The Librarian

Final Report

Michael Boyco

EEL5666 Intelligent Machine Design Laboratory
Fall 2010

Instructors: Dr. Arroyo, Dr. Schwartz
TAs:
Mike Pridgen
Tim Martin
Ryan Stevens
Devin Hughes
Thomas Vermeer

12/07/2010

Table of Contents……2

Abstract

This final report covers the entire development and execution of a simple autonomous robot called the Librarian. The Librarian is a simple round body robot whose goal is to locate “loud” sounds via its sound triangulation system and then turn to that sound and fire a silencing foam object at this source. It exhibits three basic behaviors: obstacle avoidance, sound detection and location, reacting to the sound source.

Since I am a Chemical Engineering student, this is my first real exposure to robotics, microcontrollers, diodes, sensors, et cetera. It was very amazing and the experience of developing a robot from a 3D rendering to an actual physical thing (with arguably its own temperament) was amazing, to say the least.

Executive Summary

The Librarian consists of a very typical round body, two-level chassis. This platform was created using SolidWorks modeling software. The platform is made of wood (provided by IMDL) and milled out from SolidWorks using a modified T-Tech PCB machine.

Control of the robot was provided by a Pridgen-Vermeer Robotics XMega 128A1 microcontroller board. This very capable board was used to monitor the immediate vicinity around the Librarian in order for the robot to perform its main task: listen, detect and locate sound sources loud enough to trigger the robot and then respond to this source.

In addition to the XMega board, the robot had a sensor suite with which to interpret the world around it. This suite eventually consisted of an infrared sensor, two bump sensors, and three microphones. The microphones were arrayed in such a way as to be used to detect and locate sound sources in conjunction with the infrared sensor. The infrared was also used with the two bump switches for basic obstacle avoidance and to help the robot move from location to monitor to the next.

Upon detecting sound sources and locating them, if the appropriate conditions were met, the robot would then respond to the source with a "silencing" mechanism obtained from the Nerf toy company. There was the possibility, however, for the sound source to stop emitting sound, in which case the robot would carry on listening.

This behavior constitutes the major portion of the robots function and purpose. Indeed, it would be suited to function in a variety of environments but was designed with a quiet environment where loud sound sources would be very easy to detect and locate, i.e. a library.

Introduction

Sound triangulation is a difficult problem to solve in general. Sound is fast, sound can bounce off of objects and you have to have very dedicated hardware to get meaningful results. Compounding the problem is the fact that when you hook up a microphone to a microcontroller, now you have to figure out how to sample your environment in order to detect sounds (the ADC on the XMega board takes "snapshots" of events occurring at the sensors, giving you a patchy view of the world). For example, referring to page 142 of Mobile Robots by Jones, Seiger and Flynn gives us that a handclap is a sound even lasting approximately one millisecond. How often do we sample in order to pick up this event? (The answer is provided by the Nyquist Theorem).

A rough answer to that question is: very frequently. But if I am devoting all of my microcontroller's resources to sound detection, when does the robot simply roam? Check its other sensors? Blink a light? As you can see, the problem of sound triangulation was quickly becoming a major one, especially for this very amateur roboticist.

In addition, microphones are not cheap electronics. A good omni-directional microphone - one that can pick up your voice and relay that to speakers, amplifiers, etc. - costs somewhere in the neighborhood of $30 (RadioShack price). But then how do you interface that to a microcontroller (again, from my point of view)? If something goes wrong (likely), now I have to go buy another microphone and hope it is in stock. While $30 is not a lot of money in the bigger picture of robotics, I eventually ended up with the much cheaper (and very slimmed down) $7.95 SparkFun Electrotet microphone. I think that decision in hardware may have ultimately affected the Librarian's performance.

While I did not want to hack into something as complicated as a microphone, I was willing to hack into a USB controlled missile turret which ultimately ended up just being a bunch of DC motors and a strange firing mechanism that consisted of springs and some gears. While I did eventually get the missile turret working, it would also eventually get destroyed.

The scope of the project was to produce a robot that could perform simple obstacle avoidance and then, of course, sound detection and location. This was to be accomplished via the sound "triangulation" system that I found on the internet, but which ultimately evolve into something different (and which I can call my own).

A rough project sequence of events could be given as: design platform in Solidworks; mill out platform and assemble; add electronics and servos; show obstacle avoidance; start working with microphones; continue working with microphones; implement "silencing" mechanism; a lot of experimentation; demos.

Integrated System

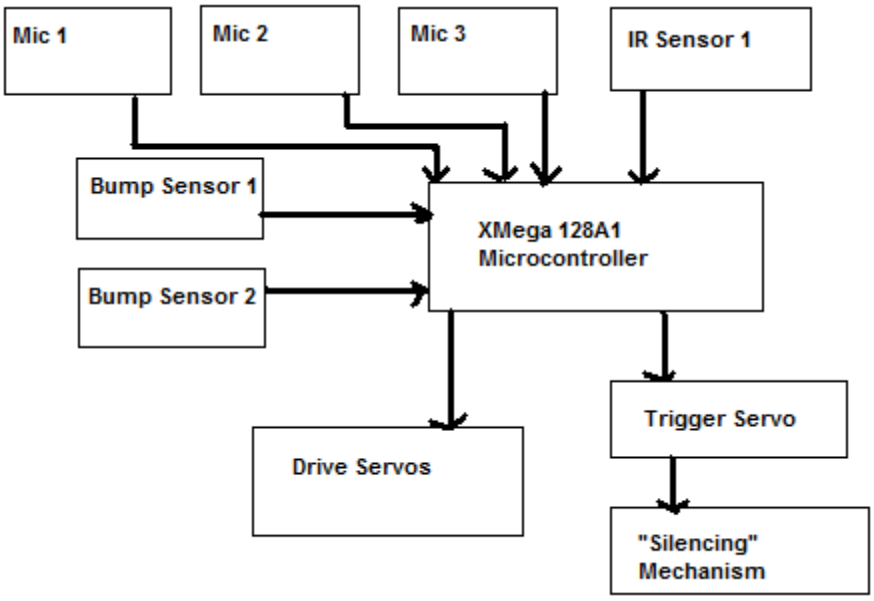

Fig. 1: Mapping of total system integration

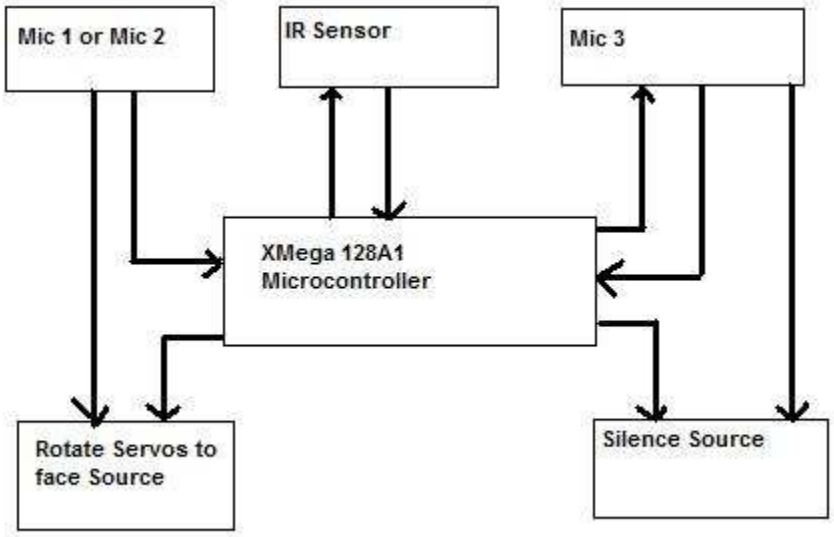

Fig. 2: Mapping of Special Sensor cascade of events

Mobile Platform

My platform consists of a circular, two-tier design with two drive wheels and a third omni-directional. This is a very typical design, especially for basic or first time robots (such as the Librarian). In addition to being fairly popular, I also favored this design since it would be the best way to implement my special sensor. The only real specification was that the platform be seven inches in diameter and that it be two tiered. The platform used circuit spacers in order to achieve the two tier system. Sandwiched in between the tiers is where I housed most of the electronics. The entire platform was designed in SolidWorks and milled out using a modified T-Tech PCB machine.

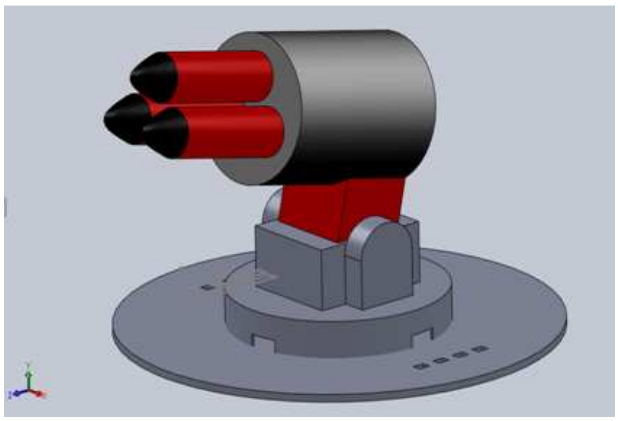

Fig. 3: SolidWorks, 3D rendering of concept.

I designed pieces of wood to fit into the bottom level via a brick cut out (very much inspired by Tim Martin's SolidWorks lecture) and these are where the servos were going to be attached. Originally these cut outs would be joined by another length of wood to form a sort of shelf along the bottom of the robot. Despite my best efforts at measuring all the dimensions, however, I still managed to get the sizes wrong. The wheel radii were greater than the length of wood and hit the bottom of the platform. What I ended up doing was grabbing the bottom of the shelf (i.e. the longest cut of wood) and cutting it in half. Since the shelf bottom had the same brick cut out, it fit into the original slots and was long enough to accommodate the servos and wheels. The omni wheel was bought at Lowe's. I measured its height, cut out a piece of wood to fill in the rest of the height so that it would sit flat with the drive wheels, then screwed the whole thing together. The nice thing about this wheel (and its "mounting block") was that it provided a nice heavy end to the robot, but low enough to the ground to not tip over. This was useful for supporting the weight of the Librarian's power source, six AA batteries; definitely one of the heavier components of the robot.

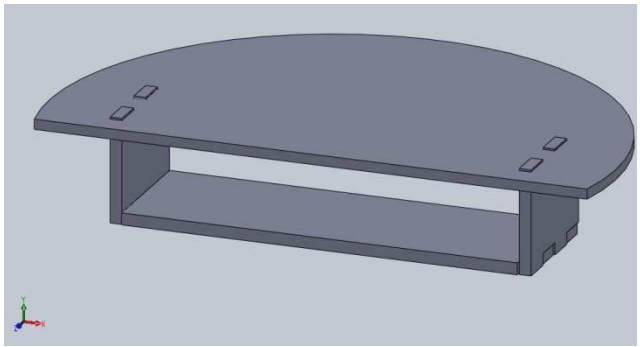

Fig. 4: SolidWorks, 3D rendering of lower-tier of platform with shelf for servos and wheels. Cross-sectional view.

After I hacked into (and left mostly in pieces) my original "silencing" mechanism, I was left with a good quantity of small screws. I used these to attach my microcontroller to the platform, in addition to using these screws to securely attach the servo mounts to the rest of the platform.

Due to lack of foresight, I ended up having to cut out a small rectangular section of wood for the LCD screen to poke through. I also made a cut out on the top tier for my “silencing” mechanism but ultimately did not use it.

There were several advantages to using wood (and I enjoyed all of them): cheap (in our case, provided by the lab), easy to work with, can drill out holes or drill in screws, easy to paint. I painted my robot black and would eventually drill a ton of wholes into the platform. My only real disadvantage with the platform came from lack of planning. I got lucky several times.

Actuation

My actuation came from two servos that I hacked for continuous rotation. Hacking the servos was very interesting and was one of the first real “hands-on” things I did in this class. I bought my servos from Warrick's Hobby Super Store in Plantation, South Florida. I bought TowerPro SG-5010 servos. Their operating speed is 0.20sec/60degree (4.8V) and 0.16sec/60degree (6.0V). The PVR board supplies a 5.0V source for servo operation, so my speed must be somewhere in between. That being said, each servo has its own operating characteristics. I suspect that this comes from the hack job I did.

Each servo came with a potentiometer (to control the range of motion a servo will move). When I hacked them for continuous rotation, I removed the potentiometer and replaced it with a network of 2.2 kiloohm resistors. To figure out the values to set each individual servo so that they would move the same speed, a lot of experimentation was done (a lot of trial and error). Even so, the servos were still liable to do their own thing, especially when they were set to “zero speed”. I experienced current problems with my board, so I assume this is just another symptom of that.

The wheels I used are for a model race car. I bought them at HobbyTown USA here in Gainesville. While expensive, they are made of foam that is hard enough to roll over a surface, but with just enough give to provide a very nice traction surface (I had no clue how heavy my robot my get, so I figured this was a nice just-in-case feature). As mentioned before, the third wheel is an omni-directional wheel from Lowes. I believe it is actually a replacement for an office desk.

My original “silencing” mechanism was a USB Foam Missile Turret (read: harmless) from ThinkGeek.com. Over the Thanksgiving break, I managed to destroy the turret (after I had figured out how to fire it using the PVR board). I went to Wal-Mart and bought a cheap plastic replacement “silencing” mechanism. Using a servo that I got from the “for free” box in lab in the first week of class, I rigged a firing mechanism. The “silencing” mechanism was heavily modified: I cut away plastic pieces so that it could sit on top of my platform comfortably and also cut a bit of a spring in order to ease the tension in the trigger. I was able to successfully implement this new “silencing” mechanism for Media Day.

Sensors

My final sensor suite consists of a single infrared (IR) sensor, two bump switches, and the sound triangulation system.

IR sensors are typically used for range-finding applications, especially obstacle avoidance. I used Sharp Infrared Proximity Sensors, part number GP2Y0A21YK (SparkFun No. SEN-00242). Note that this is not the “long range” or “short range” version. For my final design, a single IR was located in front of the robot and pointing directly forwards. Previously, there were two IR sensors, both off-center and oriented so that they pointed out and away from an imaginary center line along the front-back axis of the robot. This old design was to prevent the wheels and their axles from getting snagged on any obstacles. To implement my special sensor, however, the IR array was replaced with a single forward-facing IR.

Once I understood how the analog-digital converter worked, I was able to catalog the performance of my IR sensors. While we were constantly warned that the data sheets that accompany most sensor hardware you buy is untrustworthy, I found that my sensors generally behaved exactly as advertised, especially at short range. At longer ranges, performance dropped off, especially for black backgrounds.

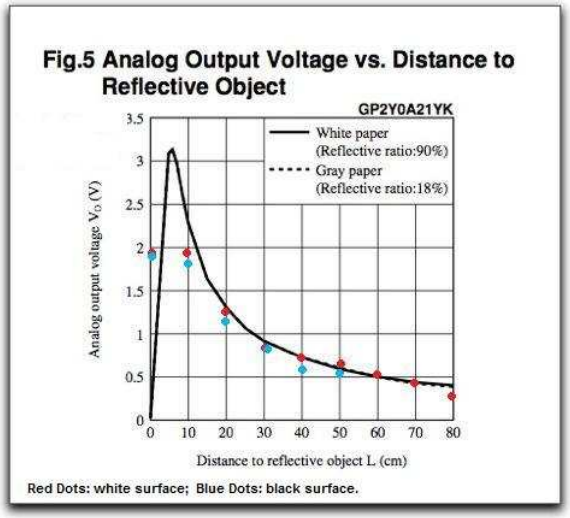

Fig. 5: IR range test under incandescent lighting.

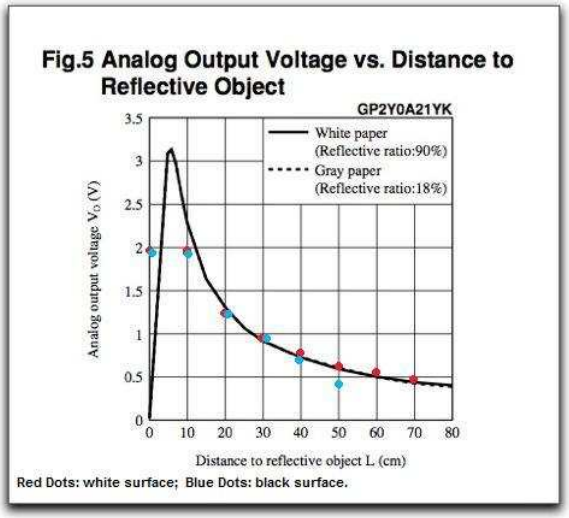

Fig. 6: IR Range test under fluorescent lighting.

The bump sensors were located on metal tabs that were placed around the wheel mounts. Originally they were a last line of defense to keep the wheels from getting caught on any obstacles that the IRs didn't detect. With the change in design, however, they became the only way to detect if the robot got caught on something. The bump sensors were also scavenged from the original "silencing mechanism". My bump sensors were wired so that closing the circuit gave a digital input of "logic one".

Wiring Diagram:

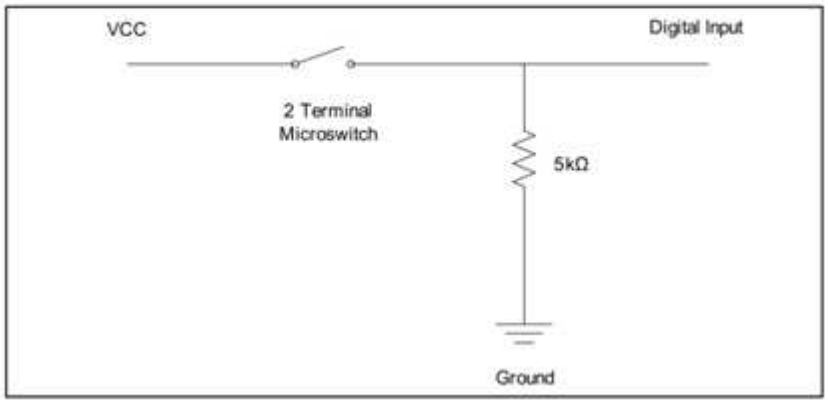

Fig. 7: Wiring diagram fom bump sensors. Make use of a pull-down resistor.

Special Sensor System

My special sensor (and really, the purpose of this whole robot) was the "sound triangulation" system. My special sensor was the single hardest aspect of the robot. Originally, the idea was conceived from something I found online. Specifically, the PeanutBot by a team of student at Cornell University. The robot had its own project page which explained how the robot was assembled, programmed and executed. It also included the relevant theories behind its operation.

The system of the PeanutBot consisted of three microphones arrayed in a circle around some center point. The system depended on the microphones all "hearing" some sound source. Using geometry, arrival times are calculated for each microphone. The differences in arrival time are then used to calculate the sound source's location with respect to one microphone labeled as "mic 1". The Cornell team designed and built their own microphones and put the circuit online. When I tried to implement the circuit myself, however, I quickly found myself lost and out of my league. I am not an electrical engineering student and this circuit was considered advanced - even for an electrical engineer – by the TAs. The circuit itself left more questions than answers and I am not sure I ever built it correctly. I ordered the same microphones as the Cornell team (Panasonic Unidirectional Back Electret Condenser Microphone Cartridge, Part no. WM-55A10) and attempted the circuit. I immediately ran into problems.

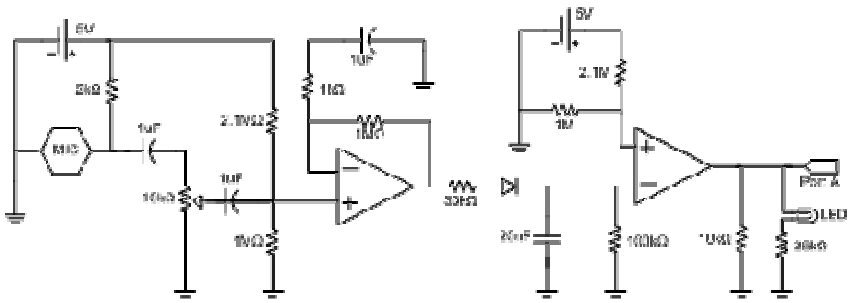

Fig. 8: Cornell PeanutBot mic circuit. Consists of: passive lowpass filter, half-wave rectifier with a capacitor and analog comparator to discretize the signal.

Eventually, I was introduced into the SparkFun electrotet microphone which included a break out board (SparkFun no. BOB-09964). The boar d comes with an LM-334 op-amp which amplifies the signal 100x. I settled on this design out of a desire to make my life simpler and not waste time on details that weren't relevant to the overall project. While the Cornell team's mic was very interesting (for example, it put out only 0V or 4V, unlike a standard mic), more reading on their webpage revealed that they were a group of undergrads doing work for a large advanced research group, meaning who knows what resources and expertise they had. I moved on.

Fig.9: The SparkFun Electrotet Mic with Breakout Board, front view.

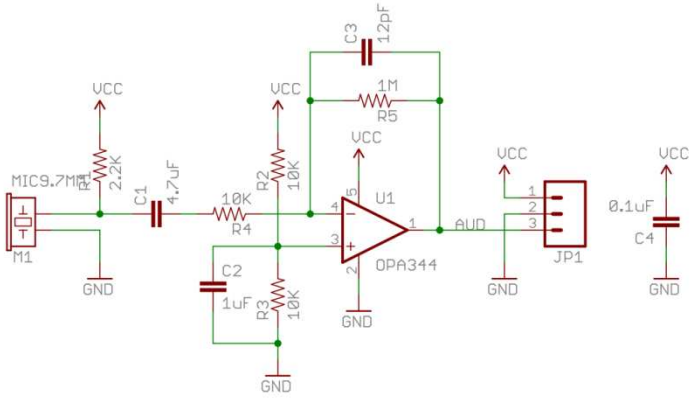

Fig. 10: The SparkFun Electrotet Mic with Breakout Board circuit diagram.

Along with a new microphone, I also abandoned the idea of using time stamps to determine sound location. I adopted a much simpler algorithm that while less elegant was much easier to implement. Ultimately, this ended up being a good idea since the PeanutBot can only operate in very quiet conditions (not a good representation of the real world.)

Regrettably, now I would be hindered by sheer ignorance of how sound works. A sound wave is a sine wave. This means that a sound wave is centered around some level and oscillates around this level. While an ideal sine wave on paper might oscillate around the x-axis, actual sound waves oscillate around some non-zero, non-negative value. After much headache and a lot of wondering as to what my mics were doing, I was recommended to put the mic on an oscilloscope.

My problem was that I made a major assumption regarding the microphone output. Looking at a typical example output from a microphone, it is easy to see that the output is very erratic (and doesn't look like what is usually considered a sine wave). So, seeing one of these example outputs, and then seeing what I had as an input from the mic on my LCD, I thought everything was behaving "normally" for a microphone, even if it made no sense, and that I would just have to work around it. This assumption proved to be incredibly wrong and almost derailed the whole project.

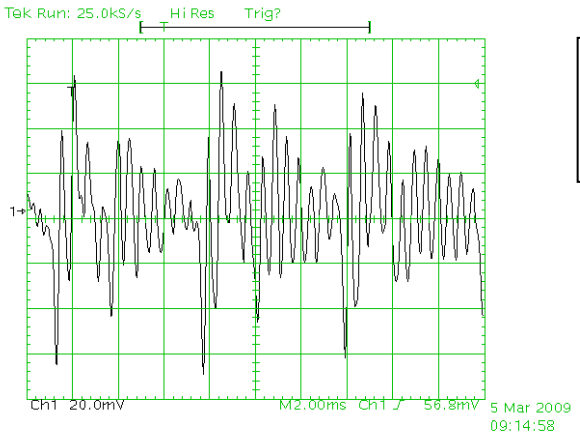

Fig. 11: Typical microphone output. Microphone output centered around 1mV.

Using the oscilloscope, I was able to see that the microphones were centered around 1.65-1.7V. Any loud sounds heard by the mics meant that the voltage jumped up to some level above 1.65-1.7V and all the way down to 0.5 volts. But… My analog-digital converter was set to have the reference voltage as VCC/1.6 or about 2V. This explained a lot. Every time my mics heard a sound, I was only seeing the lower portion of the sine wave behavior. The upper portions were so close to the reference voltage that I was saturating the ADC and not getting any meaningful data. I reset the reference voltage to 3.3V and then performed half-wave rectification within the programming of the PVR board (the rectification was easier to implement then a Schmitt trigger-like correction since my sound behaviors relied on having a single "max" value. The Schmitt trigger was way more complicated, in my eyes, than simple rectification). This realization explained the mystery of why the mics seemed to operate on an opposite scale as the IR sensors.

Also, using Chester Udell's suggestion of a foam cup to help isolate the microphones from each other greatly reduced the amount of noise they heard that was not in front of the mic, meaning the robot was less responsive to loud sounds that *should* trigger other microphones.

While my mics were still behaving rather erratically, I now understood what was happening and the erratic behavior was somewhat decipherable. Also, one of the mics looked like it wasn't working properly anymore. I re-arranged my microphones so that two were facing towards the front of the robot, but off-center and facing away from the center front-back axis. Ultimately, I was able to reintegrate the "broken" mic and use it for the final sound detection step.

With enough programming, I was able to get the microphones to function enough that my robot was able to function the way I said it would. That being said, the mics still went off for no apparent reason (i.e. they showed spikes in voltage consistent with hearing a very loud sound when in fact I was sitting quietly, alone in a room). Ultimately, the microphones were unreliable enough that I implemented an IR sensor in the sound behaviors in order to try and get more accurate targeting.

All in all, the special sensor system went through at least three major revisions. What *was* consistently present throughout all the revisions were: basic sound listening/locating behaviors (discussed next) and sampling frequency.

Sampling frequency was prescribed by the Nyquist Theorem. The Nyquist Theorem says, in a nutshell, that you must sample at least twice as frequently as your shortest expected frequency. So, say the Librarian wants to be able to react to a hand clap. A hand clap is an event that last about 1 millisecond (according to Google and the Mobile Robots handbook). Therefore, in order to pick up the handclap, the Librarian must sample every 0.5 ms.

Behaviors

The Librarian exhibits three basic behaviors. They are: wandering, listening/locating, silencing.

Wandering

Wandering is basic obstacle avoidance. The Librarian roams around for some predetermined period of time, while simultaneously checking its environment for obstacles via its single IR sensor and two bump sensors.

Listening/Locating

This behavior is really the hear t and soul of the Librarian. This behavior encompasses the actual detection of loud sounds and subsequent tracking of the sound source's location. Because of all the problems associated with the microphones unreliability (before and after I figured out how the microphones work), I implemented a three-strike system before the robot decides it has actually heard a loud sound and located it.

First the robot must actually hear a sound. The two microphones arrayed around the front listen and when one hears a loud enough sound, the Librarian re-orients itself so that the microphone that heard the loud sound is now facing in the general direction of the sound source.

The second strike time window now opens. After the robot has oriented itself towards the sound source's possible general direction, it performs a rotation across this area. The IR takes samples as the robot sweeps across this arc and the values are saved in an array. When the robot is finished sweeping, it calculates the largest value in the array and then turns back to that position via further calculations (each array space has an associated integer value which can be used to calculate how much the robot must sweep back to face the IR sensor's largest reading location). Theoretically, if there is a sound source near enough to the robot to set it off, then it stands to reason that this source is also close enough to detect via the IR. The downfall is that if the robot is right next to a wall or some other obstacle, this other object might generate the largest IR value.

This leads us to the final strike. The third microphone is positioned facing directly forward. If the sound source makes one more peep (now that the robot should be facing the source), the robot assumes that it is facing the source, that the source has had enough chance to be quiet all on its own, and that it can carry out the next behavior. If the IR picked up some inanimate object, then it will not make a sound (a wall wouldn't make a sound to set off the robot, theoretically) and therefore not fire and exit the listening/locating behavior, dismissing the previous actions as a false alarm.

Silencing

Three strikes. You're out. The Librarian has performed its sole function in life and you are quiet. This is performed via the "silencing" mechanism. See pictures for more details.

Details of the Behaviors and their Arrangement

Because of the desire to be able to react to loud sounds (and the assumption that most loud sounds are also short events, i.e. a handclap) the Librarian must devote a lot of its time sitting around listening. This meant that the standard subsumptive model for robot behavior was going to be difficult to implement for the Librarian. So, for now the Librarian sits around listening for some period of time. There will be some false alarms (some genuine, some due to the schizophrenic nature of the mics) and then the robot escapes the listening loop. It wanders around, avoiding obstacles, roving and eventually returns to the listening loop. On the plus side, since the robot cannot really listen *and* rove around and avoid objects, when it does sit around listening, it does not have to account for the noise generated by its own servos or sounds caused by impacts with objects.

Most of the Librarian's main behavior is devoted to ensuring the Librarian does not randomly fire. Coincidentally, these behaviors give the sound source the chance to be quiet of its own accord and escape detection.

Experimentation

Most of the experimentation was qualitative, or trial and error. The only solid attempt at a genuine experiment was the IR range experiments (whose results you can observe in the Sensors section of the report). A typical experiment for the Librarian was done in code, then compiled and left to run while I stood back and observed with the aid of the LCD.

For example, a simple "for" loop was used to step through servo values for each servo until the servos slowed down, then stopped, then started up again but in the reverse direction. That particular experiment yielded the speeds for my "slow()" and "stop()" servo functions. A similar experiment yielded a good speed for "turn_right()" and "turn_left()" functions.

A majority of experiments were done in order to determine rates of rotation for the robot. This was crucial for execution of the first and second strike time frames of the listen/locate behavior. A pretty good estimate for the rate of turn for the robot seemed to be about 5 seconds per 180 degrees of rotation. To get any fraction of the turn value, I performed simple arithmetic to get whatever value I needed.

Most of the code for the second strike time frame (the IR detection) was done to figure out a robust method to detect the presence of a sound source and then accurately track back to that source's location. Due to the last minute final revision of the special sensor array, not all the bugs were quite worked out (indeed, the code – and consequently the robot's actions - made no sense until I realized I was still using the ADC channel for a speaker instead of the IR sensor); this unfortunately manifested itself on Media Day when the Librarian overshot its target (although I have reason to suspect there was an errant

hand in its immediate vicinity), although the Media Day presentation was decidedly more impressive than the Final Demo Day.

Conclusion

The Librarian turned out to be a very different project from the one I envisioned at the beginning of the semester. In my final opinion, I am glad that I was able to get the Librarian to perform and demonstrate that it could do what I said it would do. That being said, I feel that I very loosely accomplished what I said I would. Like I said, I envisioned something quite different (and much more impressive as well).

I think that the choice to use audio input as a special sensor was both a good idea (it's unique: doesn't seem to be used very often and I am sure that I am the only person in the Fall 2010 class to use audio input as a special sensor) and a nightmare idea (audio in general is difficult; to get good, reliable microphones, you probably need to spend more than $8 a mic; honestly, experience counts. A lot.). Clearly, something is weird is happening somewhere between the microphone input and whatever the servos do. Putting the mics on the oscilloscope proved that at least the one worked. I wonder if higher quality mics would be able to cope with background noises better. Or perhaps an op-amp that does not amplify the signal as much as the on provided on the SparkFun breakout board? Either way, I suspect that audio should be left to the graduate students.

Other possible improvements include adding a second board and then splitting duties. The second board's sole function (and main loop) would be to continuously monitor for sounds. The main board could handle all other tasks, including obstacle avoidance. When sounds are detected, the second board could communicate simple details to the main board and the main board could start maneuvering around to determine source location. The relatively cheap price of Arduino microcontrollers and use of serial communications make this a viable option. A major obstacle would be to account for motor noise (the mics are sensitive enough in quiet environments). This would also allow the robot to operate more fluidly and allow for a more familiar subsumptive behavior pattern.

Another improvement would be to possibly redesign the two-tier chassis for something more accessible. Programming the board became more and more a hassle as I added more features to the robot. More wires and more weight on the top tier (from "silencing" mechanism, for example) made programming a more and more daunting task. Also, a bigger platform would allow for more electronics (like a slave board).

This robot ended up being (more than anything) an exercise in being able to change plans rapidly and constantly think of new ideas and ways to implement those ideas. That is one aspect of the project that I actually enjoyed (although I spent many sleepless nights praying for a miracle). I honestly believe that I benefitted from this class more than I did from my senior design class for chemical engineering. To see the idea I started out with to the robot I ended up with is actually quite amusing and a testament to just how much improvisation I had to do.

If I had the chance to redo this project, I am not entirely sure I would choose this particular robot design again. If I was more familiar with robotics, microcontrollers, sensors, and electronics in general, I would have almost certainly chosen something else, probably something involving swarm robotics. I am sure that I would still get the same healthy mix of incredible frustration when nothing works and the amazing sense of accomplishment when something does actually work (especially if it works in the sequence intended).

Finally, one of the major reasons I joined this class was to learn about robots and electronics. Mission accomplished. I made it a habit to ask around in lab to see what other people were up to. Between speaking to other students, the TAs, and the great books assigned by the professors, I can honestly say I would love to take the course again, just to have a good excuse to build another robot (preferably without audio). I learned way more about electronics than I would have in a more traditional EE class and met some truly remarkable, very capable people, so despite all the trouble I had with my robot's special sensor, I am glad I took the class.

Resources

https://sites.google.com/site/boycosautobots/home

All documentation of the Librarian project.

SparkFun.com

Various sensors, etc.

http://courses.cit.cornell.edu/ee476/FinalProjects/s2007/ai45_hkc2_sbs43/ai45_hkc2_sbs43/index.html

Cornell's Autonomous PeanutBot, provided inspiration for Special Sensor

http://mil.ufl.edu/imdl/

Course website, various resources available.

Mobile Robots by Joseph Jones, Bruce Seiger, Anita Flynn

Source of reference and inspiration

MIT 6.270

Source of reference

http://plaza.ufl.edu/rhaegar/XMega%20Manual.pdf

Manual for the Pridgen-Vermeer Robotics XMEGA 128A1 microcontroller board.

Google.com / en.Wikipedia.org

The mother/father pair of all resources/references/etc. Use at your own risk.

```
#include <avr/io.h>
#include "PVR.h"
#include <math.h>

/*****************************************
        Written by Michael Boyco
        EEL5666
        IMDL Fall 2010
        10/21/2010
*****************************************/


#define FAST_MOTOR_L -100
#define FAST_MOTOR_R 100
#define SLOW_MOTOR_R 27
#define SLOW_MOTOR_L 19
#define ZERO_MOTOR_R 12
#define ZERO_MOTOR_L 4


//*** VARIOUS FUNCTIONS
*********************************************************************************8
//left is servo d3, right is d4

void startup(void)
        {
        xmegaInit();                                    //setup XMega
```

```
	delayInit();				//setup delay functions
	ServoCInit();				//setup PORTC Servos
	ServoDInit();				//setup PORTD Servos
	ADCAInit();				//setup PORTA analong
readings
	lcdInit();				//setup LCD on PORTK
	lcdString("The Librarian");		//display "PV Robotics" on top line
(Line 0) of LCD
	lcdGoto(1,0);				//move LCD cursor to the
second line (Line 1) of LCD
	lcdString("EEL5666");			//display "Board Demo" on second line

	PORTQ_DIR |= 0x01;			//set Q0 (LED) as output
	PORTB_DIR |= 0x01;
	PORTH_DIR |= 0x01;
	PORTJ_DIR |= 0x01;
}

// full stop
void stop(void)
	{
	ServoD4(ZERO_MOTOR_R);
	ServoD3(ZERO_MOTOR_L);
	delay_ms(2000);
	}

//full speed ahead
void fast(void)
```

```
        {
        ServoD4(FAST_MOTOR_R);
        ServoD3(FAST_MOTOR_L);
        }

//slow ahead
void slow(void)
        {
        ServoD4(ZERO_MOTOR_R+15);
        ServoD3(ZERO_MOTOR_L-15);
        }

void left_turn(void)
        {
        ServoD4(ZERO_MOTOR_R-15);
        ServoD3(ZERO_MOTOR_L-15);
        }

void right_turn(void)
        {
        ServoD4(ZERO_MOTOR_R+15);
        ServoD3(ZERO_MOTOR_L+15);
        }

//heard a sound, lets turn and then move towards it
void heard_something(int dir)
```

```
{
        if(dir==1)
                {
                delay_ms(1000);
                fast();
                delay_ms(1000);
                stop();
                lcdInit();
                lcdString("LOUD! center");
                delay_ms(3000);
                        }
        else if(dir==3)
                {
                right_turn();
                delay_ms(1000);
                stop();
                fast();
                delay_ms(1000);
                stop();
                lcdInit();
                lcdString("LOUD! right");
                delay_ms(3000);
                        }
        else if(dir==2)
                {
                left_turn();
```

```
                delay_ms(1000);
                stop();
                fast();
                delay_ms(1000);
                stop();
                lcdInit();
                lcdString("LOUD! left");
                delay_ms(3000);
                        }
        }

//check for repeat of sound source before firing
int check(void)
        {
        PORTQ_OUT = 1;
        for(int i=0; i<1200; i++)
                {
                int center=ADCA5();
                delay_ms(0.3);
                        if(center<2000)
                                {
                                slow();
                                delay_ms(1000);
                                stop();
                                PORTQ_OUT = 0;
                                return 1;
```

```
                    } //close if
            } //close for loop
        PORTQ_OUT = 0;
        return 0;
        } //end function

void check_left(void)
        {
            int loud[24], i, center;

            right_turn();
            delay_ms(2000);
            stop();
            delay_ms(300);

            left_turn();
            for(i=0; i<24; i++)
                {
                delay_ms(.3);
                center=ADCA2();

                loud[i]=center;
                lcdInit();
                lcdInt(center);
                }
```

```
        stop();

        int j;
        int m=loud[0];
            for(int i=1; i<23; i++)
                if(m>loud[i])
                    {
                    m=loud[i];
                    j=i;
                    lcdInit();
                    lcdString("Found Max");
                    delay_ms(500);
                    }

            j=(j-22)*(-1);

            lcdInit();
            lcdString("right");
            right_turn();
            int turn=210*j;
            delay_ms(turn);
            stop();
            delay_ms(1000);
    }

void check_right(void)
```

```
{
        int loud[24], i, center;

        left_turn();
        delay_ms(2000);
        stop();
        delay_ms(300);

        right_turn();
        for(i=0; i<24; i++)
                {
                delay_ms(.3);
                center=ADCA2();



                loud[i]=center;
                lcdInit();
                lcdInt(center);
                }

        stop();

        int j;
        int m=loud[0];
                for(int i=1; i<23; i++)
```

```
                    if(m>loud[i])
                        {
                        m=loud[i];
                        j=i;
                        lcdInit();
                        lcdString("Found Max");
                        delay_ms(500);
                        }

                j=(j-22)*(-1);

                lcdInit();
                lcdString("right");
                left_turn();
                int turn=130*j;
                delay_ms(turn);
                stop();
                delay_ms(1000);
        }

void wander(void)
        {
        int ir_right, ir_left;
                ir_right=ADCA2();
                ir_left=ADCA1();
```

```
            if(ir_right>2500)
                    {
                    left_turn();
                    delay_ms(450);
                    }
            if(ir_left>2500)
                    {
                    right_turn();
                    delay_ms(450);
                    }
            else fast();
    }


//** MAIN
*****************************************************************************************
*******

void main(void)
{
    startup();

  PORTQ_DIR |= 0x01;                                    //set Q0 (LED) as output
  PORTB_DIR |= 0x01;
  PORTH_DIR |= 0x01;
  PORTJ_DIR = 0x00;
```

```
delay_ms(1000); //time to setup etc


PORTQ_OUT=1;
PORTH_OUT=1;
PORTJ_OUT=1;
delay_ms(1000);
PORTQ_OUT=0;
PORTH_OUT=0;
PORTJ_OUT=0;
delay_ms(1000);



double right1, right2, right3;
double left1, left2, left3;
double left, right;
double max;
double suml, sumr;
int dir;



while(1)
{
```

```
        lcdInit();

right1=ADCA6()-50;
left1=ADCA5();
delay_ms(0.3);


right2=ADCA6()-50;
left2=ADCA5();
delay_ms(0.3);


right3=ADCA6()-50;
left3=ADCA5();
delay_ms(0.3);

        //rectification of sound wave


if(right1<2110)
        right1=4075-right1;

if(left1<2110)
        left1=4075-left1;
```

```
if(right2<2110)
        right2=4075-right2;

if(left2<2110)
        left2=4075-left2;


if(right3<2110)
        right3=4075-right3;

if(left3<2110)
        left3=4075-left3;

        //average of values


left=(left1+left2+left3)/3;
right=(right1+right2+right3)/3;

        //find max value from three mics

max = right;
dir=1;
        if(max<left)
                {
                max=left;
```

```
                dir=2;
                }


if(max>3900)
  {
  if (max==left)
        {

                lcdInit();
                lcdString("left");
                delay_ms(1000);

                left1=ADCA5();
                delay_ms(0.3);

                left2=ADCA5();
                delay_ms(0.3);

                left3=ADCA5();
                delay_ms(0.3);

                if(left1<2110)
                        left1=4075-left1;

                if(left2<2110)
```

```
        left2=4075-left2;

if(left3<2110)
        left3=4075-left3;

         left=(left1+left2+left3)/3;

if (left>3800)
        {
        check_left();
                {
                                int final=ADCA7();

                                        if(final>3800)
                                                {
                                                lcdInit();
                                                lcdString("fire!");
                                                delay_ms(200);
                                                ServoD0(70);
                                                delay_ms(1000);
                                                ServoD0(-70);
                                                delay_ms(1000);
                                                ServoD0(70);

                                                }

                }//end check left
```

```
                    }// 2nd check
        } //end max left loop

  else if (max==right)

{

                lcdInit();
                lcdString("right");
                delay_ms(1000);

                right1=ADCA6();
                delay_ms(0.3);

                right2=ADCA6();
                delay_ms(0.3);

                right3=ADCA6();
                delay_ms(0.3);

                if(right1<2110)
                        right1=4075-right1;

                if(right2<2110)
                        right2=4075-right2;
```

```
if(right3<2110)
        right3=4075-right3;

         right=(right1+right2+right3)/3;

if (right>3800)
        {
        check_right();
                        {
                                int final=ADCA7();

                                        if(final>3800)
                                                        {
                                                        lcdInit();
                                                        lcdString("fire!");
                                                        delay_ms(200);
                                                        ServoD0(70);
                                                        delay_ms(1000);
                                                        ServoD0(-70);
                                                        delay_ms(1000);
                                                        ServoD0(70);
                                                        }

                        }//end check right
```

```
                    }//end 2nd check
        } //end max right loop



}

else
        {
        lcdInit();
        lcdString("No sounds heard");
        }

delay_ms(1000);

}



} //end of main
```