system is reasonable, this common wisdom would hold and the problem is solved. But what happens if we push the parameters of the problem a bit?

When Does It Begin to Break Down?

Any time that “everybody knows” something, it is probably time to reexamine it. This solution is quite adequate for small systems, but it doesn’t scale well. Five zones, each watered for half an hour in a day, would work quite nicely, but expand the number of zones to 50 and a very basic problem shows up. The total watering time now exceeds the length of a day! In addition, all the valves have to be wired to a central controller, which increases wiring costs.

The primary weakness of this system is the fact that it is only watering a single zone at a time. This forces moderation in terms of using the water pressure, but it inherently assumes that the zones are sized correctly to use (and not overuse) the central resource efficiently. This assumption is what does not scale well.

One simple way to solve this problem is to expand the size of each zone rather than adding additional zones. That would be fine if all the areas within the now-larger zones required the same watering time, but what if that were not the case? Now we are restricting the system artificially to the capabilities of our solution rather than building a solution that completely meets the requirements. We end up with a system that does not adapt well to the realities of the problem it is trying to address.

What if instead we simply add more sprinkler controllers to the system? That would enable us to have more than one zone active at a time and therefore keep the overall watering time at a reasonable level. This solution works nicely unless some combinations of the zones exceed the supply capability of the water pressure. In this situation, the lack of coordination among the sprinkler controllers leads to some of the zones not receiving the amount of water they need because the capacity of the central resource is being exceeded. The basic approach of increasing the number of valves is right, but we need to add some more intelligence to the system to coordinate the action of the independent controllers.

Solution #2: Distributed Control with Local Feedback

One way or another we need to add coordination to the system. We might be able to work out a solution in advance by adding this coordination in a static fashion, but what might be a valid solution on one day might not work on the next day when the available water pressure is a little lower. If we try to plan for the lowest value that water pressure will reach in a worst-case scenario, we start approaching the case again where the watering time exceeds a full day. One way or the other, the complexity of a preconfigured solution can be overwhelming and inflexible. This is a situation that calls for adaptive control and feedback.

Sensors Allow Intelligent Adaptation

The first step is to add feedback into the system. The easiest way to do that in this case is to add pressure sensors on the supply side of each valve. These pressure sensors don't have to be expensive devices that are calibrated, but can instead be simple on/off indicators that show the presence of a nominal amount of water pressure. These simple sensors can be combined with intelligent management to create a system that is much more adaptable to what is happening around it.

Care should be taken here, however. While it is certainly possible to add an impressive array of sensors to measure the local soil moisture content or even the growth rate of the crops, these measurements can quickly turn our practical system into an expensive, unmaintainable research project. We will see later that it is quite possible to gain much of the same advantage by using the centralized data resources in our system.

Moving Intelligence to the Decision Point

The second step is to add enough intelligence at the local controller to allow the simple sensor to be used effectively. This intelligence does not have to be a full PC running a sophisticated operating system with a keyboard, mouse, and video monitor, but can instead be a relatively simple controller. In fact, there are real advantages to using a controller such as the RCM3200 with access to various serial ports, plus on-board Ethernet, or the RCM3400 with on-board A/D for reading sensor values in an application like this. Not only is it much less expensive than a PC, but it also uses a lot less power. My assumption is that a distributed application like this is a very good candidate for solar power, which is much easier and less expensive with the smaller amount of hardware in a small microcontroller-based solution.

Sample Implementation

The addition of sensors and increased intelligence allow us to make a local controller that can begin to adapt to changing conditions. Specifically, the simple on/off list of states now becomes the following list.

1. Waiting to water
2. Watering
3. Ready to water when enough pressure is available

The programming on the local processor now can take advantage of the sensor input to override the hard-coded start times for each of the zones. When the time to begin watering a zone arrives, the processor can first check for adequate pressure on the line. If there is not enough pressure, the watering schedule can be postponed until there is enough pressure.

The biggest change in the design is that the addition of the pressure check now allows us to treat each zone independently. The previous design enforced a sequential aspect because of the assumption that the water pressure could be overtaxed when more than one zone was active at a time. Now, in essence, each zone can manage itself independently. The revised design shown in Figure 2 is that of a zone timer object, some number of which can be resident on each controller.

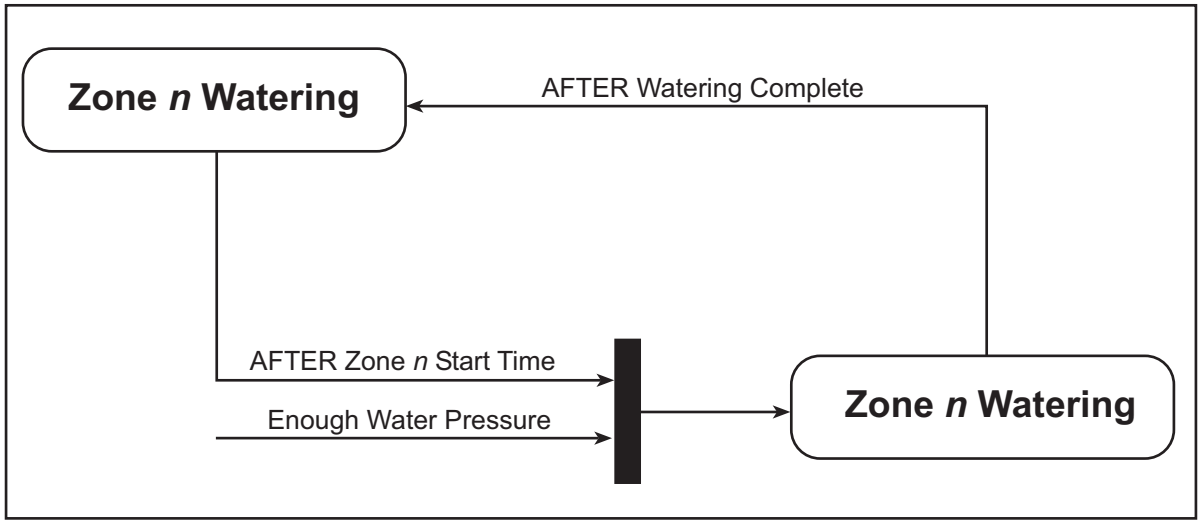

Figure 2. Zone Control Object

There are some interesting feedback-loop implications that must be addressed as well. What if there are two or more controllers waiting for enough water pressure to start watering? Once the current controller shuts off a zone, there could be several new contenders turning on at once, causing a severe drop in pressure. The best way around this problem is to build in a delay, an amount of time during which adequate pressure must be present before a controller can activate a zone. Note that this time should vary from controller to controller, or else you simply postpone the problem. This variance can either be static (giving us a hard-coded higher priority for controllers with the shortest times) or random.

Interestingly enough, this is exactly how network communications are arbitrated on a shared-Ethernet segment. Data collisions in that environment are very similar to the pressure-overuse problem we have in this scenario.

Strengths of This Approach

Our system now will compete for access to the central resource, but it will do so within a framework of rules that we have defined. In essence, we have built a degree of consideration into the system. Each individual controller can now fulfill its local requirements in the context of the availability and limitations of the central resource.

This system can also adapt to a degree of changing conditions. If the water pressure changes from day to day (or even within a day), then the watering pattern will adjust to maintain the minimum pressure that is required. In fact, if the pressure drops to such a point that it will not support watering at all, the system will postpone operating at all.

This system will also allow “rude” controllers to operate under a different set of rules. For example, a simple controller such as the ones described in Solution #1 will operate at will, causing the other more considerate controllers to adjust around it. The best part is that the controllers will not be prone to “road rage,” which is one of the ways that people deal with such deviant rude behavior.

When Does It Begin to Break Down?

There is something truly pathetic about the sight of a sprinkler running in the middle of a rainstorm. We have given our system a larger context, but it still does not have an understanding of the fact that water requirements are not static from day to day. Likewise, a blisteringly hot day may call for an increase in watering time for crops that are sensitive to heat, but that call won't be heard by our new and improved sprinkler controllers. This has implications for the use of our central resource, since these conditions can still lead to nonoptimum use of the resource.

Solution #3: Distributed Control with Centralized Coordination

The only way for our system to gain a truly global context for its decisions is to connect it to a global source of data. In a very real way this will allow it to "think globally and act locally," as the saying goes.

This is not as far-fetched or complex a concept as you might think. The recent advances in wireless communications allow us to build reliable capabilities into even the simple systems we have been discussing here. Standards such as 802.11 and cellular data networks provide the "wiring," while those built around TCP/IP and XML provide the capability for meaningful and efficient M2M (Machine-to-Machine) communications. These standards provide the infrastructure that makes it possible to build even niche applications that make technical and economic sense.

Networked Local-Area Communications

The first step is the organization of the independent controllers into a LAN (Local Area Network). In Solution #2 the controllers could only communicate indirectly by measuring the water pressure. If that was too low, they knew that someone else was using that resource. If the controllers can communicate with the other controllers, they can schedule even more efficiently since they don't have to guess whether or not they are next in line for access to the central resource.

In the past this would have meant stringing wire all over the field, a scenario that would have probably doomed the effort from the start. Now the network can be connected wirelessly through 802.11b protocols that make that wiring nightmare obsolete.

Centralized Control Through Wide-Area Communications

Local area connectivity won't give us that truly global perspective, however. For that we will need to connect our local network to a larger internet. This may or may not be the Internet (note the capitalization), but it certainly should provide access to information that cannot be gleaned from the local field. This information may be as simple as an up-to-date programming capability or as complex as the latest weather predictions and information as to the weather that is currently happening. If it is about to rain (or even raining right now), why should we bother running the sprinklers?

Note that this can be a bidirectional path as well. The farmer might be very interested in the performance of his watering system. Low pressure on a particular zone of the watering system could mean a broken pipe. Likewise, it might be a good idea to readjust the schedule if some crops are not being sprayed at the right time during the day.

Sample Implementation

We are still not talking about putting full-blown PCs into the field. The same RCM3200 modules we used in Solution #2 can be employed here with the addition of some communications options. Adding a wireless Local Area Network can be done simply by adding a Compact Flash (CF) slot to the controller and using off-the-shelf 802.11b (Wi-Fi) CF-based cards. Cellular-based modems are simply interfaced using an RS-232 port. One of the strengths of using Rabbit-based core modules is that they provide for a variety of I/O options. For example, there are 6 serial ports on the Rabbit processor, which can be configured for RS-232 as well as an 8-bit parallel I/O bus, which makes it easy to interface to buses such as the Compact Flash bus with as few components as possible.

Two application kits from Z-World, the M2M and the Wi-Fi Application kits, provide reference designs for both of the these solutions. Besides a complete software solution, including device drivers for GSM cellular modems and PRISM-based Wi-Fi cards, TCP/IP stack, and SMS libraries, these kits provide a Wi-Fi card with Compact Flash adapter and a GSM modem, including all the cables and the antenna needed. In the past, a communications network like this would have required a significant amount of network-specific system-level programming. Fortunately, the same TCP/IP protocols that power the Internet can be brought to bear on even a network as small and specialized as this. Software options for the RCM3200 include support for both GPRS/GSM and Wi-Fi that interface to the MAC layer of the TCP/IP stack. There are also options to support communications protocols such as SMS that can be used for alarm messaging. If none of these fits your application, there is also support for PPP through whatever device can be attached to one of the many serial ports provided by the core module.

The network implementation for this particular application is illustrated in Figure 3 below. This network can either be created as a separate standalone intranet, or it can be interfaced to the Internet through one of the gateway options shown. The choices are to either form a strictly local network through a combination of Ethernet and Wi-Fi, or to route traffic from the field through a GPRS connector, which can connect via a carrier data route to a local or remote server through the Internet.

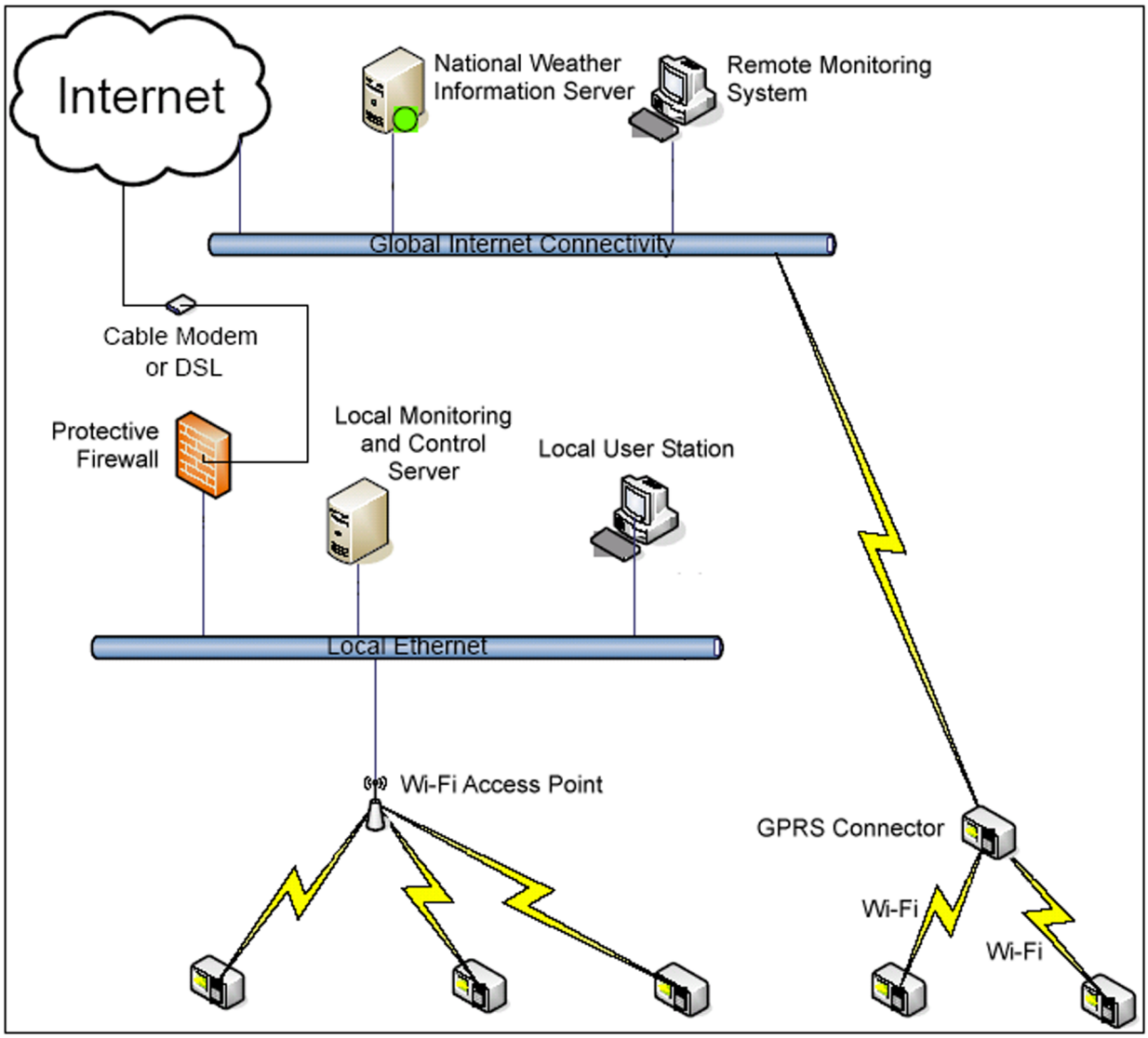

Figure 3. Network Connectivity

The programming changes in the RCM3200 modules vary from being relatively straightforward to being as complex as necessary. At this point, data from any number of sources can be integrated into the system. The watering schedule can be updated based on the current and predicted weather, as well as the specific plant needs for a particular point in their growth cycle. These changes and information could literally come from any point on the planet, and they can be validated using any number of protocols commonly in use on the Internet. At this point, the watering system could literally be part of the worlwide Internet.

How Far Can This Scale?

In a very real way we have come full circle. We started out with centralized control, which we decentralized. We then gathered our decentralized system around a new center, which gave us the tools to effectively manage a larger network of controllers. This is exactly the cycle that is happening in any number of command and control systems the world over.

This specific solution can scale to some point, at which time it will be necessary to build decentralized versions of it in parallel. At some point the tools will be available to centralize control of those parallel systems at a higher level of abstraction. This cycle will repeat until the perspective of the system is as high as it can be to still derive value.

Conclusion

The problem we have illustrated here is certainly not revolutionary, but the techniques we used to solve it have wide applicability. In a very real way, computing power and communications are the Ying and Yang of the solutions, each complementing the other.

We should also pay attention to the form in which these dual forces are delivered to the problem. In the not-too-recent past there were those that could not comprehend why any single family would want a computer in their home. If you go back a little further you can find the same sentiment expressed about the concept of a personal telephone. What these people failed to take into account is the evolution of these devices. The computers that we made personal and the telephones that we now carry in our pockets no longer require special air conditioning or power. They have evolved to fit in the environment, just as the controllers and communications described in these solutions evolved to fit the environments of the wider world.

More importantly, control can be truly distributed by using a combination of local area and global networking. A controller can be programmed to handle local issues on the controller while considering environmental conditions that may have been downloaded via the global network. Of course, no system is perfect. In case of errors or problems, the controllers in a distributed system can act on some error conditions locally while sending error and status messages upstream to a GPRS gateway. The user can then download error logs and be alerted to error conditions that may require intervention.

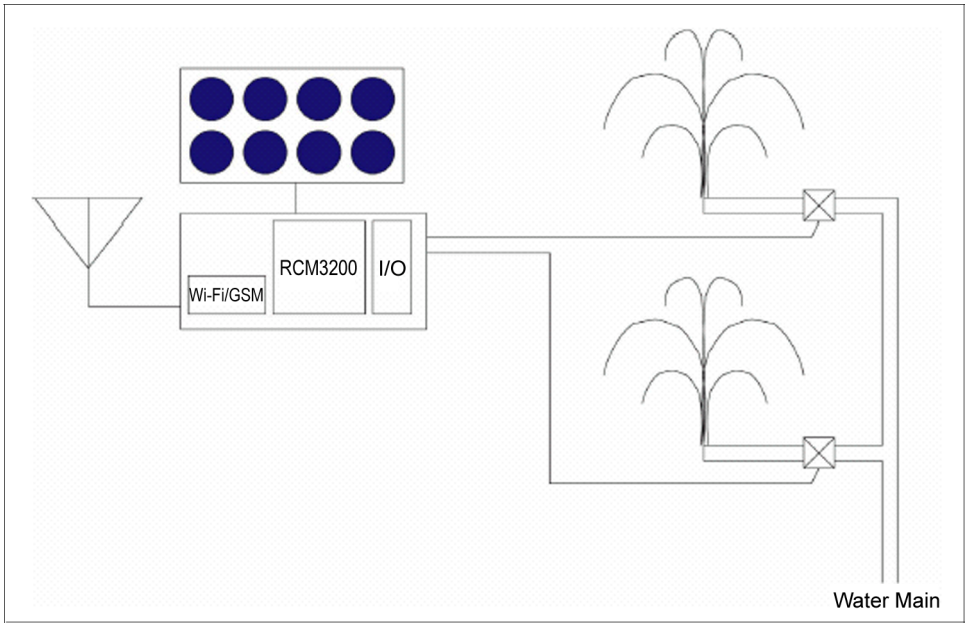

Figure 4. RCM3200-Based System

For example, while the controller can already manage water resources by monitoring the pressure of the supply line, it can now inform the gateway or other controllers if it has not been able water sufficiently. If this is a persistent problem, or if the gateway could correlate that several controllers in the LAN could not water because of low pressure, it can now flag an alarm to the user by sending an SMS message or e-mail via the cellular network. Since the gateway most likely also has been programmed to keep a log of warning and errors, the user can now diagnose the problem in detail and dispatch someone to fix a specific problem.

All of these capabilities are available today, and are a taste of things to come as we become a society of wiressely networked devices. In many countries the number of cellphones is approaching the number of inhabitants. For more information on RabbitCore modules and the application kits mentioned in this article, please visit the Z-World/Rabbit Semiconductor Web sites (www.rabbit-semiconductor.com)/(www.zworld.com).

About the Author

Larry Mittag is a contributing editor for the ***Embedded Systems Programming*** magazine, and is a member of the advisory board for both the Embedded Systems and the Communications Design conferences. He is also the lead consultant for Mittag Enterprises, to which he brings his quarter century of experience in embedded and wireless systems.

Z-World, Inc.

2900 Spafford Street
Davis, California 95616-6800
USA

Telephone: (530) 757-3737
Fax: (530) 757-3792

www.zworld.com

Rabbit Semiconductor

2932 Spafford Street
Davis, California 95616-6800
USA

Telephone: (530) 757-8400
Fax: (530) 757-8402

www.rabbitsemiconductor.com