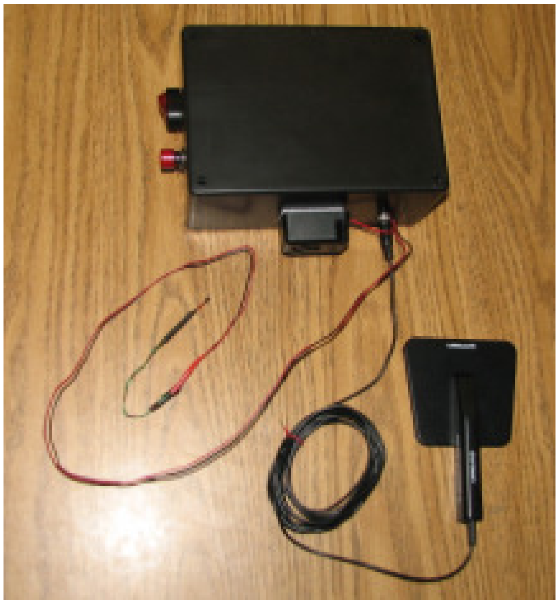
CPR For Dummies

Bathtub Drown/Burn Prevention



James C. Young
Justin O. Young

December 20, 2006
Computer Engineering
Senior Project Documentation

Table of Contents

Introduction

Your bathtub is a serious health hazard to your children. In the United States alone, over 200 children drown in bathtubs each year and close to 112,000 people are treated for scald burns each year. According to Safe Kids Coalition, about 37,000 of these people are 14 or under, and about 18,000 are 5 or under. Children's skin burns faster than adult's skin because their skin is thinner. Everyday, 300 young children are taken to emergency rooms for scald burns caused by household water that was too hot. Annually, close to 3,000 of these children require hospitalization. These two health hazards are the second leading cause of death for young children ages 0 to 5. Whether a child drowns or is severely burned, young children fall victim to other people's carelessness, and in most cases the results are fatal.

We have first hand experience as to why this project is so important. Recently one of our cousins drowned in a bathtub because it wasn't drained properly. One of their older siblings took a bath that morning and forgot to drain the bathtub.

The CPR For Dummies (a.k.a. CPRFD) project minimizes the time window in which an accidental drowning or burning can occur. CPRFD will automatically drain a bathtub if it detects the water temperature is too hot (temp > 40 C) or if it hasn't detected sound for a given amount of time (up to 4 min 15 sec).

Functional Description

Hardware Installation

There are four hardware components which will be installed in the bathroom.

- *First*, a 7x5x3 black plastic water proof case with the following already attached to it: a rocker switch, a pushbutton switch, and an alarm. Mount this to one of the outside walls of the bathtub.
- *Second*, a thermistor which hangs from the bottom of the project enclosure. Place this anywhere inside the bathtub where it will submerge under water.
- *Third*, a microphone which is plugged into the bottom of the project enclosure. Place this close to the bathtub where it won't get wet (preferably on top of the project enclosure).
- *Fourth*, a draining unit which hangs from the bottom of the project enclosure. Mount this just below the latch that drains the bathtub. Make sure the side with the rope is facing up towards the latch. Tie the rope around the latch. Make sure the remainder length of the rope only allows the metal cylinder to be 0.12 to 0.50 inches out of the solenoid.

User Interface

The user interface is through the rocker switch and pushbutton switch located on the left side of the project enclosure. The rocker switch is used to power the system and the pushbutton switch is used to reset the system (start the timer over).

Primary Functions

There are two primary functions which are performed continuously while the system is on.

- *First*, it will determine whether or not the water temperature is hot enough to burn a child's skin (temp > 40 C).

  If it determines the water temperature is too hot then it will sound an alarm and wait ten seconds for a response before it drains the bathtub. If within this ten second time frame, either the reset button is pressed or the water temperature cools down (temp < 40 C), then the system resets itself and turns off the alarm. If neither of the above two mentioned things occur within this ten second time frame then the system will drain the bathtub and continue to sound the alarm until the reset button is pressed or the system is turned off.

- *Second*, it will determine whether or not sound was detected for a given amount of time (up to 4 min 15 sec).

  If it determines there hasn't been any sound detected for the given amount of time then it will sound an alarm and wait ten seconds for a response before it drains the bathtub. If within this ten second time frame, the reset button is pressed, then the system resets itself and turns off the alarm. If the reset button is not pressed within this ten second time frame, then the system will drain the bathtub and continue to sound the alarm until the reset button is pressed or the system is turned off.

Verification

To verify that CPRFD works, install it in any bathroom that has a bathtub (installation steps located above), fill the bathtub with water (temp < 40 C), and turn the system on. Now try the following tests:

- *Sound Test*: Don't make any noise and let the system sit for about 5 seconds. Then tap or talk into the microphone for a given amount of time (up to 4 min 15 sec), this prevents the system from sounding the alarm. The system should reset itself (including the timer) every time sound is detected.
- *No Sound Test*: Don't make any noise and let the system sit for a while. Within a few minutes (up to 4 min 15 sec) the system should sound the alarm. After the alarm sounds, push the reset button.

- *No Sound & Drain Test*: Don't make any noise and let the system sit for a while. Within a few minutes (up to 4 min 15 sec) the system should sound the alarm. After the alarm sounds, let the system sit for ten more seconds. The system should then drain the bathtub and continue to sound the alarm. Push the reset button.
- *Water Temperature Test*: Pull the thermistor out of the bathtub so it's completely out of the water. Empty the bathtub and refill it with only hot water. Place the thermistor back into the bathtub so it's submerged under water. Within one second after the thermistor is fully submerged under water, the system should sound the alarm. After the alarm sounds, wait about 3 seconds and then pull the thermistor out of the bathtub so it's completely out of the water. Within one to two seconds after the thermistor is completely out of the water, the system should reset itself (including the timer) and turn off the alarm.
- *Water Temperature & Drain Test*: Pull the thermistor out of the bathtub so it's completely out of the water. Empty the bathtub and refill it with only hot water. Place the thermistor back into the bathtub so it's submerged under water. Within one second after the thermistor is fully submerged under water, the system should sound the alarm. After the alarm sounds, let the system sit for ten more seconds. The system should then drain the bathtub and continue to sound the alarm. Push the reset button.

Design Description

Overview

CPRFD contains both hardware and software components. The hardware components illustrated in Figure 1 below, consist of a power supply, thermistor, microphone, alarm, draining unit, and a microcontroller. The only software component in this project is a program that controls the microcontroller. The interface between the thermistor, microphone, alarm, and draining unit are managed by the microcontroller.

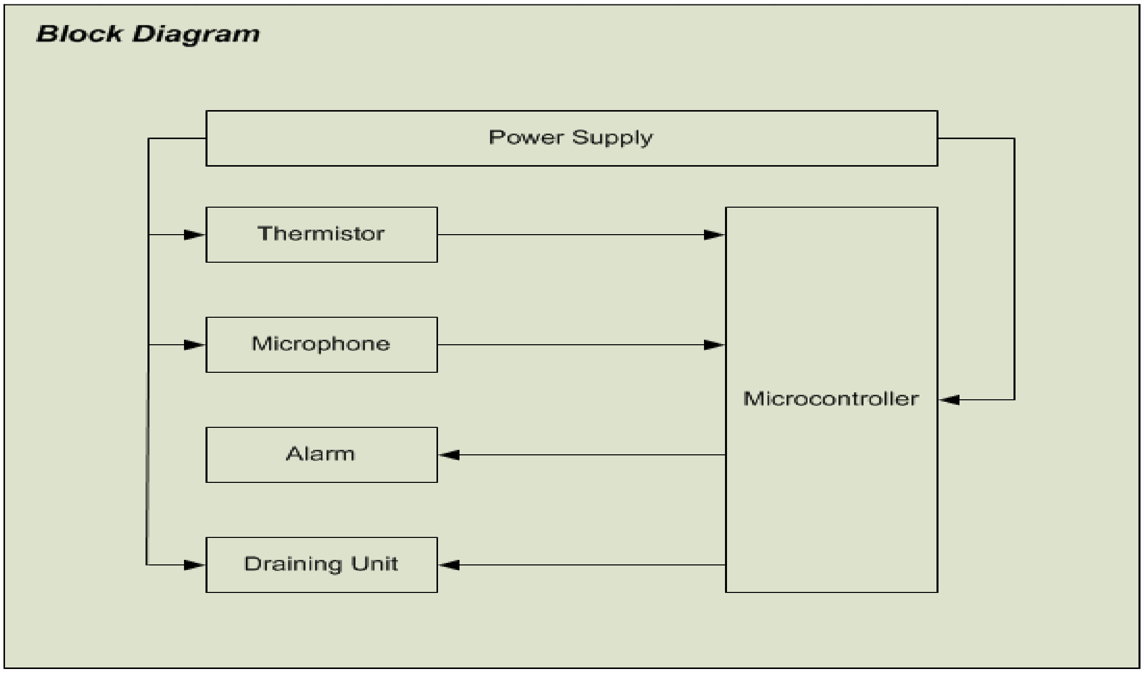


Figure 1 – Overview.

The majority of the hardware is enclosed in a 7x5x3" water proof case which is illustrated in Figures 2 and 3 below.

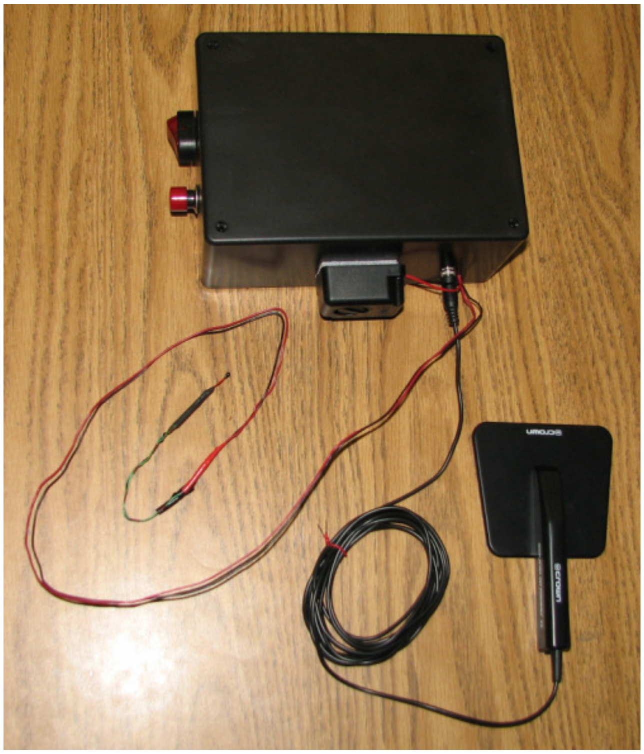


Figure 2 – CPRFD Unit.

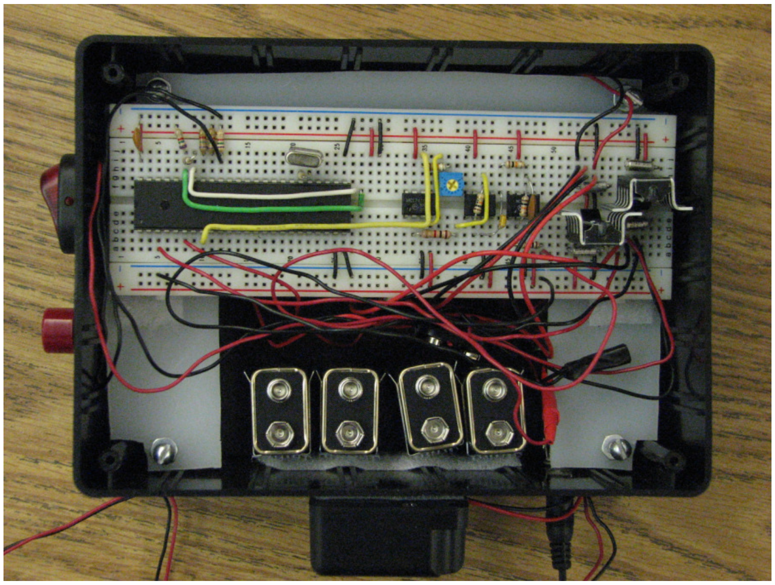


Figure 3 – CPRFD Unit.

Power Supply

The system is being powered with 4 9V batteries and some voltage regulator circuitry. We need +12V and -12V for our operational amplifiers in our thermistor circuitry. We need +5V and -5V for our operational amplifiers in our microphone circuitry. We need +5V for our microcontroller. Figures 4 through 7 below, illustrates how we implemented our power supply to attain the voltage needed.

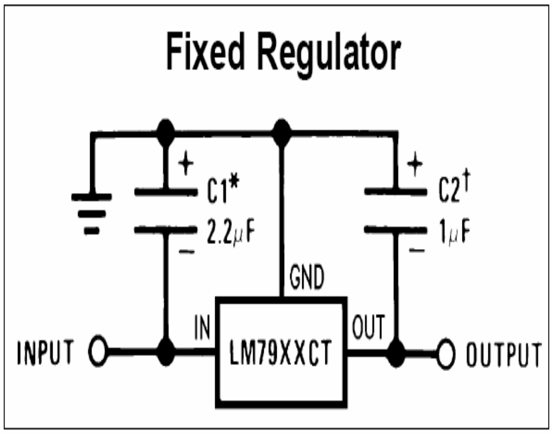


Figure 4 – Negative Voltage Source.

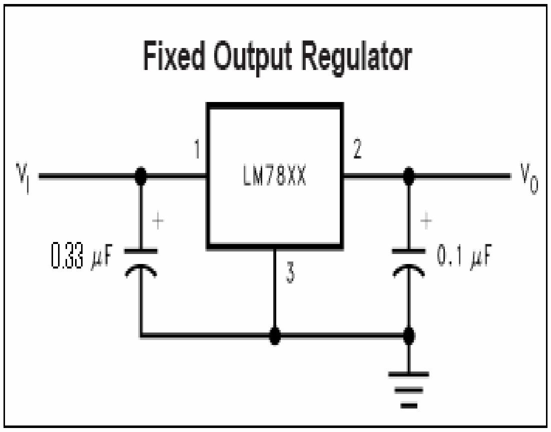


Figure 5 – Positive Voltage Source.

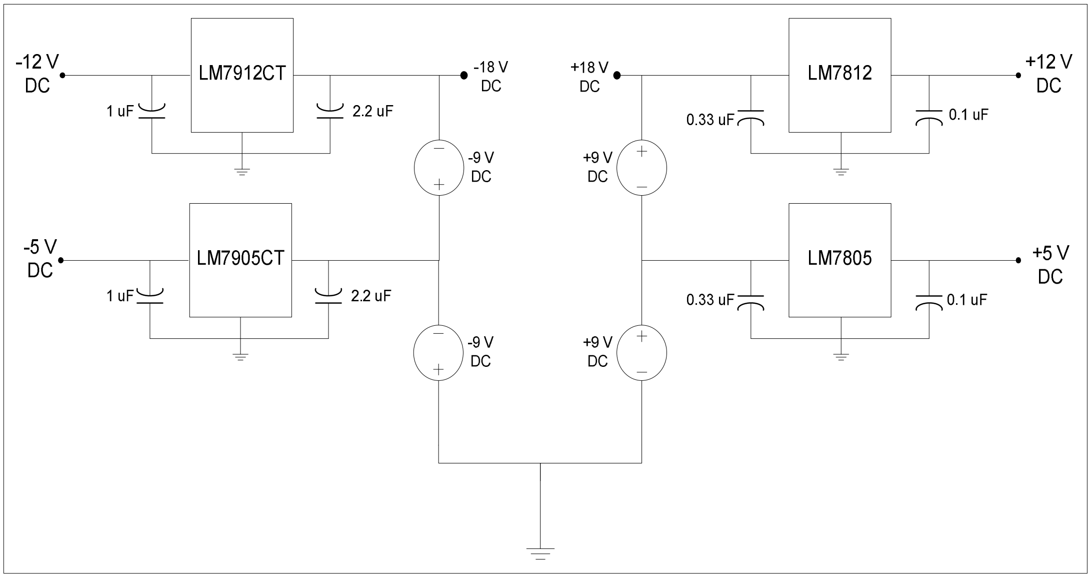


Figure 6 – Power Supply Schematic.

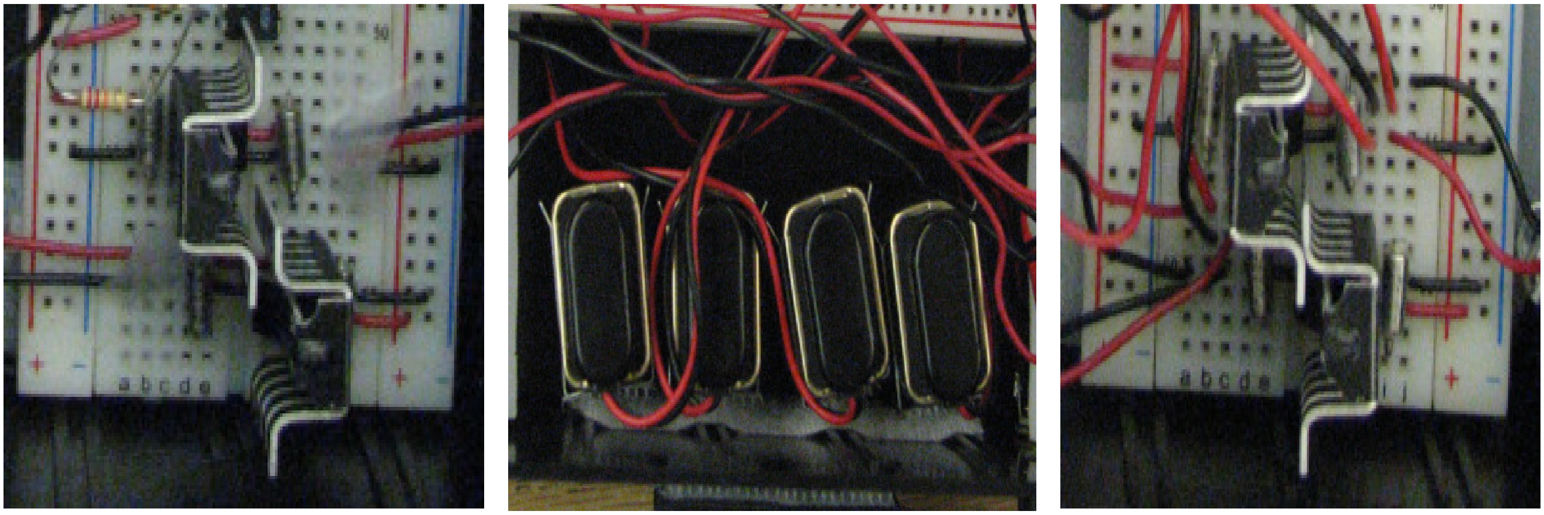


Figure 7 – Power Supply Circuit.

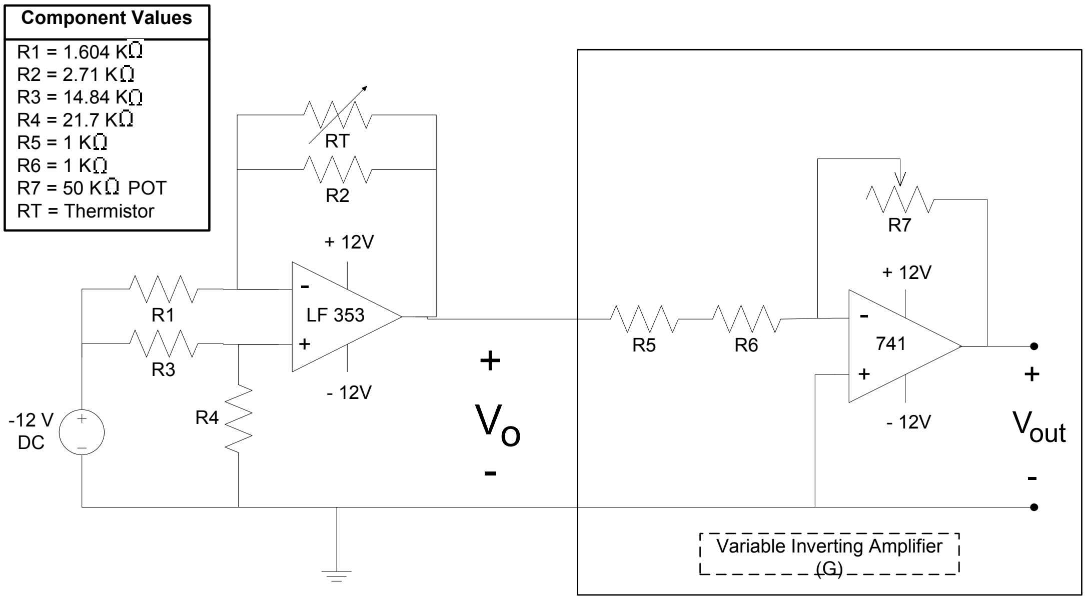
Thermistor

The purpose of the thermistor is to check to see if the water temperature is too hot (temp > 40 C). It will measure the temperature of the water in the bathtub and then send a signal to the microcontroller based off that temperature reading. The microcontroller will analyze the signal and determine the appropriate action to take.

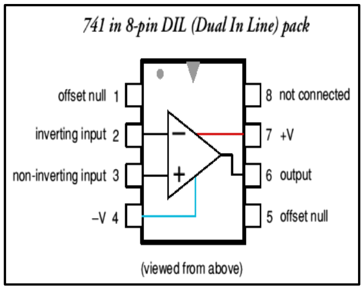
We used the same thermistor that was used in ECE 1000 Lab 1. Its resistance is directly related to the temperature of its surroundings. Figure 8 shows an adequate model of its resistance as a function of temperature. Our microcontroller needs a voltage signal (not a resistance signal) so we needed some way to relate the thermistors resistance as a measure of voltage. The circuit we implemented to accomplish this is also illustrated in Figure 8. Once we did the resistance to voltage conversion, we had to amplifier the signal because the voltage wasn't large enough for the microcontroller to see it. We only wanted our microcontroller to see temperatures greater than 40 C, any temperature lower than or equal to 40 C shouldn't be seen by the microcontroller. So we needed to know exactly how much the signal needed to be amplified to reach the desired temperature range. The two conversion tables located in Figure 8 helped us accomplish this.

Note: After we calculated what resistor values our circuit would need, we found that some of those values didn't exist and we had to get resistors that were close to the exact value calculated. Also the thermistor wasn't a 100% accurate with its temperature measurement and sometimes it would be 1 to 2 C's off. None of this really affected our results because the conversion tables allowed us more pin-point precision when trying to reach our desired temperature range.

Thermistor Circuitry


Component Values

R1 = 1.604 KΩ
R2 = 2.71 KΩ
R3 = 14.84 KΩ
R4 = 21.7 KΩ
R5 = 1 KΩ
R6 = 1 KΩ
R7 = 50 KΩ POT
RT = Thermistor



Thermistor Model

$$R_T = R_0 * e^{\beta(1/T - 1/T_0)}$$

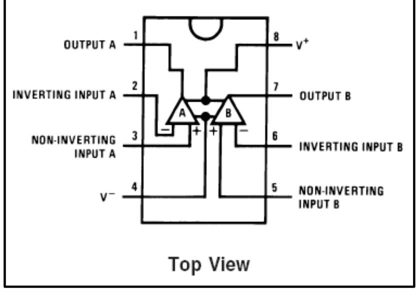
where

T is the temperature in degrees Kelvin
T_o is a reference temperature (typically 300°K)
B is a constant
R_O is the value of R_T when T = T_O

R_o and B are given by the manufacturer

Note: R_o = 8796.4 & B = 3964.2

LF 353 Dual Operational Amplifier



Temp C°	R_T (KΩ)
69.5	1.729
62.1	2.169
57.0	2.612
51.7	3.170
46.0	3.940
40.6	4.874
33.3	7.531
26.0	8.723
18.3	13.291
12.8	17.280
6.6	22.900
2.3	28.138
0.4	31.674

Temp (C°)	V_O (volts)
66.9	3.8227
55.0	2.9667
48.1	2.6254
43.8	2.1572
37.3	1.8114
34.1	1.4530
30.0	1.1131
21.0	0.57036
14.7	0.14125
8.0	-0.1336
1.2	-0.44755

Figure 8 – Thermistor Schematic and Conversion Tables.

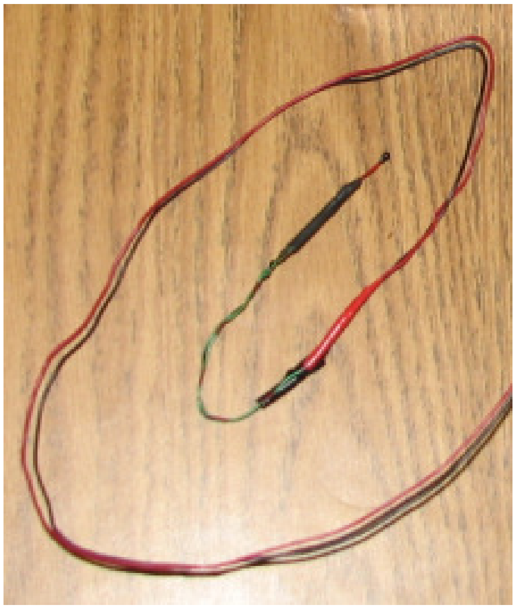


Figure 9 – Thermistor.

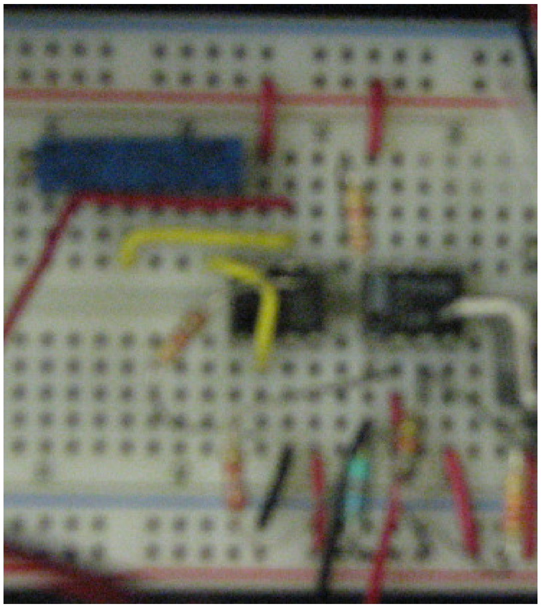


Figure 10 – Thermistor Circuit.

Microphone

The purpose of the microphone is to check to see if the bathtub is empty (has water but no human). It will detect sound waves and then send a signal to the microcontroller based off these sound waves. The microcontroller will analyze the signal and determine the appropriate action to take. Ideally, we want a microphone that is small and able to interface easily with the microcontroller.

We used a Crown Sound Grabber II PZM illustrated in Figure 11. Here are its specs:

- Frequency response (typical): 50 Hz to 16 kHz.
- Polar pattern: Hemispherical (half-omni) on a large surface.
- Impedance: 1600 ohms, unbalanced.
- Sensitivity: 20mV/Pa (-54 fBV/Pa).
- Power sensitivity: -42 dBm.
- Cable: 10 foot with mini phone plug, ¼” phone plug and micro phone plug adapters.
- Power: One 1.5v AAA alkaline battery.

Figure 11 – Crown Sound Grabber II PZM.

A PZM microphone helps reduce noise more than a conventional microphone. Figure 12 below illustrates this.

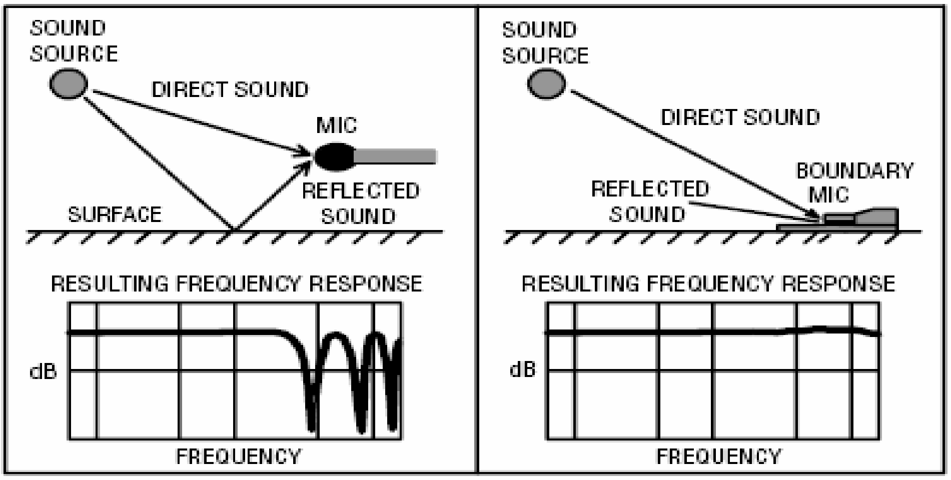
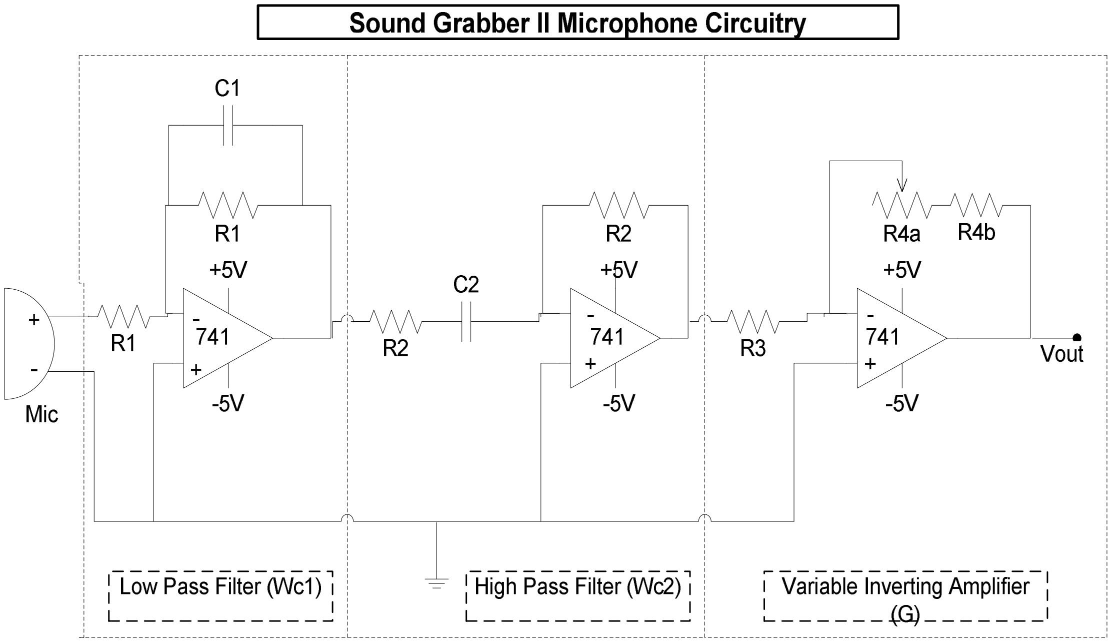
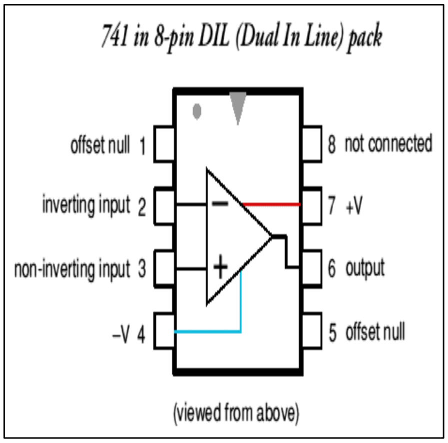


Figure 12 – Conventional mic vs. PZM mic.

We sent the microphone signal through a low-pass filter, used to filter out high frequency noises, and fed the resulting signal into a high-pass filter, used to filter out low frequency noises. We then sent the filtered signal to an amplification circuit to amplify it enough for the microcontroller to recognize the appropriate level of sound to determine that someone is in the tub. The circuit schematic and equations are illustrated in Figure 13.





Component Values	

R1 = 10 kiloOhms
C1 = 8.8 nanoFarads
Wc1 = 1808 Hz
	R2 = 10 kiloOhms
	C2 = 0.32 microFarads
	Wc2 = 49.7 Hz

R3 = 1 kiloOhms
R4a = 50 kiloOhms Variable
R4b = 10 kiloOhms
G = 10.0 – 60.0	

Microphone Equations	
Wc1 (Low Pass Cutoff Frequency) Wc1 = 1/(2*Pi*R1*C1)	Wc2 (High Pass Cutoff Frequency) Wc2 = 1/(2*Pi*R2*C2)
G (Amplifier Gain) G = (R4a + R4b)/R3	

Figure 13 – Microphone Schematic and Calculations.

Figure 14 below, shows a sample of what the analog signal coming from the microphone looked like.

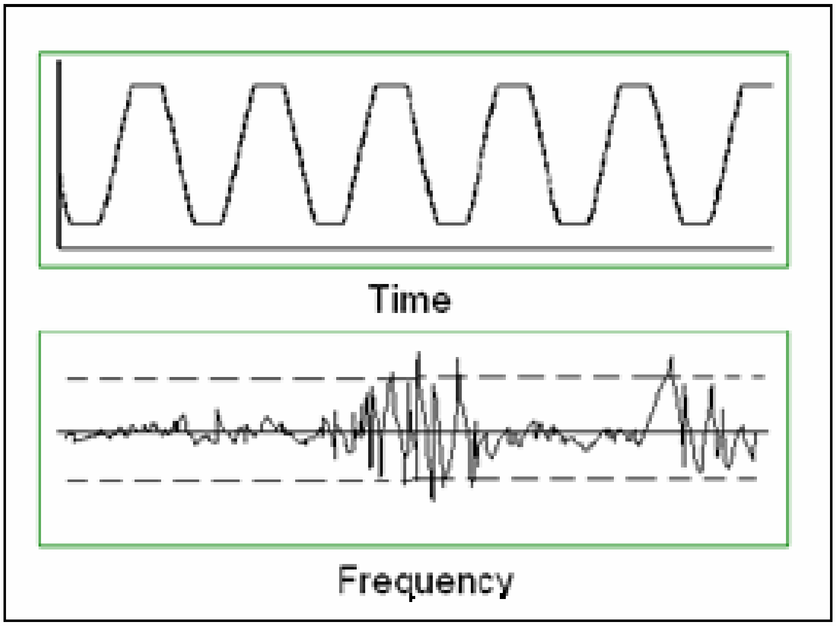


Figure 14 – Waveforms generated from microphone.

The dotted line is the amplitude at which the microcontroller detects a signal from the microphone circuitry. We determined this to be just above 2.0V. To reduce false alarms we set the variable resistance of the amplifier for the microphone to amplify it such that only a direct thud or close range sound would generate a signal of a magnitude of 2V.

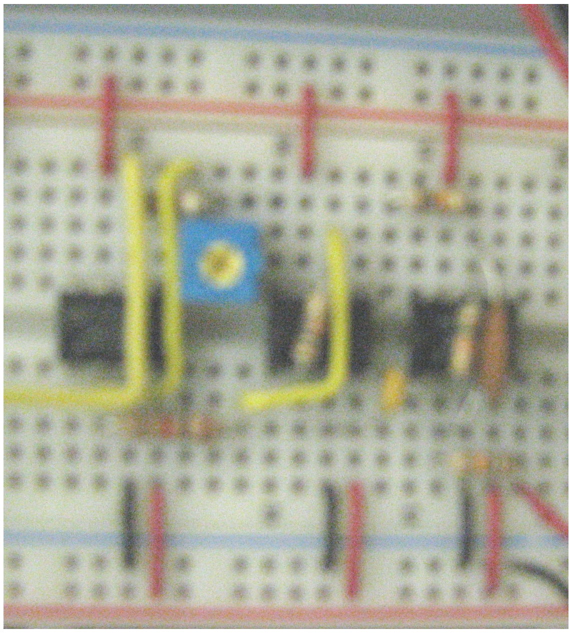


Figure 15 – Microphone Circuit.

Alarm

The purpose of the alarm is to warn you when the water temperature in the bathtub is too hot, or when the bathtub has been empty too long. The microcontroller will send a signal to the alarm if it has determined the water temperature is too hot or the bathtub has been empty too long.

We used a 12 VDC Piezo Siren for the alarm system. Figure 16 illustrates it. Note: No additional circuitry was required for the alarm.

• Voltage range:	6-14VDC
• Rated voltage:	12VDC
• Current consumption:	150mA max at 12VDC
• Sound pressure level:	102dB min at 30cm/12VDC
• Operating frequency:	2000-4500Hz
• Operating temperature:	-4 F to +140 F (-20 C to +60 C)

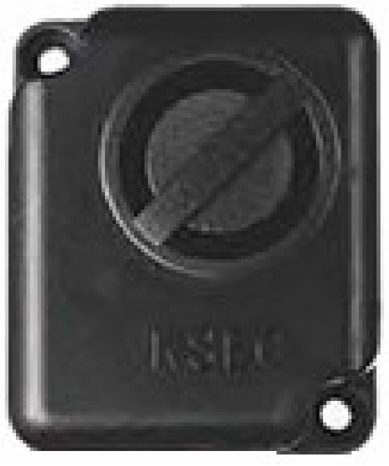


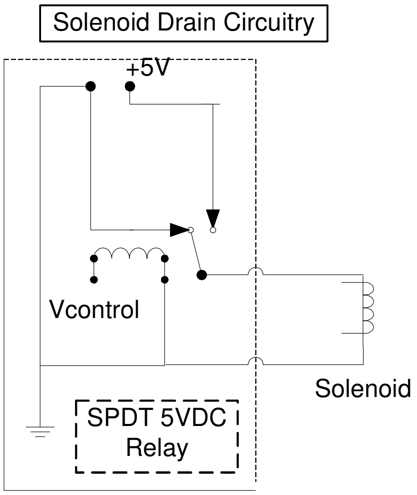
Figure 16 – Alarm.

Draining Unit

The purpose of the draining unit is to drain (unplug) the bathtub. The microcontroller will send a signal to the draining unit if it hasn't received a response within the 10 second time frame after the alarm has been activated.

We built the draining unit from a relay switch, a solenoid, and some rope. The relay switch is attached to a +5V signal and the solenoid. The solenoids cylinder is tied (with rope) to the draining mechanism on the bathtub. The microcontroller will send a signal to the relay causing the relay to switch positions sending a +5V signal directly to the solenoid which will cause the solenoid to pull in its cylinder which in turn pulls down on the draining mechanism on the bathtub and causes the water to drain. The circuit schematic and specs are illustrated in Figure 17.

Note: When the solenoid is connected to the same +5V as the microcontroller then it resets the microcontroller every time the microcontroller sends a signal to the relay switch. We believe the reason for this is because the microcontroller isn't sending enough current to the solenoid and so the solenoid takes power from the microcontroller. We still need to build a circuit that will amplify the current coming from the microcontroller to the relay switch that controls the draining unit.



Solenoid & Relay Specs	
Solenoid (11-I-6 VDC Guardian Electric)	Relay
Coil Resistance = 1.90 Ohms Operating Voltage = 5-6V Duty Cycle = Intermittent Load Force (Ounce-Inch) = 45-0.12 / 10-0.50 Operation = Pull	Type = SPDT Coil Voltage = 5 VDC Pick-up Voltage = 3.5 VDC Drop-out Voltage = 0.25 VDC Coil Resistance = 56 Ohms Nominal Current = 89.3 mA

Figure 17 – Draining Unit Schematic and Specs.

Figure 18 – Solenoid.

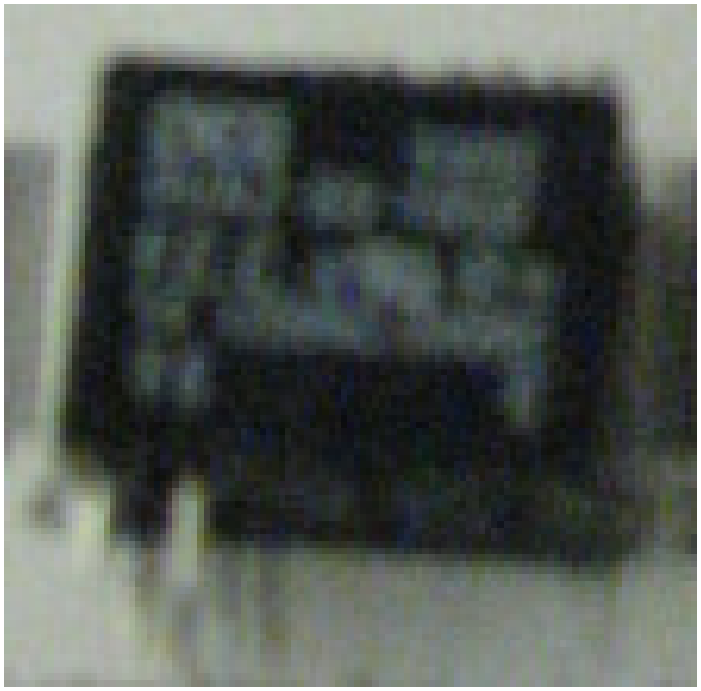
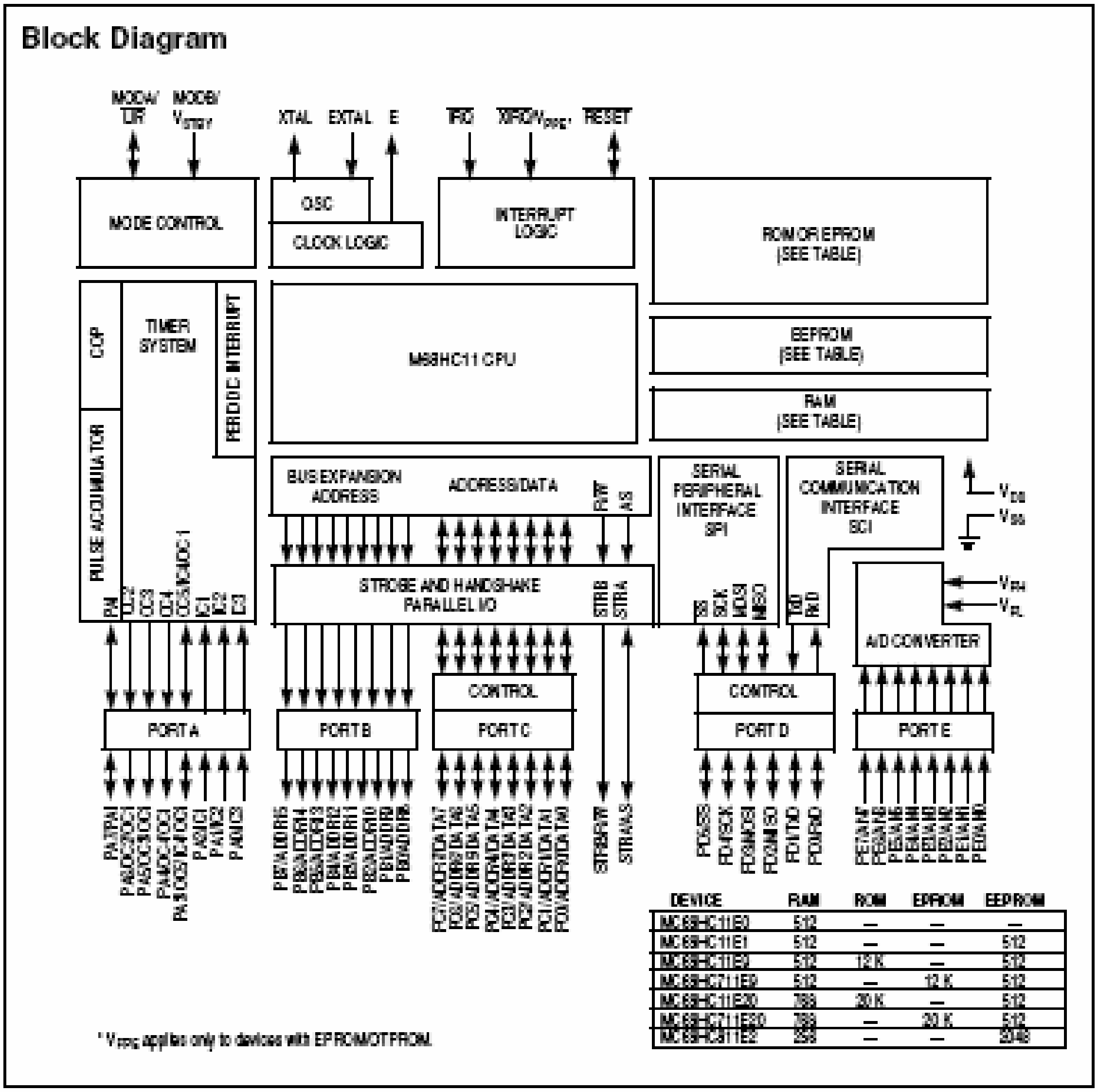


Figure 19 – Relay Switch.

Microcontroller

The purpose of the microcontroller is to act as the brains of the system. All of the systems hardware components (except the power supply) are interfaced through the microcontroller. The microcontroller will analyze input signals (thermistor & microphone) and determine the appropriate action to take.

We used a Motorola MC68HC11E1 microcontroller. It's the same microcontroller that we used in ECE 3720 so we we're familiar with its capabilities and instruction set. It has 512 bytes of RAM and 512 bytes of EEPROM, which is more than we need for our system. It has freeware which includes a C compiler, an assembly compiler, and a simulator (wookie) to aid in debugging. Its internal mapping is illustrated in Figure 20.



DEVICE	RAM	ROM	EPROM	EEPROM
MC68HC11E0	512	—	—	—
MC68HC11E1	512	—	—	512
MC68HC11E9	512	12 K	—	512
MC68HC711E9	512	—	12 K	512
MC68HC11E20	768	20 K	—	512
MC68HC711E20	768	—	20 K	512
MC68HC811E2	256	—	—	2048

Figure 20 – Motorola MC68HC11E1 Microcontroller.

The microcontroller is interfaced with the following components:

- *Thermistor*: The thermistor is connected to the PA2 input pin on the microcontroller. The microcontroller is programmed to continually check its PA2 input pin to see if it detects any voltage on that pin. If it detects voltage on PA2 then that means that the thermistor detected temp > 40 C and the microcontroller needs to take the appropriate action (sound the alarm). Note: the thermistor will not send a voltage signal large enough for the microcontroller to see unless it detects temp > 40 C.
- *Microphone*: The microphone is connected to the PA0 input pin on the microcontroller. The microcontroller is programmed to interrupt every time it receives a rising edge on its PA0 input pin. If it receives a rising edge on PA0 then that means that the microphone detected sound and the microcontroller needs to take the appropriate action (reset & clear all variables).
- *Alarm*: The alarm is connected to the PA5 output pin on the microcontroller. The microcontroller is programmed to send a signal to its PA5 output pin if it has detected a signal on its PA2 input pin (thermistor) or if it hasn't received a rising edge on its PA0 input pin (microphone) for a given amount of time (up to 4 min 15 sec).
- *Draining Unit*: The draining unit is connected to the PA6 output pin on the microcontroller. The microcontroller is programmed to send a signal to its PA6 output pin if it hasn't received a response within the 10 second time frame after it sent a signal to its PA5 output pin (alarm).

When we decided how to interface each of the above components with the microcontroller, we had to test each input and output pin. We first determined the amount of voltage it took for the input pins to acknowledge the signal. Then we determined how much voltage it put out on its output pins. The circuit schematic is illustrated in Figure 21 and the source code can be found in Appendix B.

Note: We originally thought about using the Motorola MC68HC811E2 microcontroller. It has 2048 bytes of EEPROM and we weren't sure how much memory we'd need for our struggle detection analysis. Instead we decided to add an external memory device to the MC68HC11E1 microcontroller if we needed more memory. It turns out that 512 bytes of EEPROM was enough for our system, but then again we didn't get struggle detection working like we wanted to. If we could implement struggle detection how we wanted, then the MC68HC811E2 microcontroller would probably be a better choice.

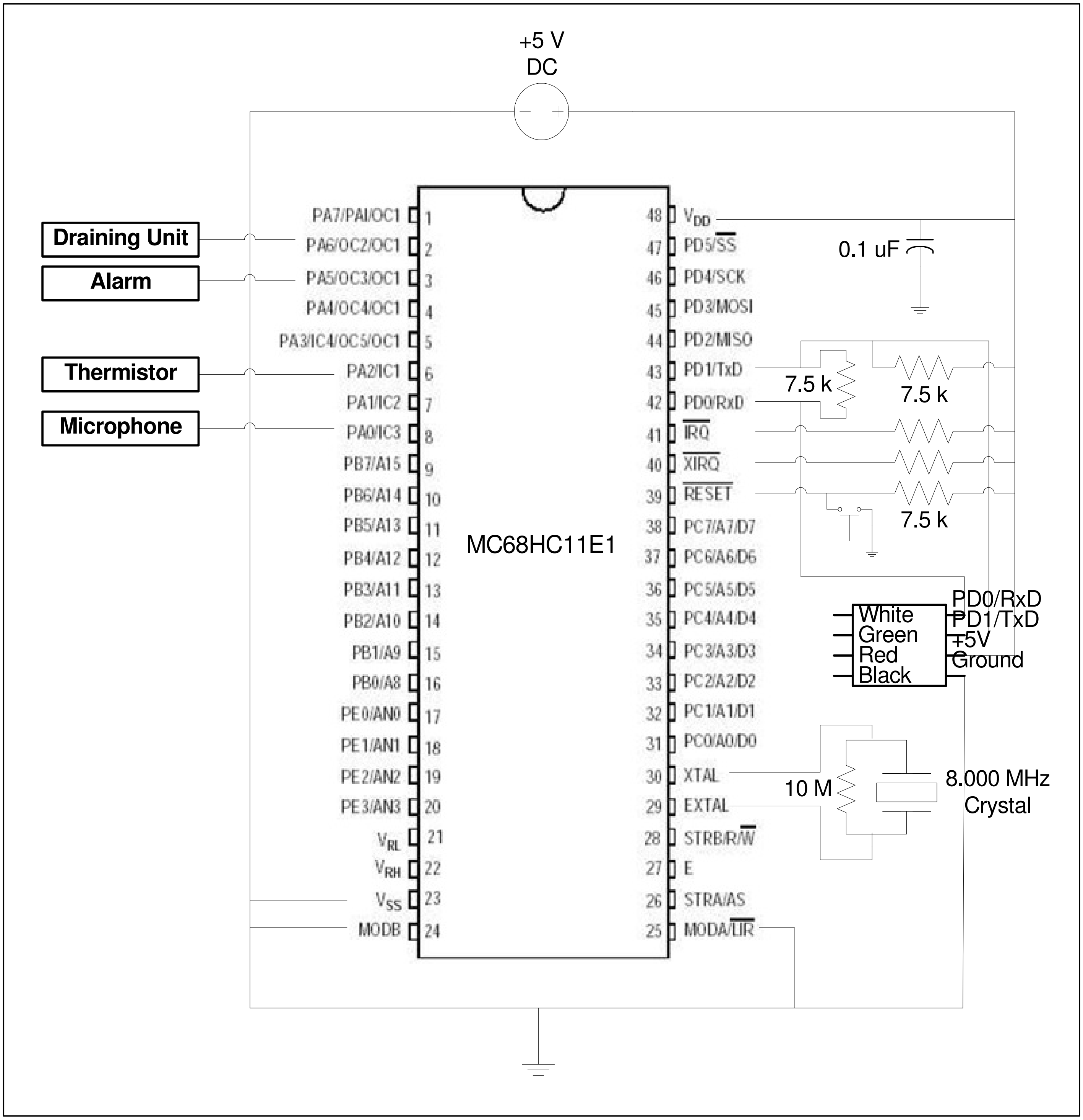


Figure 21 – Microcontroller Schematic.

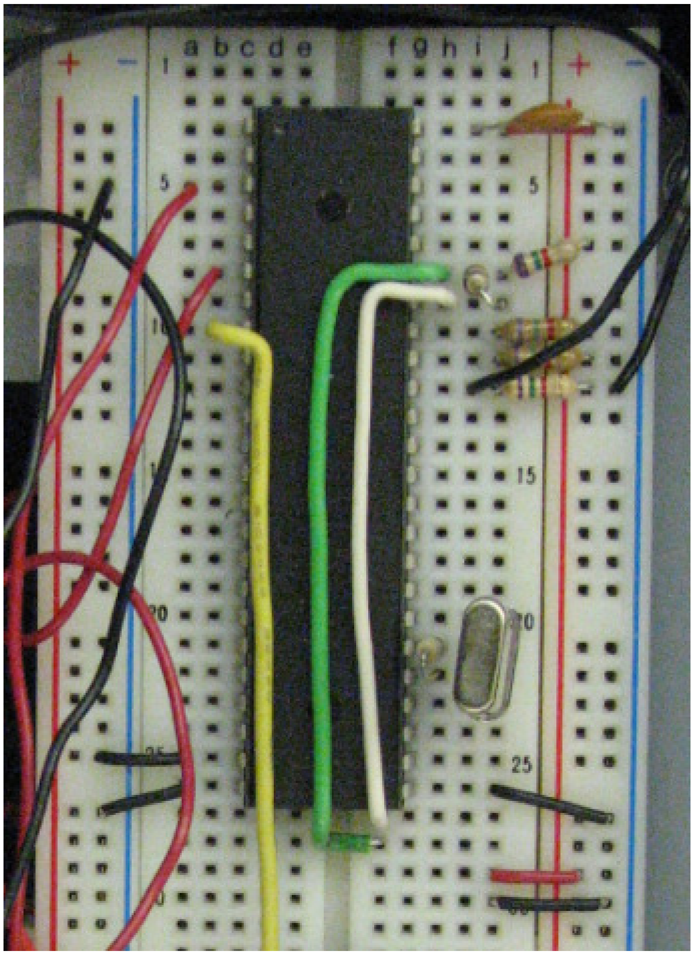


Figure 22 – Microcontroller Circuit.

Conclusion

We learned a great deal during the course of this project. Both of us were more software oriented but after completing this project then our hardware background are stronger. I think the main thing that was difficult was being able to manage our other classes along with work along with our families. The whole process was like on for 1-2 weeks and then off for 1-2 weeks. We need to learn how to manage our time better. We've included some ideas about improvements and extra features that one could add to our project.

Improvements

If we had a better understanding about hardware then we would like to implement the following improvements:

- *Efficient Power Supply*: The only thing powered by batteries would be the microphone (1 AAA battery).

Extra Features

If we had more time then we'd like to jazz up CPRFD with the following extra features:

- *Struggle Detection*: Determine if someone is struggling in the bathtub.
- *Wireless Phone Module*: Once the system begins to drain the bathtub it will also make a phone call to a pre-programmed number (most likely 911) and play a recorded distress message.

In closing, this project (CPRFD) will help reduce the number of fatalities caused by bathtub drowning & burns. We don't want people to let there guard down thinking they don't have to drain the bathtub or check the water temperature because this product will do it for them. Instead we want to provide this product as an additional safety feature that complements ones common knowledge to make sure the bathtub is drained properly and the water temperature is safe. For example, just because you have seat belts in your car doesn't mean you can drive recklessly and expect to be 100% safe. Seat belts are a last resort safety feature, as is our project. Ask yourself how much your child's life is worth to you, if the answer is priceless then so is this product.

Acknowledgements

We would like to acknowledge and thank the following people and resources for their help:

1. *Professor Al Davis* – He helped us with ideas and initial designs (especially power designs). He also did some research on microphones and draining units for us.
2. *Junior Hardware Laboratory* – They gave us free parts (resistors, capacitors, and voltage regulators).
3. *Mark (EE Stockroom Employee)* – He helped us analyze voltage regulator diagrams. He also allowed us to work in the ECE 3720 lab after hours and on weekends.
4. *Ronnie Boutte* – He helped us analyze operational amplifiers.
5. http://cva.stanford.edu/classes/cs99s/datasheets/LM340.pdf - We modeled our positive power supplies (LM7805 & LM7812) from the LM78XX diagram on page 11. We also used this diagram in our power supply documentation.
6. http://www.jaycar.com.au/images_uploaded/LM7905.PDF - We modeled our negative power supplies (LM7905CT & LM7912CT) from the LM79XXCT diagram on page 1. We also used this diagram in our power supply documentation.
7. http://www.cdc.gov/NASD/docs/d000701-d000800/d000702/d000702.html - This website gave us a lot of our statistics.
8. http://www.med.umich.edu/1libr/pa/pa_hotwatr_hhg.htm - This website helped us decide what range of water temperature is too hot.
9. http://www.drspock.com/article/0,1510,5837,00.html – This website helped us decide what range of water temperature is too hot (temp > 104 F).
10. http://www.texloc.com/closet/cl_fah_cel_chart.html - This website helped us go back and forth between Fahrenheit and Celsius.
11. *ECE 1000 Lab 1 Handout* – We modeled our thermistor circuit from the thermistor diagram on page 2. This handout also helped us understand what an adequate model of a thermistor is.
12. *ECE 1000 Lab 1 Notebook* – We used the temperature vs. resistance table and the temperature vs. voltage table as guides to help us adjust the resistance in our thermistor circuit.
13. http://ccrma.stanford.edu/courses/250a/docs/opamps/LF353.pdf - In our thermistor documentation we used the LF353 pin-out diagram on page 1.

14. http://www.st-andrews.ac.uk/~jcgl/Scots_Guide/datasheets/Opamps/741.html - In our thermistor & microphone documentation we used the 741 pin-out diagram on page 1.

15. http://www.crownaudio.com/pdf/mics/101502.pdf - This website helped us understand how the microphone works. We also used one of its microphone pictures in our microphone documentation.

16. http://www.crownaudio.com/pdf/mics/136367.pdf - This website helped us understand how the microphone works. We also used one of its diagrams in our microphone documentation.

17. *ECE 1000 Book "Electric Circuits"* – This book helped us design our microphone circuit (low pass filter, high pass filter, and inverting amplifier).

18. *M68HC11E Series Programming Reference Guide* – This reference guide helped us understand how the microcontroller works and how to program it. We also used its block diagram in our microcontroller documentation.

19. *M68HC11 Microcontrollers Reference Manual (Appendix A)* – This reference manual helped us understand the instruction set for the microcontroller and how to program it.

20. *ECE 3720 Lab 1 Handout* – We modeled our microcontroller circuit from the instructions on pages 2-4. This handout also helped us understand how the microcontroller works and how to program it.

21. *ECE 3720 Book "Embedded Microcomputer Systems"* – This book helped us understand how the microcontroller works and how to program it.

Appendix A – Bill of Materials

OVERVIEW

Qty	Cost	Vendor
4	FREE	Junior Hardware Laboratory
12	$4.00	EE Stockroom
1	$20.00	Newark InOne
17	$37.61	Radio Shack
1	$70.00	Crown

Total	35	$131.61	5

DETAILS

Qty	Part	Cost	Vendor
1	Project Enclosure	$5.99	Radio Shack
1	Automotive Illuminated Rocker Switch	$3.99	Radio Shack
1	SPST Momentary Pushbutton Switch	$3.29	Radio Shack
4	9V Alkaline Enercell Battery	$9.99	Radio Shack
4	9V Battery Holder	$1.98	Radio Shack
4	Heavy Duty 9V Battery Snap Connectors	$2.59	Radio Shack
2	0.33 uF Solid Tantalum Polarized Capacitor	$0.50	EE Stockroom
2	0.1 uF Solid Tantalum Polarized Capacitor	$0.50	EE Stockroom
2	2.2 uF Solid Tantalum Polarized Capacitor	$0.50	EE Stockroom
2	1 uF Solid Tantalum Polarized Capacitor	$0.50	EE Stockroom
1	LM7805 +5 VDC Voltage Regulator	FREE	Junior Hardware Laboratory
1	LM7905CT -5 VDC Voltage Regulator	FREE	Junior Hardware Laboratory
1	LM7812 +12 VDC Voltage Regulator	FREE	Junior Hardware Laboratory
1	LM7912CT -12 VDC Voltage Regulator	FREE	Junior Hardware Laboratory
4	Heat Sink	$2.00	EE Stockroom
1	Sound Grabber II PZM Microphone	$70.00	Crown
1	12 VDC Piezo Siren	$5.29	Radio Shack
1	Solenoid	$20.00	Newark InOne
1	Relay Switch	$4.49	Radio Shack

Appendix B – Microcontroller Program

```
*************************************************************************
* James Young & Justin Young
* Computer Engineering
* Senior Project (Fall 2006)
*
* Project:     CPR For Dummies (CPRFD)
* Description: Bathtub drown/burn prevention.
*************************************************************************

*************************************************************************
* Equates
*************************************************************************
* Registers
REGBS                   EQU    $1000  start of registers

PORTA                   EQU    $1000

PORTB                   EQU    $1004

PORTC                   EQU    $1003
DDRC            EQU     $1007  Port C I/O pins (0 = input, 1 = output)

TCNT            EQU     $100E  current time
TOC5            EQU     $101E  OC5 Compare Reg
TCTL2           EQU     $1021  IC Active Edge Reg
TMSK1                   EQU    $1022  IC and OC Interrupt Control Reg
TFLG1           EQU     $1023  IC and OC Flag Reg

OCRate                  EQU    $C350 OC interrupt rate (50000 cycles = 25 ms)

*************************************************************************
* Stack
*
* The stack grows downward from the last byte before the pseudo-vector
* jump table.
*************************************************************************
STACK                   EQU    $00C3

*************************************************************************
* Helper variables (un-initialized)
*
* These will be placed in the first 256 bytes of internal RAM to support
* direct addressing.  They must be explicitly initialized by the code.
*************************************************************************
        ORG     $0000
ICCount         RMB     1               stores the number of IC3 interrupts
OCCount         RMB     1               stores the number of OC5 interrupts
```

```
sCount  RMB     1               stores the number of seconds elapsed

**********************************************************************
* Entry point for program. Initialize stack and pseudo-vectors.
**********************************************************************
*       ORG     $C000           used for wookie simulation
        ORG     $B600
Init    sei                     disable interrupts to make atomic
        lds     #STACK                  set the stack pointer
        ldy     #REGBS                  use IY for accessing registers

*       **********************************************************************
*       * Initialize port A.
*       *       PA0 = microphone input
*       *       PA2 = thermistor input
*       *       PA5 = alarm output
*       *       PA6 = drain output
*       **********************************************************************
        clra
        staa    PORTA

*       **********************************************************************
*       * Initialize port B (displays microphone detection).
*       **********************************************************************
        clra
        staa    PORTB

*       **********************************************************************
*       * Initialize port C (displays seconds elapsed).
*       **********************************************************************
        ldaa    #$FF            set Port C pins (bits 7-0 outputs)
        staa    DDRC
        clra
        staa    PORTC

*       **********************************************************************
*       * Initialize interrupt registers.
*       **********************************************************************
        ldaa    #$01
        staa    TCTL2           capture on rising edge (PA0 mode)
        ldaa    #$09
        staa    TMSK1                   arm IC3I (microphone) & OC5I (clock)
        staa    TFLG1           clear IC3F & OC5F

*       **********************************************************************
*       * Initialize interrupt jump pseudo-vectors.
*       **********************************************************************
        ldaa    #$7E            Op code for JMP
        staa    $00E2           Timer input capture 3
        staa    $00D3           Timer output compare 5
        ldx     #IC3Han
        stx     $00E3           JMP IC3Han

        ldx     #OC5Han
        stx     $00D4           JMP OC5Han

*       ************************************************************************
*       * Initialize helper variables.
*       ************************************************************************
        clr     ICCount
        clr     OCCount
        clr     sCount
        ldd     TCNT            current time
        addd    #OCRate                 first OC5 in 25 ms (50000 cycles)
        std     TOC5
        cli                     enable interrupts

Start   clra
        staa    PORTA
        clr     OCCount
        clr     sCount
        ldaa    TMSK1
        oraa    #$01
        staa    TMSK1

Main    ldaa    PORTA
        anda    #$04            thermistor = PA2 (0000 0100)
        cmpa    #$04            check thermistor value
        beq     Alarm1          branch if thermistor = hot

        ldaa    sCount
        staa    PORTC                   display number of seconds elapsed
        cmpa    #10             check s counter
        bgt     Alarm2          branch if sCount > 10 seconds

        bra     Main

Alarm1  ldaa    TMSK1                   old value
        anda    #$FE            IC3I = 0 (1111 1110)
        staa    TMSK1                   disarm IC3I (microphone)

        ldaa    #$20            alarm = PA5 (0010 0000)
        staa    PORTA                   sound the alarm

        clr     OCCount                 reset OC counter
        clr     sCount          reset s counter

Wait1   ldaa    sCount
        staa    PORTC                   display number of seconds elapsed
        cmpa    #10             check s counter
        bgt     Drain           branch if sCount > 10 seconds
        ldaa    PORTA
        anda    #$04            thermistor = PA2 (0000 0100)
        cmpa    #$04            check thermistor value
        bne     Start           branch if thermistor != hot
        bra     Wait1

Alarm2  ldaa    TMSK1                   old value
        anda    #$FE            IC3I = 0 (1111 1110)
        staa    TMSK1                   disarm IC3I (microphone)

        ldaa    #$20            alarm = PA5 (0010 0000)
        staa    PORTA                   sound the alarm

        clr     OCCount                 reset OC counter
        clr     sCount          reset s counter

Wait2   ldaa    sCount
        staa    PORTC                   display number of seconds elapsed
        cmpa    #10             check s counter
        bgt     Drain           branch if sCount > 10 seconds
        bra     Wait2

Drain   ldaa    #$60            drain = PA6 (0110 0000)
        staa    PORTA                   drain the bathtub
        bra     Drain


************************************************************************
* Subroutines
************************************************************************

************************************************************************
* IC3Han - Microphone Detection
************************************************************************
IC3Han
        sei                     disable interrupts to make atomic
        inc     ICCount                 update IC counter (detected sound)
        ldaa    ICCount
        staa    PORTB                   display number of sounds detected

        clr     OCCount                 reset OC counter
        clr     sCount          reset s counter
        ldd     TCNT            current time
        addd    #OCRate                 next OC5 in 25 ms (50000 cycles)
        std     TOC5

IC3Han_
        ldaa    #$09            clear IC3F & OC5F
        staa    TFLG1           acknowledge IC3 interrupt, ignore OC5 interrupt
        cli                     enable interrupts
        rti
**********************************************************************


**********************************************************************
* OC5Han - Clock
**********************************************************************
OC5Han
        sei                     disable interrupts to make atomic
        inc     OCCount                 update OC counter (25 ms elapsed)
        ldaa    OCCount
        cmpa    #40             check OC counter (40 * 25 ms = 1 second)
        bne     OC5Han_                 branch if OCCount != 40 iterations
        inc     sCount          update s counter (1 second elapsed)
        clr     OCCount                 reset OC counter

OC5Han_
        ldd     TOC5
        addd    #OCRate                 next OC5 in 25 ms (50000 cycles)
        std     TOC5
        ldaa    #$08            clear OC5F
        staa    TFLG1           acknowledge OC5 interrupt
        cli                     enable interrupts
        rti
**********************************************************************


**********************************************************************
*Delay
*       ldx     #$F000                  delay value
*loop   dex                     x = x - 1
*       bne     loop
*       rts
**********************************************************************
```