Mobile Studio's Wireless Expansion: a low-cost, wireless toolset for expanding Mobile Studio's instrumentation suite

By

Mathew Philip Wilson

A Thesis Submitted to the Graduate
Faculty of Rensselaer Polytechnic Institute
in Partial Fulfillment of the
Requirements for the degree of
MASTER OF SCIENCE
Major Subject: ELECTRICAL ENGINEERING

Approved:

____

Dr. Don Millard, Thesis Adviser

Rensselaer Polytechnic Institute
Troy, New York
December, 2008
(For Graduation December 2008)

Table of Contents

Acknowledgments

Many people were involved in making the Wireless Expansion possible. I would like to thank the Physics department for giving me the opportunity to work on a project of this scale and impact. I would also like to thank my loving family who has been supportive through all the twists and turns this project has taken. As for the others who worked with me on this project your help and guidance has been invaluable. First off I would like to thank Jason Coutermarsh who is my mentor and my friend, without him I never would have continued for my masters. I would like to thank Eric Allen for selflessly providing support when I most needed it; it is because of him that the wireless boards were assembled on time. I would like to thank Bill Brubaker for being a constant source of help and support, even when he was most busy he made time for me. Without Bill I would likely still be stuck trying to figure out bugs and structural concerns in Mobile Studio. Lastly, I would like to thank Don Millard, my advisor and friend who I have been with for four years now. From freshman year he has given me opportunities to explore advanced topics and mentored me through my college career. Thank you, without your help I would not be who I am today.

Abstract

The Mobile Studio Project and the Wireless Expansion are tools designed to provide affordable, simple, and powerful teaching aids to lab courses. Although it was originally designed to provide cost-effective access to expensive test instrumentation, the Mobile Studio hardware and pedagogy has spread into advanced engineering courses as well as other disciplines. While the original hardware for Mobile Studio has already been used in multiple settings, the Wireless Expansion allows for new types of environments to be created under the same philosophy: "Let the students play."

Originally conceived in 1999, the Mobile Studio vision was founded on the observation that incoming students' intuitive understanding of how circuits and electrical systems worked was declining year after year. The Mobile Studio rekindles the ability to tinker and provides the opportunity for every day to become an experiment day. The Wireless Expansion system continues in these themes and allows immediate study of phenomena as well as the ability to test and construct experiments and projects to satisfy one's curiosity/needs; without requiring a physically wired connection to the monitoring/controlling computer. This document contains information relating to the design, development, fabrication, operation, and functionality of the Mobile Studio's Wireless Expansion system.

1. Introduction

1.1 Genesis of Mobile Studio

Integrated chips are now prevalent among electronics, resulting in discrete circuit topologies being found in fewer devices. It has become harder to tinker and gain experience with hardware. In order to manipulate an iPod you need significant background in electronics. You cannot even try to exchange the battery of an iPod for a custom power source of equal voltage, since the iPod communicates with the battery to make sure it is properly licensed. As the level of complexity of devices is increasing, it becomes harder for classes to relate the material to the interests of the students. The students want to know how an iPod works, although they don't care about how an RC circuit reacts with an AC signal. There is a gap of understanding to how an RC circuit can be used to manipulate the signal of an iPod.

In order to bridge this barrier a new method of teaching is required. Tens of thousands of dollars in lab equipment for a pair of students to spend 5 hours a week tinkering with the lab equipment is not yielding the returns given the investment. Many students spend most of the class setting up the lab, only to have to tear it down when the time runs out. Here at RPI, in order to combat the declining hands-on experience, we have implemented studio courses. Studio courses take the standard lecture, lab, and recitation model and combine them into a lecture-and-lab hybrid. In each class there is a lesson portion after which the students work on a physical instantiation of that lesson. Though the benefits from this method are apparent, the cost to implement studio teaching would for many schools be an impassable impediment. The studio method requires all lecture periods to have access to laboratory facilities. This is where Mobile Studio, now referred to as Mobile Studio Project, comes in. In order to keep the benefits of a studio class, RPI needed a cost effective lab suite that could be easily deployed. On top of that, the Mobile Studio is able to give students tools to use for projects along their trip through college.

1.2 Mobile Studio

The Mobile Studio Project was originally designed to be geared towards electrical and computer systems engineering education. This meant a heavy focus in electrical measurement. The Mobile Studio I/O board [8] went through 3 different versions with multiple revisions for each version. The latest version of the Mobile Studio I/O board is revision C of the RED2 version. For the purpose of this paper only the RED2 Mobile Studio I/O Board (RED2) is discussed, since it is the only board with on-board wireless capabilities. The RED2 was designed to be a lower-speed, low-cost circuit measurement workstation. This includes, but is not limited to, an oscilloscope, function generator, and +/-4V_{DC} power supply. In addition to the base functionality of the Mobile Studio boards' functionality, a lot of thought was put into how other areas of education could benefit from the advantages and improvements that circuits and electronics courses at RPI had gained. Two main efforts are currently underway to extend the board's utilization.

The first is the development of a Component Object Model (COM) object. The COM object is a programming interface that allows other programs to directly access the base functionality. This offers access to raw data streams with no graphical interface. Both LabVIEW and Matlab have been successfully used to view and manipulate the information from the I/OBoards using this object. The COM object opens many possibilities for using the board to monitor signals and alsoproduce them. For instance, LabVIEW code has been written to demonstrate control algorithms. PID controllers currently studied only in lecture can now be studied through experiment at only the cost of a motor and a board. This also opens up endless opportunities to use the board in projects which would have originally been daunting to students.

The second step is the development and use of a wireless antenna on the RED2. This was added to allow for the design of an unlimited number of applications that could be used through the Mobile Studio system. It was originally thought that different boards would be manufactured to utilize the wireless functionality; experimentation has since led to the development of a more generic, multifunctional Wireless Expansion board. One goal is to take the programming of wireless protocols out of the hands of the developer in order to create much cheaper adapter cards which then utilize the Expansion's link.

1.3 Mobile Studio's Wireless Expansion

As with the Mobile Studio I/OBoards, the Wireless Expansion is targeted for an audience desiring cost-effective but feature-rich utilities to replace expensive lab equipment. With the Wireless Expansion, (WEXP), the scope of the Mobile Studio's pedagogical effectiveness in different classroom environments has greatly increased. Although most phenomena can be observed as analog signals, the devices to translate those observations into useful voltage levels is a task on its own. On top of that, taking those analog voltages and recording them in software requires another completely different skill set. For this reason, even though interested parties may have the capability to make a sensor, they may lack the ability to create a device that can be used in a multi-user environment such as a classroom. RPI's physics department first approached the Mobile Studio developers with this problem.

RPI's physics department is using a system called Logger Pro. Logger Pro is a system that uses software on TI calculators in order to collect data, but that data is not accessible for real time manipulation. Logger pro has a computer connection as well, however the panels are somewhat complicated to use. The physics department desired a new system but was limited by cost and technical considerations. They could buy new sensors, but the software to control the system would remain the same. If they switched to a new system, they would lose the functionality of all of their old sensors and the updated software was out of the price range of the department. In parallel with the deployment of Mobile Studio in Physics II, Physics faculty expressed a request for similar Mobile Studio functionality for Physics I. Physics II is the study of electrical phenomena, where as Physics I is the study of mechanical. Given that the software was being distributed with the purchase of a board and had an updated user interface, exploring the potential to transfer was obvious. The preliminary results for the RED1 showed that the quality of signals was not sacrificed for the types of experiments being done in Electric Circuits; therefore the Physics I faculty wanted to explore the viability of using the Mobile Studio to acquire dynamic data associated with the course's experiments.

After Physics I's proposal was made, the immediate advantages of such a system were apparent, and work began on the Physics I daughter card. The RED1 had an expansion slot which allowed for another card to connect to it to use the USB as if it were a UART channel. UART is a simple communication protocol which can be used between systems, or even used between different ICs. Data is sent asynchronously and most microcontrollers have the hardware already implemented internally. On the other hand, USB (Universal Serial Bus) is difficult to implement since it requires not only custom code for managing the channel on the microcontroller side, but also requires code on the desktop to monitor the channel on the host side. By having the expansion connector on the Red1, even though none of the functionality on the Red1 was used, it allowed for new applications to be developed that used the same USB drive software. The complexity of the physics daughter card was greatly decreased by this feature, and demonstrated that Mobile Studio did not have to be just one product, but could be a slew of different products for different applications.

The original physics daughter card was completed but never used in a classroom, due to two main problems. The first was the streaming mode required the device to be tethered to the RED1 board. The tether ended up being the USB cable which limited the range of the board to about 3 feet for streaming. It was good for testing but could not be used in an experiment. The second problem was the non-tethered mode was just as complicated as the Logger Pro software. To run an acceleration experiment, you would use the GUI to set up the desired settings and then disconnect the board. At this point you would run the experiment by pressing a start button on the board but you could not see the graph in real time as the data was collected. This severely limited the effectiveness for use in a class. While it measured the data correctly, it did nothing to reinforce the link between the data and the actual course followed by the physical object being tracked for a new student who did not know what to expect. At this point, the Physics I department requested that an entirely wireless solution be created. This was the genesis of the Wireless Expansion.

After the physics project was put on hold, the BLUE1 board was created. This board was a much higher-performing board geared toward upper level classes. The BLUE1 board was initially used in Physics II, was again generating a desire for some solution that could be used with the Mobile Studio system in real time. It was when the RED2 was developed that this capability was finally added. The RED2 included a printed wireless antenna that utilizes the 2.4 ISM band. The main goal of using the antenna was to replace the daughterboard connection with wireless connections in order to create tether-free daughter cards. At this time, work began on the wireless version of the Physics I board. Through the course of the development components were added to the board to enable other devices to utilize the wireless utility without knowing how to interface with the wireless protocol. As the features for the expandability increased, it became obvious that two things were being developed. One was the Physics I card and the other was an extensible system meant to help take some of the work out of developing future wireless cards. The downside was the physics card began to get expensive for the features it provided, and the expansion had a limit. Anything more complex than a relatively simple sensor would require a new wireless chip anyway. A simple sensor being, a basic signal such as an analog voltage or digital reading that did not require new programming. At this point, all of the desires that had been expressed during brainstorming schemes came together and it was clear that the design had gone past designing a physics board and a multi-purpose wireless adapter. With this in mind, all the Physics I components onto a separate device and work began work on strengthening the scope and extensibility of what was to become the Mobile Studio Wireless Expansion.

2. Historical Review

2.1 The Mobile Studio System

The Wireless Expansion is built upon Mobile Studio software environment and existing hardware. To do this custom hardware had to be build to create the wireless bases since the RED2 used Cypress's wireless USB which other existing systems could not interface with. It is possible to write code to interface existing hardware with the Mobile Studio software but the main concern was the cost of the existing systems.

2.2 Similar Systems

Other systems were investigated in an effort to create a product that would not only accomplish the goals in the short term, but also be a lasting tool that could be used outside of coursework. Mentioned here are three devices which covered a sampling of different applications. The MICAz, from Crossbow, are designed to be used in low power sensor networks where distributed processing is used in order to relay results back to a main system. The Arduino, on the other hand, is specifically meant for embedded control of devices. National Instruments also has wireless products available with which LabVIEW can be used to control or display signals from the device. Each of these systems has some sort of expansion connection method, but all of them are geared towards different applications.

2.2.1 MICAz

MICAz are small devices about the same size as the WEXP. They contain an expansion port meant to interface with daughter cards. At the time of the conception of the WEXP, Crossbow, the company that produces the MICAz, had just announced the upcoming release of the Imote2. It is extremely important to note that during the design of the hardware for revisions A, B, and C none of the specs for the Imote2 were known. Now that the Imote2 has been released and revisited, the similarity between the two seem like it was done by design. In reality it is the result of two independent development paths that ended picking similar ideas, layout topologies, and even the same power switch and connectors.

During the original investigation of existing platforms Crossbow had the MICAz. MICAz seems to be the original Imote even though it has a different product name. The MICAz contain a base board with an expansion slot that communicated wirelessly to other boards that could be stand alone entities or tethered to a computer. On top of that, the MICAz had an embedded operating system called TinyOS along with other tools meant for developing not just wireless sensors, but sensor networks [4]. Minus the embedded operating system, the MICAz was pretty much the exact functionality that the wireless expansion was looking to leverage. The MICAz could have been used instead of developing custom hardware except for the MICAz one flaw. Each board cost $100 dollars. In order to build an extension to mobile studio two MICAz devices would have to be purchased along with the tether costing over $275 dollars. The MICAz's wireless connection has a reported data rate of 250kbps which is the same as Cypress's wireless USB.

The most interesting part of the MICAz was the TinyOS operating system. For advanced users it provided an easy way to create custom autonomous sensors. Thought was put into implementing something like or even TinyOS itself and has not been ruled out yet. The current goal is to provide the ability to control the device from computer software lowering the background entry bar to use the system. In the future this should be revisited in an effort to

2.2.2 Arduino

While the MICAz could accomplish pretty much any embedded task it was extremely expensive for a user who was looking to get into embedded control, but hadn't gotten their feet wet yet. The Arduino is a device that is geared towards hobbyists who are just getting started with embedded control. None of the specs are impressive as the device is geared mostly toward digital and serial communication in an effort to create software that is as simple as possible. To do anything with an external chip the code will always go through the same process. This limits the Arduino to interfacing with high end products but to the targeted market there is always a chip that can interface through some serial protocol.

The Arduino does not have any wireless capabilities but has expansion boards that can give it Bluetooth. The cost of an Arduino is much cheaper the MICAz at $35.00 dollars, but the Bluetooth expansion is $150.00 dollars. Even though this is not geared towards wireless sensors the Arduino has had success in attracting new developers in the field of embedded design that Mobile Studio wishes to emulate.

The success of the Arduino is attributed to three things. First, the Arduino is an open source hardware project which means that boards can be printed and put together for the cost of materials. Second, the Arduino base is cheap to buy from a 3rd party. The Arduino has its own programming language which simplifies many tasks to follow the same format. From the comparison of MICAz and the Arduino, the lessons taken away are that to gain beginners to the embedded world it needs to be cheap, but to keep the more experienced developers interested there needs to be more functionality then just serial communications. The other interesting point is that both of these systems have their own programming language.

2.2.3 Wi-Fi DAQ

The Wi-Fi DAQ system is on the sensing side of things. LabVIEW can be used in order to control the daughter cards but most of the daughterboards have sensing devices and ports exposed. LabVIEW's Wi-Fi DAQ system runs at 1.2Mbps. The device is obviously not comparable to the WEXP and it shouldn't be since the price range is in the $500.00 dollar range. The nice thing about looking at the Wi-Fi DAQ system is that is showed that pretty much any system that is geared towards multiple types of sensors is going to have some sort of expansion card mechanism.

National Instruments claim that the system can be used wirelessly with just eight AA batteries. The Wi-Fi DAQ clearly is not meant for embedded use, but it has the advantage of being able to be used with LabVIEW. To those who own a license to LabVIEW, and that is everyone at RPI, this trumps having a custom programming interface. Users with no programming interface are able to use LabVIEW to create complex tasks for hardware. This insight raised a certain possibility, Mobile Studio already works with LabVIEW and it would be a small task to add the WEXP device to the LabVIEW library. This would, for the short term, replace the need for developing a custom operating system and give students a chance to use embedded sensors and control embedded devices without the requirement of pre-exposure to programming.

3. Wireless Expansion Hardware

The synergy of hardware, firmware, and software has helped to enable theMobile Studio's success. The hardware consists of the actual physical device, which is composed of integrated circuits which are connected in a way that (perhaps with some help from the firmware) are able to carry out various tasks. The firmware is specific to each processor and board configuration. The firmware deals with knowing the specifics of the hardware and given a generic task the firmware is in charge of orchestrating the hardware into a state in which the task can be performed. The software is what communicates the tasks to the firmware and interacts with the computer user. All of these are interconnected; i.e. when designing the hardware, decisions for the firmware and desktop software have to be made in conjunction with each other. For ease of reference, the following overview of the WEXP is structured as if the hardware, firmware, and software are independent systems.

Many decisions that are discussed for the hardware are not just for increased performance, but have resulted due to a balance of many factors. From end-usability to mechanical constraints, each of these factors has played a huge role in the design for the wireless base. The WEXP went through 3 revisions before it was used by anyone outside of the design team, and further revisions are sure to come as more feedback is obtained.

3.1 Design Specifications

The introduction described how the WEXP is the outgrowth of intended desired wireless physics card. The original guidelines for the physics card included the following requests for an ability to:

- Use current physics sensors
- Measure acceleration
- Measure force
- Measure range
- Trigger on events
- Be powered for at least two hours (One class period)
- Operate in multi-user environment (at least 50 students)

Other than these requirements, the physics department was happy to have the device in any format. The only other predetermined constraint was that the wireless system had to use Cypress's wireless USB protocol. The RED2 was already deployed and utilized Cypress's wireless USB transceiver hardware. The Cypress transceiver chip was chosen mainly due to cost. Wireless communications are expensive, but Cypress offered relatively inexpensive solutions with an average bandwidth, as far as portable devices go. If higher bandwidth is desired in the future a daughter card for the RED2 can be built which would contain a higher speed wireless transmitter. Such a daughter card would be connected via a wired connection to the I/OBoards. To clarify, the wireless expansion also has its own daughter boards. These daughter boards interface through an expansion port on the WEXP.

A specific physics-oriented wireless card was not built due to a desire to allow all students to benefit from a wireless card. The vision is to have students and developers outside of the physics area be able to take advantages of the system. It became obvious that the physics portion was better seen as an add-on to a more flexible Wireless Expansion system. The rest of the requirements from the WEXP are as follow:

- Small, no bigger than 2"x2"
- Easily rechargeable
- Run on 50mA
- Parts for board should be less than $30 in bulk
- Analog voltage range of at least 3.3V
- 4 ADCs
- 4 DACs
- 4 Digital IO / Pulse width
- External Processor Interface
- Mechanically stable
- Firmware can be updated by user
- Interface with COM object
- Cross Platform

3.2 Revision A

Revision A satisfied all of the design constraints and added features to increase usability. This led to the design taking far longer than simply completing a prototype. The goal was to test the usability so that future revisions would be easier to use then the current lab tools. The original sensor board design was just a simple jack to plug in existing sensors to demonstrate the capabilities of the Wireless Expansion.

As a result of the initial experimentation and testing, the wireless expansion system was split into a base board and a sensor board. To clarify, the Sensor Board was designed to be connected to the Base Board through a connector which has standard pins to interface with any future sensor boards. The base board contains all of the intelligence and capabilities to communicate with the RED2 board. A Sensor Board is connected to the base board which contains the hardware for acquiring sensors.

3.2.1 MCU

The next step was to select the main Micro Controller Unit, MCU. Several MCU processors were evaluated. The RED1 I/O board as well as the original physics board was built with an ARM device; the ADUC2076. At this point, the only thing that was confirmed was the wireless chip had to be one of Cypresses Wireless USB chips. The ADUC2076 worked well in both previous applications and had only been abandoned in the pursuit of faster, more accurate signals. However, with the limit of 250Kbits per second, which meant one channel of an 8-bit Analog to Digital Converter (ADC) could only record a 30 Khz signal at maximum speed; therefore the ADUC2076 would have performed well as it would not be the bottle neck. The connector would be laid out to give access to analog in, out, and digital ports along with some administrative connections. Since we already possessed the development tools and had experience with this processor it seemed like an obvious choice. On the other hand, even though the ADUC2076 seemed like the best choice in terms of speed for development of the wireless base, Cypress offered an integrated microcontroller and wireless communications product.

The attention turned to Cypress to evaluate their combined microcontroller. At this time, we still valued the speed with which we could develop on the 2076. However, with the integrated chip that Cypress offered, time would be saved on the development of the wireless communications, which was the majority of the project. Cypress's CYWUSB6953 was the first device that was investigated. The CPU clock, at 12MHz, was far above the bottleneck threshold of communicating at 250Kbits a second. It had enough digital pins to add to a connector and had the wireless receiver built in. It also had something Cypress referred to as analog and digital blocks. The CYWUSB6953 claimed to be able to configure a pin to have access to digital input and output or analog input and output. This feature was the driving motivator to switching to a never before used processor.

The analog and digital blocks turned out to be exactly what the wireless expansion needed to set it apart from other devices with expansion slots. These blocks could be configured to do almost any signal processing function desired. It is basically a Complex Programmable Logic Device (CLPD), packaged inside a microcontroller. The expansion connector could now utilize filters, counters, amplifiers, and many, many more functions. To accomplish this with the ADUC2076the connector would have to be 10 pins longer, or else there would be a complex signal switching process and it would still only cover the base configurability of the digital and analog blocks. The downside to this processor was that functions such as counters each take up one of these resources. The wireless communication required a few digital blocks to aid in the timing of the wireless transmissions. In order to be able to use the configurable blocks as well as have enough resources for wireless communications, we began looking at Cypress's Programmable System on Chip (PSoC), line of processors.

Several PSoC processors were then evaluated. The main desire was to garner extra digital blocks since the Serial Peripheral Interface (SPI), a protocol for inter-chip communications, and timer blocks must always be active. During debugging the UART also needs to be active. This requires 4 digital blocks to debug the processor. The CY8C21x34, CY8C2x23, and CY8C2x43 series were investigated while learning more about how the configuration of blocks worked. In the end, the CY8C2x43 series had the eight digital blocks necessary as well as twelve analog blocks. The last thing to select was the processor packaging style.

At this time, the assembly is currently being done in-house. This meant whichever package was chosen had to match the in-house ability to solder the component. This led to initially picking a processor in a TSOP form. It was a worry, however, that the signals acquired would be noisy since there was only one ground pin on the device. This meant the ground was distributed from that single pin to all of the other internal connections. Any noise from one resource would potentially produce noise for all of the resources since the ground connection was routed past all of the components involved. The QFN (Quad Flat No-lead) package had a bottom ground plane which allows for noisy signals to be compensated by the whole of the ground plane. Resources could be routed directly to the ground plane without having to share a noisy bus. Even though it is more difficult to mount this type of packaging, the 48QFN CY8C2643 processor was picked. With the processor chosen, the selection for the wireless IC was comparatively quick.

3.2.2 Wireless USB

At the time of this stage of the development, there was only one version of Cypress's new protocol available, the CYRF6936. This implemented their LP protocol which combined two of their previous protocols focusing on reducing power and improving range. The wireless library could be used since the microcontroller selected was a cypress microcontroller,. With the processor and wireless chip selected, the next step was to design the power system.

3.2.3 Power

The board was intended to be available for use in many different areas of education. In order to accomplish this, it was necessary for the board to be supplied between 3.7V and 9V. The 3.7V value is the voltage of a single lithium ion battery that can be found in cell phones or an iPod. The 3.7V is the standard battery connected to the system for use as a sensor. To accommodate use in a robotics project, the 3.7V is designed to be disconnected in order to take advantage of the robot power supply. The max of 9volts was picked so that a 9V battery could be used as well. In order to use

3.7V as a power source, the other chips must function at either 3.3V with an extremely low dropout regulator, or at 2.5V.

The processor and wireless chips were both capable of operating at 3.3V. The processor was able to run at 5V with a speed of 24Mhz. The speed is limited to 12Mhz when running at 3.3V. The processor is able to function at 3V in low-power mode but reduces the expected performance level and limits the voltage that external sensors could interface with. For this reason, digital regulators were first investigated. Digital regulators have the advantage of having an extremely low Dropout Rate, which refers to how much voltage drop there has to be between the input and output for the regulator to function properly. Digital regulators basically rapidly switch the voltage on and off to the circuit. The longer the time it spends on, the higher the voltage. The only problem with this is it provides a noisy power source. For digital systems this might not have been such a problem, but this board was supposed to accurately handle precision ADCs (up to 14 bits). With this in mind, attention turned back to finding a linear regulator with a low dropout voltage.

Though linear regulators have higher voltage dropouts than digital versions, the amount of current the system would draw was small. As current pull increases, the dropout of a linear regulator increases. The system was being designed to draw 100mA when fully powered. The wireless chip draws 25mA on average when communicating. This leaves another 25mA for the rest of the system and 50mA max for a sensor without external power. The regulator chosen can supply up to 500mA but this will come at the cost of significant reduced battery life. The battery picked to power the device is an iPod nano battery.

The iPod nano battery supplies 500mAH (milli-amp hours) of continuous current. This would give the board a life of 10 hours of constant communication. This battery made it possible to find a linear regulator which would work. The TPS71333QD linear regulator from Texas Instruments has a dropout voltage of about 25mV at 100mA. This means the base board will function with the battery drained to 3.325V. This put the operation of the device below the useful limit for most lithium ion batteries. The device would operate for the full range of the battery discharge curve.

3.2.4 Sensor Connection

The next step was designing the connector interface to complete the minimum for the board to function as the base for a sensor board. One of the most challenging usability aspects for revision A was finding connectors that gave good mechanical connections, but also were not bulky. The board was being manufactured into a 1"x2" board, so space was a critical issue. The connector needed to be a board-board connector, since making the device interconnection as simple as possible was desired. Wire connectors could have been used in-between the boards with some type of mechanical fastening used to hold the two together, but when a single connector could solve both problems the wire connectors would have been just a complication. The third desire was to have a connector that developers could easily work with. A breadboard, for instance, is basically a big socket connector which allows wires to directly interface with it. A similar thought was applied to the connector of the base board. Instead of an awkward custom connector, we preferred to have an open socket connector which could be directly interfaced with.

After selecting a socket type connector the next step was to decide which pins would be available. The original plan was to break the connector up into multiple connectors. This would isolate different systems from a developer's standpoint. For example, one connector would have all analog connections and another connector would contain all digital connections. This makes it easier when learning a new system to be able to focus on the specific task at hand. The intent was that each sensor board would interface with only the connectors required. If multiple connectors were used, this would increase the mechanical strength of the boards. Ideally, a production sensor board would use all four connectors.

The functions were split into four areas; the primary connector, the expansion connector, the identifier connector and the parasite connector. The primary connector has all of the components that would allow for a basic sensor board to function. Included on this connector are four configurable analog/digital input/output ports, power and digital ground, as well as analog ground and the band gap reference. The expansion connector consisted of eight extra exposed pins from the processor. Four of these pins can function as analog in pins or digital in/out pins. The remaining four can only function as digital in/out pins. All eight pins on the expansion port have the ability to be connected to the digital blocks, and four of those pins to the analog blocks. The processor is able to connect pins to blocks through a shared bus. The CY8C2643 has sixteen individual lines of this bus available. Three of these lines are taken up by the SPI communication system leaving thirteen lines open. The primary connector and the expansion connector utilizes twelve of these lines if each pin was connected to a digital or analog block. The remaining bus line was left unused. When in debug mode, two pins from the expansion connector are used for UART communication. The expectation is that most constructed sensors would use the primary connector and the expansion connector. The remaining two connectors are meant for more advanced systems.

The identifier connector is for a sensor board that has been registered with Mobile Studio. Once Mobile Studio connects with a wireless board, a prompt asks which kind of sensor is attached. A registered board will automatically open the proper feature inside Mobile Studio when the sensor connector is in use. This is to help limit possible damage that could occur to a sensor by accidentally loading the wrong configuration. The last connector is the parasite connector. This is meant for advanced developers who wish to interface with Mobile Studio but need direct control over some of the resources. One example of when someone would want to use this is for an experiment involving sensor networks. One such experiment could enable multiple sensors to aggregate data and test different communication methods, while still using Mobile Studio's advanced UI to analyze results. To do this, a sensor board using the parasite connector would have its own processor. The parasite processor could tell the base board's processor to transmit data back to Mobile Studio, or it could arrest all of the WEXP's processor's actions and take complete control.

Conceptually, this allowed for further expandability and improved mechanical strength of the connections, since the connectors could be laid out in a square. However, the space these connectors require is significant. Also, at the point that someone is able to use all of the functionality of parasite mode it would be easier to create another board which had the custom hardware onboard; using the Mobile Studio's communication protocol. Since the majority of sensors would not be using the parasite connector, it was removed and the remaining three connectors were merged into a single twenty-eight pin connector in order to save space. All of the functionality of the primary, expansion, and identifier connectors remained the same.

The last thing to do was to actually find the connector to be used. The connector needed to be small, but still able to interface with a solid core wire. It needed to be 28 pins long. It also needed to have the mechanical strength to hold the base and sensor boards together even if pressure was not applied above the connector. The torque problem was considerable because the connector was not aligned with the center of the board. So, when just handling the device, pressing in the middle could disconnect the two boards. Though this might not seem important, relaxing this constraint caused problems in later revisions of the board. The connector chosen was the DF11 Series 28 pin socket connector from Hirose Electric Co Ltd. With the selection of the connector, the attention turned to the usability of the device in the classroom.

3.2.5 Power Switch

The goal of the Wireless Expansion is not only to create an extendable wireless platform, but to replace the burden of existing systems. In the case of the Physics department this was Logger Pro. In order to accomplish this, the Wireless Expansion needed to alleviate the pitfalls of the existing system. The major complaint is the existing systems are hard to use due to complex rituals in order to get the hardware and software to work together. Since the field of wireless sensing equipment is well developed, the items that set the Wireless Expansion apart are the ability to use the nodes not only as sensors but also as control boards, broadening the usability.

Since the original use for the initial release was as a sensor for Physics I students, the initial usability design started there. The sensors needed to be usable for lab when it was lab time. One of the sacrifices of Wireless Sensors is the loss of power. Originally a simple slide switch was going to be used to turn on and off the device, but this posed one serious usability problem. The device should be smart enough to tell when it is not being used and be able to shut itself off if accidently left on. The last thing that a lab course needed was to have students unable to participate because they forgot to turn off their devices. This problem was compounded by the fact it was not an easy process to just switch out the batteries.

Thus, software-controlled power was implemented. Software-controlled power utilizes the MCU to turn itself off. To turn on the device however, there needs to be powered logic gates in order to be able to turn on the regulator. In later revisions this was refined, but at the time the logic gates protected the processor from high voltages that could be present when powering the device from 12V. In order to power the logic gates, a second regulator needed to be added. Though the current that this second regulator used was in the micro Amps, it was an unavoidable cost of having a software controlled power source. The advantage is that instead of a device being drained over-night by accidently leaving the device on, it would take a device to be off for a month without charging to drain it.

The second regulator picked was the LP2980-ADJ, and it was set to a value of 3.3V. A NOR gate was used to take the push button and the processor's enable pin and control the regulators enable pin. A push button allowed for the software to actually turn off the device as a slide switch would have to be moved back to the original position even if the processor tried to shut itself off. The only problem with the push button was that the sensor board on top would restrict access to it. In order to deal with this the sensor board would have an outline that it could fit into. If a sensor board needed more space it could extend in the other directions but not towards the front of the board. Though this seems prohibiting, there were other systems at the front of the board that needed the air clearance which will be discussed later.

3.2.6 Analog Regulator

In an effort to increase the precision of the Analog to Digital Convertors (ADCs) and the Digital to Analog Converters (DACs), another regulator was added. The MCU had pins for bandgap, and Analog Ground (AGND). AGND could be driven by the MCU or by an external source. Since AGND is dependent on a voltage divider given by VCC it is likely there would be fluctuations in AGND. In order to alleviate this and save money, another LP2980-ADJ regulator was used. Ordering two of the same part saves money in the long run because of bulk discounts. The regulator was set to 1.65V and was powered from the main 3.3V regulator so that the routing of power to the chip was simpler.

3.2.7 RED2 Interface

The next big usability feature is the RED2 interface. original thought was to connect the boards physically in order to connect the board to the RED2. This offered several advantages, the main one having the RED2 serving as a power base station. The thought was to build a daughter card that was permanently connected to the RED2. This daughter card would contain ports which the wireless base would connect to and be housed on. Through this connection, the base would receive new programming, power, and storage. Originally, it was specified to have the ability to house four sensors. This would allow students to contain all of their lab materials in a 3x3x2in^3 area and allow for charging whenever the RED2 was being used. Students could start charging their boards at the beginning of class and be able to use them for an experiment after half an hour. However, since it is more often seen as cumbersome to plug in a device for syncing, etc. these days, a wireless mechanism was added.

The wireless solution implemented was an infrared (IrDA) port which is a standard communications protocol when using infrared communications. The user would point the device at the daughter card on the RED2 and briefly press and release the same button used for switching power on/off to sync. (Holding down this button would turn the device off.) Once synced, the daughter card would then communicate with the RED2 and automatically configure Mobile Studio for the given attachment on the base. The IrDA port would also notify the base of which channel to use. In a multi-user environment where many of these devices are being used at once, each device operates on its own frequency to allow simultaneous communication. If there are not enough open channels available different codes would be given out to allow co-domain wireless communication.

3.2.8 Infrared Sync System

The last thing to mention before describing the details of how the communication is set up is that the hard connection to the RED2 contains UART, I^{2}C, and a single GPIO (General Purpose Input/Output) connection. I^{2}C is another inter-chip communication protocol like SPI except instead of having a pin that selects a slave, an address is sent across the data line to initiate communication. The I^{2}C and GPIO were originally meant to be able to program a processor. The daughter card adds the benefit of removing the need for new firmware programming of the RED2 processor, a Blackfin digital signal processor (DSP), to handle the programming and communication of the connected wireless cards since the code for transmitting UART directly to the desktop software is already present. This requires the daughter card to have its own processor to handle the programming of a WEXP device. Since a processor was already needed and it would be a waste if it were only used for programming one device, the multiple storage method was implemented. Each connector contained a multiplexed UART, I^2C, and GPIO connections, and was designed with the same pin out as the RED2 daughterboard connector. This allows, with some Blackfin programming, the ability to connect the board directly to the RED2 with no daughterboard in-between. This was done to mitigate the risk that the daughterboard's benefits might not make up for the cost and time of development for RevA.

In order to enable this functionality an infrared transceiver and emitter were added along with a UART to IR controller. The UART chip is connected to the same bus as the UART on the connector communicating with the daughterboard. A line is tied from one of the grounds on the connector to the enable of the IR transceiver, thus acting as a shutoff whenever the main board is connected. The UART to IrDA controller required a clock speed of 16 times greater than the UART signal. This put an upper limit of 9600 baud on the speed the UART could function at. This also required the use of another one of the digital blocks to function as a PWM generator for the clock. After pairing and during normal operation this block is released, as the IR receiver is no longer in use. This unfortunately begins to complicate things because dynamically changing blocks adds more code to manage the device. In the end however, the plan is for all of the pins to be reconfigurable which involves more than just switching blocks, but also involves switching the internal routing of pins.

It turned out that the daughterboard was never built. Even though it added the ability to charge and manage multiple sensors as well as provide housing, the average user initially would only carry around one sensor. All of the ideas for the usability stuck with the board as it progressed. It was clear that having a connector to sync the boards together or program them would make them just as cumbersome as existing systems and future revisions have new ways to overcome the problems of syncing, channel division, and programming of the WEXP devices.

3.2.9 Flash

The last part of revision A added in an effort to cover all areas of the vision for the Wireless expansion is onboard flash memory. The minimum a device which is part of a wireless sensor network needs to have is a processor, power, and memory. The wireless expansion had an MCU and power, and the flash memory was added to complete the requirements. This was done in an effort to see if others would be interested in using the WEXP devices in sensor networks, network flow experiments, control algorithms or any other wireless network experiments. The WEXP would allow simulations to be used in a practical arena where things like communication protocols could be tested in actuality. Crossbow's products dominate this arena currently [9,11] and one of the advantages the Crossbow system is that their wireless motes run TinyOS. This allows users who may not be familiar with embedded programming to still use Crossbow's products as event driven machines. One of the thoughts is to port TinyOS onto the WEXP devices in the future. Some other plans are to create LabVIEW modules and MATLAB code which compiles directly into the embedded language of the WEXP devices. This would be the next step in network simulations as LabVIEW is ideally suited for those who may have extensive knowledge in the wireless network field, but may not have much or even any programming experience. TinyOS still requires the knowledge of a low level C style language whereas LabVIEW is a graphical and extremely extensive programming tool. With the hardware for RevA completed the board was then laid out and produced.

3.2.10 Antenna

The Wireless IC uses a printed antenna to communicate on the 2.4 GHz ISM band. Unfortunately for revision A, the wireless communication did not work properly. Building the antenna is extremely frustrating because the documentation has some errors. In the end there were several differences between the documentation, RED2 antenna, and the fabricated antenna.

One of the major differences is that the wireless base uses capacitors and inductors that are of package 0805. The 0805 corresponds to how many mils the component is so, for instance, 0805 is 8 mils by 5 mils. The main reason to do this was simply to make it easier to assemble. The RED2, however, uses parts in a 0603 package. Using the different packages meant that the components used in the RED2, which worked, could not be used. The components chosen matched in value but when the wireless communication failed to function this was one of the primary differences looked at.

A major mistake with the revision A layout was the pad layout for the crystal. The datasheet shows a top view and a bottom view for the pin layout. The bottom view was accidently used leaving the pins reversed leading to an improper connection with the crystal. To get around this, the crystal had to be tilted at a 45 degree angle. An oscilloscope was used to verify the functionality of the crystal. The readings showed a little bit more noise than the crystal on the RED2 but since the wireless communication did not work, the crystal wasthe primary candidate for the malfunction.

The last potential problem needing to be addressed was the distance the components are from the antenna. Though only a few mils away, they are more spread out to make the board easier to assemble. With all of these discrepancies it was impossible to isolate and figure out what contributed to the failure of the antenna. In revision B all of the components are the same as the RED2 and the crystal layout is fixed.

3.3 Revision B

3.3.1 Power

None of the power requirements changed as far as the minimum and maximum values for the input voltage in this revision. One of the major things that did not work out in revision A was the daughterboard connection to the main board. One of the main functions of the daughterboard was the ability to charge the boards. In order to deal with this the original thought was to add the charging circuitry to revision B. The only problem with that plan was that external circuitry needed to be added to protect the charging circuit when powering from an external board.

In order to alleviate this problem, the power is supplied for Revision B through an external board. This offers several advantages for working in the classroom. If a device is accidently left on, the battery could be swapped out for a spare. Since every power board had its own charger, classrooms could have spares for such an occasion. The external power board would contain all of the circuitry necessary for charging the given power source. The main version of the power board contains the same 3.7V iPod nano battery. This power board can be charged from a 5V DC power source or a USB cable.

The other major change for the power system is the power control circuitry. A lot of space was taken up by the pushbutton powered by an external 3.3V Logic regulator. Revision B's main focus is to limit the complexity of the device as much as possible to create a simpler system. With the push button gone, the space above it is also free to be occupied by a sensor board. The majority of the space savings in this revision is from the power system and the board size was able to be reduced from 1"x2" to 1"x1.75". Though this quarter inch does not seem significant it was an obvious enhancement when finally fabricated.

3.3.2 Sensor Connection

The original sensor connector chosen for Revision A was a 2 mil pitch rectangular socket connector. The original thought was to allow easy access to the pins through solid core wire. Unfortunately the connectors did not have a long life when mistreated. If the wrong type of wire is pressed into the socket it can render the connector useless. It also had become obvious that it would be relatively cheap to create a development board which would contain more features that could be used for hardware development or in projects. With a development board being considered, the focus on the connector changed from accessibility to space.

As with revision A, many types of connectors were evaluated for Revision B. The main features present in the connector that are currently implemented in Revision B are that the board-to-board height is small, the pitch between the pins is tiny, and the overhead of the shroud is as small as possible. In Revision B the mechanical stability was not as much of a concern, with the thought that other purely mechanical devices such as screws and spacers could be used to ensure the mechanical stability.

3.3.3 Analog Regulator

In revision B, a capacitor is placed at the analog ground pin. Analog ground in also exposed on the connector. Therefore, an external regulator can be used on a sensor card, if needed. This allows a sensor card to employ a high precision analog ground voltage which may or may not be half of Vcc, which is 3V. If an external regulator is not used, then the capacitor allows the processor to use the analog ground with an external cap. The cap provides lower noise and increased voltage accuracy.

3.3.4 RED2 Interface

A simple connector is used to interface with the RED2 in this revision, instead of building a complex system. A custom connector needs to be fashioned to connect directly to the daughterboard port on the RED2. Though this removes the cost of yet another board, it added the cost of this custom connector, which is easier to fabricate in small quantities. The connector on the RED2 is not uni-directional so this method has the downside that it is possible to accidentally connect the boards backwards.

Another big problem is that the connector for the RED2 is not easily accessible. There is a cover that is over the necessary connector so if the wireless board is to be connected, the RED2 must be disassembled. This was not seen as a big problem, since the connector only provides programming and it would not have to be done very often. Unfortunately if a new firmware version did come out, then the user needed to be able to upgrade the firmware so cables had to be provided to every user.

In order to get around fashioning cables for students, a connector that could be plugged directly into the board was used. The layout for the connector is placed in the upper right corner which helps enforce a uni-directional connection. If the board is plugged in the wrong way it is impeded by other components on the board. It turns out that the RED2 is incapable of using that connector because of the manufacturing process. The RED2's daughterboard connector contains holes in the bottom to allow for a connector to pass through. These holes are filled up during the solder reflow and have to be unplugged with a soldering iron in order to use. This was not discovered until RevB was manufactured.

3.3.5 Antenna

Overall RevB worked successfully for all of the desired features. The wireless communication worked, as well as the ADCs, which allowed a demo to be created which streamed wireless acceleration data. RevB had a few flaws that made it unusable though, namely the difficulties in programming it due to the connector problem. Since the antenna works in this version, the following describes the basics for getting the 2.4Ghz wiggle antenna operational.

The documentation for the wiggle antenna is not correct. This shows up when looking at the bottom of the antenna. The height of the extended ground plane before the ground plane for the whole device is 60 mils. On the top side the height of the extended ground plane is at least 60 mils for the arm, with 30 more mils on top of that for the base of the antenna to come through. The fix for this is to change the length shown on the bottom from 60 to 120 mils.

The components used in revision B of the wireless base are the same components used in the RED2. Though it was never confirmed if this contributed to revision A’s failure, it is highly recommended to use the high frequency caps instead of Cypress’s suggested components for enhanced performance. The inductors that Cypress recommends are incorporated.

3.4 Revision C

Where RevB underwent many cutbacks on features, RevC added a few new ones in order to create something that would be able to last for a while without requiring further revisions. Several major changes occurred: a new processor was chosen, a new power system was designed, a USB chip was added, a second processor was added and another connector was added. The dimensions of RevB to RevC went from 1”x1.75" to 1.25”x1.75”.

3.4.1 Processor

While working with the processor for RevB, trouble arose with the digital blocks. There weren’t enough resources to support the administration functions of internal timers and SPI communication, along with having blocks to spare for things like the

ADCs. The blocks are described in detail later in section 4.2. In order to alleviate this problem, a new processor was chosen which wasn't previously available, the CY8C29666. The CY8C29666 had the same pinout as the RevB processor but contained 16 digital blocks as opposed to eight. The cost in price increased by $2.00 but allowed for the device to have more resources available for different configurations.

3.4.2 Power

The power board in RevB did not work out too well. There were just too many parts that had to be connected together in order to get something functioning. Another concern was with the slide switch. Accidently draining a device of power before an experiment would result in frustration from the Physics department since they would see this as a new problem as opposed to the benefit garnered by switching from a wired sensor to a wireless one. Another concern was that without the detachable power board, it would be hard to use the device in projects with different voltage levels. There was also the problem with charging the device, but that was solved quickly with the addition of a USB device (mentioned in the next section).

In order to alleviate the above concerns, the power system was redesigned once again. A push button is used instead of a slide switch. Instead of having a logic supply, a Zener diode is used to ensure voltage protection on the pins, but allows for no current flow when the device is off. A max1555 charger is in place to charge a single cell lithium ion battery from the USB power. In-between the battery and the rest of the system is a diode. This diode will protect the battery if the voltage supplying the system is greater than the battery. This was done to allow external power to be able to still power the board. The battery is now permanently attached to the base but no current is drawn when the device is off and with the ability to externally power the device without damaging the battery, all goals were able to be achieved.

3.4.3 USB

In an effort to steer towards platform independence and with the need for a power connector on the board because of the new power system, a USB interface was added. The only thing tying mobile studio to Windows for developers is that the current USB driver is only for Windows. With the USB interface added, projects can be built on other platforms which use the Wireless Expansion's capabilities.

3.4.4 Connector

After experimenting with the connector in RevB it was decided to go back to the original plan from RevA and use multiple connectors to insure mechanical stability. Functionally, the connectors are also differentiated. The 19-pin connector contains all of the signal lines as well as the SPI interface to allow for more complete integration with daughter cards. The 9-pin connector contains power and ground along with the status lines of the sensor. The status lines are the same as in RevB: the first indicates the presence of a daughter board, the second indicates if the board is in development mode or not, and the last indicates whether or not there is the one wire ID chip present.

4. Wireless Expansion Firmware

4.1 Methodology

The objective of the wireless expansion is to provide an API that can be used to interface the base with new hardware expansions. In order to accomplish this, the programming ideology is to have the base act as a passive configurable microcontroller. The desktop will configure the MCU on the wireless base to enter the proper mode instead of upgrading the firmware. For example, to set up an analog stream the desktop software would tell the processor on the wireless base to load the chosen ADC resources into the analog blocks. It would then configure the sample rate and set properties of the ADC such as the bit resolution. The desktop software knows that the incoming data is from the ADC in the form requested and will display it properly in a graph or whatever user interface method the particular resource uses.

Expanding this process to a more advanced case allows developers to ultimately control the wireless base completely from configuration code. The wireless base will not have to be constantly receiving instructions in order to operate. Instead, resources will be set up and provided with trigger events. When a trigger event occurs, the given resource will execute its stored process. Though this is not meant to be an OS like the MICAz utalize, it can be used to allow simple configuration from the control application. Like the MICAz's OS, TinyOS, there are a set of resources that can be configured to work together from a set of triggers. A resource is basically any complex function which is available on the base. This consists of different analog and digital blocks as well as any SPI resources on the base.

TinyOS is an event driven language which contains resources as well. Each resource can have callbacks and drive other resources through a similar triggering mechanism. Eventually, it is the hope that the wireless base's firmware could implement anything that TinyOS would implement except in a higher-level language, thus opening the programming to more than just people with embedded programming experience. An example to demonstrate the difference is as follows: both TinyOS and the base's firmware could have configuration code written to read from an analog block every 10 seconds and after the read, save the result to flash. The base's firmware could not be configured to act as a communication host and communicate with several of the wireless nodes without the desktop software. It comes down to the fact that there will not be any decision implementation to start off with. In other words, there will be no way to check to see if an analog read is above a certain threshold and notify the desktop software upon triggering this event. Instead the data will be reported regularly and the desktop software will have to check to see if the data is above the threshold. The desktop software can then tell the base to enter a new configuration.

4.2 Cypress's PSoC

The processor family used in all of the revisions is from Cypress's PSoC line. PSoC stands for Programmable System on Chip. It works a lot like a Complex Programmable Logic Device (CPLD) in which different "blocks" can be configured in order to provide hardware implementations of many signal manipulation functions in both digital and analog. Working with the PSoC for the first time can be difficult to get started and hard to debug. With the addition of unverified hardware it becomes difficult to track down a configuration error vs. bad hardware. It is highly advisable to inspect both the configuration and initialization code of the blocks.

4.2.1 Digital IO

To get started, note that the expansion pins let out on the connector were specifically chosen because many of them can act as analog in or out, along with other digital functions. Any I/O pin can be configured as a digital input, output, interrupt, or connected to a digital block. The input and output aren't straightforward As an input, the pin can be configured as high impedance, pull-up, or pull-down pin. In pull-up or pull-down mode, the pin can also function as an output. Working with just the pure input and output is straightforward. There is a port address which you can read or write to which reads or sets the port. In order to use the pin as an input, you must set the pin to High Z. The other option is Analog High Z, which is actually a misnomer. Putting a pin in Analog High Z turns the pin off and does not function in analog applications. To use the pin as an output, it needs to be configured as either a Strong, or Strong Slow. The only difference is the rising time between the logic levels for Strong Slow takes longer.

4.2.1.1 Port Definitions and Configuration

The last notable thing about the pure input/output modes is that by assigning the pin to something, an address and masks are defined for use in the program. All ports can be accessed through PRTxDR and a specific pin is accessed by bit-masking that address. PRT0DR & 0x01 will perform a read on port 0 pin 0. If port 0 pin 0 was named RedLED in the file PSoCGPIOINT.h, several definitions are made, of which RedLED_Data_ADDR and RedLED_MASK are most important. By including this file the code can access these aliases in order to make changes in future revisions more portable. If a pin is changed, all that has to be done is that the pins on the configuration panel need to be renamed and the code will function the same. This is especially useful when moving around resources that require the manipulation of multiple pins.

The other definitions are important for more advanced usage of the chip. Continuing with the previous example, 3 drive mode masks are defined as GreenLED_DriveMode_x_ADDR, where x is 0 through 2. The drive mode is what configures the pin to be Strong, High Z, Pull-up, etc. Using these drive modes allows reconfiguration of the pins at runtime, which is the key to interfacing with different sensors. In addition to the drive modes, three interrupt control registers are defined as GreenLED_IntCtrl_x_ADDR, for x as 0 and 1, and GreenLED_IntEn_ADDR. The IntCtrl sets the interrupt mode to active high, low, or change from read while the IntEn enables the interrupt. The last definition created is the GreenLED_GlobalSelect_ADDR, which allows the user to enable a connection to the global bus (described later) depending on how the Drive Modes are configured. In order to figure out all of the different combinations, several documents with descriptions are available. By using these registers, any pin can be reconfigured at run time to interface with almost any daughter card's I/O needs. In order to access the daughter card's resources, other types of processing needs to occur which often would require a hardware interface such as an analog-to-digital converter or a filter. The PSoC's digital and analog blocks allow this without the addition of new hardware. But first, one needs to understand the pull-up and pull-down states of the I/O configuration, as this can be confusing at times.

4.2.1.2 Shadow Registers

The confusion of pull-up and pull-down drive modes is from the fact that in these states the pin is operating as an output and as an input. When configuring a device as a pull-up resistor,writing logic high to this pin will result in proper operation of this pin as a pull-up resistor. Reading this pin gives the result of the voltage on the pad. For instance, if a switch is connected to logic low and is activated, even though the pin is being forced logic high, when reading the voltage it will register as logic low. If logic low is written to the pin, then the pin stops functioning as a pull-up and will force logic low. Current will flow through the internal pull-up resistor of 5.6kOhms. This may not seem unexpected, but the problem occurs when using standard bit masking.

For this example, port 0 will have pin 0 configured as an internal pull-up resistor. Connected to the pin is a push switch which connects the pin to ground when activated, and is no-connection when not activated. Port 0 pin 1 will have an LED attached to it with active low logic. The goal is to write a program to turn the LED on when the button is pressed. The following code could be used if a pull-up was not being used.

```
while(1) {
    if(PRT0DR & 0x01) {
      PRT0DR &= ~0x02 // Turn LED On !Causes problems for pull up
    } else {
      PRT0DR |= 0x02 // Turn LED Off !Causes problems for pull up
    }
}
```

If all pins were configured in High Z or Strong mode, this program would work fine. This is because PRT0DR reflects the state of the pin that was set. In the example of the pull-up PRT0DR &= ~0x02 will compile into PRT0DR = PRT0DR & 0x02. This means it will perform a read of the pins before the AND operation. If pin 0 is logic low, a logic low will be written to pin 0 and result in disabling the pull-up. In order to get around this, shadow registers are used in which the shadow registers are modified and keep track of the desired values. This illuminates the read from PRT0DR and isn't too difficult to implement. The only problem is when using other libraries that don't use shadow registers or have a different naming schema, some editing has to be done to make them compatible. Below is the proper code when a pull-up or pull-down pin is present.

```
BYTE PRT0SHADE = 0x03; //LED off Pull Up High
PRT0DR = PRT0SHADE;
while(1) {
    if(PRT0DR & 0x01) {
      PRT0DR = (PRT0SHADE &= ~0x02) // Turn LED On
    } else {
      PRT0DR = (PRT0SHADE |= 0x02) // Turn LED Off
    }
}
```

Since PRT0SHADE now holds the state of the port, there is no accidental overwrite of the desired state because of a read from the current state. With that brief aside, the last thing to mention about the PSoC line is how the blocks work.

4.2.2 Configurable Blocks

These configurable blocks represent different hardware functions the processor can implement and interface with through software. It does this through configuring internal lines to connect up components such as capacitors, amplifiers, and resistors. There are two types of resources: digital and analog. Both of these contain a different interface to the physical pins of the device. In RevC, there are 16 digital blocks and 12 analog blocks.

4.2.2.1 Digital Blocks

When working with the digital blocks, there are a few key concepts to keep track of. The digital blocks represent the hardware which is configurable into the desired function. These include but are not limited to: counters, digital communication, pulse width modulation, and many others. There are 16 of these blocks, but that does not mean it is always possible to use up all of the resources. There are sixteen global out and sixteen global in buses. These buses connect the blocks to the pins on the processor. Some blocks can require the use of multiple pins. Take for instance the SPI block. It requires three pins, MOSI, MISO and SCLK. MOSI and SCLK will take up two lines of the global out bus and MISO will take up one line of the global in bus. It would be impossible to have fifteen pulse width modulated pins in this configuration because there aren't enough lines on the global out bus. Instead, a max of fourteen pulse width modulated pins would be possible leaving one digital block free.

There are also additional restrictions. On top of the sixteen line limit for the global bus, every four blocks has a row bus. The row bus consists of four lines for input and four lines for output. The purpose of the row is to connect the block to the global bus. In the previous example, the SPI block will use up two lines of the output row bus, leaving only two available. In order to use the other three blocks, it is necessary that combined they only require the resources of two output row lines or less and three input row lines or less, as the SPI also takes up one input row.

The last big constraint is that eight of the digital blocks are allocated as communication blocks. These blocks can function with any of the digital resources attached but are the only ones capable of supporting the communication resources. This makes it necessary to allocate all of the communication blocks first and then fill in the other functionality. This can become tricky since analog blocks can have digital blocks associated with them. Placing an integrating ADC requires the use of 3 digital blocks of a counter and a sixteen bit PWM generator. The best way to figure out if it is possible to create a configuration is to sit down with the designer and try working it out. By switching the type of resource, by using a SAR ADC instead of a delta-sigma ADC for instance, it is often possible to find some configuration that will work.

Another interesting capability the digital blocks contain is the broadcast rows. There are four of these, one for each grouping, which can be connected to another broadcast row or a block. The broadcast row can be used as input for any of the digital blocks and even a clock source. This row is a way to get signals from one grouping of blocks to another without using up any of the resources of the global bus.

As far as the allocation for the wireless expansion goes, three blocks are permanently in use. An 8-bit timer, an 8-bit counter, and the SPI master. In debug mode, two more blocks are used for the UART RX and TX. The expansion connector contains four dynamicIO and nine expansionIO. The only difference is the dynamicIO can be analog output as well. This allows for the remaining thirteen pins to each be connected to its own digital block and would maximize the digital resources. As soon as one starts generating mixed signal configurations, the number of digital blocks available can drop dramatically as some analog blocks can take up to 5 digital blocks. The digital blocks are designed to allow the pin-out to change yet still maintain the same functionality as before by rerouting the internal configuration. The analog blocks have a completely different configuration setup which is much less flexible.

4.2.2.2 Analog Blocks

The CY8C29666 has twelve analog blocks. Similar to the digital blocks, the analog blocks are grouped together, but in threes instead of fours. Unlike the digital blocks, these groups are dependent on one another making the design of an analog system much harder. A couple examples on the restrictions for the analog section are: all the blocks in a group share an output bus, all the blocks in a group share the same clock, and there are only four dedicated output pins. There are many more restrictions that are uncovered as configurations are created which aren't always apparent. For example, the configuration editor only allows one variable incremental ADC but it allows 4 twelve-bit incremental ADCs.

Since the restrictions are much more stringent depending on the type of analog block used, the exact maximum specifications are not readily ascertained. Each analog block contains its own set of restriction rules and no general groupings are apparent. On top of the restrictions being hard to determine, the datasheets often contradict themselves. For example, a single or double stage incremental ADC claims in the features to support a 46.8 ksps at 6 bit resolution. This corresponds to 46.8 khz sample rate. Reading the datasheet further reveals that the equation used to get the 46.8 ksps requires the source clock to be set at 24 MHz which is the maximum clock speed at 5V operation. Continuing from there the max sample rate for the 3V configuration is now 23.4 ksps since 12 MHz is the maximum clock speed. Reading further reveals that the maximum clock speed that that can be used is actually 8 MHz due to restrictions on the switched capacitors. This limits the max sample rate to 15.6ksps for a 6 bit ADC. There are many other examples like this for the different blocks so care needs to be taken when designing new configurations.

Given these issues, the following are some guidelines to help determine available configurations. There is a 6-bit successive approximate ADC (SAR) which is capable of sampling at 40 kHz. The SAR works by using an DAC to implement binary search. The DAC is set to half of the value between the min and max known range and a comparator reports weather the actual voltage is higher or lower. The SAR is only available in 6-bits so when trying to get more bits, the sample rate rapidly declines. If needed a higher bit SAR could be built manually using a higher bit DAC at the cost of more analog blocks. At 7-bits an integrator ADC has to be used, dropping the sample rate down to 10.4 kHz and at 14 bits, the speed is down to 121 Hz. The maximum number of ADCs allowed is 4, no matter which ones are chosen. DACs have better performance but the maximum number of bits for a DAC is 9. The 9-bit DAC has a sample rate of 125 ksps and a 6-bit DAC has a sample rate of 250 ksps. The maximum amount of DACs allowed is also 4. This allows for a maximum of 4 ADCs and 4 DACs working simultaneously. In that configuration all of the analog blocks are used since each ADC needs an amplifier and eight digital blocks are used for counters and PWM generators. By changing the types of ADCs the digital block count can be decreased but results in really low resolution ADCs. There are many other types of analog blocks, but the most interesting thing that can be done is the combining of multiple blocks in order to create some function. This could be as simple as a low pass filter or something more complex like a frequency modulator. Keep in mind what makes this system comparable to having external circuitry is that the configuration is actually done in hardware as opposed to software emulation.

Configuring the analog blocks can be more confusing since there are multiple ways the system is designed to accommodate large configuration possibilities. To start with each group of three analog blocks has to share the same clock. This clock can come from several sources: VC1, VC2, Analog Clock Select 1 or Analog Clock Select2. For some reason VC3 is not available and can cause problems when using VC3 to configure the clock rate for the digital portion. If an analog block has a digital portion, the clock in the analog section must match the clock of the digital block. If VC1 and VC2 are in use and a different clock speed is needed then a counter or timer can be configured to act as a clock divisor which can be given to the digital block via the broadcast row, or an output row. On the analog side, the use of Analog Clock Select 1 or 2 gives access to the digital blocks and can be configured to use the output of the counter or timer as the clock. Since there are only 2 clock selects it is impossible to have 4 analog blocks that use 4 different clocks that aren't VC1 or VC2. This may not sound like a problem but often it is the clock that sets things such as the sample rate. When trying to create a programmable sample rate by using a timer by switching the sample rate it will affect all of the clock speeds for all the blocks using that timer.

There are four types of analog blocks: continuous time B and E and switched capacitor C and D [1]. Like the digital block segmentation of generic and communication, different analog blocks support different analog functionality. Each analog block contains one inverter. Switched Capacitor (SC) blocks contain capacitors placed around the amplifier which are controlled from two clocks φ_1 andφ_2. These two clocks allow for many different types of amplifier configurations to be created, resulting in the high degree of configurability that is present. The difference between the type C and type D is the layout of the caps around the amplifier. The Continuous Time (CT) blocks have no capacitors present and only consist of resistors around the amplifier. In the CY8C29666, there are only type B continuous time blocks.

The input to the blocks is done through four multiplexors. Some blocks have the ability to be connected to a pin directly from port 2. Port 2 is not utilized as an analog input since any of the pins on port 0 can be routed to the analog blocks. The MUX is only available as an input to the first block in each column. The first block in each column is a CT block, meaning only amplifiers can use the MUX. From there the output of the amplifier can be distributed to the other blocks. Going back to the decision about port 2, port 2 can feed directly into some of the switch capacitor blocks but this does not free up any resources since basically all of the analog resources besides the amplifiers require SC blocks. The only case this would be useful is when using multiple stage op amps and using up more than one CT block but still needing to feed in input to one of the SC blocks. At that point, the configuration is too application-specific and should be made in external circuitry.

As previously mentioned, there are only four output pins which are directly mapped to each grouping of analog blocks. In order to change the output at a pin, the blocks need to be rearranged to give the proper block access to the output bus. On top of the analog out bus, there is an analog look up table (LUT) bus. Some analog blocks have an output which can be connected to the LUT and then used as input in the digital section. The LUT bus has some logic functionality for its neighboring bus. With the ability to have digital and analog blocks interact in a configurable way it is likely that most any circuit function can be reproduced in some manor by the PSoC.

4.3 Set Backs

There was a great deal of trouble dealing with the RED2 firmware, which turned out to have several issues. Since the RED2 main processor is not a Cypress chip, a custom library had to be written in order to interface with the wireless IC. This custom library was then built upon for the duration of this project and led to many errors that weren't easily reproduced.

The first major issue that occurred is the wireless section would just shut down, with no apparent cause. The firmware would still be responding but the wireless chip could not recover no matter what error code was present. This turned out to be a threading issue. The processor would be writing to the wireless IC and in the middle handle an interrupt. The interrupt had the possibility of trying to send a different command to the wireless IC which then would result in the IC entering random states. In order to fix this, some heavy thread safe code was implemented. This can now start to be cleaned up as some of the other errors have come to light.

The next major issue that seemed impossible to track down was that the device would slowly stop working. For some reason the wireless IC would record a length bigger than the payload size even though the payload was correct. The library would then try to write all of the bytes to the supplied buffer which would result in a buffer overflow. In the Cypress implementation, there are checks to make sure the received length isn't greater than the expected length. In the RED2 firmware, the variable for the length is set, but nothing is done with it to ensure proper operation. Currently this still needs to be fixed. As a temporary workaround, there is a case which checks to make sure the size is within a proper limit and if it is not, it aborts.

4.4 Future Work

The setbacks caused the most visible gap in the firmware completion. Though the methodology is sound and thorough, the implementation does not manage to fully accommodate the planned functionality. A state machine is present for which there are four states: boot, unpaired, paired and physics1. The firmware's breakdown is apparent at the paired state. Instead of waiting for configuration commands, the physics1 state is loaded. To match the methodology there should be no physics1 state, instead there should be an active state which is entered after configuration information is transmitted.

Besides working on the firmware implementation to meet the desired methodology, there are a few more items that could be worked on in the future to enhance the range of applications the wireless expansion could accommodate. In order to fully interface with digital resources on daughter cards through SPI, mini instruction sets need to be created for controlling these at runtime. This will have to be developed in future work and begins the discussion of whether or not development should go into creating something like TinyOS, or perhaps even implementing TinyOS where at runtime configuration code and instruction code is loaded in order to truly alter the capabilities of the base.

The communication mode currently uses 16-byte packets and uses Cypress's transmission protocols. The first thing that could be done to increase data throughput would be to implement streaming mode on the RED2. The streaming mode for RevC is already written by Cypress. In streaming mode, data packets of up to 256 bytes can be sent at a time and would at least double the throughput of the band. Some care does need to be taken, because longer transmission times will be associated with larger data payloads. The chance of a packet being corrupted increases the longer it is in the air and will require far more overhead to retransmit than if a smaller payload packet is lost. Figuring out the optimal balance between speed and overhead given packet losses would be a good candidate for future work.

Cypress advertises a max of 250Kbits/sec when using their wireless protocol in 8 DR mode. Using raw Gaussian Frequency Shift Keying (GFSK) Cypress boasts a 1Mbit/sec data rate link. That means there is a 75% overhead between 8 DR mode and

GFSK. It is likely that spending some time optimizing a custom protocol which runs on the GFSK band would reduce the overhead significantly.

5. Mobile Studio Integration

5.1 Architecture Changes

The architecture for the Mobile Studio desktop software underwent major changes, not just in naming schemas but also in terms of the class hierarchy and functionality. The main highlight is expansion boards of any type are now considered to be devices. An expansion board comes in the form of any utility that uses an I/OBoard as the physical link to Mobile Studio. This can be a daughter card, the wireless base, an expansion board to the wireless base, or any other derivatives which may occur in the future. The other major software activity occurs in representing the resources discussed in the firmware section. Though this is not an architecture change (since it needed to be created anew), in the future it is the hope that this will become an integral part of Mobile Studio. Before going into detail about the changes, a little background may be helpful in understanding what motivated them.

5.1.1 Background

One of the key guiding principles of Mobile Studio is to get the wider educational and hobbyist communities involved by having the ability to create custom panels for running experiments. For instance, it is possible to create a graph which will only display data after an external trigger has occurred. Though this functionality is not shipped with Mobile Studio, a third party can program their own "feature", which is a user interface for showing and interacting with the data to and from the RED2. Currently, development of this particular feature is underway by a student who has not had access to Mobile Studio in the past. Normally this would be extremely difficult for a programmer outside of the development team, but Mobile Studio has an extensive powerful plug-in interface which is able to run different devices, features, and allow even the creation and integration of new hardware.

Mobile Studio functions around five main interface types: a Device, a Device Plug-in, a Feature, a Feature Plug-in, and an IOHost. These five can be simplified down to three as a Plug-in is responsible for creating the type of interface it is. A Device Plug-in is responsible for creating a Device, and a Feature Plug-in is responsible for creating a Feature. Reducing these to three interfaces leaves a Device, a Feature, and a Host which run all of Mobile Studio. An interface is a set of functionality an object of code has to implement. The development team has sole access to the Host interface and does not have plans to release it to be modifiable by the public. Given that stance, the goal is to have as little of the functionality driven by the Host interface as possible, leaving all of the control to the Devices and Features. The Host Interface functions as the glue between the Devices and the Featuresand is visibly the window which contains the buttons to open the different features available for a given device.

A device contains all of the functions a feature calls to get or set data on the board. It is a wrapper for a connection to the computer which translates those function calls into firmware byte commands. In the case of the I/OBoards, the connection is through a USB cable. The flow of communication follows as a Feature or the Host will make a function call to a device. The device will translate that function's goal into a set of commands to communicate with the firmware on the attached board, which is called a physical device and is where Device gets its name. The firmware will respond accordingly and return any results back up to the Device. The Device then parses this into data in a form that the Feature or Host can use. The function then returns with the requested data.

A Feature is the GUI or graphical user interface which displays results and allows user interaction. The Feature is what takes the board's capabilities, and makes them accessible to the users of the system. A Feature is passed a Device when it is created and is able to query the device to see what is supported for that Feature. For example, the Digital IO Feature can ask a device how many digital ports are available. These functions are available through another set of interfaces which simplify the function calls for each Feature. These resource interfaces can be reused or combined by other Features in order to access the resources on the Device. A possible way to structure a system when creating a new Feature would be for each new Feature to have its own resource interface which the Device may implement. Another way would be to have a Feature use several generic resource interfaces. In either case it is up to the developer of the Feature and there is no standard, aside from the basic interfaces to which the Feature must conform.

With the addition of the expansion board, Mobile Studio has gone through two cycles of adding special classes of Devices to the framework. The first iteration of adding expansions onto the base I/OBoards was the physics daughter card. The physics daughterboard was treated as something completely new to Mobile Studio and had its own interfaces to implement. Special control code was written alongside the existing architecture. Almost none of the code was reused for the "sub device" from the storing of the object to the passing of the object to features, even though from the perspective of the user there was no difference. The difference between a Device and a "sub device" is with the communication to the computer. A "sub device" used functions the Device implemented to communicate with the computer. The Device in the case of the RED2 and physics board acted as a UART to USB Bridge. At the time, the way the desktop software was implemented took less time, and since it was not foreseen that more than one type of "sub device" would ever exist, there is no architecture available to allow "sub devices" to use the existing code. During the era of the RED1 and the physics daughter board, the "sub device" was referred to as the daughter board device and would likely not have been changed had not another "sub device" come along.

5.1.2 Sub Devices

The wireless board is the second device to be added to Mobile Studio that is of a class considered a "sub device". This refers to the fact that the "sub device" does not contain a direct connection to a computer. A Device is basically a USB object with a bunch of functions that the Feature calls. The Device translates the function calls into the byte code which is sent to the firmware. The only reason a "sub device" came about was because the Device, for simplicity's sake the RED2, needed to have firmware calls in order to support the Device attached to it. The desktop software went through two revisions which corresponded with RevB and RevC of the hardware.

In RevB the existing sub device code was removed. Instead of classifying RevB as a sub device it was promoted to the status of a Device. The original Device contained many Features which the WEXP-B could never support. Since in the past all of the I/OBoards had similar Features, the method in which the Device was used in the Feature was to call a function which returned an interface with the appropriate functions. If a

Device did not support that Feature it returned null. This is likely one of the main reasons for the sub device split. Instead of having the I/OBoards return null for sub device Features, and sub devices return null for I/OBoard Features, two different interfaces were made. The WEXP-B desktop revision did not combine I/OBoards and WEXP Devices seamlessly, but they were put under the same umbrella. A Device Interface was created which had two subclasses, an I/OBoard Device Interface and a WEXP Device Interface. This began to allow the same code which only needed the shared functionality to handle WEXP Devices as Devices as well. All of the Feature specific methods were moved into the I/OBoardDevice and a new method for adding Feature methods was used for the WEXP devices. Instead of the WEXP DeviceInterface requiring all WEXP Devices to contain implementations for all of the existing Features for any WEXP Device, a WEXP Device implements only the Features it supports by being able to select which interfaces for different Features to inherit. The only additional function that a WEXP device had to implement was pairing.

As far as additions to Mobile Studio go, custom code for RevB was written to handle special Features of the wireless boards. Instead of each device having its own panel with Features there was one panel which populated itself with all of the connected wireless nodes displayed as Features. The given Feature was selected based on what daughter card was connected. To accomplish this, the Host had to differentiate between the I/OBoards and the WEXP Devices, which unfortunately took away from some of the benefit of being able to reuse the code.

Once RevC was finished Mobile Studio underwent one more major architecture change. Instead of having a separate GUI display for the I/OBoards and the Wireless Devices the same mechanism is used. The WEXP Device Interface is removed and a WEXP Device now uses just the standard Device Interface. The I/OBoard Device Interface is still present, but only for backwards compatibility. As the old features are updated, the I/OBoard Device Interface will be phased out.

6. Discussions and Conclusions

6.1 Future Work

There were many setbacks that occurred during the development, which are discussed above. Most of the setbacks were overcome swiftly, but one setback in particular was overcome only in the last few days of the project. This setback turned out to be a problem with a support library, but several weeks were spent trying to understand and fix this problem. This took a toll on the amount of content that could be created to support the wireless devices. Enormous amounts of time went into creating hardware which would alleviate many of the frustrations in both capabilities and usability. The software did not get as much time dedicated to the development of tools and features, which really took away from the WEXP's performance in the students' eyes.

In order to accommodate the desires of the Physics department, time was sacrificed for the main board in order to generate a product that was going to be functional for the Physics department before the end of phase 1. From the perspective of the physics card, the microphone and the range finder need to be given panels in the desktop software, but the firmware needs to be written as described prior in Section 4. Currently the firmware is written specifically for the accelerometer. It would be worth the time to take a step back and rewrite the firmware to require configuration of the different blocks by the desktop software. This way, it will be easier to add on new devices as time goes on.

Also mentioned in the future work of the desktop software, some time needs to go into looking at the structure for Devices, Features, and Resources. Currently, a 3rd party developer would have to create a new Device, and have it implement a new resource for each new feature. This will severely start to fragment the market if multiple developers began creating their own Devices. All of the resources need to be hammered out and implemented for the different Devices. For the wireless expansion, this means some complex firmware in order to have a programmable set of instructions for interacting with things like SPI devices on daughter cards.

The last step that needs to be done is the rest of the chip features need to be enabled. The flash, USB, and programmer chips are not implemented currently. The USB is going to be a major undertaking since software drivers will have to be written for the PC. Focusing on the programmer, will enable the wireless expansion to be released and allow development to continue without fear of isolating users. In the hardware section there was discussion about how to program the device and in the end distributing cables or trying to build connectors wouldn't work for the average user. Enabling the programmer will also require the enabling of the flash leaving only the USB to be implemented in the future. Since the programmer made it acceptable to have users try to update firmware, the advanced configuration and USB updates can be distributed later.

With these three things done, the potential growth for the wireless expansion could boundless. One of the major advances Mobile Studio could leverage is that the wireless expansion could begin to take the system off Windows dependence. There would be no desktop software on other platforms but, it would allow for a programming interface which could even be used in portable devices such as the iPhone or the G1. In addition to adding platforms for developers, Mobile Studio is ready to become a developer tool for not just education-based systems, but for creating new hardware. The wireless expansion allows for the creation of custom wireless devices that will be able to utilize the existing programming interface, allowing the devices to be used in many different programs like MATLAB and LabVIEW.

6.2 Final Thoughts

The success of the Wireless Expansion can be measured by how readily adaptable/adoptable the system becomes. Creating better graphical panels to display data in a clean way, adding some tools to analyze incoming data, and adding features to the analog stream - are all tasks that require the future development of content. This content development can be done by others who are not experts on the system and will be able to be done quickly. The bulk of the work went into flushing out all of the bugs from the existing wireless code in the Red2and constructing an expandable wireless system from scratch. The quality of that system was confirmed when the Imote2 was revisited. The wireless expansion, which was designed independently, has all of the features and much of the same design that Crossbow, a company that has been in the wireless sensor market for years, has developed using a full engineering department – for a significantly higher price.

LITERATURE REFERENCED

1. **M. Basinger.** PSoC Designer™ Device Selection Guide – Revision I. *Cypress MicroSystems, Inc*, January 18, 2007 http://www.psocdeveloper.com/uploads/tx_piapappnote/an2209_03.pdf
2. **J. Stewart.** In-System Serial Programming (ISSP™) Protocol - Revision D. *Cypress MicroSystems, Inc*, March 17, 2004. http://www.psocdeveloper.com/uploads/tx_piapappnote/an2026.pdf
3. **Crossbow Technology, Inc.** Imote2: High-Performance Wireless Sensor Network Node, *Crossbow's Web Site*, http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/Imote2_Datasheet.pdf
4. **Crossbow Technology, Inc.** MICAz: Wireless Measurement System. *Crossbow's Web Site*, http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICAz_Datasheet.pdf
5. **V. Subramanian.** Transport and Link-Level Protocols for Wireless Networks and Extreme Environments. *Rensselaer Polytechnic Institute*, April 2008
6. **J. Coutermarsh.** Mobile Studio: a low-cost, highly portable, PC-based electronic instrumentation suite. *Rensselaer Polytechnic Institute*, December 2007.
7. **M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow.** RFC2018 - TCP Selective Acknowledgment Options, October 1996.
8. **Rensselaer Polytechnic Institute.** The Mobile Studio Project. *Mobile Studio's Website*, August 2008. http://www.mobilestudioproject.com/
9. **Crossbow Technology, Inc.** Customer Reference, *Crossbow's* Website, November 28. http://www.xbow.com/Industry_solutions/CustomerReference.aspx
10. **Cypress Microsystems, Inc.** WirelessUSB™ Antenna Design Layout Guidelines – AN5032, *Cypress's Web Site*, March 30, 2005. http://download.cypress.com.edgesuite.net/design_resources/application_notes/contents/wirelessusb_tm__antenna_design_layout_guidelines___an5032_12.pdf
11. **F. Zhao and L. Guibas,** Wireless Sensor Networks – An Information Processing Approach, *Morgan Kaufmann Publishers*, 2004.