Control and Coordination of Micro-Robots

Lydia A. Lei and Christopher L. Perkins

Abstract - **Due to space limitations, micro-robots must operate with restricted processing and sensing capabilities. Consequently, obtaining intelligent and coordinated behavior from a swarm of micro-robots is a challenge. Our goal is to develop a swarm of micro-robots that rely only on their on-board processing and communication capabilities. Prior researchers have developed algorithms for movement/ control and explored distance sensing using Received Signal Strength Indication (RSSI). We plan to begin the implementation of this research in mini-robots, as well as develop inter-robot communication protocols and an alternative technique for distance sensing (Time Difference of Arrival (TDOA)) with the ultimate goal of achieving coordinated swarm activity.**

Index Terms – **Received Signal Strength Indicator (RSSI), Time Difference of Arrival (TDOA), Coordination, Goertzel Algorithm, Signal Processing**

I. Introduction

The long term goal of our project is to eventually build a swarm (10+) of autonomous mini-robots that are able to coordinate and communicate with one another in a smooth fashion. We are focused mainly on movement coordination -- having the robots follow a leader or having the robots move in simple formations. However, in order for the robots to move together, they must first know where they are and how far they are relative to the other robots; distance measurement is the first step. The main goal this summer was to implement a reliable and accurate distance measurement algorithm onto individual robots. After investigating the use of Received Signal Strength Indicator (RSSI) as a distance measurement, and experimenting with the algorithm, we discovered that RSSI was only reliable for short distances under 0.5 meters. When we continued our research and experimented with using Time Difference of Arrival (TDOA) as a distance measurement, our experiments yielded much more accurate, and faster, results. We completed our distance measurement experiments earlier than expected, so it gave us time to explore communication protocols, which will be explained further in the Discussion and Conclusions section.

Manuscript received August 2, 2010. This material is based upon work supported by the National Science Foundation under Grant #0755224
Lydia Lei is with the Department of Electrical and Computer Engineering, University of Maryland, College Park, MD 20742 (email: lydialei@umd.edu)
Christopher Perkins is with the Department of Electrical and Computer Engineering, University of Maryland, College Park, MD 20742 (email: cperkins@umd.edu)

II. Background

A. The TI eZ430-RF2500 development board

For this research, we used the TI eZ430-RF2500 development board (Figure 1). The MSP430F2274 microcontroller is the heart of the board, with 32KB + 256B of Flash Memory and 1 KB of RAM. The MSP430 is a 16-bit RISC microcontroller[1]. There are 18 external pins on the board, some of which are extension pins of the MSP430, used for debugging purposes. We can program the MSP430 using the IAR Embedded Workbench Kickstart, freely available online. The board also has a radio chip, the CC2500. The CC2500 functions in the 2400 - 2483.5 MHz frequency band, with a default 200 kHz channel spacing, and is a great option for this research because of its low power characteristics. The board has 20 external pins connected to the radio chip [1].

Fig. 1: The eZ430-RF2500 development tool.

B. Received Signal Strength Indicator

The CC2500 is able to transmit and receive radio packets between the eZ430-RF2500 TI boards. Received Signal Strength Indicator (RSSI) is a method engineers use to measure the power present in a received radio signal [2] and can sometimes be used to indicate distance between a transmitter and a receiver. Elizabeth Kenyon, a student from last summer's UMD CS Scholar program, designed an experiment that proved the concept for using RSSI as a distance sensor on the TI eZ430-RF2500 boards. The results from her experiment demonstrated that the boards were able to use RSSI and measure distance accurately in a 0.5 m range. However, in her experiments, Kenyon incremented the distance of the robots by 1 foot per trial; in order for RSSI to be used on these robots, we needed to be certain that the results would be just as accurate in smaller increments (5 cm).

[1] A 16-BIT RISC architecture has registers that all hold 16 bits.

C. Time Difference of Arrival

There are several other low power methods that are used by engineers today to measure distance, one method being Time Difference of Arrival (TDOA). The idea of TDOA is based on the fact that emitted signals are received at different times if (a) there is one signal and the multiple receivers are at different distances from the signal source or (b) there is more than one signal and the signals are in two different mediums, thus travel at different speeds to the receiver [5]; the robots in this research will be using the latter mechanism. Light and sound travel at different speeds[2]. Similar to how people calculate how far the lightning is from them in a thunderstorm[3], the receiver is able to measure distance by using the time difference of arrival of a RF packet (which travels at the speed of light) and an audio signal, both emitted from one transmitting source.

III. Design Process

A. RSSI

As stated earlier, the distance range for using RSSI in our application is roughly under one meter and in small increments (about 5 cm). To verify that RSSI would function as expected, we set up a simple experiment. We used two test boards, one transmitting a RF signal on two alternating frequencies and the other taking RSSI measurements on those frequencies. We left the receiving board stationary and moved the transmitting board away in 5 cm increments. We found that the RSSI data did not increase monotonically as the distance increased, though some surfaces produced better results than others (Figure 2).

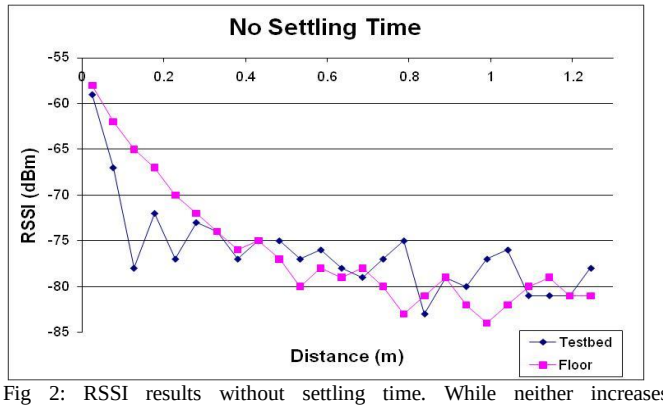

Fig 2: RSSI results without settling time. While neither increases monotonically, the floor yielded better results than the testbed.

[2] speed of light: ~$3*10^8$ m/s, speed of sound: ~343 m/s

[3] "Light travels at 186,000 miles in a second, almost a million times the speed of sound. Sound travels at the slower speed of one-fifth of a mile in the same time. So the flash of lightning is seen before thunder is heard. By counting the seconds between the flash and the thunder and dividing by 5, you can estimate your distance from the strike (in miles)." Using math and assuming that light isn't instantaneous, we find the same results: that distance = time*0.02125 = time / 5, with 'time' equal to 'counted time'. http://weathereye.kgan. com/cadet/lightning/thunder.html

The initial algorithm we used to calculate RSSI was taken from a tutorial, by Thomas Watteyne, on the ez430RF2500 board [1]. Upon examining this algorithm in more detail, we realized that the RF signal was not allowed enough time to settle. Due to this, there was an excessive amount of noise contributing to the final RSSI measurement. In order to remedy this, we modified the algorithm so that there would be enough time for the RF signal to settle before the final RSSI measurement was taken. Even though this resulted in better results (Fig 3), one can see that the RSSI measurement is useful as a distance metric only to a distance of roughly 0.48 meters. Aside from the useful range being shorter than we had hoped for, the new addition of settling time to the algorithm greatly slowed the measurement to about 4.7 seconds per measurement. This inspired us to pursue a different method of distance sensing.

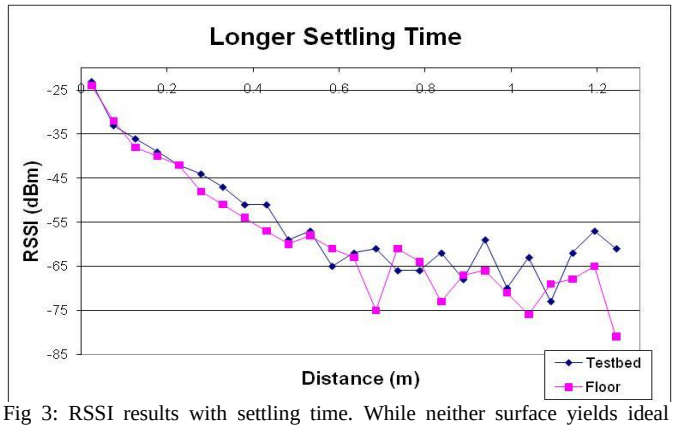

Fig 3: RSSI results with settling time. While neither surface yields ideal results, the settling time obviously improves on the testbed.

B. TDOA Hardware

In order to begin designing our TDOA system, we had to first determine the hardware. Since our end goal of this project is to build a swarm of micro-robots, we had to find hardware that was low power and small enough to fit on the eZ430-RF2500 TI board. After hours of researching, we settled on the CMC-2742PBJ-A, an electret omni-directional microphone from CUI Inc., and the PS1240P02CT3, a piezo buzzer from TDK Corporation; both small and low power instruments. A piezo buzzer is made from two conductors that are separated by piezo crystals. When a voltage is applied to these crystals, they push on one of the conductors and pull on the other. The result of this push and pull movement is a sound wave [3]. An electret microphone is a type of condenser microphone which uses electret, a stable dielectric material that has a permanently embedded static electric charge, to power the microphone - the few volts that are needed are used to power the built-in FET buffer [4]. The bias circuit to power the microphone is shown in Figure 4 below.

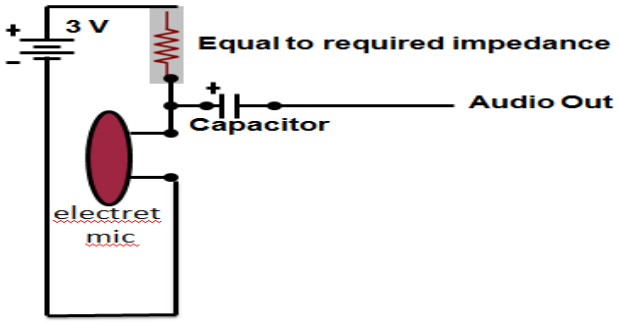

Fig 4: Biasing circuit for the microphone.

After we determined a suitable bias circuit to power the microphone, we found that the output signal of the microphone needed to be amplified in order for our algorithm to see and detect the desired frequency. After prototyping an external amplification circuit on a breadboard, we discovered that there were two embedded operational amplifiers on the MSP430F2274. The on-board op amps offered eight different function controls (OAFCx) as well as pre-programmed feedback resistances (OAFBRx) in their OAxCTL1 register. Since we needed the largest amplification available, we decided on using the MSP430 configured with two non-inverting programmable gain amplifiers, each with feedback resistance ratios of 15 /1, yielding a total gain of 256.

We found that another biasing circuit was needed for the input of the op amps. The input signal was AC coupled around zero, but the MSP430 was powered between 0 and 3V. The only way the entire amplified input wave can be fully visible without clipping would be to center the original input wave between 0 and 3V, offset by its amplitude. In cases where the signal is too large and there is clipping, by having the amplified signal centered between 0 and 3V, the clipping would be even on both sides. The second biasing circuit is shown in Figure 5.

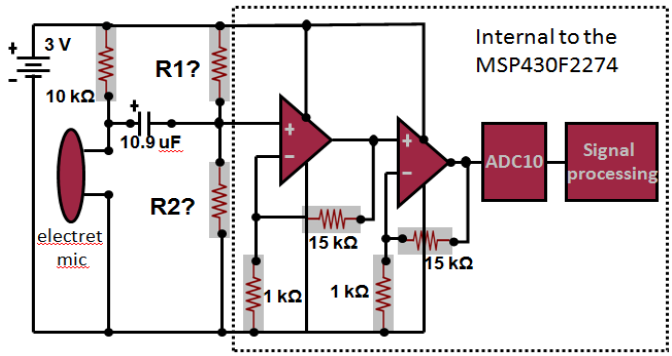

Fig 5: The microphone circuit used to power and amplify the input signal, the biasing resistors unknown.

In order to determine the resistances of the two resistors in the second biasing circuit, we worked backwards and performed the following calculations. To find our offset, we had to first determine the largest peak possible of our unamplified offset input signal. By finding this peak, we are able to determine the amplitude of the offset signal, which will become our offset (Equation 1). From the desired offset, we could then calculate the two needed resistances by using the voltage divider rule (Equations 2 and 3).

$$offset = \frac{2.6}{256 * 2} = 0.005078125\ Volts \quad (1)$$

Given:

$$Vout = 0.005078125\ Volts$$
$$Vin = 2.6\ Volts$$
$$R1 = 12.01\ M\Omega$$
$$Gain = 256$$
$$R2 = ??$$

$$\frac{Vout}{Vin} = \frac{R2}{R1 + R2} \quad (2)$$

$$\frac{R1}{256 * 2} = R2\left(1 - \frac{1}{256 * 2}\right) \quad (3)$$

Solving this equation leads to:

$$R1 = 12.01\ M\Omega$$
$$R2 = 23.502\ k\Omega$$

After finding our two biasing resistors, we were able to complete our microphone circuit (Figure 6):

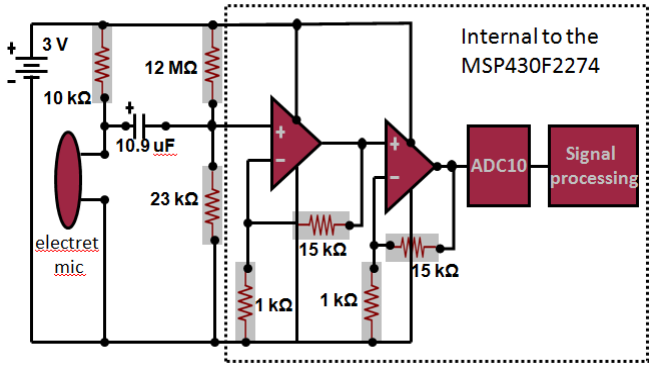

Fig 6: The microphone circuit used to power and amplify the input signal

As for the speaker, we wanted to emit a square wave at a specific frequency. Since all the robots in the swarm would be emitting at their own individual audio frequency, we wrote the algorithm in a way that would be simple to change the emission frequency. After reading through the TI board's data sheet, we found that it was possible to make a square wave using the on-board Pulse Width Modulator (PWM) using Timer-B. Using Timer-B, we were able to specify the period, the duty cycle, and the output mode for the PWM. By just changing a variable in the algorithm, we are able to control the frequency at which the speaker emits.

C. Signal Processing

Once we were able to successfully amplify the audio signal, we were faced with the problem of actually converting the signal to a digital representation and performing digital signal processing to detect when the sound reached the microphone in relation to when the RF packet reached the antenna. The first step was to convert the signal to a digital representation. The MSP430F2274 has an internal 10-bit analog-to-digital converter (ADC10) unit with a data transfer controller (DTC). The DTC is able to transfer the sampled data directly into memory without CPU intervention. We internally connected the output of the op-amps used in the amplification circuit directly to the ADC10. By using the embedded op-amps and ADC unit, we were able to greatly minimize the external components needed to implement TDOA. We programmed the ADC10 to acquire a burst of 255 continuous samples, which is the maximum number that the DTC can transfer at one time.

During our initial experiments, we found that the ADC10 had a sampling rate of roughly 162,000 samples/sec. It may seem that such a high sampling rate would be a good thing, but this rate is actually too fast. At 162,000 samples/sec sound travels only 0.54 m after 255 samples (Equation 4).

$$D = \frac{1}{n} \times \mathrm{N} \times s \qquad (4)$$

n = sampling rate, N = number of samples taken,
D = distance travelled, s = speed of sound

Since we are interested in roughly a 1 meter range, this was not a sufficient distance. In order to solve this problem, we programmed the ADC10 to divide the main clock by 2, which yielded a sampling rate of roughly 80 k samples/sec. At this rate we can get a 1.09 m range, which is ideal for our application. Also, at 80,000 samples/sec the Nyquist rate[4] is roughly 40,000 samples/sec, which is plenty considering our microphone and speaker combination has a bandwidth of roughly 9 kHz to 20 kHz. We could further slow the clock rate of the ADC10 and increase the distance range, but this would be at the cost of distance resolution. Ultimately, the sampling time defines our distance resolution, so the slower our sampling rate, the more coarse our distance resolution (see Fig 6).

After we decided on a sampling rate, the next step was to find a way to analyze the 255 sample block so that we could determine precisely when the sound had reached the microphone. We decided on using an algorithm called the Goertzel algorithm (see appendix A). The Goertzel algorithm is essentially a second order digital filter which uses a single Fourier coefficient to isolate the energy present in a signal at a specified frequency [8]. This algorithm is sometimes used in

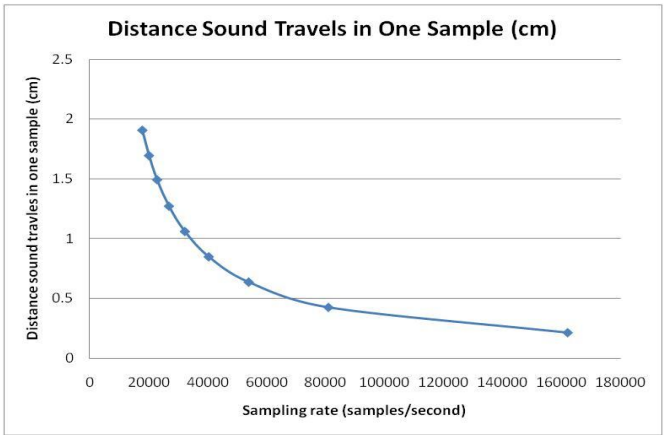

Fig 6: The distance sound travels in a sample at a given sampling rate.

touch-tone phones to identify which buttons have been pressed [9]. Essentially, the Goertzel algorithm analyzes a predetermined number of samples (sample window) and outputs one value representing the energy present in that window at the specified frequency. Fig 7 and 8 show a simulated example of what the Goertzel algorithm tuned to 12 kHz will output when given an input signal of silence, and then a pure sin wave of 12 kHz.

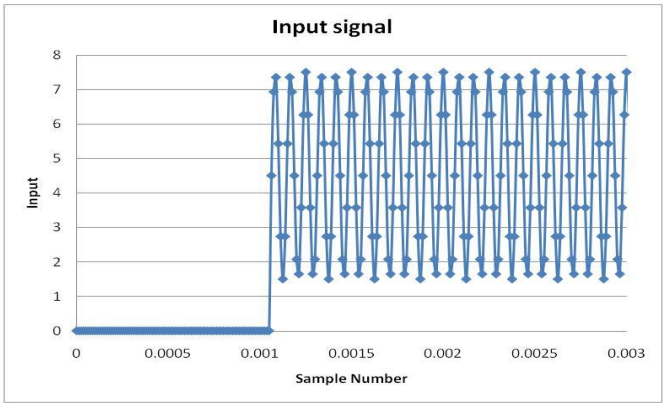

Fig 7: The input signal from Fig 8.

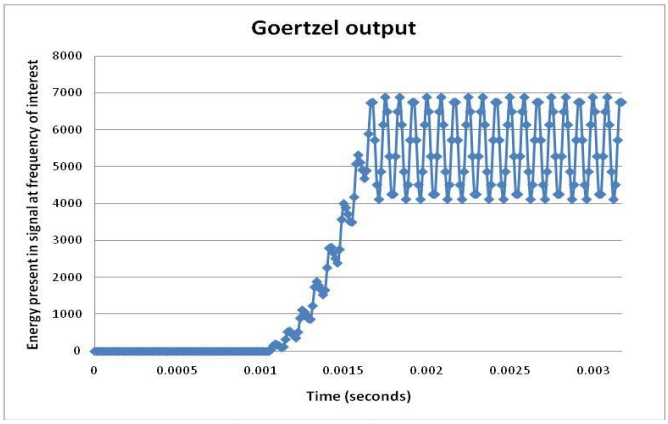

Fig 8: The Goertzel output from the input of Fig 7.

[4] A lower bound on the sampling rate required to prevent aliasing.

We also applied a variation of the Hamming window function (see appendix A) to the Goertzel filter in order to smooth the overall curve of the Goertzel filter's frequency response. The bandwidth of this response is determined by the sampling rate, the size of the sample window used, and the total number of samples used in the windowing function. We want the bandwidth to be small enough to isolate specific frequencies, and minimize the response to other frequencies since the intended use of this technique is in a multi-robot environment. We plan to use a different frequency for each robot. We implemented our system using 4 frequencies at 9 kHz, 12 kHz, 15 kHz, and 18 kHz. We chose a 50 sample window size which yields roughly a 3 kHz bandwidth (see Figure 9). Because we have a 3 kHz gap between the frequencies of interest, having a 3 kHz bandwidth is sufficient to distinguish the 4 frequencies of interest. In order to save on processing power we pre-computed all of the Goertzel and Hamming coefficients and stored them in the on-board flash memory.

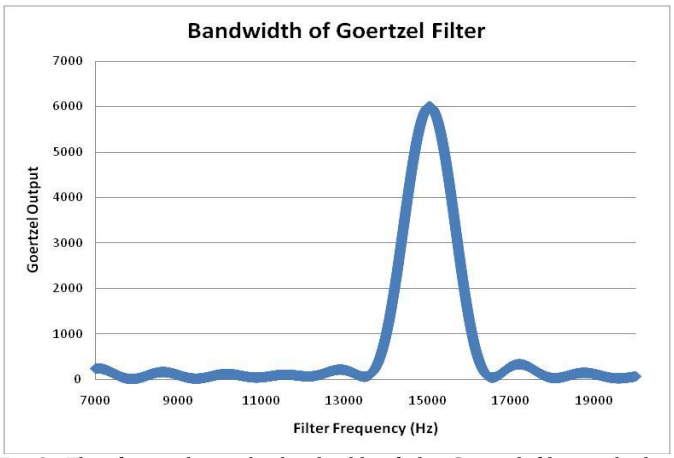

Fig 9: This figure shows the bandwidth of the Goertzel filter with the Hamming window applied at an 80 k sample/second sampling rate with a window size of 50 on a total of 255 samples when the filter is tuned to 15 kHz.

Just having a working Goertzel filter is not enough to implement TDOA distance sensing. The signal must be examined, and a decision must be made as to whether the frequency of interest is present, and if so when it became present. In order to process the entire audio signal, the Goertzel filter is applied to samples 0 – 49, then shifted and applied to samples 1 – 50, and so on. If the Goertzel output meets a certain criteria, then it is considered a hit. When a hit occurs, the first sample in that window is considered to be the moment that the sound arrived at the microphone. For example when samples 100 – 149 are examined, and a hit is found, we consider the sound to have arrived at sample 100. Another complexity to finding the time of arrival is that the Goertzel output is slightly periodic in itself (as can be seen in Figure 8) and the magnitude of the Goertzel output is related to the magnitude of the input signal. Because of this, it is not possible to consider all outputs above a predetermined threshold to be a hit. In order to remedy this problem, our signal processing algorithm begins keeping track of the first derivative of the peaks in the output once a certain threshold is passed (i.e. threshold = $2.5*10^6$). As soon as the derivative of the peaks falls below a different derivative threshold (i.e. threshold = 0) we have found a hit (Figure 10). Once the sample number of the hit is known, the time of arrival of the sound is known, and a final distance can be calculated.

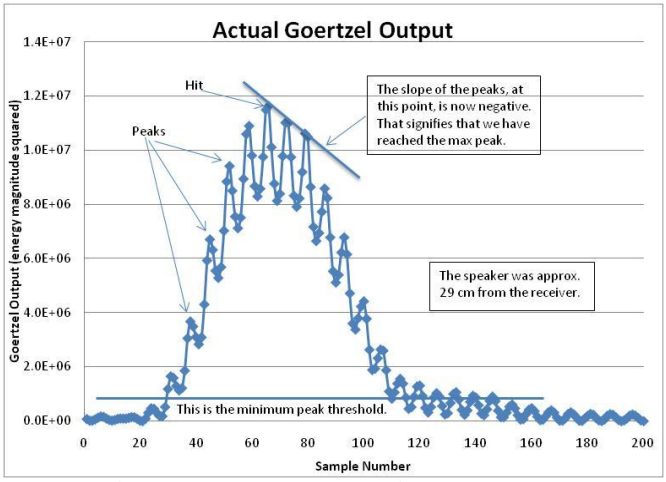

Fig 10: This figure illustrates the processes of determining when a hit takes place. This is demonstrated on actual output from our implemented system.

Once a hit can reliably be determined, the TDOA distance sensing approach can be quite easily implemented in full. In the TDOA exchange, there will be a transmitting robot, and any number of receiving robots. The transmitting robot will issue a RF signal containing information on which audio frequency it will emit. Once the other robots receive this RF signal, they enter an interrupt routine. As soon as they enter the interrupt routine, they switch to the appropriate Goertzel filter, acquire the audio samples, process them, and calculate their distance from the sender.

We implemented the described TDOA system on two eZ430-RF2500 TI boards and performed a basic experiment. One board was set to listen for the TDOA broadcast and another was set to broadcast the TDOA signals. The transmitting board was moved incrementally further away from the receiving board at 5 cm intervals. Three consecutive measurements were taken at each interval. It is obvious from Figure 11 and 12 that implementing TDOA in this way yields desirable results. In Figure 11, the slope of the trendline is 233.55. Multiplying the slope by the conversion factor of 0.0042875 meters/sample found via Equation 5 results in a value of 1.001346.

$$C_f = \frac{1}{sps} \times sos \qquad (5)$$

Where: C_f = conversion factor
sps = sampling frequency
sos = speed of sound

The distance values in Figure 12 were calculated using this conversion factor, as is evident by the slope of the trend line. Theoretically, this slope should be 1, as the actual distances should match precisely with the measured distances. In reality there is a 0.13% degree of error in the slope of the line, as well as an offset evidenced by the non-zero y-intercept. For our application, this percent error is excellent. This error could most likely be further reduced by improving the hit detection algorithm which was explained earlier. As for the offset, this is most likely a result of latency between the receipt of the RF signal and the beginning of the sampling burst. The effects of this latency could easily be counteracted by adding a delay between the sending of the RF packet and emission of the audio signal.

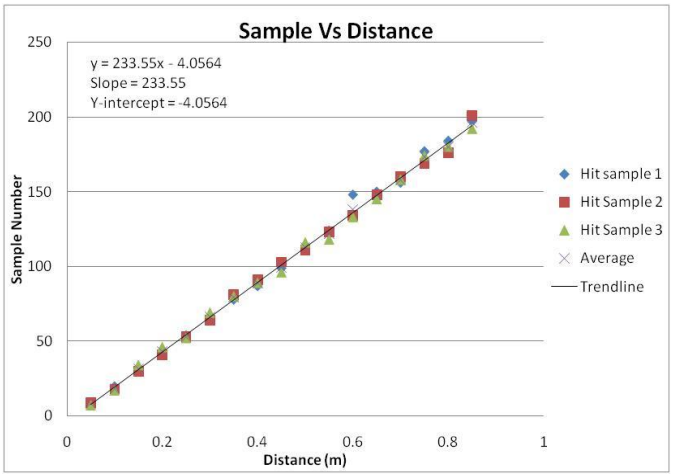

Fig 11: Sample number vs. actual distance for TDOA measurement.

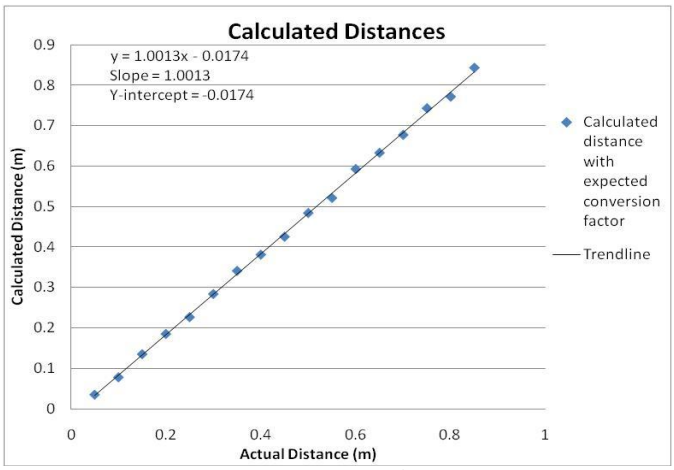

Fig 12: Calculated distance vs. actual distance for TDOA measurement.

IV. Discussion and Conclusions

If the RSSI algorithm could be optimized to execute faster than it currently does, it would be a very useful close range distance sensing method. A robot already using wireless communication would not need any additional hardware to implement RSSI distance sensing. Because the RSSI distance sensing could be integrated into the communication protocol, it could possibly be implemented with very minimal power overhead. Further investigation into distance sensing with RSSI seems to be a reasonable direction.

Our implementation of a TDOA distance sensing technique using wireless RF packets and a piezo buzzer has, thus far, been quite successful. Overall, this technique has an acceptable range of up to approximately 0.9m and acceptable performance with regards to accuracy. While we have designed an implementation that works, there is still much that needs to be done to improve this technique. For instance, the algorithm was implemented using floating point numbers. If the floating point arithmetic could be converted to fixed point arithmetic, it is possible that the speed of the algorithm could be greatly enhanced. The MSP430f2500 does not have dedicated multiplication hardware, so when multiplications occur, they are somewhat inefficient to execute. There are certain models of the MSP430, though, that do contain dedicated multiplying circuitry as well as much more on-board memory. If this TDOA technique were to be used as the main means of distance sensing in a small robotic swarm, it would be sensible to explore more advanced versions of the MSP430.

One of biggest obstacles with successfully implementing either the RSSI or TDOA methods will be overcoming issues with directivity and line-of-site. In our experiments, we have noted that the results of both systems are heavily influenced by the orientation of the two boards. This is more of a problem in RSSI, since the strength of the signal itself directly yields the distance measurement. In this aspect, TDOA is much more robust. Our implementation of TDOA does not depend on the strength of the audio signal received; it merely searches for the presence of that specific signal. We did note, though, that some configurations of microphone and speaker orientation were far from ideal (i.e. both speaker and microphone oriented up). Also, in a real life scenario, robots will have to be able to deal with objects blocking these signals altogether. Hopefully, since these robots are intended to be acting in a swarm, they will be able to deal with the line-of site problem via some sort of communication protocol or control theory.

Aside from continuing to work towards more optimized distance sensing, future work on this project involves integrating these sensing techniques with movement control algorithms which were developed by previous MERIT BIEN students. We have currently proposed a rudimentary inter-robot communication protocol, which will assist in the successful implementation of these movement control algorithms on a small swarm of robots built on the eZ430-RF2500 development boards (see appendix B for state diagram). This communication protocol strives for simplicity and efficiency so as to expedite the integration of the distance sensing systems with the movement control algorithms on

actual robots. There is no leader in our protocol, and all robots communicate in a turn based system. One of our main focuses was to ensure that the group as a whole could adapt to the addition and removal of robots without becoming unsynchronized and confused. A common problem with turn based systems is when a collision occurs (when two robots claim the same spot without opposition). Our protocol handles the issue of collisions, with the exception of adding two new robots at the same exact time. This is a problem we are currently looking to amend.

In summary, while the RSSI distance sensing technique seems, at the moment, to be unreliable, we have developed a TDOA distance sensing technique which has shown promising results. With the optimization of these techniques and the realization of our proposed inter-robot communication protocol, this project is well on its way towards implementation of a coordinated swarm of small low-power robots.

Acknowledgements:

The authors would like to thank the National Science Foundation CISE award #0755224 for funding this research, the MERIT program at the University of Maryland for this opportunity, and finally Dr. Pamela Abshire, Andy Turner, Timir Datta, and Dave Sander for their continual support, mentorship, and guidance during the course of this research.

References:

[1] Watteyne, Thomas, *eZWSN: Experimenting with Wireless Sensor Networks using the eZ430-RF2500*, http://cnx.org/content/col10684/latest/

[2] Hong, Sung-Hwa, Byoung-Kug Kim, and Doo-Seop Eom, *Localization Algorithm in Wireless Sensor Networks with Network Mobility*, IEEE Transactions on Consumer Electronics, pp 1921-1928, November 2009

[3] *Piezoelectricity*, http://www.americanpiezo.com/piezo_theory/index.html

[4] Engdahl, Tomi, *Powering Microphones*, http://www.epanorama.net/circuits/microphone_powering.html

[5] Kuile, Alexander ter, *Multilateration & ADS-B*, http://www.multilateration.com/surveillance/multilateration.html

[6] Chen, Chiouguey J., *Digital Signal Processing Solutions – Semiconductor Group*, Texas Instruments Application Report, June 1996

[8] Toledo, Sivan, *Perfect Pitch: An Accurate-Pitch Sensor and Sounder*, http://www.cs.tau.ac.il/~stoledo/lego/msp430-perfect-pitch/

[9] Selman, Suvad and Paramesran, Raveendran, *Comparative Analysis of Methods Used in Design of DTMF Tone Detectors*, IEEE Conference on Telecommunications and Malaysia Conference on Communications, pp335-339, 14-17 May 2007

[10] Banks, Kevin, *The Goertzel Algorithm*, EE Times Design, http://www.eetimes.com/design/other /4024443/The-Goertzel-Algorithm

[11] Microstar Laboratories, Inc., *Detecting a Single Frequency Efficiently*, http://www.mstarlabs.com/dsp/goertzel/goertzel.html

Appendix A

Below are the equations used in performing the Goertzel Algorithm and the equations used in finding the coefficients of the Hamming window function.

The Goertzel Algorithm:

k = the discrete Fourier coefficient

$$k = \mathrm{N} \times \frac{\mathrm{fi}}{\mathrm{fs}}$$

Where: N = number of samples in window,
fi = frequency of interest,
fs = the sampling frequency,

The recursive equation:

$$v_k(n) = 2 \times \cos\left(\frac{2 \times \pi \times k}{\mathrm{N}}\right) \times v_k(n-1) - v_k(n-2) + x(n)$$

Where: n = 0, 1, …, N [7]

Final output:

$$v_k(n)^2 + v_k(n-1)^2 - v_k(n) \times v_k(n-1) \times coef$$

Where $coef = 2 \times \cos\left(\frac{2 \times \pi \times k}{\mathrm{N}}\right)$ [10]

Hamming Window Formula:

Coefficients are found by:

$$0.54 - 0.46 \times cos(\frac{2 \times \pi \times n}{N})$$

Where: n = 0, 1, …, N
N = window size [11]

Appendix B

This is the state diagram for our communication protocol. With this protocol, the robots will be able to detect when a new robot is added and when one is taken away.

Communication Protocol State Diagram: