F.A.M.

Simon Fraser University
8888 University Dr.
Burnaby, BC, Canada

March 8, 2012

Dr. Andrew Rawicz
Simon Fraser University
8888 University Drive
Burnaby, B.C. V5A 1S6

Re: ENSC 440 Design Specification for a Fall Detection System for Seniors

Dear Dr. Rawicz:

Enclosed is a document which describes the design specification of the fall detection system for seniors being developed by Fall Alert Mechanism (F.A.M.) Inc. We are designing and implementing a system that recognizes when its elderly user has fallen and sends an alert to notify a family member by sending a text message notification. In this way, the user will quickly receive any necessary assistance or medical attention. The system will consist of a portable accelerometer-based device and central base unit with text message capabilities.

The enclosed design specification document describes the hardware and software technologies we will apply to satisfy the device requirements outlined in our functional specification document. It also explains the reasoning behind our design choices and details our approach to detecting a fall. The technologies described in this document are intended for development of a proof-of-concept system. In order to make our product commercially available, some additional design work may be necessary.

Our group, F.A.M., consists of five skilled and enthusiastic engineering students: Behdad Jamshidi, Eric Swanlund, Nastaran Naghshineh, Ted Lee, and Zack Frehlick. If you have any questions or concerns about our proposal, please contact our designated spokesman, Zack Frehlick, by phone at (778)385-3590 or by e-mail at zfa2@sfu.ca.

Sincerely,

Nastaran Naghshineh

Nastaran Naghshineh

Enclosure: Design Specifications for a Fall Detection System for Seniors

F.A.M.

Design Specifications for a Fall Detection System for Seniors

Project Team:	Behdad Jamshidi Eric Swanlund Nastaran Naghshineh Ted Lee Zack Frehlick	**Submitted to:**	Dr. Andrew Rawicz Steve Whitmore School of Engineering Science Simon Fraser University
Contact Person:	Zack Frehlick	**Date issued:**	February 6, 2012

Executive Summary

Injuries related to falling down constitute one of the main threats to the health of today's senior citizens. In the United States alone, 2.2 million falls per year require emergency medical attention and nearly 20,000 of these falls eventually result in death [1]. Clearly, the prevention and treatment of falls is an area of great interest to our aging society. To that end, our group is developing the Fall Alert Mechanism (F.A.M.) system. A small portable device, when worn by a senior, will automatically detect falls and transmit an alert. A central device can then detect this alert and notify a designated contact via text message. In this way, the individual will receive medical attention as quickly as possible and quality of care will greatly increase.

The design specifications for our F.A.M. device provide a descriptive and informative overview of the design, implementation, and development of our product. In this document, we will discuss design considerations relative to the functional requirements as specified in the document *Functional Specification for a Fall Detection System for Seniors* [2]. All design choices for the F.A.M. device are included within this document, with supportive reasoning of component selection. Future design improvements of the F.A.M. device are also considered within this document.

The portable device will contain a radio-frequency (RF) transceiver, a microcontroller, an accelerometer and a battery pack. The accelerometer on the portable device measures acceleration in g (1 g = 9.8 m/s^2) in x, y, and z directions. The portable device microcontroller will test incoming acceleration data to detect a fall, and then use the transceiver to alert the central unit when a fall occurs. The central device will contain four main components: a second RF transceiver, a GSM shield and antenna for text messaging, an Arduino microcontroller board to oversee all operations and a small LCD display. The central device transceiver will receive alerts, signaling the GSM shield and Arduino module to send a text message to a phone number programmed into the system.

The resource requirements are provided within this specification; all hardware and software components are described in full detail. Software data flow charts are included, as well as hardware component schematics. A fully descriptive test plan for the system and its subcomponents (portable and central devices) is provided at the end of the document.

Table of Contents

List of Figures

List of Tables

Glossary

ADC	Analog-to-digital converter; a device that converts an analog voltage signal into a digital signal
AT	ATtention, as used in AT commands; a type of command used in communications
F.A.M.	Fall Alert Mechanism
FPGA	Field-Programmable Gate Array; a programmable semiconductor device
GSM	Global System for Mobile Communications
LED	Light-emitting diode; a semiconductor diode that glows when a voltage is applied.
LCD	Liquid crystal display.
MCU	Microcontroller Unit
RAM	Random Access Memory
RF	Radio frequency; a frequency or band of frequencies suitable for use in telecommunications
RISC	Reduced Instruction Set Computing
SMA	SubMiniature version A; a type of coaxial RF connector
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus

1. Introduction

Our Fall Alert Mechanism (F.A.M) device has the potential to help elderly people all over the world. Initially, our intention was to design our device specifically for home care centers and then expand into all home environments. We have since realized that making the device adaptable to all home environments should be a priority and updated our design to meet that goal. The fall detector device located on an elderly person's belt will sense when the user has fallen down. Once a fall is detected a signal is sent to a central device where two things happen: an LCD display shows that the person has fallen (for home care centers), and a text message is sent to a family member (for typical home use). The purpose of the device is to allow elderly people to be able to live more independently and with less supervision while remaining safe and secure. Eventually F.A.M. will also become integrated with cell phones to monitor the customer no matter where they go during the day. The future of health care for the elderly is arriving, with F.A.M. as part of a safer and more independent daily routine for our seniors.

1.1 *Scope*

This document outlines our design specifications of the F.A.M device. The different hardware and software components of the system will be described. Information about the materials and specifications for individual components will also be provided. This document is a guideline showing the approaches which we have been and will be following to make sure we fulfill all the necessary requirements given in our functional specification document [2].

1.2 *Intended Audience*

This document is of a technical nature and is intended to be understood by an audience with some technical engineering background. Specifically, the F.A.M. design team must be able to refer to and understand all sections of the document. It will be provided to design engineers as well as anyone who will be a part of manufacturing our product. This document can also be used as a reference in future user manuals and marketing documents, for those clients who want a more complete understanding of the product.

2. General Overview

The general system design of the F.A.M. system is presented in this section. This includes an overview of hardware components and their linkages, as well as a high level view of information flow through the system. Finally, sketches of the possible appearance of the central and portable components of the system are provided.

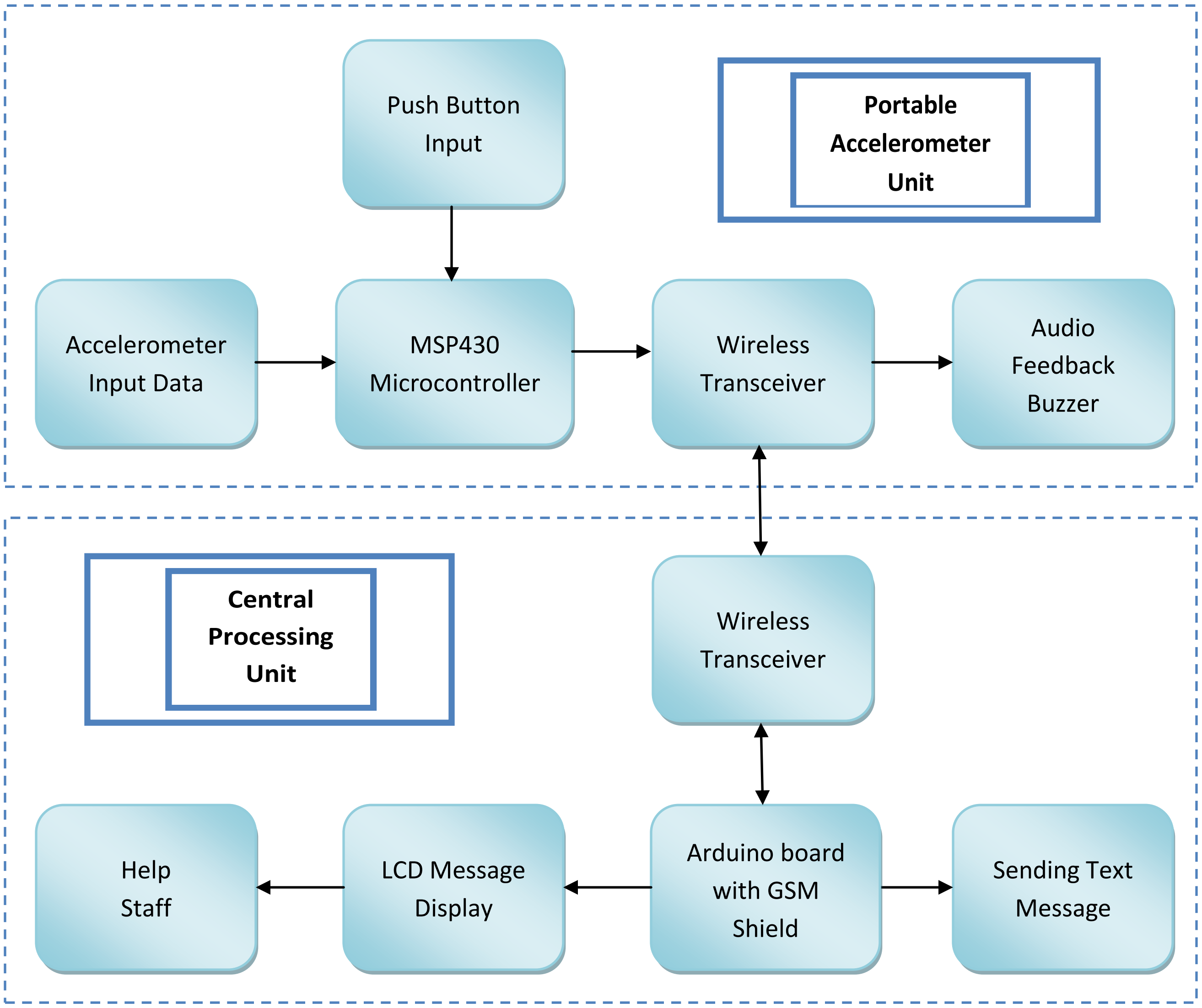

Figure 1 – General System Hardware Block Diagram

Figure 1 illustrates the general flow of information and linking of hardware components in the F.A.M. system. The portable accelerometer unit has an accelerometer and a push button as data inputs to a microcontroller running a fall detection algorithm. Upon detection of a fall, a signal is sent to the central unit using RF transceivers. The central unit sounds an alarm and displays an appropriate message to inform help staff. Since the central Arduino board with GSM shield has text messaging capabilities, the central unit can also send a text message to a family member. Figure 2 and 3 provide an example of what the Central unit and portable device will look like after development is complete.

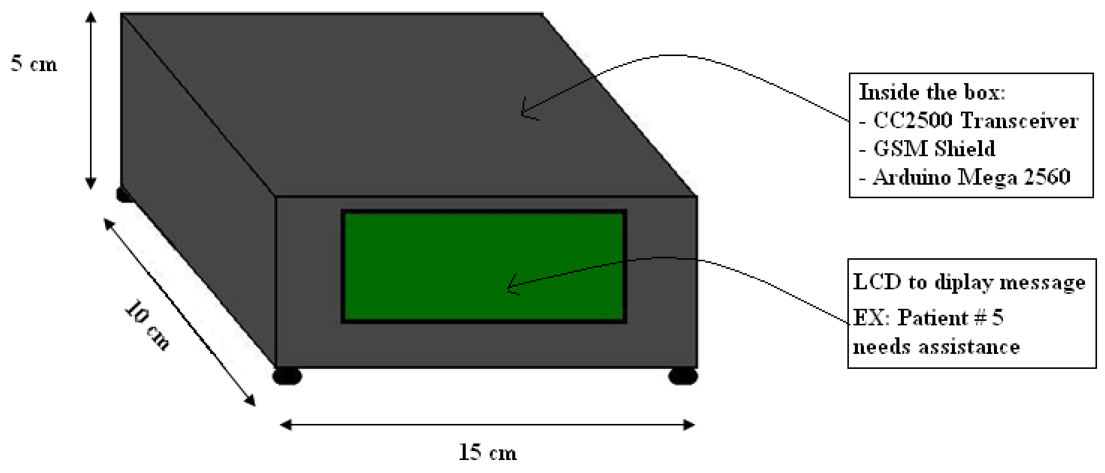

Figure 2 – Illustration of Central Unit

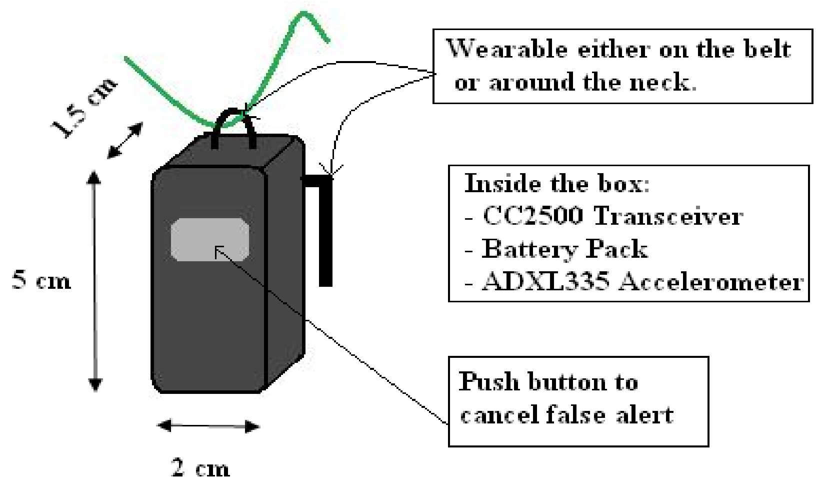

Figure 3 – Illustration of Portable Unit

Figure 4 below shows a flowchart of alert information through the entire system, starting in the portable unit and moving to the central unit.

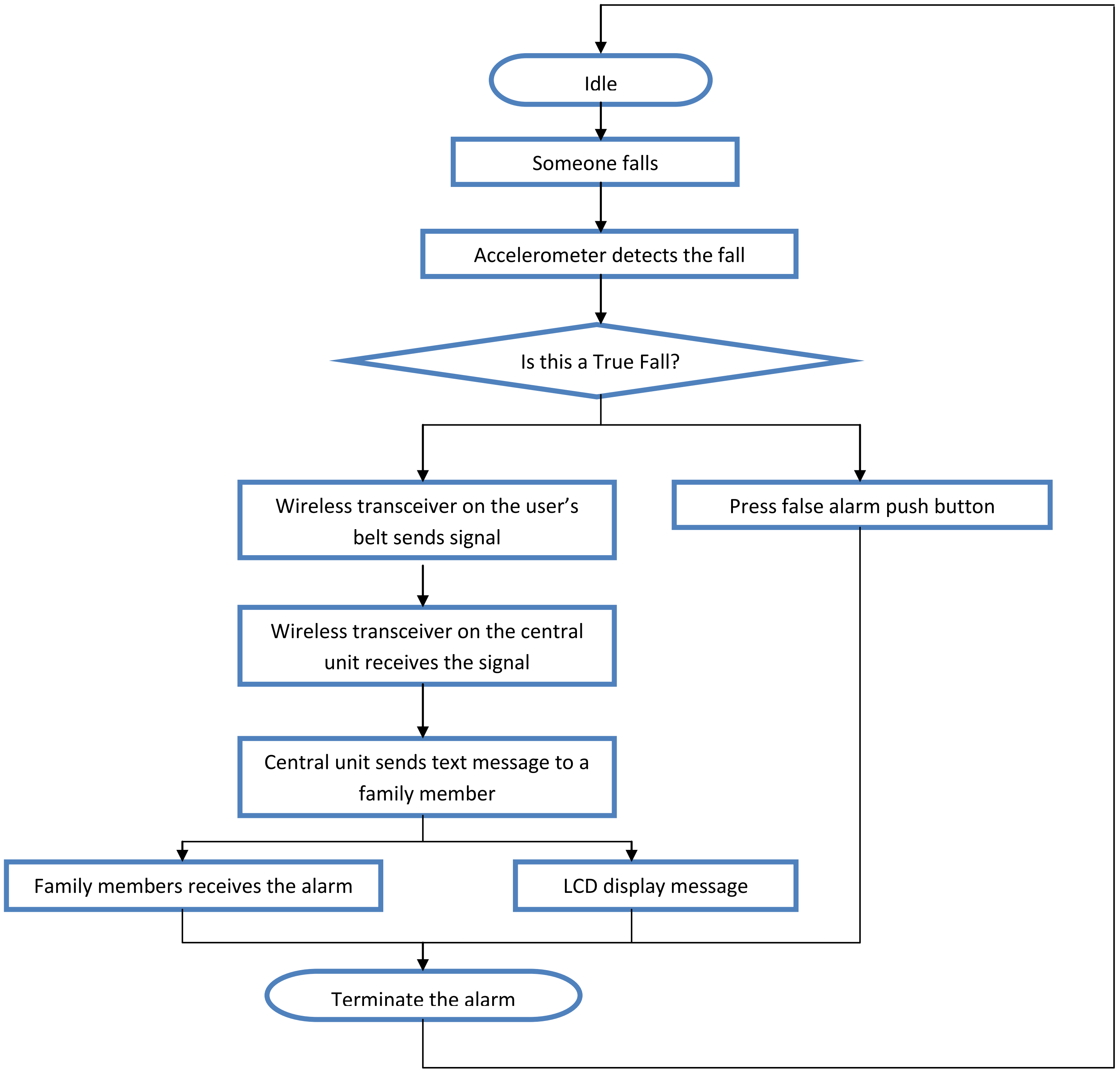
Idle
Someone falls
Accelerometer detects the fall
Is this a True Fall?
Wireless transceiver on the user's belt sends signal
Press false alarm push button
Wireless transceiver on the central unit receives the signal
Central unit sends text message to a family member
Family members receives the alarm
LCD display message
Terminate the alarm

Figure 4 – General System Flowchart

3. Central Control Unit

The following section outlines the design of the central control unit. The considerations to be accounted for include important functionalities such as reliably receiving alert messages, displaying alert messages and sending text message notifications. Generally speaking, the central device will be a stationary device, plugged in all times, which communicates with the portable unit through RF signals, with on-site help staff through an LCD display, and with designated family members through text message

3.1 *Central Unit Hardware Design*

3.1.1 Arduino Mega 2560

The central unit requires some type of microcontroller to oversee operations. While there are many options to choose from, the Arduino Mega board fits our needs nicely. The cost, at ~$100, is significantly less than that of an FPGA type board. While it does not have as much computing power, it does enough processing power and input pins to meet our fairly limited needs. Additionally, the Arduino development environment and software techniques are very easy to learn, which reduces our development.

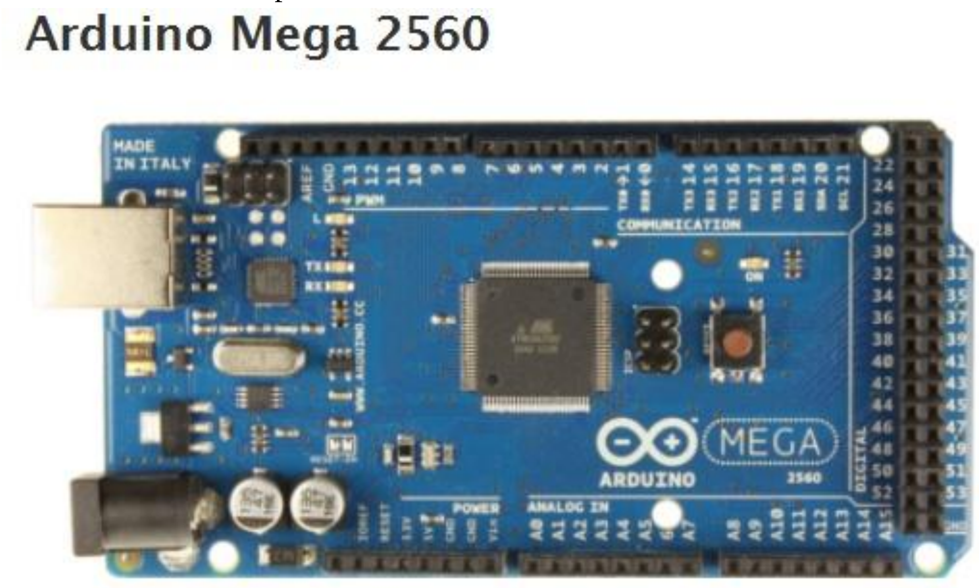

Figure 5 – Top View of the Arduino Mega Microcontroller Board

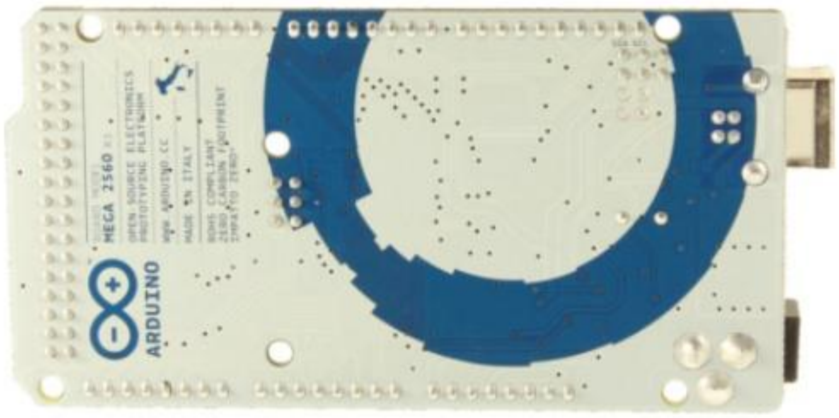

Figure 6 – Bottom View of the Arduino Mega Microcontroller Board

Operating voltage	5V
Input Voltage(recommended)	7-12V
Input Voltage limits	6-20V
Digital I/O Pins	52(14 PWM)
Analog Input Pins	16
DC Current per I/O pin	40ma
DC Current for 3.3V pin	50ma
Flash Memory	256KB of which 8KB used by boot loader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

Table 1 – Arduino Mega Specifications

Table 1 above lists several important specifications [3] of the Arduino board including clock speed, memory, current consumption, and operating voltage. The Arduino board can be powered in a variety of ways, including by DC power supply or USB connection. The method we are using is to power the board through by GSM shield's V_{in} pin; we will describe this process later. The board maximum voltage is 12V because of the potential for overheating and the minimum voltage is 7V to avoid oscillation instability. The V_{in} pin is where we can supply the board with external power. It has designated ground pins also that we use for the LCD and the GSM shield when connecting the devices.

The board has 16 analog input pins, 36 digital input pins, and also 4 designated UART Receive/Transmit serial communication ports. As an example, pin 18 and pin 19 are designated as the Rx1 and Tx1 pins, respectively, which together form a serial port. This feature will be used to communicate with our GSM shield to enable text messaging. Once this is set up we can program the Arduino board to control the GSM shield and use the device for telephone type communications like text messaging. Figure 7 on the following page shows the full layout of pins on the Arduino Mega.

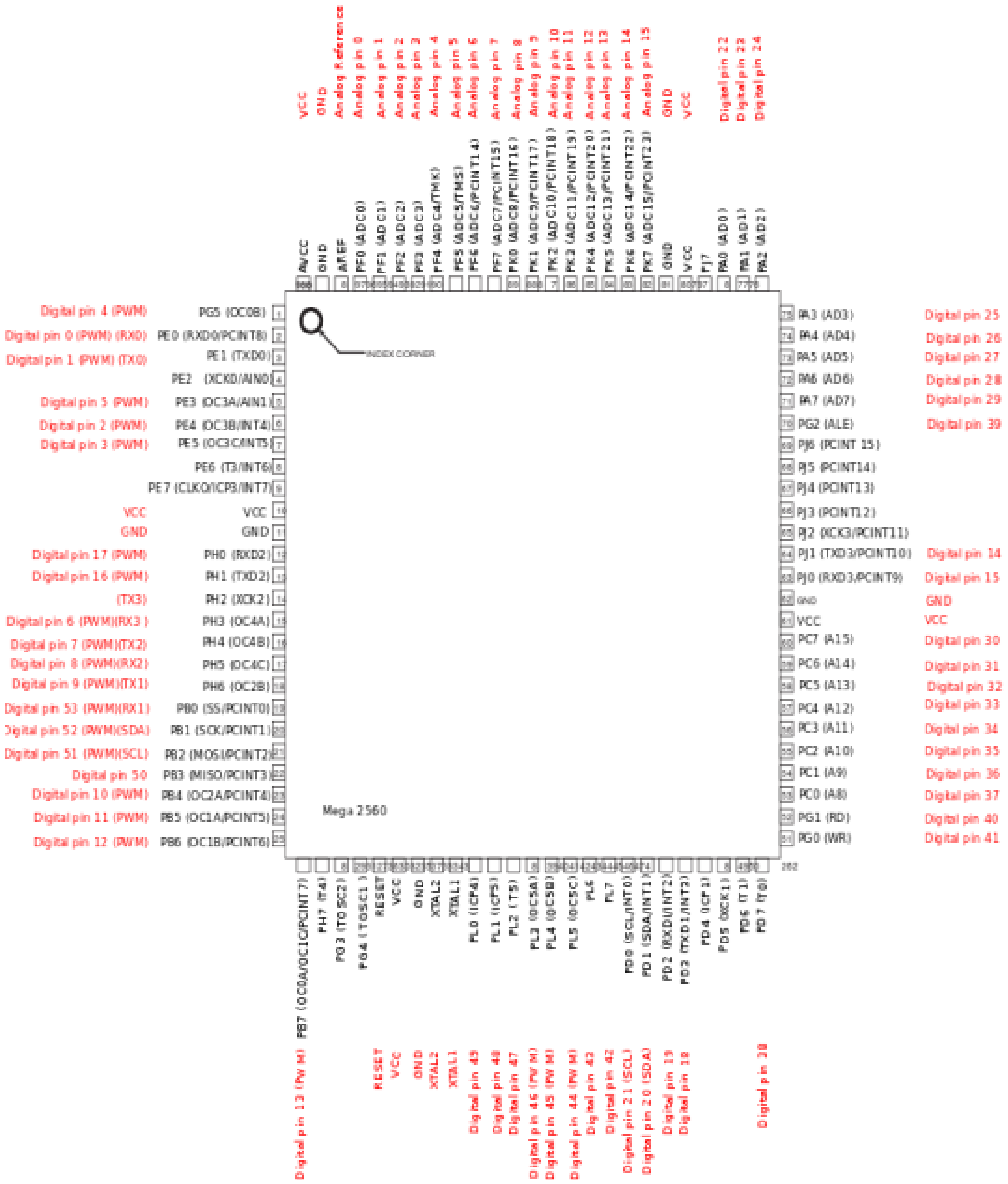

Figure 7 – Pin Layout of the Arduino Mega

3.1.2 LCD Text Display

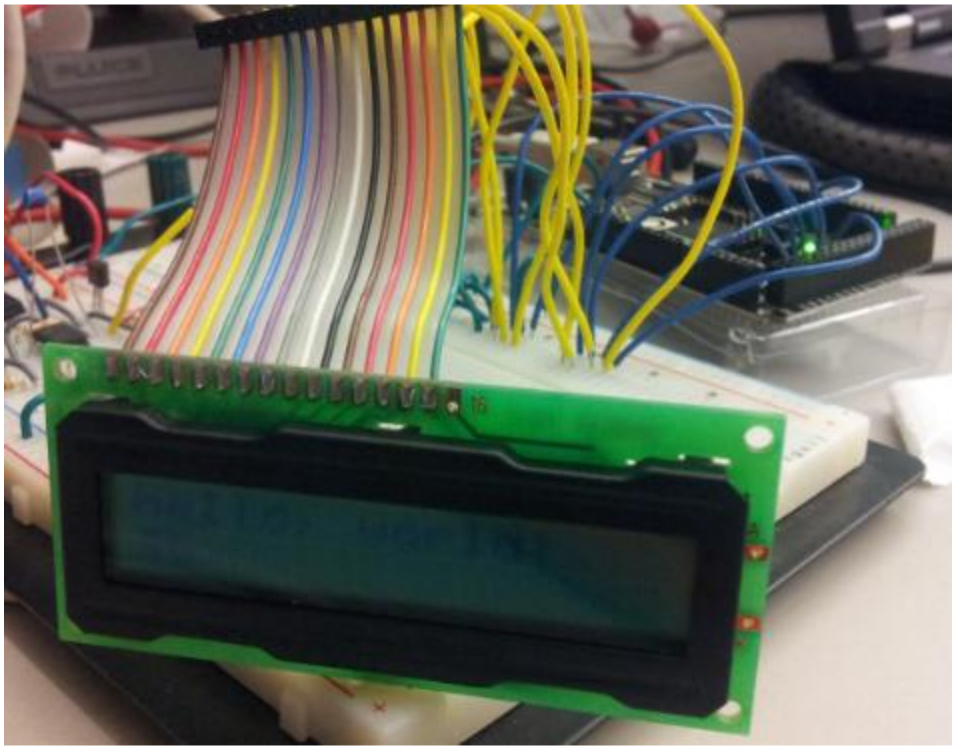
Figure 8 – LCD Display

The figure above shows an image of the LCD for the central unit. Using the Arduino, the LCD was programmed to display a typical 'Hello World' message. The model of the LCD is the 16 character by 2 line Dot Matrix LCD Module from Tianma Microelectronics. We chose this particular model because of our fairly limited needs. The LCD can only display two lines of text at once, but this is sufficient since it only needs to display one fall alert message at a time. The LCD is connected to the Arduino by digital communication, which is convenient since the Arduino has an abundance of digital pins.

3.1.2 GSM Shield

The text messaging function of the central device will be implemented using the GSM shield module shown in Figure 9 below. A GSM shield is a device which, when connected to an appropriate antenna and provided with a cellular phone SIM card, is able to carry of cellular phone-type communications including text messaging. The GSM shield we are using is the SM5100B-D model from Spreadtrum Communications. We chose this GSM shield because it is specifically designed and configured to implement cellular communications in Arduino-based projects. This makes is a perfect match for our Arduino Mega 2560 board.

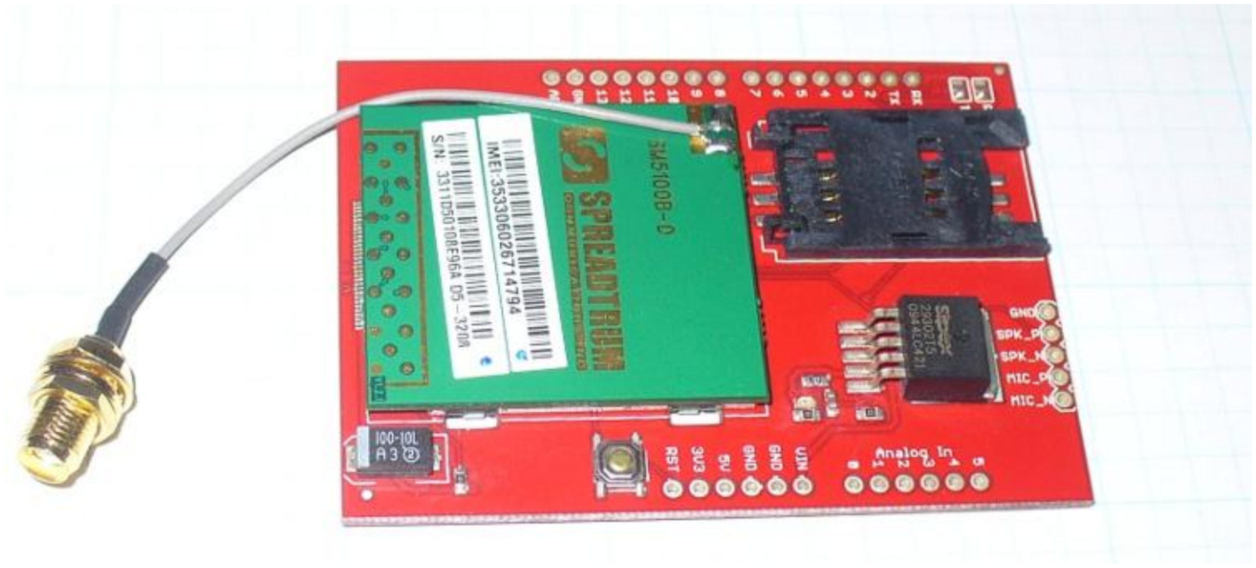

Figure 9 - GSM Shield Module

Connection	60 pins
Power Supply	3.3V-4.2V
Frequency Bands	EGSM 900 + GSM 850 + DCS 1800 + PCS 1900
Current Required	2 Amps
Supported SIM card	3V/1.8V SIM card (Auto recognized)

Table 2 – GSM Shield Specifications

As shown in Figure 10 below, the GSM shield conveniently mounts directly on top of the Arduino board and communicates with the Arduino only two serial port wires. Pin 2 and 3 from GSM shield are connected to the Pin 18 and 19 on the Arduino board for transmitting and receiving. Some important specifications [4] of the shield are listed in Table 2 above including the communication frequencies at which it can operate, the type of SIM cards it accepts, and the power requirements of the board.

We are powering the shield by connecting a 5V DC from an external battery to supply very high currents. This is because the GSM shield needs a minimum of 2 Amperes to function. Thus, we could not power the GSM shield from the Arduino board because the Arduino board can only supply a maximum of 1 A. Instead, we use the external power supply to power the GSM shield and then power the Arduino from the GSM shield. We also purchased a Dock 2.4 GHz SMA Antenna in order to receive and transmit cell phone-type calls or texts. The GSM shield does not come with its own antenna.

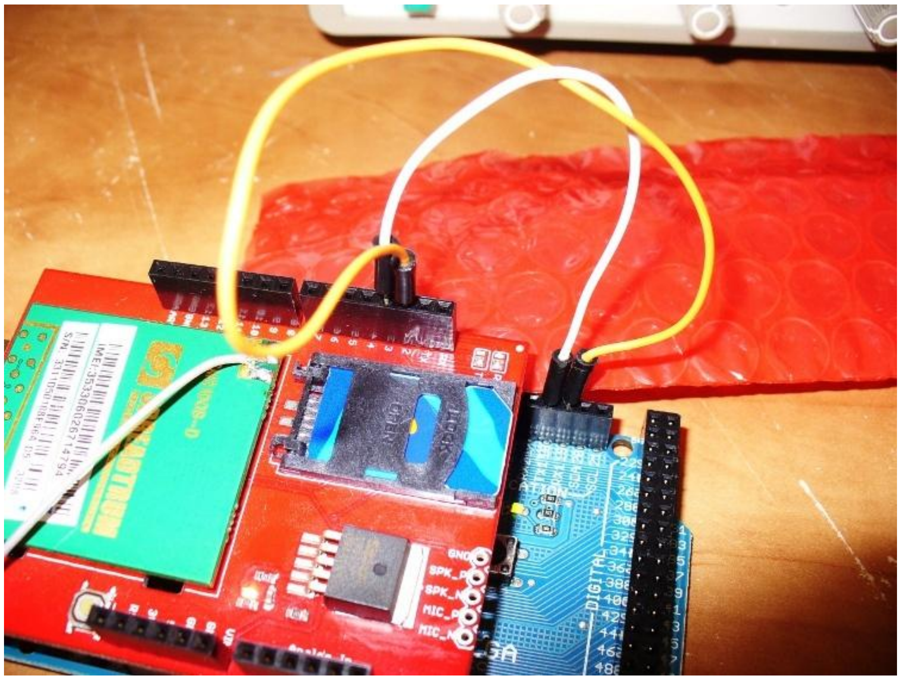

Figure 10 – GSM Shield Integrated with Arduino Board

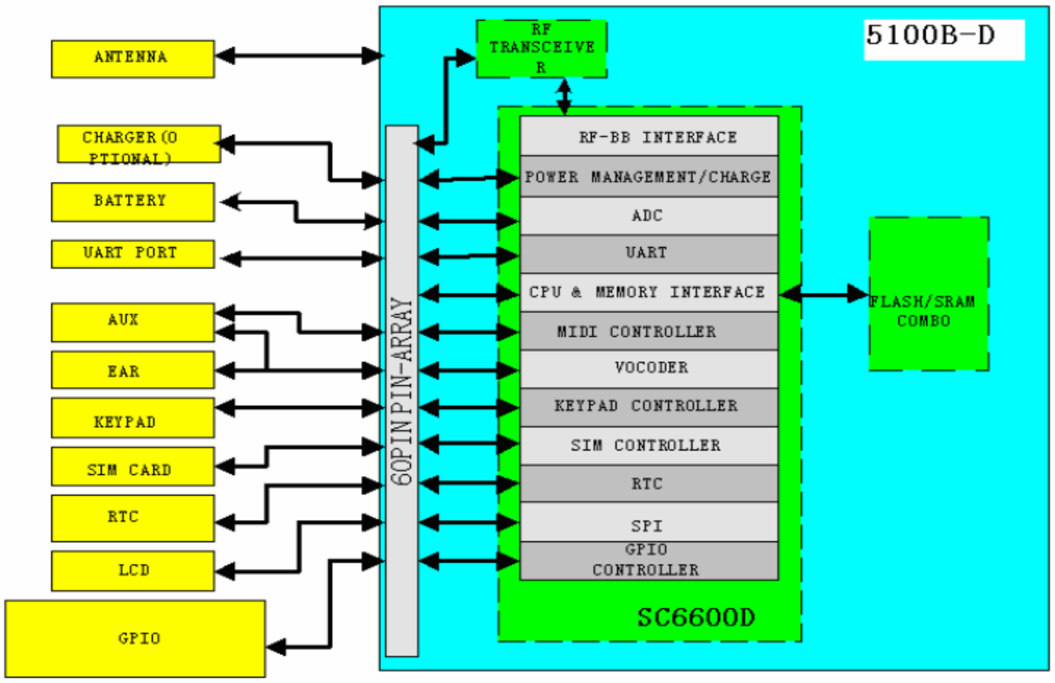

Figure 11 – GSM Shield High-Level Block Diagram

Figure 11 above and Figure 12 below illustrate the high level and low level layouts of the GSM shield. These figures illustrate the connections between board components and the pin layout. They also provide a partial illustration of the method of function of the device, which is beyond the scope of this document.

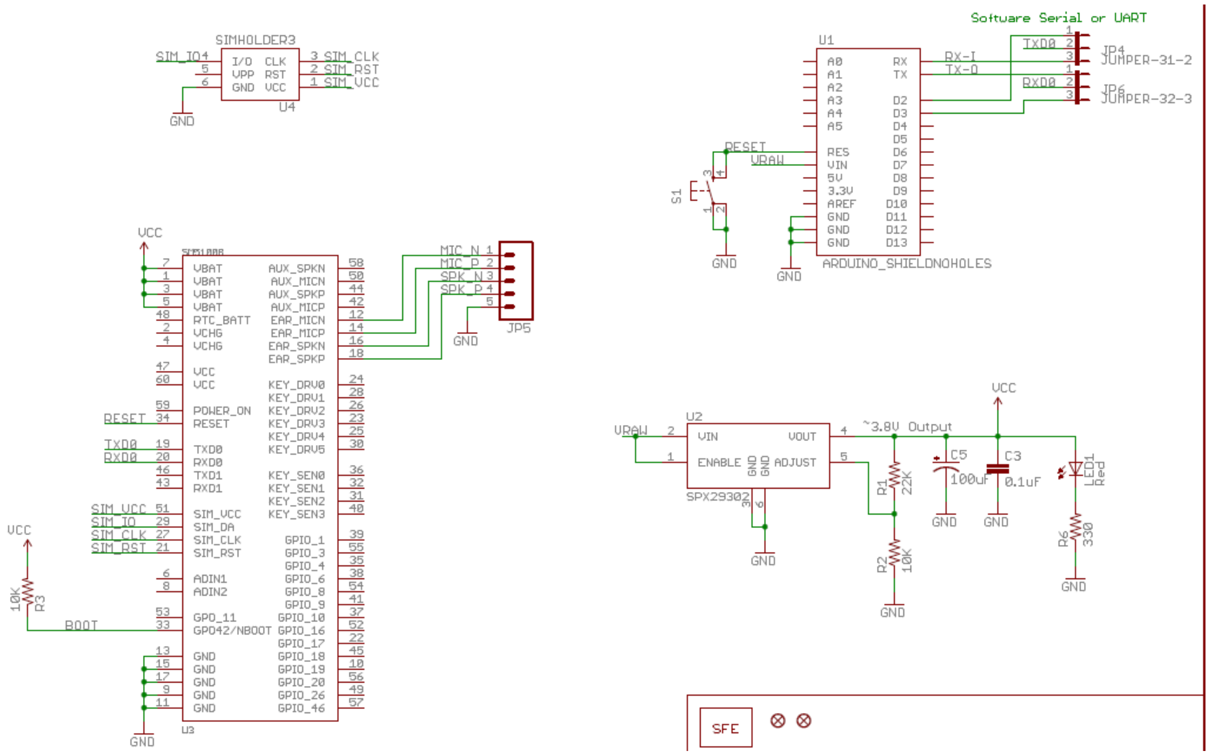

Figure 12 – GSM Shield Low-Level Pin Layout

3.2 *Central Unit Software Design*

3.2.1 Arduino Mega 2560

The Arduino board has two different types of software we use to control and monitor what is happening with it. The first is the open-source Arduino environment where we can use the provided Arduino libraries as well as C-style code to program the functions we need. The libraries for the Arduino can be found on the device website, along with some illustrative tutorials. This provides a good support base and reduces the time needed to learn how to use this technology. Figure 13 below shows a screen shot of the Arduino development environment. The second piece of software we use is ‘terminal.exe’. This terminal program provides simple commands for serial communication with the GSM shield. Using these AT commands, we can more easily implement the necessary communication between the two boards.

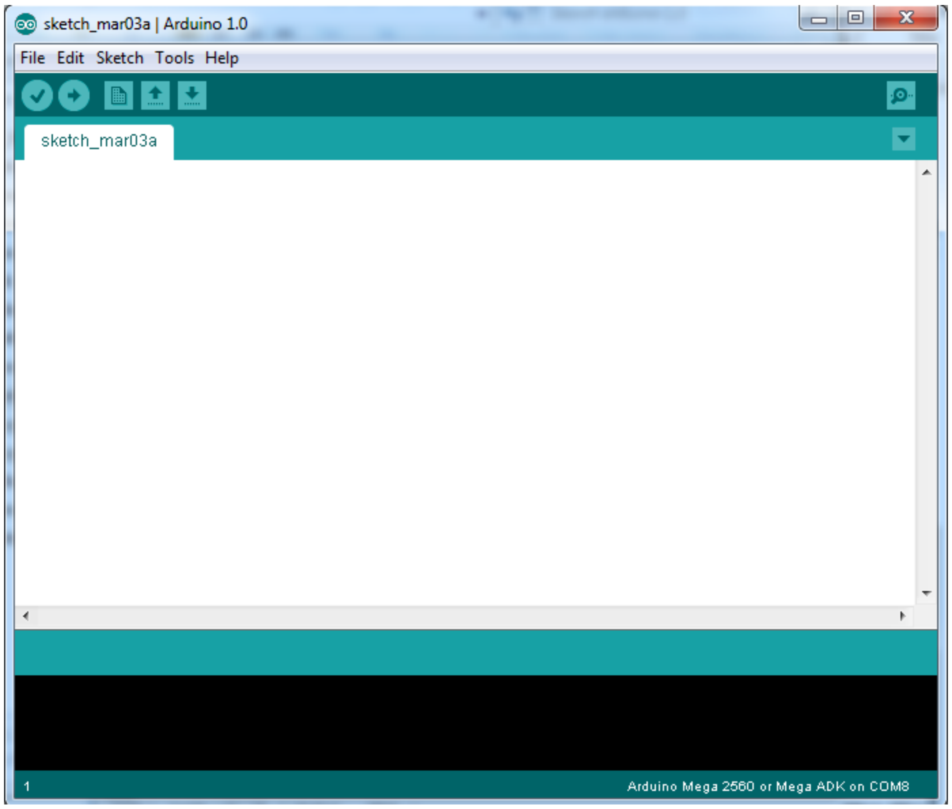
sketch_mar03a | Arduino 1.0
File Edit Sketch Tools Help
sketch_mar03a
1
Arduino Mega 2560 or Mega ADK on COM8

Figure 13 – Arduino Software Development Environment

Programming of the Arduino is done through the software serial port. We connect the Arduino to a computer via USB port and compile our program onto the board. Once a program is uploaded to the board, the board continues to run that over and over again until we decide to change it. The board can be reset to start from the beginning with the reset button on the board is pressed.

3.2.2 LCD Text Display

Once the hardware is set up and the LCD pins are connected correctly, programming what the LCD prints is very simple to do with the Arduino board. The Arduino board has a conveniently provided library for controlling LCD displays, and our LCD model is compatible with that library. The LCD automatically converts received data into properly formatted letters before displaying.

3.2.3 GSM Shield

As previously noted, we program and communicate with the GSM shield via the Arduino Mega board. There is a convenient serial communication program to facilitate that communication. We only need to know the correct AT commands in order to control the device, and these commands are conveniently available in [5].

4. Portable Accelerometer Unit

This section is concerned with the design of the portable device which will be used for fall detection and alert transmission. The device is responsible for monitoring the movements of the user using an accelerometer. The collected accelerometer data can then be processed and analyzed in real time to determine if and when a fall has occurred. When a fall is detected, the device will send a wireless alert to the central unit which can then notify a family member or nursing staff.

This device thus requires several hardware and software components. From a hardware perspective, the portable unit must contain an accelerometer, a wireless transceiver, and a microcontroller for processing accelerometer data and overseeing transceiver communications. The software requirements are twofold. First, the on-board microcontroller must be programmed to execute and oversee all operations necessary for fall detection and alert transmission. Second, in order to develop an accurate algorithm for fall detection, accelerometer data must be collected, analyzed, and tested on a computer. This requires cooperation between microcontroller code, a computer serial terminal client, and a data visualization program such as MATLAB.

There are also several secondary device features which require hardware and software support. A push button is required to enable user input to the device, for the purpose of cancelling a false alarm or triggering a non-fall related alarm (such as a heart attack alert). Also, the device must have an audio buzzer to provide feedback on its state of operations. The user must be informed that an alert is about to be sent and assured that someone has received the message and is on the way.

It is important to acknowledge that there are multiple possible ways to satisfy each of these hardware and software requirements. There are many combinations of components which could be used to implement this device. When explaining our decision to use a particular component, we will refer to a several key factors which affect all project development decisions. These factors include financial cost, development time constraints, ease of implementation, and factors like size, durability, and power consumption which relate specifically to the demands of a portable battery-powered device.

The following sections outline the hardware and software design decisions involved in our portable device in a much more detailed manner. Due to the method of implementation of wireless communication between portable device and central unit, it will be intuitive to consider the transceiver on the central unit as part of the portable device section. The implementation and programming of this transceiver will thus also be covered below.

4.1 *Portable Device Hardware Design*

4.1.1 eZ430-RF2500 Development Kit

Our portable device requires both a microcontroller and an RF transceiver. While we could purchase these components separately, the eZ430-RF2500 development kit from Texas Instruments is a commercially available solution which already contains both of these things. In addition, it is suitable for the portable device because it is extremely small and requires very low power. By purchasing two of the boards, we can also place one in the central unit and greatly simplify communication between the two devices.

The hardware components on the eZ430-RF2500 development tool are the MSP430F2274 microcontroller (MCU) and CC2500 2.4 GHz wireless transceiver. The development kit also includes the following features [6]:

- USB connector chip for programming and debugging
- 18 accessible development pins on each board
- Ultra low power MSP430 MCU with 16 MHz performance
- Two general purpose digital I/O pins connected to LEDs
- On-board interrupt-friendly push button

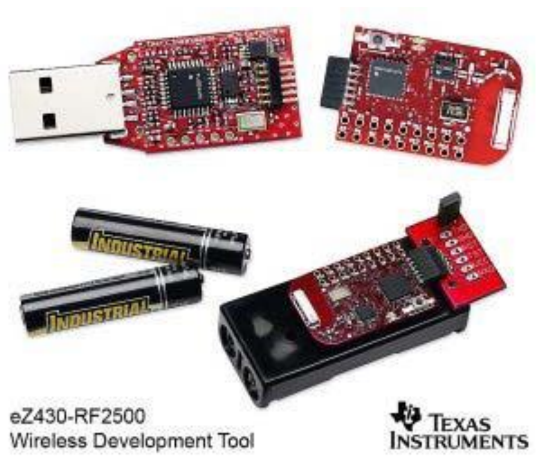

Figure 14 – Picture of ez430-RF2500 Development Kit

The ez430-RF2500 is an efficiently integrated tool ideal for sending data wirelessly from host to client, and vice versa. The available development pins, with low power requirements, allow us to connect any small device to our development kit (in our case, an accelerometer). Figure 14 above shows the ez430-RF2500 development kit and its size relative to AAA batteries. Figure 15 below shows the microelectronic schematic layout of one of the development boards.

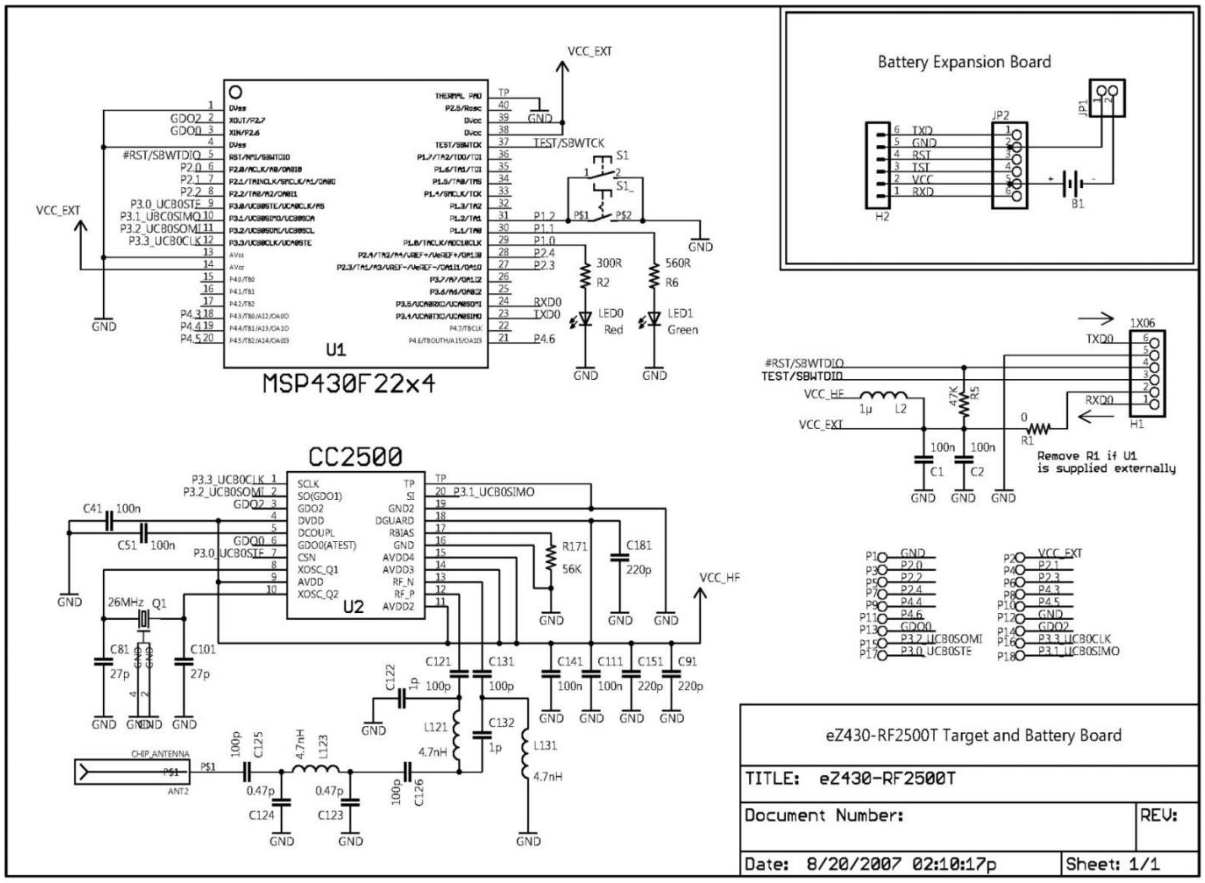
Battery Expansion Board
MSP430F22x4
U1
CC2500
U2
eZ430-RF2500T Target and Battery Board
TITLE: eZ430-RF2500T
Document Number:
REV:
Date: 8/20/2007 02:10:17p
Sheet: 1/1
Remove R1 if U1 is supplied externally

Figure 15 – Hardware Schematic for MCU Board

4.1.1.1 MSP430F2274 Microcontroller

The MSP430F2274 microcontroller is a control unit that provides a 16-bit RISC CPU, 16 bit registers, and constant generators. The MCU includes 32 KB plus 256 Bytes of flash memory, as well as 1 KB of RAM. Other features include:

- Low supply voltage range: 1.8 V to 3.6 V

- Low power consumption: Active mode of 270µA, standby mode of 0.7 µA
- Fast wake-up from standby
- Clock module configurations
- Universal serial communication interface with enhanced UART
- 10-bit analog-to-digital converter (ADC)

4.1.1.2 CC2500 Transceiver

The CC2500 is a 2.4 GHz transceiver designed for low power wireless applications. This transceiver is implemented for use for the 2400-2483.5 MHz ISM (Industrial, Scientific, and Medical) and SRD (Short Range Device) frequency band and can be tuned to any frequency in that range. Hardware support is provided for the following:

- Packet handling
- Data buffering
- Burst transmissions
- Link quality indication
- Wake-on-radio

Some other key features of the CC2500 transceiver include:

- High sensitivity: -104 dBm at 2.4 kBaud, 1% packet error rate
- Low power consumption: 13.3 mA in RX, 250 kBaud
- Programmable output power up to +1 dBm
- Excellent receiver selectivity and blocking performance
- 500 m achievable line of sight range

4.1.2 ADXL335 Accelerometer

The accelerometer we have chosen is the ADXL335 model, a low power analog output 3-axis accelerometer equipped with signal conditioned voltage outputs and acceleration measurement capabilities within the range of ±3 g. The accelerometer can measure both static and dynamic acceleration of gravity: static in tilt-sensing applications, dynamic in motion, shock, or vibration applications. The 3 axes are X, Y and Z. The accelerometer can detect dynamic accelerations in a very wide frequency band, from 0.5 Hz to 1600 Hz for the X and Y axes, and 0.5 Hz to 550 Hz for the Z axis.

Other key features of the ADXL335 include [7]:

- Small implementation chip: 4 mm x 4 mm x 1.45 mm
- Low power consumption: 350 µA
- Single-supply operation: 1.8 V to 3.3 V
- 10,000 g shock survival
- Excellent temperature stability

We chose this particular accelerometer because of the listed characteristics and because it assimilates well with the ez430-RF2500 development kit in terms of input voltage and size. By using the analog-to-digital converter we are able to efficiently convert our accelerometer voltage output to digital data, allowing for compatibility with the ez430-RF2500's digital I/O pins. While digital output accelerometers do exist, the presence of the ADC on our development board makes an analog accelerometer a valid choice. Figure 16 provides a schematic of the ADXL335 accelerometer. Figure 17 shows the convenient size of the ADXL335, on two different breakout board chips, with reference to a penny.

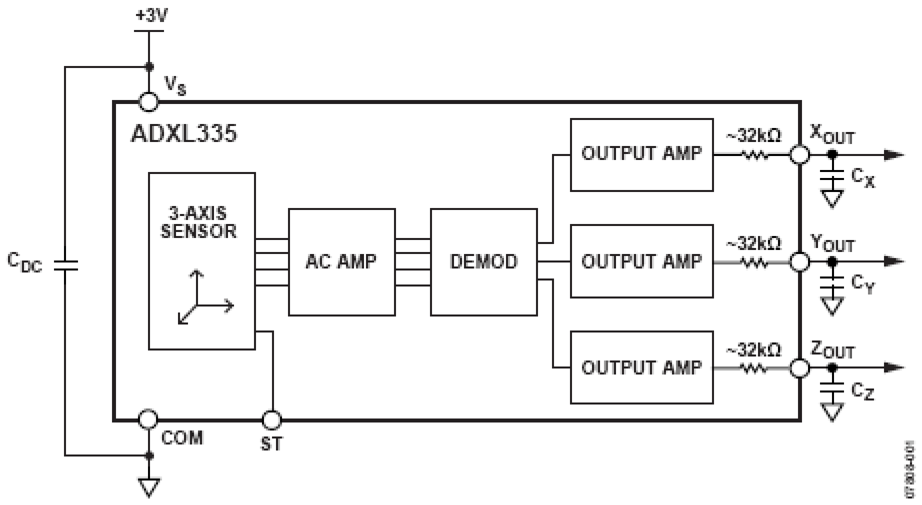

Figure 16 – ADXL335 Accelerometer Chip Schematic

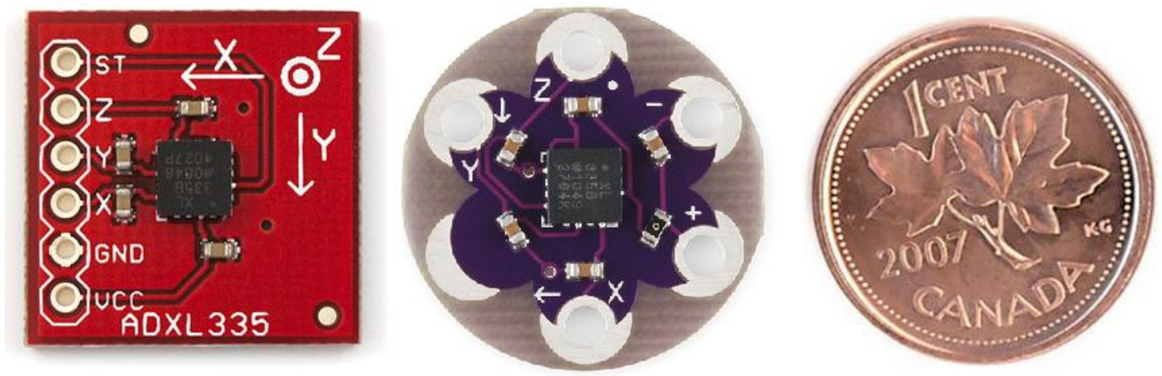

Figure 17 – Relative Size of ADXL335 Chip vs. a Penny

4.1.3 Overall Circuit Design

Figure 18 shows the implementation of the entire portable device. The ez430-RF2500 development kit and ADXL335 accelerometer are connected to a power supply via the LD1117 3.3V voltage regulator. A piezo buzzer is also included for reasons which will be described later.

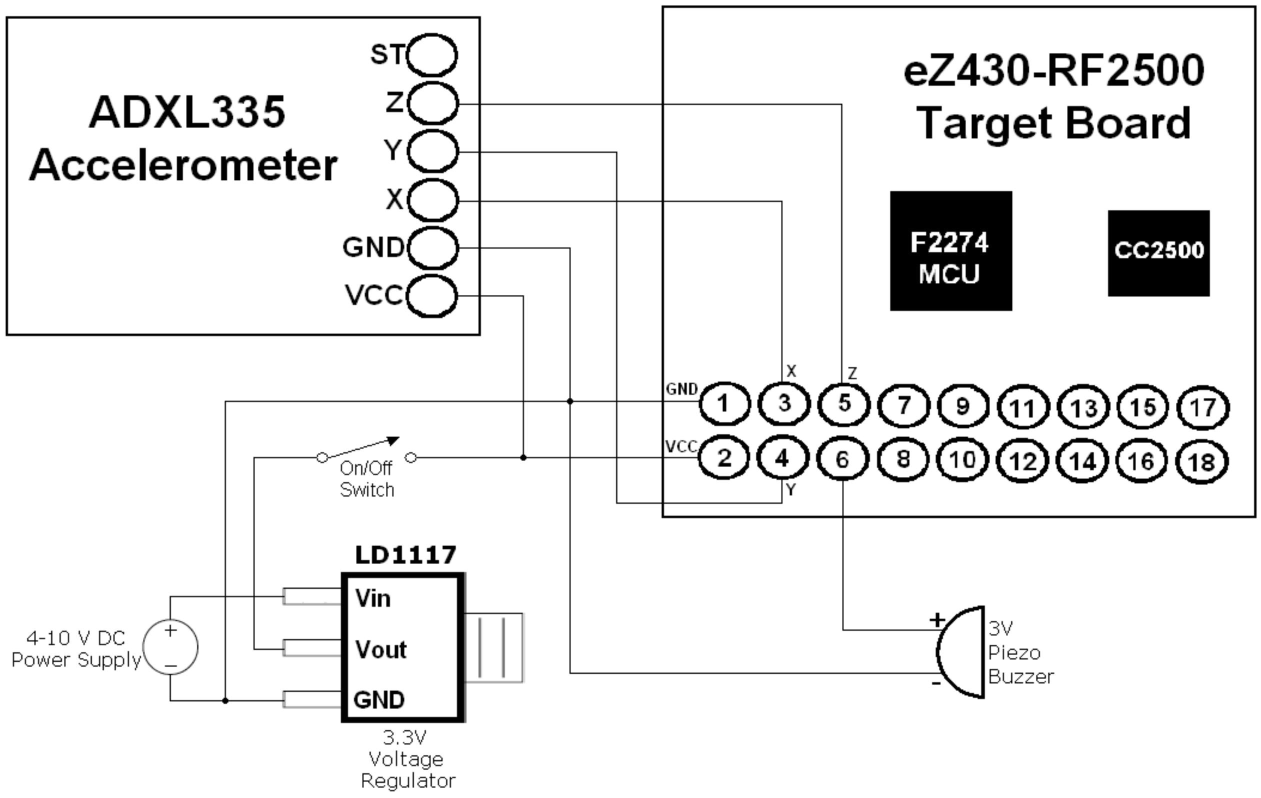

Figure 18 – Circuit Schematic of Portable Device

The portable device will run on 3.3 V with a 9V battery supplying the input voltage regulator. The device will consume ~370 uA when turned on and, assuming the user wears the device for ~15 hours per day, the portable device will last at least 1 week on one charge. This is realistic because the device is not worn when sleeping or out of the house. Transmission range for the link between portable and central devices is approximately 40-50 m. Figure 19 shows the hardware connection block diagram of the portable device and central transceiver.

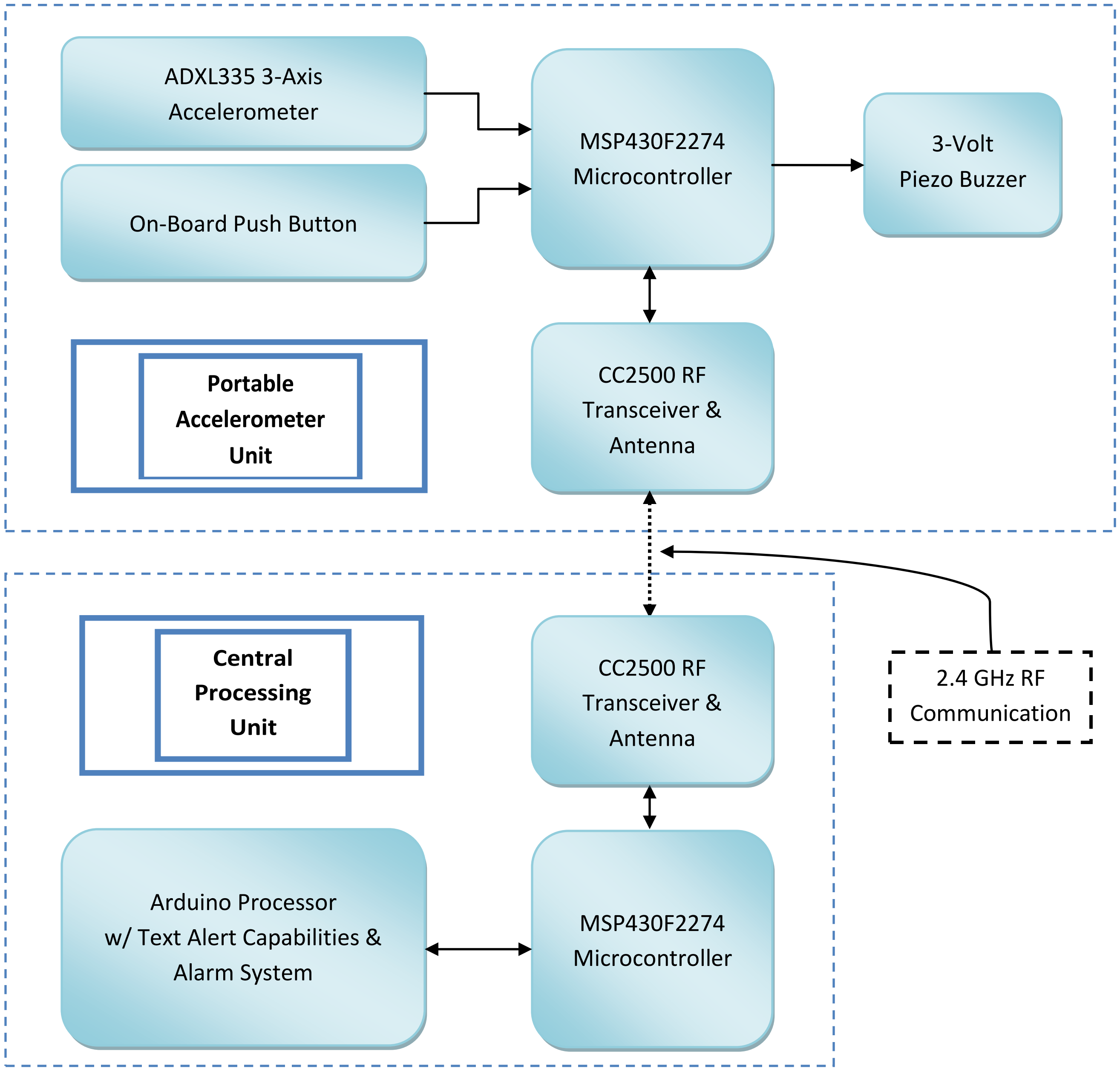
ADXL335 3-Axis Accelerometer
On-Board Push Button
MSP430F2274 Microcontroller
3-Volt Piezo Buzzer
Portable Accelerometer Unit
CC2500 RF Transceiver & Antenna
2.4 GHz RF Communication
Central Processing Unit
CC2500 RF Transceiver & Antenna
Arduino Processor w/ Text Alert Capabilities & Alarm System
MSP430F2274 Microcontroller

Figure 19 – High Level Block Diagram of Accelerometer & RF Communications

The size of the portable device will be in the range of 2cm by 2cm by 5cm, a reasonably small size. In future commercial versions of the device, it is likely that the MCU, transceiver, accelerometer, and voltage regulator will all be implemented on one chip. Additionally, a DC power supply smaller than a 9V battery will be used. It is thus reasonable to expect that device dimensions will decrease even further as the product is refined.

4.2 *Portable Device Software Design*

Having chosen hardware components and developed a hardware design plan, the next step is to design the software that will control and implement the various functions of the device. For the portable device, all software requirements relate to programming the MSP430 microcontroller provided by the eZ430-RF2500 development kit. The process of software design can be thus divided into three stages. First, choose a development environment which supports our microcontroller. Second, identify the major tasks which the MSP430 must perform and the main commands necessary to execute those tasks. Finally, create a software flowchart diagram to plan and map all required functions. These three stages are discussed below.

4.2.1 Development Environments

4.2.1.1 IAR Embedded Workbench

There are two main options for software development with MSP430 MCU devices. The first option is Code Composer Studio™, a development environment created by Texas Instruments specifically to support its brand of microcontrollers. The second option is IAR Embedded Workbench, a compiler and debugger suite which supports a wide variety of microcontrollers. Both of these options are suitable for our purposes and have free versions available online. We downloaded and investigated both programs before settling on IAR Embedded Workbench. The IAR program layout is more intuitive and familiar. Also, because it supports a greater array of MCUs, there are more online example and help resources available. We will write our code in C language and use IAR to compile to the device.

4.2.1.2 MATLAB

As noted, we will require some type of data visualization software during the fall algorithm development and testing phase of the project. MATLAB is perhaps the most widely available, versatile, and powerful tool for data analysis and visualization [8]. The applications of MATLAB within this project are discussed in detail in the Fall Algorithm Development section below.

4.2.2 MSP430 Tasks and Commands

The MSP430 MCU must oversee a variety of tasks during device operation. The first task involves reading analog data from the accelerometer input pins. The eZ430-RF2500 target board has a number of analog input pins and a single 10-bit analog-to-digital converter (ADC) to interface between the pins and the microcontroller. In order to acquire data, the MSP430 must configure several registers which control the state of the ADC. The various bits of these registers control things such as the digitization reference voltages and the input pin to measure from. Certain bits are also responsible for enabling conversion, starting conversion, and calling the interrupt handler when conversion has ended. Two sample commands are as follows [9]:

```
ADC10CTL1 = INCH_0;            // select input pin A0
ADC10CTL0 |= ENC + ADC10SC;    // enable and start conversion
int value = ADC10MEM;          //save the digital data into variable ‘value’
```

Another major task for the MSP430 is coordinating RF message transmission and reception. Associated libraries provide simple high-level functions for these purposes. The commands MRFI_Init() and MRFI_WakeUp() provide single line implementation of RF communication initialization and enabling. To transmit data, simply declare a packet variable, configure the message as an array of bytes, and then pass the packet to MRFI_Transmit(). Receiving data is equally simple. When an incoming message is detected, an interrupt is automatically generated. In the interrupt handler function, passing an empty packet into MRFI_Receive() will provide quick access to the data. The reason for the ease of communication is the structure of the development board. The on-board CC2500 transceiver controller greatly simplifies the level of detail which the MSP430 must provide. Additionally, the CC2500 automatically ensures data validity by transmitting a check value with each packet of data.

Since our portable device will use the on-board push button as a user input, the MSP430 must also configure the button correctly. This requires some minor initialization commands similar to those used for the ADC. When the button is pushed, an interrupt will be generated to handle the event, just as an interrupt is generated when a packet is received. A final important task for the microcontroller is data processing and testing to detect a fall. This is programmed using typical C code commands and relies on the computational power of the chip.

4.2.3 Detection Program

Knowing the software commands needed to control the portable device, the only task left is to plan the flow of the control program. Creating the C code is then relatively trivial. The two flowcharts below illustrate the behavior of the MSP430 program for the portable device chip and the central unit chip, respectively.

4.2.3.1 Portable Device Program Flow

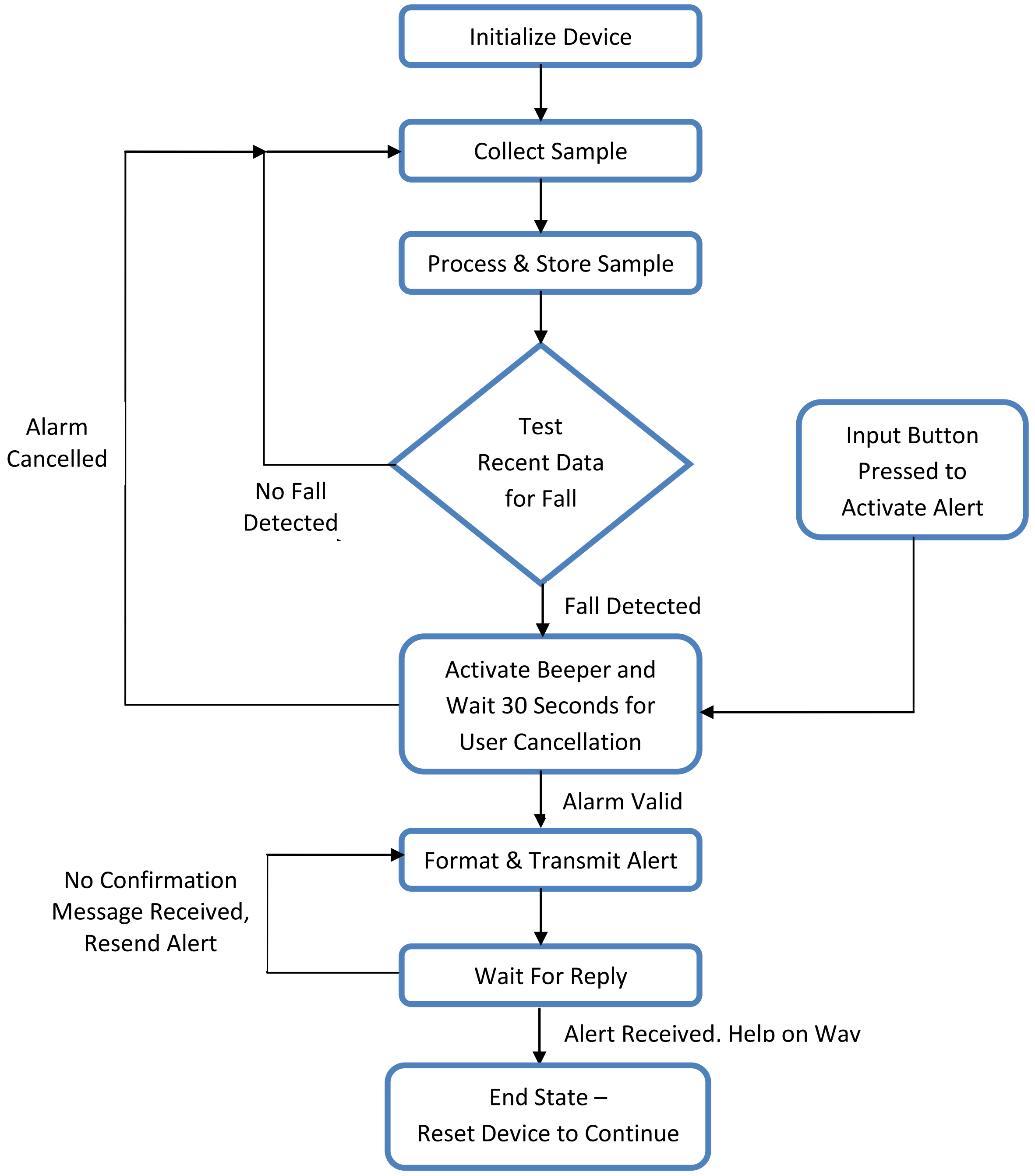
Initialize Device
Collect Sample
Process & Store Sample
Test Recent Data for Fall
No Fall Detected
Alarm Cancelled
Input Button Pressed to Activate Alert
Fall Detected
Activate Beeper and Wait 30 Seconds for User Cancellation
Alarm Valid
Format & Transmit Alert
No Confirmation Message Received, Resend Alert
Wait For Reply
Alert Received, Help on Way
End State – Reset Device to Continue

Figure 20 – Portable Device Software Flowchart

After some initializations, the portable device enters a loop of reading, storing, and processing accelerometer data and then testing the data for a fall. The loop continues until a fall is detected or the user presses the push button. At that point, the beeper is activated with sonic pulses at ~1 Hz to inform to user that an alert has been triggered. The user is given 30 seconds to press the button to cancel the alert and return to normal operation. Otherwise, the alert message will be formatted and sent. The device will then wait for an acknowledgement message and enter an end state. The user can press the button again or toggle the on/off switch to reset the device.

4.2.3.2 Central Unit Program Flow

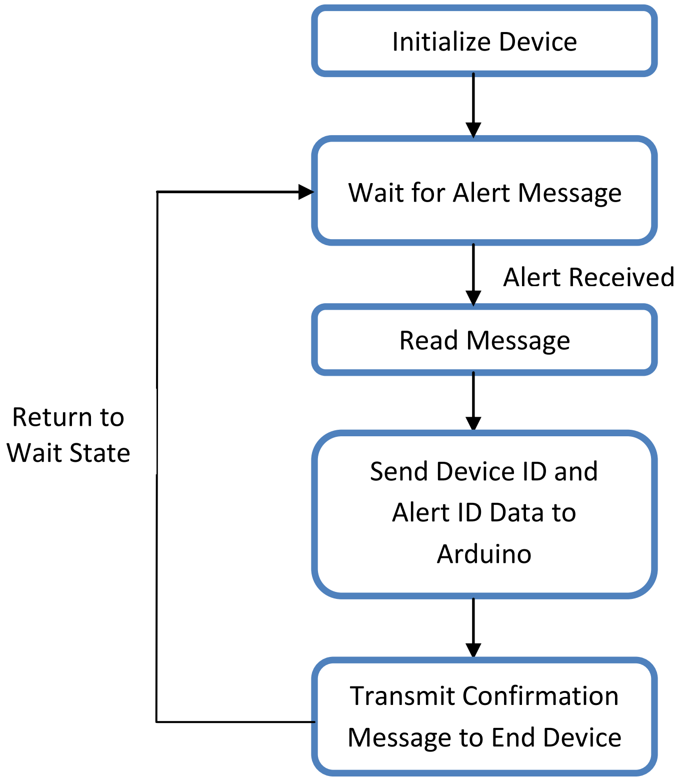
Initialize Device
Wait for Alert Message
Alert Received
Read Message
Send Device ID and Alert ID Data to Arduino
Transmit Confirmation Message to End Device
Return to Wait State

Figure 21 – Central MSP430 Software Flowchart

It should be noted that the above flowchart represents the program flow for the central device MSP430 and RF transceiver only. It does not deal with the central Arduino board, as this topic was covered in a previous section. After some initializations and setup, the MSP430 enters an idle wait state. Whenever an alert message is received, the microcontroller will enter an interrupt handler routine. The MSP430 will then pass the message to the Arduino, send an acknowledgement to the portable device, and return to its idle state.

4.3 *Fall Algorithm Development*

Obviously, the success of this device is dependent on the development of a reliable algorithm for fall detection. Good hardware and software design are irrelevant if the device cannot perform its intended task with a high degree of accuracy. To that end, there is a necessary series of procedures which we must perform in order to develop this algorithm. The first task is to collect accelerometer data of various fall situations for future analysis and testing. This requires software code for the pair of MSP430 chips. Secondly, the data must be presented in graphical form to enable visual analysis. Finally, based on the collected data sets, the acceleration properties of a fall must be carefully defined and threshold values must be determined. These three steps are outlined below.

4.3.1 Data Collection

Two flowcharts below represent the software code for data collection which correspond to the portable and central units. These programs are currently functional and have already been used to collect some accelerometer data. The flowcharts are much simpler than those above. The portable device continuously collects acceleration samples and transmits them as data packets. The central unit receives the packets and writes them, via serial interface, as text into a terminal server. In this case, we make use of the USB dongle provided by the eZ430 development kit in order to connect the central MSP430 microcontroller to a computer serial port.

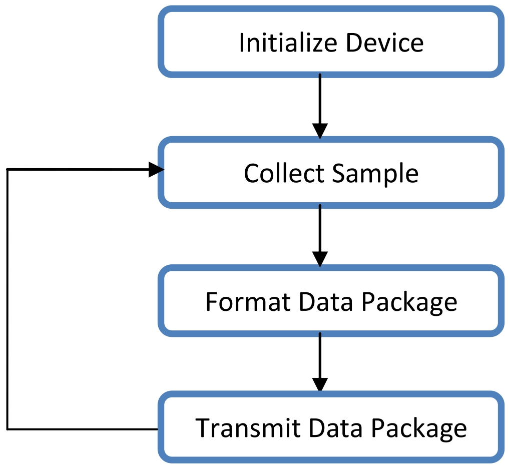

Figure 22 – Data Collection Software Flowchart for the Accelerometer End Device

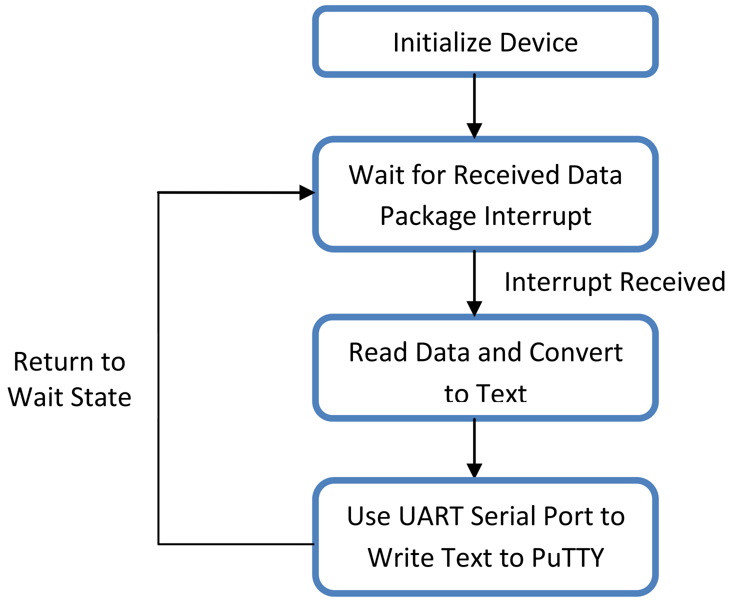
Initialize Device
Wait for Received Data Package Interrupt
Interrupt Received
Read Data and Convert to Text
Use UART Serial Port to Write Text to PuTTY
Return to Wait State

Figure 23 – Data Collection Software Flowchart for USB-Connected Receiving End

Since the computer is receiving data via serial port, we require some type of terminal server program to collect the data and display it as text. For this purpose, we use PuTTY which allows us to select a serial port and then simultaneously display the received data and write into a text file.

4.3.2 Data Visualization

As stated previously, we will use MATLAB in order to plot the acceleration data visually. This will make data analysis much more straightforward. There is also some processing which MATLAB must perform on the data in order to produce meaningful and intuitive plots. The accelerometer data which the computer receives and which is passed into MATLAB is digital. Due to the analog-to-digital conversion on the MSP430, each acceleration value is represented as an integer between 0 and 1023. The full 6g range (+3g to -3g) of the accelerometer is spread over this set of numbers. Zero acceleration is represented by 512. Rather than display the data on this somewhat arbitrary scale, it is preferable to display the data in terms of g. Simple arithmetic operations can convert each data value to a more intuitive number.

4.3.3 Data Analysis

The figure below shows an acceleration dataset which was collected using PuTTY and plotted using MATLAB. As discussed, MATLAB has converted the vertical axis scale of the data into g, while the horizontal axis represents time. The three axes of data correspond to the three axes of the accelerometer. The dataset represents the wearer of the portable device walking around and then falling down and staying in a prone position for a brief period.

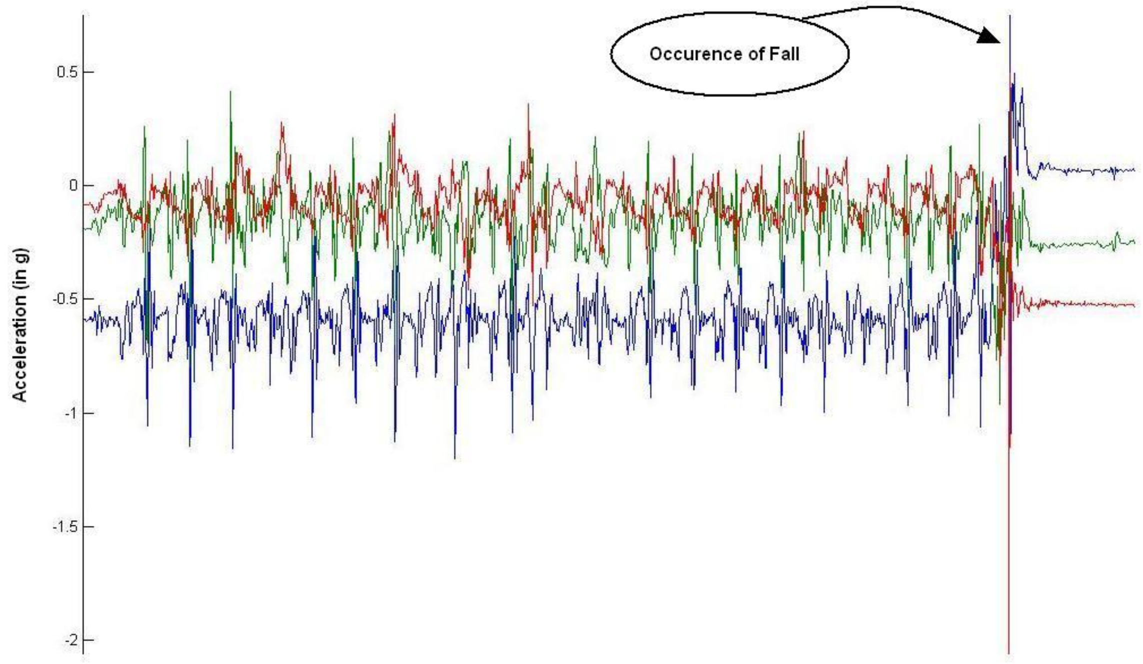

Figure 24 - MATLAB Plots of Accelerometer Data Showing a Fall

In the above dataset, there are several characteristics of interest which need to be recognized and will likely play a role in the final fall detection algorithm. We see that initially the blue acceleration axis is centered near -1g, while the other two axes are centered near zero. This is because the blue axis points downwards while the wearer is standing. After the fall, the blue axis measures approximately 0g and the red axis has taken on a value near -1g. This series of events indicates a change in orientation of the device (and individual) and is one sign that a fall has occurred.

During the period when the person is walking, all three data axes appear spiked and noisy. The spikes generally correspond to the steps and turns of the individual's waling pattern. It would be preferably if the spikes were of lesser magnitude and the noise were less noticeable. However, this dataset was collected without a proper case for the device. With a proper case, some of the vibrations and noise would be damped down. Also, some low pass filtering and temporal averaging of the data could be used to eliminate noise. In any case, the magnitude of the data spike produced by the fall is much greater than at any other point. Thus, an abnormally larger spike is another feature by which to identify a fall.

To develop a reliable fall detection algorithm, we will require a large array of datasets similar to the one above. With a larger bank of data, we will be able to identify more robust thresholds and fall characteristics to test against. We will also need to plan against potential cases which could produce a false alarm. Testing our fall algorithm will be one of the greatest challenges of this project and will require a significant amount of time and effort. Some discussion of possible test cases and procedures can be found in the system test plan section below.

5. System Test Plan

There are many things to consider in constructing a test plan to ensure proper functioning of the F.A.M. system. Since the system encompasses multiple devices, the portable and central units must first be tested individually and then together as a system. Each part has to be tested thoroughly to make sure that everything works well together.

For the portable device that sits on the user's belt, the first test is to make sure that the device can connect to the central device in many situations. We will be testing to see how far the device can speak to the central device through direct sight. Once we establish that, we will start using it in different areas where walls and different object obscure the path. We will see which situations need to be addressed and take note to let users know when installing the device. Next, the device needs to be able to withstand a human fall. Both devices also need to be tested in a variety of different environments to ensure that changes in temperature or humidity do not affect performance. Another series of tests will be required when pressing the cancellation button in case the user unintentionally activates the device.

Secondly, we need to run some tests on the central device. This device will receive signals sent from the portable device. As it's connected wirelessly to the portable unit, awaiting a fall detect, a test case will involve picking up a signal from the portable unit. This signal doesn't necessarily have to be produced from a fall, but can be a "fake" signal that F.A.M. developers will force. The test scenario described above will signify connection between the central box and portable unit.

Our final phase will involve testing the sufficiency of the power supply. A test case program will be hard coded to test all components working simultaneously. We can monitor the voltage provided by the accelerometer (through software) and will therefore be able to effectively inspect our system during power tests. Lastly, we will monitor the heat distribution of our device, making sure the temperature stays within safe thresholds.

5.1 *Typical Usage Scenario*

The following steps describe the typical usage scenario of the F.A.M. device:

1. User attaches portable unit to belt
2. User adjusts portable unit to a comfortable position along waist line, then powers on the device by on/off switch
3. The central device is plugged into the wall and establishes contact with the portable unit
4. Central device now awaits a fall detection signal from the portable unit.

5.2 *Test Scenarios*

1. Test to determine the maximum distance at which the portable device and the central box communicate.

2. Test to see what factors limit communication with the central device.

3. Test various scenarios of falls to make sure the device works properly at least 95% of the time. Scenarios to be tested include:

- Falling forwards, sideways, and backwards from a standstill.
- Falling forwards, sideways, and backwards while walking.
- Sitting down very quickly (false alarm).
- Falling on top of the device (ensure structural soundness).
- Dropping the device (structural soundness and false alarm).

4. Test to see if we can cancel the sending of a transmitted message signal.

5. Make sure that the Arduino board sends text messages to the correct number.

6. Check to see the reception of the GSM shield with the network in various locations.

7. Measure the time it takes for a normal text message to be received.

6. Conclusion

The design specifications explained in this document explain how we plan to implement the system, safety, and performance requirements which our F.A.M. device must meet. Hardware specifications, software specifications and test plans are all outlined in this document. The expected completion date for the proof-of-concept model of F.A.M. is April 5th 2012. On this date, all primary device requirements will be met and the system will be fully functional for demonstration purposes.

7. References

[1] Centers for Disease Control and Prevention, "Falls Among Older Adults: An Overview". Internet: http://www.cdc.gov/homeandrecreationalsafety/falls/adultfalls.html [Jan 2012].

[2] F.A.M. Inc., "Functional Specification for a Fall Detection System for Seniors", Simon Fraser University, Burnaby, BC, Canada, February 2005.

[3] Arduino, "Arduino Mega 2560 Overview". Internet: http://arduino.cc/en/Main/ArduinoBoardMega2560 [Feb 2012].

[4]Sparkfun, "SM5100B-D Datasheet". Internet: http://www.sparkfun.com/datasheets/CellularShield/SM5100B%20Datasheet.pdf [Feb 2012].

[5]Sparkfun, "SM5100B-D AT Command". Internet: http://www.sparkfun.com/datasheets/CellularShield/SM5100B%20AT%20Command%20Set.pdf [Feb 2012].

[6] Texas Instruments, "MSP430 Wireless Development Tool". Internet: http://www.ti.com/tool/ez430-rf2500 [Feb 2012].

[7] Analog Devices, "ADXL335L Small, Low Power, 3-Axis ±3g Accelerometer". Internet: http://www.analog.com/en/mems-sensors/mems-inertial-sensors/adxl335/products/product.html [Feb 2012].

[8] MathWorks Inc., "MATLAB". Internet: http://www.mathworks.com/products/matlab/ [Mar 2012].

[9] Texas Instruments, "MSP430x2xx User's Guide". Internet: http://www.ti.com/lit/ug/slau144i/slau144i.pdf [Feb 2012].