CS25710 Assignment One 2010-2011

Peter Maynard
Aberystwyth University,
pem9@aber.ac.uk

May 4, 2011

Contents

1 Components

1.1 Microcontroller

The microcontroller is required to communicate with the other devices such as the SD card, accelerometer and real-time clock. It will be used to gather the data from the sensor, in this case the accelerometer, and store it on the SD card along with the time from the real-time clock. All of these devices are required to the use Serial Peripheral Interface (SPI) bus. It is recommended to use a PIC24H microcontroller due to the many features it can provide. It is recommended to use the PIC24HJ12GP201 due to the fact that is small and powerful. These are, but not limited to, 16Bit architecture, 40 MIPS operation, 12KB of program memory, SPI support and low power consumption. This means that the device will have more than enough speed to complete the specification requirements. There is also plenty of room for the program code. The device will also be able to communicate with the other devices due to its SPI and I2C bus connections. It is recommended to use the PIC24 instead of the PIC16 and PIC18 as the PIC24 is easier to use and more powerful for the same price.

An alternative to the PIC24H, could be a microcontroller that is base on the 8051 chip. This will proved a SPI bus, along with enough processing power. Though these devices are older and will not be as fast at the PIC24H, they would still complete the requirements with out much trouble.

1.2 Storage

The data that will be stored is the time of the recording in this format. Apr 18 21:56:18. This takes up 16 bytes in its self. The device will also store the X, Y and Z results from the accelerometer, this is outputted in 16bit twos complement, which is 2 bytes. So for each data recording there will be 18 bytes of storage space required. If the storage device was capable of storing 1GB, it would be possible to store 59,652,323 records. If the device was to take a record every second for a whole day it would take up 1.48Mb to space.

For storage it is recommended to use an SD or microSD storage device. Due to the storage requirements it is not possible to store the recorded data on the microcontroller. It is required to use an SD card socket and SD card that supports the use of the SPI bus. This is to make sure that all the components will interact with each other, with out too much a problem. Though forcing the SD card to communicate in SPI mode reduces the performance, relatively to the SD mode which enables the wide bus option.

A valid alternative would be a to use a separate Electrically Erasable Programmable Read-Only Memory (EEPROM) chip, to store the data. The AT45 DB161D is a good device that can support up to 16Mbit of storage, which would be plenty of space to store at lease a whole days results. The device would all so use less power and last for longer compared to an SD card. It also supports SPI bus. The only down side to using a EEPROM chip is that it is not possible to simply remove the unit and insert into a standard computer. You would have to

take the whole unit and download the contents using software that is designed for it.

1.3 Real-Time Clock

To make the data that is recorded as accurate as possible it is required to have a Real Time clock built into the unit. This clock will keep track of the time, even when the unit has enabled sleep mode. This device will be called when the unit records a reading from the sensors to the storage device. The DS3234 clock is a good option due to the device supporting the SPI bus. This means that the device will be able to connect up with the the devices with out much hassle. It counts in real time the seconds, minutes, hours, date, month and year (Including leap year). The device is valid until 2099. An alternative clock could be the DS1307, if for some reason the unit required the use of a serial device instead of SPI. Aside from the the way they communicate they are both of the devices are the same.

1.4 Sensor

To detect the location of the animals head, we need to have some sort of sensor. To capture accurate data it is recommended to use a triple axis accelerometer, this will capture the movement in three directions. It is also possible to use a two axis accelerometer, but will not capture as much data. The Analog Device ADXL345 is recommended primarily because of its Serial Peripheral Interface(SPI) and Inter-Integrated Circuit (I2C) bus features. This allows for standard communication with the microcontroller, and other devices. It is a low powered 3axis accelerometer with up to +_16g measurement, data is outputted as 16bit twos complement format.

An alternative to this device could be the MMA7455 3-Axis Accelerometer Module, all though this does not provide +_16g but +_8g, this would still provide plenty for the application required. The MMA7455 supports the use of SPI and I2C. The reason that MMA7455 is not used over the ADXL345 is the fact that it requires more operating current in both stand by mode and full operation.

1.5 Alternative Design

An alternative design could be to have a mobile unit attached to the animal with an RF link to a base station. This design would only really work when the animal is located within range of the station. The station would be an RF receiver connected to a computer or storage device. The data that is sent from the mobile unit would be results of the accelerometer and any other sensors. This has the advantage of using a less powerful microcontroller as it only has to get data from the sensor and transmit using the RF transmitter. An advantage of this design would be that it is possible to view the results in real time on the base station, as well as providing near unlimited storage. Because all of the processing has been removed from the mobile unit, this will allow for longer

running times. Which would result in a more accurate recording of the data, due to the fact that the device will all ways be transmitting.

2 Interconnections between devices

The diagram below shows a very basic representation of how all the devices communicate. From figure 1 you can see that all of the devices are connected to the microcontroller. The SD storage and Real-Time Clock are connected to the microcontroller via the Serial Peripheral Interface. The devices are set as slaves and the microcontroller the master. This allows for a daisy chain of the SPI bus allowing for multiple devices to be used on the one SPI bus that the microcontroller supports. The reasons why these devices are on the SPI bus is because it provides a higher throughput than I2C or SMBus, this is required for the operation of these devices, mainly the SD storage. The SPI also requires less power than the I2C alternative, so it makes sense to run most of the devices on this than I2C.

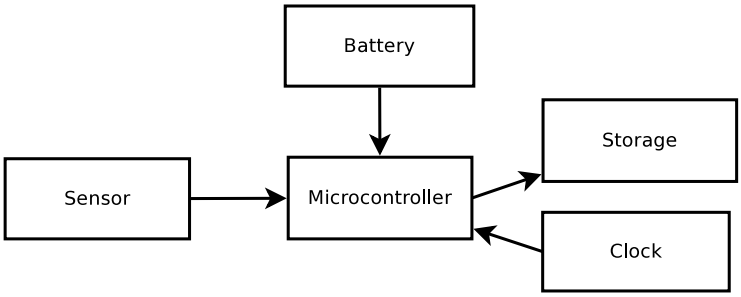

Figure 1: Basic representation of interconnections.

The accelerometer is run on the I2C bus due to the fact that it is only possible to communicate with one slave at a time on the SPI bus. So to get the most reliable and accurate data, its recommended to separate the sensor and the Real-Time clock. This will allow for the microchip to request data from the real-time clock and the sensor at the same time. Figure 2 shows a more complex version of the way that the unit is interconnected.

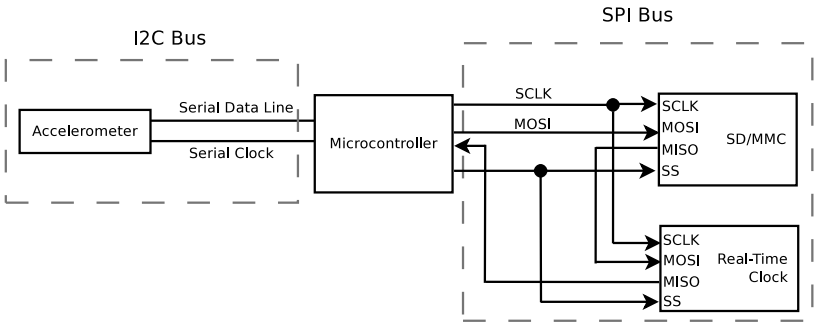

Figure 2: Showing the protocols used in interconnections.

3 Power Requirements

For the main microcontroller to run it is recommended to provide it with a minimum of 3.0V. It is possible for the device to draw up to 3.6V when the microcontroller is running at full power, and has a lot of data to process. Also the device can provide up to 200mA from all the ports. The microcontroller can draw up to 90mA when under full load.

The SD storage units are design to be used in mobile devices with limited power sources available. Providing the card inserted into the SD socket is build to a specification later than 1.0.1 then it will draw a maximum of 15mA. Once the card receives the CMD5 or ACMD41 initialization commands, the average current will be at most 50mA. For basic communication (CMD0, CMD15, CMD55, ACMD41) the device will require a minimum of 2.0V and a maximum of 3.6V. For other commands and memory access it is required that it has 2.7V minimum and 3.6V maximum. It is also possible to run the storage device with in SDLV mode, this will allow low voltage mode, with operating voltages in the range of 1.6-3.6V.

The real time clock, uses a small cell battery (typically running at 3.0V) to keep the correct time when the device is powered down, but during operations with the device it is required that it be supplied a minimum of 2.0V. This will allow for the clock to provide the time and keep counting. It it not recommended to provide the device with more that 5.5V as this is the maximum volts that it can take. It typically requires 3.3V for general use, and will draw up to 700µA

The accelerometer uses the least amount of power compared to the other devices on the unit. It requires 40µA in measurement mode and 0.1µA in standby mode. This is good as this device will be used a lot of the time, and is the main feature of the unit. The typical voltage required by this device is 2.5V. With the minimum of 2.0V and maximum of 3.6V.

It might be required to have a voltage regulator in place between the power source and the real-time clock, as it is possible for the clock to draw more than the standard of 3.6V, that the other devices use. The voltage regulator can be

used to prevent the clock from drawing too much current.

3.1 Energy Budget

From the table below you can see the maximum amount of energy that the unit will require in order to run. Its not necessarily going to use this amount of energy constantly, but if it does we need to be able to provide the unit with enough power.

Device	**Voltage**	**Ampere**	**Power**
PIC	3.6	90mA	0.324W
SD/MMC Socket	3.6	50mA	0.18W
Real-Time Clock	5.5	700µA	3850µW
Accelerometer	3.6	40µA	144µW

Table showing the energy requirements of each device.

The type of power source that is recommended for this unit are Lithium Ion and Lithium Polymer cells as these support 3.7V as their nominal voltages. Other types of batteries are such as Alkaline, Zinc carbon and Nickel Cadmium cells do not provide enough voltage for the system to operate. Although it could be possible to step up the voltage, it will be simpler just to restrict the voltage.

It is recommended to use a battery with around 2700mAh capacity. This will allow for 19hrs of continuous usage at full work load, this is not recommended by the manufactures, due to risk of damage. To improve the lifetime of the unit it is recommended that the device be put into sleep mode, to save on power consumption. Due the fact that the unit needs to be able to monitor the animal for signs of the abnormal behavior, it needs to be online for a minimum of one day. This will be 24hrs. If the unit were to activate and record data every two seconds, this would increase the length of operation from 19hrs to 38hrs, which will cover the minimum of 24hrs. The unit should be put into sleep mode, where it will use near to nothing power, every other second.

4 Size and Weight

The unit is comprised of five major components. As well as a variety of other smaller but still essential components, which are outside the scope of this report. Three of the components are small light weight microchips. These do not take up a lot of space, but are essential to the success of the unit. As you can see from the table below, the biggest item is the battery, this weighs the most and takes up much of the room. Following the battery the next largest devices is the SD/MMC Socket, all though this is not heavy or particularly large. It needs to be placed in an easy to reach position for the user to be able to insert and remove the SD/MMC card.

Device	Length	Width	Height
PIC	35.56	8.50	5.8
SD/MMC Socket	28.50	28.50	2.80
Real-Time Clock	13	10	5
Accelerometer	3	3	5
Battery	65	51	8

Table showing the size of all components in mm.

Figure 3 shows a prototype layout for the unit. From the figure, you can see that the SD/MMC socket is easily available to the user on the left hand side of the unit. The battery is located at the top of the unit, this also to provide easy access, and attempt to balance out the board. The total length of the unit will be 80mm and the width 65mm, with a maximum height of 8mm.

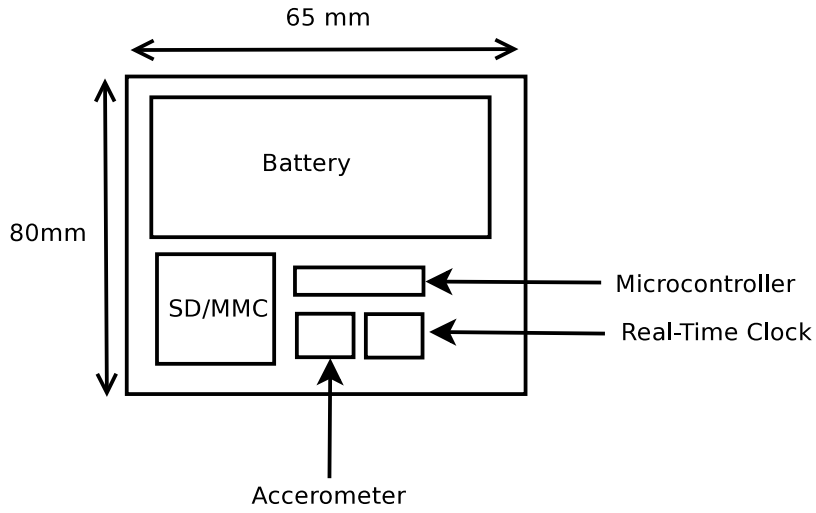

Figure 3: Prototype layout not to scale.

The total weight of the unit will be around 54g - 57g. This can depend on how heavy the weight of the other devices such as regulators and capacitors, will take up. Along with the weight of the circuit board. The heaviest item will be by far the power source, which weighs in at 52grams.

5 Costing

The price for a PIC24HJ12GP201 PDIP, the chip used in the measurements, from the manufactures site costs 1.89 GBP. But the minimum number allowed to buy is two. So you'll get a spare. The SD/MMC socket costs 2.42 GBP, this comes with a dummy card helpful during construction. The Real-Time

clock costs 6.09 GBP, it is also advisable to purchase a 12mm coin-cell, and holder to allow for battery back up of time keeping. The ASXL345 triple axis accelerometer, costs 9.15 GBP. The most expensive item on the unit is the lithium ion polymer battery which costs around 13.60 GBP.

The total cost of the unit will be around 33.15 GBP. This is with out any other materials or components that will be required to make a fully functioning prototype unit. For further information on the costing please see the "Costing" section in the bibliography.

6 Software overview

This section will cover the how the software will run on the unit. Figure 4 is a state chart of how the microcontroller will run. It boots up and if the button is pressed it will enter the configure mode, this is where the main program can be edited on a computer. From the configure mode it is possible to reboot to the boot loader or run the new program installed on the microcontroller's memory. If the button is not pressed it will run the main program, more on that later. If there is no main program and the button is not pressed the microcontroller will continue to load the boot loader indefinitely.

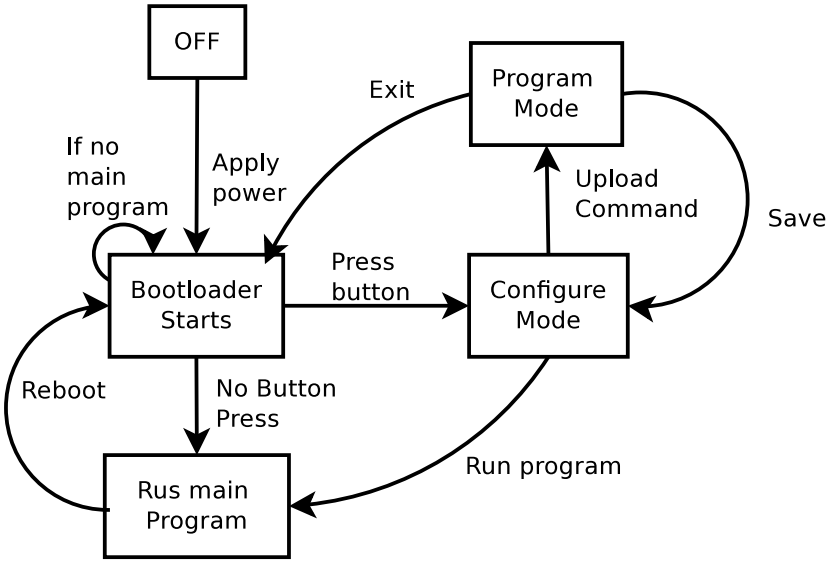

Figure 4: State chart of the microcontroller.

Figure 5 shows the overall flow of the main program. It starts by first making sure that all the hardware is working, and initializes all of them. If they are all working okay, then it will begin collecting data until it is shutdown or runs out of power. If it is unable to initialize the hardware it will end the program

and shut the device down. Figure 6 shows what is going on inside the "data collection" process.

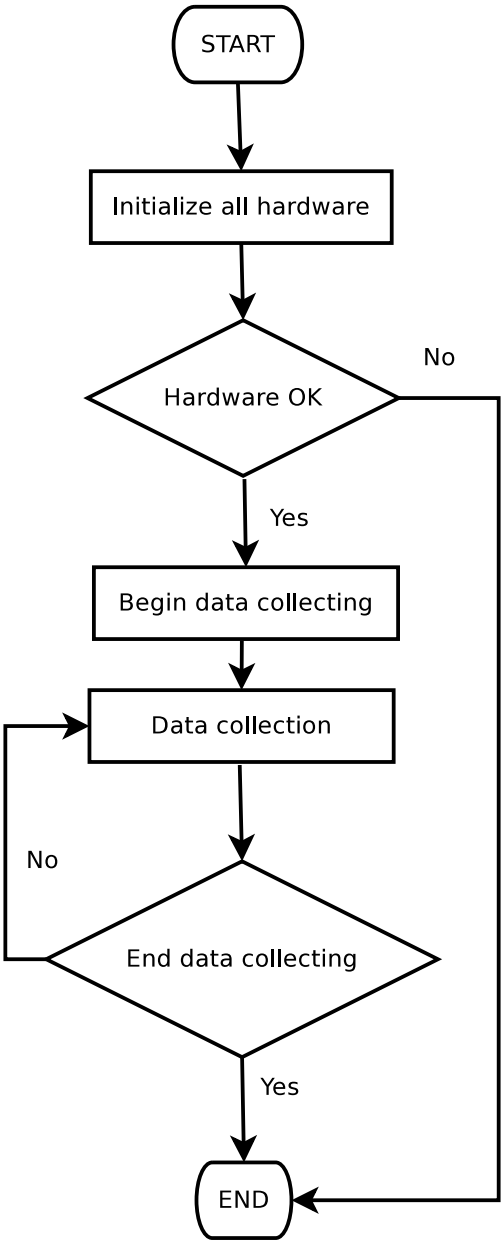

Figure 5: Flow chart of the main program.

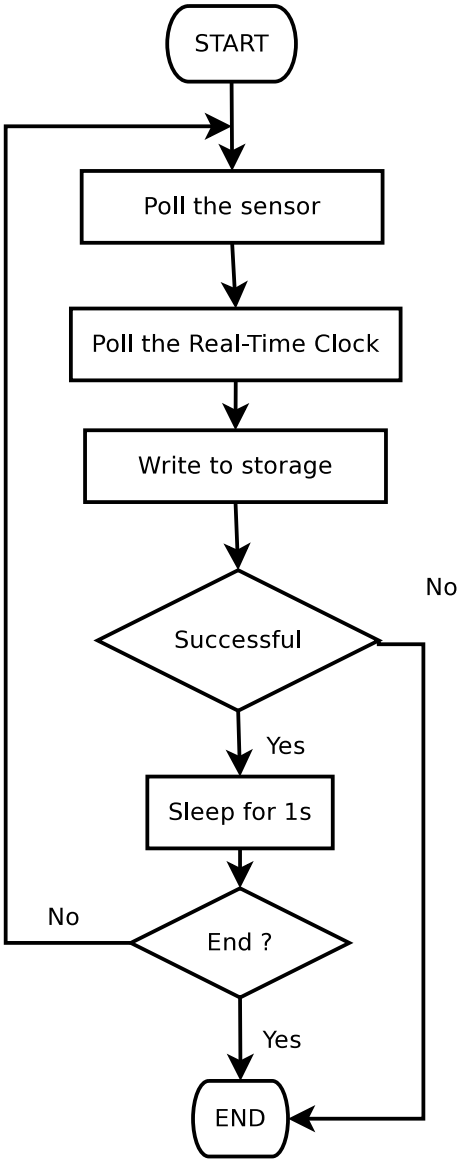

Figure 6: Flow chart of the data collection process.

The data collection process polls the sensor and the real-time clock, to get there values and stores then inside its internal memory. Once that has been done it is to output the vales as a easy to read value to the SD/MMC storage device. If this is successful, e.g there is enough room or a storage devices exists and is writable, then it will sleep for one second. There there is some sort of

error it will end the data collection process which in turn will end the main program and the device will shut down. After it has slept for one second it will check if the device has been told to end, and then return to the beginning of the process to start it again.

7 Main Program

The main program takes over operation of the unit once the boot loader gives control to it. This is a form of control over the unit, to make sure that it is able to complete the required task. The first function that gets called is "initialize hardware". This is basically a power on self test (POST), making sure that all the hardware is able to communicate and is working with in operational parameters. If they all pass, they will be left in sleep mode, where they will not be recoding data but able to be polled once required. If the hardware for what ever reason does not pass the test then the main program will shut down the device and attempt to recode this on the SD/MMC if possible.

Once all the hardware has been checked to make sure they are operational, the main program will then start the data collection process. It will wait until the collection process has completed its task, once it has been completed it will shutdown the devices then power everything down.

8 Data Collection Process

The data collection process's main function is to capture data from the sensors, and record it to the storage. When this function is run all the hardware will be running in sleep mode to save power. The first thing that this function will do is to power up the devices to get a reading from from them. The data is a combination of the results from two devices. These are the accelerometer and the real-time clock. Once the values have been read, the function should set the hardware back into sleep mode. Now that the data has been polled, and is stored in the microcontroller's memory, the function needs to power up the SD/MMS socket and write the data to the storage device, and put it back into sleep mode.

If it is unable to store the data then the data collection process should end, this in turn shutdown the entire unit. Other wise if the function is able to write the data, then it should set the microcontroller, to enter sleep mode for one second. Once the microcontroller is to wake up from sleep mode, after one second it will start the function again.

9 User Configurations

It is possible to change the length of time that the unit sleeps for after it has taken a reading, this can increase or decrease the length of time the unit can run for. This can be done from the configure mode, which can be accessed by

pressing and holding the button before the boot loader loads the main program. Once the unit is connected to a computer it is possible to change the configuration file, which stores the amount of time and where to store the data on the SD card. These are stored as a header file in the code. To change these values the user will have to re-compile the main program with these new values and then load the program on the microcontroller.

10 Software Development

The best way to implement the software would be to use the free and Integrated Development Environment (IDE) provided by the microcontroller manufacture, called MPLAB. This enables complete integration with the hardware and the software, making changes and development easy. MPLAB allows debugging and tests to be carried out. It is also possible to view to total amount of memory used when the program is running. It is also recommended to use a tool called a PICkit. This allows for inline debugging and logic analyzing.

10.1 Code

The main program has two major functions, these are then split into more functions. This is to help with debugging and overall code management. The first of the two main functions is "initialize hardware". This has three functions which will return a value based on the results of the devices that they are initializing. If all the devices are in working order and then this function will return a positive result. This could take up around 30 - 60 lines of code for each initializing function, and then around 5 lines to gather the and return the results.

The next major function is "data collection", this will have around five functions. These functions will be inside a loop, to ensure that the program continues to gather data. The first function is to poll the accelerometer for the values, and then put it to sleep. This will put the values into a register of the microcontroller, this will allow for other functions to access these values. This then happens for the real-time clock in another function. Both of these functions will be around 30 lines of code.

There will also be a function to write the vales that are in the microcontroller registers to the SD/MMC storage device. This function will need to clear the registers after the data has been written. Depending on the results of writing to the storage, it will return a value. This function will be around 60 - 120 lines of code. The next function will be called if the write function returns true, this puts the microcontroller to sleep for a specified time, default one second. This could be around 20 - 40 lines.

The last function is used to break out of the loop, and shutdown the unit. This will ether wait for a button press or if it gets given a value to terminate from one of the other functions. This is a very small function, around 10 lines.

In total there will be around 245 - 415 lines of code. This is just an estimation from my experience. The functions to check if the hardware is working will not require much memory. The other functions which store data will need a minimum of 18bytes to store the data. I believe that the whole program should take no more that 32bytes of memory. Again this is from my very little experience.

10.2 Library Code

It is highly recommended to use the library header files that come with the microcontroller. Some recommended items are the p24Hxxxx.h file, libpic30-coff.a which come with MPLAB C30 toolsuite. These files will help with the development of the code, they have predefined data structures and macros. These are useful for interfacing with the microcontroller.

References

[1] PIC Microcontroller, *http://en.wikipedia.org/wiki/PIC_microcontroller*
Last Access 24 April 2011.

[2] Information on 8051 Microcontroller, *http://www.engineersgarage.com/8051-microcontroller*
Last Access 24 April 2011.

[3] Intel MCS-51, *http://en.wikipedia.org/wiki/Intel_MCS-51*
Last Access 24 April 2011.

[4] Atmel - 8051 Architecture,
http://www.atmel.com/products/mcu8051/default.asp?category_id=163&family_id=604&source=left_nav
Last Access 24 April 2011.

[5] Accelerometer Background, *http://en.wikipedia.org/wiki/Accelerometer*
Last Access 24 April 2011.

[6] Information on the PIC24H, *http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en520472*
Last Access 24 April 2011.

[7] SPI Information for PIC24H, *http://ww1.microchip.com/downloads/en/DeviceDoc/70243b.pdf*
Last Access 24 April 2011.

[8] Serial Peripheral Interface Bus, *http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus*
Last Access 24 April 2011.

[9] AN1045 - Implemening File I/O, *http://ww1.microchip.com/downloads/en/AppNotes/01045a.pdf*
Last Access 24 April 2011.

[10] SD Memory Card Specifications, *http://tinyurl.com/5s3kkwu*
Last Access 24 April 2011.

[11] Sandisk SD card spec, *www.sandisk.com/Assets/File/OEM/Manuals/SD_SDIO_specsv1.pdf*
Last Access 24 April 2011.

[12] SD/MMC Socket datasheet, *http://www.sparkfun.com/datasheets/Prototyping/SD-Socket-PP-19607.pdf*
Last Access 24 April 2011.

[13] Real time clock Background, *http://en.wikipedia.org/wiki/Real-time_clock*
Last Access 24 April 2011.

[14] Background Information on SPI bus and application, *http://www.youtube.com/watch?v=FvsXcpM2qA4*
Last Access 24 April 2011.

[15] MPXY8300: Microcontroller, Pressure Sensor, X-Z Accelerometer and RF Transmitter, *http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=MPXY8300*
Last Access 24 April 2011.

[16] RFM12B-S2 Wireless Transceiver, *http://www.sparkfun.com/products/9582*
Last Access 24 April 2011.

[17] Lithium ion polymer battery, *http://en.wikipedia.org/wiki/Lithium-ion_polymer_battery*
Last Access 24 April 2011.

[18] Lithium ion the ideal battery?, *http://batteryuniversity.com/learn/article/is_lithium_ion_the_ideal_battery*
Last Access 24 April 2011.

[19] Lithium ion cells list, *http://www.ibt-power.com/Battery_packs/Li_Ion/Lithium_ion_cells.html*
Last Access 24 April 2011.

[20] Lithium ion polymer battery specs, *http://www.adafruit.com/blog/2011/01/24/new-product-large-lithium-ion-polymer-battery-2700mah*
Last Access 24 April 2011.

[21] Lithium ion battery, *http://en.wikipedia.org/wiki/Lithium-ion_battery*
Last Access 24 April 2011.

[22] Nickel cadmium battery, *http://en.wikipedia.org/wiki/Nickel-cadmium_battery*
Last Access 24 April 2011.

[23] Amp, amp hours, watts, watt hours, volts!!!, What's all this really mean., *http://youtu.be/E2NoL8-DIGk*
Last Access 24 April 2011.

[24] Electric Potential and Potential Difference, *http://youtu.be/wT9AsY79f1k*
Last Access 24 April 2011.

[25] PICKit, *http://en.wikipedia.org/wiki/PICKit*
Last Access 24 April 2011.

[26] PIC24HJ12GP201/2, *http://ww1.microchip.com/downloads/en/DeviceDoc/70282D.pdf*
Last Access 24 April 2011.

[27] DataFlash 16Mbit AT45DB161D, *http://www.sparkfun.com/datasheets/IC/AT45DB161D.pdf*
Last Access 24 April 2011.

[28] DS3234 Real Time Clock, *http://datasheets.maxim-ic.com/en/ds/DS3234.pdf*
Last Access 24 April 2011.

[29] DS1307 Real Time Clock, *http://www.sparkfun.com/datasheets/Components/DS1307.pdf*
Last Access 24 April 2011.

[30] ADXL345 3-Axis Accelerometer, *http://www.sparkfun.com/datasheets/Sensors/Accelerometer/ADXL345.pdf*
Last Access 24 April 2011.

[31] MMA7455 3-Axis Accelerometer, *www.jameco.com/Jameco/Products/ProdDS/2109667-spec.pdf*
Last Access 24 April 2011.

[32] General Items, *http://sparkfun.com*
Last Access 24 April 2011.

[33] General Items, *http://uk.farnell.com/*
Last Access 24 April 2011.

[34] PIC24HJ12GP201 Microcontroller, *http://www.microchipdirect.com/ProductDetails.aspx?Category=PIC24HJ12GP201*
Last Access 24 April 2011.

[35] SD Socket, *http://www.sparkfun.com/products/136*
Last Access 24 April 2011.

[36] DS3234 Real Time Clock, *http://www.sparkfun.com/products/10079*
Last Access 24 April 2011.

[37] ADXL345 3-Axis Accelerometer, *http://www.sparkfun.com/products/9045*
Last Access 24 April 2011.

[38] Large Lithium Ion Polymer Battery - 2700mAh, *http://www.adafruit.com/index.php?main_page=product_info&cPath=44&products_id=328*
Last Access 24 April 2011.