An Automated System to Detect Flash Floods and Alert At-Risk Communities

by

Joshua A. Weaver

Submitted to the Department of Electrical Engineering and Computer Science in partial fulfillment of the requirements for the degree of

Master of Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2005

Author .... [illegible]
Department of Electrical Engineering and Computer Science
August 16, 2005

Certified by .... [illegible]
James W. Bales
Assistant Director, Edgerton Center
Thesis Supervisor

Accepted by .... [illegible]
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

An Automated System to Detect Flash Floods and Alert At-Risk Communities

by

Joshua A. Weaver

Submitted to the Department of Electrical Engineering and Computer Science
on August 16, 2005, in partial fulfillment of the
requirements for the degree of
Master of Engineering

Abstract

This thesis describes an automated monitoring station designed to detect flash floods occurring in the Rio Aguan river basin, Honduras. An Atmel microcontroller polls a series of sensors in the river, logging all data for later hydrological analysis and modeling. A high-power APRS radio is used to alert a central monitoring facility of impending floods. Careful component choices and power management allows the system to run for 100 days on a single deep-cycle marine battery or practically indefinitely using a supplemental solar panel.

Thesis Supervisor: James W. Bales
Title: Assistant Director, Edgerton Center

Acknowledgments

Jim, A good advisor makes or breaks a thesis. You've not only been a great advisor, you've been my teacher, my mentor, and my friend. Thank you for all your help and guidance, both on this thesis and in Strobe lab. These past years TAing for you are an experience I'll cherish forever.

To the rest of the Edgerton Center – thank you for a wonderfully eccentric and supportive environment to make a home away from the dorm. I can't think of a better place to have spent the last three years! I'll miss you guys.

Elizabeth, Kathleen, Alex, Victor and the rest of the SAT team, thank you for working tirelessly on this project with me into so many late nights. It's been a lot of fun working with you this year.

JVO and Sharon, thank you for all your love and support. I couldn't ask for better parents.

Contents

Chapter 1

Introduction

Hurricanes and tropical storms frequently devastate communities in the Caribbean. In October 1998, Hurricane Mitch alone was responsible for the death of an estimated 5,000 people, with an additional 8,000 missing and 12,000 injured[1]. At least 25 small villages in the northern part of the country were simply swept away.

The Aguán river basin, located in the department of Colón, was one of the areas worst affected by Hurricane Mitch's passage. The huge quantity of rainfall generated by the hurricane caused the river Aguánto to burst its banks, and within minutes this overflow produced a wave that swept through valley communities.

In general, heavy precipitation, caused by tropical depressions, tropical storms and hurricanes, continues to claim human lives in Honduras every year. There is a great need in this region for an accurate and robust early warning system that will alert communities in the case of such emergencies and provide them with sufficient notice to safely evacuate.

The current national early warning system relies on volunteers to report local rainfall and river scale measurements to a regional emergency committee. This committee compiles community results and issues emergency alerts via radio networks. Unfortunately, these systems are often poorly maintained and understaffed, as the volunteers tend to evacuate with their families at the first sign of excessively severe weather. In such cases, the current system has proved to be inadequate in issuing alerts for down-stream communities.

This thesis presents an automated monitoring and alert system designed to detect flood conditions without relying on volunteers, while staying affordable for developing countries.

Chapter 2

Design Overview

This chapter discusses the requirements for the full monitoring and alert system, then describes the design for the solution. It concludes by specifying the division of labor between this thesis and other students working on the project.

The following tasks were identified as the core requirements of the floodwater alert system:

- Monitor water level, recording all sensor data for later hydrological analysis and modeling
- Communicate sensor data to central monitoring facility located approximately 50 miles away
- Detect flood events early enough to give downstream communities time to evacuate

The following overall design fulfills these requirements:

The river will be monitored by redundant, field replaceable sensors whose low-cost mitigates their potential loss during a large-scale flood. These sensors are designed to detect water-level, but are built around a generic framework to allow detection of other parameters, such as water-flow, temperature, etc.[2]

Autonomous monitoring stations, located above the bank of the river, will periodically poll these sensors using an error-resistant communication link. All sensor data will be logging to temporary storage. If the sensors indicate a flood, the monitoring station will promptly alert downstream communities. Otherwise, the buffered sensor data will be burst-transmitted in a daily data upload. The use of multiple monitoring stations gives additional redundancy and the ability to track flood propagation speeds, though requires that the communication system correctly address multiple systems.

The central monitoring facility is responsible for working with the current national alert system to notify at-risk communities of impending floods. Additionally, this facility is responsible for receiving and storing the sensor data reported from all monitoring sites.

2.1 Division Of Labor

Approximately thirty people have been involved in this project, ranging from students researching possible sensor technologies through teams investigating potential monitoring sites during spring break in Honduras. This thesis directly interacts with four groups of people:

Sensor Design	This group developed the environmental sensors used to detect the state of the river. The hardware was designed by Kathleen Connolly[2] and interfaces with this thesis using the communication protocol described in Section 4.2.
Power System	This group is responsible for providing a reliable 12v power supply that satisfies the power budget presented in Section 3.2.
Environmental	This team created a weather-proof case to protect the monitoring system from the harsh environment.
Monitoring	This group is responsible for the software running in the central monitoring facility, including storing and presenting the data from the remote monitoring systems. The two students responsible for this development were Dan Jacobs and Edgar Terrero[3, 4].

This thesis represents the hardware and software of the autonomous monitoring station, recording data from the in-river sensors and transmitting to the central monitoring facility.

Chapter 3

Hardware Design

This chapter discusses the hardware developed for this thesis, as well as the rational behind its design. Based on the project requirements in the previous section, the following design goals were developed and ranked below in order of importance:

Reliable	The system must reliably operate 24/7 while accurately recording water levels. Redundant sensors avoid the risk of a single sensor clogging. Ideally, communication protocols should include some form of error detection.
Power Efficient	The system must last at least 90 days on a single battery, as it is difficult to replace batteries in remote locations. Alternatively, the system should support recharging a smaller battery using solar power.
Rugged	The system will be subjected environmental extremes typical of a Central American rain forest – high humidity with tropical temperatures. All hardware must be rated to withstand five years of continuous use.
Easy To Install	The system will be installed on a riverside in rural Honduras, so the fabricated hardware needs to be simple to assemble, test, and use.
Low Cost	The system must be constructed at minimal-cost, due to the budget considerations of the communities and risk from theft or flood damage.
Developer Friendly	A diverse range of people will be involved in this project. Compilers and design tools must be compatible with all major OSs while remaining low-cost. Open Source software is used wherever possible.

We begin with an overview of the hardware, followed by a detailed description of individual subsystems. We conclude this section with a computed power budget for the overall system.

3.1 Hardware Overview

The following text and Figure 3-1 present a high-level design overview of the hardware satisfying these design goals. Subsequent sections describe individual components in greater detail and systematically work through major design decisions.

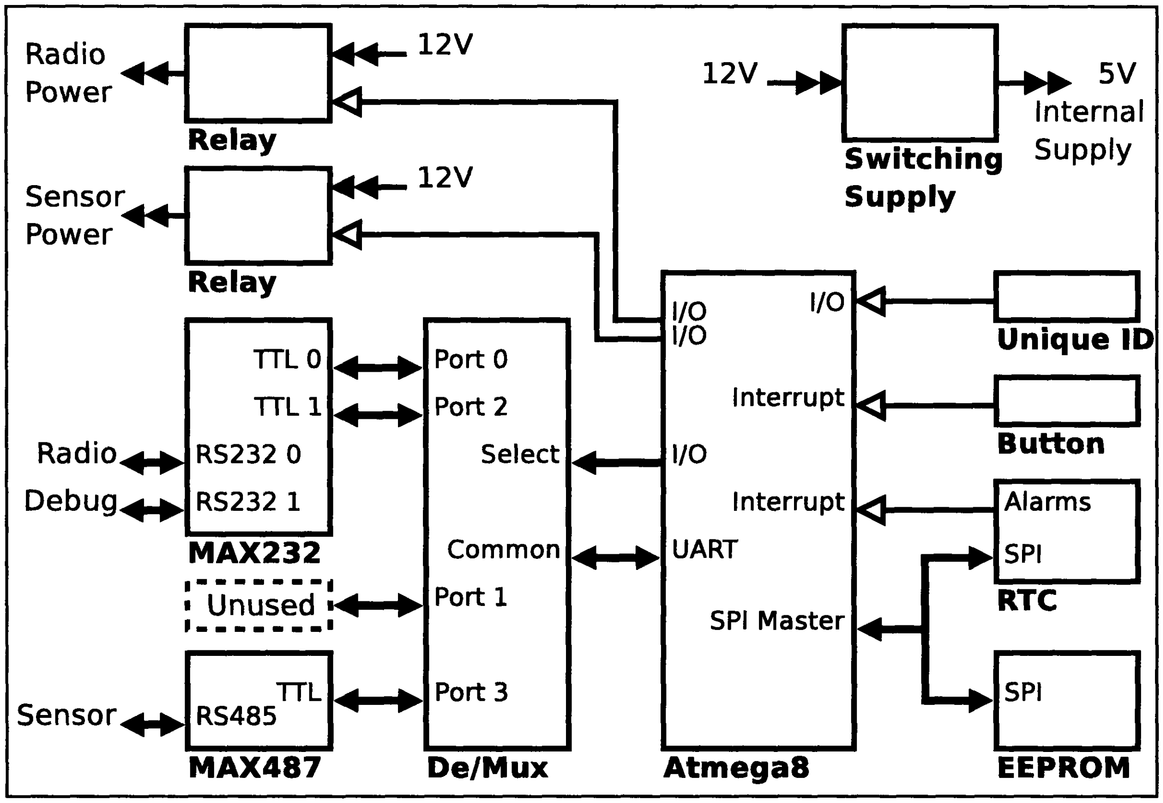

Figure 3-1: Hardware design overview showing all ICs and major communication buses.

The switching power supply converts unregulated 12 volts into a regulated 5 volt supply to power all internal circuitry. The distribution of unregulated 12V power to the associated radio and downstream sensors is provided by two independent relays controlled by the microcontroller.

The Atmega8 schedules these devices' duty cycles by setting two alarms from a real-time clock (RTC) clock chip. The Atmega8 multiplexes its single UART between three destinations, communicating asynchronously with the external radio, a local debugging port, and multiple downstream sensors. Received sensor data is buffered in the serial EEPROM before transmission in daily radio broadcasts to a central receiving station. A unique ID chip is used to generate an individual station identifier, allowing multiple radios to share the same RF communication network.

All components (except the power supply) are configured for low-power suspend modes for the majority of their life cycle. The RTC alarms wake the ATmega8 from sleep mode, which in turn activates other components as needed. The full power budget is presented in Section 3.2.

3.1.1 Power System

The system's regulated 5V power is provided by a step-down switching power supply built around National Semiconductor's LM2674-5.0 IC. The circuit shown in Figure 3-2 accepts an 8 to 40 volt input range to produce a regulated 5V output, while typically running at 90% power efficiency.

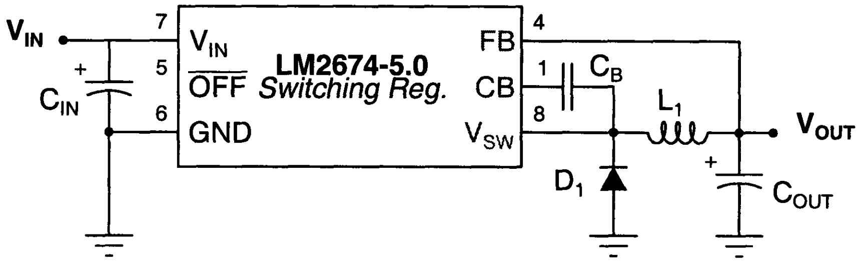

Figure 3-2: Switching regulator and supporting circuitry

This device runs at a 260kHz switching frequency, resulting in smaller filter components than devices operating at lower frequencies. Additionally, by incorporating feedback (FB pin) to realize high power efficiency, this regulator does not require a heatsink other than the copper traces on a PCB. The LM2674 requires a 2.5mA operating current, which could be reduced to 50μA if the active-low shutdown mode ($\overline{OFF}$) were incorporated into the design. A means for accomplishing this has not been established, as it would require signaling a powered-off device to turn on by its unpowered components.

Using the following component values derived directly from the data sheet[5], the LM2674-5.0 guarantees an output within $\pm1.5\%$ of 5V while sourcing up to 500mA. These additional components cost \$2.26, bringing the total cost of the power system to \$5.36 for single-unit quantities.

Additionally, the Power System also controls the attached radio and sensors through two relays. The first relay, model G6RN-1-DC5, provides up to 8A for the attached sensors. The second relay, model G2RL-14-DC5, is rated at 12A to accommodate the 11.5A peak transmit draw of the Kenwood TMD-700A APRS radio. Together, these parts cost \$5.50.

Table 3.1: Components required for power system

Component	Desc	Vendor	Model	Cost
-	High Efficiency 500mA Regulator	National Semi.	LM2674-5.0	3.10
C_{IN}	100uF 50v Radial Electrolytic	ECG	UVR1H101MPD	0.26
C_B	0.01uF 50v Radial Ceramic	Kemet	C315C103K5R5CA	0.20
L_1	47uH Choke RF Hi Current	J W Miller Magnetics	5800-470	1.48
D_1	Schottky Diode. 1.1A 50v	International Rectifier	11DQ05	0.22
C_{OUT}	100uF 10v Radial Electrolytic	ECG	ECE-A1AKS101	0.10

3.1.2 Storage System

We chose serial EEPROMs as the storage media due to their high reliability (100,000 write cycles), long data retention (100 years), low power requirements, and low cost. Atmel's AT25xxx devices were selected because they guarantee compatibility with the hardware SPI interface of the Atmega8, and offer a wide range of memory sizes (1-128Kb) in pin-for-pin compatible packages.

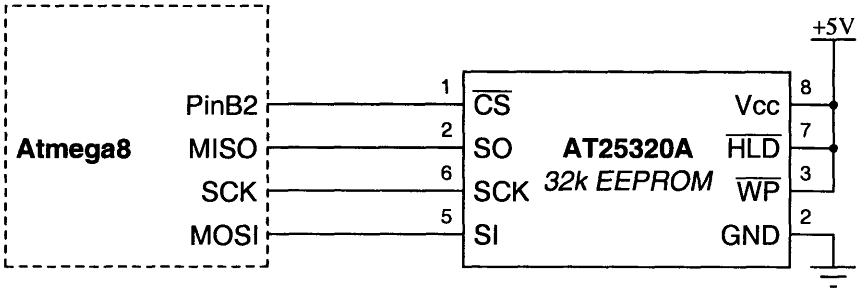

Figure 3-3: SPI Serial EEPROM and associated circuitry

Requiring no extra parts, the EEPROM's wiring is rather straightforward. The three SPI communication pins (Serial Out, Serial In, Serial Clock) connect directly to the corresponding ATmega8 hardware SPI pins.

The active-low Chip Select ($\overline{CS}$) pin enables communication with the AT25320 on a multi-device SPI bus. Except when writing data to the EEPROM, the microcontroller holds this line at V_{CC} to disable the device and allow other devices to communicate on the SPI bus.

Signal Hold ($\overline{HLD}$) allows the SPI communication to be paused in mid-transmission. This feature is not used, and is thus disabled by tying the line to V_{CC}.

The Write Protect pin ($\overline{WP}$) is the hardware aspect of a two-part scheme to prevent data from being written to the device. This pin is connected directly to V_{CC} since the EEPROM never needs to be read-only in this application and it also simplifies the software configuration.

Normal read and write operations typically requires 4mA, though the device only uses 2.0 μA when in shutdown mode ($\overline{CS} = V_{CC}$). If this small current was a concern, the EEPROM could be powered on demand directly from a spare ATmega8 I/O pin capable of sourcing 20mA.

As no additional components are required, the total cost of the storage system is $3.09, in single-unit quantities.

Table 3.2: Components required for storage system

Component	Desc	Vendor	Model	Cost
-	32k SPI Serial EEPROM	Atmel	AT25320A	3.09

3.1.3 Identification System

Rather than relying on a software ID for communication tasks – and risk a potentially misconfigured identifier – this design incorporates a hardware ID, through Maxim's DS2401 Silicon Serial Number. This device provides a 64-bit unique ID stored in a factory programmed ROM, requiring only a single connection for both power and data.

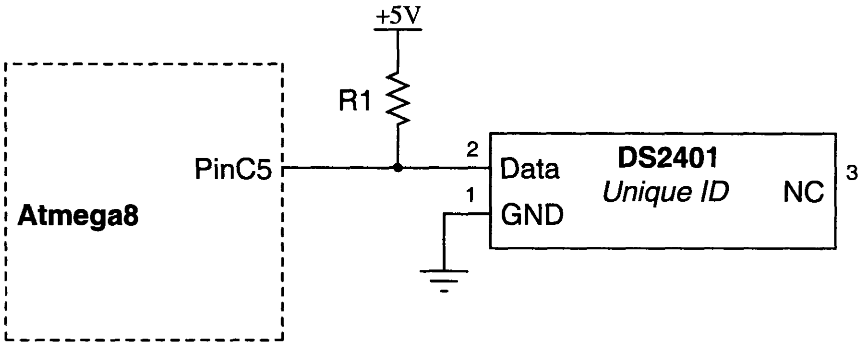

Figure 3-4: DS2401 Silicon Serial Number and associated circuitry

The 64-bit ID is divided into an 8-bit family code (0x01), a 48-bit unique ID, and an 8-bit CRC. This CRC and the "Dallas 1-wire protocol" are described in the application note[6].

As the DS2401 is accessed infrequently and its power requirement is minimal ($5\mu A$ when active and 0 standby current), this device is ignored in the power budget found in Section 3.2.

Ignoring the negligible cost of a pullup resistor, the single-unit total cost of the storage system is $2.28 for the through-hole package used, while the surface mount version of this part costs less than a dollar.

Table 3.3: Components required for identification system

Component	Desc	Vendor	Model	Cost
-	Silicon Serial Number	Dallas/Maxim	DS2401Z	2.28
R1	5k Pullup Resistor	-	-	-

3.1.4 Timekeeping System

Maxim's DS1305 Real-Time Clock provides a full calendar/clock system and two programmable alarms. Given an attached battery, this chip will transparently switch to its backup supply upon power failure.

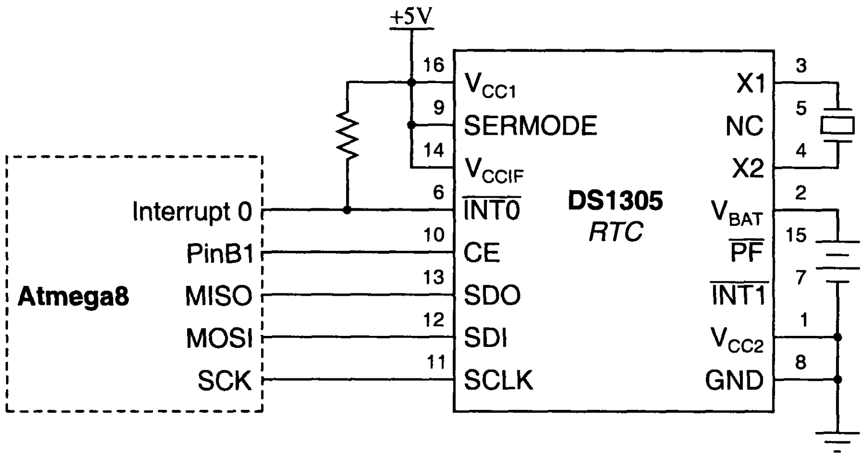

Figure 3-5: DS1305 Real-Time Clock and associated circuitry

The DS1305 uses the standard SPI wiring scheme – Serial Data Out, Serial Data In, and Serial Clock are all connected to the corresponding ATmega8 pins and Chip Enable is grounded except during clock communication.

The device is powered through the V_{CC1} interface, while the V_{CC2} trickle-charge features for rechargeable batteries are disabled. Instead, a 3v primary battery is connected directly to V_{BAT} to provide backup power.

The chip is configured so that both programmable alarms toggle the interrupt 0 ($\overline{INT0}$) output of the DS1305, so only one interrupt pin is required of the microcontroller. When the microcontroller is woken from sleep mode, it queries the DS1305 to determine which alarm generated the interrupt. V_{CCIF} is configured so the device uses 5V data signals, while SERMODE is tied high to configure SPI communication instead of Three-wire. The Power-Fail Alarm pin is not used in this design, as the microcontroller would also lose power during such an event.

The DS1305 uses a standard 32.768kHz crystal for timekeeping, and is specified to be accurate to a minute per month. A one week test found a 40-second drift in the clock, though this could result from a crystal damaged by soldering leads onto a surface-mount part.

This device was designed for low-power systems and operates with 1.3 mA during SPI communication. However, it requires only 81μA for timekeeping while using the 5v V_{CC1} supply and, during power-fail, it requires only 400nA from a battery source on V_{BAT}. Using a low-cost, 35mAh lithium coin cell provides almost 10 years of reliable timekeeping[1].

The DS1305 requires roughly $2 in supporting hardware, bringing the total cost to $4.31 in single-unit quantities.

Table 3.4: Components required for timekeeping system

Component	Desc	Vendor	Model	Cost
-	Serial Alarm Real-Time Clock	Dallas/Maxim	DS1305	2.28
R1	5k Pullup Resistor	-	-	-
Y1	32.76kHz Crystal, 6pF	ECS	ECS-.327-6-17	1.08
B1	12mm 3v Lithium Coin cell	Panasonic	CR1220	0.95

3.1.5 Communication System

The system design requires that the microcontroller communicate with three independent devices: an external radio, a local debugging port, and multiple environmental sensors. Rather than upgrading to an expensive microcontroller with three UARTs, the following design multiplexes the single Atmega8 UART over three different communication channels.

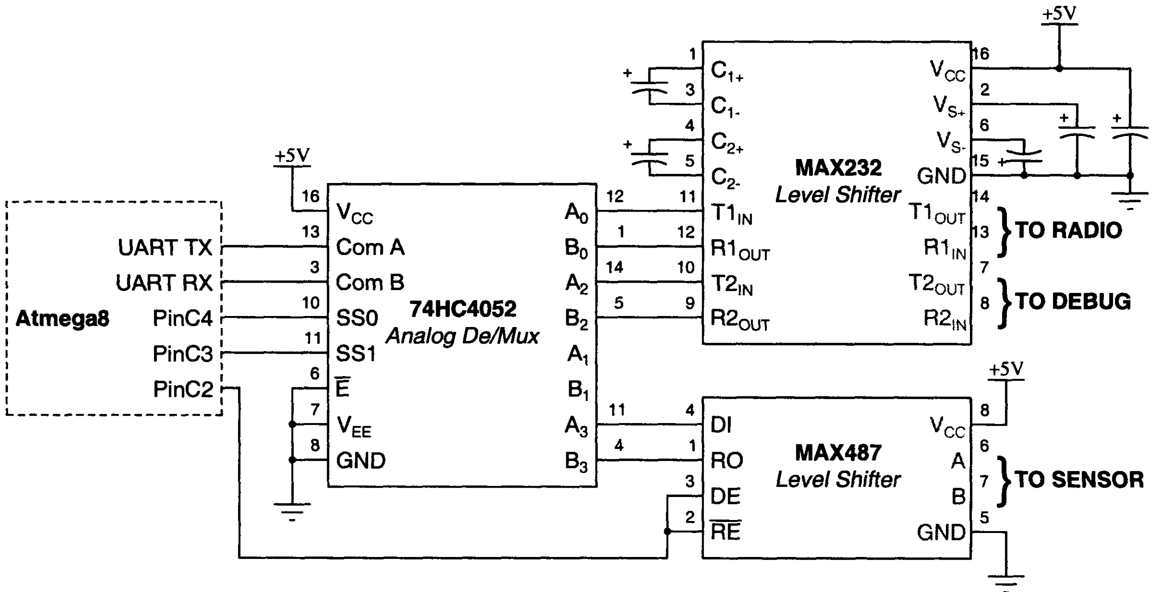

Figure 3-6: Analog Multiplexer switches Atmega UART between RS232 & RS485 converters

[1] In practice, a shorter lifetime is expected due to the battery's self-discharge.

The 74HC4052 analog multiplexer/demultiplexer utilizes low-power CMOS circuity to electronically switch a common pair of I/O pins between four pairs of bi-directional data ports. This device is designed to handle a wide range of analog voltages, thus V_{EE} is grounded to set the lower threshold of valid voltage range. The active low enable pin can disable the device, though this provides little additional power savings so the pin grounded as well. Two extra I/O pins are required for port address selection.

Port zero and two of the 74HC4052 use a MAX232 level-shifter to convert TTL voltages to those compatible with RS-232 devices. Crosstalk between the two pairs of driver/receivers is not an issue, as only one of the radio and debug channels is in use at any time. Four 1uF capacitor are required to created the ± 11 volt signal needed to implement the RS-232 protocol. A fifth capacitor buffers the MAX232's power supply during periods of peak usage. Note that the VS_{-} cap holds a "negative" voltage and thus has the positive pin connect to ground.

Port three of the 74HC4052 connects to the MAX487 level shifter, which converts TTL voltages into the RS-485 differential outputs A and B. This half-duplex driver can communicate with devices as far as a kilometer away when using shielded, twisted pair cables. The above design assumes that RS-485 termination in incorporated into the sensor cabling[2]. The Driver Enable and active-low Receiver Enable pins are tied together to simplify device interface. However, the MAX487 shutdown mode is only activated when DE=GND and RE=V_{CC}, requiring two separate control pins for additional power savings.

Each level converter requires about 10mA during actual serial communication. The MAX487 features at $10\mu A$ sleep mode and could be "shutdown" by powering it directly from a general I/O line from the Atmega8.

Together, these components cost approximately $5 in single unit quantities. Using the 74HC4052 is a significant cost savings over upgrading the Atmega8 to a higher-end microcontroller with three hardware UARTs. Additionally, the TI RS-232 converter cost less than one-eighth the price of the equivalent Maxim part (MAX232).

Table 3.5: Components required for Communication System

Component	Desc	Vendor	Model	Cost
-	Analog MUX/DEMUX	TI	74HC4052	0.53
-	DUAL RS232 DRVR/RCVR	TI	MAX232	0.90
-	RS485 Level Shifter	Maxim	MAX487	2.81
C[1-5]	1uF 50v Radial	Panasonic	ECE-A1HKK010	0.14

3.2 Power Budget

The systems described previously are optimized to be as power efficient as possible. The following table illustrates the average power required for each task the system accomplishes, while the subsequent calculations compute average power and system lifetime from a typical deep-cycle marine battery.

Table 3.6: Estimated daily power consumption, by task

Task	Power (mA)	Duration (s)	Frequency	Total (s)	Charge (mA*hour)
Sleep	3.10	-	-	82500	71.197
Sensor	575.00	20	6/hour	2880	460.00
Radio Rx	1100.00	30	1/hour	720	220.00
Radio Tx	11600.00	120	1/day	120	386.67
Total				**86400**	**1137.86**

Let us examine the tasks of Table 3.6 in more detail:

Sleep	With all devices in sleep mode, the only significant power draw is the 3mA quiescent current drawn from the 5v power supply.
Sensor	The 500mA budgeted for powering the downstream sensors composes the majority of this task, while the sensor relay and microcontroller consume another 60mA. An estimate of 20 seconds are required for the sensor to return a stable reading from a cold power-on. The design call for sampling every 10 minutes, or six times an hour.
Radio Receive	The radio is powered once every hour to check if the base station is attempting communication. The radio requires 1A in receive mode, and a further 100mA are consumed between the radio relay, microcontroller, and RS-232 converter. The radio listens for broadcasts for thirty seconds.
Radio Transmit	The radio transmits the accumulated sensor data once a day, requiring an estimated two minutes at 9600 baud. The radio consumes 11.5A for high-power transmission, and a further 100mA are needed by the radio relay, microcontroller, and RS-232 converter.

An estimated charge of 1138 mA-hours per day computes to an average power consumption of approximately 47mA. Powering the system from a single 120 amp-hour deep-cycle marine battery would yield an estimated lifetime of 105 days. Assuming some form of supplemental power supply, such as solar energy, this system could last indefinitely, and the battery could be scaled back to only supply a few days of "reserve" energy to ride out a stretch of cloudy weather.

Chapter 4

Software Design

This chapter presents a high-level overview of the software running on the Atmega8 microcontroller, then discusses the lower-level modules and concludes with specifications for the sensor and radio communication protocols.

On initial boot, the Atmega8 configures the RTC, verifies the attached radio, and detects environmental sensors using the algorithm described in Section 4.2.2. Having completed these tasks, the microcontroller powers down the peripherals and put itself into a low-power sleep mode. The Atmega8 wakes from sleep either by a preconfigured alarm from the RTC, or by a local button press to enter debugging mode.

Since the two RTC alarms map to the same physical interrupt, the Atmega8 must query the RTC to determine if it is time to poll the attached sensors (alarm 0) or interact with the radio (alarm 1). Once this mode is know, the Atmega8 wakes the required peripherals and completes the task.

The default duty cycle has downstream sensors queried every 10 minutes, with the resulting data stored in the local EEPROM until the next transmission cycle. If the sensors detect a flood condition, the radio is immediately powered to transmit an alert to the base station. Otherwise, the radio defaults to listening for communication requests from the central base station for 30 seconds an hour, and transmitting recorded sensor information once a day. Sections 4.2 and 4.3 describe the protocols used to communicate with the sensors and base station.

By pressing a switch, a local user can wake the Atmega8 from sleep mode and place it into a debugging mode. Using a standard communication packages, such as hyperterm or minicom, the user can examine the program state and initiate self-tests of the peripherals, sensors, and radio. The Atmega8 intelligently multiplexes communication through its single UART, pausing the stream to the user to communicate with the radio or sensors, then resuming the debug stream to transmit results to the user.

4.1 Module Overview

The main program is supported by the following six modules.

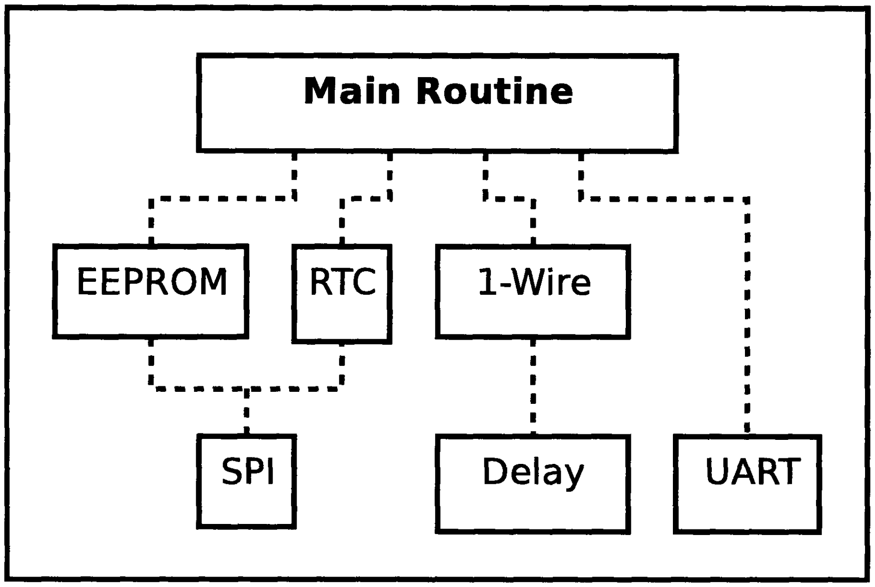

Figure 4-1: Software Module Hierarchy

Starting from the lowest level of Figure 4-1:

SPI	This module implements the low-level code needed to communicate using the Atmega8's Serial Peripheral Interface (SPI) bus. This code started from examples in the Atmel documentation and was extended to support the different SPI timing modes used by the attached SPI devices.
Delay	This library is a simple suite of delay routines to give precise millisecond and microsecond delays. The library was found on AvrFreaks[7] and was written by Martin Thomas and Peter Dannegger.
UART	This module implements the low-level interface to the Atmega8's hardware Universal Asynchronous Receiver Transmitter (UART). As only short messages are sent, this code implements the minimalistic examples in the Atmega8 documentation and does not include additional buffering or collision detection.
EEPROM	This module provides a full interface for Atmel's AT25x series of EEPROM devices. This code is extracted from Atmel documentation and provides additional functionality to detect and verify an attached EEPROM.

RTC	This module implements a straightforward interface into Maxim's DS1305 Real-Time Clock, mapping device programming into simple read and write aliases into the clock's memory registers. This memory interface is described in Maxim's DS1305 datasheet[8].
1-Wire	This module returns the UID stored in an attached DS2401 Silicon Serial Number and verifies the embedded CRC. This module was found on AvrFreaks[7], author unknown.

4.2 Sensor Protocol

The following protocol mimics the easy-to-understand spirit of the Serial ASCII Instrumentation Loop (SAIL) communication standard[9]. Both protocols ease debugging by using a syntax that is entirely human-readable, being composed of alphanumeric characters separated by spaces and terminated with a newline. In-the-field diagnostics can thus be accomplished by connecting a RS-485 line-level converter to a laptop serial port and examining the communication stream in any common communication suite (hyperterm, minicom, etc)

The protocol is driven by a single master device interacting with one or more slave sensors. As the sensors report only a few bytes of data, the protocol uses a relatively slow data rate (9600 or 19200 baud) to minimize transmission errors across long communication cables. Additionally, the protocol supports using a CRC to verify data transmissions.

The form of a command is **"#[BAU]Address Command Argument\n"**.

All commands begin with the pound sign ("#") followed by the upper-case character "**B**"," **A**", or "**U**" and terminate with a carriage return ("**\n**"). The choice of character sets the method of addressing, either by Broadcast, Alias, or Unique ID. All sensors respond to broadcast commands, while only a specific device will react to its assigned Alias or hardware Unique ID (UID). An Alias can be one of sixteen values (represented as a single ASCII hexadecimal character), while an UID represents the 64-bit DS2401 ID (transmitted as 16 hex characters). Examples of **Address** components are shown in Section 4.2.1.

There are only eight valid commands, which are either transmitted as full English words (to increase human understanding) or are abbreviated as a their first letter to reduce transmission overhead. The receiving sensor automatically truncates all but the first letter of a command. Commands either have no argument (eg, **STATUS**) or a single-character argument (eg, **ALIAS A**). All commands generate a response from the addressed device, with **"OK\n"** composing the simplest possible reply. These commands and responses are listed in Table 4.1:

Table 4.1: Valid commands in sensor protocol, all data in hex.

Cmd	English	Arg	Description	Response
L	LED	0,1	Turns LED on if =1, else turns off	OK
E	ECHO	00-FF	Echo received byte	Input Byte
U	UID	-	Delay, clears alias, returns DS2401 UID	UID (16 chars)
A	ALIAS	0-F	Alias is programmed into EEPROM	OK
I	IGNORE	-	Ignores all commands save "W"	OK
W	WAKEUP	-	Respond to further commands	OK
D	DATA	-	Returns sensor data	Data + CRC
S	STATUS	-	Returns device status	OK or ERR

4.2.1 Sensor Protocol Examples

Turning on the debugging LEDs of all attached sensors could be accomplished by sending the broadcast command **"#B LED 1\n"**, or the equivalent abbreviated form "**#B L 1\n**". The devices' response of **"OK\n"** might be garbled due to slightly different propagation delays as multiple devices simultaneously sent the same text to the master.

The master could request data from the device with alias "0" by sending **"#A0 DATA\n"**. The same sensor could be addressed by the hexadecimal representation of its hard-coded UID, such as **"#U0123456789ABCDEF D\n"**. In either case, the sensor device would respond with a reading from its environment.

4.2.2 Sensor Discovery

An algorithm based on Dallas-Semi's 1-wire protocol[6] is used to discover all sensors attached to the RS-485 communication bus.

The Master first broadcasts the **UID** command (**"#B U\n"**), causing all slaves to clear their current alias, pause a pseudo-random number of milliseconds, then respond with their UID. The delay minimizes the chance of collisions, though all UIDs are verified through their embedded CRC.

The Master device then picks the first recognizable UID, directly addresses it, and assigns it the first alias, causing the sensor to store its new alias into on-board EEPROM. This sensor is then given the **IGNORE** command, and ignores all future communication until specifically invited back into the conversation with the **WAKEUP** command or reset by cycling power.

The Master now repeats this process with the next valid UID until all devices are identified and aliased. At this point, all devices are woken up and normal usage occurs.

4.3 Radio Protocol

This project currently uses a simple protocol to allow a single monitoring site to communicate with a single base station. Accordingly, it only needs to support four tasks, namely reconfiguring the default sensor and radio duty cycles, updating the clock to account for drift, transmitting stored data to the base station, and sending flood alerts. At the same time, the protocol needs to support future expansion, allowing for multiple monitoring sites to either act as packet relays to the most extreme sensor or to communicate through a digital repeater with extended range.

The APRS protocol[10] supports all these tasks, as described in the following sections. Our chosen radio handles the actual APRS packet encoding and transmission, so the Atmega8 is only responsible for initializing the radio into the proper mode and generating the data payload.

4.3.1 APRS Standard

The Automatic Position Reporting System (APRS) uses the AX.25 Amateur Packet-Radio Link-level protocol[11, 12] for low-level communication. By utilizing AX.25 Unnumbered Information (UI) frames, APRS packets are transmitted without expecting any response or guaranteeing reception.

At a higher level, APRS supports a messaging protocol that allows users to send short messages and expects to receive acknowledgment. Similarly, the APRS protocol contains provisions for streaming telemetry data and querying station status and capabilities. This project uses these features to report project health and transmit buffered sensor data to the base station.

4.3.2 Status Reporting

The APRS protocol uses a Status Report to announce a station's current mission or other single-line status to all receivers within range. Table 4.2 summarizes Chapter 16 "Status Reports" of the APRS Protocol Reference[10]:

Table 4.2: Status Report Format

Field	Info	Time	Status Text
Format	>	DDHHMMz	Text
Bytes	1	7	0-55
Example	>160900zRadio Online		

The optional timestamp is in a DayHourMinute format, and must be sent in zulu (UTC/GMT) form. The text may contain any printable ASCII character except "|" and "~".

The monitoring system transmits only four different status messages: “Radio Online” when the radio powers on, “Radio Offline” immediately before powering down,“Radio OK” if the central site queries the online radio, and “Flood Detected” if the sensors indicate that is the case. The example in Table 4.2 shows a hypothetical status update sent when the radio powered on at 09:00 on the 16th of the month.

4.3.3 Messaging

APRS supports messages composed of free format text strings, intended to convey human-readable information. These messages are intended for a single recipient and can optionally require acknowledgement upon reception. This system utilizes APRS messaging to receive and respond to commands from the central base station. The full specification is described in Chapter 14 “Messages Bulletins, and Announcements” of the APRS Protocol Reference[10] and summarized in Table 4.3:

Table 4.3: Message Format

Field	Info	Addressee		Message Text	Message ID	
Format	:	Station Address	:	Text	{	xxxxx
Bytes	1	9	1	0-67	1	5
Example	:SITE00001:Set Clock 081605 090000{001 :BASECAMP␣:ack001					

An APRS message is a text string with a specified addressee; a fixed 9-character field (padded with spaces if necessary) following the “:” data type identifier. The addressee field is terminated by another “:”, then followed by the text of the message. This message text can include any printable ASCII character, save for “|,~,{”.

The sender requests message acknowledgment by including the optional Message Identifier, which is composed of the “{” character, followed by up to five alphanumeric characters. The sending station will repeatedly transmit such a message until it receives an acknowledgment or times out.

The receiving station uses a similar format for message acknowledgement, simply replacing the message text with “ack” followed by the five-character message ID.

The central receiving site commands a monitoring station through five special APRS messages, shown in Table 4.4.

The example in Table 4.3 shows a transmission from BASECAMP commanding SITE00001 to update its clock to transmitted timestamp and requesting acknowledgement. SITE00001 responds that it received the message, and thus performed the command.

Table 4.4: Commands sent to monitoring station via APRS Messages

Command	Arguments	Action
Set Freq	HHMM HHMM HHMM	Updates Sensor Poll, Radio Rx, Radio Tx daily interval
Set Clock	MMDDYY HHMMSS	Sets clock to included timestamp (GMT)
Send Status	-	Sends status message (Section 4.3.2)
Send Time	-	Sends current timestamp
Send Data	-	Sends last day's telemetry data

4.3.4 Telemetry Transmission

APRS supports a standard telemetry data format used by a number of transceivers, and included in amateur rocketry and weather balloon projects[13]. The full specification is described in Chapter 13 "Telemetry Data" of the APRS Protocol Reference[10] and summarized in Table 4.5:

Table 4.5: Telemetry Report Format

Field	Info	Sequence	Value 1	Value 2	Value 3	Value 4	Value 5	Digital
Format	T	#xxx,	aaa,	aaa,	aaa,	aaa,	aaa,	bbbbbbbb
Bytes	1	5	4	4	4	4	4	8
Example	T#001,211,211,211,000,000,00000000 T#002,211,212,212,000,000,00000000							

The Information field uses "T" to identify this message as telemetry data.

The Report Sequence is a 3-character value, typically a 3-digit sequence. The receiving station can verify packet count and request that missing data be resent.

There are five 8-bit unsigned analog data values (expressed as 3-digit decimal numbers in the range 000-255), followed by a single 8-bit binary number.

Upon receiving the "Send Data" command from the base station, the system sends a status message announcing the start time of the collected data and the number of frames to be transmitted. Assuming the default rates for reading the sensors (6/hour), 24 hours of data is sent as 144 frames. Note that this limits the system to six attached sensors, or requires the Telemetry Format to be modified. If fewer sensors are attached, multiple data samples could be compressed into a single data frame to decrease transmission time. The example above shows the first two frames of a data sequence with three attached sensors reporting similar values.

Chapter 5

Conclusions

This thesis presents an automated system to warn at-risk communities of impending floods. Inspired by work in Honduras, this system can be deployed in other locales as a flash flood monitor. The design is extensible to autonomously monitor a wide range of environmental parameters, occurring in remote locations, for long periods of time.

The system exceeds its design requirements, lasting for over 100 days on a single deep-cycle battery, costing less than $40 in components, and utilizing freely accessible compilers and development tools. Further, the system is believed to be rugged and easy to install, although the proof will be in the actual field deployments. However, as with any system of this complexity, there is additional work to be completed.

5.1 Future Work

This project will be extended for a number of years, gradually scaling to a full system composed of multiple monitoring sites broadcasting through a high-power repeater to a staffed monitoring center. As additional features are incorporated into the monitoring system, long-term reliability and integration testing must occur before shipping the system to Honduras.
Particular goals for the near future are:

The power-hungry Kenwood TD-700A radios should be replaced with custom APRS driver hardware and more efficient radios.

Use of the the APRS protocol should be expanded to handle addressing multiple locations and recover from packet collisions as multiple monitoring sites are brought online.

Finally, extending the number of sites covered will require some form of packet repeater to ensure the radios can still communicate with the distant central monitoring site. This will require either the purchase of a repeater or modification of monitoring station hardware to perform the same functionality.

Bibliography

[1] Hurricane Mitch Fact Sheet. http://www.metoffice.com/sec2/sec2cyclone/tcbulletins/mitch.html.

[2] Kathleen Connolly. Early Warning Flood Detection System: Sensor System Design. Technical report, Massachusetts Institute of Technology, May 2005.

[3] Daniel Jacobs. Sistema de Alerta Temprana: An Early Flood Warning System for the Rio Aguan Basin in Honduras (Software Team). Technical report, Massachusetts Institute of Technology, May 2005.

[4] Edgar J. Terrero. Early Warning System Software. Technical report, Massachusetts Institute of Technology, May 2005.

[5] National Semiconductor. *LM2674: Simple Switcher Power Conveter.* http://www.national.com, February 2005.

[6] Dallas Semiconductor/Maxim. *DS2401: Silicon Serial Number.* http://www.maxim-ic.com, February 2002.

[7] AvrFreaks. http://www.avrfreaks.net.

[8] Dallas Semiconductor/Maxim. *DS1305: Serial Alarm Real-Time Clock.* http://www.maxim-ic.com, February 2002.

[9] IEEE Standard Serial ASCII Instrumentation Loop (SAIL) Shipboard Data Communication. *ANSI/IEEE Std*, May 1985.

[10] The APRS Working Group. *APRS Protocol Reference Version 1.0.1.* http://www.tapr.org, August 2000.

[11] Tucson Amateur Packet Radio Corporationo. *AX.25 Link Access Protocol for Amateur Packet Radio Version 2.2.* http://www.tapr.org, June 1998.

[12] R.R. Parry. AX.25 [data link layer protocol for packet radio networks]. *IEEE Potentials*, 16(3):14–16, August 1997.

[13] Near Space Return Vehicle. http://www.kd7lmo.net/index.html.