# ABSTRACT

Title of Thesis: MIXED-SIGNAL SENSING, ESTIMATION AND CONTROL FOR MINIATURE ROBOTS

Michael J. Kuhlman, Masters of Science, 2012

Thesis directed by: Professor Pamela Abshire
Department of Electrical and Computer Engineering

Control of miniature mobile robots in unconstrained environments is an ongoing challenge. Miniature robots often exhibit nonlinear dynamics and obstacle avoidance introduces significant complexity in the control problem. In order to allow for coordinated movements, the robots must know their location relative to the other robots; this is challenging for very small robots operating under severe power and size constraints. This drastically reduces on-board digital processing power and suggests the need for a robust, compact distance sensor and a mixed-signal control system using Extended Kalman Filtering and Randomized Receding Horizon Control to support decentralized coordination of autonomous mini-robots. Error analysis of the sensor suggests that system clock timing jitter is the dominant contributor for sensor measurement uncertainty. Techniques for system identification of model parameters and the design of a mixed-signal computer for mobile robot position estimation are presented.

# Mixed-Signal Sensing, Estimation and Control for Miniature Robots

by

Michael Joseph Kuhlman

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Masters of Science
2012

Advisory Committee:
Professor Pamela Abshire, Chair/Advisor
Professor Sarah Bergbreiter
Professor Timothy Horiuchi
Professor P.S. Krishnaprasad
Professor Nuno Martins

# Acknowledgments

This research project would not have been possible without the support of many people. First and foremost I would like to thank my advisor, Professor Pamela Abshire for offering me this diverse research opportunity on the ANTBOTS Project. I thank her for her support and guidance over the past two years. I would also like to thank the faculty that comprise my Advisory Committee. I have collaborated with Professor Nuno Martins and Professor Sarah Bergbreiter on the ANTBOTS Project for the past two years. I also extend my thanks to Professor P.S. Krishnaprasad and Professor Timothy Horiuchi. I first met them both through the MERIT program as one of their undergraduate research interns and have taken the graduate-level classes they offer. The faculty that comprise the Advisory Committee have given me valuable insights into the Engineering Process and have shaped my career interests.

Building an entire autonomous robotics platform from the ground up is a serious undertaking. This project was in collaboration with several other labs on campus so I would like to mention the other graduate students that helped me. I would like to extend my thanks to George Gateau from the Micro Robotics Laboratory who came up with the prototype design for the Walle bot, the research platform used throughout the majority of my time on the project. I also thank Eduardo Arvelo from the Cooperative Autonomy Laboratory who helped me formulate some of the estimation problems faced on the robotics platform. Timir Datta from the Integrated Biomorphic Information Systems (IBIS) lab assisted me on many of my circuits problems. Tsung-Hsueh Lee from the IBIS lab has also been

helpful with the development of digital systems. I also thank the rest of the IBIS lab for the random insightful discussions I've had with them about life in graduate school in general. Last but not least, I would like to thank Greg Gremillion in the Autonomous Vehicles Laboratory for granting us access to his lab's 3D printer.

I have also had assistance from many talented undergraduate researchers and interns. Those that have been the most helpful in getting this project off the ground were Stacy Hand, Christopher Perkins, Matthew Phipps, Andrew Sabelhaus, David Shiao, George Sineriz, Ken Tossel, and Yuchen Zhou. I would also like to thank the remainder of the ANTBOT team.

I would also like to acknowledge help and support from some of the ECE and ISR staff members. Gwen Flaniski and Alex Cotsalas from the ECE Computer Helpdesk have kept the computers running smoothly and Victoria Berry from the ECE Business Office handled the majority of my parts orders. I thank Shyam Mehrotra for technical assistance and the IREAP and Physics Machine shop specialists for parts production for the distance-only sensor. I also thank Dr. Judith Bell and the MERIT program, since my positive research experiences during those summers as an undergraduate led me to come to the University of Maryland for graduate school.

I would like to acknowledge the financial support I have received. The material in this thesis is based upon work supported by the National Science Foundation under Award Nos. 0647321, 0755224, and 0931878, and the ONR AppEl Center at UMD.

I should also thank my cat for being diligent and waking me up on time every

morning to receive her breakfast ham. Finally and most of all, I would like to thank my dear parents for their unwavering support throughout this endeavor. They have always been there for me in times of need.

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

## 1.1 Motivation

Important tasks for robotic systems include surveillance and search and rescue. For example, snakelike robots such as the Active Scope Camera [1] can search through rubble for earthquake victims. However, this is not the only possible system design for the task of searching for earthquake victims. Suppose for example one could possess an entire fleet of micro-robots (less than 1 cm per side) that could fit in a small container. These swarming robots could then be dropped into the rubble, can search autonomously and report back if they find any survivors.

Such a decentralized swarm would be robust and if the robots are manufactured using processes similar to CMOS or MEMs could be very cheap in mass quantities. Preliminary systems combining such technologies have already been presented [2]. Such systems have been conceptualized as early as 1987 by A.M. Flynn [3] but have not yet come to fruition. One primary challenge is that the strict size and power constraints imposed by the system dramatically reduce the available on-board processing power to be able to solve the ever present problems of sensing, estimation, and control for autonomous systems. We will investigate mixed signal computing concepts to see whether or not they satisfy the strict design constraints for miniature robots.

Control of miniature mobile robots in unconstrained environments is an ongoing challenge. Miniature robots often exhibit nonlinear dynamics and obstacle avoidance introduces significant complexity in the control problem. In order to allow for coordinated movement such as following a leader or moving in simple formations, the robots must know their location relative to the other robots; this is challenging for very small robots operating under severe resource constraints in the absence of specialized environmental sensors. This suggests the need for a robust, compact distance-only sensor to support decentralized coordination of autonomous mini-robots.

We propose that the first step to realizing collective, cooperative behavior in miniature robot swarms is to demonstrate simple formation following such as following a leader robot using minimal sensing. The organization of this thesis tracks the development and deployment of this proposed distance-only sensor on a miniature robot implementing a leader rendezvous algorithm. The first platform deploying this sensor, the Walle bot in Fig. 1.1 was first showcased in the 2011 International Symposium on Circuits and Systems (ISCAS) as a proof of concept with limited capabilities [4]. Rigorous analysis over the following year suggested significant system improvements at the algorithmic, architectural and hardware levels to improve performance and level of autonomy with reduced cost. The thesis culminates in the design of the next generation robotics platform to prototype mixed signal architectures.

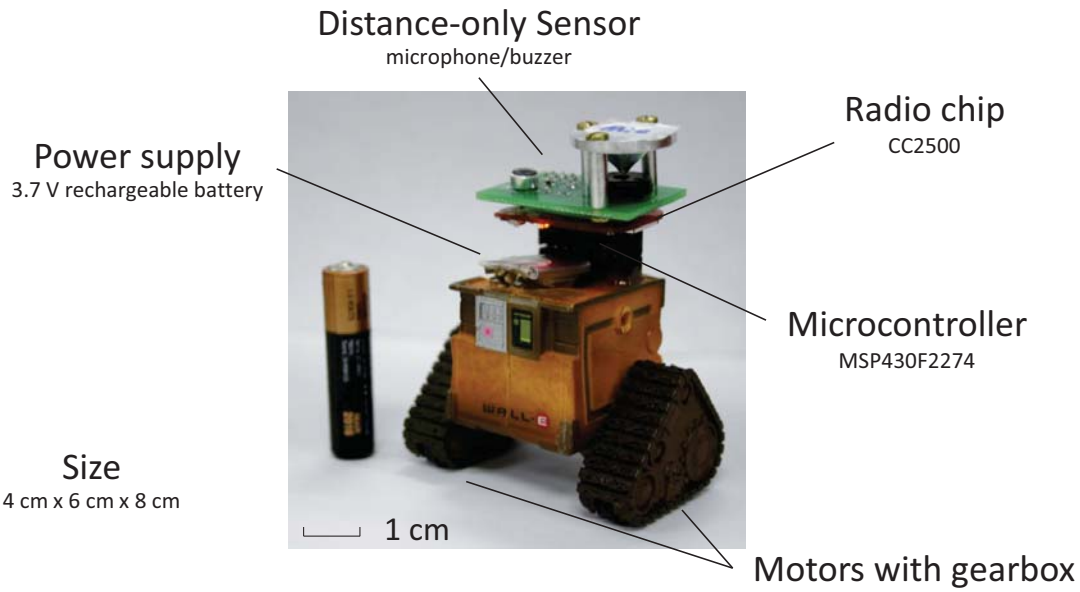


Figure 1.1: Miniature robot platform used in the majority of experiments: the Walle bot platform. 'AAA' battery shown for reference.

## 1.2 Background

Great advances have been made towards achieving autonomous mini-robots that are able to coordinate and communicate with one another in a smooth fashion [5]. In order to allow for coordinated movement such as following a leader or moving in simple formations, the robots must know their location relative to the other robots; this is challenging for very small robots. A variety of location systems have been developed for wireless sensor networks [6], [7], [8] or small robotic platforms [9]. In this context, most of the existing systems are poorly matched to the size, range, and desired resolution of distance sensing for the mini-robots, 1 in, 1 m, and 1 mm respectively. For example, Received signal strength indication (RSSI) based distance estimation has only been shown to be feasible in idealized settings [6], and typical

variability is on the order of meters which would be completely useless in controlling a swarm of mini-robots. TDOA-based distance estimation is more accurate, but existing implementations such as [10] include transducer arrays, greatly increasing sensor size. In Chapter 2 we propose a design for a TDOA distance-only sensor that requires fewer components and is more suitable to miniature robotic platforms, at the cost of losing directional specificity which can be mitigated by including a heading estimator.

## 1.3 Project Specification and Motivation for Subsequent Work

The objective of the walle bot platform is to design a resource-constrained miniature mobile robot with minimal sensing capable of simple formation following. We will focus on developing an algorithm that allows robots to follow a (stationary) leader. The only sensor on the platform is a distance-only sensor, where a single distance measurement does not provide sufficient information regarding the leader's relative location. Distance-only sensing also requires heading estimation for relative positioning using polar coordinates: $(d, \theta)$. There are no wheel encoders for odometry. However, position estimation by simulating system dynamics is required for the heading estimator and will replace odometry. The robot shown in Fig. 1.1 used the chassis of a toy robot (on board electronics consisting of IR remote control removed) to reduce system cost and make robot construction simple. We will see later that the limitations of the walle bot platform will provide insight into developing a more advanced and capable system.

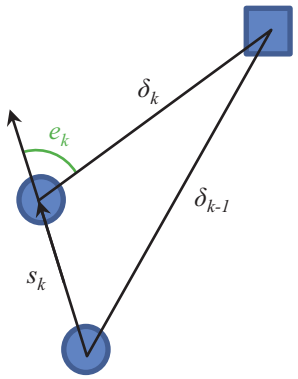


Figure 1.2: Heading estimation triangle construction

### 1.3.1 Heading Estimation

$$\delta_{k-1}^2 = \delta_k^2 + s_k^2 - 2\delta_k s_k \cos(\pi - e_k) \tag{1.1}$$

Using the law of cosines (1.1), one can compute the cosine of the angle error $e_k$. $\delta_k$ is a distance measurement taken at time $t_k$, and $s_k$ is the magnitude of displacement the robot has traveled between $t_{k-1}$ and $t_k$. Note that this definition of the heading error $e_k$ assumes that the robot is moving along a straight line which may not be the case (the robot generally moves along an arc). Given the geometry of the situation and the even symmetry of the cosine function, one cannot determine which side of the follower the leader is on (it could be either $e_k$ degrees to the left or to the right).

However, including multiple measurements and tracking the control signals while the follower robot is moving (i.e. is the robot turning left or right?), one can determine the whether the leader is to the left or the right of the follower. A difference equation sign check on (1.2) can determine whether the robot is turning

to or away from the leader (one wants to drive $e_k$ to zero or $\cos e_k = 1$). If $q_k > q_{k-1}$, then the robot should change its turning direction.

$$T_{turn} \propto q_k \tag{1.2}$$

$$\text{where } q_k = 1 - \cos e_k$$

### 1.3.2 Leader Rendezvous Algorithm

In order for the robots to rendezvous with the leader, they followed a simple strategy to drive the relative heading angle between their facing direction and the location of the leader to 0 degrees (directly in front of them). As the robots continued to move forward, they would eventually rendezvous with the leader.

1: **procedure** LEADERRENDEZVOUS
2: Take distance measurement
3: If new distance is less than 20 cm, stop! You have arrived
4: Calculate new heading error, control variable $q_k$, and turning direction
5: Turn for $T_{turn}$ seconds
6: Move forward 20 cm
7: Stop and wait for next audio pulse (go to 2)
8: **end procedure**

Fig. 1.3 shows an example of a robot trajectory using this method to attempt to rendezvous with its leader as captured by an overhead vision system. The ideal heading estimation triangles have been superimposed onto the trajectory, representing the information available to the robot at a given time. This is conceptually similar to using P control for steering laws. Driving $q_k$ to 0 will drive $e_k$ to 0. However, P control cannot be directly implemented due to chassis slip restrictions, hence why this algorithm uses a variable-time turn phase and a forward phase These slip

conditions limit the region of reliable motor controls in the control space. Turning with only one motor on, or moving forward with both motors at a comparable speed to move forward are within this desired region of operation.

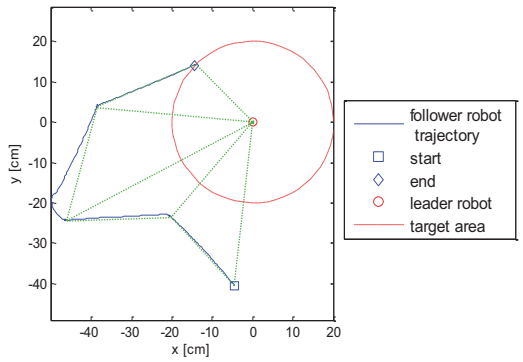


Figure 1.3: Sample robot trajectory with ideal heading triangles (dotted green) overlaid

We will later discuss some of the system limitations of this system in subsequent chapters.

### 1.3.3 Challenges in the design of the Next Generation Robot

Limitations of this original demonstration suggest five areas of research that are pertinent to developing miniature, autonomous vehicles.

- *Accurate distance-only sensing*: This requires an understanding of the necessary precision required for control algorithms. Error propagation models can suggest what sources of noise factor into measurement uncertainty, which in turn can suggest how measurement uncertainty can affect system-level performance and suggest design solutions to improve sensor accuracy and system

performance.

- *Odometry without wheel encoders*: Oddly, this seemingly straightforward task often taken for granted comes with several practical challenges. It requires more stringent system specifications and improvements in motor control. System identification or motor calibration is essential in making odometry accurate. Analyzing the accuracy of odometry coincides with analyzing the accuracy of the law of cosines heading estimator.

- *Robust estimation techniques*: The original heading estimator merely calculated the heading angle between the follower and a leader. Looking forward, understanding the robot's location in Euclidean space $(x, y, \theta)$ is critical for implementing more elaborate control systems. Extended Kalman Filtering (EKF) is the *de facto* estimation technique for fusing multiple sensing modalities (in our case, odometry with distance-only sensing) for system state estimation. A more rigorous treatment of how errors in sensing map to errors in heading estimation will provide a framework of analysis tools useful for evaluating more elaborate estimation techniques such as EKF.

- *Generalized control algorithms*: We would like to use techniques from Model Predictive Control (MPC) for control of nonlinear systems. Once a reasonable state estimate is achieved, the system can be controlled. The theoretical background concerning MPC is for the most part beyond the scope of this thesis.

- *Power consumption and analog computing*: The size constraints of the platform impose severe power and computational constraints on the robot. This suggests reducing the number of multiplications required for digital signal processing (DSP). Looking ahead, the system constraints suggest the use of mixed-signal analog computation and analog signal processing. A transistor-level simulation of a VLSI analog computer for position estimation suggests a specific design approach for mixed-signal robotics problems. Using this approach, a system-level VLSI implementation of an Extended Kalman Filter for mobile robots will be discussed.

The findings stemming from this research has been applied to the design of the next generation of robots: the KEPLR platform (KEPLR stands for Kalman Estimator PLanetary-gearbox Receding-horizon-control[1]). Several KEPLR variants are anticipated given the long term research goals of this project. The current system being designed, KEPLR-D (digital) is a fully digital system (excluding the analog front end of sensors). The system has been compartmentalized to allow future variants to substitute VLSI analog computing elements for previously made digital solutions: KEPLR-M (mixed-signal). Components can thus be incrementally designed in VLSI and tested immediately, reducing dependencies in the design process.

[1]Historically, the laboratory has named robot platforms after their type of drivetrain. This choice of acronym was most influenced by the use of planetary gearboxes in the chassis design

# Chapter 2

# High Resolution Distance Sensing for Mini-Robots using Time Difference of Arrival

## 2.1 Summary

This chapter presents an efficient, compact, and robust distance-only sensor for networked small robotic platforms with wireless communication and signal processing capabilities. The sensor determines inter-robot distances by measuring the Time Difference of Arrival (TDOA) between wireless radio frequency packets and audio pulses. Computational overhead has been reduced by an order of magnitude from our previous signal processing technique using the Goertzel Algorithm [4, 11] while the sensor resolution has been improved to 0.27 cm over a range of 75 cm, compared to a previous resolution of 1.1 cm. Error analysis identified timing jitter as the dominant contribution to measurement error.

## 2.2 Introduction

Distance-only sensors offer miniature robots such as the one shown in Fig. 1.1 a means to determine the location of other robots in a swarm. The TDOA sensor is designed using a TI eZ430-RF2500 wireless development board, which is also used for communications and motor control, with auxillary audio components

as shown in Fig. 2.1. The eZ430 includes an MSP430 microcontroller and a CC2500 wireless radio chip. The open source package SimpliciTI provided support code consisting of a minimal RF interface (MRFI) and a board support package (BSP) which provided a framework for code development [12, 13]. An auxiliary TDOA board consists of an omnidirectional microphone (CMC-2742PBJ-A, CUI Inc.), a piezo buzzer (PS1240P02CT3, TDK), bias capacitors and resistors, and headers. It is connected to two internal cascaded non-inverting op-amps of the MSP430 with a total gain $A_V = 130\ \frac{\mathrm{V}}{\mathrm{V}}$. Each board has both a microphone and piezo buzzer for bi-directional distance sensing.

The transmitting board (left, Fig. 1.1) produces an audio signal through pulse width modulation by utilizing an on board timer. The 12 kHz audio pulse is emitted by the buzzer. The receiving board (right, Fig. 2.1) first receives the wireless packet which triggers the microphone to start recording. This analog signal is stored digitally and filtered. Peak detection techniques extract the relevant features of the signal to measure the relative arrival times of the RF packet and audio pulse. A sound reflecting cone similar to previous work [14] improves the directional insensitivity of the sensor.

The choice of using a carrier frequency of 12 kHz was selected during previous work [4]. The original guidelines for the buzzer/microphone selection required small component size and (microphone) directional insensitivity for compact, distance-only sensor. Components in the audio band were found to be smaller than comparable ultrasonic components.

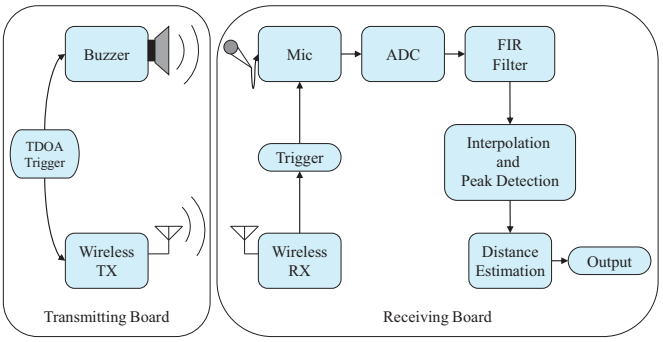


Figure 2.1: TDOA system flow chart.

## 2.3 Signal Processing

When the wireless signal is detected by the receiving board, the audio pulse is amplified and then sampled by the 10-bit Analog-to-Digital converter (ADC) internal to the MSP430 at a sampling frequency of $F_s \approx$ 86-91 kHz using the on board Data Transfer Controller to reduce system overhead [15, 16]. 255 samples are acquired to provide a $\sim$3 millisecond window for capturing the sound impulse within a $\sim$1 m distance. Larger sampling windows are not needed due to significant attenuation of the audio pulse beyond $\sim$0.8 m.

The complete discrete-time sound waveform is filtered using a 32nd order discrete-time FIR filter designed using the Parks-McClellan algorithm in MATLAB [17, 18] to remove environmental noise. It was implemented using Horner's method [19] to to reduce the filter's number of multiplications on a microcontroller without a hardware multiplier. A bandpass filter with $f_{center} = 12$ kHz and 3-dB passband edge frequencies of $f_p = 12 \pm 1.3$ kHz was designed to pass the 12 kHz sound pulse emitted by the transmitting board, encompassing the range of frequencies observed during experimentation. TI supplies application software that will directly convert

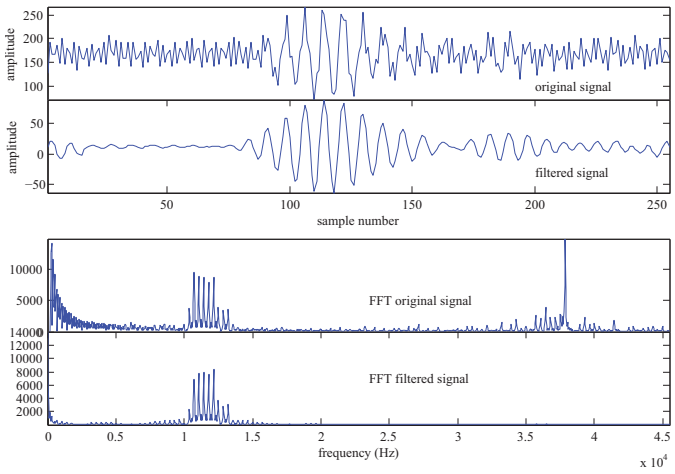


Figure 2.2: Results of MATLAB signal processing: Original Signal, FFT of Original Signal, Filtered Signal, FFT of Filtered Signal

specified FIR filter characteristics (prototyped in MATLAB for example) to efficient assembly code that can be accessed through C functions [19]. This variance is caused by timing jitter in the system clock. In Fig. 2.2 a sampled signal, along with its Fast Fourier Transform, is displayed before and after filtering.

## 2.4 Distance Estimation

In order to measure distance a unique signal feature must be extracted. The detected audio pulse has a carrier frequency of 12 kHz, and the unique feature is taken to be the peak of the signal. While many sonar systems use the leading edge of the audio pulse as a feature [20], this work uses the signal peak because of its robustness to noise and simplicity of detection. One can either locate the peak of the signal or the peak of the signal envelope. Using the signal peak the resolution would be limited to approximately the wavelength of the carrier signal. For the

speed of sound in air at sea level (343 m/s) this resolution is 2.9 cm. However, the signal envelope cannot be directly measured and must be approximated.

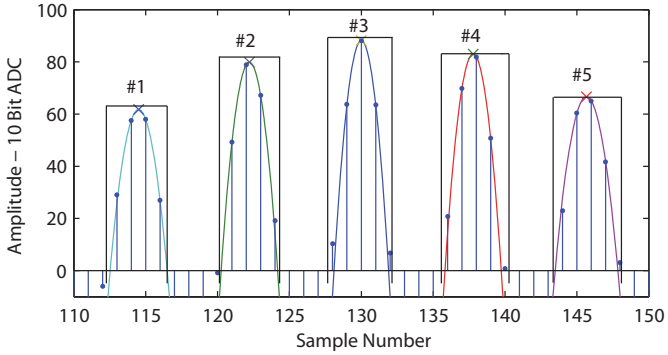


Figure 2.3: Enlarged view of a sample audio pulse displaying data regions used for different methods of interpolation, with the maximum amplitude in the center.

To provide sub carrier wavelength resolution, we used Lagrange polynomial interpolation [21] to fit parabolic functions to the signal peaks, approximating the signal envelope between the crests of the carrier signal. The choice of parabolas intuitively makes sense because the behavior of the Taylor Series expansion of the signal envelope around the signal peak should be dominated by the quadratic term. Each interpolation uses three points from the signal denoted as $(t, x) = (t_1, t_2, t_3, x_1, x_2, x_3)$.

The closed form solutions to the 2nd order polynomial fitting $(t, x)$ is found in (2.1) using Lagrange Interpolation [21]:

$$p_2(t) = x_1 \frac{(t-t_2)(t-t_3)}{(t_1-t_2)(t_1-t_3)} + x_2 \frac{(t-t_1)(t-t_3)}{(t_2-t_1)(t_2-t_3)} + x_3 \frac{(t-t_1)(t-t_2)}{(t_3-t_1)(t_3-t_2)} \qquad (2.1)$$

$$\dot{p}_2(t) = x_1 \frac{2t - (t_2 + t_3)}{(t_1 - t_2)(t_1 - t_3)} + x_2 \frac{2t - (t_1 + t_3)}{(t_2 - t_1)(t_2 - t_3)} + x_3 \frac{2t - (t_1 + t_2)}{(t_3 - t_1)(t_3 - t_2)} \tag{2.2}$$

solving for when $\dot{p}_2(t) = 0$ as in (2.2) will determine $t_{peak}$. Due to the convexity of the problem, root-finding methods can be used such as the bisection method, fixed point iteration or Newtons method [21] but we opted for closed-form solutions for $t_{peak}$ given that the solution is a rational function. A more thorough analysis may suggest that these different methods can allow trade-offs between accuracy and performance.

However, there is ambiguity in selecting the input parameters $(t, x)$ from the audio signal. Given the known pattern of the signal, we have developed two techniques to address ambiguity and to better represent the signal envelope:

*Iterative Interpolation:* For a local peak, interpolate the signal locally to better approximate the local peak. The interpolation algorithm is then executed iteratively to find the signal envelope peak using interpolated local peaks. Type A methods do not use this technique, calling the interpolation algorithm once, while Type B methods use this technique, calling the interpolation algorithm four times.

*Data Region Selection:* Select different local signal peaks relative to the peak of peaks to pass to the interpolation algorithm. We have numbered the relevant data regions containing local peaks accordingly. Each region contains a local peak and its neighbors, with region 3 defined to contain the peak of peaks as in Fig. 2.3.

We have tested four methods using different selection options guided by the two techniques:

- Method 1: Peak Type A using data regions 2,3,4
- Method 2: Peak Type A using data regions 1,3,5
- Method 3: Peak Type B using data regions 2,3,4
- Method 4: Peak Type B using data regions 1,3,5

The peak of the audio pulse is thus approximated by the peak of the interpolant. A linear (i.e. affine) relationship between the fractional sample number of the signal envelope peak $n$ and the estimated inter-robot distance exists $\delta$ with parameters $(M_s, S_0)$ determined empirically from calibration data (least squares regression).

$$\delta(n) = M_s n - S_0 \tag{2.3}$$

$M_s$ is a slope parameter mapping sample numbers to audio pulse distance traveled. The delay shift $S_0$ is a consequence of hardware latency (from both the transmitting and receiving boards) and the constant delay imposed by the FIR filter. For calibration, a set of 20 distance readings (in terms of sample number) per distance were obtained at 10 cm increments from 10 to 70 cm (140 measurements total). The calibrated parameters were $M_s = 0.379 \frac{\text{cm}}{\text{sample}}$ and $S_0 = 19.8$ cm, which corresponds to a measured speed of sound in air $v_s = F_s M_s = 345 \frac{\text{m}}{\text{s}}$.

### 2.4.1 Experimental Results

The distance resolution $\sigma_i$ was taken to be the standard deviation of the measurements $\delta_i$ given true distance $D_i$. Using leave one out cross validation techniques

to calculate $(M_s, S_0)$ for a given data subset, errors on all of the measurements in the dataset were calculated, thus computing $\sigma_i$ for each distance $D_i$. The mean resolution $\bar{\sigma}$ is the mean resolution over all test distances. For each of the methods studied, Table 2.1 shows $\bar{\sigma}$ along with the number of clock cycles required for the FIR filter, Goertzel Algorithm and interpolation to execute. The new FIR-based methods use fixed-point computations, whereas the original algorithm used floating point computation. The new design requires 10% of the execution time with improved accuracy due to interpolation.

| Method | FIR-1 | FIR-2 | **FIR-3** | FIR-4 | old [4] |
|---|---|---|---|---|---|
| $\bar{\sigma}$ [cm] | 0.56 | 0.36 | **0.27** | 0.31 | 1.1 |
| filtering [cycles] | 281k | 281k | **281k** | 281k | 3,170k |
| interpolation [cycles] | 7.65k | 7.65k | **30.2k** | 30.2k | 0 |
| total [cycles] | 289k | 289k | **311k** | 311k | 3,170k |
| time at 8 MHz [ms] | 36 | 36 | **39** | 39 | 396 |

Table 2.1: Interpolation method accuracy vs. computational efficiency in clock cycles.

It may be of interest to know what exactly the computer cycles in Table 2.1 are computing. Below, we outline what is occuring for the specified regions.

Goertzel algorithm: multiplications for (Hamming) windowing 50-sample data subsection (must be recomputed at each iteration), compute DFT for single frequency bin (50 sample window, 204 iterations) which requires a significant number

of multiplication and addition operations. Issues that arose during implementation forced the design to use floating point arithmetic.

FIR: convolution of 32 dimension vector (the coefficients of the 32nd order FIR filter) with the audio signal. This takes a fair number of multiplication and addition operations, but using Horners method, the multiplications can be reduced to a sequence of shift and add operations which is more efficient on microcontroller

Interpolation: calculate closed form solution to peak time (rational function). Some methods use interpolation iteratively, increasing the number of multiplications

## 2.5 Noise Error Propagation Model

In order to further improve distance sensor resolution, it is important to model the sources of noise in the sensor and approximate how their uncertainty propagates through the sensor resulting in measurement uncertainty. This can lead to future design choices such as the inclusion of a crystal oscillator to reduce the effects of timing jitter on sensor resolution. Error analysis was therefore performed to determine the dominant sources of noise in measurement error $\bar{\sigma}$. Error models were developed to map the sensitivity of the interpolated signal envelope peak to variations in timing $\Delta t_i$ or amplitude $\Delta x_i$. One technique to approximate $t_{peak} = f(t, x)$ is by using a linear approximation of the nonlinear function $f$ as in the Extended Kalman filter, where $\Delta t_{peak}^2 \approx \mathbf{J}\Sigma_X\mathbf{J}^{'}$ where $\Sigma_X$ is the covariance matrix. We assume that each of the random variables are independent. $\Delta t_{peak}$ thus linearly maps to a distance measurement error, $\bar{\sigma}_{noise}$. Experimental findings suggest that timing jitter is the

dominant source of error in measurement uncertainty. The nondeterministic timing jitter $\sigma_{sample}$ of the ADC's built-in oscillator was measured and is around 5% to 8% of the clock period. This corresponds to a distance measurement uncertainty contribution between 0.5mm and 0.9mm using the error propagation model, which is a significant component of the total observed measurement uncertainty of 3mm. Chapter 3 will cover the experimental design used to estimate noise covariance for different sources of noise.

Note that fitting errors using interpolation to detect signal envelope peak were not investigated. Alternative and more accurate signal envelope peak detection techniques that are more computationally intensive in digital systems include regression, Hilbert transformations or RF approaches such as frequency mixing (Heterodyning) to extract the signal envelope.

## 2.6 Flicker Noise Model

$$t_{peak} = f(t, X) \tag{2.4}$$

Observation of the FFTs in Fig 2.2 suggests that the dominant feature is flicker noise. The spectral noise density as in Fig 2.2 was fit using linear regression, then integrated over frequencies of interest to determine the RMS contribution due to flicker noise. We further assume that all $x_i$ are independent and normally distributed, i.e. $\perp X_i$, $X_i \sim \mathcal{N}(x_i, (\Delta x_i)^2)$, and propagate $\Delta x_i$ through the interpolation algorithm (2.4) using $\mathbf{J_i} = \frac{\partial f}{\partial X_i}$ to estimate $\Delta t_{peak}$. Chapter 3 will cover the experimental design used to estimate $\Sigma_X$.

### 2.6.1 Timing Jitter Model

$$t_{peak} = g(t_1, T_1, T_2, x) \tag{2.5}$$
$$= f(t_1, t_1 + T_1, t_1 + T_1 + T_2, x)$$

The nondeterministic timing jitter $\sigma_{jitter}$ of the ADC's built-in oscillator was measured and is around 5% to 8% of the clock period. To ensure independence of random variables, one most reformulate the interpolation equations in terms of the time intervals between peaks instead of the peak times themselves. This assumes that $t_1$ and the amplitudes of the three points are held fixed but the time intervals $T_i$ between the measurements are independent and normally distributed, i.e. $T_1 \perp T_2$, $T_i \sim \mathcal{N}(t_{i+1}, (\sigma_{jitter})^2)$. The timing jitter $\sigma_{jitter}$ propagates through (2.5) using $\mathbf{J_i} = \frac{\partial g}{\partial T_i}$ to estimate $\Delta t_{peak}$.

### 2.6.2 Results

120 audio recordings filtered using the FIR filter were analyzed to characterize flicker noise and estimate measurement error. Results are shown in Table 2.2 which suggests that timing jitter is the dominant contribution to variations in $t_{peak}$ and $\bar{\sigma}_{noise}$.

| Timing jitter | Flicker noise (after filtering) |
|---|---|
| 0.5-1.0 mm | 0.11 mm |

Table 2.2: RMS experimental averages for standard deviations.

## 2.7 Practical Design Considerations for the TDOA sensor

The maximum range of the sensor is fixed if the DTC is used. 256 samples at a fixed sample rate will correspond to a maximum theoretical range of 77cm using (2.3). This range can be increased by reducing the sampling rate and reducing the pulse frequency (to maintain a sufficiently high oversampling rate of the audio pulse), at the cost of measurement accuracy. Interpolation can reconstruct a fair amount of this reduction in accuracy, but these performance tradeoffs were not analyzed in detail. However, the buzzer itself must be able to transmit enough power to ensure that the received audio pulse is above the noise floor of the amplifying circuit. Given our current setup, we have experienced a maximum range just short of 70 cm.

## 2.8 Conclusion

The TDOA distance sensor can be used in any system that has wireless communication and signal processing capabilities. Parabolic interpolation of the signal envelope has proven to be an efficient means to reduce distance measurement uncertainty.

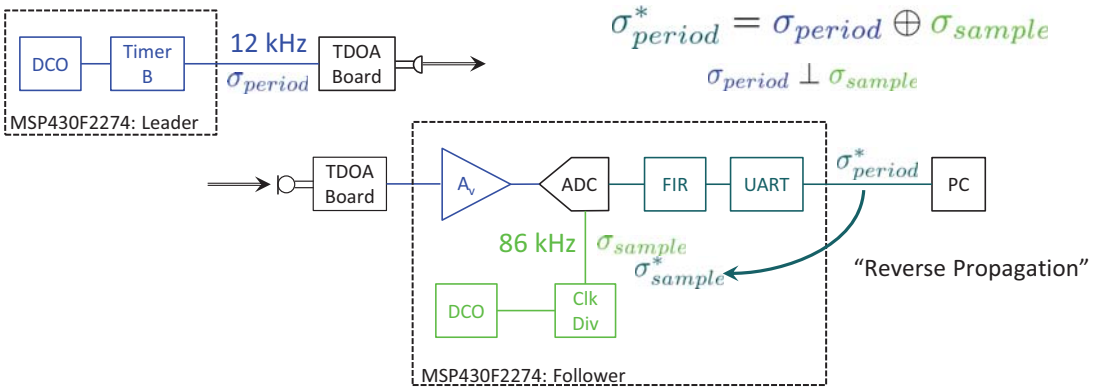


Figure 3.1: System level description of the sources of timing jitter during TDOA sensor operation.

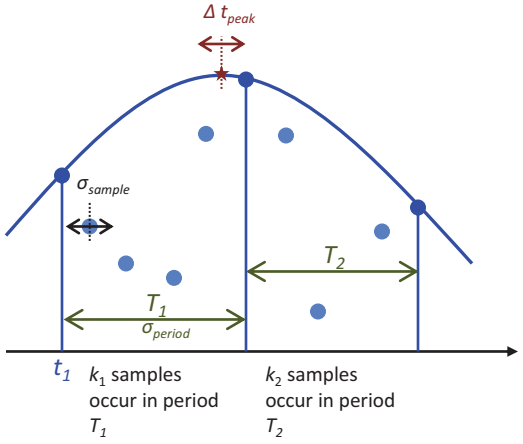


Figure 3.2: A representative digital waveform demonstrating how different forms of timing jitter propagate through the interpolation algorithm.

# Chapter 3

# Noise Characterization for the TDOA Distance Sensor

## 3.1 Effects of Timing Jitter on Distance Measurement Uncertainty

Fig. 3.1 shows how two independent clock sources with different sample rates produce different sources of timing jitter. One important question to ask is how these two sources of noise, $\sigma_{period}$ from the 12kHz audio pulse and $\sigma_{sample}$ from the ADC's sample rate add or mix to produce observable jitter $\sigma^*_{period}$. In other words, let's say that $\sigma^*_{period} = \sigma_{period} \oplus \sigma_{sample}$ by some stochastic process. Alternatively, one could measure $\sigma^*_{period}$ and "push" it back through the system to the sample level, $\sigma^*_{sample}$ that can reconstruct $\sigma^*_{period}$, which is the approach we will be taking.

Fig. 3.2 shows a graphical interpretation of the sources of timing jitter on the estimated signal envelope peak when using the interpolation algorithm. To ensure independence of random variables, one most reformulate the interpolation equations in terms of the time intervals between peaks instead of the peak times themselves. This assumes that $t_1$ and the amplitudes of the three points are held fixed but the time intervals $T_i$ between the measurements are independent and normally distributed, i.e. $T_1 \perp T_2$, $T_i \sim \mathcal{N}(t_{i+1}, k \cdot (\sigma^*_{sample})^2)$. The variance $(\sigma^*_{sample})^2$ is approximately scaled by $k = t_{i+1} - t_i$, the number of samples that occur between $t_i$ and $t_{i+1}$. The timing jitter $\sigma^*_{sample}$ propagates through (2.5) using $\mathbf{J_i} = \frac{\partial g}{\partial T_i}$ to estimate $\Delta t_{peak}$. Other metrics for timing jitter such as cycle to cycle jitter or timing

interval error [22–25] are not well-suited our error analysis model and were therefore not analyzed.

## 3.2 Experimental Setup for Measuring Period and Sample Jitter

The timing jitter measurement experiment to measure noise parameter $\sigma^*_{sample}$ is similar to other period jitter measuring experiments that use a real time digital oscilloscope [22, 23] . Typical jitter experiments using an oscilloscope measure the variances of the time between zero crossings of any periodic signal based on the number of samples that occurred between each 50% crossing. Periodic signals with symmetric rising and falling edges are preferable, leading to three likely candidates: square waves, sinusoids, and triangle waves given their ease of synthesis using function generators. Alternatively, the DC component of the digital signal is removed, and zero crossings are detected. Note that the signal must be significantly oversampled to ensure an accurate depiction of zero crossings. Typically, one would wish to have around 3-10 points around the signal transition region [22]. This leads to significant oversampling (by a few orders of magnitude) which is impractical for measurements involving microcontrollers with small storage capacity, given the constraints of the Data Transfer Controller (DTC) [16].

Given the design of the microcontroller, it is impossible to detect the timing jitter $\sigma_{sample}$ of a single sample (from now on referred to as the sample jitter) because we cannot directly observe the ADC clock. Instead, a test signal with low jitter is fed into the on board MSP430, which samples and stores this waveform. In this case, the

MSP430's ADC sample rate clock is also injecting the timing jitter into the signal. Despite using the DTC to reduce nondeterministic sample jitter, sample jitter still exists due to hardware limitations [16]. Note that there are sample rate limitations of this technique which fall below typical experimental guidelines for having more than 3-10 samples per rising or falling edge (We generally have 1-2 samples per rising edge at the higher test frequency). We conducted the experiments around the sensor's typical operating point to reduce the consequences from higher order effects, where $F_{sample} \approx 86\text{kHz}$ and $F_{input} = 12\text{kHz}$. This leads to an oversampling rate of about 7. To accommodate for this low oversampling rate, linear interpolation was used to improve the accuracy of the zero crossing measurement [25]. Period jitter $\sigma^*_{period}$ of the measured signal is calculated, and a statistical model estimates how this measurement uncertainty maps to the timing jitter of the ADC clock, $\sigma^*_{sample}$.

## 3.3 Timing Jitter Reverse Propagation Model

There are two different models (with differing limitations) to describe the relationship between observable period jitter $\sigma^*_{period}$ and the sample jitter $\sigma^*_{sample}$ of the ADC. Essentially, this model must account for the fact that the timing jitter for each ADC sample accumulates when measuring the timing jitter of the high fidelity test waveform.

The reverse propagation model is not a means to detect the sample jitter (also known as time base jitter) of the ADC or discriminate sample jitter from period jitter. Instead, it maps observed period jitter to uncorrelated, time independent

sample jitter that can reconstruct the observed period jitter. It may not reflect the actual distribution or correlations of sample intervals. For example, if a perfect sample clock was used measuring a test signal with significant period jitter, back propagated "sample jitter" would be time dependent, because the most jitter would occur around the rise/fall sections. Determining the source of the jitter would require observing something similar to a time interval error plot [22] to see the time dependence of the jitter. However, constructing a time interval error plot requires a reference clock, which is internal to (or inaccessible outside) the microcontroller, so this is not worth investigating for our application.

It will be of most interest to analyze signals including all sources of timing jitter noise that the TDOA sensor will experience during normal operation without concern to their origin. Reverse propagation is necessary because we are concerned with sub-carrier signal period measurement accuracy to understand the effects of timing jitter at the sample level instead of the carrier signal period.

One model is based on the the equation for calculating the variance of a random sum of random variables (3.1). For (3.1) to hold, we must assume that all random variables are i.i.d.

$$\mathrm{Var}(\Sigma_{i=1}^{N} X_i) = \mathrm{E}[N]\,\mathrm{Var}(X_i) + \mathrm{E}[T_i]^2\,\mathrm{Var}(N) \tag{3.1}$$

Normalizing for the clock period (which is the case when looking at discrete time sequences), $\mathrm{E}[T_i] = 1$ and assuming $\mathrm{Var}(N)$ is insignificant, (3.2) directly follows from (3.1).

$$\sigma^*_{sample} \approx \frac{\sigma^*_{period}}{\sqrt{\mu_{period}}} \tag{3.2}$$

However, given the low sampling rate and use of interpolation, the period time is not limited to discrete sample numbers. Treating this phenomenon as a random sum of random variables will neglect the fractional component of the time interval. This could bias the findings incorrectly and lead to erroneous results since the measurement no longer ends at a valid stopping time!

Another means of viewing this phenomena is to assume that the uncertainty (i.e. variance) of the measurement of a time interval given timing jitter grows linearly over time. This is analogous to assuming that the measurement of any fractional period time interval $\hat{T}$ using a clock with timing jitter is characterized by the Weiner Process (or Brownian Motion), that is $\hat{T} \sim \mathcal{N}(T, T \cdot (\sigma^*_{sample})^2)$. Assume that a time interval of known length $T$ is repeatedly measured, $\hat{T}$. The statistics of $\hat{T}$ can directly infer the model parameter $(\sigma^*_{sample})^2$. This view also suggests (3.2) without having to worry about stopping times, but no physical justification for using Brownian Motion models for subclock timing jitter measurements are provided. Experimental verification will suggest that (3.2) will yield accurate measurements.

*Timing Jitter Measurement Experiment*

The following Experiment is used to measure $\sigma^*_{sample}$ from experimental trials.

1. Take a sufficient number of 255 sample runs observing a clock on the microcontroller's ADC so that the number of (full) periods exceeds $N = 1000$.

2. Calculate the period of measured clock cycles using linear interpolation and

zero crossing detection. Use Spectral Analysis to measure the sampling rate of the ADC.

3. Verify that the distribution of clock periods is (approximately) Gaussian. Observing histograms is sufficient.

4. Calculate the mean $\mu_{period}$ and population standard deviation $\sigma^*_{period}$ of the clock periods.

5. Use (3.2) to back propagate period jitter to sample jitter $\sigma^*_{sample}$.

6. Estimate experimental/model error by forward propagating the sample jitter using the generative model.

### 3.3.1 Spectral Analysis

If the frequency of the input clock signal is accurately measured, the sampling rate of the ADC can be measured by taking a FFT of the test data. Essentially, the frequency bin with the highest energy content corresponds to the input signal (of known frequency), which can then be scaled accordingly to recover the sampling rate. To increase the number of frequency bins, the signal portion used for spectral analysis must have a sufficiently high number of samples (1024 samples results in a measurement resolution of 84 Hz).

### 3.3.2 Generative Statistical Model for Forward Propagation

The term "generative model" is an adaptation of the term's use in machine learning classification problems, where the model used can generate synthetic data points to confirm the model's accuracy [26]. This is done by generating synthetic waveforms using the system model, forward propagating $\sigma^*_{sample}$. Conducting an experiment on this synthetic dataset will generate $\hat{\sigma}^*_{period}$ which can be compared to $\sigma^*_{period}$ to approximate the model error. However in this case, the form of the function is known or hypothesized (3.2), and the problem is a form of regression (i.e. parameter estimation) instead of classification. To confirm the experimental method, a MATLAB simulation generated synthetic data by sampling a periodic waveform at normally distributed random intervals with $\mu_{sample} = 1$ and $\sigma^*_{sample}$ as measured from the experiment. The known sample intervals are not passed to the jitter estimation routine. Comparing the experimental data observations to observations generated by the generative statistical model suggests the reconstruction accuracy of the timing jitter reverse propagation model.

Viewing histograms of period lengths (in fractional sample numbers) provides insight into the fidelity of the algorithm. Because timing jitter is Gaussian, both noise propagation models suggest that the cycle periods are also Gaussian in distribution. Nongaussian distributions of cycle periods suggest biasing in the algorithm that will affect results. It is apparent that the use of 12 kHz square waves for timing jitter analysis result in biased distributions that appear more uniform, where simulation results have the highest error. This error drops when the the frequency of

the square wave drops to 5 kHz (there exist more points in the transition region). Sinusoids exhibit cycle periods with a Gaussian distribution both in experiment and simulation and are therefore the preferred test signal. Amplitude noise such as quantization error, flicker and thermal noise etc. were not modeled in the generative statistical model.

## 3.4 Experimental Setup and Results

To ensure the accuracy and repeatability of experimental results, experiments and simulations included $N \geq 1000$ cycles. $1000 < N < 10000$ is typical for timing jitter experiments.

| | Signal Type | freq. [kHz] | $V_{min}$ [mV] | $V_{max}$ [mV] | Op amp gain $A_V$ |
|---|---|---|---|---|---|
| Exp. 1 | Square | 12 | 1 | 13 | 130 |
| Exp. 2 | Square | 5 | 1 | 13 | 130 |
| Exp. 3 | Sine | 12 | 1 | 13 | 130 |
| Exp. 4 | Sine | 5 | 1 | 13 | 130 |
| Exp. 5 | Square | 11.85 | 0 | 3000 | 1 |
| Exp. 6 | Sine | 11.85 | *drifts* | *drifts* | 100 |

Table 3.1: Summary of experimental setups

The hardware setups for Experiments 1–6 are summarized in Fig. 3.3.

| | $N_{samples}$ (exp.) | $\sigma_{sample}$ (exp.) | model error |
|---|---|---|---|
| Experiment 1 | 1803 | 0.028 | 19 % |
| Experiment 2 | 1519 | 0.034 | 2 % |
| Experiment 3 | 1140 | 0.0061 | -1% |
| Experiment 4 | 1617 | 0.0054 | -1% |
| Experiment 5 | 1970 | 0.082 | 24 % |
| Experiment 6 | 1530 | 0.052 | -3% |

Table 3.2: Summary of Experimental results for estimation of sample jitter and simulation accuracy.

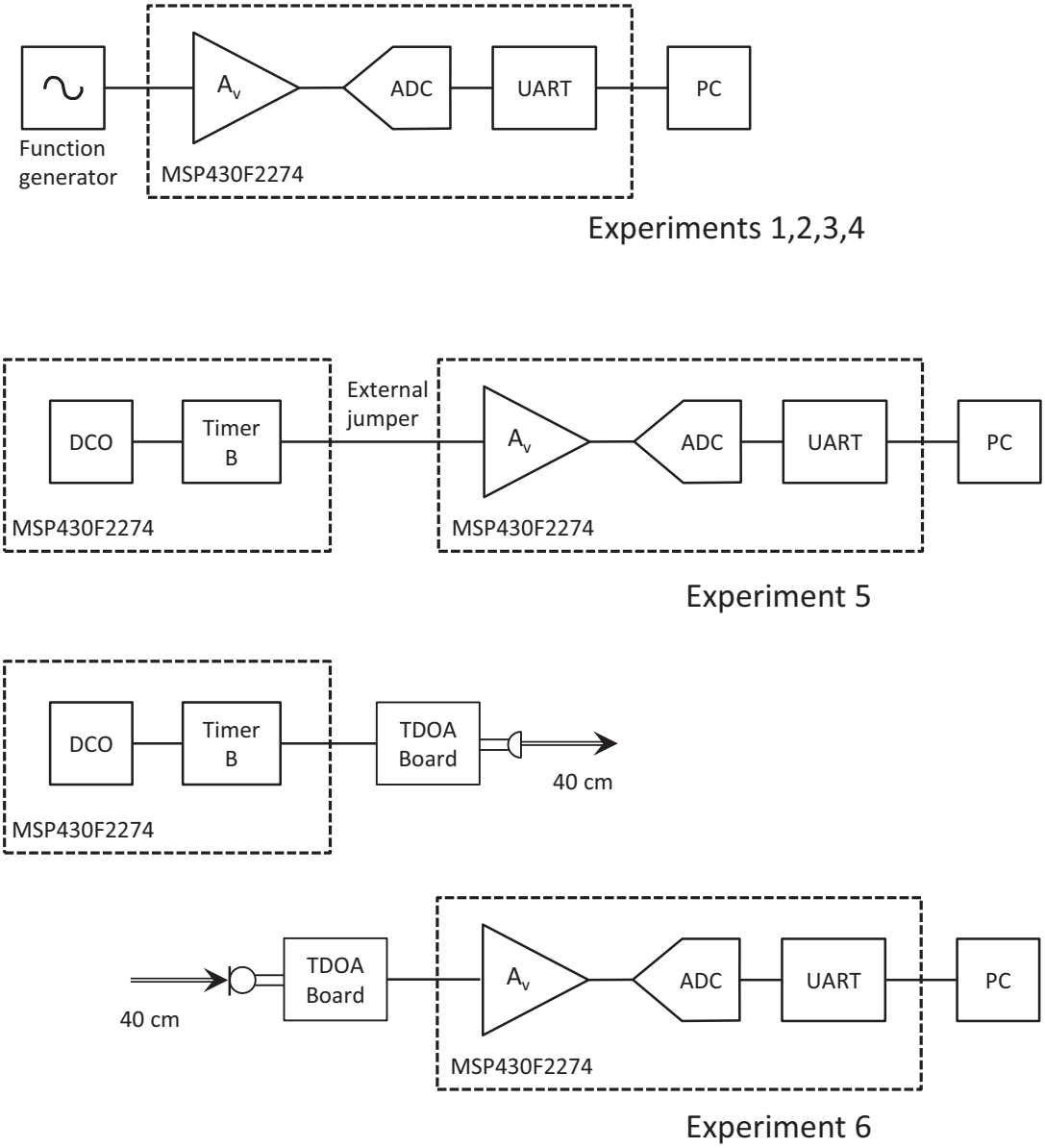


Figure 3.3: Simplified hardware diagrams of the experiments. For additional configuration parameters, refer to Table 3.1

The source of the test signal for Experiments One through Four was a function generator. Refer to Fig. 3.3 to see the experimental hardware setup. To avoid the $V_{min}$ and $V_{max}$ problem, signals were configured to a minimum voltage of 1mV and maximum voltage of 13mV. This signal was sent into the slew-rate limited

operational amplifiers configured to increase the amplitude of the signal with a gain of 100 V/V.

### 3.4.1 Experiment One: Square wave at 12 kHz

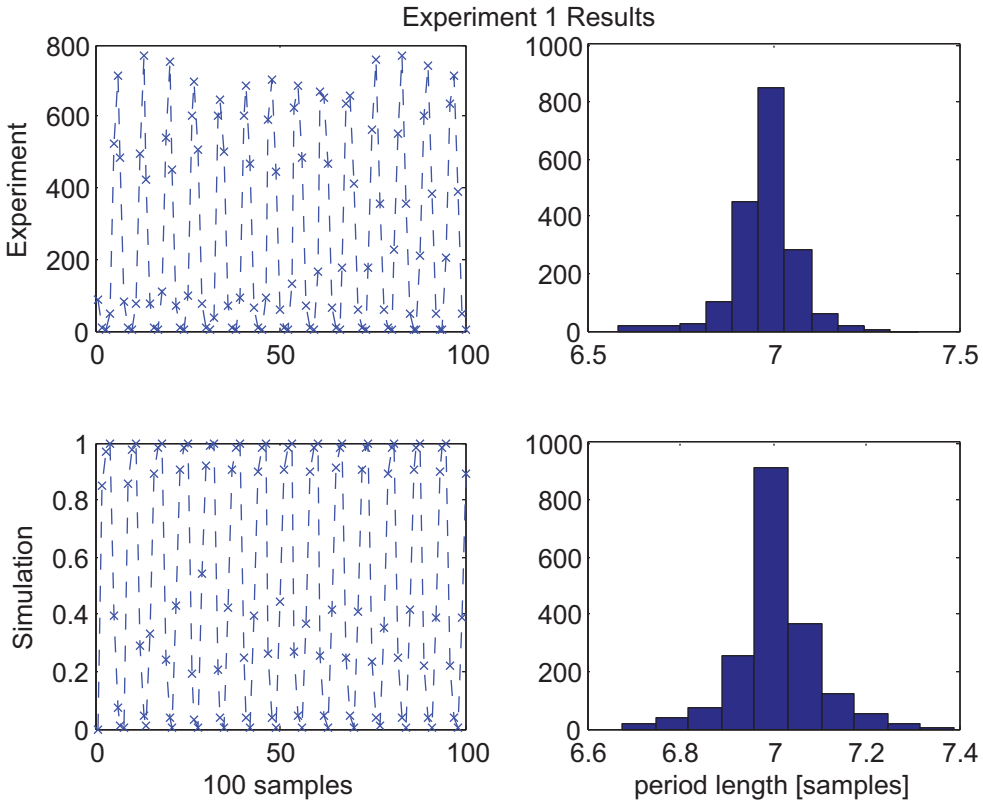


Figure 3.4: Experiment One: 12 kHz square wave. Sample signals and population statistics are shown for both experiment(top) and simulation (bottom).

By manually observing square wave signals sample by the microcontroller that the circuit's characteristics profoundly affect the rising and falling edges of the signal. A first order low pass filter with $\tau = 0.5$ samples was added to the generative model to improve the realism of the simulation. Ideal square waves result in a strictly

Bernoulli distribution for the small period jitter observed in experiment.

Figure 3.4 shows the histograms for both the experiment and simulation, with 100 example points from each for qualitative agreement of the experiment with the generative statistical model/simulation. While the qualitative shapes of the distributions agree, the resulting estimated sample jitter of the experiment and the modeled sample jitter of the simulation do not agree in Experiment One. The model has an error of about 21%. It is believed that there are insufficient points in the transition region of the signal.

### 3.4.2 Experiment Two: Square wave at 5 kHz

Experiment Two was identical to Experiment One, except the frequency of the test signal. Experimental results better agree with simulation results with a 2 % error most likely due to the fact that roughly twice as many points exist in the transition region.

Figure 3.5 shows the histograms for both the experiment and simulation, with 100 example points from each for qualitative agreement of the experiment with the generative statistical model/simulation.

### 3.4.3 Experiment Three: Sine wave at 12 kHz

Experiment three is similar to experiments one and two, differing in using a 12 kHz sinusoid signal as the test signal.

Figure 3.6 shows the histograms for both the experiment and simulation, with

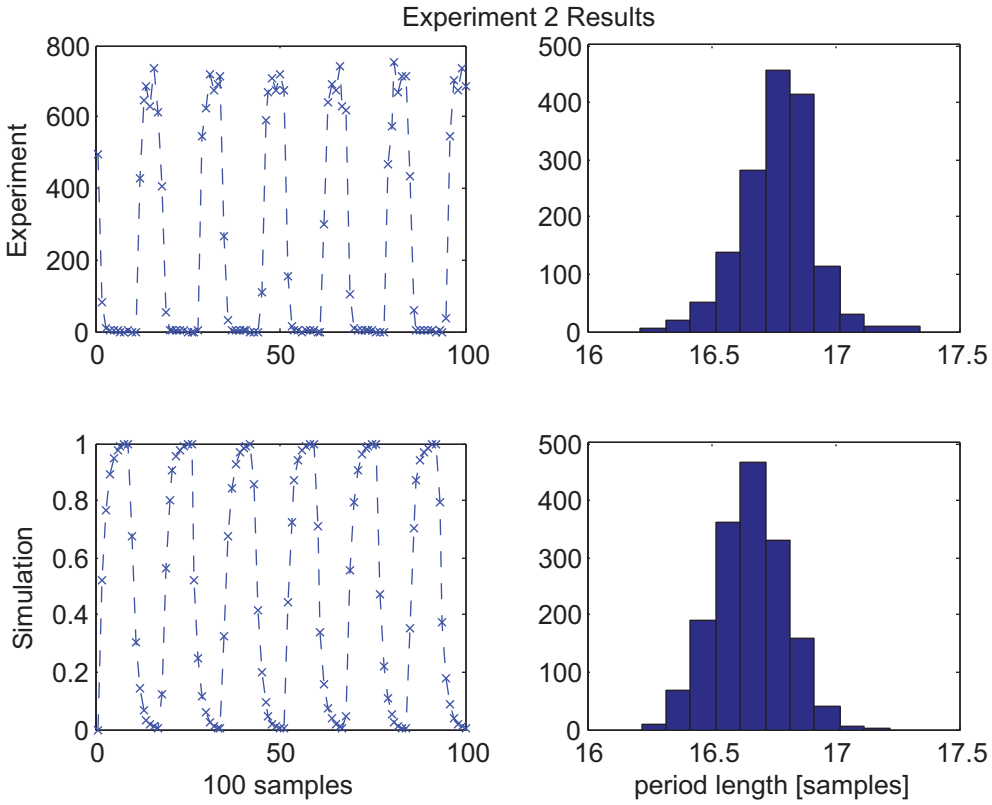


Figure 3.5: Experiment Two: 5 kHz square wave. Sample signals and population statistics are shown for both experiment(top) and simulation (bottom).

100 example points from each for qualitative agreement of the experiment with the generative statistical model/simulation.

#### 3.4.3.1 Experiment Four: Sine wave at 5 kHz

Figure 3.7 shows the histograms for both the experiment and simulation, with 100 example points from each for qualitative agreement of the experiment with the generative statistical model/simulation.

Experiments three and four exhibited better insensitivity to frequency, where

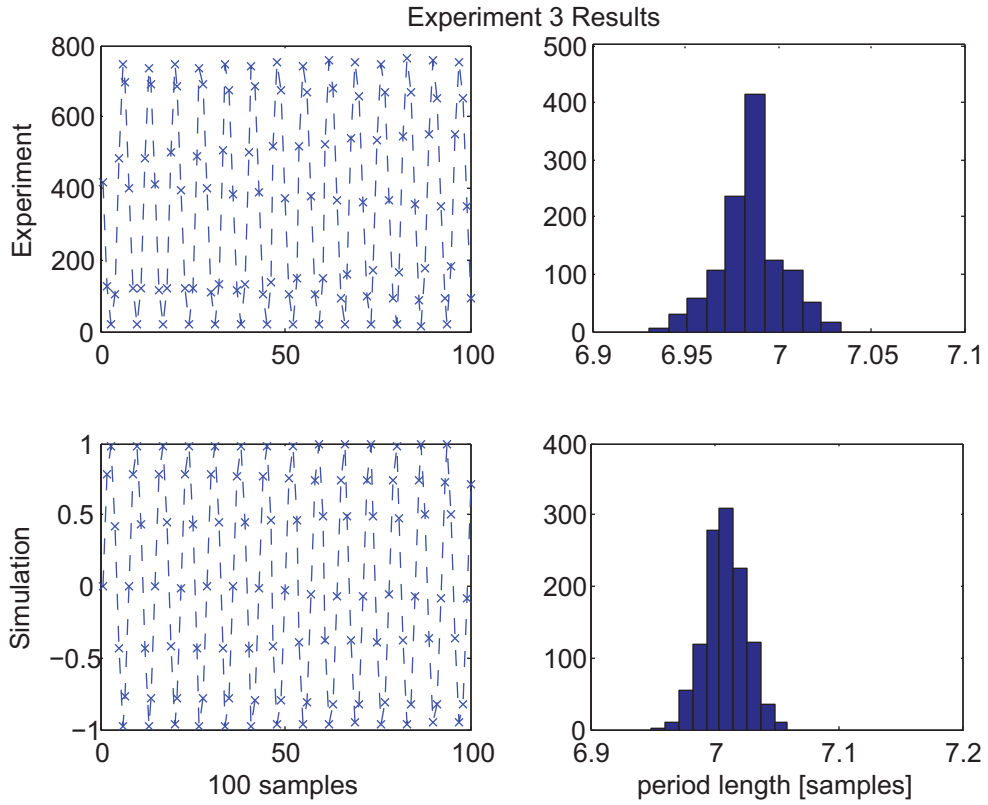


Figure 3.6: Experiment Three: 12 kHz sine wave. Sample signals and population statistics are shown for both experiment(top) and simulation (bottom).

$S_f^\sigma \approx 15\%$

### 3.4.4 Experiment Five: On Board Timer Module Jitter

The timer module used for the TDOA sensor buzzer fed a pulse train directly to the input of a slew-rate limited operational amplifier on board the MSP430F2274 microcontroller. The op amp was configured as a unity gain buffer amplifier and its output was fed into the on board ADC. Refer to Fig. 3.3 for the differing hardware configuration in this experiment. This set up captured both summation of the

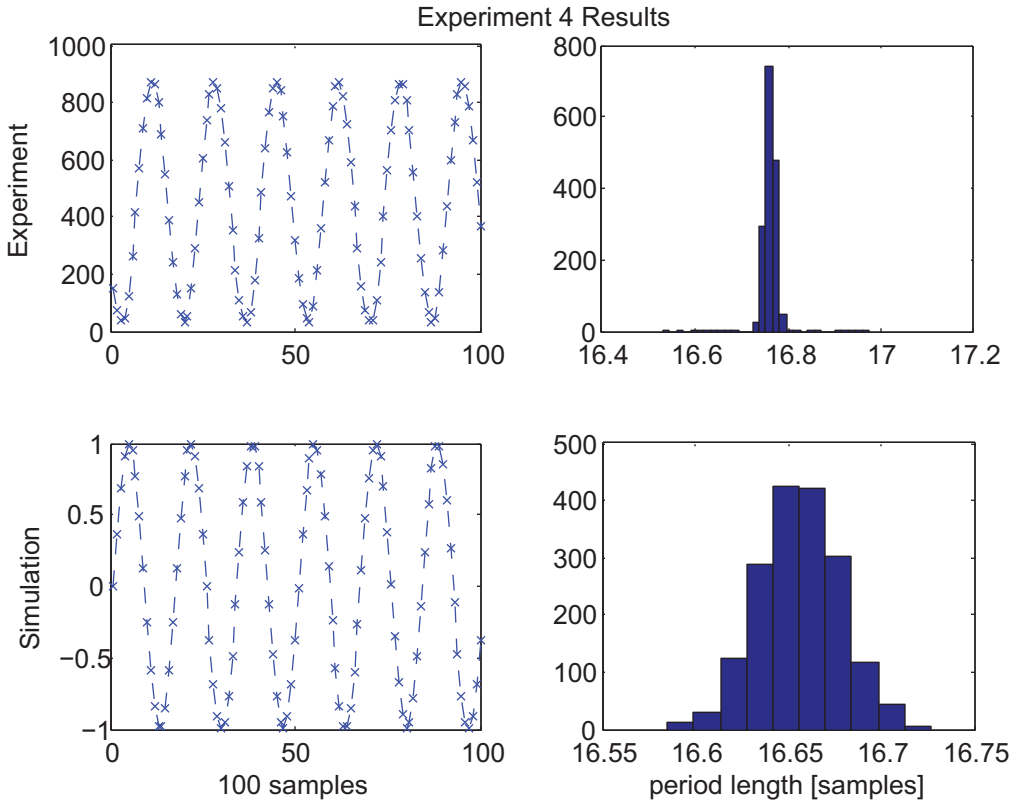


Figure 3.7: Experiment Four: 5 kHz sine wave. Sample signals and population statistics are shown for both experiment(top) and simulation (bottom). Notice that the existence of outliers in the experiment affect the range so additional bins have been added to the histogram. Neglecting outliers the two histograms are similar.

sample jitter from the ADC and the timing jitter from the timer module (which propagate through the buzzer, microphone and op amps).

Figure 3.8 shows the histograms for both the experiment and simulation, with 100 example points from each for qualitative agreement of the experiment with the generative statistical model/simulation. The simulation accuracy for Experiment Five (24% error) is comparable to the accuracy seen in Experiment One (19% error). Note that both experiments utilized a 12 kHz square wave test signal, though had

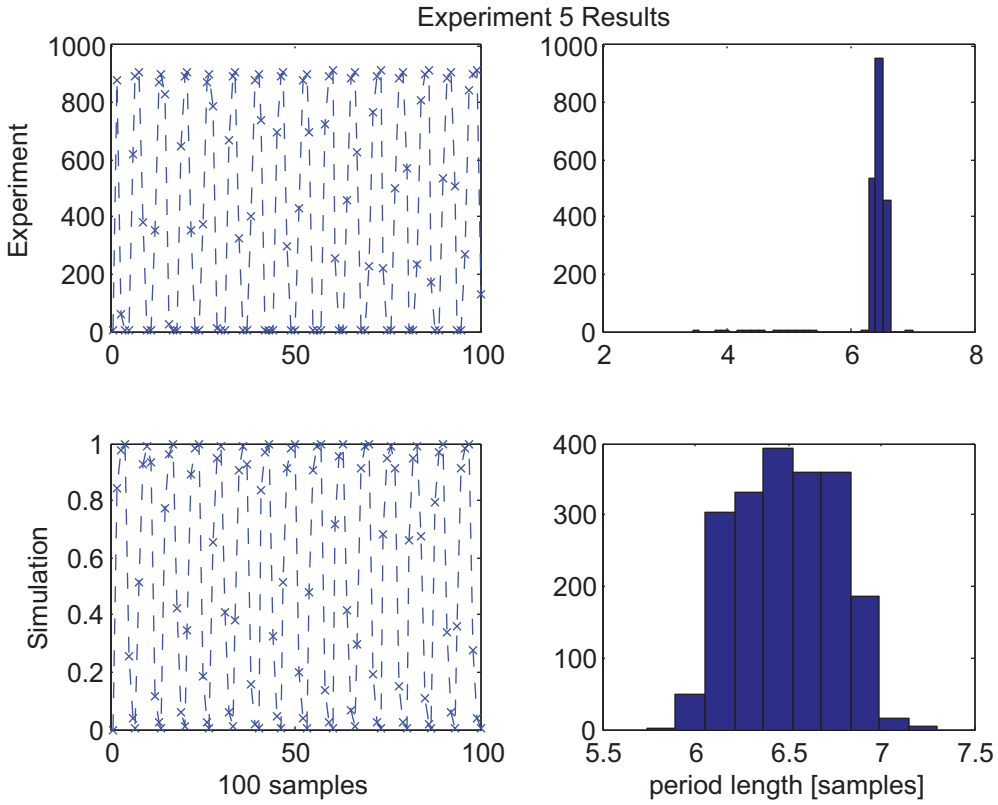


Figure 3.8: Experiment Five: 12 kHz square wave generated by the on board timer. Sample signals and population statistics are shown for both experiment(top) and simulation (bottom). Notice that the existence of outliers in the experiment affect the range so additional bins have been added to the histogram. Neglecting outliers the two histograms are similar.

different sources for the test signal.

### 3.4.5 Experiment Six: Filtered Audio Signal at 12 kHz

This experiment most closely models signals observed during the sensor's operation in the field. The TDOA buzzer driven by a 12 kHz square wave from the on board timer module buzzed continuously. The microphone was spaced 40 cm

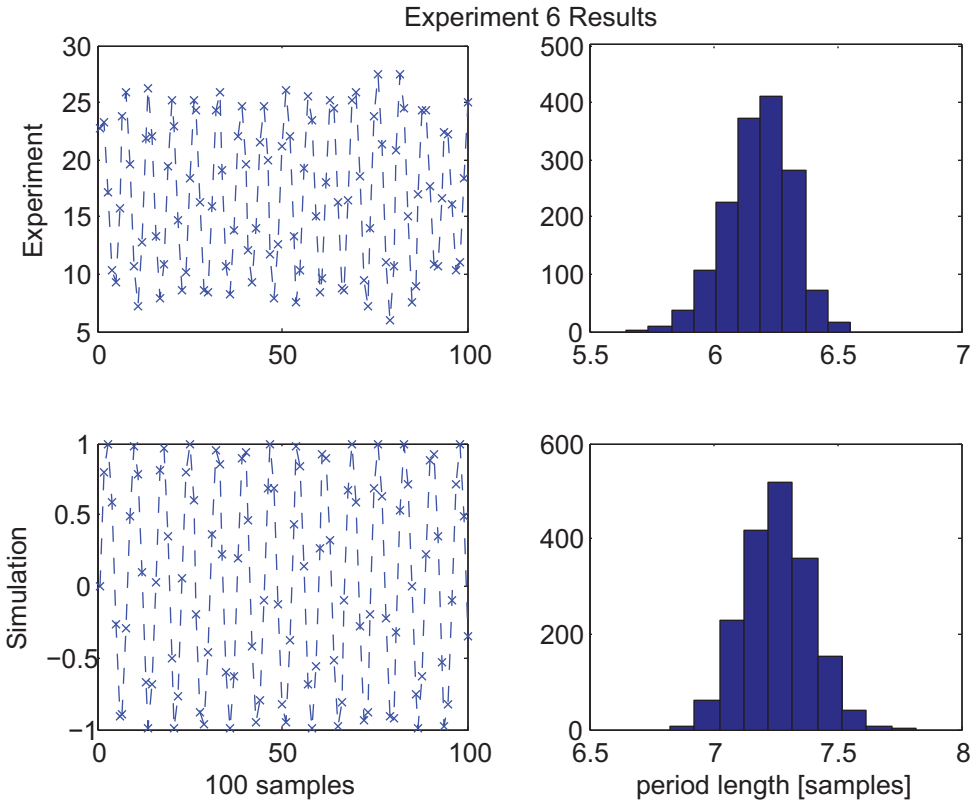


Figure 3.9: Experiment Six: 12 kHz square wave generated by the on board timer and transferred to another board 40 cm away as an audio pulse. Sample signals and population statistics are shown for both experiment(top) and simulation (bottom).

from the buzzer and the microphone signal amplified by the internal op amps was recorded. Refer to Fig. 3.3 for the differing hardware configuration in this experiment. Signals were filtered in MATLAB using the same FIR filter used on board the TDOA sensor. This was necessary because the DC level of the signal varied which would have hindered the zero crossing detections. The use of the filter also closer modeled the real world operating conditions of the TDOA sensor.

Figure 3.9 shows the histograms for both the experiment and simulation, with

100 example points from each for qualitative agreement of the experiment with the generative statistical model/simulation.

## 3.5 Summary of Experimental Findings

The estimates of $\sigma^*_{sample}$ vary dramatically based on the type of input signals, while showing reasonable insensitivity to change in frequency of the test signal (the model suggests that $\sigma^*_{sample}$ is insensitive to changes in frequency). These discrepancies are due to amount of period jitter present in the original test signal. A common generation technique for square waves is running a sinusoid through a comparator, which contributes other sources of noise such as thermal noise. It is likely that the true sample jitter $\sigma_{sample}$ is bounded by the sample jitter $\sigma^*_{sample}$ found in the sinusoids, while experiments 1,2,5 and 6 using square wave test signals inject their own sources of period jitter into the experiment. However, due to the lack of access to the sample clock, no means of accurately discriminating the source of jitter is offered. For measurement uncertainty analysis, we will consider the results from Experiments Five and Six, where $\sigma_{sample}$ is approximately 5% - 8%.

## 3.6 Measurement of Flicker Noise

Observation of the FFTs in Fig 2.2 suggests that the dominant source of amplitude noise is flicker noise. The spectral noise density as in Fig 2.2 was fit using linear regression, then integrated over frequencies of interest to determine the RMS contribution due to flicker noise. We further assume that all $X_i$ are independent

and normally distributed, i.e. $\perp X_i$, $X_i \sim \mathcal{N}(x_i, (\Delta X_i)^2)$, and ${\Delta X_i}^2 = \sigma_{flicker}$ propagates through the interpolation algorithm (3.3) using $\mathbf{J_i} = \frac{\partial f}{\partial X_i}$ to estimate $\Delta t_{peak}$. The standard noise model for flicker noise is in (3.3) [27].

$$ {\Delta X_i}^2 = K_1 \frac{I^a}{f^b} \Delta f \tag{3.3} $$

Where $K_1, a, b$ are process/circuit dependent parameters that are generally unknown or are not modeled. We will treat our digital signal as if it were a current signal, $I$.

Given and FFT of an observed signal, it is possible to fit (3.3) to the FFT using linear regression to estimate unknown model parameters $b, c$ where $c = ln(K_1 I^a \Delta f)$. Reformulating (3.3) in matrix form with a basis of the unknown parameters $[b\ c]^{'}$ yields (3.4), a precursor to formulating the regressor matrix of multiple observations $\mathbf{\Phi_{flicker}}$.

$$ 2ln(\Delta X_i) = \begin{bmatrix} 1 & -\log f_j \end{bmatrix} \begin{bmatrix} c \\ b \end{bmatrix} \tag{3.4} $$

The linear regression problem (i.e. solve the overconstrained system for x given $y = Ax + \rho$ minimizing unknown $\rho$ )can thus be formulated as (3.5).

$$ I = \mathbf{\Phi_{flicker}} \begin{bmatrix} c \\ b \end{bmatrix} + \rho \tag{3.5} $$

$$ \mathbf{\Phi_{flicker}} = \begin{bmatrix} \mathbf{1}_{N \times 1} & -\log f \end{bmatrix} \tag{3.6} $$

Where $f$ is a column vector of the frequency bins of interest (i.e. the freq band in which flicker noise is dominant) in the Fast Fourier Transform (FFT) of the digital signal, and the natural logarithm is computed element wise. $N$ is the number of samples in the relevant frequency band when computing the FFT. It is

also important to ignore the DC component (first frequency bin) of the FFT, as this will not fit to the flicker noise model.

The collection of observations has now been rewritten as an affine function (3.5) of the unknown parameters, $b$ and $c$, and can be estimated while minimizing slack variable $\rho$ using the pseudoinverse operation [28, 29] Refer to Chapter 4 for a more comprehensive treatment on the pseudoinverse operation and regression.

Once $b$ and $c$ are estimated, one can put these back into the original flicker noise model, and can integrate analytically over the frequency band of interest, yielding $\sigma_{flicker}$. This can be done over many experimental trials, and the RMS of all estimated $\sigma_{flicker}$ can be taken to be the noise parameters for error propagation analysis.

## 3.7 Conclusion

The various experiments suggest that the dominant source of timing jitter stems from the generation of the 12 kHz audio pulse by the TDOA sensor. Three experimental findings lead to this conclusion. First, Experiments 3 and 4 suggest negligible sample clock jitter. Second, Experiment 5, despite having questionable accuracy given the low sample rate and comparatively high model error, suggests that the timer module generates a clock for the buzzer with significant timing jitter. This intuitively makes sense because the timer module counts a significant number of clock pulses (i.e. adds clock periods), which increase uncertainty given (3.1). Experiment 6 demonstrates that this error propagates through the TDOA

hardware and FIR filter. Using the error propagation model developed in Chapter 2, this suggests that providing a more accurate system clock for the TDOA sensor will significantly reduce TDOA sensor measurement uncertainty. Including a high frequency crystal oscillator [30] on the new TDOA board will significantly reduce measurement uncertainty.

# Chapter 4

# Heading Estimation using Odometry and Distance Only Sensing for Miniature Mobile Robots

## 4.1 Summary

A single distance measurement does not give sufficient information about the robot's relative location to the leader when attempting to determine the leader's position. This is independent of sample rate. This requires the inclusion of a heading estimator to complement the distance-only measurements when tracking a leader's position. Essentially, the heading estimator uses trigonometry (law of cosines) to reconstruct or approximate the unknown heading angle to the leader or beacon using multiple distance measurements taken over time. Measurement uncertainty characteristics are not factored into this state estimate, unlike optimal state estimation techniques such as Kalman Filtering.

Another necessary piece of information required for heading estimation is an approximation of how far the robot has traveled between distance measurements. Miniature differential drive robots using pager motors and gearboxes do not often have wheel encoders to measure wheel velocities, making tasks such as odometry less accurate. We have developed a position estimator that is better suited for small platforms with limited computation capabilities.

## 4.2 Introduction

Previous work has used radio signal strength measurements and an Extended Kalman Filter as a means of robot localization [31]. However, such techniques cannot be implemented on smaller platforms with limited computation capabilities. The objective of the heading estimator is to determine the heading of the leader robot given multiple distance-only measurements between a stationary leader and a follower robot attempting to rendezvous with the leader. The measured angle heading along with the distance information is then fed to the higher level motor controller to make motion planning decisions.

This technique requires accurate, directionally insensitive distance sensors and odometry information that is reliable over short distances (i.e. the distance traveled between distance measurements). We will discuss the distance-only sensor characteristic and odometry with necessary motor calibration.

## 4.3 Odometry Without Wheel Encoders

Odometry is a challenging task on miniature robot platforms because the motors have no wheel encoders forcing the robot to rely upon the system model given the known control signal alone for position estimation. This greatly increases uncertainty in odometry, but it is only needed for small distances in heading estimation. Exact solutions to the kinematic equations of motion that do not require small step sizes are advantageous when relying upon motor commands alone for odometry. Large time intervals with constant robot velocities require fewer computations.

Odometry models must either solve the differential equation system directly or have correction factors to yield exact solutions. Assuming piecewise constant motor commands, one can solve for the trajectory of the robot through Euclidean space (x,y,$\theta$) given calibration data. *For the remainder of this thesis, we will define odometry to be shorthand for position estimation using modeled system response for a given motor control, i.e. without the use of wheel encoders.*

Several problem formulations using different control variables and solutions for the equations of motion exist for mobile robotics platforms [32–34]. We have opted to use the tangential velocity, $\upsilon$, and the rotational velocity, $\omega$ of the robot as our primary representation of the control variables, $\mathrm{v} = [\upsilon\ \omega]'$. *This choice of control variables is important.* It enables one to split up motor calibration from the equations of motion. This allows for a simpler representation of the system dynamics and decouples platform characteristics from odometry, so long as the platform maintains linearity conditions between motor inputs and control variables. The KEPLR drivetrain was designed to best preserve this linear mapping. In other words, this decouples the calibration data (specific to an individual robot) from the odometer, making odometry platform-independent.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \upsilon \cos\theta \\ \upsilon \sin\theta \\ \omega \end{bmatrix} \tag{4.1}$$

Assuming that $\upsilon$ and $\omega$ are piecewise constant, the solutions to the equations of motion in (4.1) are in (4.2).

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \theta_{k+1} \end{bmatrix} = \begin{bmatrix} \frac{\upsilon_k}{\omega_k}(\sin\theta_{k+1} - \sin\theta_k) + x_k \\ \frac{\upsilon_k}{\omega_k}(\cos\theta_k - \cos\theta_{k+1}) + y_k \\ \omega_k(t_{k+1} - t_k) + \theta_k \end{bmatrix} \tag{4.2}$$

$s_k$ from Fig. 1.2 (how far the robot has traveled between distance measurements), is the normed change in displacement in the $(x, y)$ plane as in (4.3).

$$s_k = \sqrt{(x_k - x_{k-1})^2 + (y_k - y_{k-1})^2} \tag{4.3}$$

## 4.4 Robot Kinematic Model

Since the robots have a differential drive, one must quantify the relationship between wheel velocities and the robot velocities to execute the desired robot velocities. The motors are in turn driven by Pulse Width Modulation (PWM) using one of the microcontroller's on board timer modules [35] and an H bridge driver (implemented virtually by the microcontroller). Barring random effects, we argue that the relationship between motor controls (average applied voltage) and the resulting robot velocities is linear for the following reasons:

- The kinematic model of a differential drive robot mapping wheel velocities to "body velocities" is linear
- The (linear) differential equations governing the motor yield steady state velocities that are directly proportional to the average input voltage and thus the duty cycle
- The transients of the system are negligible because the walle bot drive train has a high built-in gear ratio, reducing the apparent load.

However, it is important to note that slip conditions due to friction and wheel base geometry will limit the linear region.

$$\begin{bmatrix} \upsilon \\ \omega \end{bmatrix} = \mathbf{C} \begin{bmatrix} u_l \\ u_r \end{bmatrix} \tag{4.4}$$

$$\mathbf{C} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} \tag{4.5}$$

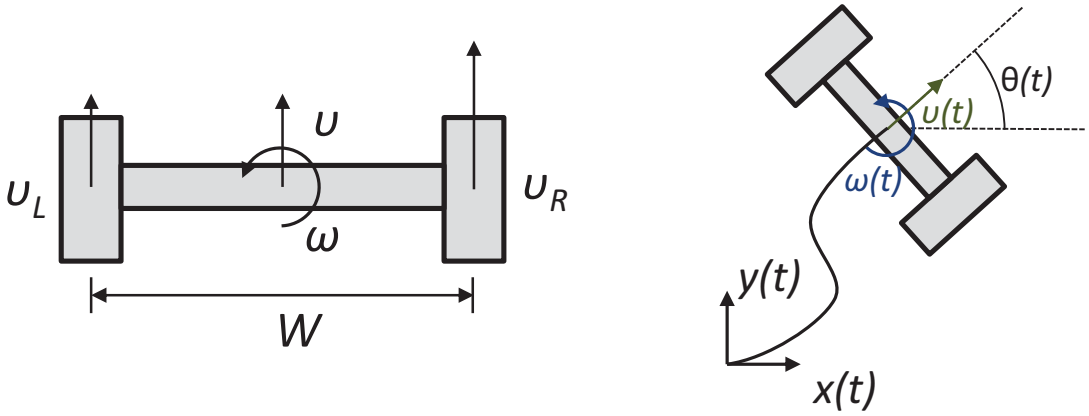


Figure 4.1: Graphical representation of the robot kinematic and odometry model.

Fig. 4.1 graphically represents the relationship between motor velocities and "body velocities"/control variables, and the odometry model.

### 4.4.1 Random effects in the kinematic model

The error in our kinematic model comes from the uncertainty in the left and right wheel velocities, and arises from wheel slippage, motor transients, nonlinear effects from the motor, and other random effects. We assume that these errors $e_L$ and $e_R$ are independent and are normally distributed though this condition does not have to hold for least squares fitting.

$$v_L = k_L \bar{V}_L + e_L \qquad (4.6)$$
$$v_R = k_R \bar{V}_R + e_R$$

$v_L$ and $v_R$ are the left and right wheel velocities. $k_L$ and $k_R$ are unknown constants that account for the wheel diameter, steady state motor characteristics and the gearbox ratio (all of which are assumed linear). $\bar{V}_L$ and $\bar{V}_R$ are the average voltages applied to the left and right motors respectively. These voltages are generated on the microcontroller by using pulse width modulation (PWM). The duty cycle of the corresponding motors will be referred to as $u = [u_L \ u_R]'$, where $u_i \in [-1, 1]$. Also, $\bar{V}_L = V_s u_l$ etc. where $V_s$ is the voltage of the onboard power supply. Resulting errors propagate through the linear system and are observed when the robot velocities are measured, compounding over time.

### 4.4.2 Motor Calibration using the Least-Squares method

Parameters of the linear map between PWM duty cycles $u$ and resulting robot velocities v are unknown. Deterministic offsets from the ideal linear map can be measured using calibration techniques, while random errors can be characterized while solving the least squares optimization problem. This technique is related to previous work [36]. Using data from multiple experiments/observations that measure observed robot motion given a known motor control input, one can generate a regressor matrix and solve a least-squares problem to develop the kinematic model of the robot.

A vision system with a flat surface and a camera with a telecentric lens uses

software libraries from OpenCV and Robot Operating System (ROS) to collect data about the robots trajectory in Euclidean space $(x, y, \theta)$ along with timestamps [37]. For each time step in the data, the calibration method computes the corresponding v using (4.7), derived from an alternative odometry model [32]. After the instantaneous robot velocities v are calculated for each time step, the results are averaged to create a $(u, \bar{\mathrm{v}})$ pair for each trial.

$$\begin{aligned} \omega &= \frac{\Delta\theta}{\Delta t} \\ \upsilon &= \frac{d}{\Delta t} \\ d &= \frac{\sqrt{(\Delta x)^2 + (\Delta y)^2}}{\operatorname{sinc}\frac{\Delta\theta}{2}} \end{aligned} \tag{4.7}$$

where the $\Delta$ operator specifies the difference between two discrete time data samples, i.e. $\Delta x = x_k - x_{k-1}$.

$$\begin{aligned} V &= \mathbf{\Phi_c} c + \rho \\ V &= \begin{bmatrix} \bar{\upsilon}^{(1)} \cdots \bar{\upsilon}^{(n)} & \bar{\omega}^{(1)} \cdots \bar{\omega}^{(n)} \end{bmatrix}' \\ \mathbf{\Phi_c} &= \begin{bmatrix} u_l^{(1)} & u_r^{(1)} & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ u_l^{(n)} & u_r^{(n)} & 0 & 0 \\ 0 & 0 & u_l^{(1)} & u_r^{(1)} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & u_l^{(n)} & u_r^{(n)} \end{bmatrix} \\ c &= \begin{bmatrix} c_{11} & c_{12} & c_{21} & c_{22} \end{bmatrix}' \end{aligned} \tag{4.8}$$

Processing data from many observations is a least squares optimization problem where $u^{(i)}$ corresponds to a motor input command and $\bar{\mathrm{v}}^{(i)}$ the mean measured robot velocity from calibration experiment trial $i$. By rearranging the system of equations in 4.4 to obtain the regressor matrix in (4.8), one can calculate the optimal calibration matrix $\mathbf{C}$ using the pseudoinverse operation based on singular

value decomposition (SVD) [28], which is a single command in MATLAB [29]. The number of experiments used in this optimization problem can be quite large and is generally around 20.

$\rho$ is a random vector that is dependent on $e_l$ and $e_r$ propagating through the (assumed) linear system, measurement error from the vision system and the numerical sensitivity of (4.7). (4.8) thus requires the assumptions specified in (4.6) to hold.

### 4.4.3 Sensitivity and Error Analysis

Several types of error are of interest in improving odometry. The sensitivity of the regressor matrix can be calculated, bounding the errors of the estimated parameters due to the experiment design and calibration dataset. Instantaneous errors (deviations between desired and measured robot velocities) can be used to characterize the random errors in the kinematic model. When the robots are attempting to reach a destination, predicting positioning error in advance is critical in decision making for taking (computationally costly) sensor measurements. When using the heading estimator however, error in estimated magnitude of displacement over a unit time of travel is essential in characterizing heading estimation error.

#### 4.4.3.1 Observation Regressor Sensitivity

The sensitivity of the estimated parameters for the calibration matrix $\mathbf{C}$ (reordered as a vector $c$ as in (4.8)) can be bounded by the condition number of the

regressor matrix in (4.10) [28]. This effectively measures the sensitivity of the experimental data and the accuracy of the least squares solution. The condition number of the regressor matrix $\mathbf{\Phi_c}$ can be calculated using SVD as in (4.9).

$$\kappa(\mathbf{\Phi_c}) = \frac{\lambda_{max}}{\lambda_{min}} \tag{4.9}$$

Where $\lambda_{max}$ and $\lambda_{min}$ are the largest and smallest singular values of $\mathbf{\Phi_c}$ . For the ideal case, $\kappa(\mathbf{\Phi_c}) = 1$. The size of $\lambda_{min}$ is also of importance because large singular values correspond to low standard deviations of estimated parameters

$$\frac{\| c - \widehat{c} \|}{\| c \|} \leq \kappa(\mathbf{\Phi_c}) \frac{\| \rho \|}{\| V \|} \tag{4.10}$$

Since the regressor matrix only contains command inputs, one can design the calibration experiments ahead of time to reduce the numerical sensitivity of the pseudo-inverse operation on the regressor. We have observed that having only one wheel turned on at a time, i.e. using an indexed set of experiments such as $U = \{[1\ 0]', [.8\ 0]' \cdots [.2\ 0]', [0\ .2]', [0\ .4]' \cdots [0\ 1]'\}$ will result in $\kappa(\mathbf{\Phi_c}) = 1$. Unfortunately, slip conditions can invalidate certain experiments which will increase $\kappa(\mathbf{\Phi_c})$ for any real world experiment. While the friction wheel slip conditions of this calibration dataset may not translate well to the rest of the velocity space, this calibration dataset is effective at balancing the motor velocities for moving straight and turning left or right with one motor on with similar rates. These sensitivity metrics for the calibration experiment are in Table 4.1.

However, one downside of the metric in (4.10) is it fails to factor poor scaling

of the units of the parameters (i.e. mixing mm and radians results in misleading condition numbers). To account for this, one can change the units of the measurements to bring both rows of the calibration matrix to be of the same order of magnitude. Given the small dimension of the problem, we opted to pick units by guessing units and inspecting the calibration routine results by comparing the order of magnitudes of each row. We have found that using tenths of a meter (decimeters: dm) and radians are well suited given the physical dimensions of our robotic platform. More advanced techniques for using *a priori* information to determine the sensitivity of calibration data exist [28].

#### 4.4.3.2 Instantaneous error

$\rho$ in the optimization problem (4.8) is the offset or bias between the measured mean velocity $\bar{\mathrm{v}}$ and the predicted robot velocity given calibration data $\mathbf{C}$ and motor commands $u$ (4.11). $\rho$, combined with the signal variance, contribute to the mean squared error of the estimator.

$$\begin{bmatrix} \rho_v \\ \rho_\omega \end{bmatrix} = \begin{bmatrix} \bar{v} \\ \bar{\omega} \end{bmatrix} - \mathbf{C} \begin{bmatrix} u_l \\ u_r \end{bmatrix} \tag{4.11}$$

#### 4.4.3.3 Position Error

Euclidean magnitude displacement error is of greatest importance to the heading estimator. Such errors are representative of the odometer's uncertainty on the heading estimator. For each trial in the calibration experiment, the odometer predicts the robot's position after moving for two seconds. Note that each experiment is

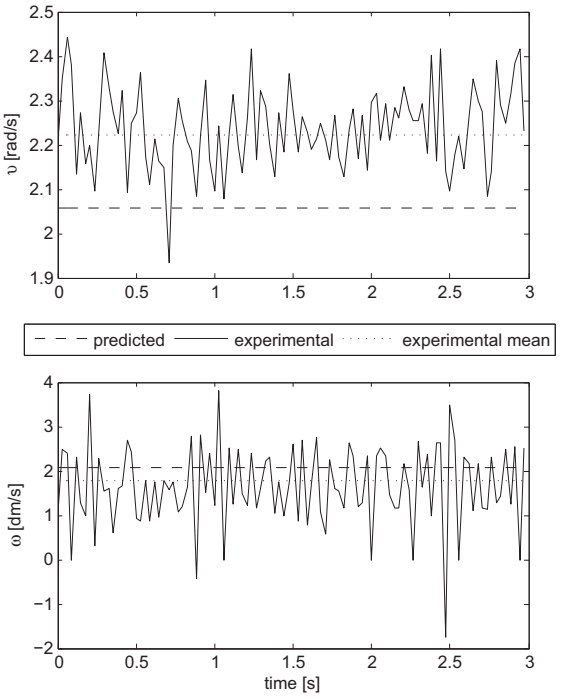


Figure 4.2: An example trial trajectory plotted in velocity space. This trial was chosen to highlight the graphical interpretation of $\rho$, the difference between the predicted v and the mean experimental value, $\bar{v}$.

of a different length, largely depending on how long the robot's trajectory is within the camera's field of view. This distance error, combined with its variance, will be useful in future uncertainty analysis. These errors for the calibration experiment are in Table 4.1.

## 4.5 Calibration Experiment and Results

From the linear least squares problem, it is important to note that the most numerically robust trajectories discovered only have one wheel powered at a time. We assume that these friction/slip conditions do not translate well to all other

| | |
|---|---|
| $N_{trials}$ | 24 |
| $\kappa(\mathbf{\Phi_c})$ | 2.1 |
| $\lambda_{min}$ | 1.3 |
| $\bar{D_{error}}$ [cm] | 4 |
| $\sigma_{D_{error}}$[cm] | 3 |

Table 4.1: Filter characteristics and associated design parameters

robot velocities (such as moving straight, both wheels turn at the same rate). For initial test purposes, we added less exciting trajectories to the dataset to reduce the (nonlinear) bias in the dataset at the cost of increasing the condition number of the data.

Cheaply made gearboxes that have a lot of backlash or play cause the robots to lurch at start up or stop time, throwing off initial angle conditions and reduce repeatability. Commercially available planetary gearboxes for pager motors such as the motors used in the KEPLR design do not have these limitations.

## 4.6 Heading Estimation Experiment

Between distance measurements, the follower robot turns at a preprogrammed turn rate using calibration data for a variable time interval determined by (1.2). The robot then moves forward about 20 cm (open loop) completing the movement phase and awaits for the next distance measurement. Several example trajectories of the robot are shown in Fig. 4.3. Note that the red circles around the leaders denote the target region for the follow, to be within 20 cm of the leader robot.

The initial two movement phases must be determined without sufficient sensor information. The structure of our code facilitated choosing initial conditions of the

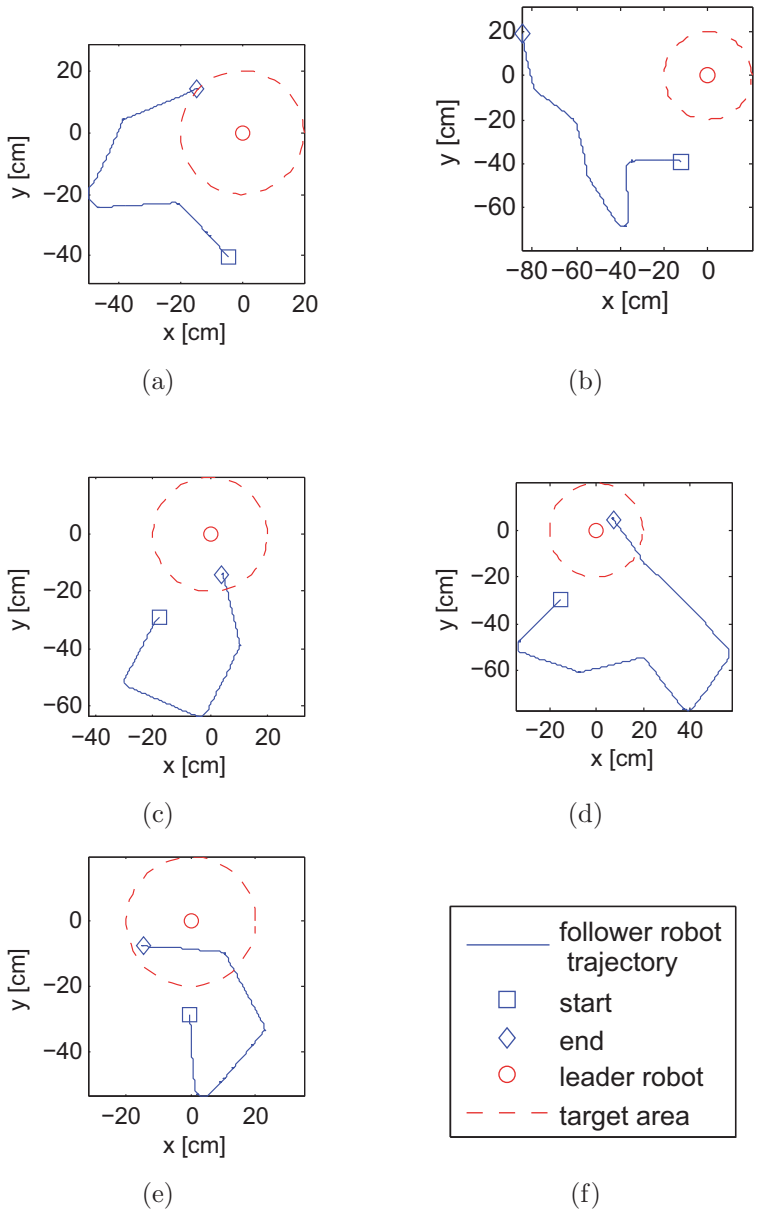


Figure 4.3: Five experimental runs using the robotic platform in Fig. 1.1. 4.3(a), 4.3(c), and 4.3(d) Robot succesfully rendezvoused with the leader. 4.3(b): Robot went outside of range of distance sensor (about 80 cm) and was unable to return given the size of the testbed. 4.3(e): Uncertainty in distance measurement (also compounded with uncertainty from the vision system) caused the robot to make an extra movement, still but still arrived at the leader's location.

required variables so the robot will move forward the first time and default turn left the second time. Once the robot makes the third distance measurement, the memory is fully populated with physical data and can predict the new direction to

turn each step.

Since the maximum of $q_k$ in (1.2) is 2 (occurs when $e_k = \pi$), the maximum time interval was chosen to have a maximum turn angle of about 120 degrees in a given motion phase.

## 4.7 Heading Estimation Error Analysis

During the heading estimation experiments, telemetry such as distance sensor data and $q_k$ was collected wirelessly from the follower robot. The points along the trajectories from the vision system where distance measurements were taken were extracted (the feature being the start of the new links or "elbows" in the path). An offline heading estimator computed $q_k$ from the vision system data. The differences in these results are shown in Fig. 4.4. Comparing these two results has a root mean square residual of 0.52.

It is also important to note that experimental values of $q_k$ have exceeded 2, which is outside its prescribed range. This is most likely because the triangle inequality for the three sides of the heading estimation triangle was violated when calculating $q_k$.

## 4.8 Analysis of Turning Decision Errors

The improved distance sensor from Chapter 2 results in improved system-level performance when estimating heading between two robots in a swarm using distance-only information. The heading angle (4.12) between a leader robot and a follower

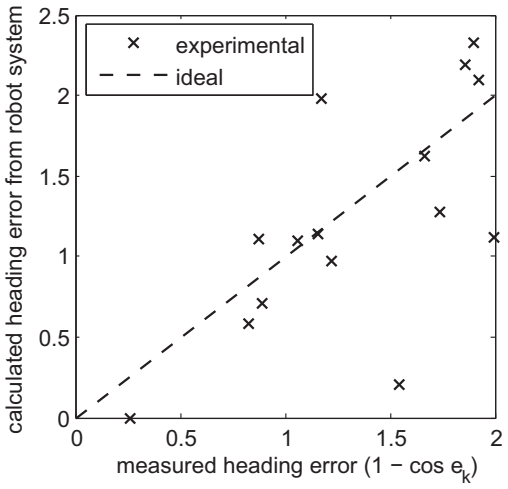


Figure 4.4: Error plot showing the difference between the computed heading error from the vision system and estimated heading error $q_k$ from the robot platform.

robot can be calculated using the law of cosines in Fig. 1.2. $\delta_k$ is measured using a distance-only sensor and $s_k$ is estimated using odometry/calibration information. Due to the symmetry of the cosine function, there is no unique solution for the heading angle. This is resolved by tracking multiple heading estimates and tracking the turning directions of the robot, using the conditional in (4.13) to drive the heading angle to zero.

$$q_k = 1 - \cos(e_k) = 1 + \frac{(\delta_k^2 - \delta_{k-1}^2 + s_k^2)}{2\delta_k s_k} \tag{4.12}$$

$$\text{If } q_k - q_{k-1} < 0 \Rightarrow \text{Change turning direction} \tag{4.13}$$

### 4.8.1 Linearized Error Propagation Model

Each of the sides of the triangle in Fig. 1.2, $(\delta_k, \delta_{k-1}, s_k)$, is a random variable with measurement uncertainty. We apply error propagation analysis which was covered Chapter 2, assuming each measurement $\sigma_{s_k}^2$ was calculated by using a vision

system observing the normed displacement of the robot between distance measurements. The robots were programmed to turn for a time interval $\propto q_k$, then move forward 10 centimeters. $\sigma^2_{\delta_k} = \sigma^2_{\delta_{k-1}} = \bar{\sigma}^2$ is the distance sensor resolution calculated by calibration experiments. The new method reduces both the RMS and maximum standard deviations by 27%, as seen in Table 4.2.

| sensor | $\sigma_{\delta_k}$ (mm) | $\sigma_{s_k}$ (mm) | $\sigma_{q_k}$ (RMS) | $\sigma_{q_k}$ (Max) |
|---|---|---|---|---|
| Old TDOA | 11 | 16 | 0.21 | 0.78 |
| New TDOA | 3 | 16 | 0.15 | 0.57 |

Table 4.2: Heading estimation standard deviation $\sigma_{q_k}$

One critical area of system performance is the sensitivity of choosing the turning direction using (4.13). Decision errors fall into two distinct classes when using uncertain measurements, $\hat{q}_k$, instead of the true values $q_k$ to calculate (4.13). For notational convenience, define $z = q_k - q_{k-1}$ and $\hat{z} = \hat{q}_k - \hat{q}_{k-1}$.

*False Positive (FP) Error*: The robot changed turning direction when it was not supposed to, i.e. $\hat{z} < 0$ when $z > 0$.

*False Negative (FN) Error*: The robot did not change turning direction when it was supposed to, i.e. $\hat{z} > 0$ when $z < 0$.

These FP/FN errors are analogous to Type I/II errors in statistical hypothesis testing. Errors of this nature drastically harm the convergence rate of the robot to its desired location since they cause the robot to turn in the wrong direction. Using the variances calculated using the error propagation model, we assume that $\hat{q}_k$ is normally distributed with variance $\sigma^2_{q_k}$, unbiased mean centered around the true value $q_k$ (4.14). We use the linear error propagation model to estimate the

probabilities of these errors occurring given sensor uncertainty.

$$\begin{aligned} \hat{q}_k &\sim \mathcal{N}(q_k, \sigma^2_{q_k}) \\ \hat{q}_{k-1} &\sim \mathcal{N}(q_{k-1}, \sigma^2_{q_{k-1}}) \\ \hat{q}_k - \hat{q}_{k-1} &\sim \mathcal{N}(q_k - q_{k-1}, \sigma^2_{q_k} + \sigma^2_{q_{k-1}}) \end{aligned} \tag{4.14}$$

(4.14) assumes that $q_i$'s are normally distributed and that $\hat{q}_k \perp \hat{q}_{k-1}$. This is clearly *not* the case since each computation of $q_k$ is dependent on $(\delta_k, \delta_{k-1}, \delta_{k-2})$ and $q_{k-1}$ is dependent on $(\delta_{k-1}, \delta_{k-2}, \delta_{k-3})$. The assumption simplifies the addition of two normal random variables.

### 4.8.2 Experiment Results

The experimental dataset consists of 14 trajectories of a traveling robot using the old TDOA sensor in [4] with the heading estimator. Each trajectory comprises a sequence of true heading error $q_i$ as captured by an overhead vision system, and a sequence of estimated angle error $\hat{q}_i$ calculated on the embedded system using distance-only measurements. The relationships between $z$ and $\hat{z}$ as observed over 14 different robot trajectories are shown in Fig. 4.5. Each quadrant of this graph corresponds to either a correct decision or FP/FN error.

In Fig. 4.5, the error bars shown for each data point are the estimated measurement standard deviations. The residuals of the dataset were distributed with sample mean 0.007 $\frac{\text{cm}^2}{\text{cm}^2}$ and standard deviation 0.75 $\frac{\text{cm}^2}{\text{cm}^2}$. Empirical probabilities yielding the percentage of data points in each quadrant of Fig. 4.5 are summarized

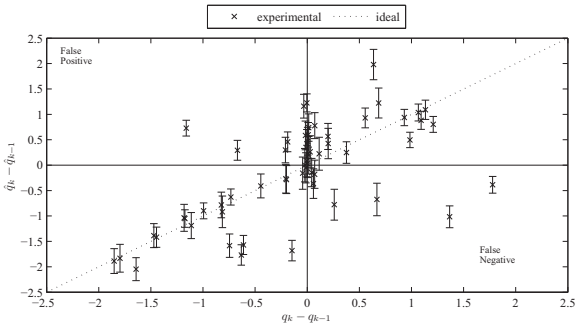


Figure 4.5: Comparison of $z$ to $\hat{z}$ highlighting turning decision errors from observed robot trajectories. The system used the old TDOA distance sensor when acquiring this data.

in Table 4.3.

| Region | Correct | False Positive | False Negative |
|---|---|---|---|
| Fraction in region | 0.71 | 0.16 | 0.12 |

Table 4.3: Empirical decision probabilities for the old TDOA sensor.

When $z \approx \hat{z}$, an estimate of the probability of the robot making a Type FP or FN error given measurement uncertainty can be calculated using (4.15).

$$\Pr(\text{Error}) = \begin{cases} 1 - \Phi(z, \sigma_{q_k}, \sigma_{q_{k-1}}) & \text{if } z < 0 \text{ (FP)} \\ \Phi(z, \sigma_{q_k}, \sigma_{q_{k-1}}) & \text{if } z > 0 \text{ (FN)} \end{cases} \tag{4.15}$$

The assumptions in (4.14) that are used to compute (4.15) assume that the means of the distributions are centered on the ideal line $z = \hat{z}$. Note that for many of the Type FP/FN errors, the data points were far from the line $z = \hat{z}$. It is most likely that the observed Type FP/FN errors did not occur as a result of normal noise. Pathological sensor errors can stem from other causes, for example, exceeding maximum range. Also note that errors may be correlated since multiple $\hat{q}_k$'s are dependent on the same $\delta_k$.

## 4.9 Discussion

The combination of the distance only sensor and heading estimator has proven an effective technique for a follower robot to locate and rendezvous with a leader. In practice, most errors in the heading estimation arise when the distance sensor saturates (the current system returns any distance measurement over 80 cm as 80 cm). Extending the range of the sensor (a practical design implementation issue) can mitigate this issue in future.

# Chapter 5

# Mixed-Signal Architecture of Randomized Receding Horizon Control for Miniature Robotics

## 5.1 Summary

Control of miniature mobile robots in unconstrained environments is an ongoing challenge. Miniature robots often exhibit nonlinear dynamics and obstacle avoidance introduces significant complexity in the control problem. Furthermore, miniature robots have strict power and size constraints, drastically reducing onboard processing power and severely limiting the capability of digital implementations of nonlinear model predictive controllers. To accommodate the demands of this application area, we describe the architecture of a mixed-signal mobile robot control system using randomized receding horizon control. We compare the proposed mixed-signal implementation with purely digital control systems in terms of power requirements and precision and find that the mixed-signal implementation offers significant reductions in power consumption at an acceptable loss of precision.

## 5.2 Introduction

Advances in sensing, actuation and battery technology have allowed for the development of very small robots (sub-$cm^3$). However, on-board computation for

small platforms has been limited by available micro-controllers, which are relatively large, consume significant power, and are too slow for some applications (such as flying micro robots). In this paper we propose a mixed-signal architecture implementing receding horizon control (RHC) strategies on small robotic platforms under significant power and space constraints. The architecture is designed for the control of a differential-drive miniature robot such as the one shown in Fig. 1.1. RHC controllers are highly adaptable to changing environments and can be used in a wide variety of systems. Our mixed-signal architecture is based on a randomized search of the allowable control inputs (actions) which can potentially be fast, small and low power compared to digital systems.

### 5.2.1 Overview of Receding Horizon Control

Receding horizon control (also known as model predictive control) is a class of control strategies in which the control input (action) at time $k$ is obtained by solving a finite horizon optimization problem that models the system's current state and future behavior of the system. The solution to this problem is a finite sequence of control actions, *but only the first element is applied to the real system.* At the next time step $k+1$, the procedure is repeated (see Fig 5.1). The repetition of this procedure effectively "closes the loop," providing updated information about the current system state to the controller. When the system state is not immediately available, an estimator that relies on sensor measurements is used.

In order to implement RHC, the future behavior of the robot must be simulated

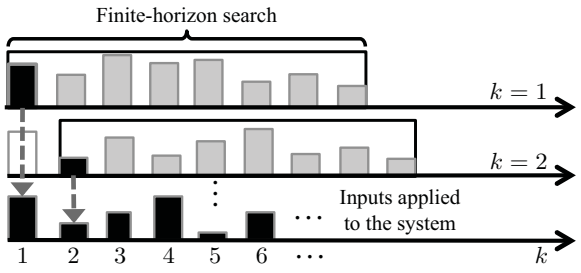


Figure 5.1: Overview of receding horizon control. At each time instance a finite solution is found, but only the first element is applied to the sytem.

using a model of its dynamics. The approximate behavior of the discretized system is given by:

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k) \tag{5.1}$$

where $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$. Based on the knowledge (or an estimate) of the current state $\mathbf{x}_k$, the controller needs to find a control sequence $\mathbf{u}_k^N = [\mathbf{u}_{1|k}, \cdots, \mathbf{u}_{N|k}]$ that minimizes a cost given by:

$$J_N(k) = \sum_{j=1}^{N} g(\mathbf{x}_{j|k}, \mathbf{u}_{j|k}) \tag{5.2}$$

where the stage cost $g : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ is convex and $\mathbf{x}_{j|k}$ satisfies (5.1) with $\mathbf{x}_{0|k} = \mathbf{x}_k$. The cost is used to set the objective of the robot, and usually penalizes deviation from a desired state and power consumption. Moreover, the states and control actions are constrained in that:

$$\mathbf{x}_{j|k} \in \mathbb{X}_k, \ \mathbf{u}_{j|k} \in \mathbb{U} \tag{5.3}$$

for all time instances $j \in \{1, ...N\}$. For robotics applications (5.3) represents obstacle constraints and is allowed to change over time and actuation constraints.

Moreover, the final state in the horizon has to satisfy a terminal constraint (which guarantees stability):

$$\mathbf{x}_{N|k} \in \mathbb{X}_k^T \tag{5.4}$$

for some set $\mathbb{X}_k^T$, which can be tuned to achieve a desirable performance. Finally, once a solution is found, only the first element $\mathbf{u}_{1|k}$ is implemented. At the next time step $k+1$, the process is repeated with knowledge of $\mathbf{x}_{k+1}$ (or an estimate $\hat{\mathbf{x}}_k$). This recursive procedure results in a sequence of control actions given by: $[\mathbf{u}_{1|0}, \mathbf{u}_{1|1}, \cdots, \mathbf{u}_{1|k}, \cdots]$. For a comprehensive survey on RHC, see [38].

### 5.2.2 Randomized RHC

Each finite horizon subproblem is usually solved using numerical algorithms, which enables RHC strategies to handle "hard" problems. This, however, poses a challenge: the time alloted for the controller to decide on a control action is limited by the sampling period of the system dynamics. For this reason for many years RHC was used only for problems whose dynamics were slow enough so that the optimization program had time to reach an acceptable solution. Many different ways of coping with this real-time constraint have been suggested. Most methods are beyond the processing capabilities of a state-of-the-art microcontroller [39, 40]. We propose an alternate solution addressing real-time processing constraints using a mixed-signal architecture.

Our architecture performs a randomized search in the space of allowed control inputs. Randomized strategies have been suggested in [41] that use control Lyapunov

functions, which may not be available. When using a randomized approach, it is practically impossible to find an optimal solution. Therefore, we need to accept feasible solutions that are not optimal, but still stabilize the system. In [42], it is shown that feasibility is sufficient for stability if we impose an extra constraint on the cost. For this architecture, we consider that the added stabilizing cost constraint is given by:

$$h(J_k) \leq 0 \tag{5.5}$$

where the function $h : \mathbb{R} \mapsto \mathbb{R}$ can be tuned to achieve a desirable performance.

The sub-optimality relaxation changes each optimization problem to the problem of finding one (any!) feasible solution. Here a feasible solution is a finite sequence $\mathbf{u}_f$ that satisfies (5.3), (5.4), (5.5). The high-level description of the proposed algorithm is as follows:

1. Set $k = 0$
2. Estimate $\mathbf{x}_k$.
3. (Randomly) generate a candidate solution $\mathbf{u}_k^N$.
4. Propagate states $N$ steps using (5.1).
5. Check constraints (5.3), (5.4) and (5.5). If check fails, return to step 3.
6. Implement $\mathbf{u}_{1|k}$. Set $k = k + 1$. Return to step 2.

In the subsequent sections, we propose a mixed-signal architecture to implement randomized RHC for a differential-drive two-wheeled robot.

## 5.3 System Architecture

For the RHC miniature robot controller, we assume that $\mathbf{x}_k = (x_k, y_k, \theta_k)$ and $\mathbf{u}_k = (u_{lk}, u_{rk})$, where $n = 3$ and $m = 2$. $\mathbb{U}$ is the space of all feasible motor controls.

The overall system architecture implementing RHC on a miniature differential drive robot is in Fig. 5.2. Primary sensor input comes from a distance-only sensor using Time Difference of Arrival as described in Chapter 2 [43]. An observer (Extended Kalman Filter, EKF) maps these observations into changes in the system state space $(x, y, \theta)$. The random trajectory generator uses multiple random number generators (RNG) and feeds them into an analog shift register, effectively creating a piecewise constant control signal with fixed time period $T$ between steps but random step values. Such control signals are easy to replicate with the motor controller using pulse width modulation (PWM). The dynamics simulator enables feedforward control and testing of candidate control signals by modeling the equations of motion of the vehicle. There are obstacle avoidance constraints using the IR sensors and stability or performance constraints using information from the dynamics simulator. The constraint and cost checker determines how feasible the candidate control signal is. Once a feasible control has been found, the first element of the control sequence will be sent to the motor controller for execution.

### 5.3.1 Random Number Generator & Shift Register

To sample the control space, we randomly generate $X_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$, so that $u$ is parameterized by a collection of random variables $X_i$. In order to produce

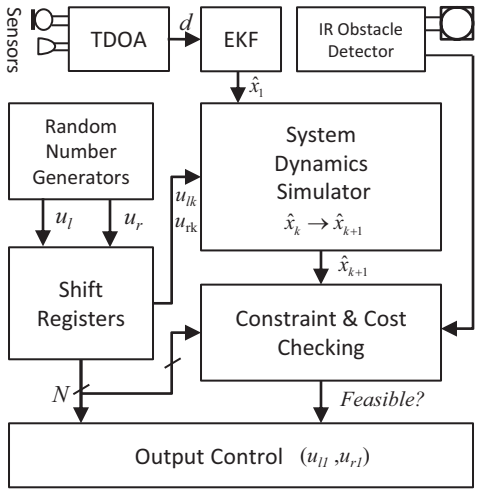


Figure 5.2: System-level design of a mobile robot with RHC

random variables that are approximately normal in distribution, the system will use several compact, Bernoulli true random number generators (RNG) based on amplified thermal noise [44]. These RNG's are combined using a digital to analog converter (DAC) to produce an analog random variable, which has a distribution similar to the Gaussian random variable. To have $N$ steps in the piecewise constant signal, 2 RNGs will be fed into two parallel to serial analog shift registers each of length $N$ (one for each motor controller).

### 5.3.2 Simulation of System Dynamics

In order to implement RHC, one must model system state dynamics. We have developed a mixed-signal kinematic model that maps motor commands to estimated and predicted changes in position in Euclidean Space (SE(2)). Specifically, given piece-wise continuous functions of time, we would like to solve the nonlinear

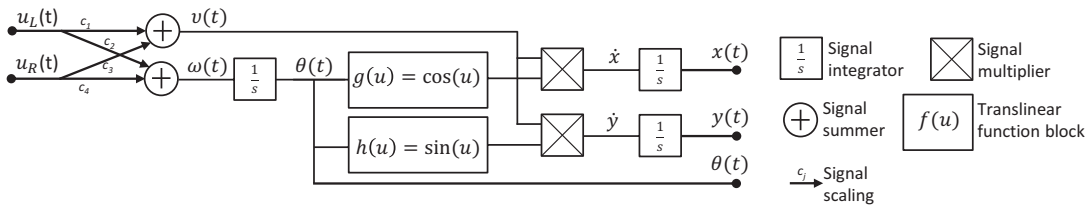


Figure 5.3: System-level design of analog robot kinematics simulator highlighting signal flow and primary computational blocks.

differential equations of motion (4.1). Closed form solutions of $(x, y, \theta)$ exist for piecewise constant $(v, \omega)$ which can be implemented directly in a digital system (4.2).

In an analog system, it is preferable to directly solve (4.1) given the fewer operations required to model the nonlinear system dynamics. At the computational level, this will require 4 signal scaling elements, two summing nodes, three integrators, two trigonometric function blocks and two signal multipliers (see Fig. 5.3). Translinear circuits offer significant computational capability and will be used to approximate trigonometric functions [45]. It is important to note that since we wish to solve the equations of motion in faster than real time, the circuits representation of time will be much faster (by several orders of magnitude) than the real world clock. The required precision of these components will be discussed later.

### 5.3.3 Constraint Checker and Cost Tracker

The two primary pathways for constraints checking are for obstacle avoidance and stability/performance. Information from the dynamics simulator plus the control signal can be used to assess the convergence of the robot to the desired

state. First, controller should progressively shrink the region of acceptable terminal conditions over time, using (5.6):

$$\|[x_k\ y_k]'\| < \beta\|[x_0\ y_0]'\| \tag{5.6}$$

$0 < \beta < 1$ in (5.6) remains fixed, scaling the size of the ball geometrically over time, and $\|\cdot\|$ is a (convex) norm. $L1$ norms are preferable for circuit implementation because they are simpler and operate well over a wide dynamic range. Absolute value circuits for use in the $L1$ norm do not require current scaling unlike squaring circuits for the $L2$ norm used in quadratic costs. In order to guarantee stability, one must also bound the cost $J_N(k) < c$, where $c$ is a tuning parameters that may change for each iteration of the RHC strategy

In addition, IR proximity sensors positioned around the robot detect any obstacles blocking the robot's trajectory. The IR sensors feed directly into the constraint checker, effectively "short-circuiting" the controller to run in a fail-safe mode upon imminent collision, increasing the flexibility of RHC strategies for mobile robot control.

## 5.4 Anticipated Performance: Analog vs. Digital

### 5.4.1 Circuit Power

The power consumption requirements will differ for the analog and digital implementations of the simulator. From the system diagram, we can approximate the power consumption of the analog circuit by estimating the number of bias currents

required. We assume the use of Gilbert cells for signal multiplication, an operational amplifier with a capacitor for integration, and the equivalent of five differential pairs in circuits used to approximate trigonometric functions [45]. This requires a total of 14 bias currents. Table 5.1 further elaborates on the number of bias currents required. Assuming $I_{bias} = 100\text{nA}$, $V_{DD} = 5\text{V}$, and $10\mu\text{s}$ to 1ms of circuit operation time, the analog circuit consumes 70pJ to 7nJ,

For the digital system, we assume that the closed form solutions to the equations of motion (4.1) are implemented on a microcontroller comparable to the TI MSP430 with a hardware multiplier. (4.2) requires 11 multiplications, 1 division, and 12 additions using Taylor series expansion. We assume that each operation takes five clock cycles ignoring memory access costs. In addition, we also ignore the limiting case of $\omega_k \to 0$, which would further increase the complexity of the digital implementation. Dramatically greater computational costs would occur if time-stepping approximations were used to solve the equations of motion. The energy required to perform the computation is independent of the clock, but assuming $100\mu\text{A/MHz}$, $V_{DD} = 3\text{V}$ and a 32 MHz clock, the equations of motion can be solved

| Component | Implementation | $N$ elements | $N$ $I_{bias}$ per element |
|---|---|---|---|
| signal multiplier | Gilbert multiplier [46] | 2 | 1 |
| trigonometric function | tanh approx. [45] | 2 | 5 |
| integrator | op amp and cap | 3 | 1 |
| mode conversion | | 2 | 1 |
| Total | | | 14 |

Table 5.1: Circuit components, proposed design and expected power consumption for analog implementation.

in 0.1$\mu$s consuming about 35nJ of power.

### 5.4.2 Circuit Precision

For the analog circuit, we assume that each of the components have a linear distortion error, i.e. $\sigma < 1\%$ [46]. Assuming that the errors are independent, and compound at each stage as a summation of Gaussian random variables, the maximum depth of the circuit is 5 stages so the circuit error would be bounded by a 2.2% error where $\sigma_{system} = \sqrt{\sum_{i=1}^{N} \sigma_i^2}$. This would result in a lower bounded signal to noise ratio ($SNR$) of 45/1, where $SNR = \frac{1}{\sigma}$. To achieve an $SNR$ of 100/1, the RMS error of all stages would need to be $< 0.4\%$.

The $SNR$ of the digital circuit is determined by the number $M$ of bits used in the circuit. 8 bit calculations result in a maximum theoretical $SNR$ of 256/1, where $SNR = 2^M$. This $SNR$ can be scaled exponentially by a linear increase in the number of bits and power consumed [47] but subsequent operations would not necessarily scale linearly.

### 5.4.3 Summary

The previous analysis suggests that the analog implementations can model the system dynamics with significantly less power usage. However, with digital systems, greater precision can be achieved efficiently by increasing the bit length of the signal. Therefore, analog systems are well suited for fast, low power, but less precise computations, which is applicable to problems in robotics.

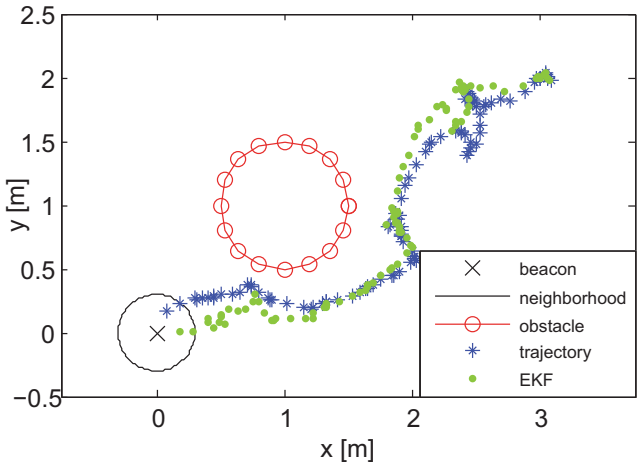


Figure 5.4: Simulated example of the proposed architecture.

## 5.5 Simulation

We developed a system-level simulation of the proposed architecture in Fig 5.2, including obstacles. The robot must track the position of a single beacon and move to its location avoiding obstacles. Measurement uncertainty, model parameters and system noise were extracted from sensor and motor calibration experiments on the platform shown in Fig. 1.1 and were incorporated into the simulation. Fig. 5.4 shows an example trajectory generated by the controller.

The RHC algorithm is sensitive to the distribution of the control variables that are randomly selected. Picking trajectories in either the motor speed space $(u_l, u_r)$ or the robot body speed space $(\upsilon, \omega)$ will also bias the simulation. For example, a small imbalance in selecting motor speeds for a robot with a small wheelbase can cause the robot to spin quickly if care is not taken.

These effects can be better understood by incorporating the linear map associating motor speeds to robot body velocities. Chapter 4 discusses the relation


Figure 5.5: Simulated example of the proposed architecture.

$[\upsilon\ \omega]' = \mathbf{C}[u_L\ u_R]'$. $\mathbf{C}$ will affect the biasing of random variable generation. The magnitudes of the elements in the two rows of $\mathbf{C}$ can differ by over 1.5 orders of magnitude, causing the robot's tendency to spin for example.

### 5.5.1 Obstacle Detection in Simulation

Fig 5.5 shows how the active sensing region of an obstacle detection sensor is represented. If a point on the boundary of an obstacle goes inside the the sensor's sensing region (represented as a triangle by three points relative to the robot body frame), an obstacle is detected. The obstacle detection algorithm uses a point-triangle collision detection scheme to determine whether a 2D point (or a set of points) that comprises an obstacle is inside a triangle using the following test (testing point $P$ inside triangle $ABC$):

$$\text{Area } ABC \stackrel{?}{=} \text{Area } PAB + \text{Area } BPC + \text{Area } PAC \tag{5.7}$$

These areas are efficiently calculated using the geometric interpretation of the cross product (area of the parallelogram) calculated using a determinant. If (5.7) holds, then $P$ is inside $ABC$ and the obstacle is detected by the proximity sensor.

## 5.6 Conclusion

The mixed-signal randomized RHC controller satisfies the strict design requirements of a miniature mobile robot. Detailed analysis of the dynamics simulator suggests that an analog or mixed-signal implementation can dramatically reduce power consumption at an acceptable loss in precision. Reductions in power consumption using a mixed-signal implementation compared to a digital implementation will decrease from 35nJ to 70pJ-7nJ per odometry computation. The change in SNR of 255/1 for 8 bit computations to the specified SNR of 100/1 for mixed signal circuits is an acceptable loss in precision.

The authors thank the ANTBOT team. This material is based upon work supported by the National Science Foundation under Award Nos. 0647321, 0755224, and 0931878, and the ONR AppEl Center at UMD.

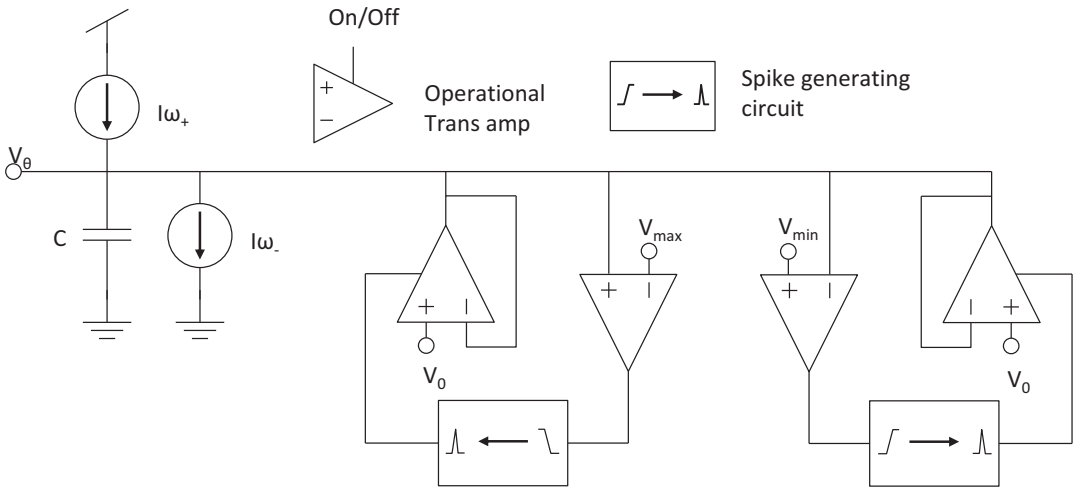


Figure 6.1: System-level design of integration and modulus circuit. Not that equivalent blocks are not necessarily identical at the transistor level.

# Chapter 6

# Mixed-Signal Odometry for Mobile Robotics

## 6.1 Current Integration and Modulus Circuit for Turning Rate and Angle

The first section of the circuit to discuss is the integration of the signal $\omega$ to $\theta$. Integration of differential current-mode signals is conceptually straightforward. The positive component of the signal is sourced onto a node with capacitance $C$ and the negative component sinks current away from the same node. Given the current-voltage relationship of a capacitor, current mode signals are integrated, resulting in a voltage that's scaled by the capacitance at the node. This results in $\Delta I_\omega = I_{\omega_+} - I_{\omega_-}$ and $V_\theta(t) = \frac{1}{C}\int \Delta I_\omega(t)dt$

### 6.1.1 Time Scaling and Resulting Choice of Capacitance

The selection of the size of the capacitor and the range of currents used in the circuit is dependent on the circuit time scaling. Assuming that the current mode signals have amplitudes between 1nA and 100nA (this range is well within operational limits of subthreshold circuits in the ON 0.5 $\mu$m process), and the time scaling constant $\tau_{scale}$, defines how many seconds of world time are simulated in one second. For the circuit, we set the maximum rotation rate to $\Delta I_{\omega,max} = 100nA \leftrightarrow \omega_{max} = 2\pi\frac{\text{rad}}{\text{s}}$. This maximum turning rate is more than sufficient to model the maximum stable rotation rate of the robotic platform in Fig. 1.1 and other foreseeable platforms currently in development. The maximum slew rate $SR$ for the integration circuit and corresponding capacitance $C$ can therefore be calculated (6.1).

$$\begin{aligned} SR &= (V_{2\pi} - V_0)\tau_{scale} \\ C &= \frac{\Delta I_{\omega,max}}{SR} \end{aligned} \tag{6.1}$$

Capacitor sizing considerations must also be made to minimize chip area. Increasing $\tau_{scale}$ will reduce $C$ and thus the chip area required for the capacitor. MOSCAPs have the highest capacitance per unit area for the proposed process (2400 aF/$\mu$m$^2$) but suffer from nonlinearities (confirm?). For this design $\tau_{scale} \approx 10^5$ and (6.1) suggests $C \approx$ 1pF. Limitations aside, a square 1pF MOSCAP would have to be drawn with $W = L = 20\mu$m which is quite large but feasible to be on chip. Note that three capacitors around this size are required for the entire odometry circuit.

## 6.2 Modulus Circuit

We define the trigonometric shaping circuits over two periods to prevent multiple subsequent snaps in case the signal $\theta(t)$ is biased around 0 or $2\pi$ with some alternating small signal component. Small perturbations in this case will cause the modulus circuit to continually fire. Instead, having two modulus circuits wrapping $\pm 2\pi$ back to zero will alleviate this issue.

Fig. 6.1 shows a system level design and operation of the modulus circuit. First, $V_\theta$ is compared to the threshold using a comparator. At the transistor level, this is simple transconductance amplifier such as the one in Fig. 6.2.

When $V_\theta$ crosses a threshold $V_{min}$ or $V_{max}$, the respective comparator rises high. This rising edge is fed into a spike generating circuit, converting the rising edge into a spike. This voltage spike turns on a buffer amp to drive $V_\theta$ to $V_0$. In principle only one buffer amp is necessary but for design flexibility we use two buffer amplifiers.

### 6.2.1 Comparator

The only important metric of the comparator is the $\frac{V_{DD}}{2}$ cross over point of the voltage output. This crossover should correspond with $V_\theta = V_{min}$ or $V_{max}$. The circuit's linear region or gain is not of great importance since the output will be filtered through several CMOS inverters which are effectively high gain amplifiers.

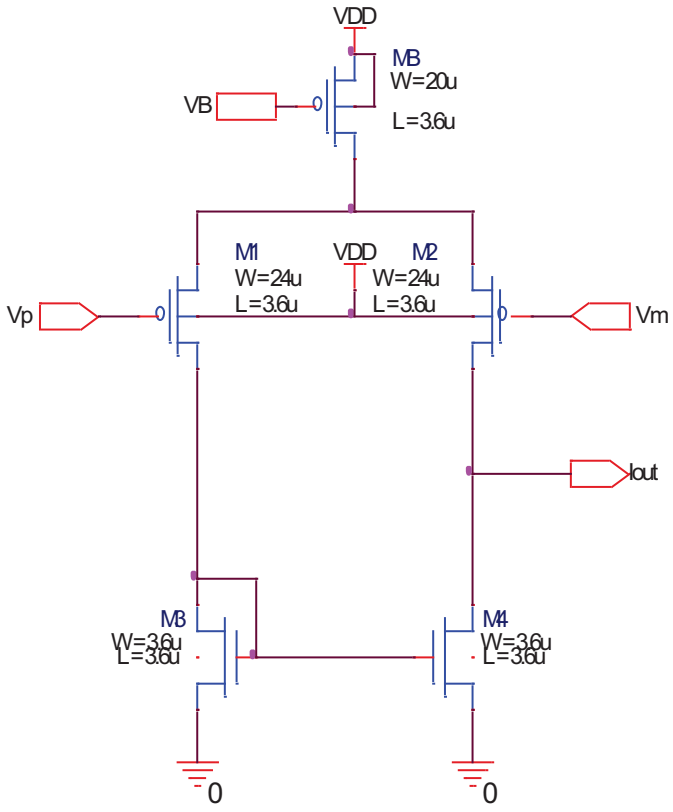


Figure 6.2: PFET variant of the transconductance amplifier used for both the comparators and buffer amplifiers. In some cases NFET variants are used to resolve the $V_{max}$ or $V_{min}$ limitations of the PFET trans amp. Sizing parameters are from a buffer amplifier

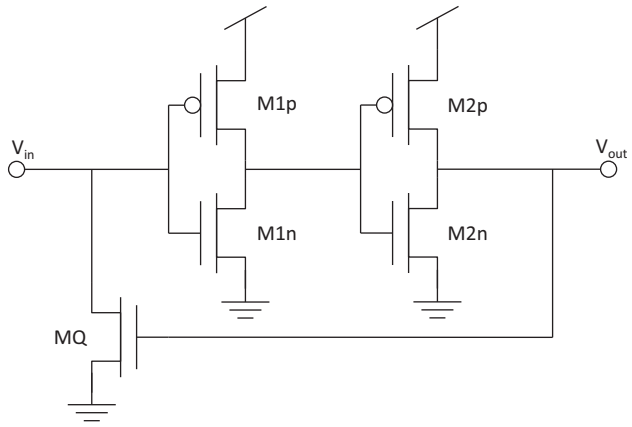


Figure 6.3: Spike Generating Circuit.

### 6.2.2 Spike Generating Circuit

Fig. 6.3 shows the NFET spike generator.

Note that if a PFET buffer amplifier is used, the spike generator output is run through an additional inverter before controlling the PFET buffer amplifier, thus flipping the spike. Inverters have been biased to have their switching point $V_t = \frac{V_{DD}}{2}$.

Subsequent work incorporated cascode transistors into the spike generator to increase control over spike parameters. Properly setting the cascode bias voltages will "starve" the inverter's current output and will give independent control of the spike's rising and falling edge. An alternative option is to cascode feedback transistor MQ in Fig. 6.3.

The feedback transistor MQ quenches the output of the comparator circuit after the output of the. If the generated spike does not sufficiently reset $V_\theta$ and turn off the comparator, instabilities from this feedback transistor arise. It is likely that this circuit can act like a 3-ring oscillator.

The current design is capable of producing a spike that with a peak amplitude around 1.75V (NFET case) and a duration of 80ns (20ns above 1V). This corresponds to 8 ms in real world time (signals that occur within this window are ignored by the integration circuit), which is more than sufficient regarding odometry accuracy.

### 6.2.3 Buffer amplifier

One important metric of the modulus circuit is the buffer amplifier's slew rate. This slew rate must be several orders of magnitude faster to both override the input current and quickly reset $V_\theta$. Any current signals during this reset time are

irrecoverably lost.

However, too large of a slew rate (or equivalently, too short of a reset time) requires wide transistors, in turn increasing parasitic capacitances that can store collapsing currents resulting in $V_\theta$ voltage offsets after the reset. Subsequent work to reduce the effects of this leftover charge is to incorporate dummy transistors into the design to "soak up" any remaining charge while the currents collapse.

### 6.2.4 Modulus Circuit Results

Simulation results for the modulus circuit are shown in Fig. 6.4. An offset of $\Delta V = 200\text{mV}$ was common for the circuit simulation. Increasing spike width by adding current-starving cascode transistors and adding dummy transistors to sink collapsing currents was able to reduce this offset from $\Delta V = 200\text{mV}$ to $\Delta V = 50\text{mV}$. This is without modeling or parameter tuning, which could further reduce this offset by incorporating the circuit model into proper parameter selection.

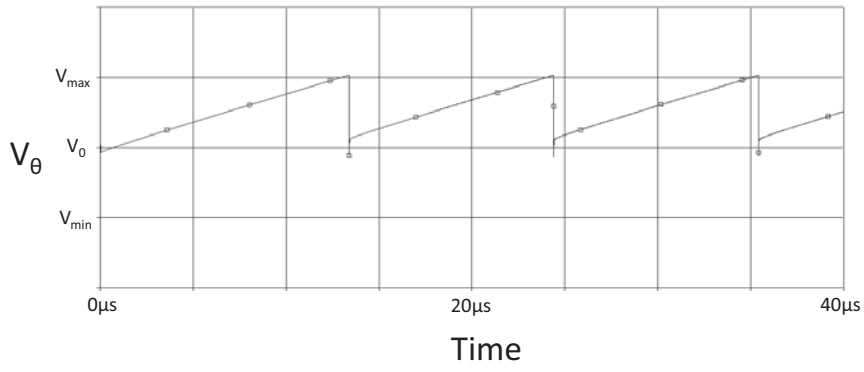


Figure 6.4: Modulus circuit simulation results with $\Delta I_\omega$ =100nA.

## 6.3 Sine Shaping Circuit

### 6.3.1 Design Overview

The functionality of a sine shaping circuit is to map a (DC) voltage to its sine, i.e. $V_\theta \mapsto \sin(V_\theta)$. Compared to Taylor Series expansion, series expansion using hyperbolic tangents [48] offers the capability to define the approximation on an arbitrary range. From the large signal model of the MOSFET in weak inversion, it is straightforward to show that the large signal model of the differential pair is hyperbolic (6.2):

$$\Delta I_{out} = I_B \tanh(\frac{\kappa}{2U_T}(V_\theta - V_{ref})) \tag{6.2}$$

This suggests using MOSFET differential pairs operating in the subthreshold region to model trigonometric functions. A rigorous basis for the relationship between sinusoids and (infinite) summations of hyperbolic tangents is provided [48]. Connecting five differential pairs in parallel can define a valid sine approximation over $\pm 2\pi$.

Self-cascoded current mirrors produce the tail currents in the differential pairs.

Unless otherwise specified, all transistors in the circuit were drawn with $W = L = 3.6\mu m$.

Two modifications were made to the circuit design in [45]. First, differential pair biasing was achieved by a resistor network instead of changing the well potential of the PMOS transistors.

Also note that the current use of PMOS transistors in the final design is more from the mentioned circuit modifications than by design

$V_{DD} = 5V$ may seem excessive for powering subthreshold circuits, but this range was necessary to keep all differential pairs in the sine shaping circuits operating correctly.

#### 6.3.1.1 Source Degeneration

It is possible to model source degeneration of the differential pair as a reduction in $\kappa$ of the original transistors in the differential pair. Adding source degeneration reduces $\kappa$ by the relation in (6.3):

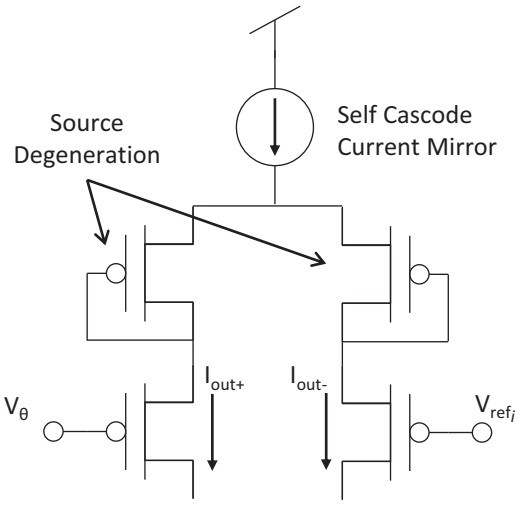


Figure 6.5: source degenerated differential pair used in the sine shaping circuit.

$$\text{SD: } \kappa \mapsto \frac{\kappa^2}{1+\kappa} \tag{6.3}$$

Analyzing the IV characteristics of a source degenerated transistor suggests that the approximation in (6.3) is only valid in the specified $V_{sg}$ range of 1-2.25 V.

$\kappa$ is reduced even further outside that range. $\kappa$ as a function of $V_{sg}$ of a source degenerated PFET transistor is shown in Fig. 6.6.

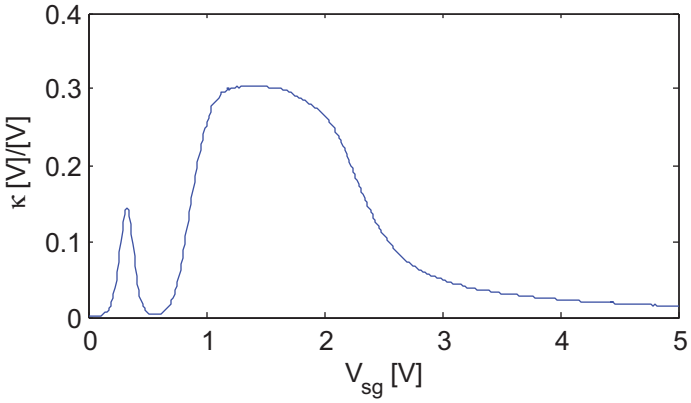


Figure 6.6: DC sweep of $V_{sg}$ measuring $\kappa$ of the source degenerated multiple transistor element.

### 6.3.2 Modeling and Analysis

The large signal model of the sine shaping circuit assumed the use of ideal current sources. The summation of differential pairs modeled using (6.2) with source degeneration using (6.3) is compared to PSPICE DC sweep simulation results. The sine shaping circuit characteristics can be modeled with (6.4):

$$\Delta I_{out} = \sum_{i=-2}^{i=2} (-1)^i I_B \tanh(\lambda(V_\theta - V_{ref_i})) \approx \sin V_\theta \tag{6.4}$$

where $\lambda = \frac{SD(\kappa)}{2U_T} = \frac{\kappa^2}{2U_T(\kappa+1)}$ and $V_{ref_i} = \{1, 1.5, 2, 2.5, 3\}$.

The large signal model had bounded (i.e. maximum) error accuracy error of 5% and an RMS error of 3%. The more important metric, sinusoid fitting error (i.e. the accuracy of the sinusoid approximation), are around 8% and 5% respectively. However, these uncertainties are not reliable for two reasons. There is ambiguity in

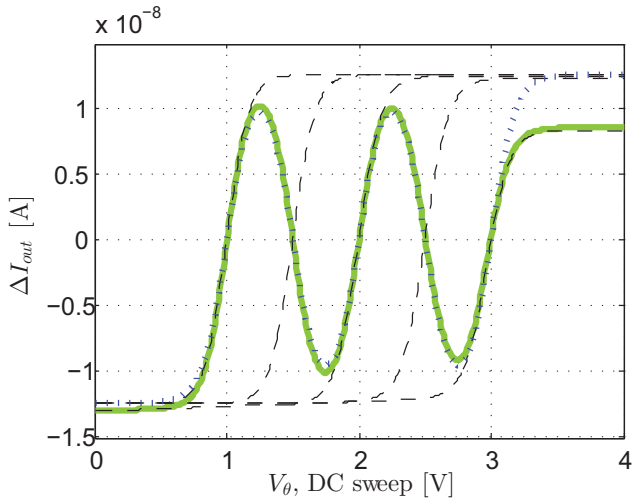


Figure 6.7: A comparison of the large signal model to the circuit simulation. Blue is the large signal model (MATLAB), green is the PSPICE simulation, and the black dotted lines are the individual differential pair current outputs.

determining the amplitude of the sine wave to fit to the PSPICE data. Maximum error is especially sensitive to this choice. Second, optimization techniques such as gradient descent have not tuned circuit parameters to reduce this error, so these numbers could be misleading or unfair.

(A quick note on parameter tuning and gradient descent: if the objective is to calculate the norm of the error between two continuous functions, the problem lies in an infinite dimensional vector space but to circumvent this tediousness we will only look at the error for a discrete collection of points, reducing the problem to the $\mathbb{R}^N$ case where $N$ is sufficiently large ($100 < N < 1000$).

One important point to make is that the RMS model error (3%) is *above* the desired RMS sinusoid fitting error of $0.1\% < \sigma < 1\%$. It is doubtful but not yet disproved that techniques such as gradient descent can reduce fitting errors to below model errors. For accurate sine shaping, more comprehensive circuit models may

have to be developed. Early effects and nonideal current sources may need to be accounted for in future circuit models. Modeling variations in $\kappa$ as a function of $V_{gs}$ i.e. $\kappa(V_{gs})$ to capture the effects shown in Fig. 6.6 also seems like an interesting avenue to improve model accuracy.

### 6.3.3 Cosine shaping circuit

Three design modifications are necessary to the sine shaping circuit to create a cosine shaping circuit. First, the trigonometric identity $-\sin(\theta - \frac{\pi}{2}) = \cos\theta$ suggests using a quarter period shift (0.25V) in all the reference voltages, plus flipping the differential pair current outputs to account for the negative sign to design the cosine shaping circuit. However, this also shifts the operating range of the approximation to $[-2\pi - \frac{\pi}{2}, 2\pi - \frac{\pi}{2}]$. An additional differential pair must be added to ensure that the interval $[-2\pi, 2\pi]$ (the interval that the sine shaping circuit is accurate) is contained in the approximation of the cosine shaping circuit. Unfortunately, adding this additional differential pair adds an additional current to source or sink, shifting the current output when $\sin V = 0$ to $I_B$. An additional constant current sink to shift this zero-level current output is necessary.

## 6.4 Multiplier Cells

Several criteria were necessary for the multiplication circuitry. Subthreshold operation is a necessity. An RMS multiplication error of $\sigma < 1\%$ is also a design specification.

Both the velocity signal and the outputs of the trigonometric signals are differential current-mode signals. This suggests a four quadrant current-mode multiplier. Four quadrant Gilbert cells have been existence since the BJT era [46], but for this design we have opted for a modern CMOS design currently used in artificial neural networks [49]. This modern design consists of a four quadrant gilbert cell with differential current-mode signal inputs and outputs, and additional transistors (current mirrors) to set the common mode of the circuit.

Analyzing the current mode Gilbert cell in Fig. 6.8 using the Translinear Principle yields sufficient results that agree with the large signal model since transistors are not stacked and there are an equivalent number of transistors in each direction of the loops (Invoking the Translinear principle when stacking occurs will miss $\kappa$ terms that appear in the large signal model and an imbalance in the number of transistors per side leads to missing quiescent currents $I_0$ that appear in the large signal model.).

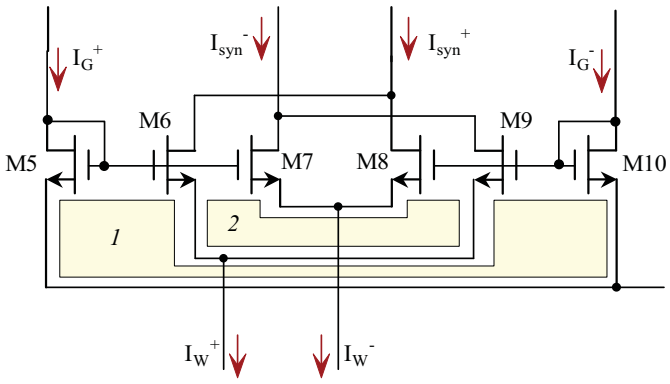


Figure 6.8: Current-mode four quadrant Gilbert cell from [49]. This figure is copied from the reference.

Fig. 6.9 shows some results for the current mode Gilbert Multiplier cell. The

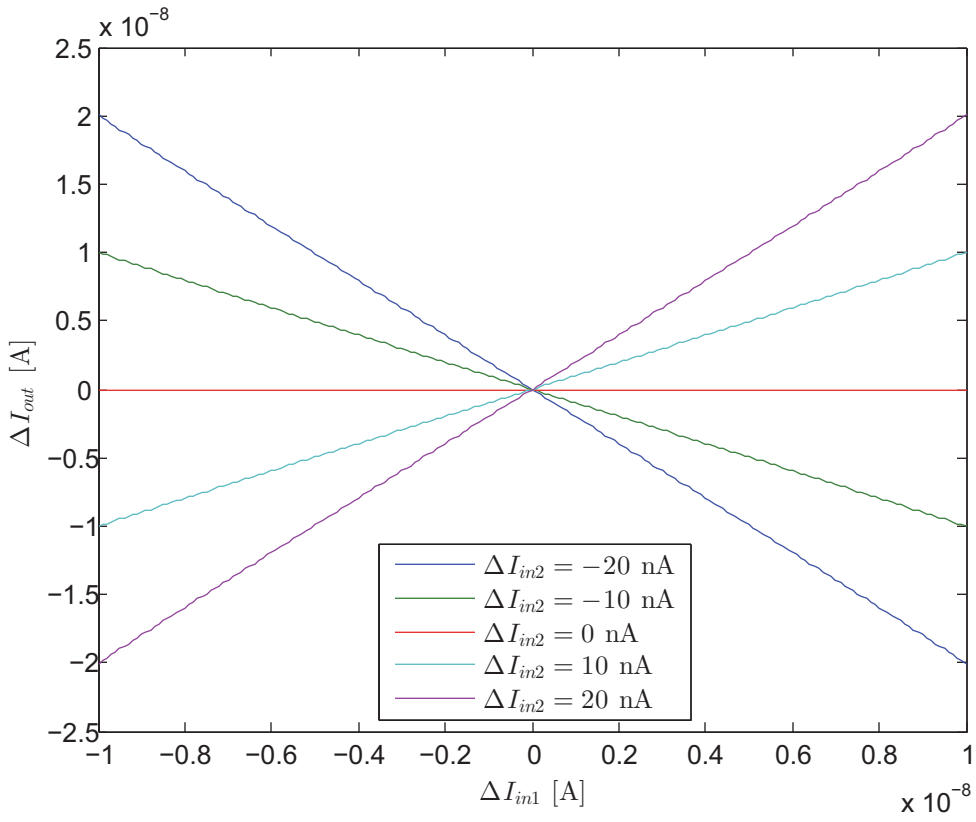


Figure 6.9: Gilbert Multiplier DC Sweep Simulation results.

multiple curves are for $\Delta Iin_2$ = [-20 -10 0 10 20] nA. $\Delta Iin_1$ was swept from -20nA to 20nA which is the x axis of the graph. Circuit parameters have not been automatically tuned to further reduce error. Fig. 6.10 shows the error of the Gilbert Cell for the same simulation (subtracting the ideal multiplication operation from the circuit output). An RMS error was computed for each value of $\Delta Iin_2$. The maximum RMS error was 0.6% and occurred when $\Delta Iin_2 = \pm 20nA$, which is within the specified design tolerances.

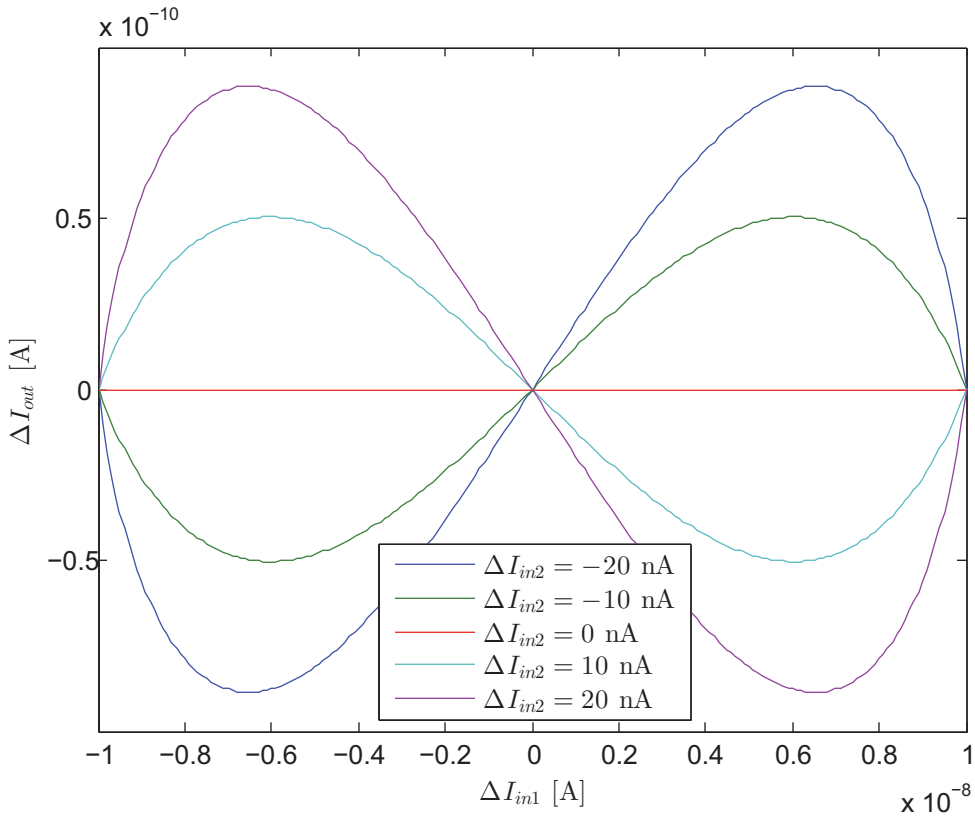


Figure 6.10: Gilbert Multiplier DC Sweep Simulation error results.

### 6.4.1 Self-Cascoded Current Mirrors in Translinear Circuits

One question worth asking is whether or not self-cascoding current mirrors improves their performance in subthreshold, large-signal (i.e. translinear) circuits. Self-cascoding dramatically increases the output resistance of the small signal model. Self-cascoding effectively has little effect on the DC terms (barring the $V_{max}$ and $V_{min}$ problems) and drastically reduces the first order effects of voltage mismatch. The effects of higher order terms are not analyzed, but analagous to Taylor Series approximation tricks, self-cascoding does improve circuit performance as observed in copying currents for the gilbert multiplier cell output for integrating values of $x$ and $y$.

While often the $V_{max}$ problem is a circuit limitation, reducing $V_{max}$ may be beneficial for reducing the Early Effect if $V_{DD}$ is high. Alas, unlike in the above threshold case, adding the self cascode only requires an additional 100mV so they cascoding does not affect the common mode as much.

## 6.5 System-Level Integration

One of the challenges of the system-level integration is properly sinking or sourcing the output differential current of the gilbert cells. A current mirror copies the negative component to be able to sink it off the integrating capacitor. However, a simple current mirror has too low an output resistance which will sink *additional* current off the node. Cascoding this current mirror solves this problem.

Including additional copies of the modulus circuit regulating the integration nodes for the $x$ and $y$ signals resolves any $V_{min}$ or $V_{max}$ problems. The modulus circuit was used for this purpose merely for design re-usability and illustrative purposes, while a saturation suited would produce more numerically stable solution as in digital systems.

## 6.6 System-Level Simulation Results

Fig. 6.11 shows the result of the transistor-level simulation of the odometry circuit in PSPICE. Constant differential currents representing constant $(\upsilon(t), \omega(t))$ are fed into the odometry circuit. The displayed waveforms in Fig. 6.11 correspond to $(x(t), y(t), \theta(t))$. The resulting solutions of the equations of motion in (4.1) suggest

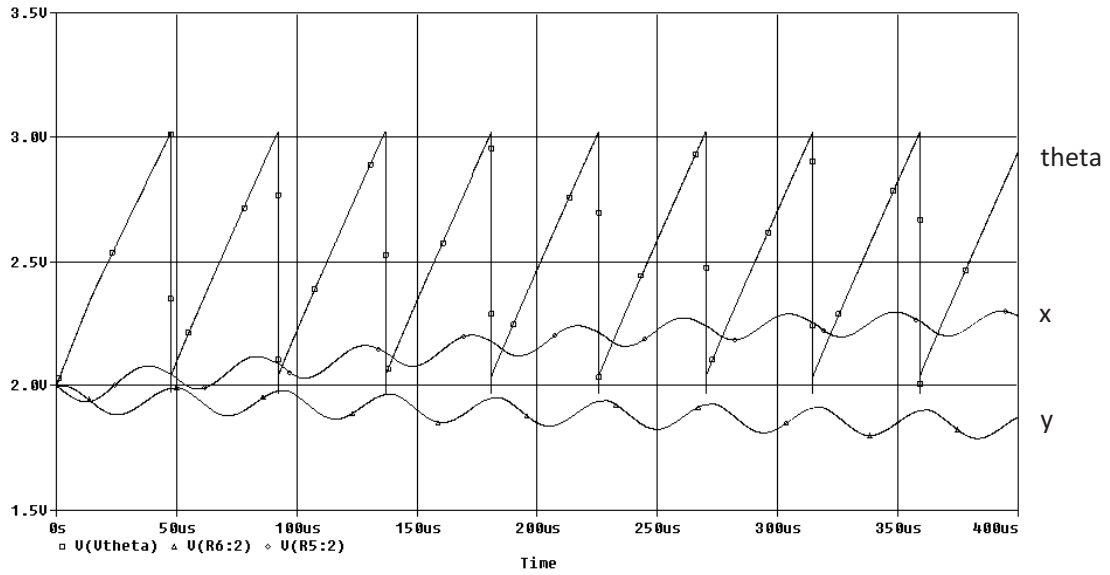


Figure 6.11: System-Level odometry circuit results

that for constant $(\upsilon(t), \omega(t))$, $x(t)$ and $y(t)$ should be sinusoids of equal amplitude and 90° out of phase while $\theta(t)$ is a ramp function with the same period rising between 0 and $2\pi$. There are apparent issues with the transients which are due to the voltage operating points of the current mirrors used in copying currents for the integration nodes for $x$ and $y$. Qualitatively, simulation results agree with expected system solutions.

## 6.7 Conclusion

The mixed-signal odometry circuit satisfies the strict design requirements of a miniature mobile robot. Detailed analysis of the dynamics simulator suggests that an analog or mixed-signal implementation can dramatically reduce power consumption at an acceptable loss in precision.

# Chapter 7

# System-Level Integration and Future Work: KEPLR-D

## 7.1 Summary

The long terms goals of this research project anticipate the use of mixed-signal architectures to reduce the size and power constraints, such as the architecture in Chapter 5. To be able to implement such an architecture, this requires complete control of the hardware design and a modular architecture to be able to swap out digital components for mixed-signal components. While there are many practical challenges to designing a new robot chassis that are not typically the focus of research, it is essential that they be addressed to continue research. Given the desired small size constraints of the platform, board-level commercial solutions are impractical. Commercially available integrated circuits and components will fulfill the majority of system requirements. Research progress made on the Walle bot platform has discovered several hardware limitations for the digital implementations. The fully digital design of the robot, KEPLR-D, addresses *all* of these limitations at an acceptable increase in circuit complexity and power consumption. We will also discover how findings from Chapters 2, 3, 4 and 6 tie together and will guide the process of developing a platform to implement the mixed-signal architecture as proposed in Chapter 5. Pictures of KEPLR's chassis and drive train are shown in Fig. 7.1 and 7.2.

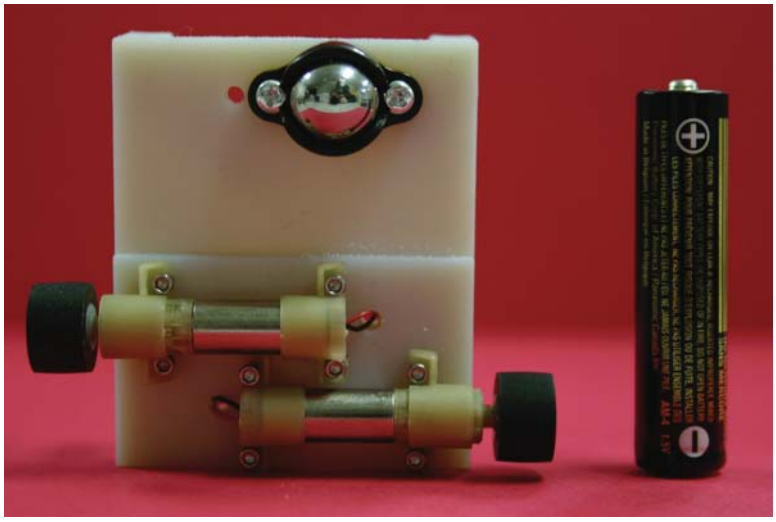

Figure 7.1: KEPLR'S chassis: bottom view. This provides an excellent view of the planetary gearmotors (gearboxes come preinstalled on pager motors) and the "caster wheels." AAA battery shown for reference.

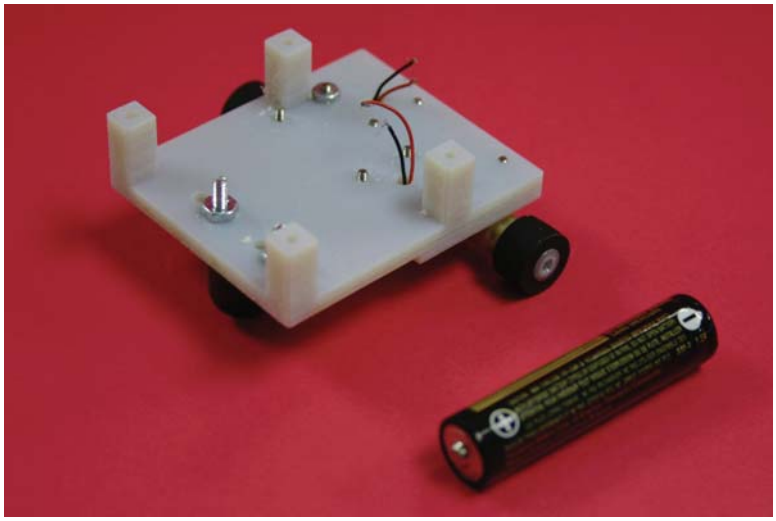

Figure 7.2: KEPLR'S chassis: isometric view. AAA battery shown for reference.

## 7.2 Specified Capabilities of the KEPLR-D platform

Returning to the research challenges outlined in Chapter 1, and adding other intentionally overlooked system requirements (such as obstacle avoidance), a list of system specifications was made.

- TDOA Distance-Only Sensing capabilities (functionally unchanged from Chapter 2)
- Odometry
- Extended Kalman Filter that fuses odometry and TDOA sensing
- Planetary gearmotor drivetrain and control
  - "configure and forget" motor control that has a sense of real world time for odometry
  - motor calibration/parameter estimation/system identification as in Chapter 4
  - "caster wheels" for free motion about ground contacts to reduce slip conditions as seen in Fig. 7.1
  - planetary gearmotors and chassis design as in as seen in Fig. 7.1 whose noise characteristics that do not affect performance of TDOA sensor.
- Randomized RHC/Model Predictive Control (postponed to KEPLR-M)
- Obstacle detection using IR proximity sensors

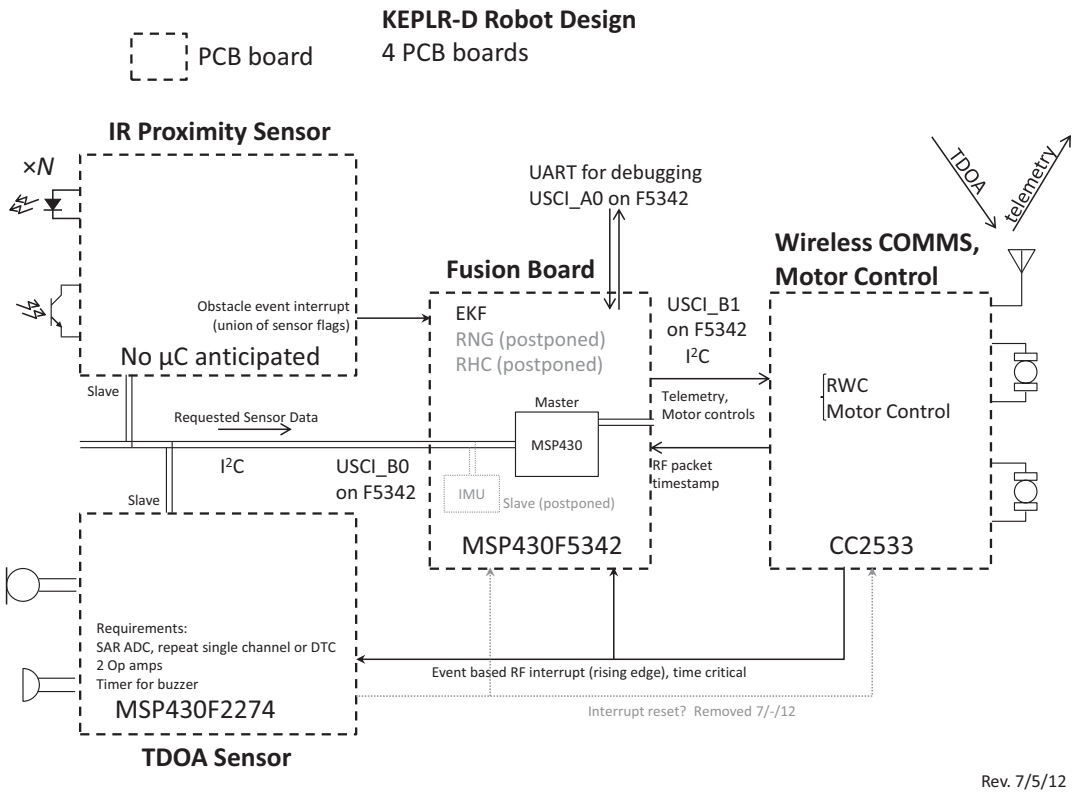


Figure 7.3: KEPLR'S design architecture at the board level. Communication buses are shown.

- Board-level Mixed-signal/Digital subsystem reconfigurability for future work when transitioning to KEPLR-M
- Wireless digital communications
- Randomized RHC/Model Predictive Control has been postponed given the scope of this project.

## 7.3 Proposed System Architecture

The primary limitation to the Walle bot was the insufficient number of unique timers on board. Three timer modules were on the EZ430. Timer A was used to

drive the motors, Timer B was dedicated for the TDOA sensor, while the Watchdog timer had to remain disabled when the wireless communications peripheral, the CC2500, was in use.

The use of odometry on an embedded system requires a sense of real-world time and an accurate clock with a resolution of about 0.01 sec to have system performance with an accuracy on the order of less than 0.1 sec for a runtime of less than an hour. To use a hardware timer already purposed for another task (with a different timing interval) would increase system overhead significantly. Essentially, two counters, a hardware timer with an overflow period of about 0.01 sec would trigger a low priority interrupt to divert program flow to increment a 32 bit global variable `clock`. Rare, timing critical events that results in 1-2 lost clock pulses would be insignificant.

The Motor/COMMs board is an upgraded version the base hardware from the TinyTERP from the Micro Robotics Laboratory. The CC2533 microcontroller on this board (an updated 8051 family microcontroller) will handle wireless communications, motor control, and contain the Real World Clock (RWC), not to be confused with Real Time Clock (RTC) which uses a 32,768 Hz crystal to track time with a 1 sec interval.

The Fusion board's primary task is to run the Extended Kalman Filter using the larger MSP430F5342. In effect, it fuses distance information with odometry for state estimation and uses proximity sensing to avoid obstacles. It will also issue motor commands and relay telemetry information to the motor/COMMs board. Given the limited computational resources of this board, it will not execute Receding

Horizon Control. For this functionality, larger, more powerful processors such as DSP chips with higher clock speeds, larger memory and hardware enabled fixed point arithmetic capability will be necessary.

### 7.3.1 Communication Channels

Having purely digital communication buses between devices is ideal for both the KEPLR-D and KEPLR-M platforms. The advantage of this architecture is its modularity. Mixed signal components can be easily swapped out for purely digital systems with no modifications to the platform, so long as all communications standards are met. The front end to sensors can still be completely analog such as in other low power architectures like the hibernet [50].

Consider a mixed-signal circuit such as the odometry circuit described in Chapter 6. The VLSI chip must have a means to interface directly with the other digital systems on the platform. Fig. 7.4 outlines how these mixed-signal chips would interact with other devices on the digital buses.

It is also possible to generate digital circuit layouts from verilog code or other hardware description languages in Cadence, streamlining the layout process. Verilog code describing the hardware for I2C is also readily available. This setup will require onboard ADCs and DACs directly on the chips we design, which could consume significant chip area and power consumption. Also, as a backup, all VLSI chips should have analog pin-outs for debugging if possible.

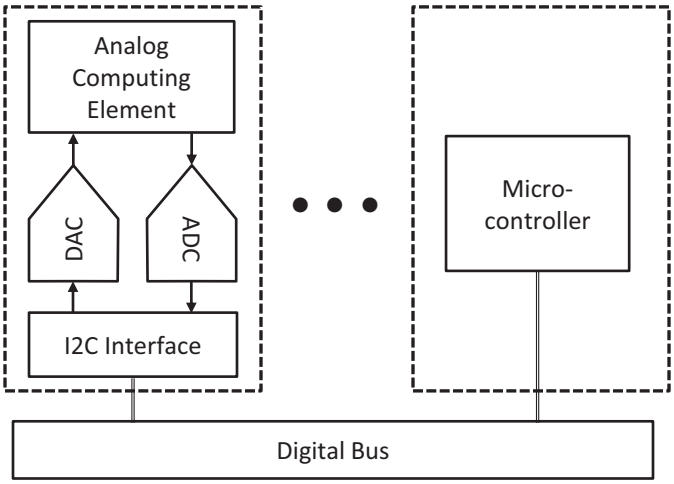


Figure 7.4: Chip-level communications and computation between devices on a single platform.

### 7.3.2 TDOA Sensor Accuracy and Clock Generation

A combination of the findings from Chapter 2 and Chapter 3 suggest reconfiguring the clocks available to the TDOA sensor board to reduce distance measurement uncertainty. The original Walle bots used a digitally controlled oscillator (DCO) calibrated at 8 MHz as the system clock. The DCO is internal to the MSP430 and is sensitive to changes in the power supply voltage (This is why all mentioned frequencies throughout this thesis are approximate). However, these sensitivities to voltage are deterministic in nature and do not affect the monotonicity of the sensor readings (in fact, the linearity of distance measurements to actual distance are preserved). Filter characteristics can change due to such deterministic clock period drift, especially if multiple bands are used close to each other. The addition of a crystal oscillator to KEPLR'S TDOA board should reduce the system clock's timing jitter, sensitivity to temperature and power supply voltage swing.

### 7.3.3 "Configure and forget" motor control

Previously, motor control configured specific motor velocities and had to be actively reset by the micrcontroller to update motor controls. The only implicit time reference the robot had was to halt in a `for` loop of known length (!), which meant halting all other program execution! Synchronizing motor control updates with a clock external to the CPU is essential for effective use of computing hardware. The need for a RWC thus becomes fully justified for a "configure and forget" motor controller. During each increment of the global variable `clock`, the controller can compare `clock` to reference values and adjust motor control as necessary. This also requires that motor control and the RWC are handled by the same microcontroller to reduce communication bandwidth between chips.

Chapter 4 suggests the existence of two means of describing control signals. Body velocities are platform/calibration independent, while motor velocities are closer tied to the operation of the motor controller. Motor calibration data $\mathbf{C}$ can approximate this mapping and be stored in the motor controller's flash memory to be able to execute both representations of motor controls.

Options to concatenate commands in a first in, first out (FIFO) queue to build piece-wise constant motor commands or overwrite options to replace previous commands (as in Receding Horizon Control) will be useful. Several other capabilities are necessary for backwards compatibility for legacy code. Discrete motion states (move forward, turn left, turn right, rotate left, rotate right, stop) in the original Leader Rendezvous Algorithm presented in Chapter 1 can be assigned a specific

body velocity. Commands of infinite time duration (with overwrite) will replace the original functionalities of a motor controller without a sense of time.

## 7.4 Extended Kalman Filtering

Extended Kalman Filters (EKF) are the *de-facto* standard technique for estimation and Simultaneous Localization and Mapping (SLAM) on robotics platforms [51]. This area of research is well-vetted and for this problem; the proposed solution is a direct application of Kalman Filter Theory. For those who are not familiar with Kalman Filtering, there are excellent simple applications of Kalman Filter Theory in the literature (single state, fusing two sensors) [52].

Chapter 5 breezed over a MATLAB simulation using a discrete time EKF [53]. The only non-obvious aspect of this EKF prototyped in MATLAB was the computation of the process noise covariance $Q$, which will be discussed in context of developing mixed-signal or hybrid Kalman Filters.

Given the design constraints of the platform, we have selected the MSP430F5342 for the Fusion board. First, it had the most memory of any MSP430 in a 7mm x 7mm package, the largest package we felt could fit on our sized boards. Second, it had a hardware multiplier that could multiply two 32 bit integers for fixed point arithmetic. Proper use of the hardware multiplier can improve fixed point multiplication speed by a factor of 20 [54]. Another known problem of EKF on small processors is the use of fixed-point arithmetic and resulting inaccuracies. We have opted to use MATLAB Coder, MATLAB Fixed-Point toolbox, and Embedded Coder to port the

MATLAB EKF code into robust C code [29].

### 7.4.1 Hybrid Extended Kalman Filter for Mixed-Signal Systems

The Hybrid Extended Kalman Filter [53] is a means to incorporate discrete system observations with a model of the system formulated as a system of nonlinear continuous time differential equations. It is a combination of concepts from the discrete-time EKF and the continuous-time EKF.

Given the initial system estimate $\hat{x}_0$, the mixed-signal odometry computer from Chapter 6 can model the system's evolution using $\dot{x} = f(x, u)$ as defined in (4.1). In addition, the system state covariance $P$ is tracked using (7.1), a special case of the system's Riccati equation.

$$\dot{P} = AP + PA^{'} + Q \tag{7.1}$$

Where $A = \frac{\partial f}{\partial x}$, and $Q$ is the process noise. $Q$ stems from uncorrelated, time independent motor noise with covariance $\Sigma_{motor}$ (a diagonal matrix) that propagates through the linear map obtained by motor calibration (Chapter 4), $\mathbf{C}$, and the operating point of the system with respect to the input $u$, $\mathbf{B}$ where $\mathbf{B} = \frac{\partial f}{\partial u}$. Also note that $P$ is symmetric, which will further reduce circuit implementation complexity. Therefore, $Q = \mathbf{BC}\Sigma_{motor}\mathbf{C}^{'}\mathbf{B}^{'}$.

Note that system Jacobians $\partial f$ such as $\frac{\partial f}{\partial x}$ or $\frac{\partial f}{\partial u}$ would in fact be matrix-valued continuous functions. Element wise, they would be to be broken up into components that can be implemented using translinear circuits such as how odometry, $f$ was

decomposed into computational blocks as in Fig. 5.3.

System updates from discrete measurements are equivalent to updates in the discrete-time EKF case. Parameters from the continuous-time system are sampled at measurement time for the discrete-time update. One valid concern regarding discrete time updates is the required matrix inversion for (near-optimal) Kalman gain calculation. Given the proposed EKF with a single beacon, the matrix inversion required for the discrete-time computations for the hybrid EKF is scalar for our case! Future work involving measurements from multiple beacons could initially assume measurement to be decoupled, reducing the complexity of the problem at the cost of losing valuable system state information. This progression in scaling of complexity for mixed-signal EKF would closely parallel the historical development of discrete-time EKF for SLAM on larger robotics platforms [51].

# Appendix A

# Statement of Contributions to Jointly Contributed Works Contained in the Thesis

The majority of the work discribed throughout the thesis was the work of Michael J. Kuhlman. This work was done under the advisement of Prof. Pamela Abshire.

- An accurate, miniature distance-only sensor as described Chapter 2
  - Michael Kuhlman: System-level/algorithmic design and analysis, error propagation model development (50% )
  - George Sineriz: Hardware specific implementation, algorithmic improvements, and experimentation (50%)
  - Reference publication: G. Sineriz, M. Kuhlman, and P. Abshire, "High Resolution Distance Sensing for Mini-Robots using Time Difference of Arrival," in IEEE International Symposium on Circuits and Systems, May 2012.
- In depth noise analysis in Chapter 3
  - No joint contributions
- Heading Estimation and Motor Calibration experiments in Chapter 4
  - Michael Kuhlman: System identification, odometry, error analysis (75%)
  - Yuchen Zhou: PWM motor controller development, feature extraction for the vision system data for system identification (25%)
- Mixed-signal architecture development for miniature robots in Chapter 5
  - Michael Kuhlman: Extended Kalman Filter simulation, Obstacle detection simulation, Odometry circuit design and comparative architecture analysis (65%)
  - Eduardo Arvelo: Receding Horizon Controller design and simulation (35 %)

  - Done under the advisement of Prof. Nuno C. Martins
  - Reference publication: M. Kuhlman, E. Arvelo, S. Lin, P. Abshire, and N. Martins, "Mixed-Signal Architecture of Randomized Receding Horizon Control for Miniature Robotics," to be published in IEEE International Symposium on Circuits and Systems, August 2012

- Mixed-signal odometry in Chapter 6
  - No joint contributions
  - Done under the advisement of Prof. Timothy K. Horiuchi.

- The development of KEPLR in Chapter 7 is really an overview of current and future work and the contributions of the following people are not covered in much detail in this thesis.
  - Michael Kuhlman, KEPLR design lead: System requirements and specifications, device selection, Architecture design
  - Andrew Sabelhaus: Motor/COMMs board design and layout
  - Matthew Phipps: Motor/COMMs board software development
  - Stacy Hand: upgraded TDOA board design and layout
  - David Shiao: KEPLR chassis design and motor/TDOA sensor interference analysis
  - Tsung-Hsueh Lee: IR obstacle detection sensor development

# Appendix B

## Main Function for Walle bot Follower

This code is the latest version of the `main` function used on the walle bot follower executing the Leader Rendezvous algorithm.

```
//this is the first test at integrating all source code for
   the walle bots
#include "robotConfig.h"
#include "mrfi.h"
#include "radios/family1/mrfi_spi.h"
#include "tool_box.c"
#include "microphone.c"
#include "buzzer.c"
#include "antBotMotionPlanners.c"
#include "motor.c"
#include "walle_bot_comms.c"


// *****************
// Global variables:
// *****************
int16_t samples[NSAMPLES]; //This needs to be dynamically
   alocated to save memory.
uint16_t frequency = 0;     //The frequency of interest.
   Found in received RF packet.
int16_t peak_center_y = 0, peak_center_x = 0;
int16_t peak_left_y = 0, peak_left_x = 0;
int16_t peak_right_y = 0, peak_right_x = 0;
float y1, y2, y3, t1, t2, t3, distance;
int16_t x1, x2, x3;

extern int FIR_filter(int);

volatile int16_t input_delay0 = 0;
volatile int16_t input_delay1 = 0;
volatile int16_t input_delay2 = 0;
volatile int16_t input_delay3 = 0;
volatile int16_t input_delay4 = 0;
```

```
volatile int16_t input_delay5 = 0;
volatile int16_t input_delay6 = 0;
volatile int16_t input_delay7 = 0;
volatile int16_t input_delay8 = 0;
volatile int16_t input_delay9 = 0;
volatile int16_t input_delay10 = 0;
volatile int16_t input_delay11 = 0;
volatile int16_t input_delay12 = 0;
volatile int16_t input_delay13 = 0;
volatile int16_t input_delay14 = 0;
volatile int16_t input_delay15 = 0;
volatile int16_t input_delay16 = 0;
volatile int16_t input_delay17 = 0;
volatile int16_t input_delay18 = 0;
volatile int16_t input_delay19 = 0;
volatile int16_t input_delay20 = 0;
volatile int16_t input_delay21 = 0;
volatile int16_t input_delay22 = 0;
volatile int16_t input_delay23 = 0;
volatile int16_t input_delay24 = 0;
volatile int16_t input_delay25 = 0;
volatile int16_t input_delay26 = 0;
volatile int16_t input_delay27 = 0;
volatile int16_t input_delay28 = 0;
volatile int16_t input_delay29 = 0;
volatile int16_t input_delay30 = 0;
volatile int16_t input_delay31 = 0;
volatile int16_t input_delay32 = 0;
volatile int16_t output = 0;
int16_t i;
//uint16_t samples[NSAMPLES]; //This needs to be dynamically
    alocated to save memory.
//volatile uint16_t frequency = 0;     //The frequency of
   interest. Found in received RF packet.

//uint16_t distanceOld =3000; uint16_t distanceNew = 4000;

uint16_t distanceOld = 3000;
uint16_t distanceNew = 4000;
float cosErrorOld = 0; float cosErrorNew = 0;
uint16_t headingError;

enum MOTION_STATE motionState = stop;
//enum MOTION_STATE motionStateOld = forward;
//float twist[2];
```

```
//distance[0] and distance_timestamp[0] are the most recent
    distance measurements

int main(void){

 unsigned long turnInterval;

  WDTCTL = WDTPW + WDTHOLD; // disable watchdog (WDT)
  BSP_Init();    // this modify the DCOCTL to 011 01100 and
     BCSCTL1 to 1 0 00 1101. Key thing is MCLK and SCLK is 8
     MHz
  WALLEBOT_SERIAL_PORT_Init();
  MRFI_Init();
  //The following is used to adjust the power level of the
     signal which is sent
  //and then the channel. This is mainly used for the rssi
     lab, 5_3.
  //The signal strength here should be at max, which I
     believe is about 0 dBm.
  mrfiSpiWriteReg(PATABLE, 0xFF);
  //Below is simply setting the channel.
  mrfiSpiWriteReg(CHANNR, 0x07);
  //IE2 |= UCA0RXIE; // Enable USCI_A0 RX interrupt
  MRFI_WakeUp();
  MRFI_RxOn();
  WALLEBOT_MOTOR_Init();
  WALLEBOT_MICROPHONE_Init();
   //motionState = potentialSourceMotionPlanner(1000, 1600,
      stop);
   iscas_motor_control(stop);

  P1DIR |= 0x03;            //Set P1.0 to output direction (
     Both LEDs)
  P1OUT |= 0x02;       //Green LED is turned on to signal
     initialization complete.


  //__bis_SR_register(GIE+LPM4_bits);      //Entering low
     power mode4 with interrupts. before addition of
     watchdog timer
  //__bis_SR_register(LPM0_bits + GIE);        // Enter LPM0
     w/ interrupt
  __bis_SR_register(GIE);

    while(1){
```

```
if(frequency != 0){ //if a freq value was transmited,
    this implies
//that a distance measurement is in progress and one
    must run the GA
distanceOld = distanceNew;

input_delay0 = 0;
input_delay1 = 0;
input_delay2 = 0;
input_delay3 = 0;
input_delay4 = 0;
input_delay5 = 0;
input_delay6 = 0;
input_delay7 = 0;
input_delay8 = 0;
input_delay9 = 0;
input_delay10 = 0;
input_delay11 = 0;
input_delay12 = 0;
input_delay13 = 0;
input_delay14 = 0;
input_delay15 = 0;
input_delay16 = 0;
input_delay17 = 0;
input_delay18 = 0;
input_delay19 = 0;
input_delay20 = 0;
input_delay21 = 0;
input_delay22 = 0;
input_delay23 = 0;
input_delay24 = 0;
input_delay25 = 0;
input_delay26 = 0;
input_delay27 = 0;
input_delay28 = 0;
input_delay29 = 0;
input_delay30 = 0;
input_delay31 = 0;
input_delay32 = 0;

for( i=0; i<255; i++)              //This filters the
    signal
{
     output = 0;
     input_delay0 = (signed short)samples[i];
```

```
        FIR_filter(samples[i]);
        samples[i] = output;
    }

/** This code finds the left/center/right peak coordinates *
   */

    peak_center_y = 0;
    peak_center_x = 0;

    for( i=0; i<255; i++)
                                                //Center Peak
    {     if (samples[i] > peak_center_y)
          {  peak_center_y = samples[i];
             peak_center_x = i;
          }
    }

    peak_left_y = 0;
    peak_left_x = 0;

    for( i= (peak_center_x - 15); i< (peak_center_x - 3); i
       ++)  //Left Peak
    {     if (samples[i] > peak_left_y)
          {  peak_left_y = samples[i];
             peak_left_x = i;
          }
    }

    peak_right_y = 0;
    peak_right_x = 0;

    for( i= (peak_center_x + 3); i< (peak_center_x + 15); i
       ++)  //Right Peak
    {     if (samples[i] > peak_right_y)
          {  peak_right_y = samples[i];
             peak_right_x = i;
          }
    }
//     for( i=0; i<255; i++)
//     {    output = samples[i];
//          itoc(output);                    //This prints the
   Filtered Signal to Serial Port
//          TXString("\n", (sizeof ("\n"))-1);
//          TXString("\r", (sizeof ("\r"))-1);
```

```
//    }

/*************** This part of the code computes the final
   peak **************/

        y1 = samples[peak_left_x];          //Peak Computation
        y2 = samples[peak_center_x];
        y3 = samples[peak_right_x];

        x1 = peak_left_x;
        x2 = peak_center_x;
        x3 = peak_right_x;

        t1 = (float)(1.0*y1*(1.0*x2*x2-1.0*x3*x3)-1.0*y2
           *(1.0*x1*x1-1.0*x3*x3)+ \
          1.0*y3*(1.0*x1*x1-1.0*x2*x2))/(2.0*y1*(x2-x3)-2.0*
             y2*(x1-x3)+2.0*y3*(x1-x2));

        distance = 0.38276 * t1 - 19.547;
        //distance = TDOA_CAL_SLOPE * t1 -
           TDOA_CAL_DIST_OFFSET;
        //Distance can become negative if the pulse is not
           detected (i.e. it's beyond 70 cm)
        //The motion planning algorithms are more stable if
           the sensor "saturates" at 70cm
        if(distance < 0){
          distance = 71;
        }
        distanceNew = (uint16_t)(10 * distance);




/** END OF DISTANCE MEASUREMENT **/

        itoc((int)distanceNew);                    //This
           prints the Filtered Signal to Serial Port
        TXString("\n", (sizeof ("\n"))-1);
        TXString("\r", (sizeof ("\r"))-1);

      //distanceOld = 300;
      //distanceNew = 400;
```

```
if(distanceNew < D_MIN){ //D_MIN
  iscas_motor_control(stop);
  mrfiSpiWriteReg(CHANNR, 0x05);
  headingError = (uint16_t)(cosErrorNew*1000);
  mrfiPacket_t packet;
  packet.frame[0] = 29;
  packet.frame[1] = RID;
  //packet.frame[1] = motionState;
  packet.frame[2] = (uint8_t)(distanceNew >> 8);    //8
     MSB
  packet.frame[3] = (uint8_t)(distanceNew - (distanceNew
      & ~0xFF)); //8 LSB
  packet.frame[4] = (uint8_t)(9999 >> 8);    //8 MSB
  packet.frame[5] = (uint8_t)(9999 - (9999 & ~0xFF)); //
     8 LSB
  MRFI_Transmit(&packet, MRFI_TX_TYPE_FORCED);  //And
     this actually transmits the packet.
  MRFI_RxIdle();
  MRFI_RxOn();
  mrfiSpiWriteReg(CHANNR, 0x07);

  frequency = 0;
  P1OUT &= ~0x02;        //Green LED is turned off
  continue;
}

P1OUT ^= 0x01;  //toggle the red LED
cosErrorNew = headingEstimator(distanceNew, distanceOld)
    ;
//cosErrorNew = 1; //for debugging purposes
P1OUT ^= 0x01;  //toggle the red LED



if(cosErrorNew > cosErrorOld){//if robot is turning the
    wrong way, change turning direction
  if(motionState == turnL){
    motionState = turnR;
  }
  else {
    motionState = turnL;
  }
}
P1OUT |= 0x02;        //Green LED is turned on to signal
    movement.
```

```
iscas_motor_control(motionState);

itoc((int32_t)(motionState));
TXString("\n", (sizeof ("\n"))-1);
TXString("\r", (sizeof ("\r"))-1);


turnInterval = (unsigned long)(cosErrorNew*T_TURN); //
   error e[0,2] when angle error e[0,pi] 1500000 12222

//itoc((int)error);
  //   TXString("\n", (sizeof ("\n"))-1);
     //TXString("\r", (sizeof ("\r"))-1);

for(unsigned long i = turnInterval; i!=0; i--){ //
   1000000 for ~1 sec interval, increase for debugging
   1100000
  __no_operation();
}

iscas_motor_control(forward);
//for(unsigned long i = 0; i < 1500000; i++){ //i <
   1500000 for sk = 250 cm //1000000 for ~1 sec interval
   , increase for debugging
for(unsigned long i = 0; i < 750000; i++){ //sk = 15cm?
  __no_operation();
}

iscas_motor_control(stop);
P1OUT ^= 0x01;  //toggle the red LED


//itoc((int32_t)(error*100));
//TXString("\n", (sizeof ("\n"))-1);
//TXString("\r", (sizeof ("\r"))-1);

//send data to datalogger
mrfiSpiWriteReg(CHANNR, 0x05);
headingError = (uint16_t)(cosErrorNew*1000);
mrfiPacket_t packet;
packet.frame[0] = 29;
packet.frame[1] = RID;
//packet.frame[1] = motionState;
packet.frame[2] = (uint8_t)(distanceNew >> 8);    //8 MSB
```

```
    packet.frame[3] = (uint8_t)(distanceNew - (distanceNew &
        ~0xFF)); //8 LSB
    packet.frame[4] = (uint8_t)(headingError >> 8);    //8
       MSB
    packet.frame[5] = (uint8_t)(headingError - (headingError
        & ~0xFF)); //8 LSB
    MRFI_Transmit(&packet, MRFI_TX_TYPE_FORCED);  //And this
        actually transmits the packet.
    MRFI_RxIdle();
    MRFI_RxOn();
    mrfiSpiWriteReg(CHANNR, 0x07);

    itoc((int32_t)(cosErrorNew*1000));
    TXString("\n", (sizeof ("\n"))-1);
    TXString("\r", (sizeof ("\r"))-1);

    frequency = 0;    //clear the fequency flag



    }//end if
  }//end while
}//end main



void MRFI_RxCompleteISR(){
  //iscas_motor_control(stop);

//disable interrupts?
//TXString("entered, ", (sizeof ("entered, "))-1);
  //grab the packet
 char frames[] = {"____________________"};
  uint8_t i;
  mrfiPacket_t packet;
  //Receiving the wifi packet:
  MRFI_Receive(&packet);

  //turn on microphone & sample data
  ADC10CTL0 &= ~ENC;
  while(ADC10CTL1 & BUSY);  //Wait if ADC10 core is active.
  ADC10SA = (unsigned short)samples;          //Data buffer
      start.
  ADC10CTL0 |= ENC + ADC10SC; //Sampling and conversion
      start.
```

```
  __bis_SR_register(CPUOFF + GIE);  //LPM0, ADC10_ISR will
     force exit

  for (i=0;i<29;i++)
  {
    frames[i]=packet.frame[i];
  }

 //parse packet
  frequency = extractFrequencyfromPacket(&(frames[0])); //
     5/7 tested: correct hit (did not test on empty packet)
  //iscas_motor_control(motionState);
}


// ******************************
// ADC10 interrupt service routine
// ******************************
#pragma vector = ADC10_VECTOR
__interrupt void ADC10_ISR(void){
  __bic_SR_register_on_exit(CPUOFF);  //Clear CPUOFF bit
     from 0 (SR)
}
```

# Bibliography


[1] K. Hatazaki, M. Konyo, K. Isaki, S. Tadokoro, and F. Takemura, "Active Scope Camera for Urban Search and Rescue," in *IEEE International Conference on Intelligent Robots and Systems, 2007.* IEEE, 2007, pp. 2596–2602.

[2] T. Datta, P. Abshire, and J. Turner, "Towards a Legged Chip," in *IEEE International Symposium on Circuits and Systems (ISCAS), 2011.* IEEE, 2011, pp. 2501–2504.

[3] A. Flynn, "Gnat Robots (and How They Will Change Robotics)," *MIT Artificial Intelligence Laboratory*, 1987.

[4] C. Perkins, L. Lei, M. Kuhlman, T. Lee, G. Gateau, S. Bergbreiter, and P. Abshire, "Distance Sensing for Mini-Robots: RSSI vs. TDOA," in *2011 IEEE International Symposium on Circuits and Systems (ISCAS).* IEEE, 2011, pp. 1984–1987.

[5] G. Caprari and R. Siegwart, "Mobile Micro-Robots Ready to Use: Alice," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems,.* IEEE, 2005, pp. 3295–3300.

[6] A. Savvides, C. Han, and M. Strivastava, "Dynamic Fine-Grained Localization in Ad-hoc Networks of Sensors," in *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking.* ACM, 2001, pp. 166–179.

[7] J. Chen, L. Yip, J. Elson, H. Wang, D. Maniezzo, R. Hudson, K. Yao, and D. Estrin, "Coherent acoustic array processing and localization on wireless sensor networks," *Proceedings of the IEEE*, vol. 91, no. 8, pp. 1154–1162, 2003.

[8] S. Hong, B. Kim, and D. Eom, "Localization algorithm in wireless sensor networks with network mobility," *Consumer Electronics, IEEE Transactions on*, vol. 55, no. 4, pp. 1921–1928, 2009.

[9] N. Priyantha, A. Chakraborty, and H. Balakrishnan, "The Cricket Location-Support System," in *Proceedings of the 6th annual international conference on Mobile computing and networking.* ACM, 2000, pp. 32–43.

[10] L. Navarro-Serment, R. Grabowski, C. Paredis, and P. Khosla, "Millibots," *Robotics & Automation Magazine, IEEE*, vol. 9, no. 4, pp. 31–40, 2002.

[11] K. Banks, "The Goertzel Algorithm," *Embedded Systems Programming*, vol. 15, no. 9, pp. 34–42, 2002.

[12] T. Watteyne. (2009) eZWSN: Experimenting with Wireless Sensor Networks using the eZ430-RF2500. [Online]. Available: http://cnx.org/content/col10684/1.10/

[13] —. (2012) Introduction to SimpliciTI. [Online]. Available: http://www.ti.com/lit/ml/swru130b/swru130b.pdf

[14] I. Alan Dwight Hulsebus, "Cone reflector/coupler speaker system and method," Jul. 10 2001, uS Patent 6,257,365.

[15] —, *The MSP430x2xx Family User's Guide*. Texas Instruments, 2008.

[16] M. Buccini, *Using Direct Data Transfer to Maximize Data Acquisition Throughput*. Texas Instruments, 2002.

[17] IEEE Acoustics and Speech and Signal Processing Society. Digital Signal Processing Committee, *Programs for Digital Signal Processing*. IEEE, 1979.

[18] R. Losada, *Practical FIR Filter Design in MATLAB*. The MathWorks, Inc., 2004.

[19] K. Venkat, *Efficient MSP430 Code Synthesis for an FIR Filter*. Texas Instruments, 2007.

[20] B. Barshan and B. Ayrulu, "Performance Comparison of Four Time-of-Flight Estimation Methods for Sonar Signals," *Electronics Letters*, vol. 34, no. 16, pp. 1616–1617, 1998.

[21] M. Heath, *Scientific Computing*. McGraw-Hill, 1997.

[22] —, *Clock Jitter and Measurement*. SiTime Corporation, 2009.

[23] ——, *Jitter Analysis: A Brief Guide to Jitter*. Tektronix, 2009.

[24] N. Soo, *Jitter Measurement Techniques*. Pericom Application Brief, 2000.

[25] T. Xia and H. Zheng, "Timing Jitter Characterization for Mixed-Signal Production Test Using the Interpolation Algorithm," *IEEE Transactions on Industrial Electronics*, vol. 54, no. 2, pp. 1014–1023, 2007.

[26] C. Bishop, *Pattern Recognition and Machine Learning*. Springer New York, 2006, vol. 4.

[27] P. Gray and R. Meyer, *Analysis and Design of Analog Integrated Circuits*. John Wiley & Sons, Inc., 1990.

[28] C. Presse and M. Gautier, "New Criteria of Exciting Trajectories for Robot Identification," in *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on*. IEEE, 1993, pp. 907–912.

[29] MATLAB, *version 7.14.0.739 (R2012a)*. Natick, Massachusetts: The MathWorks Inc., 2012.

[30] P. Spevak and P. Forstner, *MSP430 32-kHz Crystal Oscillators*. Texas Instruments, 2009.

[31] E. Menegatti, A. Zanella, S. Zilli, F. Zorzi, and E. Pagello, "Range-Only SLAM with a Mobile Robot and a Wireless Sensor Networks," in *IEEE International Conference on Robotics and Automation, 2009.* IEEE, 2009, pp. 8–14.

[32] C. Wang, "Location Estimation and Uncertainty Analysis for Mobile Robots," in *Robotics and Automation, 1988. Proceedings., 1988 IEEE International Conference on.* IEEE, 1988, pp. 1231–1235.

[33] G. Antonelli and S. Chiaverini, "Linear Estimation of the Odometric Parameters for Differential-Drive Mobile Robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2006.* IEEE, 2006, pp. 3287–3292.

[34] K. Chong and L. Kleeman, "Accurate Odometry and Error Modelling for a Mobile Robot," in *IEEE International Conference on Robotics and Automation, 1997.*, vol. 4. IEEE, 1997, pp. 2783–2788.

[35] B. Merritt and R. Murugavel, *PWM DC Motor Control Using Timer A of the MSP430.* Texas Instruments, 2000.

[36] G. Antonelli, S. Chiaverini, and G. Fusco, "An Odometry Calibration Method for Mobile Robots Based on the Least-Squares Technique," in *American Control Conference, 2003. Proceedings of the 2003*, vol. 4. IEEE, 2004, pp. 3429–3434.

[37] K. Tossell, A. Hammond, E. Arvelo, and N. Martins, "Visual Mini-Robot Identification, Tracking and Control," 2010.

[38] D. Mayne, J. Rawlings, and C. Rao, "Constrained Model Predictive Control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

[39] A. Alessio and A. Bemporad, "A Survey on Explicit Model Predictive Control," in *Nonlinear Model Predictive Control*, L. Magni, D. Raimondo, and F. Allgöwer, Eds. Springer Berlin / Heidelberg, 2009, pp. 345–369.

[40] M. Zeilinger, C. Jones, and M. Morari, "Real-Time Suboptimal Model Predictive Control Using a Combination of Explicit MPC and Online Optimization," *IEEE Transactions on Automatic Control*, vol. 56, no. 7, pp. 1524–1534, 2011.

[41] H. Tanner and J. Piovesan, "Randomized Receding Horizon Navigation," *IEEE Transactions on Automatic Control*, vol. 55, no. 11, pp. 2640–2644, 2010.

[42] P. Scokaert, D. Mayne, and J. Rawlings, "Suboptimal Model Predictive Control (Feasibility Implies Stability)," *IEEE Transactions on Automatic Control*, vol. 44, no. 3, pp. 648–654, 1999.

[43] G. Sineriz, M. Kuhlman, and P. Abshire, "High Resolution Distance Sensing for Mini-Robots using Time Difference of Arrival," in *IEEE International Symposium on Circuits and Systems*, May 2012.

[44] P. Xu, Y. Wong, T. Horiuchi, and P. Abshire, "Compact Floating-Gate True Random Number Generator," *Electronics Letters*, vol. 42, no. 23, pp. 1346–1347, 2006.

[45] R. Fried and C. Enz, "MOST Implementation of Gilbert sin(x) Shaper," *Electronics Letters*, vol. 32, no. 22, pp. 2073–2074, 1996.

[46] B. Gilbert, "A Precise Four-Quadrant Multiplier with Subnanosecond Response," *IEEE Journal of Solid-State Circuits*, vol. 3, no. 4, pp. 365–373, 1968.

[47] B. Hosticka, "Performance Comparison of Analog and Digital Circuits," *Proceedings of the IEEE*, vol. 73, no. 1, pp. 25–29, 1985.

[48] B. Gilbert, "Circuits for the Precise Synthesis of the Sine Function," *Electronics Letters*, vol. 13, no. 17, pp. 506–508, 1977.

[49] F. Diotalevi and M. Valle, "An Analog CMOS Four Quadrant Current-Mode Multiplier for Low Power Artificial Neural Networks Implementation," in *15th European Conference on Circuit Theory and Design, ECCTD*, vol. 1, 2001.

[50] B. Rumberg, D. Graham, and V. Kulathumani, "Hibernets: Energy-Efficient Sensor Networks Using Analog Signal Processing," in *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*. ACM, 2010, pp. 129–139.

[51] H. Durrant-Whyte and T. Bailey, "Simultaneous localisation and mapping (SLAM): Part i the essential algorithms," *Robotics and Automation Magazine*, vol. 13, no. 99, p. 80, 2006.

[52] L. Tarassenko, L. Mason, and N. Townsend, "Multi-Sensor Fusion for Robust Computation of Breathing Rate," *Electronics Letters*, vol. 38, no. 22, pp. 1314–1316, 2002.

[53] D. Simon, *Optimal State Estimation: Kalman, $H_\infty$ and Nonlinear Approaches*. John Wiley and Sons, 2006.

[54] L. Bierl, *The MSP430 Hardware Multiplier: Function and Applciations*. Texas Instruments, 1999.