Avtar Khalsa
Cornell University ECE 2008
166 Village Street
Millis, MA, 02054
774-270-4617
ask43@cornell.edu

T&AM 492
4 credits
Swing Leg Proximity Sensor
Cornell Biorobotics and Locomotion Lab
For Professor Andy Ruina
Spring 2007

Part I
Abstract:

My first project was to design a sensor to put on the feet of the Cornell Ranger, so that it would know when the swing leg was about to hit the ground and be able to push off with the stance leg. This, in theory, could yield up to a 400% performance benefit. There were a lot of issues to be dealt with though in trying to find a sensor to do this. Mainly, the sensor needs to work over a short range, and be accurate to within about a half centimeter. The sensor also needed to be quick enough to take measurements many times per step, to get a high enough resolution to be helpful. Finally, the sensor would ideally be unaffected by its surroundings, so that it wouldn't need to be recalibrated all the time.

Introduction:

Pushing off just before heel strike with the stance leg has potential to yield a performance boost of up to a factor of four. This is because it both restores energy and decreases the amount of energy lost in the collision between the swing leg and the ground. [1]. Given this potential benefit, it certainly makes sense to try to develop a sensor which could tell the robot when the swing leg is only a little bit above the ground, and thus allow it to push off at the right moment to maximize performance.

Design Requirements

There are some challenges though in developing something like this. To begin with, finding a sensor which is accurate to within half a centimeter, which is the approximate accuracy required for this is not easy, and they are frequently either very expensive, consume a lot of power, or sometimes both. Furthermore, there is the issue of the sensor being fast enough. A step takes on the order of a second, and ideally it would have, at the very least, a resolution of several hundred samples per step. Thus, it's important to have the sensor sampling at several hundred hertz. Finally, it's useful to have the sensor not be sensitive to its surroundings, for example, color or texture of the surface it's walking over.

Summary

To summarize, the important design considerations to consider are:

1) Accuracy
2) Max Sampling Frequency
3) Insensitive to surroundings

Realistically, it would be very difficult to implement a working sensor that doesn't satisfy all three of these, although its conceivable that a sensor that was sensitive to its surroundings could be recalibrated during each step, but this represents a much larger design problem, and probably should be avoided.

Types of sensors

There are several types of range sensors which could potentially be used for this type of application. The three main types are acoustic, capacitive, and optical. Each of which had some issues.

Acoustic Sensors:
The issue with acoustic sensors is speed. While they are potentially quite accurate, they are limited by the speed of sound. Sound travels at 340.29 meters per second, thus if the sensor is measuring over a distance of approximately 1 meter (the minimum distance the sensor can measure at while maintaining adequate accuracy), the sound wave will need to travel two meters to get to the target and then back to the sensor. This would take approximately .006 seconds. This means the very fastest one of these sensors could go is 170 Hertz. This would require a sensor that was built to only measure over a meter. Practically speaking, none of the acoustic distance sensors on the market are only built to measure over such a small distance. They tend to operate at more like 40 Hz, and can cover a much further distance. Unfortunately 40 Hz, isn't nearly fast enough for this type of application. Thus to make an acoustic sensor, we would probably need to custom design one to go much faster than the normal ones on the market, and it still would be somewhat on the slow side. The final issue with acoustic sensors is that they tend to consume a quite a bit of power, which could also be an issue.

Capacitive Sensors:
Capacitive sensors have a completely different set of issues. First and foremost, they were very sensitive to their surroundings. It would not be feasible to make a sensor which could just be left alone in the code. It would constantly need to be recalibrated for the surroundings. Additionally, they all appeared to be quite complicated circuits and would take a long time to implement. Many of the capacitive sensors needed an extremely high voltage to operate which could be an issue on a robot that's designed to operate at low power. Finally, most of the capacitive sensors on the market seemed to be quite expensive.

Optical Sensors
Optical sensors seemed fairly well suited for the application, but did have some issues as well. To begin with, the sensor would need to be modulated in some capacity, which complicates the circuit for the receiver because it would need to filter out all light that’s not from the emitter. This can be done in several ways. To begin with, a simple optical filter could screen out almost all light that isn't at the right frequency. Additionally, the circuit could be set up as explained below to remove all light that isn't further modulated at the right frequency. At first glance it seems like an optical sensor will still be somewhat vulnerable to differences in the reflectivity of the surface below it. This makes it impossible to calibrate the microcontroller to determine distance based specific light intensities. However, we can still use relative light intensities to determine distance as will be explained in the modulation section below.

Summary:

	Advantages	Disadvantages
Accoustic	Accurate	Too much power, designed for longer distances
Capacitive	Accurate enough when properly tuned	Too much power, very sensitive to environment
Optical	Cheap, and accurate	Needs to be modulated and custom designed

In spite of the issues that seemed inherent to an optical sensor, we decided it was in fact the simplest solution, and that its inherent problems were probably the easiest to overcome.

Method and Results:

Sharp Sensor

The first sensor we tried was a Sharp Infrared Proximity sensor. A picture can be seen in Figure 1.

Fig 1
Sharp Proximity Sensor

The data sheet for this sensor indicated that it had the characteristic curves seen in Figure 2.

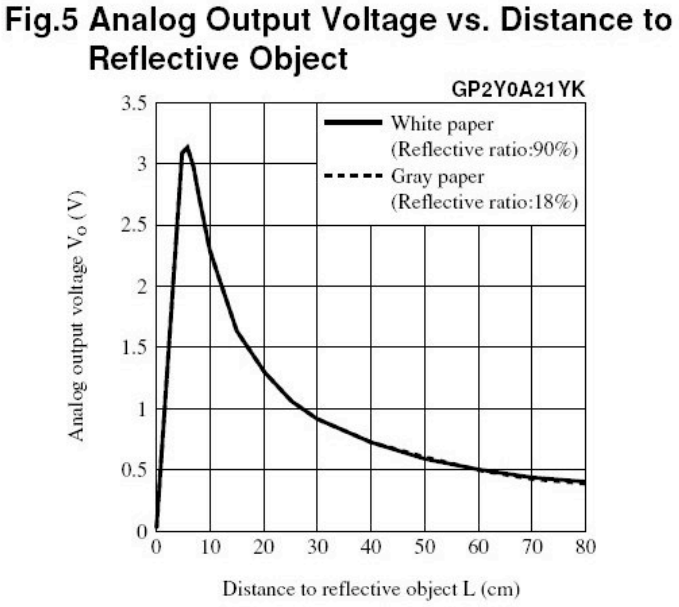

Fig 2
Theoretical Voltage vs Distance curve

Figure 2 indicates that the sensor should have been insensitive to the reflectivity of the surface it was aimed at, in addition to yielding an accuracy level that would be adequate for the application. Unfortunately the graph proved to be a poor picture of the sensor's actual output. It was neither insensitive to surface reflectivity nor was it particularly accurate or consistent. To begin with, in two trials using a piece of metal as the object being detected, and the using the milling machine to keep a very accurate track of distance, the sensor yielded the output curves shown in Figure 3

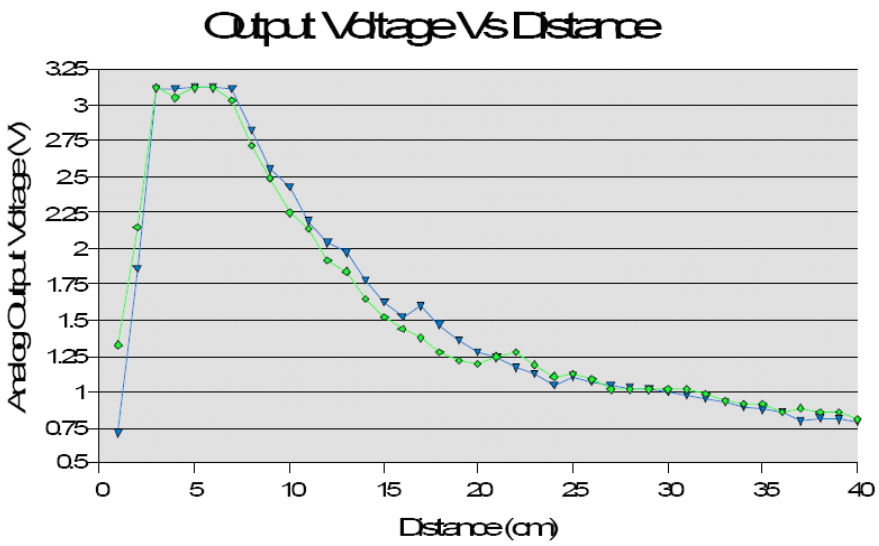

Fig 3
Actual Voltage vs Distance Curve

In Figure 3, it is clear that the sensor output is not particularly consistent. Additionally, by switching the piece of metal with an even more reflective orange piece of metal, the output voltage at 40 cm jumped all the way to 1.02 volts. Unfortunately, this means that a voltage reading of 1.02 volts could represent anything from 40cm distance (with an extremely reflective orange surface) to 30cm (with a simple metal surface shown in Fig 3 above). Consequently, this sensor was clearly not feasible for use.

It's worth noting that the sensor did screen out ambient light effectively. It did this with a simple optical filter. This filter was effective enough to keep the output constant at a given distance with the lights in the room either on or off.

Custom Sensor

The issues this sensor had really highlighted the problem with using a simple optical light intensity sensor. The reflectivity of the surface will always change the intensity the receiver detects. In order to account for this, we decided to make our own optical sensor which would hopefully be indifferent to the reflectivity of the surface below it. The sensor would be operated by a small "satellite" microcontroller, which could then inform the main network brain when the heel was the appropriate distance above the ground. The design we eventually came up with involved a detector with a very narrow detection angle. The idea is shown in Figure 4

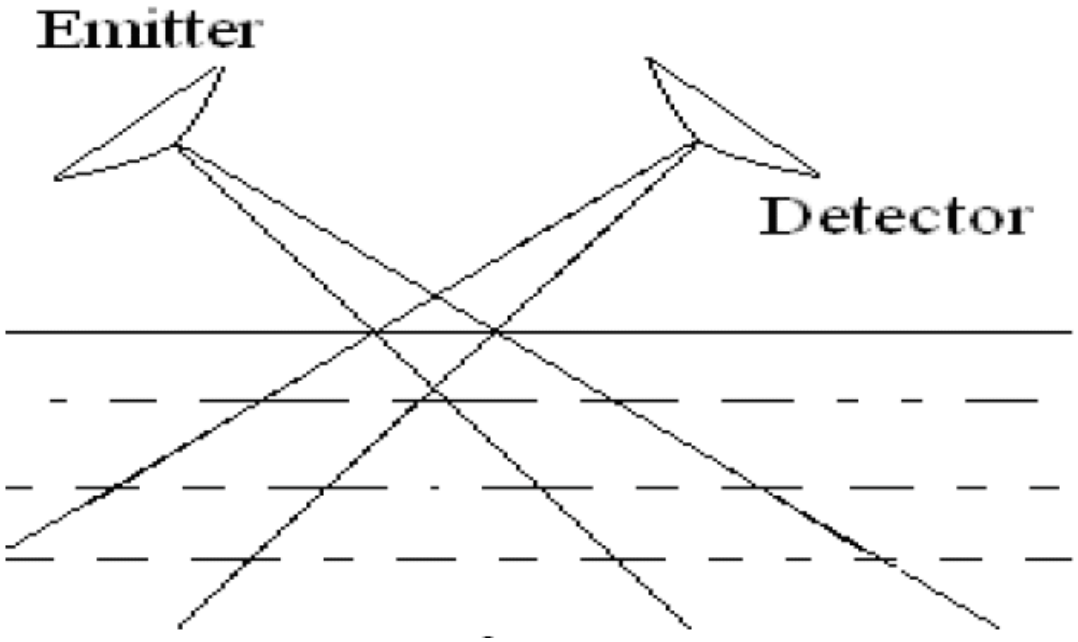

Fig 4
Theoretical Sensor Design

In the image, the dashed lines represent the ground. When the ground isn't at the right vertical distance from the sensor, the emitter will be shining to a different part of the surface than the detector is looking at. Thus as the surface moves towards the sensor, the analog output curve will be bell shaped, as shown in Figure 5

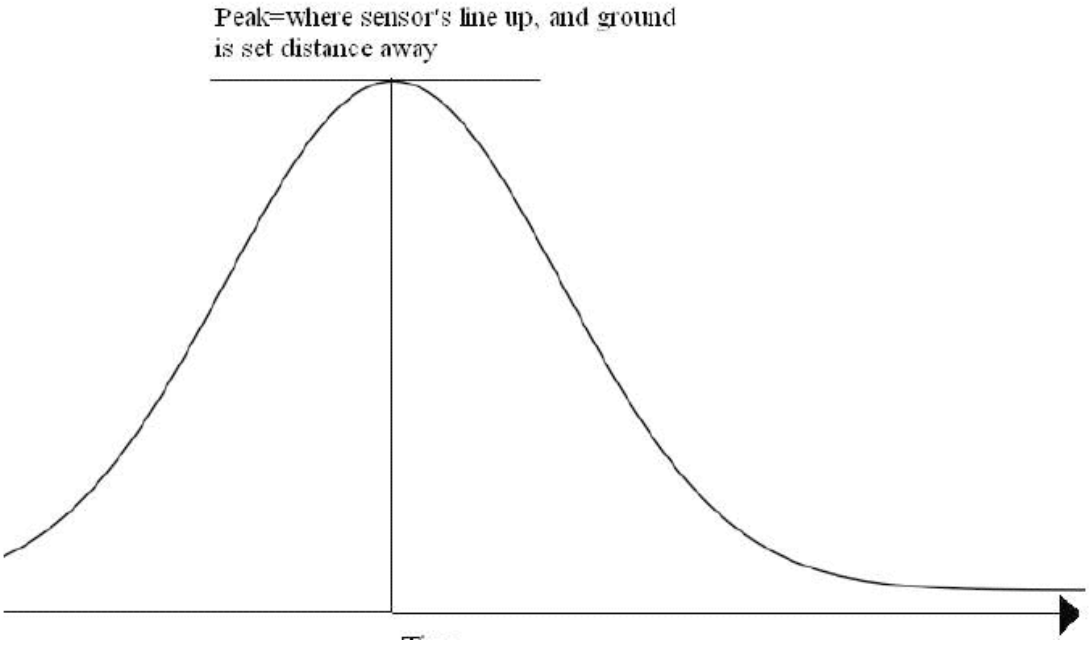

Fig 5
Theoretical Sensor Output

The peak of the bell will always be at the distance from the sensor where the receiving beam and the emitted beam line up. Thus regardless of the reflectivity of the surface, it doesn't change where the peak would be. It will always occur when the surface is at the distance from the sensor where the beam from the emitter exactly lines up with the spot the sensor is looking at. The actual amplitude of the curve will be dependent on the reflectivity of the surface, but as long as a microcontroller is controlling the sensor, it can just be monitoring for a peak.

Slightly Bigger Picture

A simple block diagram relating how the sensor communicates with the robot itself is shown in Figure 6 below.

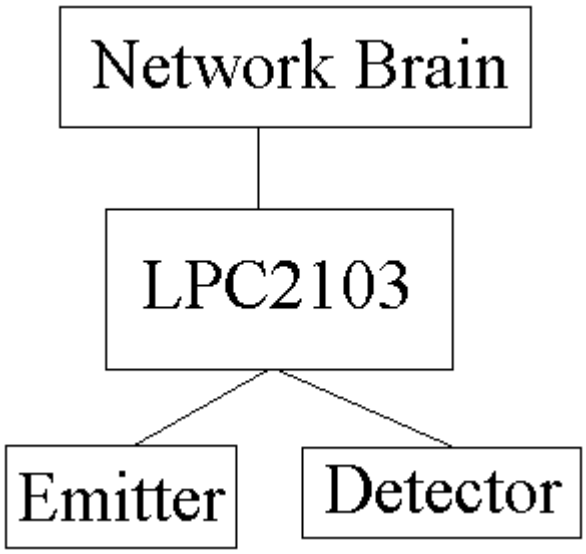

Figure 6
Simple block diagram of how the sensor communicates with the network brain

Modulation

Another check that we decided to add was to modulate the signal. Ideally, the light source would be modulated at about 3 kHz, to allow for many samples per step. 3KHz, is ideal because it is the minimum speed that still yields an almost clean simulation assuming noise at 60Hz. This is demonstrated in the example below, in as much as the noise signal has a frequency $1/50^{th}$ of the light modulation frequency. The detector should be receiving a modulated version of the signal shown in figure 5. This would look something like figure 7.

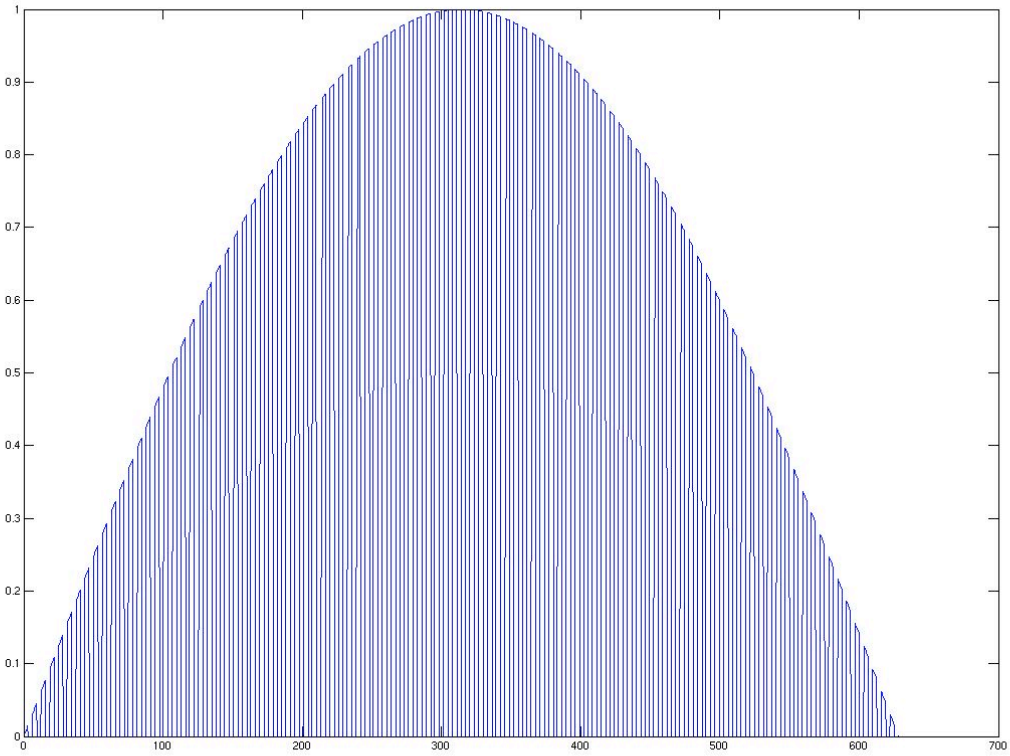

Figure 7 modulated signal received by detector

The goal is to reconstruct the image from figure 5, except with all the external noise weeded out. We do this by taking a sample over a full period of the emitter and comparing the intensity the detector receives when the emitter is on, versus when it is off. This gives us a baseline for external noise for every single sample. The microcontroller can then subtract these two values to hopefully generate a signal like the one in figure 5. The advantage of using this reconstruction is that it should completely remove noise. For example, let's say there is an extreme case with external noise shown in Figure 8 below.

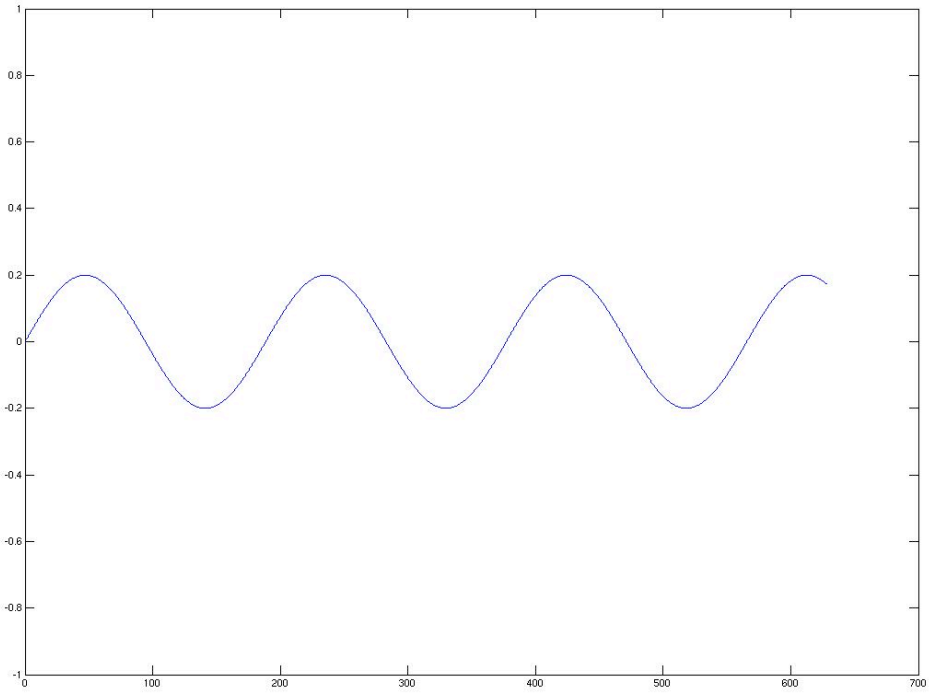

Figure 8: noise signal

The detector would receive the sum of figure 8 and figure 7 which is shown in figure 9.

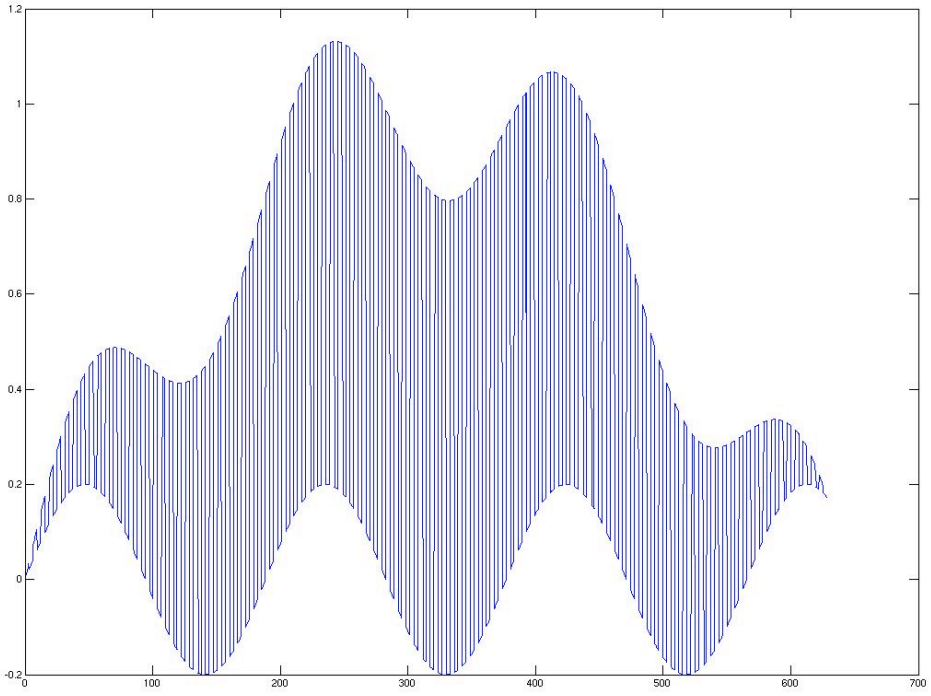

Figure 9: Signal received by detector including noise

While the signal from figure 9 looks like a mess, using the values from when the emitter is off as a baseline for each signal, allows the microcontroller to construct the signal shown in figure 10.

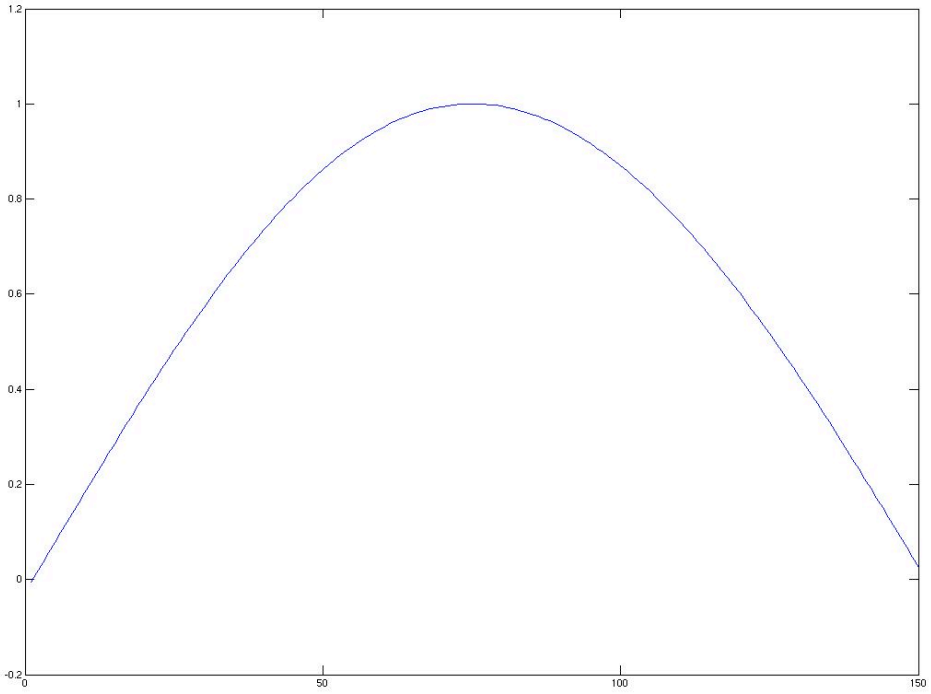

Figure 10: Decoded Signal

While figure 10 isn't quite a perfect noiseless signal, it is pretty close. More importantly, it is much better than the signal we would have received by just looking at the input and neglecting to use a baseline value. Given the same noise, and the same signal, but without the decoding scheme, we would have received the signal shown in Figure 11 below.

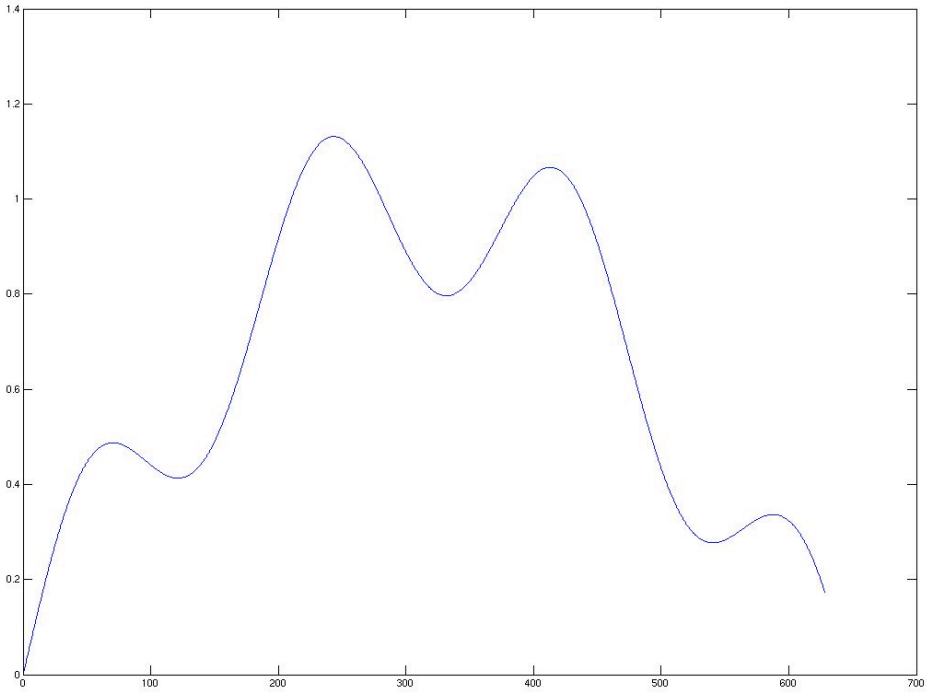

Figure 11: Received Signal with no Modulation

The decoding scheme, while not perfect, should give us much more tolerance for external noise. It does operate under the assumption that the ambient light is constant over the period of the modulated source. In this case that would be about .0003 seconds. So as long as the ambient light is fairly steady over that time, the microcontroller will completely remove it and the intensity will be just that from the emitter. A cost of this scheme is that we will actually have half the sample resolution we would otherwise have because every other sample is used to establish a baseline for the following sample. Thus, the emitter and the detector will need to be oscillating and making readings respectively at twice the frequency at which we actually want to collect data.

Assumptions of the scheme

One assumption that this scheme relies upon is that the photo transistor is linear in nature. If it isn't (which is actually, very likely,) it somewhat complicates the situation. It should however be resolvable using the microcontroller. If we take the current output from the transistor when the emitter is off, and use this as a baseline measurement, then, assuming we know the transistor's input light intensity to output current curve, we can still determine the exact intensity of the light due to the emitter.

Experiments

In order to implement this sensor, we bought separate emitters and detectors which were set to operate at about the same wavelength. In order to test these though, we simply lined them up across from each other, and pulsed the emitter at various speeds. The results from the receiver at 10Hz, 100Hz and 1kHz are shown in Figure 12, 13 and 14 respectively.

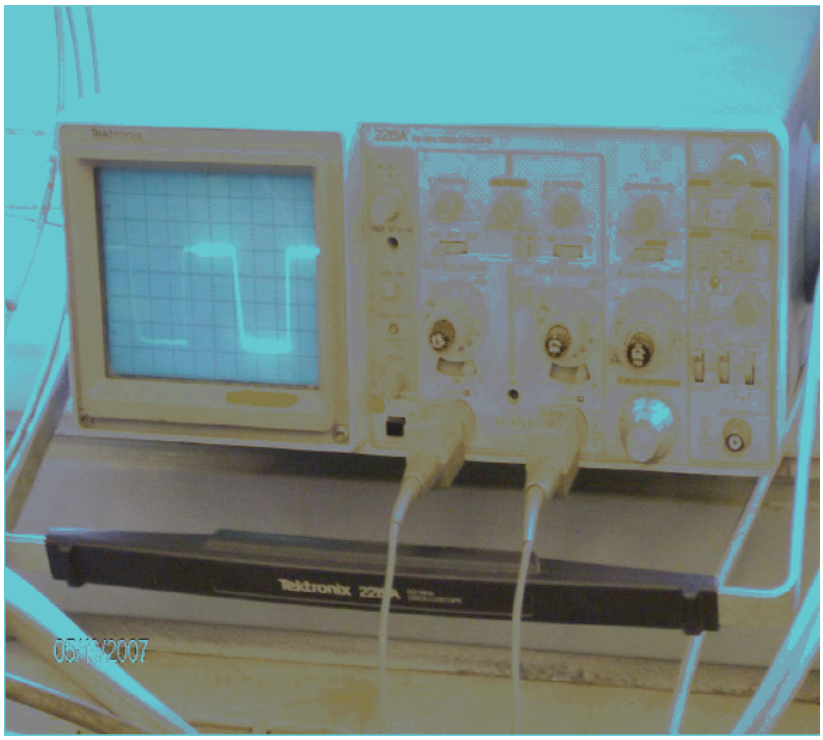

Figure 12
Output of detector at 10Hz

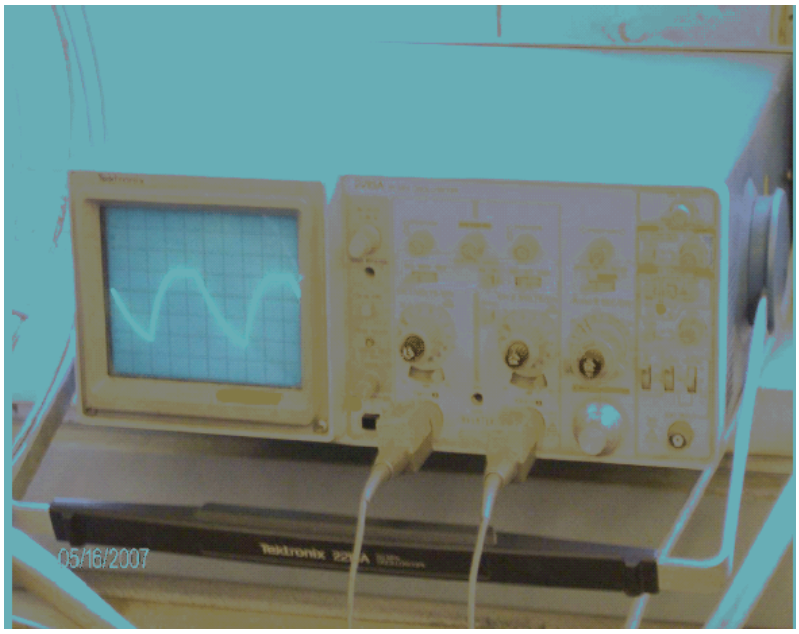

Figure 13

Output of detector at 100Hz

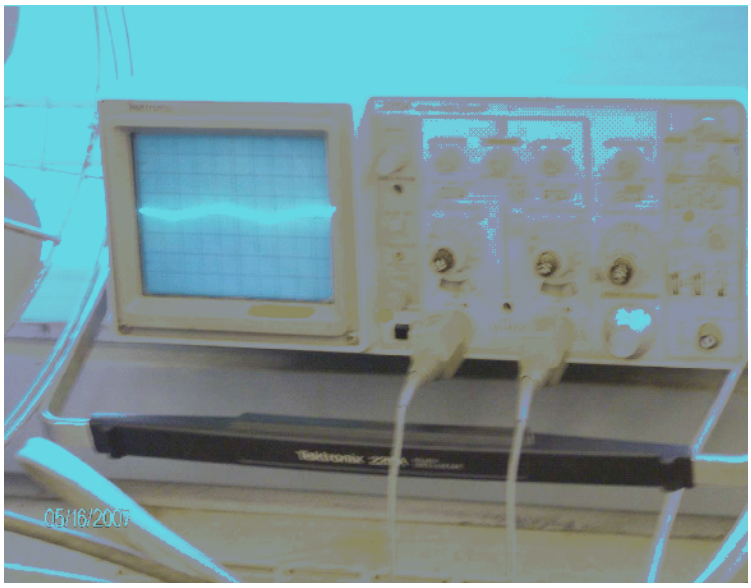

Figure 14
Output of detector at 1kHz

The important thing to note is that as the frequency went up, either the detector or the emitter stopped being able to respond correctly in time. This is clearly shown by the amplitude going down on the oscilloscope as the frequency goes up. If they were both operating properly, the amplitude should have stayed the same.

Discussion

It is very difficult to determine whether or not it was the emitter or the detector that wasn't working correctly at high speeds. However, it seems clear that this is a problem of capacitance. In Figure 12, if you look closely, you can see that the transitions do not have sharp edges, but rather curve just like a charging and discharging capacitor. Since we know that the emitter has additional circuit elements like resistors etc, it seems like the logical place to start. The same company that makes the emitter we bought makes an emitter specifically rated to oscillate at very high speeds. The one we purchased did not actually have a frequency rating. The first step in solving this problem is to purchase these faster emitters and try them instead.

If this still doesn't fix the problem, its possible that the detector we bought was simply not fast enough, in which case, we will need to buy a faster detector. The three main considerations that need to be taken into account when choosing a detector are that it must be fast enough, it must have a fairly narrow viewing radius, and of course that it detects over the same frequency as the emitter. The detector does not need to have a narrow band though, as we already have optical filters to block out light at other frequencies.

Additionally, while experimenting with the sensors, I did see enough evidence to merit looking more closely into whether or not the detector would actually be able to detect any difference in the light when the emitter wasn't aimed right at it. The changes in the voltage output of the detector were small enough to warrant perhaps looking a little more closely into the overall feasibility of the design before any kind of significant time is spent trying to get it to work.

Conclusion

While I was unable get any concrete results with the sensor, I think that some important progress was made. To begin with, we have a preliminary design, which should be workable one way or another. Additionally we weeded through an obvious solution which turned out to not work, in the Sharp proximity sensor. On a whole, I think the semester was spent going through a lot of the necessary trial and error that is inherent in any project, and hopefully will save either myself, or whoever else takes the project over a healthy amount of time going through solutions that don't work.

Part II

Introduction:

The second goal that I worked on over the course of the semester was looking into getting the satellite microcontrollers up and running in conjunction with Sam Lee's project. The main idea for his project was to have many small microcontrollers operating each of the sensors on the robots. The satellite microcontrollers then communicate with the network brain through a hierarchy of other microcontrollers. I was trying to pick out a specific model of microcontroller and figure out exactly what goes into programming it, in addition to hopefully being able to use it with the sensor I tried to implement.

The main priorities that we had for these microcontrollers were that they needed an ADC with as many bits as possible, and consumed low power. Typically 10 bits would be reasonable, and 12 would be excellent for the ADC. The number of bits the ADC has is easy to determine since it's just stated in the data sheet for the microcontroller. Power consumption on the other hand can be a somewhat difficult value to calculate without directly measuring it, as the microcontroller is running. This is because power consumed is very dependent on what operations the microcontroller is doing, and how often.

The other major goal of this part of the project was to investigate any pitfalls that might come up when trying to implement various different microcontrollers. As much as it’s nice to say that we can just plug many different microcontrollers into a computer, and just be able to use them easily, it’s not always realistic. Every microcontroller takes time to get used to programming with. There is a lot which needs to be experimented with to get it all working correctly, and understand exactly how it needs to be programmed.

Unfortunately, during the course of this experimentation, I was unable to make that much progress, although hopefully I uncovered quite a few pitfalls that whoever is working with these next will be able to avoid.

Method and Results:

Background on Microcontroller Selection

The first microcontroller we tried was the LPC2131, from New Micros. There were quite a few reasons for this. To begin with, it was extremely inexpensive, costing just $30. It had fairly low power consumption, although it's impossible to say exactly what the power consumption is without actually running it with the correct code and measuring. Additionally, we were under the impression that Sam had successfully used a similar one, and thus assumed that getting it up and running would be fairly straightforward. Finally, it was an extremely small board. The board we were using wasn't a full development kit; just a little header board holding only the microcontroller itself and a few connecting ports. The exact dimensions were 1.0"(W) x 1.3"(L) x 0.4"(H).

New Micros LPC2129

Unfortunately, the problems began before we even got the microcontroller. Since we had an extra LPC2129 from New Micros in the lab anyway, I tried programming it initially to get comfortable with the process in general. We weren't using these boards because they were $89 as opposed to the $30 for the 2131. Unfortunately, this microcontroller blew out the first time we tried plugging it into the power supply. We still don't definitively know what happened in this case. According to the schematic, which is shown in Figure 15 below, we should have plugged power into pin 2 and ground into pin 4 (highlighted by the red and black boxes respectively for clarity) of port J1.

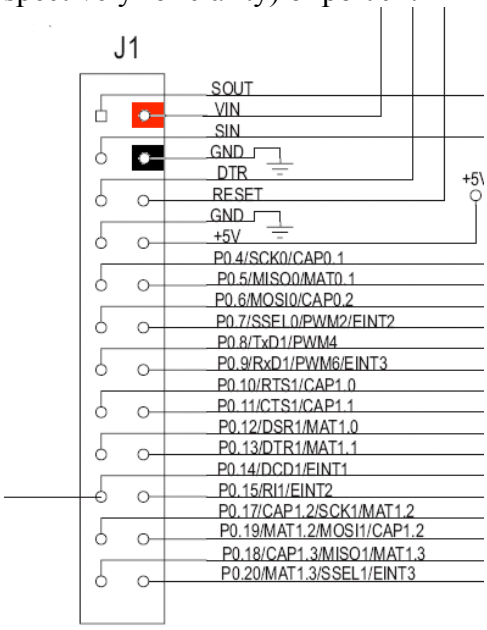

Figure 15
J1 port on LPC2129

The full schematic is shown in the Appendix C below. It's conceivable that I plugged it in backwards, which certainly would have blown out the microcontroller. However, before actually plugging it in, I showed the schematic to Sam, and we discussed that plugging it in backwards would blow out the processor, and checked it repeatedly to make sure that this wasn't the case. In light of this, it seems very unlikely that we still plugged it in backwards. If this indeed was not the case, the only possibility remaining is a hardware malfunction. Since that specific processor had not been tested before, it does seem plausible that it was a defective board, although there is really no way to prove it.

Fortunately, there was a second 2129 processor that wasn't being used, and apparently had been tested before. When I plugged it in the same way the lights came on, and it seemed to function normally. Unfortunately, when I tried plugging it into my computer, not only did the IAR workbench not connect to it, after a few minutes, the LEDs turned out, and the microcontroller seemed to be off. This was especially strange, because I hadn't changed any of the wiring when it stopped, it just seemed random. After some experimenting, we found the problem to be a lose connection on one of the voltage regulator. As can be seen in Figure 16, the LPC 2129, has two voltage regulators that stick up some distance above the board.

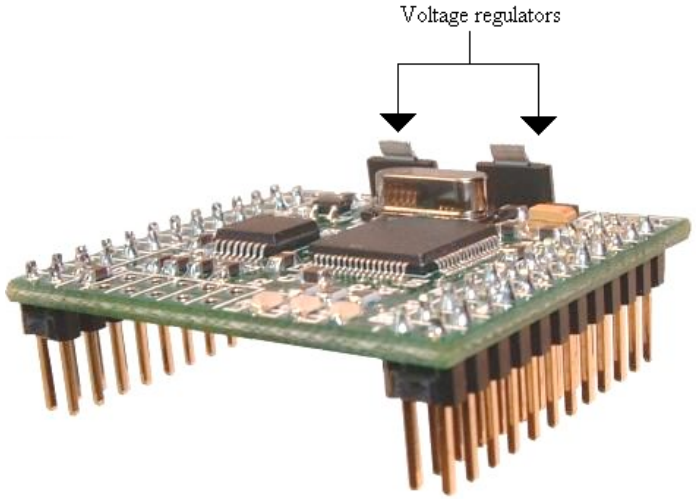

Figure 16
Voltage Regulators are clearly the part that sticks furthest off the board

This is unfortunately a very poor design considering that the regulators only have a couple of flimsy connectors to the board and they don't appear to be able to tolerate any pressure without getting loose. The voltage regulator had come a bit loose and this was preventing to board from functioning. After Jason was helpful enough to re-solder the tiny connections, the board started working again. In spite of this apparent progress, the board still wouldn't program. The specific error I was getting was that the workbench didn't recognize the LPC2129 on the JTAG chain.

New Micros LPC2131

At this point, the LPC2131's came and I decided to switch over to using those, since they were the ones I was going to be using anyway. They did immediately turn on without

having any malfunctions with the voltage regulators which seemed like a good start. The power connection was identical to that shown in Figure 3.2, although the full schematic was quite different. It can be found in the appendix section as well.

Unfortunately, this was about as far as we managed to get with the LPC2131, the rest of the experience was just a frustrating lesson in what to avoid with these satellite microcontrollers. The first issue was the connector from the LPC2131 to the JTAG. Sam had previously made one for the 2129, but it was impossible to assume that it would be the same. Thus, when IAR workbench didn't recognize the 2131 as being connected to the JTAG, even though they were both clearly on, that was the first thing I checked. The New Micros site had the following Pin Out shown in Table 1 for the JTAG connection on the board.

J2 JTAG Connector

Pin	Signal
1	+3.3V
2	GND
3	TDI/P1.28
4	RTCK/P1.26
5	TDO/P1.27
6	TMS/P1.30
7	TCK/P1.29
8	P0.8
9	RESET'
10	TRST/P1.31

Table 1
J2 Connector LPC 2131

Additionally, the JTAG had the Pin Out Information shown in Table 2.

JTAG Pin Out

Pin	Signal
1	VTref
2	Vsupply
3	nTRST
4	GND
5	TDI
6	GND
7	TMS

8	GND
9	TCK
10	GND
11	RTCK
12	GND
13	TDO
14	GND
15	RESET
16	GND
17	DBGRQ
18	GND
19	DBGACK
20	GND

Table 2
JTAG Pin Out

Looking at these two Pin Outs, it was pretty clear that the connector needed to tie the pins shown in Table 3 together.

2131 Connector	JTAG Pin Out
1	1
2	4
3	5
4	11
5	13
6	7
7	9
8	leave hanging
9	15
10	3

Table 3
Connections for JTAG to 2131

It was fairly trivial although a bit time consuming to actually make this connector once I had all the connections figured out. Unfortunately, when I connected the 2131 to the JTAG with this connector, I still got the same error. I tried making the connector a second time in case I was reversing pins without realizing it, or somehow leaving something unconnected, but it still didn't work.

As something of a last ditch effort, I emailed New Micros and IAR to see if they had any insight into the problem. New Micros did seem to have some insight pointing out that we needed pull-up and pull-down resistors in several of the connections. The only place that this was actually mentioned was on the ARM website. There was no mention of it anywhere on the new micros website, or on the JTAG website. The Arm Website clearly lists exactly which pins on the LPC2131 would need a pull-up resistor and which would need a pull-down resistor. These are shown in Table 4 below.

(On my computer at home, will add before final version)

Table 4

Unfortunately, this meant I was going to need to make yet another connector in the hopes of making this work. This connector was actually much more complicated, because it was going to need pull-up and pull-down resistors along it. The design I eventually settled on was to use a small protoboard with a 10 pin male connector on one side and a 20 pin male connector on the other side. This way, I could just plug the JTAG in on one side, and the connection from the 2131 on the other side. Then using the wire wrap tool, I could connect the appropriate pins both to each other and to the pull-up and pull-down resistors that were located on the side of the board. This design seemed to remove almost all the strain on any of the parts in the board, so that it hopefully wouldn't break.

With this finished, and all the connections double checked with the multimeter, I connected it to the computer again, and it still didn't detect that it was there. While checking everything again, I noticed the status lights go out while the board was still plugged in. After checking all the connections, I found that the same poorly designed voltage regulator as in the 2129 had come loose. This was very clearly the issue, because the board could be plugged in and the status lights would be out, but if you merely nudged it, the lights would come back on.

Olimex LPC 2103

Finally, we decided to give up on the New Micros board, and try a different processor. We went with the Olimex LPC 2103. Physically it was very similar to the New Micros LPC2131, with one major difference. It had a devoted connection for JTAGs with the pull-up and pull-down resistors built right in. Aside from that though, it had 10 bit ADCs, was 2.0"x0.75" (about the same size as the other). The power consumption also appeared to be similar to the LPC 2131, based on the worst case rating for the voltage regulator. Beyond that though, it's very difficult to tell how much power it would consume without actually running.

Thanks to the new board, complete with its JTAG connection, it was programmed successfully the first time I tried in lab, which is what we had been hoping for on the other board. Furthermore, I managed to get the status LED to blink after playing with the code for a day or two. It was a little challenging to do this because it didn't come with any simple example code. All the example code from IAR was designed for use with a full development board, and performed fairly complex operations .While this may be useful in the future, it makes picking up how the processor works a bit more challenging. The easiest program I found involved several timers and setting off a couple of LEDs which would be connected to the IO pins on the development board. This "Simple" program, actually involved 8 separate files, with many different class definitions. Most of these were irrelevant, and just needed to be ignored. The status LED was connected to pin 26, and only required a few modifications to the code. They are summarized in the appendix below.

Discussion

While on the surface it doesn't appear like I made very much progress with the microcontrollers, I do feel like a learned several valuable lessons, which there was probably no easy way to find out without spending this kind of time making all these mistakes. First and foremost, if it isn't clear enough from the discussion above, I don't think we should buy microcontrollers from New Micros any more. While their parts are quite inexpensive, you really are getting what you pay for, as they appear to be very poorly designed. This is evident from how they consistently broke without being put through much of any abuse. I tried to never handle them without the antistatic band on, and certainly static discharge does not seem to be the culprit in this case, as it couldn't cause working parts to come lose. Additionally, I was not the only one who had problems with their boards, as Jason also ordered several that simply came defective. In light of all this evidence, it seems reasonable to just remove them from our list of potential suppliers.

I also found that getting microcontrollers not explicitly designed to work with JTAG's is significantly more difficult than we expected. Namely, we didn't know about the pull-up and pull-down resistors that New Micros deemed so obvious that they didn't even bother mentioning them in any of the documentation. In the future there are two very easy ways to avoid this problem. The first, and easiest, is to always buy boards with dedicated JTAG 20 pin connectors, and avoid having to make our own. If this is not feasible for some reason though, then it is very important that the user looks at the circuit on the board where they are connecting the JTAG. In Figure 17, you can see the part of the schematic for the LPC2131 where it should connect to the JTAG.

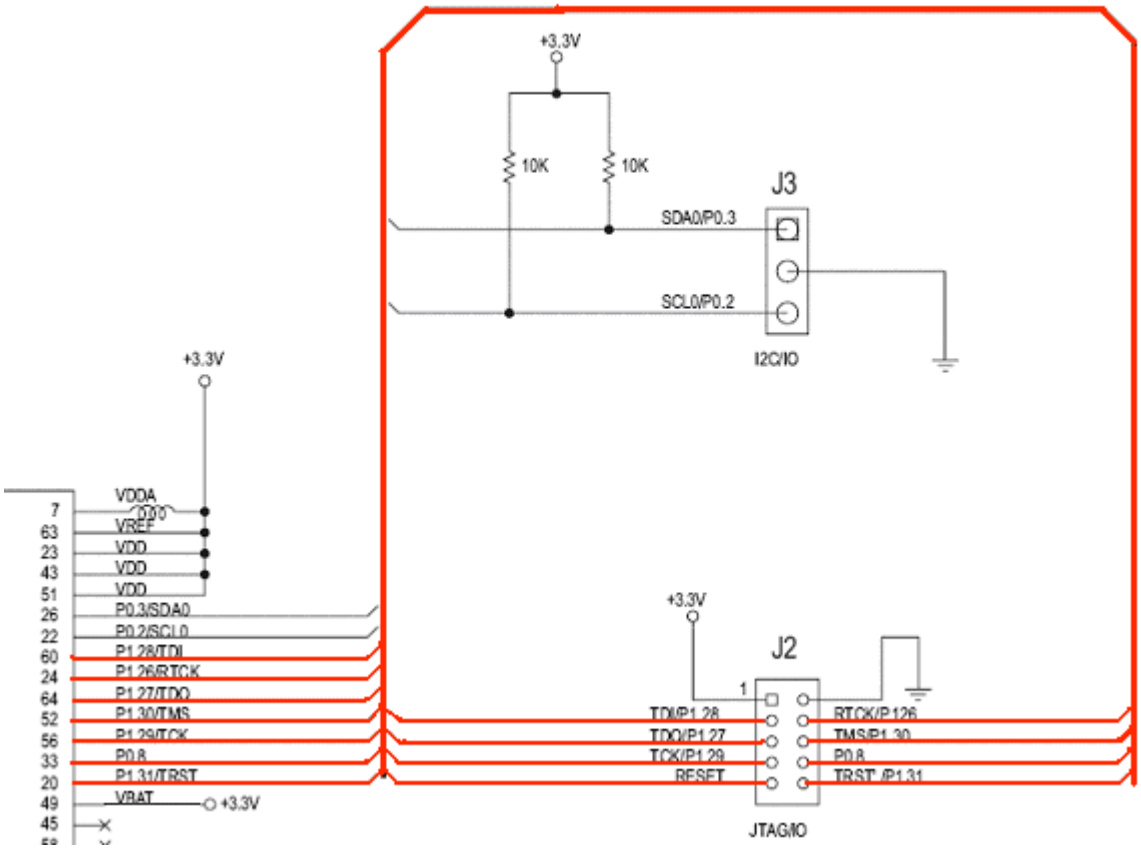

Fig 17 LPC2131 schematic

What is important to note is that the connectors go directly from J2 to the 2131, and are effectively hanging when not connected to anything. These connections are highlighted in red. These missing resistors cause those pins to float, even when connected to the JTAG. This prevents it from properly communicating with the 2131. On the other hand, the 2103 has these pull-up resistors built right into the circuitry of the board, as can be seen in Figure 18 below.

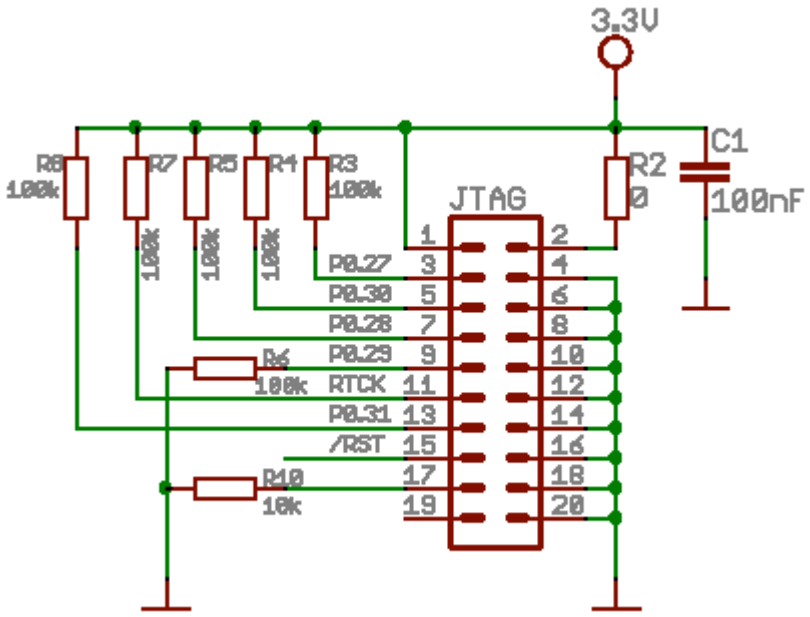

Fig 18 LPC2103 schematic

The difference in this schematic is clear. Almost every pin here is connected with either a 100k resistor or a 10k to either VDD or ground. With all of this built in, it made the process of connecting to the JTAG much easier. This raises the question: “why is it built into one and not the other?”. The answer is that the J2 connector on the LPC2131 is not purely designed for connections to a JTAG, and consequently there might be external circuits which the resistors would interfere with. The JTAG connector on the LPC2103 on the other hand is designed purely with the JTAG in mind, so there is no reason to leave the resistors out.

Conclusions

While it certainly doesn’t look like there was much progress made during the course of the semester, I do think the mistakes I made and problems that I ran into can still prove to be useful down the road. We were eventually going to find out that New Micros boards were very poorly designed and certainly after this whole semester, it seems to me like the case against them is fairly clear, and we won’t have to trouble any other students with their boards. Additionally, someone was eventually going to have to figure out the importance of the pull-up and pull-down resistors in any board that doesn’t have a dedicated JTAG connector. It took a bit longer to reach this conclusion thanks to the trouble with the boards themselves, and the poor documentation on the web site, but at the same time, it was going to cost someone many hours of frustration at some point. Next time we do decide to use a board without a dedicated JTAG connector, we will know to check the schematic for the resistors right away. Both of these pitfalls seem like things we were going to stumble upon at some point regardless. Thus, while on the surface, the progress this semester does seem limited, that ignores these two important findings that may not advance the project, but will save someone else a reasonable

amount of time and frustration down the road.

Acknowledgments

It almost goes without saying that this project would have been significantly harder without Jason Cortell's help. I certainly wouldn't have made a fraction of the progress I did without him. Additionally, Sam Lee was extremely helpful with figuring out all of the issues with the microcontrollers, and just generally helping me get comfortable in the lab. Finally, Professor Ruina was very helpful in coming up with the final design on the sensor, specifically, the turning the emitter on and off once per measurement, to eliminate noise.

Citations:

[1] Andy Ruina, John E.A. Bertram and Manoj Srinivasan , "A collisional model of the energetic cost of support work qualitatively explains leg sequencing in walking and galloping, pseudo-elastic leg behavior in running and the walk-to-run transition" Journal of Theoretical Biology., vol. 237, no. 2, pp. 170-192, 2005.

Appendix A: LPC2103 Schematics

http://www.olimex.com/dev/images/lpc-h2103-sch.gif

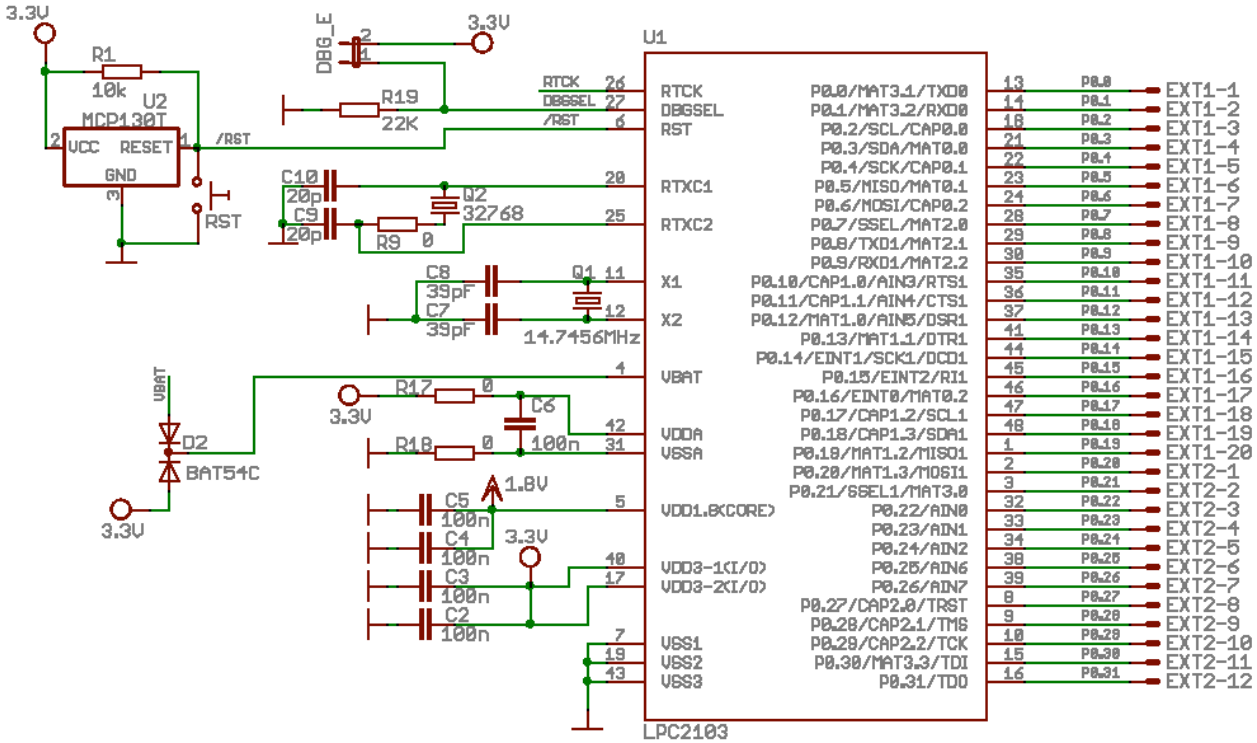

Figure A.1 Schematic for LPC2103 near the processor

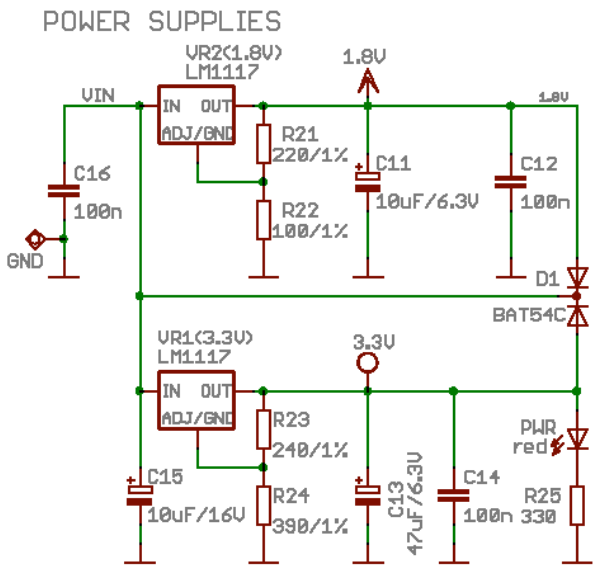

Figure A.2 Power Supplies on LPC2103

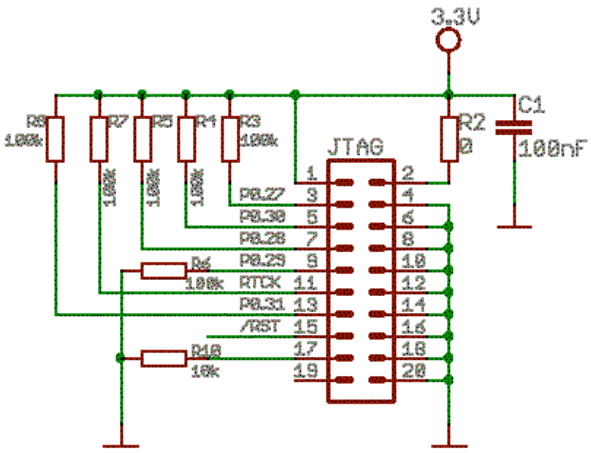

Figure A.3 JTAG connection LPC2103

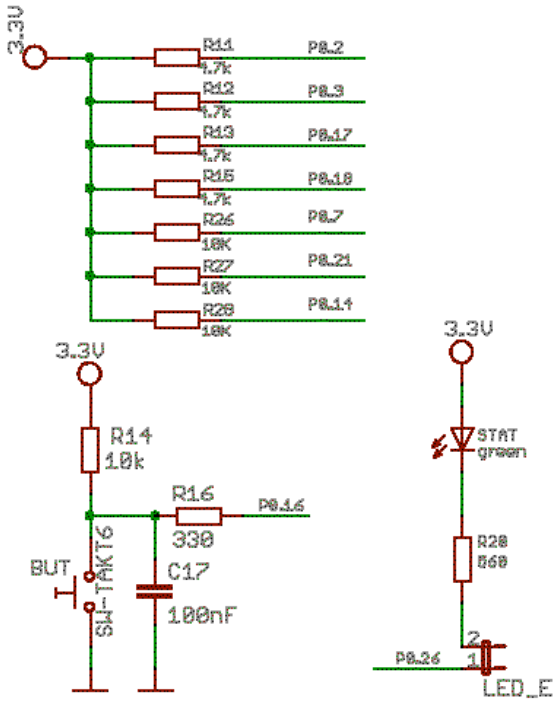

Figure A.4 Onboard I/O LPC2103

Appendix B: LPC2131 Schematics

http://www.newmicros.com/store/product_schematics_pdf/Tini2131_Sch.pdf

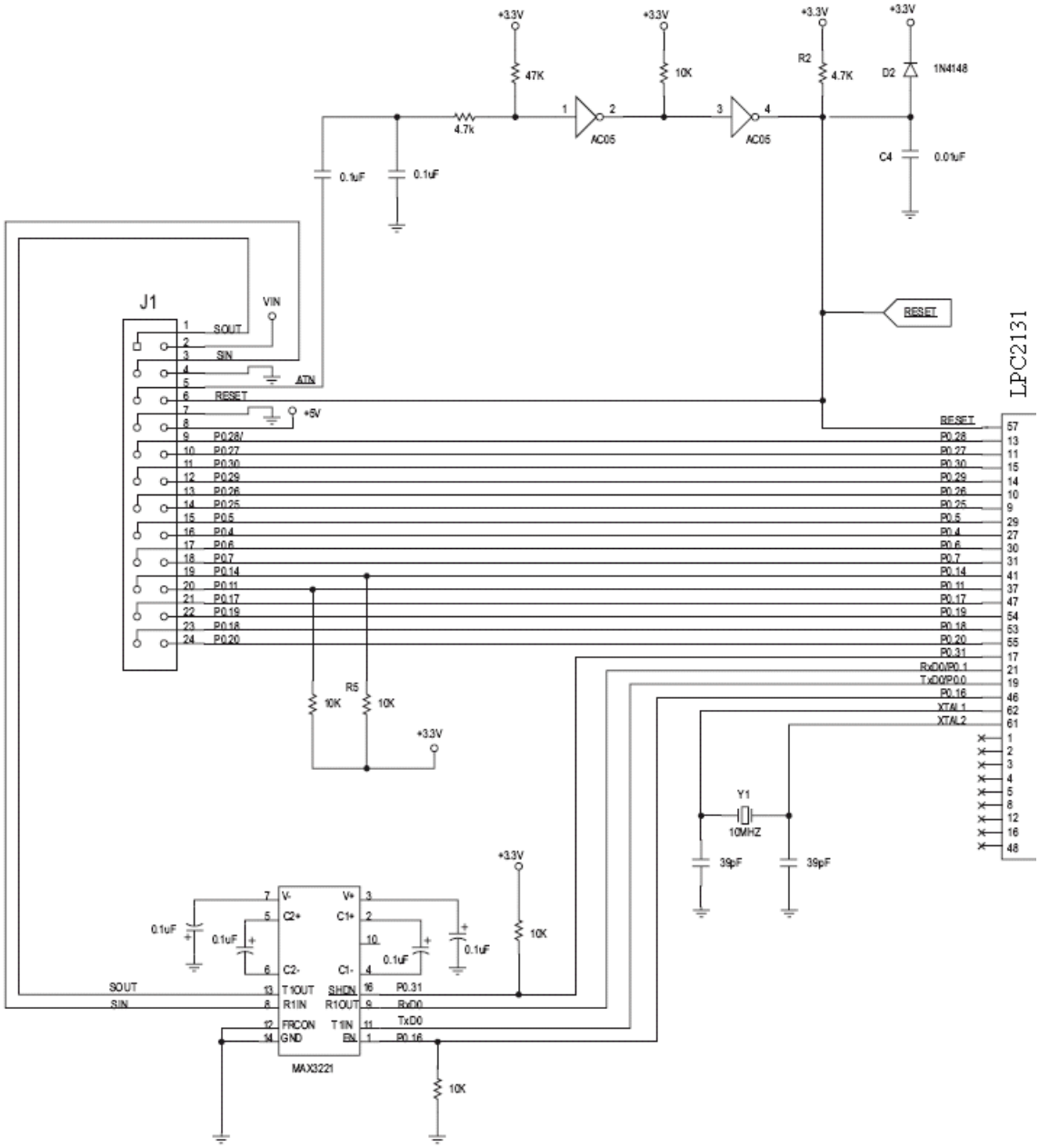

Figure B.1 Left Side of LPC2131

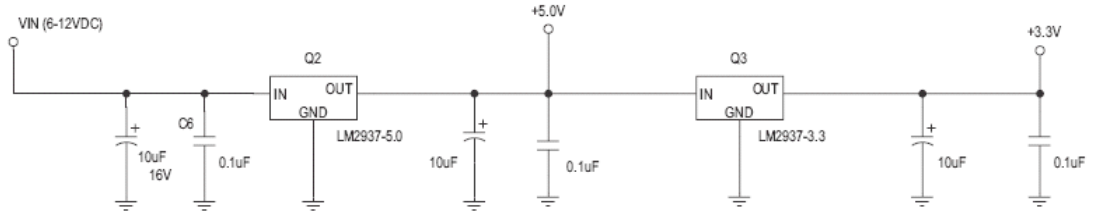

Figure B.2 Voltage Regulator Circuit

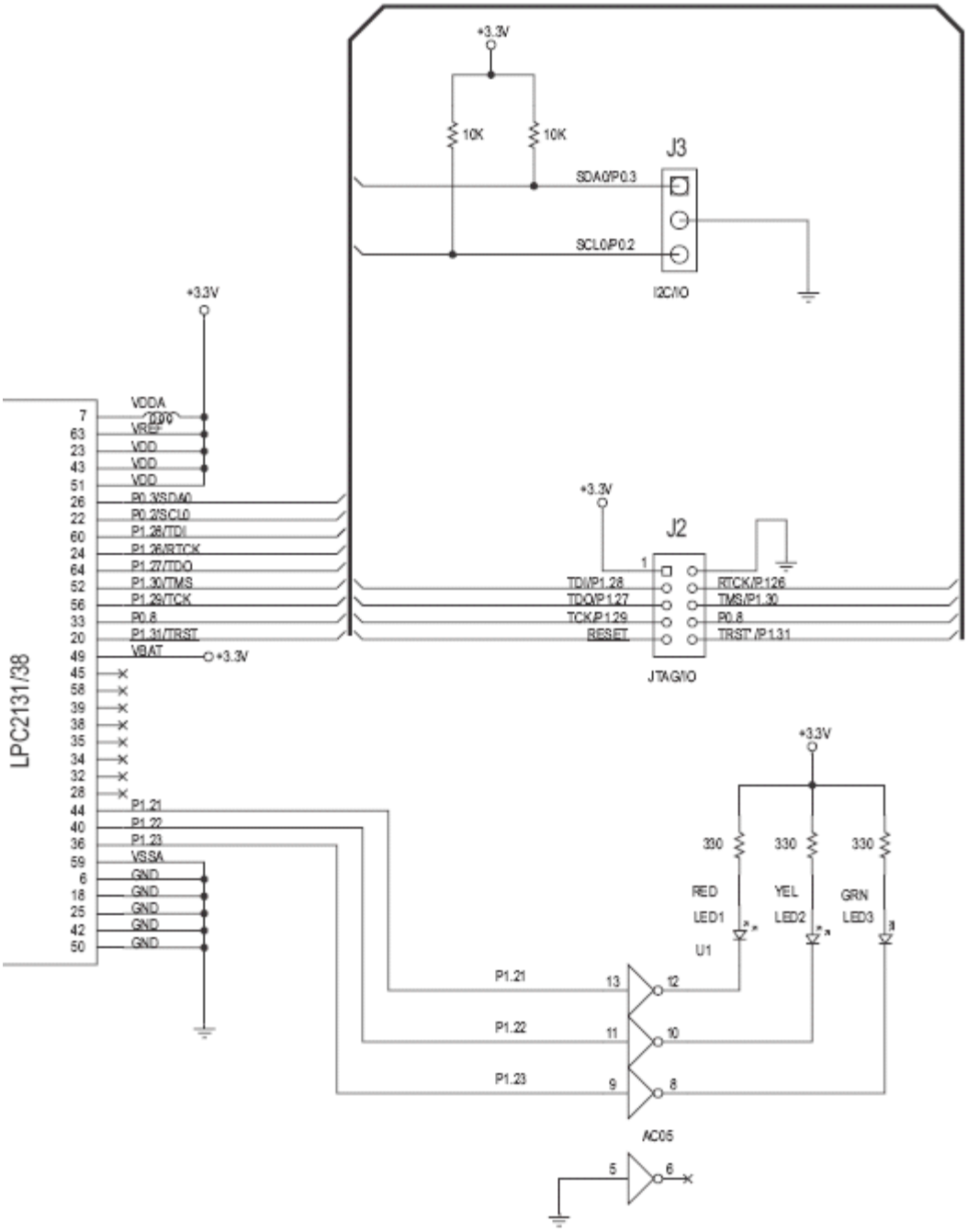

Figure B.3 Right side of LPC2131 including external connections (JTAG/J2)

Appendix C: LPC2129 Schematics

http://www.newmicros.com/store/product_schematics_pdf/plug-an-arm.pdf

Appendix D: Blinking Status Light Code explained

To get the status light to blink, I took the RTC_Interrupt code from the sample that came with the IAR workbench, and in the header file added the definition #define PIN0_26 0x1A, which is just 26 in hex. Also, in the RTC_Interrupt.c file,I changed every instance of PIN0_4 toPIN0_26. Finally, in the RTC_Interrupt.c file, I also changed PINSEL0_bit.P0_4=0; to PINSEL1_bit.P0_26=0; . While this all sounds like an easy set of changes, it was more difficult than it looks since the actual project includes 8 different files which I had to sift through, most of which was purely irrelevant and just class definitions. The code for the new microcontroller on a whole is a little painful to sift through and pickup, but it isn't too complicated and an experienced programmer should be comfortable with it in a matter of days.

Appendix E: Blinking Status Light Code

(On my computer at home, and will be included in final version of the report)