Irrigation Leak Detection
Using Flow Rate Sensors to Detect Breaks in an Irrigation System

by Adam Openshaw and Kalvin Vu
Spring 2010

A Senior Project
presented to the Faculty of the Computer Engineering Department,
California Polytechnic State University, San Luis Obispo
In Partial Fulfillment of the Degree
Bachelor of Science

Abstract

This report details the exploration and invention of a mechanism that can detect breaks in an irrigation system. With the ultimate goal of conserving water, we have designed an inexpensive, self sustaining flow rate monitor that can be used to identify anomalies in an irrigation system and wirelessly communicate the status of the system to a base station. Our implementation can also determine the direction of the anomaly based on the nature of the anomaly itself. This means that multiple monitors can be used to pinpoint the location of a break within the system in addition to merely detecting its existence. The results of our research have indicated, however, that our flow rate sensor concept is not combatable with a water-based environment.
# Table of Contents

Introduction  - - - - - - - - - - 4  
Motivation  - - - - - - - - - - - - - - - 4  
Summary  - - - - - - - - - - - - - - 4  

Project Overview  - - - - - - - - - - - - - - - - - - - 5  
Design Decisions  - - - - - - - - - - - - - - - - - - - 5  
Ideas for Sensors  - - - - - - - - - - - - - - - - - - - 5  
Heating the Sensor  - - - - - - - - - - - - - - - - - - - 6  
Waterproofing the Coil  - - - - - - - - - - - - - - - - - - - 7  
The Microcontrollers  - - - - - - - - - - - - - - - - - - - 7  
Power  - - - - - - - - - - - - - - - - - - - 8  
Testing  - - - - - - - - - - - - - - - - - - - 8  
Design Overview  - - - - - - - - - - - - - - - - - - - 9  
Concept Fan  - - - - - - - - - - - - - - - - - - - 9  
Components, Parts, & Costs  - - - - - - - - - - - - - - - - - - - 11  

Implementation  - - - - - - - - - - - - - - - - - - - - - - - - - 20  
Microcontrollers  - - - - - - - - - - - - - - - - - - - - - - - - - 20  
Transmitter  - - - - - - - - - - - - - - - - - - - - - - - - - 20  
Receiver  - - - - - - - - - - - - - - - - - - - - - - - - - 21  
Sensor & Coil  - - - - - - - - - - - - - - - - - - - - - - - - - 22  
Wireless Communication  - - - - - - - - - - - - - - - - - - - - - - - 24  
Power  - - - - - - - - - - - - - - - - - - - - - - - - - 25  
Output  - - - - - - - - - - - - - - - - - - - - - - - - - 26  
Software  - - - - - - - - - - - - - - - - - - - - - - - - - 26  
    Wireless Comm.  - - - - - - - - - - - - - - - - - - - - - - - - - 26  
    Sensor Readings  - - - - - - - - - - - - - - - - - - - - - - - - - 27  
    Output  - - - - - - - - - - - - - - - - - - - - - - - - - 30  

Evaluation  - - - - - - - - - - - - - - - - - - - - - - - - - 31  
Testing  - - - - - - - - - - - - - - - - - - - - - - - - - 31  
    PVC Apparatus  - - - - - - - - - - - - - - - - - - - - - - - - - 31  
    Acquiring Useful Data  - - - - - - - - - - - - - - - - - - - - - - - - 36  
    Analysis  - - - - - - - - - - - - - - - - - - - - - - - - - 42  

Related Work  - - - - - - - - - - - - - - - - - - - - - - - - - 44  

Future Work  - - - - - - - - - - - - - - - - - - - - - - - - - 46  
    Microcontroller Porting  - - - - - - - - - - - - - - - - - - - - - - - - - 46  
    Packaging  - - - - - - - - - - - - - - - - - - - - - - - - - 46  
    Increase of Scale  - - - - - - - - - - - - - - - - - - - - - - - - - 46  
    Upgrade Heating Coil  - - - - - - - - - - - - - - - - - - - - - - - - - 47  
    Temperature Sensor Mold  - - - - - - - - - - - - - - - - - - - - - - - - - 47  
    Additional Notification Methods  - - - - - - - - - - - - - - - - - - - - - - - - - 48
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Water Flow Rate Sensor</td>
<td>48</td>
</tr>
<tr>
<td>Upgrade Solar Panels</td>
<td>48</td>
</tr>
<tr>
<td>Interface to Irrigation System</td>
<td>49</td>
</tr>
<tr>
<td>Air Bubble Formation</td>
<td>49</td>
</tr>
<tr>
<td>One Way Valve</td>
<td>50</td>
</tr>
<tr>
<td>Conclusion</td>
<td>51</td>
</tr>
<tr>
<td>Review</td>
<td>51</td>
</tr>
<tr>
<td>Level of Success</td>
<td>51</td>
</tr>
<tr>
<td>Closing Statements</td>
<td>51</td>
</tr>
<tr>
<td>References</td>
<td>52</td>
</tr>
</tbody>
</table>
Introduction

Motivation:

Every year it is becoming increasingly more important to conserve water and energy. Irrigation systems account for up to 60% of all the world’s fresh water withdrawals and California, in particular, uses by far the most water for irrigation of any state in the nation (22% of total irrigation withdrawals, Idaho in 2nd with 12%)[6]. It is, therefore, incredibly important for irrigation water to be used efficiently. Any breaks in large irrigation systems that go unnoticed for extended periods of time could potentially waste devastating amounts of fresh water, as well as time, labor, and money. It is clear that detecting and locating breaks in these systems quickly and efficiently is desired, if at all possible, and the motivation for our project is realized.

Summary:

Currently, the only way to locate breaks in an irrigation system is through visual observation or by using expensive flow monitoring equipment. As the size of the system increases, the ability to do this becomes increasingly difficult and time consuming. Our goal was to find a way to do this task inexpensively with a microcontroller-based monitoring system.

The only obvious way to determine if a break has occurred in an irrigation system is to measure the flow of the water. Because current flow monitoring devices are extremely expensive (in excess of $500), our task was to develop an inexpensive way to electronically monitor the rate of flow within the pipes at any given time. We are less concerned with accuracy of measurement, and more concerned with detecting large abnormalities. Thus, our implementation can be much simpler. Unfortunately, no easily available or affordable sensors have been created with this purpose as far as our research has shown, and so it was on us to create our own flow rate sensor.
Design Decisions – Ideas for Sensors:

We originally had several ideas for flow rate sensors based on different mechanical concepts. One was to have some sort of turbine based sensor with the ability to count revolutions of the turbine. Based on the speed of the turbine, we could calculate the flow of the water. This could have proved to be a very accurate sensor; however, we determined that the nature of a mechanical sensor such as this was harder to build, harder to install, and would be much more prone to mechanical failures than a sensor with limited moving parts.

Another idea for a sensor was to have a bobber in a pressurized column that would raise or lower based on the flow of the water. However, designing a sensor like this looked to be even more complicated than the turbine idea, and we weren’t even positive the concept would work as we anticipated.

An additional option was to have some sort of vertical lever inside the pipe that would become increasingly depressed as the flow rate of the water increased. The concept was simple, but we couldn’t foresee an easy way to waterproof a moveable joint. As a result, we were not ready to commit to this idea either.

We then came across a method used in compressed gas systems that we thought may theoretically also work with water in which a heated temperature sensor takes readings within the pipe. As the flow within the pipe increases, more heat is dissipated away from the sensor, lowering the value of its reading. Not only is this method mechanically simple due to the absence of any moving parts, but it is also very affordable, since temperature sensors are inexpensive and abundant.

Because our system would ideally contain many sensors, we decided to use the temperature based sensor concept. We found a small, cheap temperature sensor called the TMP36. It has 3 leads;
one for \(V_s\) (source voltage), one for \(V_{out}\) (data), and one for GND (ground). This sensor can easily be powered and read by a small microcontroller, so it was perfect. The next step was to figure out how to effectively heat the sensor, without using too much power.

**Design Decisions – Heating the Sensor**

Now that we had established our plan to use a heated temperature sensor, we needed to figure out how to generate the heat we would need. It soon became obvious to us that the easiest and most affordable way to create a heating element would be to simply create a coil out of a metal and run an electrical current through it. The resistance of an element in a circuit is proportional to the heat it gives off, so we mainly needed to find a metal with a suitable resistivity.

We initially tried creating the coil out of copper wire by wrapping it repeatedly around a small nail. However, the copper proved to have much too low of a resistance to generate much heat. After hunting around for a while we came upon some 28 gauge (.41mm diameter) steel wire that we purchased from ACE Hardware in Los Osos. Steel, being roughly an order of magnitude more resistive than copper, was much more suitable for a heating coil.

Having our sensor and our coil, the next task would be to waterproof the two together somehow, but still ensuring that no electronic connections would be shorted and heat could still be dissipated. So we began the search for a specialized epoxy that would satisfy our needs.
**Design Decisions – Waterproofing the Sensor & Coil**

Being able to heat the sensor was one thing, but being able to have the heating element and the sensor submersed in water within a pipe is an entirely different story. We needed the bundle to be waterproof, that much was obvious, but we also needed to make sure that heat could be efficiently dissipated as well. In addition, we also needed to ensure that no electronic connections would be compromised once coated in the waterproofing substance.

We shopped around looking for an epoxy with such specialized functionality, but were unable to locate anything of this nature at any local hardware stores. We eventually resorted to browsing the web for our mystery epoxy and ended up discovering this excellent company called OMEGABOND.

OMEGABOND had a specialized epoxy in stock called OB-101; a heat conductive, electrically insulating epoxy that adheres to most surfaces and cures at room temperature. This was absolutely perfect for our implementation. So, we purchased two packages of the two-part epoxy and began thinking about microcontrollers.

**Design Decisions – The Microcontrollers**

For our design we knew we needed at least 2 microcontrollers. One that would monitor the temperature sensor bundle, and one that would be a base station that would send and receive signals from the monitoring sensors regarding the status of the irrigation system. We originally planned to use some small, cheap microcontrollers with built in radio transmitters we found online, but diverted to using two Arduino Duemilanove hobby boards.

The reason we ended up using the Arduino boards was due to the fact that we would have needed to mount the other microcontrollers, get a programmer board in order to program them, and have to figure out how to interface the radio receiver peripherals. The simplicity of using an Arduino hobby board would allow us to focus on the more important aspects of our project.
Design Decisions – Power

Having many sensors located at various places in the irrigation system is ideal. However, running power lines out to each individual sensor is absolutely out of the question. Therefore, we knew it would be in our best interest to make the sensor and transmitting microcontroller rig self-sustaining.

We ordered some solar panels from a hobby engineering store online that we felt could power our microcontroller. It was only after we started to test our heating coil that we realized that the two solar panels alone would not provide enough current for the coil and the microcontroller together. As a result, we bought 8 rechargeable AA batteries. Our plan was now to use the batteries as a main source of power for the microcontroller and coil, and use the solar panels as a means to charge the batteries in the down time. This we felt should work since most irrigation systems only run once per day, so we would have lots of time to charge the batteries back up to capacity after we take our readings.

Design Decisions – Testing

We knew it wasn’t practical, or wise, to try to test our sensor on an already existing irrigation system, so we decided to build a rig capable of testing the various conditions we hoped to be able to detect. The three situations we would be testing for were: some flow (normal operating conditions), no flow (a break upstream of the sensor), and a lot of flow (a break downstream of the sensor).

In order to test all three of these conditions in the same environment, we designed a PVC pipe apparatus that had a sprinkler, two valves, and a hose joint (shown in Figure 21, page 32). Using this design, we could replicate normal conditions (both valves closed), a break downstream of the sensor (downstream valve open), and a break upstream of the sensor (upstream valve open). We would then just need to drill a hole in the pipe and epoxy our sensor into the pipe securely before taking readings.

Now that we have covered all of these design decisions, our complete design concept comes together nicely. In the next section, we provide a fan-out and comprehensive overview of our design.
Design Overview – Concept Fan:

Figure 1: The image above is the concept fan for our design. It encompasses the entire project and illustrates each aspect’s relationship with the general architecture.
The concept fan illustrated on the previous page is a wonderful way to visualize the project in its entirety. There are two main aspects to our implementation, hardware and software. The hardware portion of our project constitutes most of the work, seeing as how the Arduino Duemilanove boards make writing the software quick and painless.

The three main components of the Hardware section include the two proto-boards, and the power circuitry that powers the coil as well as the transmitting microcontroller. The schematic for our hardware design and power circuitry can be viewed in Figure 18, page 20. Our power circuitry would ultimately rely on the ability for the solar panels to recharge the battery array during the downtime.

The software portion of our design is very straightforward. We planned to simply heat the coil for a certain period of time, monitor the value that the temperature sensor outputs, and use those values intuitively to calculate the conditions within the pipe.

More detailed information regarding our implementation can be read in the Implementation section, beginning on page 20. Before we go into the implementation details, we are going to provide a comprehensive list of the components and parts used to create our project, as well as their costs.
Design Overview – Components, Parts Required, and Costs:

**Electronic Components (Total Cost = $146.61):**

*Microcontroller(s):*

The microcontroller boards we used are called Arduino Duemilanove. Arduino is an open-source electronics prototyping platform and these boards are very convenient to use. We purchased two of them for this project; one to monitor the sensor, and one to receive wireless communication.

*COST: $29.95 (x2) = $59.90*

*Temperature Sensor:*

Important characteristics of a temperature sensor for our application were that it be small, cheap, easy, and efficient. The sensor we found is called the TMP36. It fit every single one of our needs and performed admirably.

*COST: $1.10 (x4) = $4.10*

![Figure 2: Arduino Duemilanove Proto-board](image)

![Figure 3: TMP36 Temperature Sensor, and Sensor+Coil Bundle with Thermal Epoxy](image)
**RF Transmitter:**

The RF Transmitter is what we use to transmit wireless serial data from the sensor monitoring board to the other. We chose to use the WRL-08946 434MHz RF Link Transmitter because it is small, cheap, and compatible with our Duemilanove boards.

![Figure 4: WRL-08946 434Mhz RF Link Transmitter](image)

*COST:* $3.95

**RF Receiver:**

We also purchased a receiver that would be able to read the wireless data being transmitted by the transmitter shown above. We chose to use the WRL-08950 434MHz 4800bps RF Link Receiver for the same reasons as the transmitter.

![Figure 5: WRL-08950 434MHz 4800bps RF Link Receiver](image)

*COST:* $5.95
Power Relay:

In order to control the heating coil with software, we needed to have a mechanical switch that we could somehow flip using the Duemilanove. We looked into using MOSFET’s with high enough current ratings, but after much research we came across the perfect device. A power relay is an electrically operated switch, meaning we could use a small voltage signal from the Duemilanove to switch the coil on or off. We purchased a 5V, 1A power relay from RadioShack.

![Figure 6: 5V 1A Power Relay](image)

COST: $4.69

Batteries & Holders:

Choosing the appropriate method to power our design is of high importance, and some things still need to be done (see Future Work). We needed to have enough power to heat our coil sufficiently, so we bought eight rechargeable AA batteries and two series holders.

![Figure 7: Rechargeable AA Battery and Series Holder](image)

COST: Batteries-$12.49 (x2) = 24.98, Holders-$2.41 (x2) = $4.82
Solar Panel(s):

As we mentioned earlier, running power lines all the way out to various places in an irrigation system is impractical. So, we decided it would be a good idea to find some relatively cheap solar panels that we could use to charge the batteries in the down time. We found some nice panels from HobbyEngineering.com, rated at 6.7V 30mA. We purchased two, and planned to connect them in series with a diode in order to recharge the batteries. Using these, we hoped our implementation would be self sustaining, eliminating the need to run power cables out to the sensor. They are also relatively small, 37mm x 66mm.

![Solar Panel Image]

**COST:** $10.79 (x2) = $21.58

Power Adapter:

Because we also wanted to power the microcontroller with our battery array, we needed a specific adapter. RadioShack had the one we needed, so we picked it up.

![Power Adapter Image]

**COST:** $1.65
**Steel Wire (Heating Coil):**

The material we ended up using for the heating coil was 28 gauge galvanized steel wire. We found it at a hardware store in Los Osos. We created the coil by wrapping it repeatedly around a small nail. The final coil was approximately 8” and yielded 3Ω of resistance.

![Figure 10: Galvanized Steel Cable and Partial Heating Coil](image)

*COST: $4.84*

**Resistor:**

In order to manage the flow of current through the heating coil, we purchased a few 1Ω, 10W resistors from RadioShack. Using these, we could fine tune our circuit to allow for sufficient heating of the coil, while maintaining good power efficiency.

![Figure 11: 1Ω, 10W Wire-wound Resistor](image)

*COST: $0.99*
**Diode:**

Because the solar panels are connected in parallel with the batteries, we need to insert a diode into the circuit. We do this in order to prevent the power from the batteries from reaching the solar panels and causing irreparable damage.

*COST* = $0.12

![Figure 12: Diode](image)

**Solder:**

Many electrical connections needed to be made, which means we needed to use solder. We bought some .015”-Diameter Silver-Bearing Solder from RadioShack to use for making these connections.

*COST:* $5.49

![Figure 13: .015%-Diameter Silver-Bearing Solder](image)

**CAT6 Cable:**

The leads coming off of the sensor-coil bundle were much too short to be able to reach our board, so we purchased some CAT6 cable from Home Depot. We soldered the leads onto the wires in the cable and color coded them for our reference.
**Heat Shrink Tubing:**

We purchased some one half inch heat shrink tubing from Home Depot to protect our exposed leads connecting the sensor and coil to the CAT6 cable.

*COST: $1.95*

**Adhesives and Epoxies (Total Cost = $22.99):**

**OMEGABOND OB-101 Thermal Epoxy:**

Perhaps the most important element of our project was this OMEGABOND OB-101 Thermal Epoxy. It provided us with the ability to waterproof our sensor and coil while maintaining electrical insulation and also allowed for effective heat dissipation.

*COST: $9.00 (x2) = $18.00*
**Marine Epoxy:**

The other epoxy we used is a marine epoxy we purchased from Home Depot. The purpose of this epoxy was to seal the sensor within the pipe. So the epoxy needed to be strong, and waterproof.

![Figure 16: LOCTITE Marine Epoxy](image)

**COST:** $4.99

**PVC Pipe Testing Structure (Total Cost = $35.42):**

The following list contains all of the pieces of the PVC apparatus that was created for testing. An explosion diagram of the apparatus can be viewed in Figure 21, page 32. A picture of the apparatus completely assembled can be viewed in on page 34. Each piece and its cost are given below.

- **1”x10’ PVC Pipe:** - - - - $1.57
- **1” Slip PVC Valve:** - - - - $5.15 (x2) = $10.30
- **Hose Attachment & 1” Adapter:** - - $0.54
- **1” Slip PVC T Joint:** - - - - $0.57 (x2) = $1.14
- **1” Slip PVC I Joint:** - - - - $0.41 (x2) = $0.82
- **90 Degree PVC Ell:** - - - - $1.28
- **¼ Circle Spray Sprinkler:** - - - - $2.84
- **Riser:** - - - - - $0.39
- **Teflon Tape:** - - - - - $1.18
- **PVC Adhesive:** - - - - - $3.95
- **PVC Scissors:** - - - - - $12.98
Tools (Total Cost = N/A):

The following tools were either owned by us originally or borrowed for the duration of the project. All of these tools were used extensively. Additional tools may help, but these are the bare minimum.

- **Soldering Iron**
- **Multimeter**
- **Power Drill**
- **Wire Cutters**
- **Needle-Nosed Pliers**

Figure 17: Assorted Tools
Implementation

At the heart of our system are two Arduino Duemilanove [1] development boards, each with its own ATmega328 microcontroller on board. These boards include built-in I/O, enabling us to easily interface them to the temperature sensors and each other. One board is fitted with a wireless RF receiver module, henceforth known simply as the “receiver”, and the other is fitted with a wireless RF transmitter module, temperature sensor, and power supply, and will be henceforth known as the “transmitter”.

Transmitter

The transmitter board does most of the work. It is connected to the temperature sensor through a built-in ADC and continually polls it to acquire its current temperature. Since the sensor is analog, it...
continually outputs a voltage proportional to its temperature, making it possible to quickly read data at a high resolution. These temperature readings are used in our software algorithm to determine normal or deviant water flow rate. The transmitter runs this algorithm continually as it acquires temperature data over time. Essential to the temperature sensor is the heating coil which is controlled by the transmitter and powered directly by the power supply. Using a power relay, the transmitter is able to enable or disable the heating coil at will, according to the needs of our software algorithm. This is useful not only in defining the behavior of the temperature sensor, but also in saving power when the system is not in use.

Since the transmitter has a wired connection to the temperature sensor and heating coil, it must be installed in an outdoor location near the irrigation system. A power supply consisting of rechargeable batteries backed up by solar panels allow the transmitter to run indefinitely. It also provides enough power for the heating coil to operate in bursts; the transmitter disables the coil as needed, lowering the overall power consumption.

The connected wireless RF module communicates status updates to the receiver. As the transmitter runs the software algorithm, it will periodically send its perceived status of the irrigation system. When it detects a problem, it will immediately send the appropriate signal to indicate the existence of the anomaly. The module has a range of approximately 500 feet, which is adequate for most small irrigation systems.

**Receiver**

The receiver board has only one purpose: to receive incoming transmissions from the transmitter. It is connected to a wireless RF module that continually receives status updates from the transmitter. Included on the microcontroller board is a single LED that we use to display the status of the irrigation system. When the receiver receives notification of an anomaly, it will use the LED to
convey to the user the existence of the anomaly. This provides a convenience for the user; he/she need not check the transmitter, which will be inconveniently located outdoors, and may simply refer to the receiver to obtain the status of the irrigation system.

**Sensor & Coil**

The analog TMP36 [2] temperature sensor forms the eyes of our system. It constantly outputs a voltage proportional to its detected temperature. By reading the voltages with our microcontroller, we can detect relative water flow rates based off the rise and fall in temperature. The fully constructed sensor is composed of three critical elements: the temperature sensor itself, a heating coil, and a special epoxy that holds everything together.

![TMP36 Voltage Characteristics](image)

*Figure 19: TMP36 Voltage Characteristics*

The temperature sensor itself is small and fragile; it is unsuitable for use underwater. It was necessary to modify the sensor such that not only will it be waterproof, but also thermally conductive (so that heat can freely travel to and from the sensor) and electrically insulated (to prevent shortage when in contact with water). Our best solution was an epoxy made by Omega Engineering that is
specially designed to be thermally conductive and electrically insulated; it also resists acids and solvents (and presumably water). This epoxy forms the basis of the fully constructed sensor and makes it possible to use the sensor for our purposes.

The epoxy also holds the temperature sensor together with another necessary component of the overall sensor module: the heating coil. The coil is made simply from a long strand of coiled galvanized steel wire that is then wrapped around the temperature sensor. While we used steel wire out of convenience, it is possible to use materials with a higher resistivity to obtain better heating. With the coil wrapped around the sensor, we re-epoxied both components together into one package. This allows the coil to take advantage of the properties of the epoxy: heat from the coil travels to the sensor and dissipates into the water more easily, and the heat-producing current running through the coil is preserved.

The sensor also contains other necessary but non-critical components. We attached a long, insulated cable to reach the microcontroller. We found that a CAT5 was the right size and provided more than enough leads to connect the sensor power, sensor data, and coil power for a total of 5 leads. This cable is necessary as the sensor will most likely be installed underground with the microcontroller above ground nearby. We also used heat shrink tubing to help further insulate and contain wires. Finally, once the sensor is placed into the pipe, we used regular epoxy to seal the pipe and hold and the sensor in place. While any non-corrosive epoxy will likely do, we used marine epoxy to ensure that it would not corrode with repeated use. Additionally, we chose not to use the special thermally conductive epoxy for this task as we feared that using too much over a large surface area would cause the sensor to lose too much heat too quickly.
Wireless Communication

Because the microcontroller must be placed within a short distance to the installed temperature sensor, we required a way to send notifications to the user, who may not be present. To avoid long wiring, we decided to implement wireless communication to notify the user of an anomaly or to confirm the status of the irrigation system.

The wireless RF transmitter WRL-08946 [3] and receiver WRL-08950 [4] designed for use in remote controls, wireless mice, and car alarm systems. Only one transmitter module can work at a time, but many receiver modules can listen to a single transmitter module. They both operate at 434 MHz and have an approximately 500 foot range. The transmit power of the transmitter can be increased by increasing the voltage up to 12V; likewise, the receiver's sensitivity can be increased by increasing the voltage up to 5.5V. Both methods will increase the effective range of communication. Both modules also include a pin for connecting an external antenna to increase range. Since the typical 500 foot range is acceptable, and the microcontroller's peripheral power supply provides only 5V, we opted not to attempt to maximize the range. The maximum data rate is 4800 bps; since our range of possible messages is extremely small, and latency is not an issue, this data rate is adequate for our needs.

Both the transmitter and receiver modules use serial communication to interface with a microcontroller. No configuration is needed; any data sent to the transmitter module will be immediately transmitted, and any data currently over the air will immediately be received by the receiver module and sent over the serial line. This enables us to treat the microcontrollers as if they had a wired connection, simplifying our system's configuration.
Power

Our remote transmitter microcontroller will likely be placed in a position without access to the main power grid. Thus, our ideal solution is a sustainable power supply for the transmitter and temperature sensor. Our design combines cheap rechargeable batteries to supply large bursts of current and more expensive solar panels to recharge the batteries during downtime. The solar panels are connected to the batteries with a diode to prevent backwards current flow, which the batteries are connected to the microcontroller’s power input.

The microcontrollers each require 7-12V which is provided by the rechargeable batteries. The microcontrollers in turn provide 5V and up to 200mA to connected components. The wireless RF receiver module uses 2-3mA; the temperature sensor draws only 50uA; and the wireless RF transmitter module uses 20mA. Each of these are able to use the 5V supplied by the microcontroller and use much less combined than the maximum 200mA.

However, the heating coil requires over 1A of current in order to produce sufficient heat, so we connect it directly to the battery pack. Since we need control of the heating coil to feed heat to the temperature sensor at appropriate times, and constantly drawing 1A would quickly drain our batteries, we attach a power relay that we control with voltage supplied from the microcontroller. We use a Single Pole Double Throw (SPDT) relay with the normally-open contact connected to the heating coil (with a 1 ohm resistor to limit current) and the normally-closed contact left open. This simulates a normally-open Single Pole Single Throw (SPST) relay; with no voltage applied to the switch, the circuit stays open and the heating coil is turned off, and with 5V applied the circuit is closed and the coil begins producing heat. Our system is currently configured in this way because we were only able to obtain a SPDT relay; a SPST relay could easily replace it provided the switch is powered by 5V and the relay’s current rating is 1A or higher.
Output

To notify the user of a detected anomaly, we use an LED integrated into the microcontroller. The LED is normally in its off state to indicate normal status of the irrigation system. Turning the LED on indicates that an anomaly has been detected.

Software

Wireless Communication

The wireless RF modules both use a serial connection to communicate with the microcontrollers. Since data is immediately and automatically transmitted/received, the modules act like a wired connection; there is no need to configure the modules, which simplifies the software. However, this also means that the modules do not implement any kind of protocol other than that of serial (start and stop bit, parity bit). Normally this is not a problem, since we only need to implement one way communication with a very small set of possible messages, but the receiver module has an odd behavior: when it is constantly receiving data, it will work as expected, but any silence will cause it to output garbage data.

We work around this limitation by implementing our own simple communications protocol. Every data packet is enclosed within three elements: a preamble, a network address, and a checksum. The preamble is an arbitrary number that indicates the beginning of a transmission. This is followed by the network address, which is also an arbitrary number that indicates the session. Since many receiver modules can listen to a single transmitter module, we can use the network address to choose which module will receive a particular packet, although our system only contains a single receiver. Next is the
data itself, and then a checksum which is calculated by adding the preamble, network address, and data together. This gives some insurance that the packet was received without errors.

Each element, including the data, is a single byte, making the entire packet four bytes long. While the preamble and network address can theoretically be any number, care must be taken so that the checksum does not overflow when being calculated. This problem solves itself, however, since the set of possible messages are limited; choosing sufficiently small numbers and ensuring that each case does not cause an overflow is trivial.

The transmitter sends status updates to the receiver at a constant rate. This allows the user to check the status of the irrigation system regardless of an active anomaly. When the transmitter detects an anomaly, it will constantly send notification until the anomaly is cleared or the transmitter is reset. While the transmitter continually runs our anomaly detection algorithm and sends the appropriate messages, the receiver's only job is to listen for messages and output the status to the LED.

The maximum data transfer rate of the wireless modules is 4800bps; however, our purposes do not require a high rate, nor does it require low latency. Thus, we lower the data transfer rate to a mere 1200bps, which is more than enough. Lowering the data rate also helps reduce transfer errors and increase range, both of which our system can benefit from.

Sensor Readings

The microcontroller uses a simple algorithm that determines the existence of a leak through monitoring the temperature of the sensor package. We do this by measuring the first delta temperature between the sensor's initial temperature and maximum temperature, and the second delta temperature between the sensor's maximum temperature and final temperature. By comparing these deltas, we can infer the status of the irrigation system. In a normal flow state, the sensor will gain heat slowly, but dissipate heat very quickly. In an open backward state, the sensor will gain heat more quickly and reach
a higher max temperature, but will dissipate heat very, very slowly. Thus, if the two deltas are sufficiently different, then we can infer a normal flow state, and if the two deltas are similar, then we can infer an open backward state.

When the system is activated (signaled by the beginning of the irrigation system's active duty cycle), we turn on power to the heating coil for 40 seconds to sufficiently heat the temperature sensor. The beginning temperature (at t=0) is recorded. At t=40, the heating coil is switched off and the sensor package is allowed to cool off. Residual heat from the heating coil will continue to raise the temperature before reaching a maximum. We record this maximum temperature and calculate the first delta temperature. At t=150, we record the final temperature and calculate the second delta temperature. With these deltas, we can calculate the percent difference between them, and depending on that value we infer the state. We define the cutoff percent difference for distinguishing between normal flow and open backward states as 45%. Our tests show a low 20% difference for the normal flow state and a high 70% difference for the open backward state; averaging the two yields 45%.

If an open backward state is detected, a message will be sent wirelessly to notify the receiver of an anomaly. Otherwise, a message to confirm the integrity of the irrigation system will be sent.
Figure 20: Sensor Algorithm Flowchart
Output

The integrated LED is assigned a pin on the microcontroller board and thus can be used like any digital I/O. Setting the pin high will turn on the LED, while setting the pin low will turn off the LED. When the receiver receives the appropriate message, it will set the LED on. Otherwise, the LED is off to indicate normal status.
Evaluation

Testing:

Testing our implementation did not go as expected. While we were able to construct our testing apparatus and collect good readings, there was ultimately a massive flaw with the concept of this project. Water’s ability to dissipate heat is overwhelming. Whether the flow is fast, slow, or nonexistent, the temperature of sensor increases and decreases too uniformly. We will go into further detail regarding the readings, their analysis, and what led us to this conclusion shortly.

PVC Apparatus:

Before we get into the readings, we will show you how we went about testing our implementation. As mentioned earlier, we constructed a PVC apparatus that would allow us to replicate the three conditions of which we hoped to be able to recognize electronically. Shown on the next page is an explosion diagram of the pipe structure. It illustrates how the PVC apparatus was constructed, while allowing you to see each individual component and its orientation.
Figure 21: Explosion Diagram of PVC Apparatus
By altering which valves were open at any given time, we could replicate the three states: normal flow, upstream break, and downstream break. In order to mimic normal conditions, both valves remain closed. To mimic an upstream break or downstream break, the corresponding valve is opened while the other remains closed.

This rig worked wonderfully, despite the fact that the concept at its core is flawed. Testing with the apparatus went smoothly, except for one instance. When we first installed our sensor permanently into the pipe with epoxy and tried to obtain some readings, water bore a hole through a thin layer of epoxy on the inner side of the pipe and it shot up through our cable and all over our microcontroller. Thankfully, we had not plugged in the power to the board or any of the other components. Because of this, our board suffered no damage whatsoever. Once we dried out the components, we were good to go. So we re-applied the epoxy on the inner end of the pipe and from then on everything was fine.

On the next few pages, you can see some pictures of our testing structure, along with our board. Notice the difference between the first images of the Duemilanove and breadboard and the later one. It’s amazing what a little bit of organization can do when it comes to circuitry.
Acquiring Useful Data:

Fortunately, the Duemilanove has the capability of outputting serial data through the USB. Using this, we formatted our output to have a half second timestamp followed by the value read from the temperature sensor. We then took this data and inserted it into Microsoft Excel, which we then used to create graphs of each reading. Once we had multiple graphs for each case, we could analyze them and see how to properly distinguish between the three modes, if possible. We initially used four batteries in series, providing 4.8V to the coil.
The graph shown above is our first reading. We filled the PVC apparatus with water, and then turned the water off. We wanted to see what the no flow condition looked like so we could compare it to the other flow types. You can see the temperature rise up until the 80 timestamp (40s), at which point we turn off the coil. The temperature then begins to decrease back to its original value. We don’t do any conversion from sensor value to temperature, because the relationship between the two is linear.

After seeing what the no flow condition looked like, we began taking readings of the normal flow condition. We had two methods for acquiring readings that we chose to use. The first was to heat the sensor up for the 40 seconds without turning on the water. Then, we would turn the water on at the 40 second mark (once the coil was switched off) so the flow of the water was only affecting the decrease in temperature. The second was to have the coil heat up during the flow, so the flow would affect both ends of the curve.
Normal Flow (No Flow While Heating)

Sensor Value vs. Time (0.5s)

[Graph showing sensor values over time for normal flow with no flow while heating]

Normal Flow

Sensor Value vs. Time (0.5s)

[Graph showing sensor values over time for normal flow]

Series 1
The two graphs above show the readings for the normal flow state using the two techniques mentioned earlier. While there is a difference of 5 between the two starting sensor values (140 and ~145), there is only a difference of 7 between max sensor values. Disappointingly, we do not see a very distinct difference between the two graphs.

After these readings, we opened the downstream valve and took readings using the same technique. Again, opening the downstream valve would replicate a break in the system that lies downstream of the sensor. We hoped that the difference between small flow and large flow would be much more obvious. The graphs are shown below.
Even more disappointingly, the downstream break condition is not noticeably different from the normal flow condition or even the no flow condition. The sensor may heat up a few additional degrees in the no flow condition, but this is by no means enough for us to formulate a reliable algorithm to distinguish these scenarios.

We then connected all eight batteries in series, now providing 9.6V to the coil. We hoped that with the additional current, we could heat the coil up significantly higher. Our idea was that the differences in the graphs would be accentuated at higher temperatures. We also decided to scrap the idea of heating the coil without flow, because initial temperatures of the system would mess with our algorithm. The results are shown below.
Again, the normal flow condition and downstream break condition are practically indistinguishable. Shockingly, however, the upstream break condition appeared to be completely different from the other two. It seemed that we would actually be able to tell if there was no flow in the system when we heat the sensor hot enough due to the slow rate of dissipation seen above.

Much to our dismay, after performing more readings on a separate occasion the results produced above were unable to be replicated. The upstream break readings once again seemed to be homogenous with the other readings. We had an idea of what may have happened, which we will discuss in the analysis on the next page.
Analysis:

According to the data we collected, our implementation was unable to accurately distinguish between the three different states. We attribute this failure in full to the natural characteristics of water. Its immense ability to dissipate heat makes the flow of the water negligible. As long as there is water around our sensor, the rate at which the sensor cools is almost identical regardless of the flow.

With the particular case of the upstream break data that could not be replicated, we deduced that there must have been some sort of air pocket within the pipe that insulated our sensor. The reason we think this is that the same sort of characteristics can be replicated outside of the pipe. After putting this implementation to the test, we can see now why this concept is primarily used in gas-based systems.
Related Work

The concept of using temperature to measure flow rate is currently practiced in industry today using thermal mass flow meters. They use the same principle that we use for our system: as flow rate increases, heat is dissipated more quickly, allowing us to determine flow rate through a temperature sensor. However, these systems are typically used to measure gas; since liquids dissipate heat much more quickly than gas, it is much more difficult to obtain reliable readings when trying to measure the flow rate of water. In addition, the flow rate is often not detected by reading the actual temperature as in our system, but the current or voltage needed to maintain a temperature: as flow rate increases and causes the instrument to dissipate more heat, more current/voltage is needed to maintain heat.

Sierra Instruments produces flow rate meters using a number of different principles, including the principle of temperature flow [5]. Two types of thermal flow meters that they and other industrial manufacturers produce are capillary and immersible meters. Capillary thermal meters use a sensor tube that deviates from the main pipe for a certain distance before reintegrating. This tube must be built and calibrated in a way that maintains laminar flow; in other words, the flow rate in the tube is proportional to the flow rate in the main pipe. Two temperature sensing coils are wrapped around the sensor tube, and changes in temperature are measured through the change in voltage of the circuit. An advantage of this setup is that the sensors are non-invasively wrapped outside the tube.

Figure 22: Capillary Thermal Meter
Immersible thermal meters are more similar to our system. Two temperature sensors are immersed into the gas; one gas temperature sensor measures the temperature of the gas as a reference, and the other velocity sensor has its temperature constantly maintained through a heating element. This results in a constant temperature difference between the two sensors. As gas begins to flow, heat is dissipated from the velocity sensor, causing the meter to increase current in an attempt to maintain the temperature difference. This increased current can be measured to determine the flow rate of the gas. An advantage that this setup has over the capillary thermal meter is that it is possible to measure larger flow rates in harsher environments.

![Figure 23: Immersible Thermal Meter](image)

Our own system draws from some of the concepts used in immersible thermal meters. However, we opted to use a single temperature sensor, hoping that it would be sufficient for our needs since we do not need to know the actual flow rate to detect an anomaly. We also are unable to constantly heat the sensor; while our current system has sufficient power to activate the heating coil periodically, we would need an enormous amount of power to keep the heating coil running most of the time. This also means that we could not measure the current to determine flow rate, as our heating coil would not be constantly activated. Thus our system runs solely on temperature readings.
Future Work

Microcontroller Porting

Our system is currently implemented with two Arduino Duemilanove development boards. While these boards are useable, the power regulators, relatively large number of I/O pins, and LEDs are among the features that are either unnecessary or overkill for our purposes. A more inexpensive and compact microcontroller solution with a DAC and ADC is a possible upgrade. This would considerably shrink the size of the package as well as reduce power requirements.

Packaging

The microcontrollers are currently attached to breadboards that are connected to the wireless RF modules and temperature sensor. The bulky package and unwieldy wiring may be fine for a prototype, but is unsuitable as a final design. An appropriate future task would be to create a circuit board design with a more compact microcontroller. This would give us the ability to create a robust, professional package and possibly mass-produce the system.

Increase of Scale

Our current implementation only uses one temperature sensor; however, we designed our project to be able to be scalable to larger irrigation systems. Adding more temperature sensors increases the resolution of our anomaly detection and gives the user greater precision in locating the anomaly. Our algorithm can be expanded to manage many more sensors; however our wireless protocol
may need to be adjusted (most likely an increased packet size) to handle the increased set of possible messages to the receiver.

**Upgrade Heating Coil**

While our galvanized steel heating coil produces sufficient heat for our temperature sensor, obtaining a material with a higher resistance would both decrease the required current and increase the heat output. Heating coils for appliances such as toasters or hair dryers commonly use Nichrome, a non-magnetic alloy of nickel and chromium, which has a resistance six times that of steel. Materials such as Manganin (with a resistance three times that of steel) and stainless steel (five times) would also be viable. This would greatly increase the power efficiency of our system and enable us to use less rechargeable batteries in order to reduce the bulk. It would also allow our solar panels to more quickly bring the batteries to full charge.

**Temperature Sensor Mold**

We currently construct our temperature sensors by hand. This creates an uneven and inconsistent final sensor and increases the risk of shorting exposed leads. The heating coil is also spun by hand and is wrapped more loosely around the temperature sensor than is ideal. Designing a type of mold or other manufacturing process to apply the thermally conductive epoxy and heating coil to the temperature sensor in a standardized fashion would greatly increase the reliability of the sensor. It would also likely require less epoxy and shrink the size of the sensor package.
**Additional Notification Methods**

While the receiver's LED displays all the necessary information needed to know the status of the irrigation system, the user must continually check it to be informed of a detected anomaly. Another component may be interfaced to add a visual or aural notification system, such as adding a buzzer to inform the user of a change of status, or an LCD to convey additional messages if needed. It is also possible to interface a PC to receive messages or email the user status updates.

**Water Flow Rate Sensor**

Our design decision to use a temperature sensor to measure water flow rate is driven by cost-effectiveness and scalability. The trade-off is in the accuracy of flow rate measurement. Since our design boils down to detecting flow rate in order to detect anomalies, it is possible to replace the temperature sensor with a more effective flow rate sensor. However, such a modification would require rewriting the software algorithm and possibly a more powerful power supply, and would increase the overall cost of the system. This would ultimately decrease the scalability (unless cost and spacial requirements are not considered) but increase the accuracy of the system.

**Upgrade Solar Panels**

With more powerful solar panels, we will be able to reduce the amount of rechargeable batteries needed, substantially reducing the package size. Since our system drains very little power apart from the heating coil, the upgraded solar panels alone would most likely be able to power the system indefinitely. However, a few batteries are still needed for the high current heating coil and for cloudy days.
Interface to Irrigation System

Our current implementation requires that the user activate the system at the beginning of the irrigation system's active duty cycle. This is required to ensure that the system takes temperature readings as water is flowing. In order for a fully automatic system, an interface must be established between our system and the irrigation system. The only information needed is the state of the irrigation system so that our system only activates at the appropriate times. We also may benefit from future information, such as when the irrigation system will deactivate, so that we can take two every active duty cycle: one at the beginning and one right before the end.

Air Bubble Formation

Our current readings indicate that water dissipates heat too easily to be able to accurately and reliably determine water flow using temperature. However, we encountered a case where we could easily tell the difference between the normal state and open backward state: during the open backward state, an air bubble formed by the temperature sensor by chance, giving us drastically different characteristics from that of the normal state. This difference is what we based our software algorithm on, and although it does not hold for most normal cases, if we can ensure the formation of an air bubble around the temperature sensor during the open backward state, we can easily detect an anomaly. In this case, we stray from measuring the flow rate of water to detecting the presence of air in a pipe that otherwise should be constantly filled with water. Since the heat characteristics of air are very different from those of water, this method ensures reliable detection.

One possible solution is to elevate the area of pipe that holds the temperature sensor so that when the pipe is depressurized (in the case of the open backward state), water will recede and create a
pocket of air in the elevated pipe. We can then take readings as usual and detect an anomaly when it happens.

**One-way Valve**

Another solution to water's high heat dissipation is to insert two one-way valves in the pipe around the temperature sensor. During the normal flow state, the pressure will be sufficient to push the water through the valves, allowing fresh water to dissipate heat from our heating coil. During the open backward state, the pipe will depressurize and the valves will prevent water from entering or exiting the temperature sensor’s chamber. Without fresh water drawing heat from the chamber, we expect the stagnant water to heat up much more quickly and dissipate that heat much more slowly, mimicking the characteristics of air.

The choice of valve is significant in this case as we need one that is activated at the correct pressures: the valve must allow water to flow during the pressurized normal flow state and block water flow during the depressurized open backward state. Ideally, it must also be enough of a thermal insulator that heat is not dissipated through the valve and into the adjacent fresh water.
Conclusion

Review:

The ultimate goal of our project is to save water. We hoped to create a sensor that could monitor the rate of flow of water within a pipe. We used easily attainable components, along with our knowledge of electronics, software, and embedded systems to create an embedded system of applying heat to a temperature sensor and monitoring it. We then hoped that the heat we produced would dissipate at variable rates depending on the flow of the water.

Level of Success:

While we were unable to achieve our goal as we had wished, we have learned some valuable information. Water is simply too good of a heat dissipater for this type of implementation to work without additional things being done.

I would like to say that our design was a success in the sense that we did what we set out to do. We devised a sensor system that performed as we expected, even though the results were not ideal. Some aspects are not complete as we have mentioned, and much work can be done in the future, but we have certainly accomplished a lot.

Closing Statements:

We have learned a great deal over the course of this project. The past two quarters have been quite the experience, as we have watched our project grow from a mere concept into a full blown implemented design. There is nothing like creating technology with your own two hands. It is an indescribable feeling to see your work come together and start functioning on its own.
To anyone who is planning to continue where we’ve left off, or if you plan to devise your own system, we hope our research has helped you. You are also more than welcome to contact us regarding any questions you may have with our system. We would also like to thank Cal Poly and Professor Hugh Smith, CPE Director, for the opportunity to undertake this project.

References


