A Standalone Capacitively Coupled Occupancy Sensor

by

William H. Thompson

S.B., Massachusetts Institute of Technology (2010)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2012
[FEBRUARY 2013]

Author .....
Department of Electrical Engineering and Computer Science
September 30, 2012

Certified by .....
Dr. Steven B. Leeb
Professor of E.E.C.S. & M.E., MacVicar Faculty Fellow
Thesis Supervisor

Certified by .....
Al-Thaddeus Avestruz
Doctoral Candidate
Thesis Supervisor

Accepted by .....
Prof. Dennis M. Freeman
Chairman, Master of Engineering Thesis Committee

A Standalone Capacitively Coupled Occupancy Sensor

by

William H. Thompson

Submitted to the Department of Electrical Engineering and Computer Science
on September 30, 2012, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

This thesis presents the design and implementation of a standalone, capacitively coupled, occupancy sensor. Unlike previous iterations, the new sensor is decoupled from the fluorescent lamp. A well controlled, high voltage amplifier and autotransformer are designed as a replacement source and operate over a wide range of output frequencies and amplitudes. The sensor electronics are implemented with active sensing electrodes and resistive and capacitive feedback modes are evaluated for performance. Optimal front end feedback, electrode spacing, and signal source amplitude are explored. The sensor achieves a detection range of 11 feet for occupancy detection, with capacitive measurements down to the attofarad level. The application of the sensor as a material detector, using a linear mixing algorithm, is also found to be feasible.

Thesis Supervisor: Dr. Steven B. Leeb
Title: Professor of E.E.C.S. & M.E., MacVicar Faculty Fellow

Thesis Supervisor: Al-Thaddeus Avestruz
Title: Doctoral Candidate

Acknowledgements

I would like to thank everyone that made this thesis possible. I am extremely grateful to my thesis supervisors, Steve Leeb and Al-Thaddeus Avestruz, for their constant guidance and support.

My friends and colleagues in the lab and at MIT have also provided incredible support, encouragement, and distraction. They include, but are not limited to, John Cooley, Steven Herbst, Shahriar Khushrushahi, Jim Paris, John Donnal, Arthur Chang, Lizi George, and Amanda Levesque. I am especially thankful to Tina Drake for her endless support.

Lastly, I like to thank my parents for pushing me, believing in me, and always supporting me.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation and Prior Work

The standalone occupancy sensor aims to challenge existing motion and occupancy detection technology and present a superior alternative. Passive infrared (PIR) sensors are commonplace devices and have achieved widespread use in various settings, e.g. offices, laboratories, and even restrooms. This is largely due to their simplicity, which allows for low cost and high volume access to the sensors. However, a byproduct of the PIR sensor's simplicity is its limits of use. As described in reference [5], a PIR sensor is effectively limited to motion detection, as opposed to pure occupancy detection, by low frequency noise and drift due to electronics and changes in the background infrared radiation.

One of the goals of the capacitively coupled occupancy sensor is to detect true occupancy, and remove any dependency on movement. Implementing large scale capacitive sensing in this fashion has been proven feasible in previous work [1, 5]. Therefore, pushing the limits of effective occupancy detection is the next step in the development of the standalone sensor. Doing so is heavily reliant on mitigating the same noise and drift problems that plague existing sensors.

Previous applications of the capacitively coupled occupancy sensor relied on fluorescent lamps as a signal source. The primary motivation for this thesis is to eliminate the need for a fluorescent bulb and to create an independent, standalone sensor. Despite the prevalence of fluorescent lamps, their utility as a signal source for the occupancy sensor is limited. As a preinstalled system, parameters of the signal source and sensor configuration that are useful to the application of the occupancy detector, as a retrofit implementation, are not necessarily controlled. This limits some applications of the sensor by reducing the degrees of freedom in the system.

An advantage of the standalone system is the freedom to engineer the system with full control over the operating parameters. Without a fixed and given signal source, like the fluorescent lamp, the standalone system has the freedom to operate over a range of source frequencies and amplitudes.

Furthermore, by removing the requirement for the fluorescent lamp as an installation framework also removes the limits of the sensor electrode configuration. Thus, the standalone sensor is better suited to be adapted for use in a wider variety of applications.

1.2 Applications

1.2.1 Occupancy Detection

As with previous designs, one of the primary applications of the standalone sensor is occupancy detection based purely on presence. The benefits of eliminating dependencies on motion are numerous and impact the possible applications of the device. The simplest application is a basic replacement of the PIR sensor. In an office or laboratory setting, where motion sensors commonly control lighting, the benefit of pure presence detection is the reduction of false negatives, where the building lights may turn off while occupants are still present. Breaking this paradigm is the advantage of pure occupancy detection.

The standalone sensor can also be used to detect motion and occupancy in high security areas. By not having a lamp as a signal source, the sensor can operate in an ad hoc manner. The standalone sensor may be used in a conventional high security, limited access area. Along with fixed or permanent installations, the sensor may also be set up in the field or on the go, as necessary, because no lamp infrastructure is required.

1.2.2 Object Detection

An alternative and innovative application of the standalone sensor is object detection and differentiation. One method of differentiating between materials is based on the principle that the frequency response of the permittivity and conductivity of a particular material is different. The design for the standalone sensor allows for this because the signal source allows for simultaneous multi-frequency operation.

Once again, this application may be employed in security applications. In a high security settings such as airports, weapons detection and threat assessment, among other preventative measures are of high priority. The ability to sense a potentially harmful materials is invaluable in many situations.

1.3 Thesis Overview

This thesis discusses the design and implementation of a standalone occupancy sensor. Emphasis is placed on increasing the range and sensitivity over previous implementations of the occupancy sensor. In doing so, a new approach to the front end sensing circuitry is evaluated and sources of noise and error are minimized.

The design and implementation of the high voltage signal source, that replaces the fluorescent, is discussed in Chapter 3. Analysis and design of the front end sensing amplifier and signal conditioning circuitry is presented in Chapter 4. The overall system performance and results of the system tests performed are discussed in Chapter 5. Finally, the thesis is concluded in Chapter 7 with a discussion of the results, system limitations, and future work. Appendices follow with relevant schematics, board layouts, and code.

Chapter 2

System Overview

A lumped element system model for the standalone sensor system, shown in Figure 2-1, may be applied to simplify and visualize the measurements of interest. Furthermore, the model provides useful insight into the system to aid and drive design.

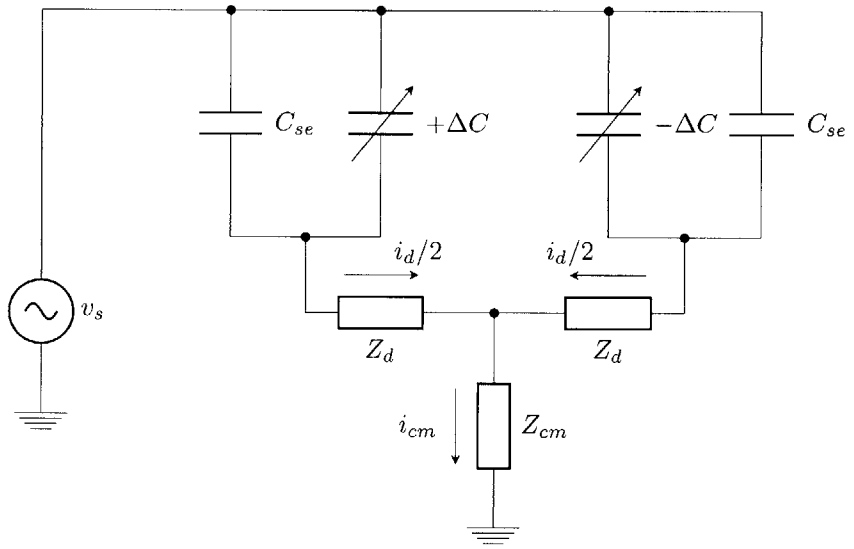


Figure 2-1: The lumped element model for the capacitively coupled occupency sensor.

The model depicts the dominant lumped capacitances of the system, comprised by the source to receive electrode capacitance, C_{se}, in parallel with the variable capacitance from the source to the electrodes, $+\Delta C$ and $-\Delta C$, which are dependent on the presence of an occupant. These variable capacitances sensed by the system are significantly smaller than the fixed source to receive electrode capacitance. Previous work modeling the system provides approximate capacitances for the model which are used in the design of the standalone sensor [1].

The measurement of interest is the current into each of the front end op-amps, whose inputs are modeled by a small impedance, Z_d. The impedances through the front end amplifier are small

because in feedback they appear as nearly a short circuit. A common-mode feedback path drives each amplifier to a known potential, which has some impedance path to the source ground through Z_{cm}. Because the common mode feedback drives the amplifiers and attempts to attenuate the common mode signal, its path to the source ground is small.

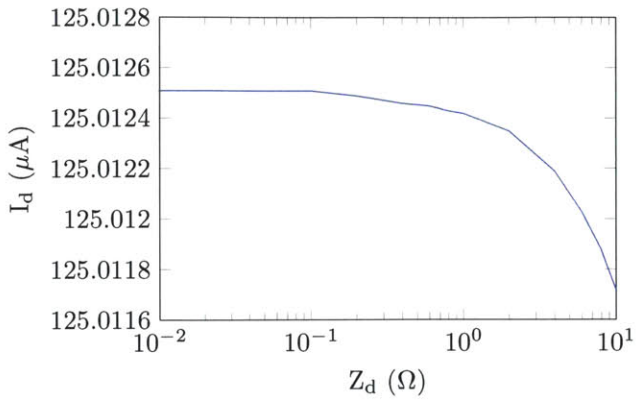


Figure 2-2: Modeled sensed current, I_d, as a function of Z_d. $V_s = 200\,V$, $Z_{cm} = 1\,m\Omega$, $f = 10\,kHz$

As shown in Figure 2-2, for a fixed ΔC, the sensed current increases for smaller Z_d and Z_{cm}. This suggests that large common mode rejection is desirable in order to maximize the sensed current. As Z_d and Z_{cm} go to zero and the loop gain of the front end becomes large, the sensed current to one electrode is

$$i_{sense} = v_s \omega_c \left(\frac{C_{se} \Delta C}{C_{se} + \Delta C} \right). \tag{2.1}$$

This holds for all cases where the occupant is relatively far from the sensor, and $\Delta C \ll C_{se}$. In the end, this model allows for a calculation of the minimum detectable change in capacitance for the standalone system

$$\begin{aligned} \Delta C_{\text{min}} &= \frac{i_{\text{sense}}}{\omega_c v_s} \\ &= \frac{1}{\omega_c v_s} \cdot \frac{v_{\text{min}}}{G_{\text{sys}} Z_f}, \end{aligned} \tag{2.2}$$

where $i_{\text{sense}} = i_d$, v_{min} is the minimum detectable voltage at the input to the analog to digital converter, G_{sys} is the overall system gain, and Z_f is the front end transimpedance gain.

A block diagram depicting a standalone system is shown in Figure 2-3. The full system consists of the following elements:

- A high voltage signal source to drive the source electrodes
- A front end amplifier on the receive electrodes
- A signal conditioning chain and microcontroller interface on the sensor board

The high voltage source drives an electric field. The targets present in the sensing space perturb

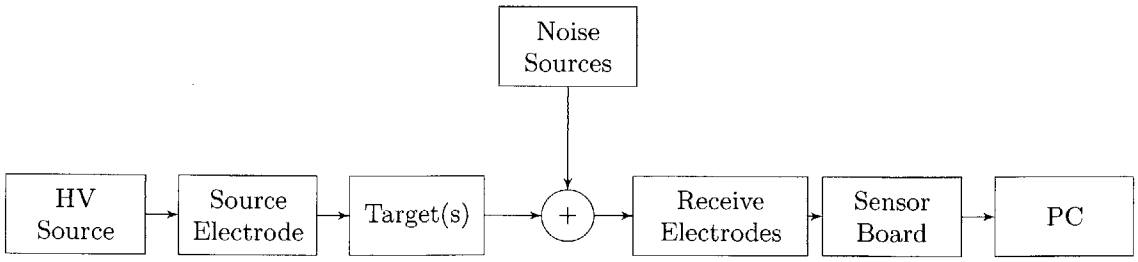


Figure 2-3: Diagram of the generalized standalone occupancy sensor.

the field and affect the coupling to the receive electrodes. The front end amplifiers, located on the receive electrodes, transduce the currents at their inputs, according to the local feedback network. Additional gain is applied and the sensed signal is transmitted via a coaxial cable to the signal conditioning electronics on the sensor board. The received signal is further amplified as necessary and then synchronously detected using a full bridge. A set of RC low pass filters remove the ac component and pass it to an instrumentation amplifier, which level shifts the signal to the midscale value of the supply range of the ADC. Lastly, the digital representation of the sensed capacitance change is passed to a computer for data collection and analysis.

2.1 System Requirements

The primary goal for the standalone sensor is to lower the minimum detectable capacitance change. Previous sensors achieved capacitance measurements as low as 10 fF [5]. Pushing the minimum detectable capacitance even lower is desirable in order to increase the sensitivity of the system. As noted from Equation (2.2), this can be done in a number of ways, including increasing the source amplitude and operating frequency. A conservative calculation places the minimum capacitance detectable by the standalone sensor at

$$\Delta C_{\min} = \frac{100\,\mu\text{V}}{(2\pi 10\,\text{kHz})(200\,\text{V})(10)(10\,\text{k}\Omega)} = 80\,\text{aF}. \tag{2.3}$$

Though exceptionally small, it is not uncommon for capacitance measurements to be of such a small resolution. Impedance spectroscopy, electrochemical impedance measurements, and ac current sensing atomic force microscopy are just a few applications in which attofarad measurements have been performed [7, 8, 9, 10, 11]. An important difference is the physical scale of the measurement space. The standalone system aims to make these measurements over a large volume in comparison to those described in the references.

In order to measure such small capacitance changes, the sensor must maintain a suitably low noise floor and be able to reject unwanted, erroneous signals. Furthermore, errors, in the form of offset and drift, introduced by the sensor electronics must also be kept as low as possible. One

method of doing so is to null the differential measurement made by the electrodes. This means that the receive electrodes couple to the source equally; thus, in the absence of an occupant, and any detection, the differential output of the sensor is zeroed.

2.2 Fundamental Limits of Detection

2.2.1 Occupancy Detection

Occupancy detection refers to the sensing of an object purely based on proximity, with no dependence on motion, heat, or sound. For detecting an average person, about 5 feet 10 inches tall and 180 pounds, the detection range of the system is defined to be the farthest distance away from the sensor that yields a correct positive result, i.e. detecting an occupant when an occupant is actually present. Naturally, the larger the detection range, the larger the volume over which an object may be sensed.

There are two paradigms: near and far detection. Near detection, or detection localized about the sensor when distance is not the goal, does not suffer from low sensed signal amplitudes. Rather the more important factor, when purposefully within the detection radius, is the system sensitivity and minimizing the smallest detectable occupant. Far detection, as noted, attempts to push the range of the system. In either case, the signal to noise ratio (SNR) is the key metric to be maximized. Near the sensor, the dynamic range of the system must be sufficient to handle a wide swing response. Far from the sensor, the noise floor of the system must be kept low in order to decrease the minimum detectable signal.

The proximity sensing, described further in Chapter 4, method uses the magnitude of the measured in phase and quadrature signals from the synchronous detector. For this application, magnitude errors are the greatest concern and limit the resolution.

Noise, Drift, Offset and Dynamic Range

As an object moves farther and farther away from the sensor, the point to point change in capacitance decreases along with the sensed input current. When the input current drops below the noise floor of the system, the object is no longer detectable. This noise floor is defined as the root mean square (RMS) noise of the entire system. The system noise is assumed to be represented by a Gaussian distribution. This means that the three sigma rule may be applied to detection decisions. The rule states that 99.7% of the values of a distribution are within three standard deviations, or sigmas, of the mean. In the context of the sensor, this suggests that the level of confidence with which a measurement may be considered a true positive detection can be determined by its value above the noise floor. Thus, lowering the noise contributions of the electronics and mitigating noise contributions from the environment are a must in order to increase the detection range.

Furthermore, the negligible drift enables presence based detection. System drift, regardless of the source, hinders the detection capabilities of the system by corrupting the relationship between past and current measured values. For instance, a static object within range of the sensor should result in a constant measured response. This may not always be the case with drift present and the system may begin to break down.

Measurement offsets may also negatively affect system performance by reducing dynamic range. This is primarily a concern when close to the sensor, where the signal amplitude is largest. A reduction in the dynamic range translates into a reduction of sensitivity because the sensed signals may be clipped by the limited range.

Electric Field Drop Off

Physical limits also apply to the capacitively coupled occupancy sensor. The farther away an object is from the signal source, the smaller the disturbance in the electric field measured by the system. Occupancy detection is limited by strength of the electric field and the rate at which it weakens as a function of distance.

The drop off of the electric field for different electrode configurations is currently an active area of research. The parameters that affect the field include, but are not limited to, the signal source amplitude, the electrode configuration, and the size of the electrodes. The effects of these parameters for some example configurations are investigated in Chapter 5.

2.2.2 Object Detection

Object detection has similar concerns to occupancy detection. It differs slightly because it aims to differentiate between materials. However, this sensing application is less concerned with the distance of the detection field. Material differentiation is to be performed well within the range of the system, since the granularity and sensitivity of the sensor is much more important.

Another major concern for this application is reproducibility. Based on the method to be explained in Chapter 6, the object detection algorithm uses the amplitudes of the in phase and quadrature measurements at two different frequencies. The algorithm uses calibration trials as references for object detection.

2.3 System Dynamics

Another consideration in the design of the source and sensor electronics is the placement of the closed loop poles and zeros. Close proximity between the operating frequency and the poles and zeros leads to undesired errors in amplitude and phase. In general, poles and zeros are not well

controlled both initially, as well as over time and temperature. Placing these singularities away from signal frequencies minimizes their time-varying effects.

The dynamic characteristics of each stage are specified, in large part, by passive components. These resistors and capacitors are prone to drift and mismatch; this, in turn, translates to shifts in the pole and zero frequencies. Errors arise in both magnitude and phase and their severity depends on the proximity of the system poles and zeros and operating frequency. A simple RC low pass filter, such as those found in the signal conditioning circuitry, illustrates the potential effects of these errors well. The passive components define the passband and stopband regions, denoted in the bode plot in Figure 2-4.

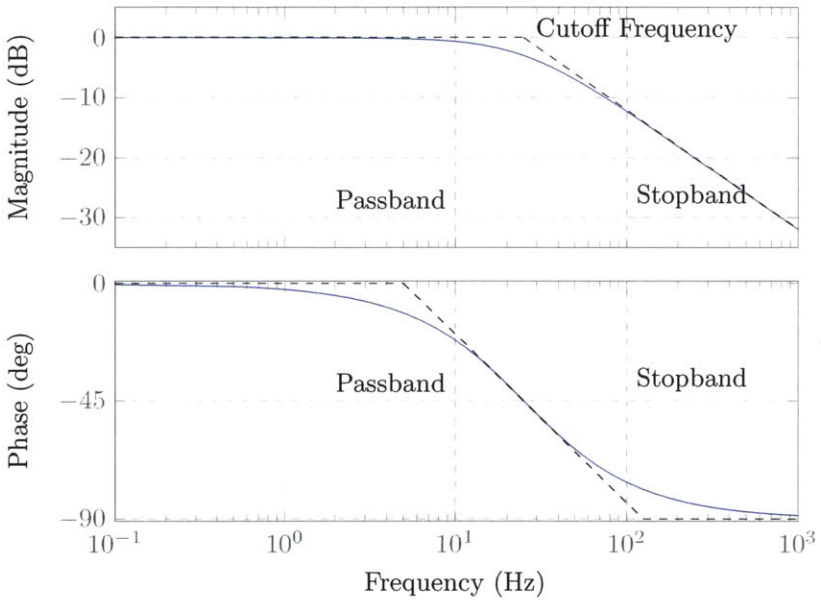


Figure 2-4: Bode plot of a model low pass filter.

A change in resistance or capacitance moves the cutoff frequency. As a result, the operating frequency may be pushed into the stopband or, conversely, the passband may be extended and erroneous inputs attenuated less. The phase incurred at the output is also affected, more so than the magnitude. This can be observed qualitatively from the bode plot. Operating in the passband a decade away from the cutoff frequency offers no significant attenuation in magnitude, but the phase incurred from the filter is already significant.

Analysis along these lines can be extrapolated for the operation of the system as a whole. Operating near the poles and zeros at any stage makes the entire system susceptible to magnitude and phase errors. Magnitude errors primarily result in signal attenuation through the signal conditioning. Phase errors indirectly affect the amplitude of the sensed signal by altering the phase of the signal at the synchronous detector. As will be described in Chapter 4, phase offset between the input signal and multiplied signal at the synchronous detector results in attenuation at the output. This

matters less for proximity detection since the magnitude of the in phase and quadrature amplitudes is found. However, these errors pose a larger problem for object detection where both individual amplitudes are needed.

2.4 System Partitioning

The gain in the system is distributed to provide a tradeoff between bandwidth and signal-to-noise ratio (SNR). The general principle for distributing the gain is conflicting. Gain should be applied as early as possible to maximize SNR. However, it should also be cascaded over several stages to maximize the gain bandwidth product.

Figure 2-3 presents a reference for the input referred noise of the standalone system. The noise at the input to the sense electrodes, n_i^2, is the combination of the noise contributed by the source, n_{src}^2 and the noise contributed by the sensor electronics, $n_{o,sensor}^2$, given by

$$n_i^2 = \mathrm{A} n_{src}^2 + \frac{1}{\mathrm{G_{sys}}} n_{o,sensor}^2, \tag{2.4}$$

where A is the attenuation ratio of the source signal to the input of the sensor and $\mathrm{G_{sys}}$ is the overall system gain. Because A is so small, the noise contribution of the signal source is minimal compared to that of the sensor electronics. This implies that the bulk of the noise in the system is from the sensing electronics.

Chapter 3

High Voltage Signal Source

The fluorescent lamp as an electric field source is inconsistent, non-uniform, and prone to drift, largely because of the properties of the plasma inside the bulb. As one of the goals of the standalone sensor is to improve upon the fluorescent lamp, a new signal source is needed to drive the electric field to sense changes in capacitance.

The benefit of the standalone source is that the idiosyncrasies of the fluorescent lamp, with its alternating linear voltage profile and lumped node model, no longer need to be addressed [1]. The electric field source consists of a plate driven by a high voltage amplifier with a sinusoidal reference. The operating range is chosen to be between 10 kHz and 100 kHz, and the output signal amplitude can be as large as 500 V from an input reference of less than 1 V. Furthermore, for sensor applications that require multiple simultaneous operating frequencies, the reference source is capable of outputting summed sinusoids.

The performance of the high voltage source determines the performance of the entire system. The most important consideration is the drift at the output of the source. Maintaining a steady output ensures that all sensing is possible and reliable.

To provide better performance than the fluorescent lamp, the signal source needs:

- Variable amplitude
- Wide operating frequency range
- Low noise floor
- Small short and long term drift

In this chapter, a general analysis for the design is discussed first. Specific implementations are detailed when necessary to illustrate design decisions.

3.1 Circuit Design

The general approach to designing the high voltage signal source is to amplify the input signal as much as possible, through op-amp stages, then rely on a transformer to further increase the output signal amplitude. The gain is broken up into multiple stages to maximize the system bandwidth. Noise performance is traded off, but as illustrated in Chapter 4, the attenuated source noise is negligible in comparison to the input referred noise at the sensing input. A block diagram of the full circuit, shown in Figure 3-1, outlines the stages of the signal source.

The circuit consists of a digital to analog converter synthesizing the input sine wave, which is fed into a preamplifier and then a closed loop power amplifier stage with an autotransformer.

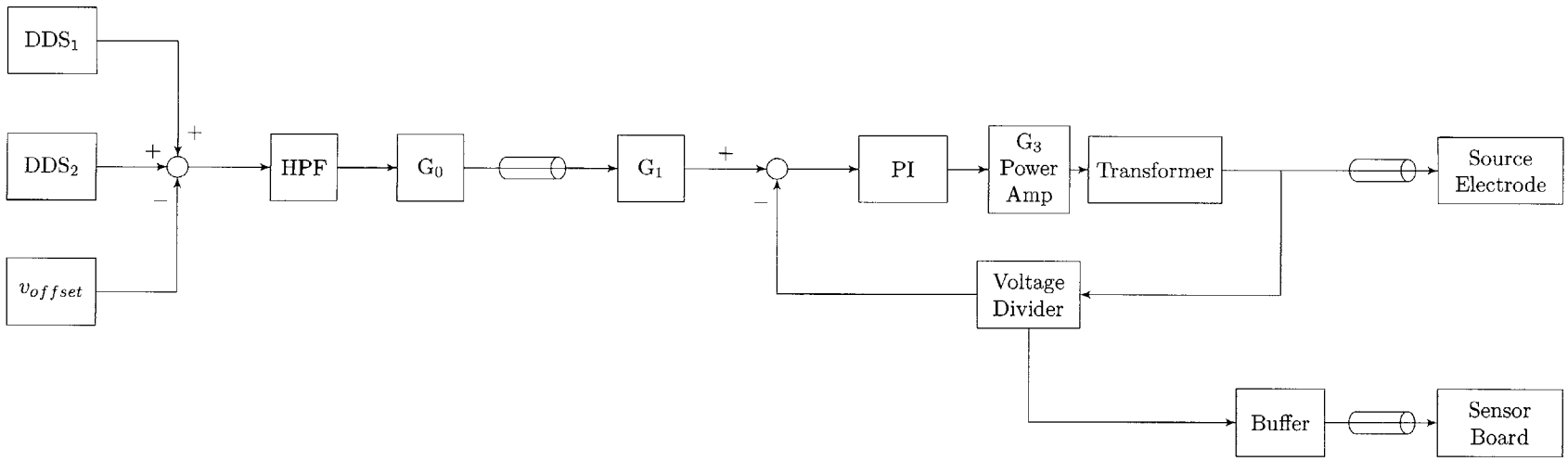


Figure 3-1: The block diagram of the high voltage signal source. Two DDS sine sources are summed and their offsets nulled. Gain is applied on the sensor board, G_o, then sent via coax cable to the HV board. A pre-gain stage, G_1, follows and then a proportional-integral controller for the power op-amp and autotransformer.

3.1.1 Digital to Analog Synthesis

The input sine wave is synthesized by a direct digital synthesis (DDS) chip, the AD9837, which is controlled via the serial peripheral interface (SPI) from a microcontroller. The AD9837 contains a digital to analog converter (DAC) along with other circuitry such as memory and logic, to generate a programmable sine wave at its output. Both frequency and phase offsets are controllable via the SPI bus.

The AD9837 output is constructed by clocking through a read only memory (ROM) that contains a lookup table. This table is used to convert the programmed input to the appropriate analog output. This output is created by an internal 10-bit DAC which makes the output voltage quantization 4 mV. The output frequency is given by

$$f_{out} = \frac{f_{mclk}}{2^{28}} \cdot \text{FREQREG}, \tag{3.1}$$

where f_{mclk} is the master clock frequency and FREQREG is the value programmed into the frequency control register. This constrains the output frequency to a resolution of 0.01 Hz.

The output signal spectrum is comprised of the output frequency with harmonics at multiples of the master clock frequency [2]. The magnitude of the harmonics falls off according to $\frac{\sin x}{x}$, where $x = \frac{\pi \cdot f}{f_{\text{mclk}}}$. If the output frequency is close to the master clock frequency, it is possible for the higher frequency harmonics to contribute significantly to the output . The highest operating frequency of the standalone sensor is 100 kHz. The master clock is maintained at 2.67 MHz, which makes the largest harmonic a small fraction of the fundamental

$$\text{A}_{\text{1st}} = \frac{\sin \pi \frac{(2.67\,\text{MHz} - 100\,\text{kHz})}{2.67\,\text{MHz}}}{\pi \frac{(2.67\,\text{MHz} - 100\,\text{kHz})}{2.67\,\text{MHz}}} \cdot \text{A}_{\text{fundamental}} = .0388 \cdot \text{A}_{\text{fundamental}}. \tag{3.2}$$

This implies that the harmonic content of the AD9837 is small.

Due to the capabilities of the DDS and its unipolar supply, the output is limited in amplitude and contains a dc offset. While neither are prohibitive, both must be addressed to best drive the high voltage amplifier. In order to accommodate simultaneous multiple frequency output, two waveforms are generated by two separate, selectable, DDS chips and subsequently summed.

A summing amplifier, pictured in Figure 3-2, is used to add and scale the outputs of the waveform generators, according to:

$$v_{out} = -R_f \left(\frac{v_{dds1} + v_{os}}{R_{dds}} + \frac{v_{dds2} + v_{os}}{R_{dds}} + \frac{v_{dc}}{R_{dc}} \right), \tag{3.3}$$

where each voltage input is scaled by the ratio of the feedback resistor, R_f, and the scaling resistors, R_{dds} and R_{dc}. The ac and dc components of each signal are explicitly written to illustrate the

additional undesired dc voltages. The applied gain at this stage is controlled by the ratio of the feedback resistance to the scaling resistances. The gain increases the signal amplitude, thereby resolving the low output amplitude of the DDS chips. However, the dc offsets are amplified at the same time.

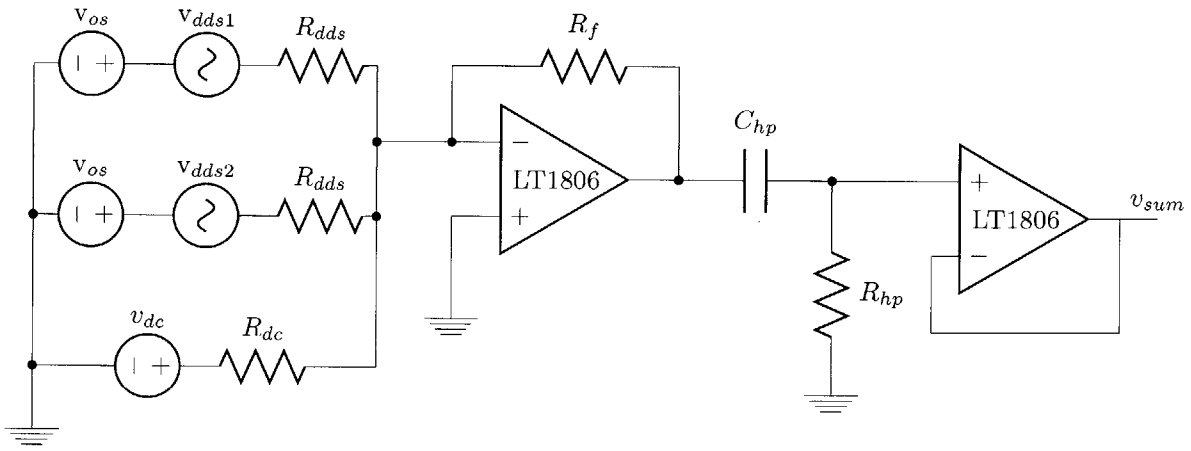


Figure 3-2: The circuit diagram of the summed, synthesized analog sinusoids. The offset subtraction is set by making v_{dc} negative. It is scaled by R_{dc}, whose value is chosen based on whether one or two DDS chips are on.

Two approaches are taken to handle the offsets of the synthesized waveforms. The first method consists of subtracting as much of the offset as possible. Doing so enables as large a gain as possible to be applied at the summing amplifier. The amount of dc subtraction required is dictated by the system configuration. Single frequency operation requires only one DDS chip and disables the other, leaving only one dc offset to negate, while multifrequency mode requires twice as much subtraction. By making v_{dc} negative, the approximately 300 mV offset applied by each DDS chip can be negated. The amount of subtraction is controlled by R_{dc} as noted in equation (3.3). With v_{dc} fixed at -5 V, R_{dc} is halved when both DDS chips are active, or kept at its nominal value with only one active DDS.

It is vital to subtract the offset as entirely as possible in order to maximize the gain applied during this stage. Any remaining offset will be amplified and lower the maximum voltage swing at the output. Thus, the largest gain that may be applied is

$$\mathrm{G} < \frac{V_{peak}}{V_{OFF} + v_{amp}} \tag{3.4}$$

where V_{peak} is the peak output voltage of the summing op-amp, V_{OFF} is the sum of the dc offset voltages, and v_{amp} is the sum of the ac amplitudes.

The remaining offset is removed by a high pass filter placed at the output of the summing

amplifier. It is important to keep the cutoff frequency of the filter

$$f_{cutoff} = \frac{1}{2\pi R_{hp} C_{hp}}, \tag{3.5}$$

far from the operating frequency in order to avoid attenuating the signal and to limit any phase offset. Pushing the cutoff lower in frequency requires increasing the resistance and capacitance accordingly. However, a larger capacitor leads to larger settling times, which affects how the filter reacts to drifts in the offset.

The last piece of the high voltage signal source on the sensor board is the cable driver. The output of the summing amplifier is sent to the high voltage board via coaxial cable. An op-amp is used with proper impedance matching to drive the cable and avoid corrupting the signal.

Implementation

The implementation uses the components from Table 3.1. The inputs to the summing op-amp are scaled equally. The offset subtraction is varied by changing R_{dc}, in order to cancel the offset imparted by the DDS chips. With the summing op-amp powered by $\pm 5\,\mathrm{V}$ supplies, a gain of 10 was chosen such that, with one DDS active, the total offset at the summing op-amp input may be

$$V_{OFF} < \frac{1}{10} \cdot \left(5\,\mathrm{V} - 10 \cdot 0.3\,\mathrm{V}\right) \tag{3.6}$$

$$< 0.2\,\mathrm{V}. \tag{3.7}$$

This is much larger than any expected offset and should be more than sufficient.

The cutoff of the high pass filter is set to 160 Hz, which is far from the lowest operating frequency of 10 kHz. At two decades above cutoff, the output is well within the passband. Thus, the magnitude error at the output is insignificant and the added phase at the output is much smaller than one degree. This also ensures that drift, in either direction, of the pole frequency has little effect on the output. Lastly, the filter settles within 6 ms, which is a tenth of the duration of the sampling window.

Table 3.1: Sine Synthesis and Conditioning Components

Component	Value
Summing Amp	LT1806
Cable Driver	LT1806
Gain	10
HPF Cutoff	160 Hz
R_{dds}	1 kΩ
R_{dc}	9 kΩ,17 kΩ
R_f	10 kΩ
R_{hp}	1 kΩ
C_{hp}	1 µF, film

3.1.2 Pre-Gain and PI Controller

Operating at high gains over the nominal frequency range places strict demands on the op-amps used in the signal source chain. These components must operate with sufficient bandwidth and slew rate. Furthermore, the inputs to each stage must be kept within the acceptable limits for the op-amps used. If these criteria are not met, the output of each stage may be distorted. Figure 3-3 presents the first two op-amp stages on the high voltage board. These first two stages are designed with the AD8620 and powered from ±12 V supplies. Despite these supplies, the inputs of the AD8620 are limited to ±10.5 V, while the outputs reach ±11.84 V. These specifications set the limits of the operating voltages for these op-amp stages.

The pre-gain stage is configured as a non-inverting amplifier with adjustable gain set by a potentiometer in the feedback path. The adjustable gain at this stage determines the final output amplitude of the signal source. This potentiometer must have a small temperature coefficient in order to lessen the amount of drift in the source. The second op-amp is configured as a proportional-integral (PI) controller with proper compensation in its feedback path. The control characteristics will be discussed in more detail in Section 3.1.5. The bottom half of its feedback network, R_{f2}, is comprised of the parallel combination of the resistors in the voltage divider feeding back a fraction of the high voltage output.

Slew Rate Limitations

The slew rate dictates the highest acceptable operating frequency and largest acceptable voltage amplitude for each stage. The slew rate of an op-amp is related to the peak voltage at the output by:

$$\mathrm{SR} = \frac{dV_{out}}{dt}, \tag{3.8}$$

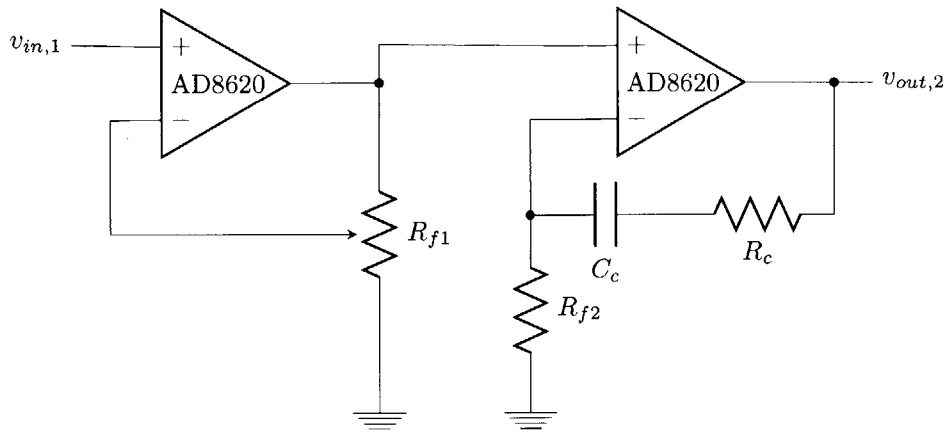


Figure 3-3: The circuit model for the first two op-amp stages: a pre-gain stage followed by a proportional-integral controller. Here, R_{f2} is the parallel combination of the high voltage divider at the output of the signal source, given by $R_{d1}||R_{d2}$.

and is determined by the internal topology and biasing. The slew rate, as a function of the input frequency, f, and amplitude ceiling, V_{pk}, is given by

$$\text{SR} \geq 2\pi f \cdot V_{pk}. \tag{3.9}$$

This relation sets the minimum slew rate necessary for the op-amp to produce an undistorted output for a given frequency and peak voltage.

For the purposes of the pre-gain op-amp and PI controller, the minimum slew rates necessary at the extreme ends of the desired operating region and for ideal operation are

$$\text{SR}_{min} = 2\pi \cdot 10\,\text{kHz} \cdot 12\,\text{V} = 0.75\,\text{V/}\mu\text{s} \tag{3.10a}$$

$$\text{SR}_{min} = 2\pi \cdot 100\,\text{kHz} \cdot 12\,\text{V} = 7.5\,\text{V/}\mu\text{s}, \tag{3.10b}$$

which fall well below the specified 60 V/µs slew rate of the AD8620 [3].

Failing to reach the minimum slew rate does not necessarily prohibit all functionality. Instead it lessens the output capabilities of the system, in one of two ways. If a stage can not meet the minimum slew rate specification, the op-amp may be driven at a lower frequency or signal amplitude to reduce the minimum slew rate. Conversely, if the slew rate criteria is disregarded the output signal will appear distorted.

Per Stage Bandwidth Requirements

Another characteristic vital to the performance of the high voltage signal source is the bandwidth of each stage. The op-amps used in each stage must be capable of running at the highest operating frequency of the system. Furthermore, the product of the applied gain, G, and operating frequency,

f, must remain constant, as dictated by the op-amp's specified gain bandwidth product (GBW):

$$\text{GBW} = \text{G} \cdot f. \tag{3.11}$$

The GBW is also based on the internal topology and technology of an op-amp and guides the design decisions regarding the maximum gain that may be applied for a given frequency. The larger the gain applied the lower the bandwidth. Thus, in applications requiring high gain, the op-amps used must have sufficiently large bandwidth in order to accommodate the large gain. The GBW of the AD8620 [3], at 25 MHz, is much larger than the highest operating frequency.

Offset Considerations

Because the eventual output of the signal source is a transformer, offsets propagated through the signal chain are of particular concern. Offsets at the input of the transformer can result in increased losses and possible saturation. Sources of additional offset may include op-amp input offset voltage and input bias current. The input offset voltage is the voltage necessary to force the op-amp output voltage to zero with no inputs present. However, when inputs and feedback are present this offset voltage will be amplified alongside the signal:

$$v_{out} = \text{G}(v_{in} + V_{os}) \tag{3.12}$$

Input bias current are a fundamental property of an op-amp. Ideally, no current flows into or out of the input of the op-amp, but in reality some current is required to bias the input transistors. For the case of the signal source gain stages, a worst case offset contribution from the input bias current of the op-amp can be expressed as:

$$v_{offset} = I_B(R_f||R_g), \tag{3.13}$$

where I_B is the input bias current and both R_f and R_g are the resistances in the non-inverting op-amp feedback network. This applies to both stages. The pre-gain stage input sees the resistance of the potentiometer. At dc, the feedback capacitor of the PI controller is effectively open and the input sees the voltage divider resistance. Therefore, if both stages use the AD8620, the total expected offset at the output can be given by the sum of the gain offset product and the offsets due to the input bias currents:

$$v_{offset} = (G_1 + G_2)V_{os} + (R_{f1} + R_{f2})I_B. \tag{3.14}$$

R_{fbx} denotes the lumped feedback resistances of each stage. Mitigating the offsets at these later

stages is an important consideration for the signal source.

Implementation

The components implemented in these op-amp stages are listed in Table 3.2. The input bias current of the AD8620 is a mere 10 pA and even with extremely large feedback resistances may be safely ignored. However, the input offset is listed at 250 µV, which means that large gains can potentially result in millivolt level offsets at the outputs of these stages. Since a 10 kΩ potentiometer is used in the feedback of the pre-gain stage, the maximum gain and bandwidth are

$$G_{max} = 1 + \frac{9\,k\Omega}{100\,\Omega} = 91, \tag{3.15a}$$

$$BW_{max} = \frac{GBW}{G_{max}} = 275\,kHz. \tag{3.15b}$$

Gain errors in the system may arise if the potentiometer is particularly sensitive to temperature. The error itself is less of a concern than the potential gain drift. A 5% error in the overall gain will lower the overall output voltage, but may be accounted for by simply increasing the gain. However, temperature related drift, which translates to a temperature dependent gain, produces long term effects that hinder the performance of the sensor. The Vishay Accutrim 1260W series 10 kΩ potentiometer used has a low temperature coefficient of ±10 ppm/°C, but because the potentiometer is used radiometrically, the actual gain temperature coefficient is much lower.

Table 3.2: High Voltage Pre-Gain and PI Controller Components

Component	Value
Op-Amps	AD8620
R_{f1}	10 kΩ pot, Accutrim 1260W
R_{f2}	10 kΩ || 1 MΩ
C_c	330 pF

3.1.3 Wide Rail Power Operational Amplifier

The third stage is designed to use the OPA452 power op-amp, which has a much wider supply range compared to the previous stages. This voltage translation allows for another stage of closed loop amplification. Increasing the supply range enables the stage to further amplify the signal and increase the potential output voltage swing. The third stage amplifier and the output transformer, with its compensation, are shown in Figure 3-4.

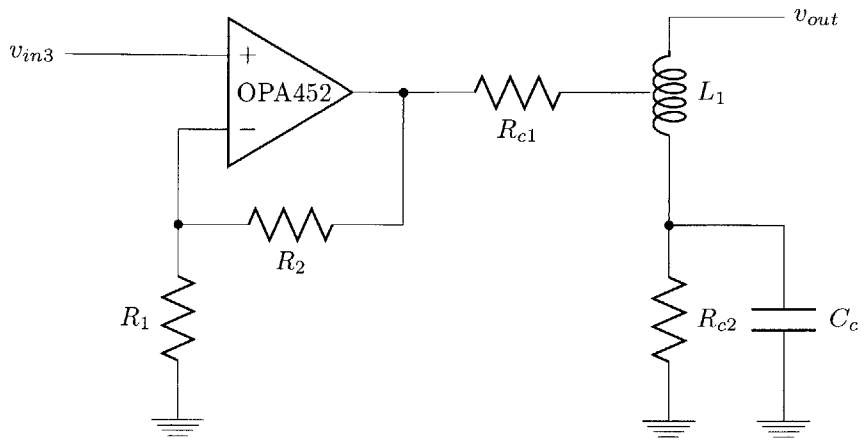


Figure 3-4: The circuit diagram of the wide swing op-amp and output autotransformer.

Gain Limitations

This stage is configured as another non-inverting amplifier with a fixed gain of two. The reason for such a conservative gain can again be explained by the slew rate and gain bandwidth product limitations. Similar to equation (3.10), the minimum slew rate necessary to utilize as much of the wider rails as possible is

$$\mathrm{SR}_{min} = 2\pi \cdot 10\,\mathrm{kHz} \cdot 40\,\mathrm{V} = 2.5\,\mathrm{V/\mu s}, \tag{3.16a}$$

$$\mathrm{SR}_{min} = 2\pi \cdot 100\,\mathrm{kHz} \cdot 40\,\mathrm{V} = 25\,\mathrm{V/\mu s}. \tag{3.16b}$$

The specified slew rate for the OPA452 is 7.2 V/µs [4]. This means that the low end of the operating frequency range is within bounds, but the high end far exceeds the capabilities of this op-amp. While this again does not prohibit all functionality at higher frequencies, it does require a lower peak voltage at 100 kHz to remain operable. The potential output voltage from this stage is capped at ±39 V from the supplies, but decreases as frequency increases based on the slew rate and operating frequency. The relationship between the peak output voltage and frequency is shown in Figure 3-5 and given by

$$v_{out} = \frac{\mathrm{SR}}{2\pi f} = \frac{7.2\,\mathrm{V/\mu s}}{2\pi f}. \tag{3.17}$$

A large gain not only pushes the voltage output limit of the op-amp by forcing it to operate near its rails, but also affects the stability of the entire system. As discussed in Section 3.1.2, large gains lower the stage crossover frequency. The gain bandwidth product of the OPA452, which is specified to be 1.8 MHz, limits the amount of gain applied at this stage. Such a low GBW, only about one decade away from the highest operating frequency, places further emphasis on remaining conservative with the gain applied at the third stage to operate the stage below crossover.

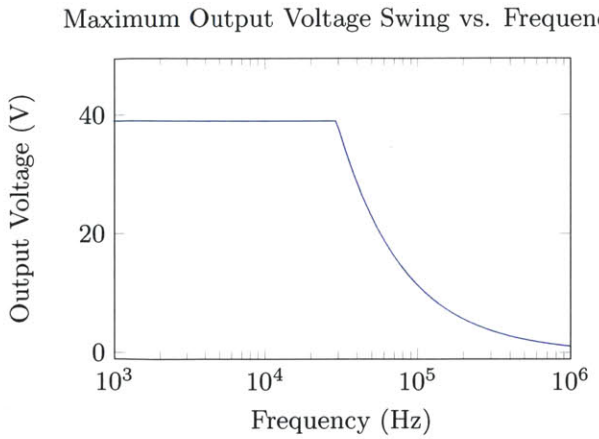


Figure 3-5: OPA452 Peak Output Voltage vs. Frequency

Output Resistance

The internal output resistance of an op-amp can cause stability issues when driving a capacitive load. Due to its configuration as a non-inverting amplifier, the apparent output resistance of the component is scaled by the ratio of the open and closed loop gains, given by

$$R_{out} = r_o \cdot \frac{A_{CL}}{A_o}. \tag{3.18}$$

As a result, small gains result in exceedingly small output resistances. This allows an op-amp to appear as a better voltage source. However, the op-amp output resistance, r_o, along with any load capacitance will create an extra pole in the loop transmission. The location of this pole may potentially cause the op-amp to become unstable. The lower the output resistance, the lower the frequency of the additional pole, which further reduces the crossover frequency of the stage. Further feedback analysis is discussed in Section 3.1.5.

Implementation

In line with the the design parameters, the third stage was set to a gain of 2, which means that the bandwidth of the stage is limited to 900 kHz. In order to accommodate the high end of the operating frequency range, the stage is driven such that the largest output voltage is not greater than

$$v_o = \frac{7.2\,\text{V}\,\mu\text{s}}{2\pi 100\,\text{kHz}} = 11.4\,\text{V}. \tag{3.19}$$

This lowers the maximum possible output voltage, but is necessary to meet the slew rate requirements for the OPA452. The rails of the op-amp were then reduced to ± 25 V since the extra headroom was unnecessary. Table 3.3 lists the implemented parameters for the wide rail gain stage.

Table 3.3: High Voltage Wide Rail Gain Stage Components

Component	Value
Op-Amp	OPA452
R_1	50 kΩ
R_2	50 kΩ
R_{c1}	10 Ω
R_{c2}	1 kΩ
C_{c1}	2.2 µF, metal film

3.1.4 Autotransformer

The final stage of the high voltage signal source is the output transformer. The transformer design for a single ended source is a single winding autotransformer, as depicted in Figure 3-4. This type of transformer consists of one winding with at least one center tap. By using an autotransformer less copper is needed and leakage inductances are smaller in comparison to an isolated transformer. Given a current density, J, and a cross sectional area, A, the current, I, through a conductor is given by:

$$I = J \cdot A. \tag{3.20}$$

Furthermore, for a transformer, with N turns on the primary or secondary, the total effective cross sectional area, and in a sense the amount of material used, is:

$$A_{total} = \frac{I \cdot N}{J}. \tag{3.21}$$

Given equal current densities and areas, the ratio between an autotransformer and two winding transformer is related by:

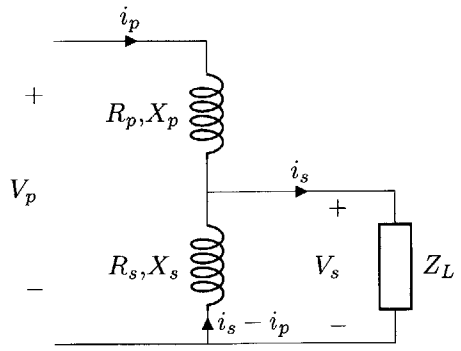
$$\frac{\text{autotransformer material}}{\text{two winding material}} = \frac{(N_t - N_s)I_p + N_s(I_s - I_p)}{N_t I_p + N_s I_s}, \tag{3.22}$$

where N_t is the total number of turns of the autotransformer, which is equivalent to N_p, the number of turns on the primary of the two winding transformer. Simplifying and recognizing that $N_t I_p = N_s I_s$ from fundamental transformer properties, results in:

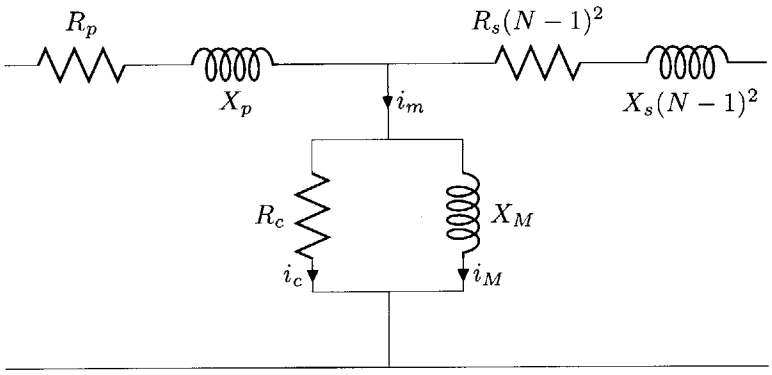
$$\frac{\text{autotransformer material}}{\text{two winding material}} = 1 - \frac{N_s}{N_t}, \tag{3.23}$$

confirming that the autotransformer makes more efficient use of its material. This makes for a physically smaller and lighter component.

The autotransformer is also more efficient electrically. Compared to a two winding transformer, it experiences lower losses through the secondary leakage impedances. This is detailed by examining



(a) Step down model for the autotransformer.



(b) Equivalent circuit model for the autotransformer.

Figure 3-6: Two circuit models for the autotransformer.

the voltage loops of the circuit model of the autotransformer, shown in Figure 3-6. Using Kirchhoff's voltage laws, these loops can be expressed as:

$$V_p = E_p + I_p(R_p + jX_p) - (I_s - I_p)(R_s + jX_s) \tag{3.24a}$$

$$E_s = V_s + (I_s - I_p)(R_s + jX_s). \tag{3.24b}$$

The losses seen at the primary due to series resistances and leakage inductances can be found by the conversion $E_p = NE_s$. Substituting this back into equation (3.24a), and simplifying results in:

$$\begin{aligned} V_p &= NV_s + N(I_s - I_p)(R_s + jX_p) + I_1(R_p + jX_p) - (I_s - I_p)(R_s + jX_s) \\ &= NV_s + I_p\big[(R_p + jX_p) + (N-1)^2(R_s + jX_s)\big], \end{aligned} \tag{3.25}$$

where the losses are:

$$R_l = R_p + R_s(N-1)^2 \tag{3.26a}$$

$$X_l = X_p + X_s(N-1)^2. \tag{3.26b}$$

The ratio between the autotransformer secondary losses and those of a transformer with isolated windings is given by:

$$\frac{(N-1)^2}{N^2}, \tag{3.27}$$

again confirming the higher efficiency of the autotransformer due to its lower losses.

The autotransformer is not without its disadvantages. For example, there is no isolation. While this may be problematic for other applications, the standalone signal source does not demand it. In fact, the opposite is the case; the non-isolated output of the autotransformer allows for a feedback path to be wrapped around the transformer to the PI controller.

The lower leakage inductance of the autotransformer is advantageous for the sensor signal source. Since the output of the transformer drives a coaxial cable, the cable capacitance may resonate with the leakage inductance. The frequency of this resonance is

$$f_{resonance} = \frac{1}{2\pi\sqrt{L_\ell C_{cable}}}, \tag{3.28}$$

where L_ℓ is the leakage inductance of the transformer and C_{cable} is the cable capacitance. The cable capacitance is a fixed parameter of the coaxial cable that increases with every additional segment. Thus, reducing the leakage inductance as much as possible forces the resonant frequency higher and farther away from the operating region. For damping, a resistor, R_c, may be placed in series with the input of the autotransformer. It may be chosen so that

$$R_c = \sqrt{\frac{L_\ell}{C_{cable}}}, \tag{3.29}$$

but too large of an R_c reduces damping of the parallel resonance with the magnetizing inductance. This implies that small leakage inductance is preferable.

Inductance, Core, and Turns Design

The design of the primary of the autotransformer is not unlike that of the two winding version. Based on the core parameters, required peak voltage, and operating frequency, the magnetizing inductance and required turns may be designed. The magnetizing inductance may be determined by

$$L_{m,min} = A_L \cdot \left(\frac{V_{pk}}{\omega B_{sat} A}\right)^2; \tag{3.30}$$

where A_L, the effective inductance per winding, B_{sat}, the maximum peak flux density before reaching core saturation, and A, the effective area, are parameters specified for a given core. The peak expected voltage at the output of the transformer is V_{pk}. The minimum magnetizing inductance is

primarily dictated by the maximum output current of the amplifier, $I_{max,amp}$, driving the coil

$$L_{m,min} = \frac{V_{primary}}{\omega I_{max,amp}}. \tag{3.31}$$

With this in mind, the minimum required turns can be found by

$$N_{p,min} = \sqrt{\frac{L_{m,min}}{A_L}}. \tag{3.32}$$

The core used in the design is the Ferroxcube P42/29-3B9 and its specifications are listed in Table 3.4. The corner cases for the minimum magnetizing inductance and primary turns of the design are printed in Table 3.5.

Table 3.4: P42/29-3B9 Core and Material Specifications

Parameter		Value
Effective Area	A_e	$265\,\text{mm}^2$
Maximum Flux Density	B_{sat}	470 mT
Inductance per Turn	A_L	$11500\,\text{nH/N}^2$

Table 3.5: Autotransformer Design Corners

V_{pk}	f	L_m	N_p
200 V	10 kHz	7.5 mH	26
200 V	100 kHz	75 μH	3
500 V	10 kHz	46.9 mH	64
500 V	100 kHz	469.5 μH	7

Implementation

The autotransformer was wound according to the 10 kHz and 200 V design, but operates up to 100 kHz and 500 V. The magnetizing inductance was doubled for margin, and other parameters are listed in Table 3.6. A threaded nylon rod and a pair of nylon nuts are used to clamp down the two halves of the core. Before clamping, it is imperative to remove any debris that may prevent a flush mating of the core halves. Even the slightest air gap between the halves is detrimental to its performance by decreasing the magnetizing inductance of the transformer, which increases the magnetizing current.

Table 3.6: Autotransformer Parameters

Winding	Turns	Gauge	L_m	L_l	R_L	$C_{interwinding}$	$f_{resonance}$
Primary	38	32 AWG	14.57 mH	12.5 μH	16 Ω	400 pF	25 kHz
Secondary	380	32 AWG	1.76 H	12.5 μH	1.7 kΩ	400 pF	25 kHz

3.1.5 High Voltage Feedback and Control

Driving Cable Capacitance

As depicted in Figure 3-7, the output of the third stage of the signal source is loaded by the parallel capacitance combination, $C_{L,p}$, of the transformer interwinding capacitance and the reflected capacitance at the primary due to the coaxial cable at the output of the secondary.

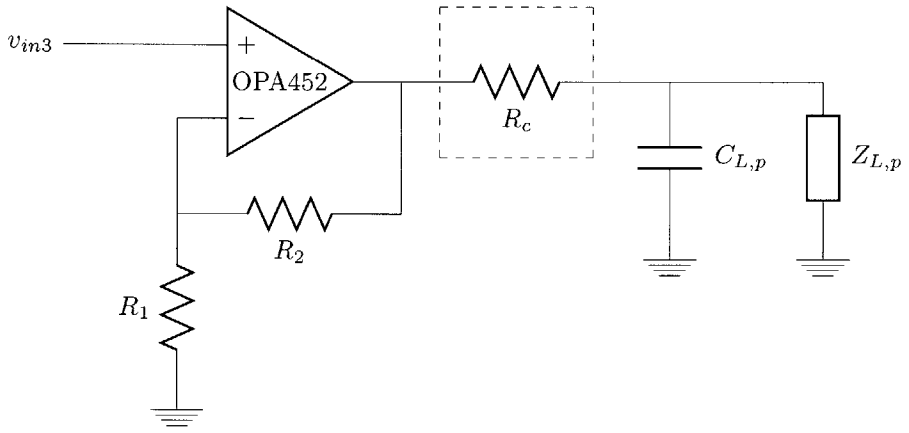


Figure 3-7: The circuit model for the third gain stage op-amp and reflected load with out-of-loop compensation resistor R_c.

The reflected load capacitance seen at the output of the third stage is found by

$$C_{L,p} = \left(\frac{N_p}{N_s}\right)^2 \cdot C_{L,s}, \tag{3.33}$$

which shows that $C_{L,p}$ can present a large capacitive load. Furthermore, since the load capacitance is dominated by the cable capacitance, the longer the cable, the larger the load capacitance. This load, in combination with the output resistance of the op-amp, adds a complex pole if the inductance is include. When overdamped, the pole that matter is

$$f_p = \frac{1}{2\pi\, r_o\, C_L}. \tag{3.34}$$

As the load capacitance increases, the frequency of the pole is lowered. If this pole is lower than the unity gain frequency of the op-amp, the phase and gain margins will suffer and consequently the op-amp may become unstable.

A zero can be added in order to mitigate the effects of the pole. This can be done by placing a resistor at the output of the op-amp, R_c in Figure 3-7, which interacts with the load capacitance creating a zero at a frequency of

$$f_z = \frac{1}{2\pi R_c C_L}. \tag{3.35}$$

Proportional-Integral Controller

The output of the transformer is passed through a voltage divider and fed back to a proportional-integral (PI) controller. The feedback network is designed to operate with unity gain at dc, while ac

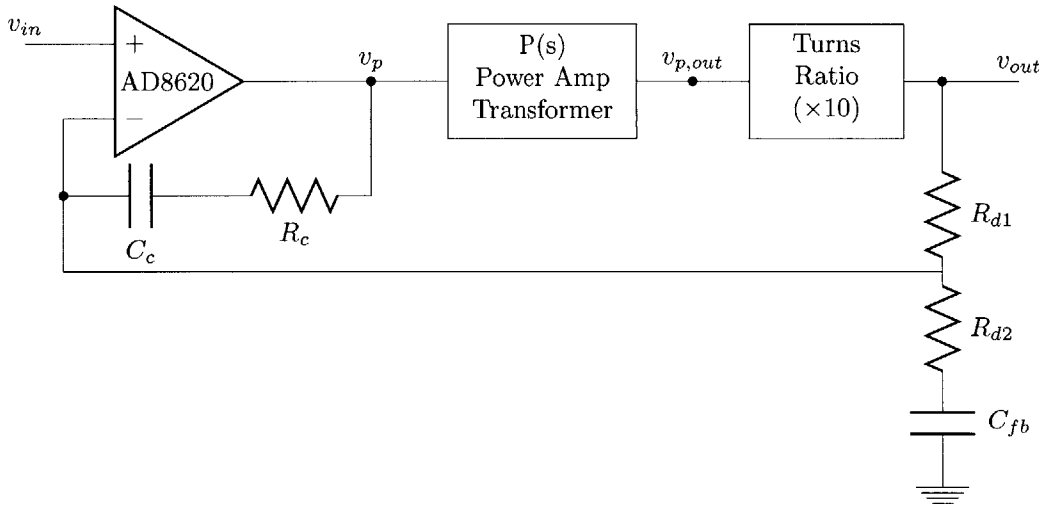


Figure 3-8: The circuit model for the PI controller and plant.

signals are fed back with a return ratio so there is amplification. The capacitor C_{fb} at the bottom of the divider is an open at dc so feedback is unity. At signal frequencies, the feedback ratio is

$$\frac{R_{d2}}{R_{d1} + R_{d2}} \tag{3.36}$$

The general block diagram for the ac control is shown in Figure 3-9.

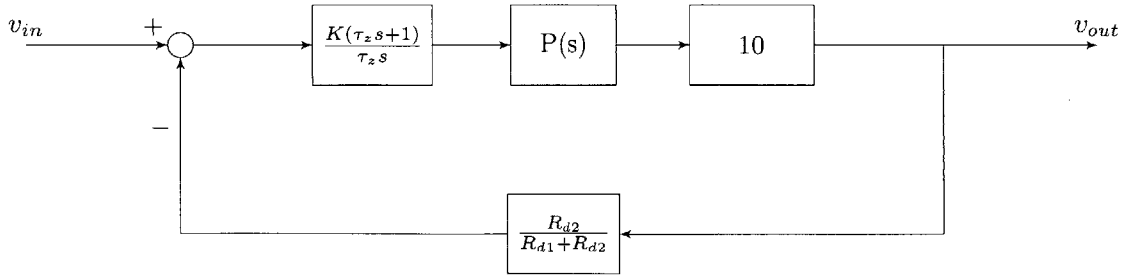


Figure 3-9: Block diagram for the proportional-integral controller.

The plant in this system, as shown in Figure 3-8, is the combination of the OPA452 power amplifier and the autotransformer dynamics. As specified in the OPA452 datasheet, the component

exhibits a two pole roll off [4]. The power amp is configured as a non-inverting amplifier, in Figure 3-10, with a gain of two, hence the return ratio. Its output however is loaded by the transformer and reflected cable capacitance. This affects the closed loop response of the power amp. The load at

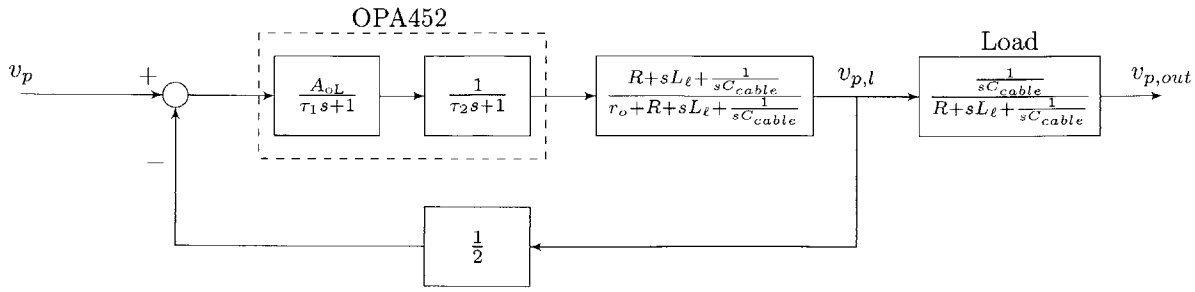


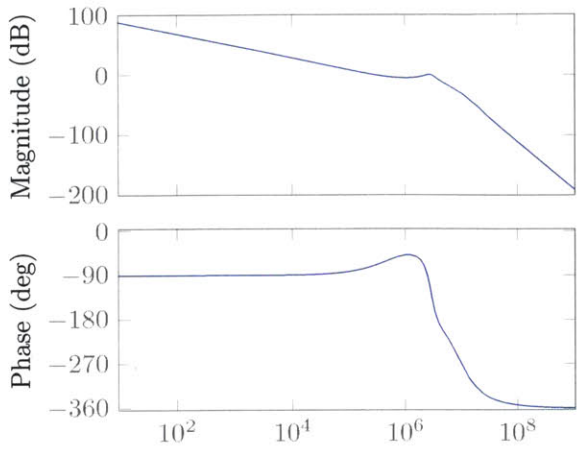
Figure 3-10: The block diagram of the plant at the high voltage source output.

the output is the series RLC circuit formed by the resistor at the output of the power amp, R, the transformer leakage inductance, L_ℓ, and the reflected cable capacitance, C_{cable}. The output is taken after the resistor and inductor, at the top of capacitor, which forms an ac voltage divider given by

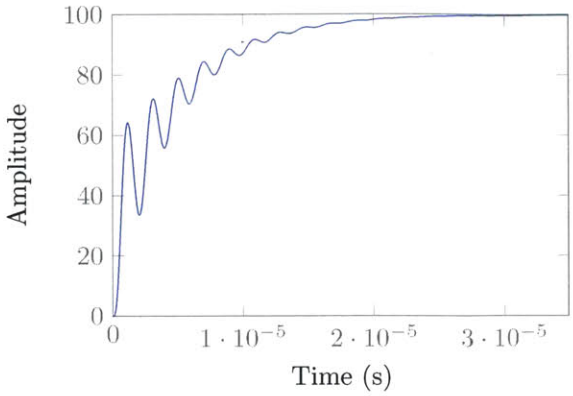
$$\begin{aligned} \frac{v_{p,out}}{v_{p,l}} &= \frac{\frac{1}{sC_{cable}}}{R + sL_\ell + \frac{1}{sC_{cable}}} \\ &= \frac{1}{s^2 L_\ell C_{cable} + sRC_{cable} + 1}. \end{aligned} \tag{3.37}$$

This is the load of the system, whose output is then divided and fed back to the controller. Finally, the full plant is the product of the closed loop response of the OPA452 and the autotransformer load.

The proportional and integral gains of the PI controller are the main parameters that control the performance of the system. These parameters manage the overshoot and settling time of the system. The proportional gain is set to about 2 and the integral gain to 3×10^{-6}. As illustrated in Figure 3-11a, the system is stable with a phase margin of 60.4°. The step response exhibits some ringing but settles in approximately 30 ms because the resonance is not well damped.



(a) The loop transmission of the high voltage PI controller.



(b) The step response of the high voltage PI controller.

Figure 3-11: Plots of the dynamics for the PI controller implemented in the high voltage amplifier.

Driven DC Offset

Lastly, as seen in Figure 3-4, an RC network is placed at the base of the primary of the transformer. The resistor prevents the transformer from appearing as a short at dc. The capacitor shorts the resistor at signal frequencies.

3.2 Summary of Tradeoffs

Since the signal source is the drive behind the capacitive sensing, limiting the amount of offset and amplitude and phase drift of the source is important. However, drift in the source is not so well negated. The problem with short and long term drifts appears when defining a detection boundary.

The most significant of considerations in the high voltage signal source are the tradeoffs between gain and offset, and gain and bandwidth. Though offset voltages are accounted for in the overall system, any lingering offset present at the input of a stage becomes significant because of the large gains applied stage by stage, and to the overall signal.

Chapter 4

Sensing Circuitry Design

This chapter will discuss the design and implementation of the sensing electronics. This includes the front end amplifiers on each receive electrode and the signal conditioning circuitry on the sensor board. The sensing circuitry is designed with many of the same considerations as the high voltage board. Low noise and small drift are still the main concerns and drive the design of the circuitry presented in this chapter.

4.1 Front End Amplifier

The goal of the front end amplifier is to measure the slight changes in the amplitude and phase of the input signal. Unlike previous iterations of the sensing circuitry, the standalone sensor uses two single ended front end amplifiers followed by a differential amplifier at the sensing plates with common-mode feedback. The two front end amplifiers form an output measurement

$$v_{out} = A_d(v_1 - v_2) + A_{cm}\left(\frac{v_1 + v_2}{2}\right), \tag{4.1}$$

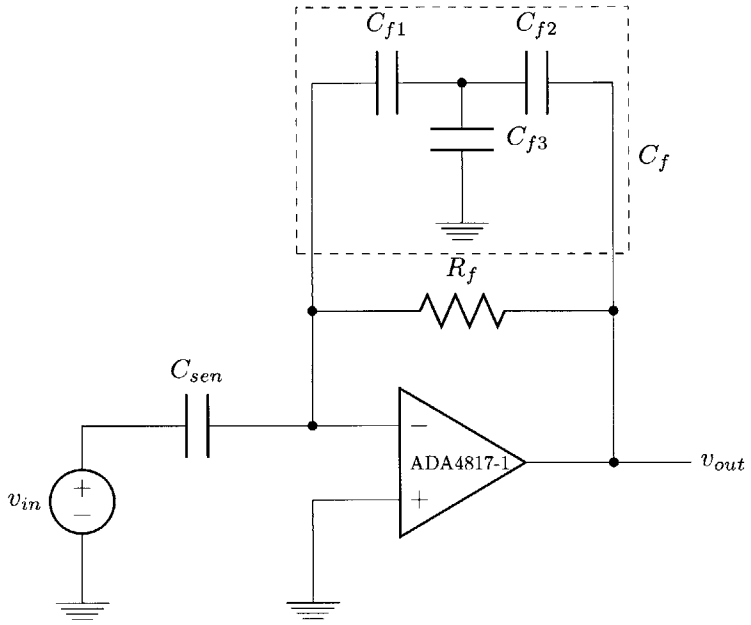
where v_1 and v_2 are the sensed signals into each amplifier, A_d is the differential gain, and A_{cm} is the common mode gain. The differential output signals are the source of the small sensed amplitude and phase changes, which correspond to a detection, and thus must be preserved as much as possible. On the other hand, the common mode signals are undesired and rejecting them is important. The method of common mode rejection is one of the most significant differences in the standalone sensor and will be detailed in Section 4.1.2.

The front end amplifier used in the standalone sensing system is physically located on the receive electrodes. Previous versions of the sensor treated the receive electrodes as simple plates separated from the electronics. This method reduces the complexity of the system, but lengthens the sensing node making it more susceptible to parasitic capacitances from the undriven cable and the

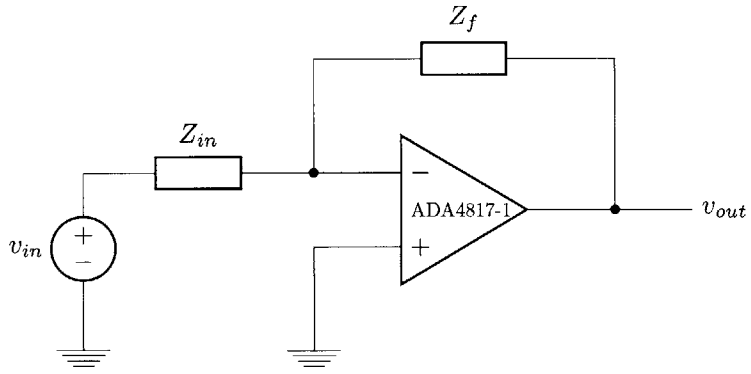
environment. Thus, the aim of mating the front end amplifier with the receive electrode is to:

- control the extent of the sensing node
- reduce the effects of cable capacitance and interference from inductive coupling

Addressing these issues is vital to improving the performance of the entire system.



(a) Full schematic of the front end amplifier. The capacitive T-network in the feedback path is modeled by the lumped capacitance C_f.



(b) Simplified schematic of the front end amplifier.

Figure 4-1: The full and simplified circuit diagrams for the front end amplifier.

A simplified version of the front end amplifier used in the standalone sensor is shown in Figure 4-1b, where the feedback network is lumped and generalized as Z_f. For simplicity, the capacitive T-network is replaced with an equivalent effective capacitance; analysis of the T-network is discussed later in this section and in Section 4.3.1. Based on the amplifier configuration, it will convert an input current to an output voltage. Since the inverting node is a virtual ground, by applying Kirchhoff's

current law the output can be found to be

$$\begin{aligned} v_{out} &= -i_{in} Z_f \\ &= -i_{in} \left(R_f \middle|\middle| \frac{1}{sC_f} \right). \end{aligned} \tag{4.2}$$

The input current is provided by the signal source, v_{in}, and the sensed capacitance to the receive electrode, Z_{in}, by

$$i_{in} = \frac{v_{in}}{Z_{in}} = v_{in} \cdot sC_{sen}. \tag{4.3}$$

Substituting equation (4.3) into equation (4.2) and simplifying, the transfer function of the front end amplifier in the ideal case can be shown to be

$$\frac{v_{out}}{v_{in}} = \frac{sR_f C_{sen}}{1 + sR_f C_f}. \tag{4.4}$$

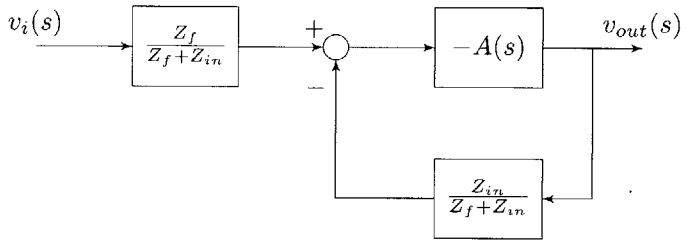


Figure 4-2: The generalized block diagram for the front end amplifier.

Figure 4-2 illustrates the block diagram of the front end amplifier and helps find several important system parameters. The loop transmission, $L(s)$, is the product of the forward path, $A(s)$, and the feedback path, $H(s)$,

$$L(s) = A(s)H(s) = A(s) \cdot \frac{Z_{in}}{Z_{in} + Z_f} \tag{4.5}$$

The forward path consists of the open loop gain of the op-amp, as defined by the gain bandwidth product and unity gain frequency of a given component. Lastly, the closed loop transfer function of the system can be derived using Black's formula and shown to be

$$\frac{v_o}{v_{in}} = \frac{Z_f}{Z_{in} + Z_f} \cdot \frac{A(s)}{1 + A(s)\dfrac{Z_{in}}{Z_{in} + Z_f}}. \tag{4.6}$$

When A(s) is sufficiently large, the transfer function can be confirmed with equation (4.4)

$$A(s) \to \infty : \frac{v_o}{v_{in}} = \frac{Z_f}{Z_{in}} = \frac{sR_f C_{sen}}{1 + sR_f C_f}. \tag{4.7}$$

4.1.1 Active Receive Electrodes

On-Plate Implementation

The sensing node is one of the most sensitive points in the system. Constraining the sensing node to the receive electrode helps to avoid parasitic capacitances, reduces the potential to pick up extraneous signals from noise sources, and limits interference from the environment. These parasitic capacitances can be modeled by a lumped capacitor to ground at the input of the amplifier.

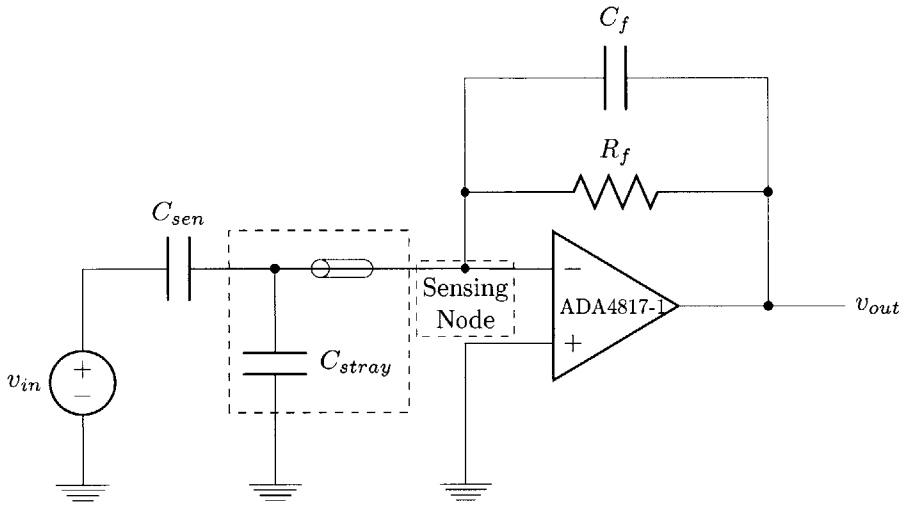


Figure 4-3: Front end amplifier with the modeled lumped cable and parasitic capacitance, inside the dashed box. The configuration of the front end amplifier used in the standalone sensor removes this capacitance.

In previous electrode configurations, the dominant stray capacitance was contributed by the coaxial cable connecting the electrode to the front end amplifier, as shown in Figure 4-3. To account for this in the dynamic response of the amplifier, the sensed input capacitance may be lumped together with any other stray input capacitances present in the environment

$$\begin{aligned} Z_{in} &= Z_{sen} \| Z_{stray} \\ &= \frac{1}{sC_{sen}} \Bigg\| \frac{1}{sC_{stray}}. \end{aligned} \tag{4.8}$$

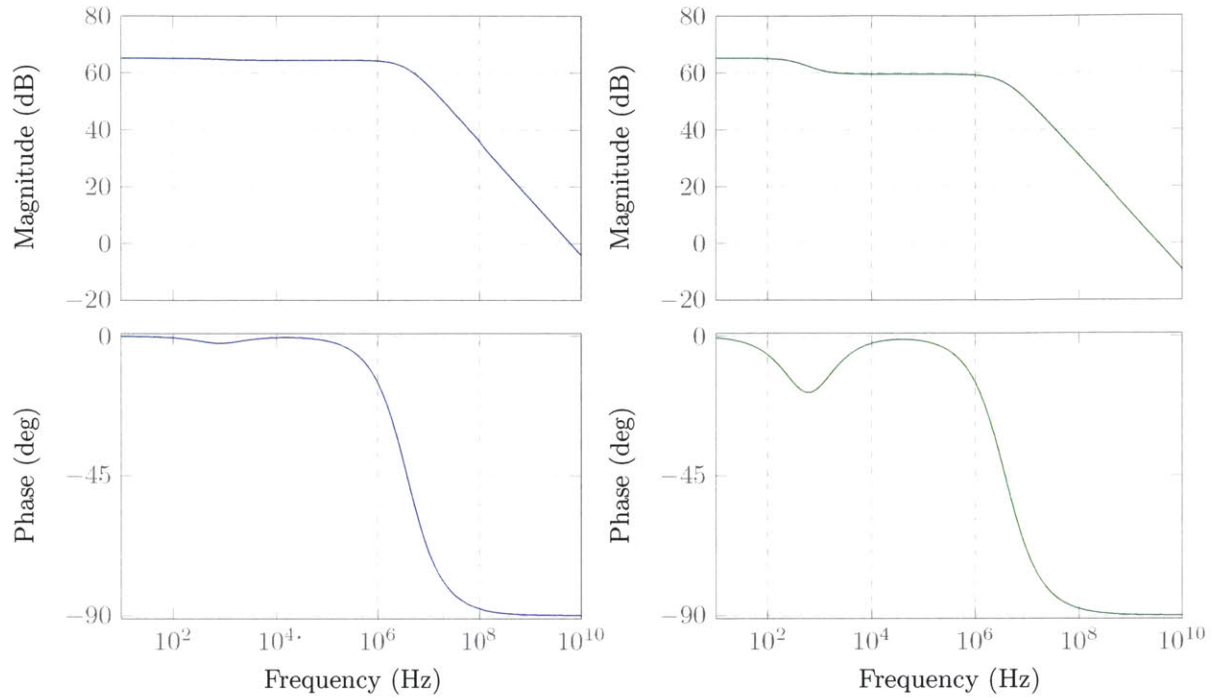
The effect of the added capacitance is observed in the new closed loop and loop transfer functions for the system, for a real $A(s)$, which become

$$\frac{v_o\,'}{v_{in}} = \frac{A(s) \cdot sR_f(C_{sen} + C_{stray})}{sR_f(C_{in} + C_{stray}) + (A(s) + 1)(1 + sR_fC_f)}. \tag{4.9}$$

$$L(s)' = A(s)\frac{1 + sR_fC_f}{1 + sR_f(C_{sen} + C_f + C_{stray})} \tag{4.10}$$

The stray capacitance lowers the loop gain and shifts the pole lower in frequency. While the closed

loop pole is shifted lower in frequency, the zero remains at the origin. The bode plots in Figure 4-4 demonstrate these effects for one set of parameters, R_f, C_f, C_{sen}, with and without C_{stray}. For the purposes of this bode plot $R_f = 10\,\text{M}\Omega$, $C_f = 180\,\text{pF}$, $C_{sen} = 10\,\text{pF}$, and $C_{stray} = 100\,\text{pF}$.



(a) Magnitude and phase frequency response without stray capacitances present.

(b) Magnitude and phase frequency response with stray capacitances present.

Figure 4-4: Loop transfer functions as affected by stray capacitance.

Because the front end amplifier is located on the receive electrode, there is no cable connected to the sensing node. As a result, the cable capacitance no longer effects the input of the front end amplifier. This drastically reduces the stray capacitances at the sensing node. A typical RG-178 coaxial cable has approximately 30 pF per foot of cable and thus may easily dominate the stray capacitances present [22]. The cable interaction at the input node of the front end amplifier is removed entirely, and instead the output of the front end amplifier is buffered and sent to the signal conditioning via a properly driven cable.

Resistive vs. Capacitive Dominated Feedback

Two feedback modes can be used in the front end amplifier: resistor dominated and capacitor dominated. The feedback network for the front end amplifier is a parallel combination of the feedback resistance and capacitance as given by

$$Z_f = \left(Z_{R_f} || Z_{C_f}\right) \tag{4.11}$$

These modes correspond to the dominant impedance in the parallel configuration. Dominance in this sense refers to the impedance through which the most current flows. In other words, capacitive mode feedback refers to the case where Z_{C_f} is much smaller than Z_{R_f}, the reverse is true for resistive mode feedback. The different dynamic characteristics that each path exhibits will be discussed here, along with the tradeoffs of each mode.

As a first approximation, when $Z_{R_f} \ll Z_{C_f}$, the impedance of the feedback capacitor may be ignored and the transfer function of the front end amplifier simplifies to

$$\frac{v_o}{v_{in}} = -\frac{R_f}{1/sC_{sen}} = -sR_fC_{sen}, \tag{4.12}$$

which is the same as the transfer function for an inverting differentiator. On the other hand, when $Z_{C_f} \ll Z_{R_f}$, the feedback resistance is insignificant and thus the transfer function of the front end amplifier becomes

$$\frac{v_o}{v_{in}} = -\frac{1/sC_f}{1/sC_{sen}} = -\frac{C_{sen}}{C_f}, \tag{4.13}$$

and the op-amp appears most similar to an inverting amplifier. The first major difference is the frequency dependence of the resistive mode feedback at signal frequencies. This frequency dependence must be accounted for when operating over a range of frequencies, as measurements at different frequencies will be scaled differently. The capacitive mode feedback does not suffer from this drawback; since the gain is the ratio of the capacitances, their frequency components are negated. The disadvantage to capacitive dominated feedback is that typically temperature coefficients and initial tolerances are much poorer for practical capacitors.

As shown in the discussion of the closed loop response of the front end amplifier, and equations (4.4) and (4.9), the former reprinted here

$$\frac{v_o}{v_{in}} = \frac{sR_fC_{sen}}{1 + sR_fC_f},$$

the product R_fC_f is responsible for the location of the system pole. It is important to keep the operating frequency in mind when considering the placement of this closed loop pole. The bode plot in Figure 4-5 illustrates the shape of the magnitude and phase plots for a general front end amplifier. At low frequencies the system is similar to the response of a differentiator. As the frequency increases and reaches the lower frequency knee of the magnitude plot, the response takes on the shape of the inverting op-amp. Realistically, because of the finite open loop gain, the magnitude rolls off due to the internal compensation of the op-amp. Operating at or around the pole is undesirable because of the dynamics of the region, i.e. the large frequency dependence of both gain and phase. Thus, for each feedback mode the operating frequency range should be well away from the pole contributed by the feedback network. There is also added benefit for placing the pole away from the very low

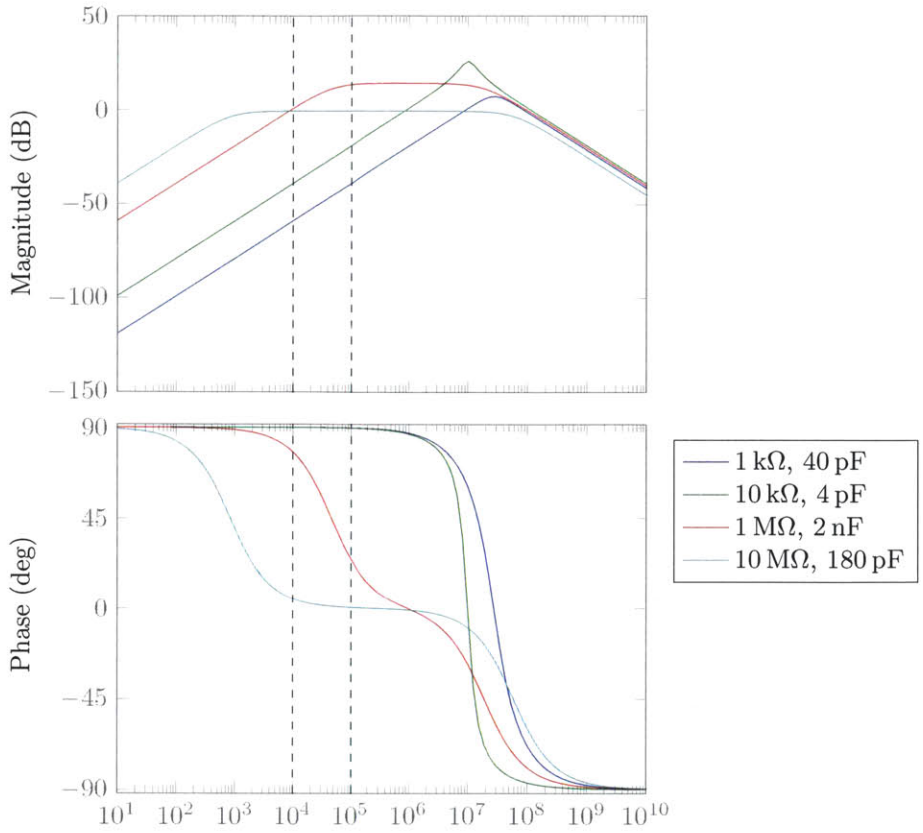


Figure 4-5: The bode plots for the front end amplifiers implemented, with finite open loop gain.

frequency range. The farther away the pole, the better the attenuation of 60 Hz pickup and other low frequency interference at the input.

The $R_f C_f$ time constants required to remain in one region of dominance straddle the pole. In other words, for resistive mode feedback the operating frequency region should be well before the pole and for the capacitive mode well after the pole. These operating regions are dictated by the requirements of the feedback mode to maintain impedance dominance. Figure 4-6 illustrates the required pole frequency for the implemented values of R_f as a function of C_f with horizontal lines marking the operating frequencies of the system. For the given resistances acceptable feedback capacitances for resistive dominated feedback lie above the operating frequencies; for the capacitive dominated feedback the opposite is true. This can also be inferred from the fact that capacitance is inversely proportional to impedance, i.e. smaller capacitances yield larger impedances which lead to resistor dominance in the feedback network.

The feedback network in the resistive mode demands small capacitances. In order to more accurately and precisely present small capacitances in the feedback path of the resistive mode a T-network is used, as previously depicted in Figure 4-1a. The effective capacitance seen wrapped

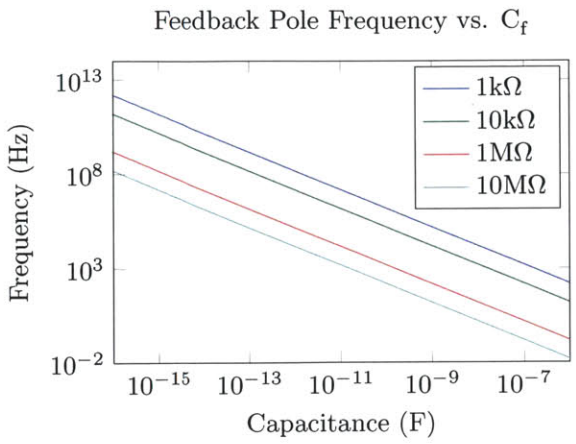


Figure 4-6: The frequency of the pole as a function of capacitance, plotted for the four resistor values implemented.

around the op-amp can be approximated by

$$C_{eff} = \frac{C_1 C_2}{(C_1 + C_2 + C_3)}, \tag{4.14}$$

by assuming the inverting input of the op-amp is a virtual ground. This result is confirmed by noting that without C_3, the expression results in the expected series combination of C_1 and C_2.

Tradeoffs

The relative merits of either feedback mode, and their respective operating regions, are largely influenced by the passive component properties, such as temperature coefficients, precision, and matching. More specifically, the effects these parameters have on the gain and phase responses of the front end amplifier determine the strengths and weaknesses of each feedback mode. Furthermore, the operating regions themselves respond differently to changes in these parameters.

The magnitude of the front end gain is affected by both the operating frequency and, more importantly, the ratio of the sensed capacitance and feedback capacitance. Gain errors in the resistive dominated feedback are primarily due to frequency instability of the signal source. This is because the mode operates on the ramp of the frequency response, before the $R_f C_f$ pole. This also means that since the operating frequency range is a decade wide, there will be a 20 dB difference between the gains at the low and high ends of the range. Moreover, the resistive mode does not operate at the largest gain possible because its operating point comes before the pole.

On the other hand, since the capacitive mode operates after the pole, the frequency response is essentially flat and thus not affected by a change in frequency. However, because the gain in this region is set by the feedback capacitor, any fluctuations in its capacitance are detrimental to the gain stability of the system. If the feedback capacitor is susceptible to thermal effects, then the gain

of the system will change as the feedback capacitance changes. By operating below the pole, the resistive mode avoids drastic gain changes due to capacitance fluctuation. A wandering capacitance can be translated into a wandering pole. If the pole is placed one decade away from one end of the operating frequency range, then a frequency shift toward the operating region will alter the gain and phase of the response.

In the end, the resistive mode feedback does not suffer greatly from gain errors due to capacitance fluctuations, but is more susceptible to gain errors due to inaccurate signal frequency. Conversely, the capacitive mode is less vulnerable to frequency dependent errors and rather is more likely to be affected by the feedback capacitor inaccuracies. Lastly, the capacitive mode has a larger gain than the resistive mode. In both modes however, the small feedback capacitances must be accurate and reliable.

4.1.2 Common Mode Feedback

As discussed earlier, moving the front end amplifiers onto the receive electrodes and off the sensor board itself, is desirable in order to limit stray capacitances, at the sensing node, and avoid erroneous pick up from the environment. The main sources of interference are 60 Hz pick up and noise from surrounding fluorescent lamp ballasts, which typically switch around 50 kHz. However, this is not possible without splitting up the fully differential front end amplifier, as used by previous sensor implementations.

The benefit of the fully differential front end is the common mode rejection, which attenuates common mode signals introduced by the environment. The effect of the common mode signals on the differential mode output of the front end, and throughout the signal chain, is what must be limited. Common mode output signals are less of a concern because the signal chain is fully differential all the way until the ADC.

A consequence of using active receive electrodes is the need to recover the common mode rejection performance at the front end. The answer to this issue is to include a common mode feedback path designed to drive the non-inverting inputs of the two front end amplifiers and null any common mode signals. A circuit diagram is presented in Figure 4-7 and the corresponding block diagram in Figure 4-8. For the purposes of this analysis, the sensed capacitance, C_s, is fixed and at a virtual ground.

The output of the electrodes is sent, via coaxial cable, to the sensor board where the common mode signal is taken from the midpoint of two resistors, shown here as R. The common mode signal is then passed through an op-amp with a topology similar to the front end. The output of this op-amp then drives the non-inverting input of the front end amplifier. Furthermore, as discussed in Section 4.1.1, the capacitance in the feedback network for the common mode amplifier is implemented with a T-network. For simplicity, the effective capacitance of the T-network, according

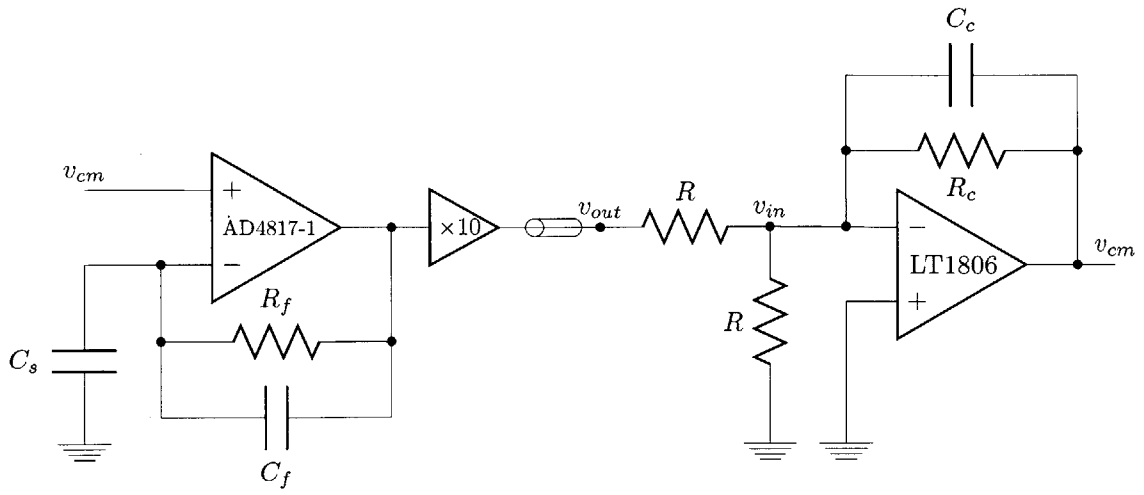


Figure 4-7: Schematic for the half circuit common mode feedback of the front end amplifier. Feedback capacitances represent lumped T-networks. Electrode circuitry is left of the cable and sensor board circuitry to the right.

to equation (4.14) is used here. Here, G is the transfer function of the common mode amplifier given by

$$\mathrm{G}(s) = \frac{v_{cm}(s)}{v_{in}(s)} = \frac{R_c}{\frac{R}{2}(sR_cC_c + 1)} \tag{4.15}$$

and H is the transfer function of the front end amplifier previously discussed in Section 4.1.1. Sources of interference, such as 60 Hz pick up and pick up from surrounding fluorescent lamp ballasts, are included at the input to the front end amplifier.

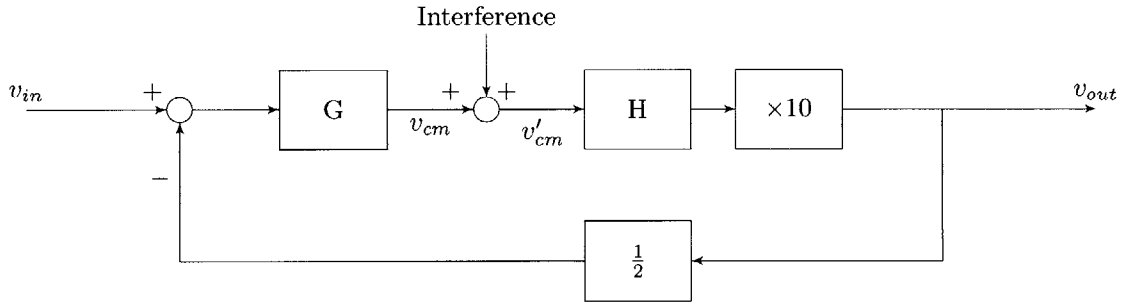
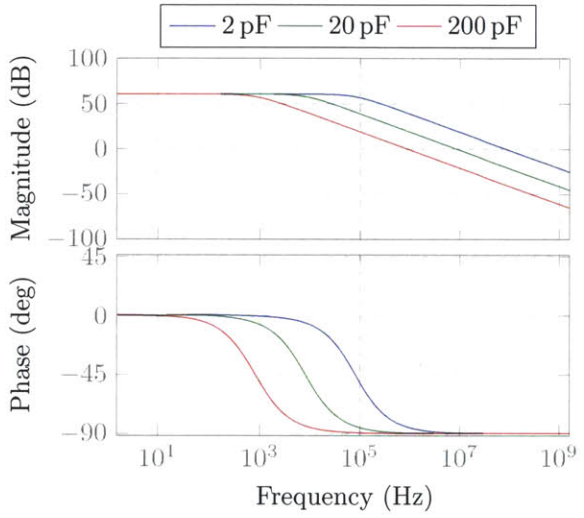


Figure 4-8: Block diagram for the common mode feedback of the front end amplifier.

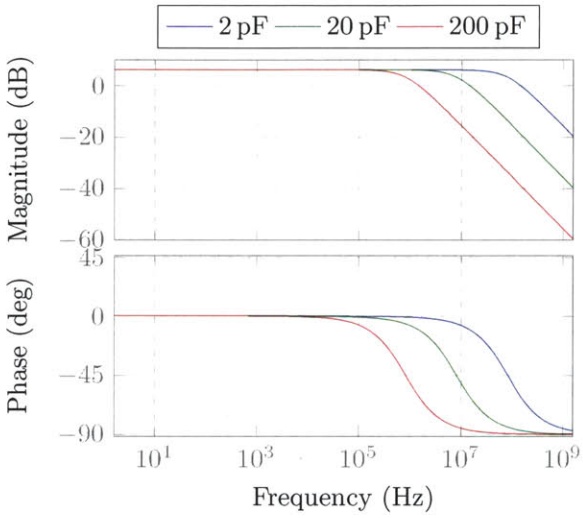
There are two features of the common mode feedback that are important to its performance. First, since the front end amplifier is driven with the feedback, it is important that the feedback amplifier not distort the sensed signal. Second, any external interference injected into the system must be well attenuated. The compensation capacitance, C_c, is selected for a given feedback network of the front end amplifier in order to maintain stability and good attenuation of common mode signals.

To illustrate the performance of the common mode feedback the front end amplifier is modeled

with a 10 MΩ resistor and 180 pF capacitor. Furthermore, the compensation feedback is modeled with a 1 MΩ resistor. Three values for the effective compensation capacitance are shown: 2 pF, 20 pF, and 200 pF. The compensation impedance, $Z_c = R_c||C_c$, sets the dominant pole. Thus, as Z_c increases the pole frequency decreases, which forces the common mode amplifier to cross over sooner. The loop transmission of the common mode feedback is shown in Figure 4-9a. It is evident from this plot that the system is stable with a phase margin of 90° even as the compensation capacitance increases. Though the stability is unaffected by increasing Z_c, the pole in the closed loop response begins to encroach on the range of operating frequencies. If the pole and operating frequency are too close, then the sensed signal may be attenuated and it may also incur some phase offset. As shown in Figure 4-9b, the phase contribution is more significant as the pole approaches the operating frequency range.



(a) The bode plot of the loop transmission of the common mode feedback for the front end amplifier for different values of Z_c. The system phase margin remains at 90° for each pole frequency.



(b) The bode plot of the closed loop response of the common mode feedback for the front end amplifier for different values of Z_c. The phase contribution increases as the pole approaches the highest operating frequency of 100 kHz.

The common mode rejection performance, shown via the closed loop response from v'_{cm} to v_{out}, is presented in Figure 4-9. Again, the plot depicts the responses for the same increasing values of Z_c. The most notable features here are the attenuations of 60 Hz pick up and 50 kHz pick up from fluorescent lamp ballasts. As the compensation capacitance decreases, the respective frequencies of the system poles and zeros increases. In turn as these frequencies increase, the frequency range of attenuated signals expands and so does the attentuation. This is particularly apparent at 50 kHz, where the smallest compensation capacitor yields the largest attenuation. A large common mode rejection is desirable for limiting the effect of common mode signals on the differential mode output of the front end amplifier.

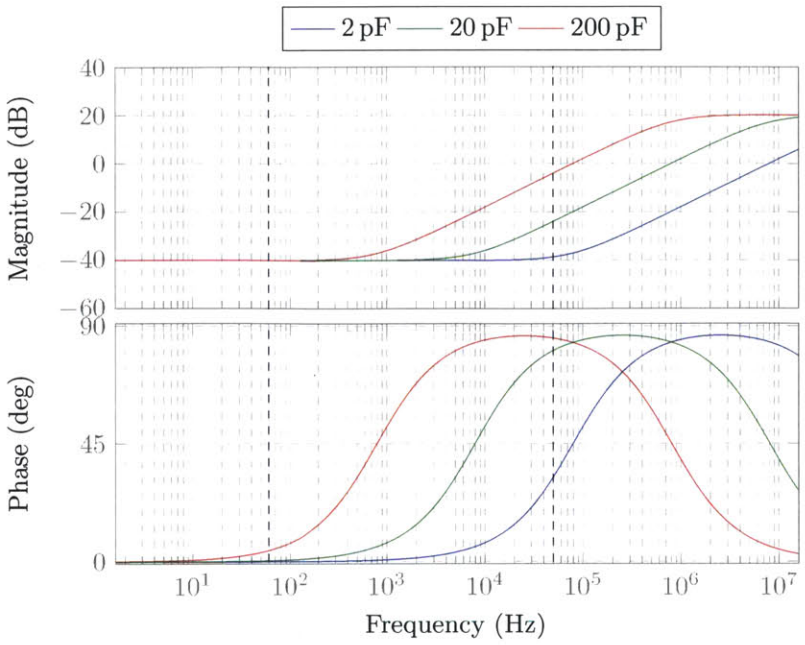


Figure 4-9: The bode plot of the closed loop transfer function for interference at the front end amplifier to the output. The largest expected interference contributors, at 60 Hz and 52 kHz, are noted by the dashed lines.

4.2 Signal Conditioning

The purpose of the signal conditioning circuitry is to optimally manage the sensed response from the output of the front end amplifier. The circuit consists of the following three blocks:

- Gain stages
- Synchronous detector and filters
- Instrumentation amplifier and ADC

The primary function of each stage and its adaptations are discussed in the following sections. A full system block diagram is depicted in Figure 4-10. The sensed signal is amplified as early as possible and then passed to the synchronous detector. This demodulates the signal, which is then low pass filtered. An instrumentation amplifier is then used to amplify and level shift the signal. The signal is again filtered and sent to an analog to digital converter (ADC) which is finally read by the microcontroller unit (MCU).

There are four identical copies of the signal conditioning circuitry beginning from the last gain stage and ending with the ADC. The purpose of this is primarily to allow for flexibility at the demodulation stage. Multiple independent channels allows for different simultaneous chopping frequencies and in phase and quadrature measurements of the same input signal. A final two copies of the signal chain are used for measuring the high voltage signal source for the purposes of normalizing the measurements by any drift present in the signal source.

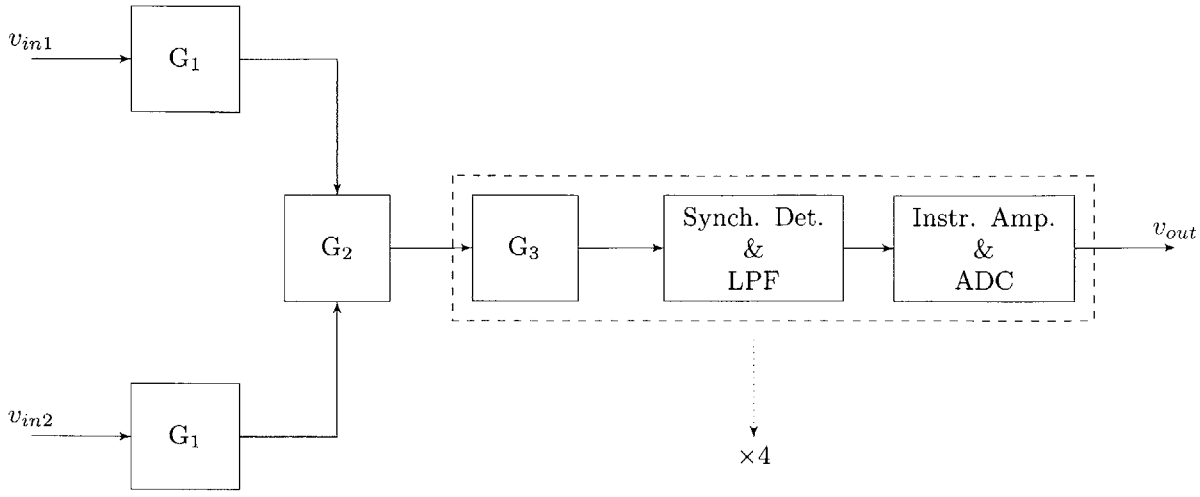


Figure 4-10: The generalized block diagram for the signal conditioning circuitry. The first gain stage is singled ended and located on each respective electrode. The following stages are all fully differential and reside on the sensor board. The dashed box is replicated four times.

Much of the general topology and design is adapted from previous implementations of the occupancy sensor [1]. The components used in the realization of the circuitry and measured performance characteristics are listed after each section.

4.2.1 Gain Stages

Three separate gain stages amplify and buffer the sensed signal from the output of the sensor front end. The first gain stage is located immediately after the front end amplifier on the receive electrodes. The op-amp is configured as a non-inverting amplifier. In addition to the gain, this stage is employed as a cable driver sending the sensed signal from the receive electrodes to the next stage of the signal conditioning electronics on the sensor board. The additional stage on the electrode helps isolate the sensing components from the rest of the circuit. A coaxial cable at the output of an op-amp appears as a capacitive load, as previously discussed in Section 3.1.5, and can add a pole to the transfer function of the amplifier. In order to maintain stability, this pole must be kept far from the unity gain frequency of the amplifier

$$f_p = \frac{1}{2\pi R_o C_L} \ll \frac{\text{GBW}}{A_{CL}} \tag{4.16}$$

where GBW is the gain bandwidth product of the amplifier, A_{CL} is the closed loop gain, R_o is the output resistance of the amplifier, and C_L is the capacitive load at the output.

The next two gain stages are placed on the sensor board with the rest of the downstream signal conditioning electronics. Unlike the stage on the electrodes, these two stages are fully differential gain stages. The incoming signals from each electrode comprise the positive and negative inputs to the second stage. The third gain stage marks the beginning of the replicated chain. Without a third

op-amp at the beginning of each replicated channel, the inputs to the four multipliers may present possibly interacting loads to the previous single op-amp second stage.

The pair of signal chains used to measure the high voltage source are structured slightly differently. The measurement of the signal source is singled ended as opposed to differential and is measured directly from a voltage divider at the output. This means there is no electrode necessary, and thus no first stage. Furthermore, in order to utilize the same path the fully differential second gain stage is configured according to Figure 4-11, which is adapted from the component application note [23].

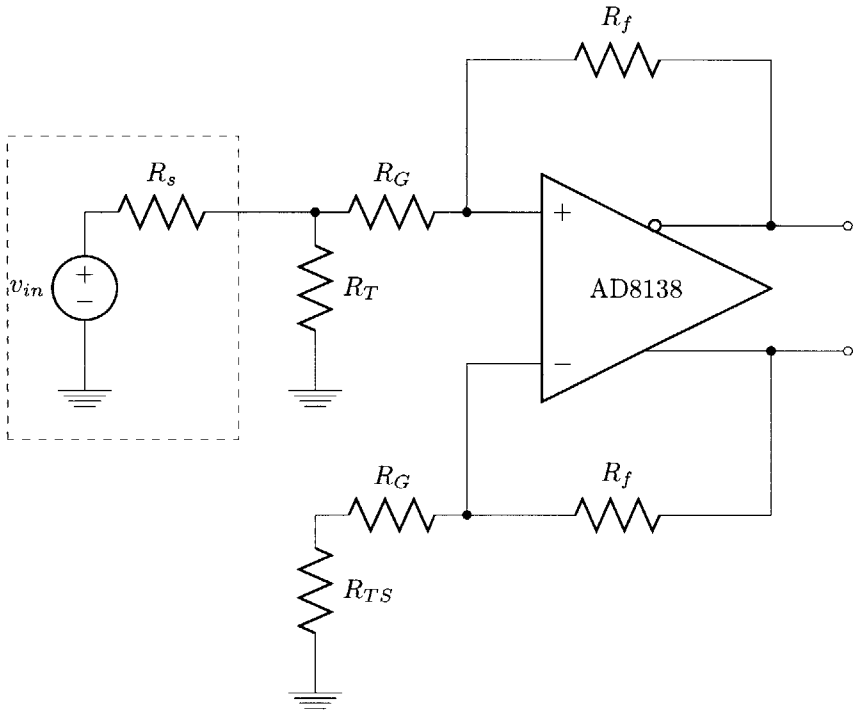


Figure 4-11: A singled ended input to differential output op-amp configuration.

4.2.2 Synchronous Detector and Filters

The synchronous detection used in the standalone sensor is carried over from previous implementations of the sensing circuitry. Explicit details regarding the principles involved have been thoroughly discussed as it pertains to the signal conditioning circuitry in this type of system, but will be reviewed more generally here [1, 6]. The multiplier and filter topology are presented in Figure 4-12.

The basic concept involves a high frequency carrier signal modulated by a low frequency signal of interest. The demodulation is then performed at the carrier frequency via a signal derived from the initial synthesized carrier signal. The result can then be filtered to recover the signal of interest. The implementation for the standalone sensor diverges from previous versions by measuring both the in phase and quadrature components of the modulated signal. To do so, the input signal is

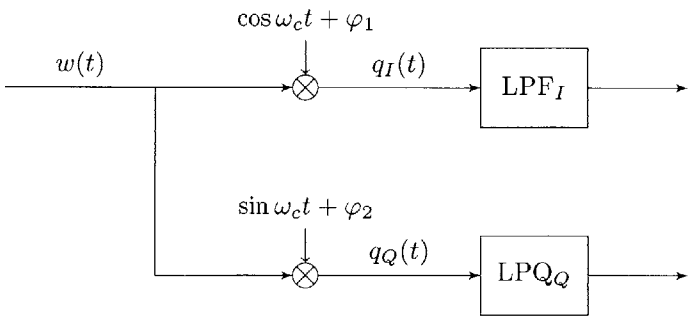


Figure 4-12: Block diagram of the synchronous detector and low pass filters.

multiplied separately by a signal in phase and a signal 90° out of phase. This reduces the impact of any phase error between the input signal and the demodulating signals. Furthermore, this method provides more, useful information about the environment. The signal conditioning circuitry is split into pairs of channels, the first channel of the pair performs in phase demodulation and the second channel performs the quadrature demodulation.

Synchronous Detection Analysis

A more in depth analysis begins by expressing the input to the multiplier as:

$$w(t) = A\,m(t)\cos(\omega_c t + \varphi) \tag{4.17}$$

or the time domain product of the carrier signal, $Acos(\omega_c t + \varphi)$, and the modulation function, $m(t)$, where φ is the phase of the signal at the input of the synchronous detector. The modulation function is what is to be recovered from the input and represents the sensed fluctuations in capacitance, and thus target movement and presence. In order to do so, the input is then multiplied by a signal at the carrier frequency. The in phase and quadrature signals can then be represented by

$$q_I(t) = w(t)\cdot\cos(\omega_c t + \varphi_1) = A\,m(t)\cos(\omega_c t + \varphi)\cdot\cos(\omega_c t + \varphi_1) \tag{4.18a}$$

$$q_Q(t) = w(t)\cdot\sin(\omega_c t + \varphi_2) = A\,m(t)\cos(\omega_c t + \varphi)\cdot\sin(\omega_c t + \varphi_2), \tag{4.18b}$$

where φ_x, the phase of the derived carrier signal, may not necessarily be equal to the phase φ of the input signal. The next step in the process requires two product-to-sum identities:

$$\cos\theta\cdot\cos\varphi = \frac{1}{2}[\cos(\theta-\varphi)+\cos(\theta+\varphi)] \tag{4.19a}$$

$$\cos\theta\cdot\sin\varphi = \frac{1}{2}[\sin(\theta+\varphi)-\cos(\theta-\varphi)], \tag{4.19b}$$

in order to reduce (4.18) into the following forms:

$$q_I(t) = \frac{A\,m(t)}{2}\Big[\cos(\varphi - \varphi_1) + \cos(2\omega_c t + \varphi + \varphi_1)\Big] \tag{4.20a}$$

$$q_Q(t) = \frac{A\,m(t)}{2}\Big[\sin(2\omega_c t + \varphi + \varphi_2) - \cos(\varphi - \varphi_2)\Big]. \tag{4.20b}$$

Previous implementations, that do not obtain $q_Q(t)$, stop at this point and pass $q_I(t)$ down the signal chain where the higher frequency copy of the signal, at $2\omega_c t$, is filtered out. The standalone sensor circuitry follows suit, but sends both $q_I(t)$ and $q_Q(t)$ through the low pass filters. The filtered signals from (4.20) may be combined by finding the magnitude of the two signals:

$$r_{IQ}(t) = \sqrt{r_I(t)^2 + r_Q(t)^2} = \sqrt{\left[\frac{A\,m(t)}{2}\cos(\varphi - \varphi_1)\right]^2 + \left[-\frac{A\,m(t)}{2}\sin(\varphi - \varphi_2)\right]^2}. \tag{4.21}$$

Using another trigonometric identity:

$$\sin^2\theta + \cos^2\theta = 1, \tag{4.22}$$

the result can be further simplified:

$$\begin{aligned} r_{IQ}(t) &= \sqrt{\left(\frac{A\,m(t)}{2}\right)^2\Big(\cos^2(\varphi - \varphi_1) + \sin^2(\varphi - \varphi_2)\Big)} \\ &= \frac{A\,m(t)}{2}\sqrt{\sin^2(\varphi - \varphi_2) + \cos^2(\varphi - \varphi_1)}. \end{aligned} \tag{4.23}$$

The eventual output of the multiplier and filters is the recovered sensed signal, $m(t)$, multiplied by a scaling factor dependent on any potential phase errors.

Implementation

The synchronous detector is realized with a set of four analog switches, shown in Figure 4-13, configured in a full bridge topology and driven by the in phase, s, and quadrature, $\bar{s}$, clock signals to form a square wave multiplier.

Square wave multiplication also necessitates the use of low pass filtering. Since a square wave can be defined as the infinite series of its Fourier series terms:

$$\frac{4}{\pi}\sum_{n=1,3,5,\ldots}^{\infty}\frac{1}{n}\sin(n\,\omega_c t), \tag{4.24}$$

harmonics of the square wave create higher frequency replicas of the modulated signal. These

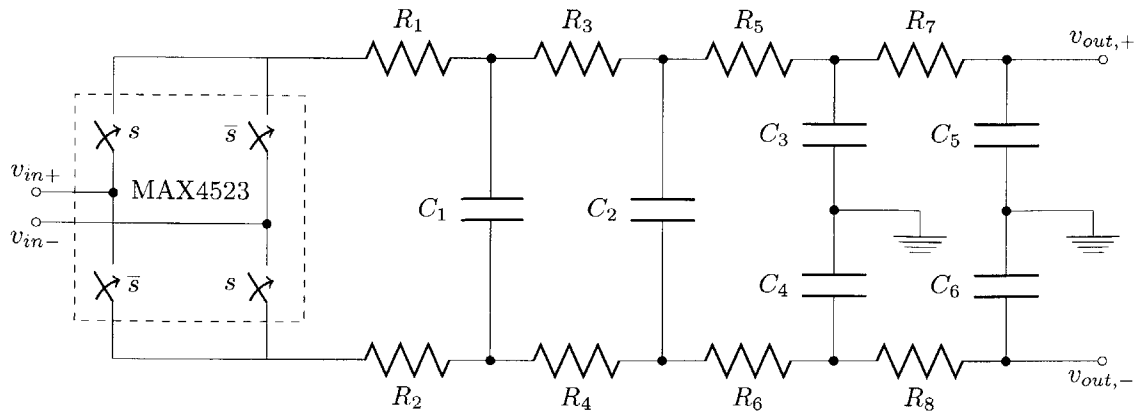


Figure 4-13: Circuit implementation of the synchronous detector and low pass filters.

replicas, along with the replicas at $2\omega_c$, will be removed by the low pass filters.

The passive low pass filters implemented after the analog switches contain two fully differential stages followed by two singled ended stages. This topology comprises a four pole RC ladder with two repeated poles. The goal is to attenuate all signals outside of the low frequency window where the signal of interest is located. The filter poles must be placed far from the operating frequency in order to limit any unwanted attenuation and phase offset.

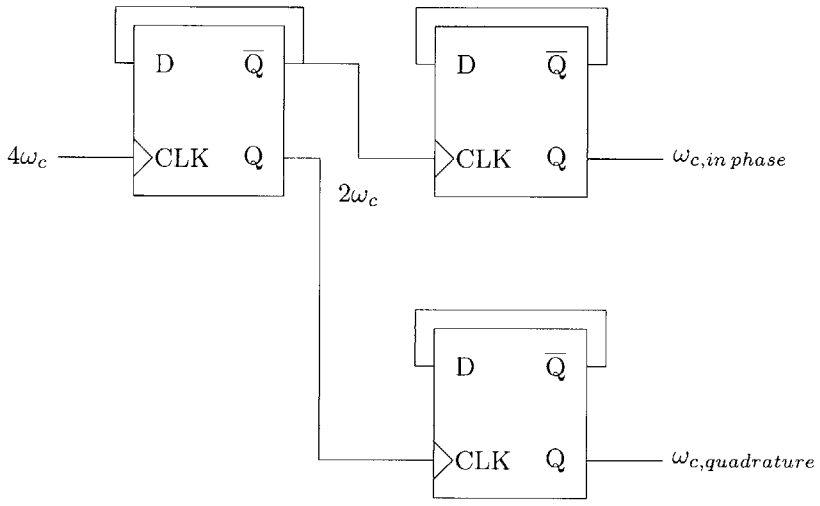


Figure 4-14: Circuit implementation of the clock signal synthesis. The flips flop depicted here are implemented with the 74HVC74.

The clock signal generation is presented in Figure 4-14. The pair of clock signals for one pair of channels are generated from the same pulse width modulation (PWM) pin on the system MCU at a frequency $4\omega_c$ and a 50% duty cycle. The clock signals are then sent through three D flip flops configured as divide by two counters; thus, the outputs are brought back down to the appropriate carrier frequency ω_c. The Q and $\overline{Q}$ of the first flip flop each feed the clock inputs of two separate flip flops. Since the outputs are triggered on rising edges, this configuration provides the required

90° phase shift for the multiplier and reduces the reliance on a precise 50% duty cycle input.

4.2.3 Instrumentation Amplifier and ADC

An instrumentation amplifier follows the low pass filters. Its purpose is to convert the differential signal to a single ended signal and level shift the output. The benefit of using an instrumentation amplifier is primarily for its low noise and low offset voltage characteristics. In this case, an auto-zeroing amplifier, the AD8230, is utilized for its very low offset voltage performance. This topology consists of a two stage, fully differential design with sample and hold circuitry for both stages. Figure 4-15 depicts the method used for level shifting the output as outlined in the component datasheet [19].

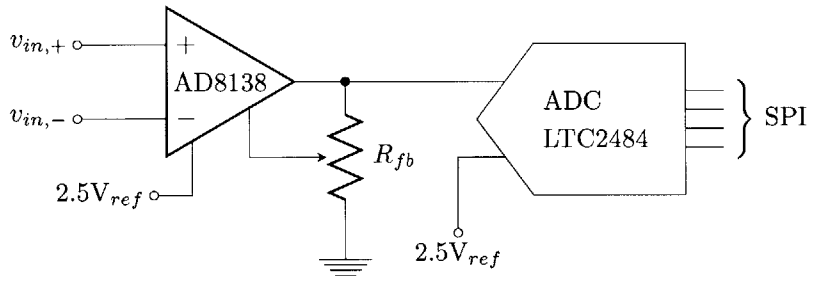


Figure 4-15: Circuit diagram of the instrumentation amp., with gain and level shifting, followed by the ADC.

The offset voltage of the amplifier is of particular concern because of its effect on the maximum voltage swing of the output signal:

$$v_{swing} = 2\,(v_{fs} - |v_{offset}|), \tag{4.25}$$

where v_{fs} is the full scale voltage of the system and v_{offset} is the output offset of the amplifier. Given a rail to rail amplifier, such as the AD8230, ideally the output voltage swing makes the most of the amplifier capabilities and truly spans from rail to rail. With some voltage offset on top of the output signal, the output voltage swing will never reach its full potential. As a result, the dynamic range of the system can suffer as well. The dynamic range of the system can be defined as:

$$\mathrm{DR} = \frac{v_{fs}}{v_{noise}}, \tag{4.26}$$

or the ratio of the full scale range and the noise floor of the system. Optimally, the gain of the instrumentation amplifier is set such that the noise floor at the input of the ADC is no larger than the least significant bit (LSB) of the ADC. The LSB is the result of dividing the full scale voltage

by the most significant bit (MSB)

$$v_{noise}\Big|_{G_{opt}} = \frac{v_{fs}}{\text{MSB}}. \tag{4.27}$$

In combining these results, the dynamic range of the system becomes:

$$\begin{aligned} \text{DR} &= \frac{v_{swing}/2}{v_{noise}} \\ &= \frac{v_{fs} - |v_{offset}|}{v_{noise}} \\ &= \text{MSB} \cdot \left(\frac{v_{fs} - |v_{offset}|}{v_{fs}} \right). \end{aligned} \tag{4.28}$$

The amount of offset voltage at the output of the amplifier must be limited in order to achieve the largest dynamic range possible. It also goes without saying that limiting the noise in the system is also beneficial to not only preserving dynamic range, but to improving the overall performance. Large gains on the instrumentation amplifier may needlessly amplify the noise along with the signal itself.

The final stage of the signal conditioning chain is the analog to digital converter. The LTC2484 24 bit sigma-delta ADC is used to convert the analog sensed signal at the output of the signal conditioning chain. Level shifting the output of the instrumentation amplifier is a necessary step in order to use as much of the system full scale range. Interfacing with a unipolar ADC requires that some DC offset be applied to the AC signal, such that the input to the ADC is non-negative. The logical level is the midscale value of the full operating range of the ADC; this level maximizes the possible voltage swing and allows for equal positive and negative swings.

4.3 Noise, Drift, and Offset Analysis

Errors in the form of noise, drift, and offset will be analyzed for each stage of the standalone sensor circuitry. Noise is represented by probability density functions, most often of a Gaussian distribution. This means that 99.7% of the instantaneous noise in a system will be contained within 3σ of the mean value of the noise floor of the system. The noise floor is defined as the root mean square (RMS) noise with no inputs present. With this in mind, the contributions to the cumulative noise present in the entire system can be analyzed. Accounting for noise is not so simple as summing the incurred noise from stage to stage. Rather, the square root of the sum of the squares of all of the independent noise sources must be found.

The RMS voltage noise for a resistor is

$$v_n = \sqrt{4k_BTR\Delta f} \quad \text{V}/\sqrt{\text{Hz}}, \tag{4.29}$$

and the RMS current noise can be found by dividing through by the resistance R

$$i_n = \sqrt{\frac{4k_BT\Delta f}{R}} \quad \text{A}/\sqrt{\text{Hz}}. \tag{4.30}$$

Here k_B is Boltzmann's constant and T is the circuit temperature measured in Kelvin. These definitions are useful when analyzing the noise performance of the circuitry for the sensor, specifically the contributions from resistors in feedback paths, filters, and other settings. Typical op-amp noise sources are twofold. They can be represented as the combination of an input voltage noise in series with one input and input current noise sources from both inputs to ground, as depicted in Figure 4-16.

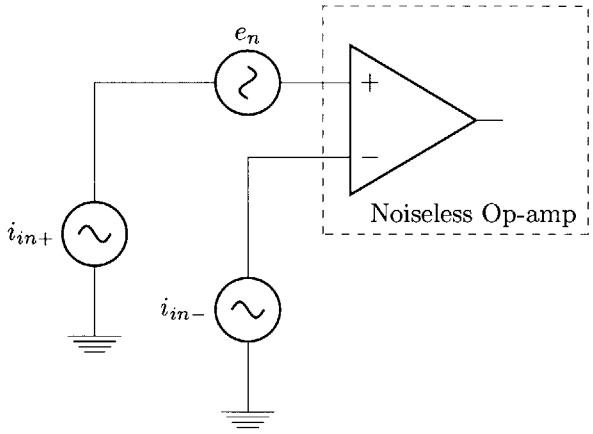


Figure 4-16: An op-amp with voltage and current noise sources included.

Characteristics of conventional voltage and current sources, i.e. linearity and independence, apply when analyzing the contributions of each noise source. Thus superposition can be used to simplify noise analyses by opening or shorting appropriate sources, investigating the contributions of each source one at a time, and finally finding the square root of the sum of the squares of each individual noise source to find the total circuit noise.

4.3.1 Sensor Front End

Gain and Phase Errors

The feedback network is affected most by the uncertainties and inaccuracies of the capacitor. Resistors are generally more precise and available with lower temperature coefficients. Recall that the closed loop transfer function of the front end is

$$\frac{v_{out}}{v_{in}} = \frac{sR_fC_{sen}}{1 + sR_fC_f}. \tag{4.31}$$

The frequency of the pole in the front end transfer function can be shown to be dominated by capacitance error, given by

$$\frac{\delta\tau}{\tau} = \sqrt{\left(\frac{\delta R_f}{R_f}\right)^2 + \left(\frac{\delta C_f}{C_f}\right)^2}, \tag{4.32}$$

where τ is the time constant of the $R_f C_f$ pole and the delta, δ, of the values is the precision or error. Assuming, 1% precision resistors and 5% precision capacitors, the frequency error becomes

$$\frac{\delta\tau}{\tau} = \sqrt{(.01)^2 + (.05)^2} = 5.099\%. \tag{4.33}$$

Therefore, the precision of the resistor is dwarfed by that of the capacitor and may be ignored. Based on the closed loop transfer function of the front end, and by taking the derivative of the gain and phase and multiplying by the error in the time constant of the feedback impedance, the gain error can be found to be

$$\begin{aligned} \delta \mathrm{G} &= \left|\frac{d\mathrm{G}}{d\tau}\right| \cdot \delta\tau \\ &= \frac{\omega^2 R_f C_{sen} \tau}{\left[(\omega\tau)^2 + 1\right]^{3/2}} \cdot \delta\tau, \end{aligned} \tag{4.34}$$

and the phase error to be

$$\begin{aligned} \delta \mathrm{P} &= \left|\frac{d\mathrm{P}}{d\tau}\right| \cdot \delta\tau \\ &= \frac{\delta\tau}{(\omega\tau)^2 + 1}. \end{aligned} \tag{4.35}$$

Front End Amplifier Noise

The noise model for the front end amplifier is shown in Figure 4-17. Superposition of the noise sources may be used to analyze each individual contribution to the total output noise. This total is the sum of the squared magnitude of each transfer function for a particular noise source:

$$\overline{v_{no}^2} = \sum_i |a_i(s)|^2 \overline{v_{ni}^2} \tag{4.36}$$

Due to the synchronous detection employed in the system, the bandwidth of the circuit is narrow. Thus, the transfer functions can be assumed to be constant over the bandwidth and any frequency dependency ignored. This also means that the total mean squared noise at the output is just multiplied by the circuit bandwidth.

From the model, it can be shown that the input voltage noise is amplified by the gain of the

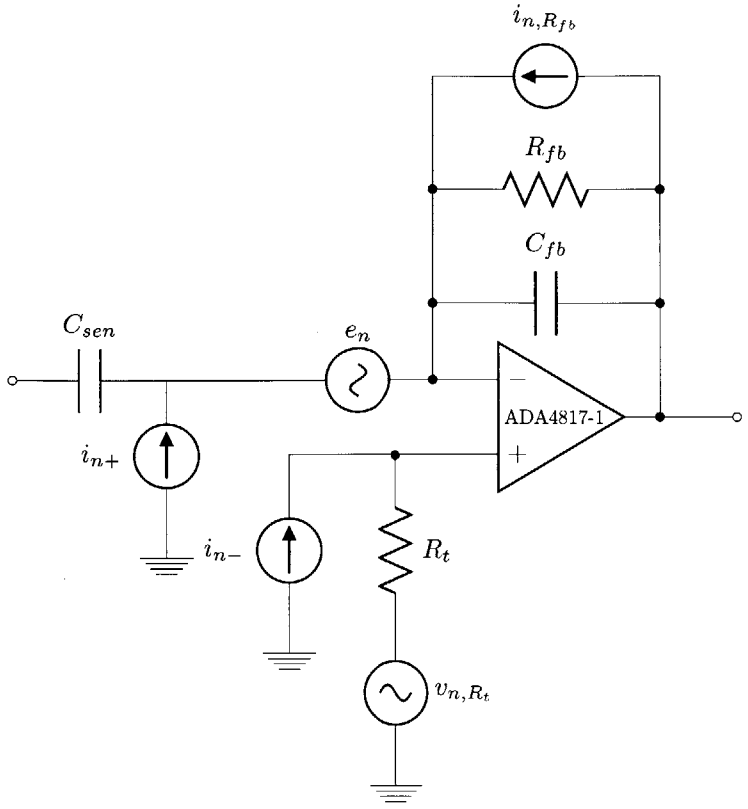


Figure 4-17: Schematic of the front end amplifier noise model.

front end and hence the transfer function becomes

$$\alpha_{e_n} = \frac{Z_f}{Z_{in}} \tag{4.37}$$

Next, the input current noise at the inverting terminal of the front end amplifier is converted to a voltage by the feedback impedance

$$\alpha_{i_-} = Z_f. \tag{4.38}$$

Since the non-inverting terminal of the op-amp is driven to the common mode of the measured signal, it will see some impedance due to the terminating resistors and cable impedance. For the operating frequency range of the standalone system, and the expected cable capacitance, $R_t \ll 1/sC_{cable}$, the impedance seen at the non-inverting input becomes

$$Z_+ = \frac{1}{sC_{cable}} \Big|\Big| R_t \approx R_t, \tag{4.39}$$

which is then amplified by the gain of the front end amplifier

$$\alpha_{i_+} = Z_+ \cdot \left(1 + \frac{Z_f}{Z_{in}}\right). \tag{4.40}$$

The termination resistor will also add its own noise, given by

$$\alpha_{R_t} = \left(1 + \frac{Z_f}{Z_{in}}\right). \tag{4.41}$$

Lastly, the voltage noise of the feedback network is the product of the current noise of the feedback resistor and the feedback impedance and is seen in its entirety at the output

$$\alpha_{fb} = 1 \tag{4.42}$$

The combined mean squared output voltage noise density is then

$$\begin{aligned}\overline{v_{no}^2} &= \alpha_{e_n}^2 e_n^2 + \alpha_{i_-}^2 i_{n-}^2 + \alpha_{i_+}^2 i_{n+}^2 + \alpha_{R_t} v_{n,R_t}^2 + \alpha_{fb} i_{n,fb}^2 \\ &= e_n^2\left(\frac{Z_f}{Z_{in}}\right)^2 + i_{n-}^2 Z_f^2 + i_{n+}^2 R_t^2\left(1 + \frac{Z_f}{Z_{in}}\right)^2 + (4kTR_t)^2\left(1 + \frac{Z_f}{Z_{in}}\right)^2 + \left(\frac{4kT}{R_f}\right)^2 Z_f^2 \quad \mathrm{V^2/Hz}.\end{aligned} \tag{4.43}$$

Equation (4.43) may be referred to the input (RTI) by dividing through by the front end gain, Z_f/Z_{in}. Evaluating the RTI noise shows that the front end attenuates all signals for which $Z_{in} > Z_f$, i.e. signals lower in frequency than the $R_f C_f$ pole. This suggests that the op-amp used to implement the front end amplifier must have suitably low current noise, as it is not mitigated by the front end configuration and is instead a function of the input impedance. Furthermore, the output noise can be lowered by increasing the resistance in the feedback network.

4.3.2 Analog Switches and Filters

Recall that φ_1 and φ_2 are the phase errors between the input signal and the in phase and quadrature clock signals. Given the result in (4.23), the error can be observed from:

$$r_{IQ}(t) = \frac{A\, m(t)}{2}\sqrt{\sin^2(\varphi) + \cos^2(\varphi - \varphi_{err})}, \tag{4.44}$$

where $\varphi_{err} = \varphi_1 - \varphi_2$. There is no φ_{err} for which the root of the sum of the squares is less than 1. Therefore, any phase error incurred between the input and clock signals does not attenuate the recoverable signal. Though there is still potential for some phase error, the total possible error is reduced by more than half compared to the in phase demodulation only method where the output is:

$$V_{out} = \cos(\omega_c t)\cos(\omega_c t + \varphi_{err}). \tag{4.45}$$

When including both in phase and quadrature signals, the output response never drops below one as φ_{err} increases. Though phase errors may still affect the output response, their significance is

diminished.

Another consideration is the possibility that the analog switches incur a voltage offset due to charge injection. Charge injection is a result of mismatched parasitic capacitances inside an op-amp. The effects of charge injection typically show up as glitches at the output. The faster the switching or smaller the load capacitance the worse these glitches will be.

The low pass filters may also exhibit what is known as kTC noise. The voltage noise contribution of a single low pass RC filter is

$$\overline{v_n^2} = \frac{kT}{C}. \tag{4.46}$$

This means that the resistance in the filter does not contribute to the noise profile of the stage. This does not absolve the filter resistors from all responsibility. If the resistors are not matched on either side of the fully differential filter, then offsets may be applied to the output of the filter. As with any other stage, additional offsets reduce the dynamic range of the system.

4.3.3 Instrumentation Amplifier and ADC

The instrumentation amplifier acts as a gain stage and a level shifting stage. As such it is prone to errors from both applications. The AD8138 has a particularly difficult time applying large gains with accuracy and precision because of a large gain drift. The datasheet for the LTC2484 analog to digital converter specify a 600 nV RMS output noise [16]. Given a 5 V rail, this specification is below 22 bits of accuracy and is considered insignificant.

4.4 Implementation and Performance

The design criteria discussed previously were used as a basis for selecting the components used in the actualized system.

4.4.1 Component Selection

On Plate Electronics

Some experiments required different pairs of receive electrodes with differing impedances in the feedback path of the front end amplifiers. The aim of these experiments was to empirically evaluate the tradeoffs between gain and noise in the first stage. The pairs of components used for each set of plates is listed in Table 5.1. The capacitances listed for the resistive modes are the lumped, effective feedback capacitances. Per the previous discussion, the pole of each pair is placed on the appropriate side of the system operating frequencies.

The required ± 5 V rails are supplied to the active receive electrodes via twisted 30 AWG wire. All signal connections, to and from the electrodes, are made over RG-178 via MMCX style connectors.

Table 4.1: Front End Amplifier Combinations

Plates	Mode	R_f	C_f	Pole
Pair 1	Capacitive	10 MΩ	180 pF	88.4 Hz
Pair 2	Capacitive	1 MΩ	2 nF	79.6 Hz
Pair 3	Resistive	10 kΩ	4.7 pF	3.39 MHz
Pair 4	Resistive	1 kΩ	40 pF	3.98 MHz

An LT1806 amplifies the sensed signal and drives the cable out of the receive electrodes to the sensor board and signal conditioning circuitry. The gain of this amplifier is set by the feedback network to be 10. With a gain bandwidth product of 325 MHz, the LT1806 is more than capable of driving the cable at the operating frequencies used in the system.

Signal Conditioning Electronics

The signal conditioning circuitry is implemented based on the design in Section 4.2 and the schematics for the gain stages, multiplier and filter chain, instrumentation amplifier, and ADC are presented again here. Values of the gain setting resistors were minimized in order to reduce additional noise. The two gain stages at the beginning of the signal path are unity gain, set by $R_{fx} = R_{gx}$, in order to avoid saturating the op-amp output. The following filters have cutoffs, 160 Hz and 1.6 kHz, chosen to appropriately filter two decades below both 10 kHz and 100 kHz operating signal frequencies. The AD8230 instrumentation amplifier follows and its gain is set to 2 to minimize the effects of gain drift inherent to the component. The analog to digital convert used is the LTC2484 part, which is designed to not require additional drivers. Another set of low pass filters are connected to the inputs of the ADC per the recommendations in the component datasheet [16].

4.4.2 Noise and Drift

The cumulative measured noise of the system, as previously analyzed in Section 4.3, are presented in Table 4.2. These measurements profile the total measured noise at the ADC. The experiments were performed with the entire system functioning and thus are an accurate representation of the total noise expected at the input to the ADC. These tests were used as controls for the system experiments, described in Chapter 5, and thus were performed for each front end amplifier configuration. The RMS voltage noise was taken from an approximately 10 second window. Table 4.2 presents the input referred current noise.

Based on the same long term data collection used to gather the system noise profile, the drift of the entire system was also characterized. The results of the tests presented in Table 4.3 are difference between the initial and final measured value over a 10 minute test period.

Table 4.2: Measured Noise Referred to Input

Front End Pair	RTI Noise at 10 kHz	RTI Noise at 100 kHz
10 MΩ,180 pF	0.14 nA_{RMS}	1.02 nA_{RMS}
1 MΩ,2 nF	0.88 nA_{RMS}	3.75 nA_{RMS}
10 kΩ,4.7 pF	0.81 nA_{RMS}	0.93 nA_{RMS}
1 kΩ, 40 pF	6.01 nA_{RMS}	3.07 nA_{RMS}

Table 4.3: Measured System Drift

Front End Pair	Drift at 10 kHz	Drift at 100 kHz
10 MΩ,180 pF	347 µV	−3 µV
1 MΩ,2 nF	27 µV	−106 µV
10 kΩ,4.7 pF	−5 µV	−25 µV
1 kΩ, 40 pF	16 µV	4 µV

4.5 Summary of System Tradeoffs

The two feedback modes that can be implemented at the front end amplifier each have their strengths and weaknesses. The resistive mode sacrifices gain for a lesser dependence on the accuracy of the feedback capacitor by operating before the closed loop pole. The capacitive mode uses the maximum gain set by the feedback capacitor. Furthermore, the feedback resistance is the dominant factor in the noise considerations of the front end amplifier. The resistive mode will contribute more noise content than the capacitive mode because it demands a lower resistance to dominate the feedback current path.

Next, gain distribution and application balances the signal to noise ratio of the the measured signal. Early stage gain is helps to amplify the signal as much as possible before additional noise is accrued. Doing so limits the gain applied to noise accumulated in later stages.

The instrumentation amplifier is valuable for providing a low offset output. Generally however, the characteristics of available instrumentation amplifiers emphasizing select qualities over others. In this sense, low output offset may come at the expense of higher gain error or output noise. The AD8230 is an example of one such instrumentation amp.

Chapter 5

Occupancy Detection Demonstration and Performance

This chapter will present and discuss the performance of the standalone sensor system as an occupancy sensor. The following sections cover the setups, conditions, and the results of each test. These tests show that an optimal system has a maximum range of 11 feet and can sense capacitance changes down to the attofarad level. Furthermore, the tests show that the system is not limited by noise or signal integrity, but rather by physical constraints.

5.1 System Configuration

The standalone sensor system configuration is largely similar to previous sensor implementations integrated into fluorescent lamps. Depending on the configuration, there may be three or four electrodes of interest: one or two source electrodes and two active receive electrodes. Thus, the system may be driven single ended or differentially, but sensing is always done differentially. There are three parameters of interest regarding the placement of the electrodes:

- Distance apart (separation)
- Coplanarity
- Height off of ground

The impact of the geometry of the electrodes, specifically that of these three characteristics, is explored in this chapter.

The source signal can be derived from any source of periodic waveforms and can be single ended or differential. A system configured with the singled ended driver will rely on stray capacitive coupling, while a differentially driven system naturally presents an explicit return path.

The functional performance of the system is highly dependent on the performance of the high voltage amplifier driving the source electrodes. The signal source has a practical operating range of tens to hundreds of volts and tens of hundreds of kilohertz. The system is capable of being pushed even further, to a thousand or more volts in amplitude and a signal frequency up to megahertz, but the utility and benefit of these extremes is not investigated here.

5.1.1 Experimental Setups

A photograph of the experimental setup used to explore the capabilities of the single ended system is shown in Figure 5-1.

Figure 5-1: Front left view of the standalone sensor setup. The source electrode is located on the highest platform and protected by electrical tape. The receive electrodes are on either side of the source electrode and the sensor board sits between them.

The setup consists of arrangeable platforms, for easy reconfiguration of the source and measurement electrodes, to satisfy the placement requirements for different system configurations. The signal source is the high voltage amplifier described in Chapter 3. The frequency response of the transformers used in the high voltage amplifiers and the noise performance of the amplifier is also discussed in Chapter 3. Data from the sensor board was taken using a combination of MATLAB®scripts and C code running on an Atmel ATmega32U4. The full code used for the experiments can be found in Appendix A.

Two power supplies are used to power the whole system, one for the high voltage amplifier and another for the active electrodes and signal conditioning board. Limiting the number of independent power supplies reduces the possibility of independent noise sources in the system. With as much

circuitry as possible on the same power supply, any ripple or noise on the rails of the system is seen equally by each device. The high voltage amplifier requires its own power supply due to its higher voltage demands. Bypass capacitors are used generously and every chip is bypassed on both rails.

5.2 Occupancy Detection

5.2.1 Range Tests

A variety of range tests were performed to measure the detection radius of the standalone sensor. The responses presented here are the magnitude of the signals measured. An example of the individual in phase and quadrature measurements is shown in Figure 5-2. The limit of the detection radius is defined by the measured noise floor of the system. One, two, and three sigmas are presented on the relevant plots to determine the edge of detection.

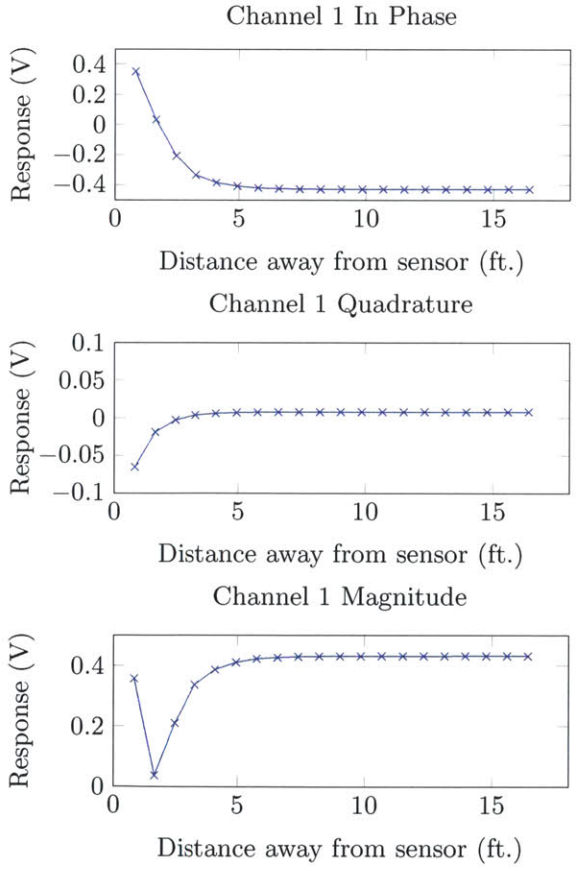


Figure 5-2: Representative in phase and quadrature contributions to the total magnitude measured from the sensor board.

Two types of range tests were performed. The first range test, the front to back test, begins with the subject standing 0.25 m away from the sensor setup, then taking successive 0.25 m steps away

from the sensor. This test comprises 20 datapoints, for a total distance of 5 m. For consistency, the backward steps are performed in a line at the same position for each test, typically the right most receive electrode when facing the setup. This position is away from the center line of the differential measurement and is a peak in the left to right response. The goal of this test is to determine the maximum detection radius for a given setup.

The second test, the left to right test, is a similarly stepped process. The difference for this test is that the subject steps across the face of the sensor at a fixed distance. The step size is about equal to the front to back test, but since the experiment area is rectangular there are fewer data points. Left to right tests typically span the full range of the testing area. When performed within a the detection radius of a given configuration, this test is useful for comparing relative signal to noise ratios.

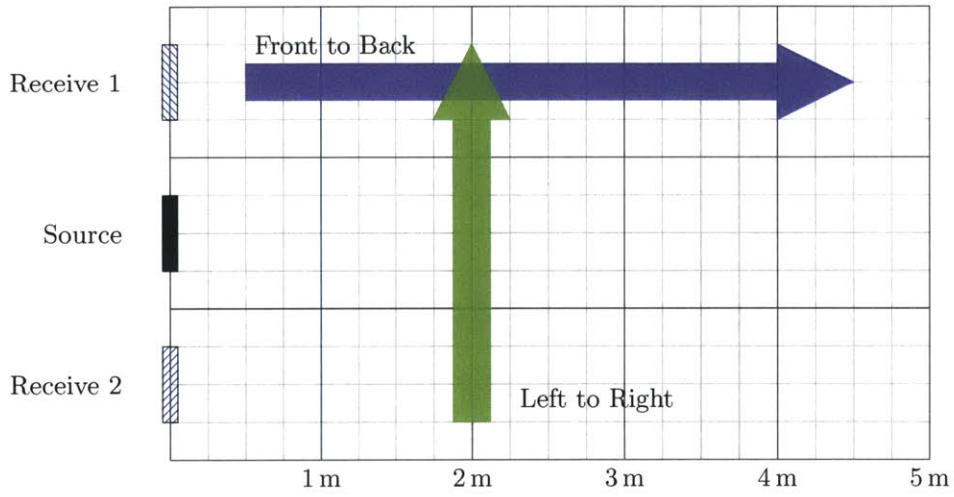
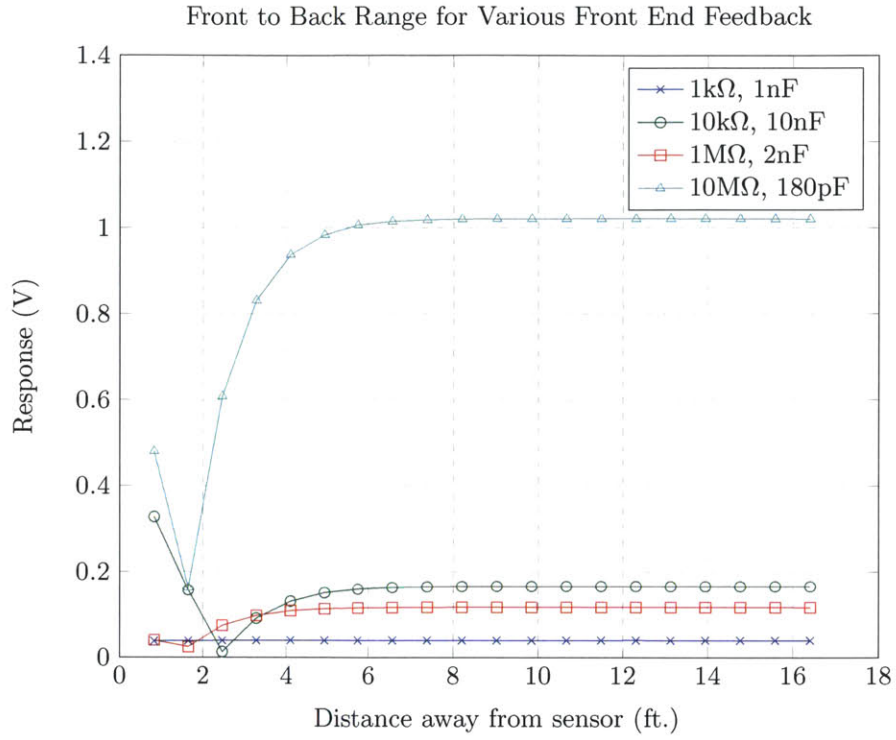


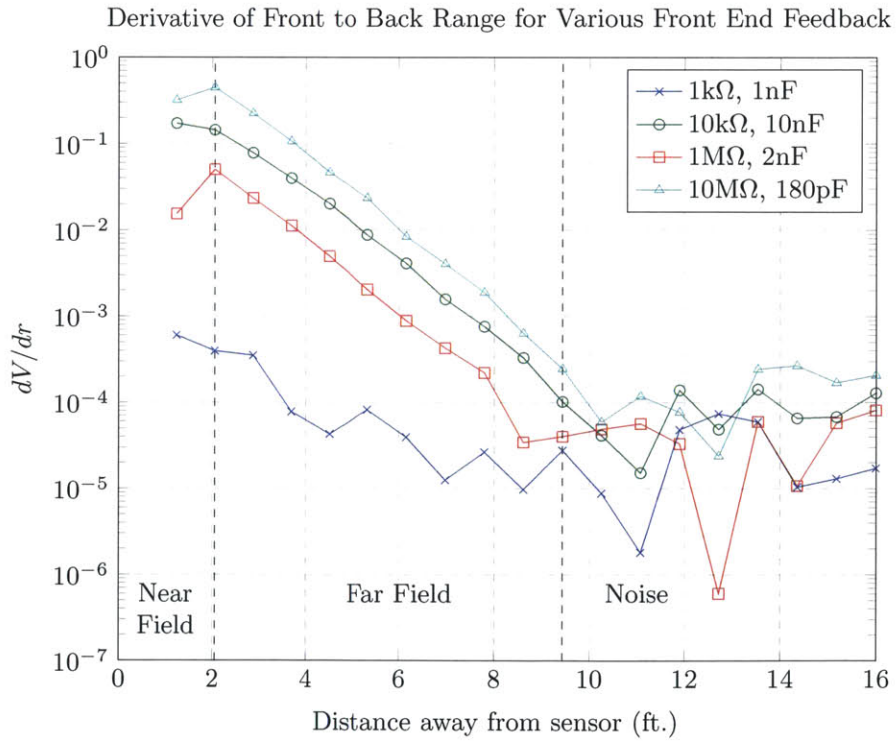
Figure 5-3: A top down illustration of the front to back and left to right testing paths.

Front End Feedback vs. Range

Sensing tests were performed with several different combinations of impedances in the front end amplifier feedback path. The front end feedback pairs evaluated are listed in Table 5.1. The goal of these tests was to determine which feedback pair provided the best sensing results. It is abundantly clear that the capacitive dominated feedback configuration is best suited for the standalone sensor. It yields the largest signals, due in part to the large gain of the capacitive mode feedback, which correspond to a larger detection range and increased sensitivity within range of the system.

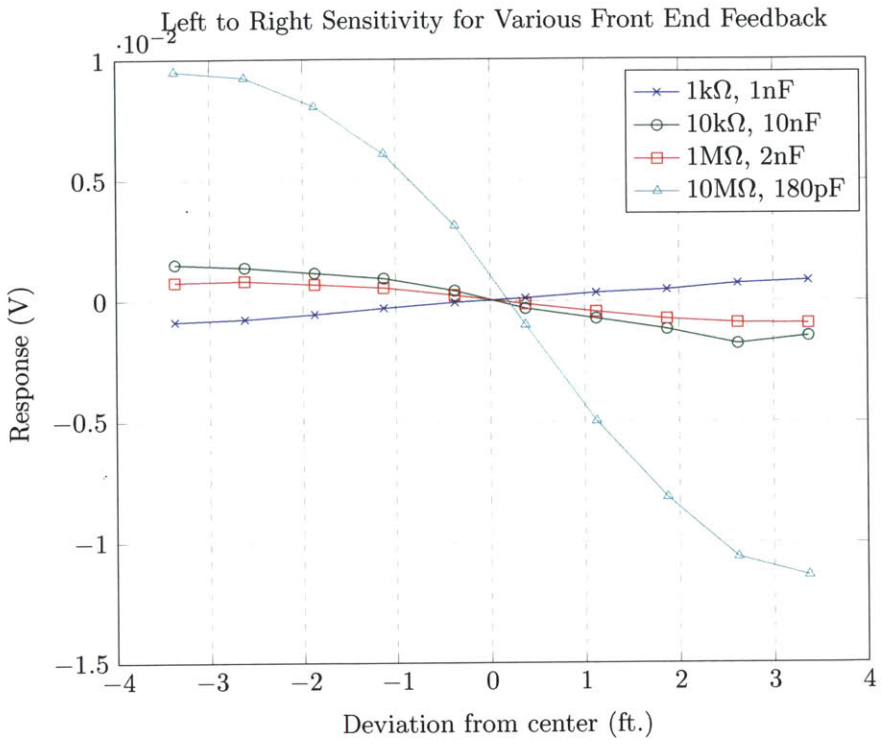


(a) Measured responses for front to back range tests with different front end compensation.

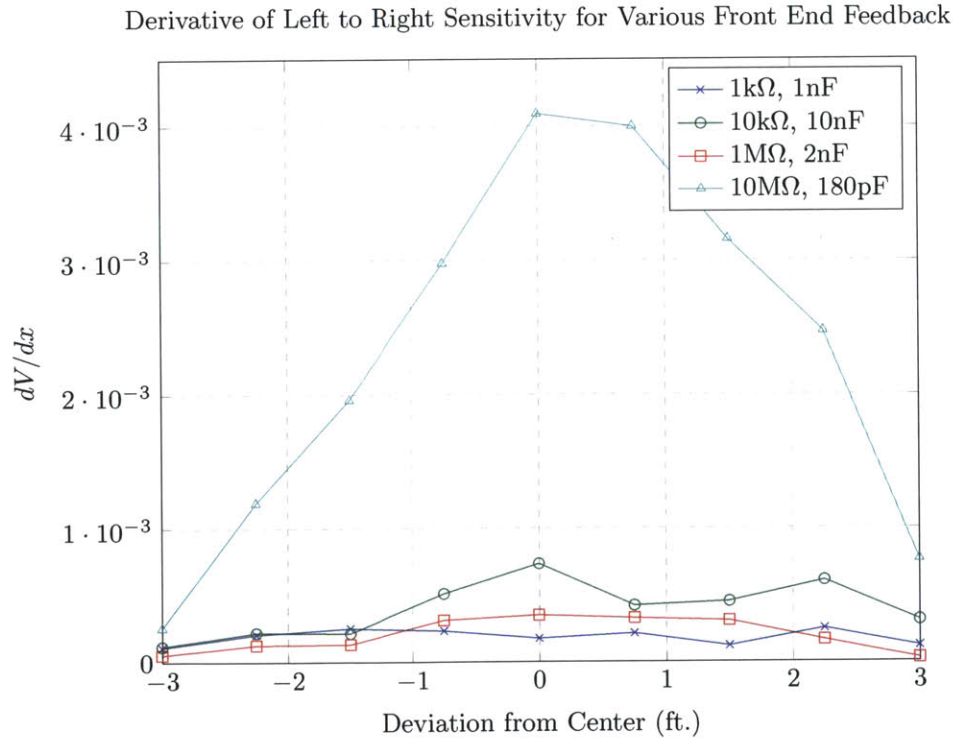


(b) Derivatives of the measured responses for front to back range tests with varied front end compensation.

Figure 5-4: Measured responses and their derivatives for front to back range tests with varied front end compensation.



(a) Measured responses for left to right range tests with different front end compensation.



(b) Derivatives of the measured responses for left to right range tests with varied front end compensation.

Figure 5-5: Measured responses and their derivatives for left to right range tests with varied front end compensation.

Table 5.1: Front End Amplifier Combinations

Plates	Mode	R_f	C_f	Pole
Pair 1	Capacitive	10 MΩ	180 pF	88.4 Hz
Pair 2	Capacitive	1 MΩ	2 nF	79.6 Hz
Pair 3	Resistive	10 kΩ	4.7 pF	3.39 MHz
Pair 4	Resistive	1 kΩ	40 pF	3.98 MHz

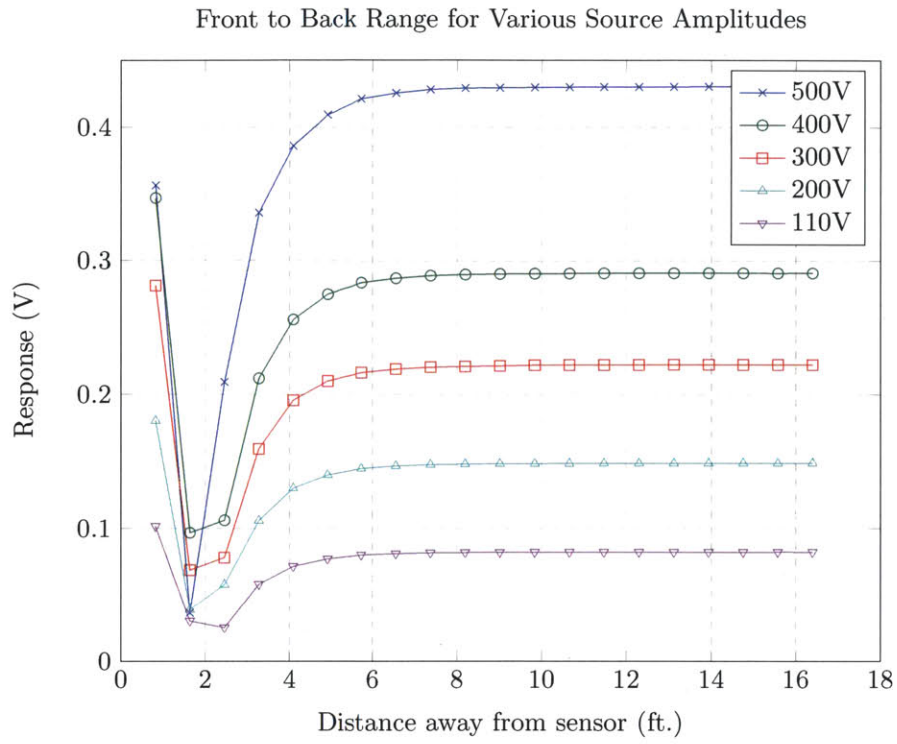
Source Voltage vs. Range

In order to explore the relationship between the source voltage and the range, tests were performed at varying source voltages. For each source voltage tested, the electrode configuration remained the same. The parameters for this set of tests is listed in Table 5.2. These tests were performed using the 10 MΩ and 180 pF front end feedback pair and with 6 feet of separation between receive electrodes.

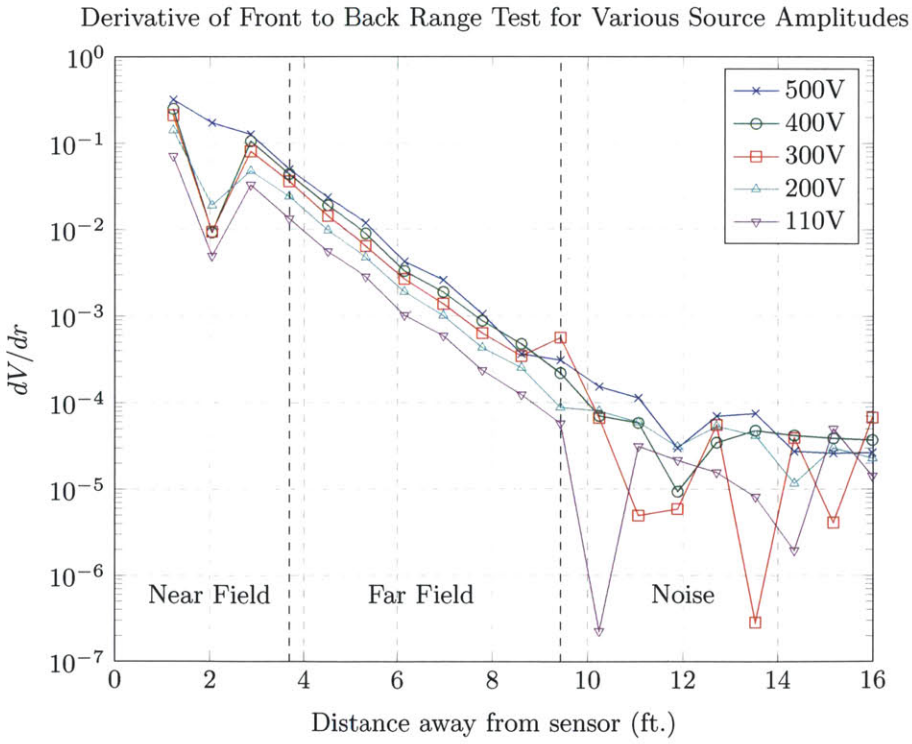
Table 5.2: Varied Source Voltage Test Parameters

Parameter	Value
Source Voltage Amplitude	110 V,200 V,300 V,400 V,500 V,
Electrode Separation	6 ft.
Operating Frequency	100 kHz
R_f	10 MΩ
C_f	180 pF

As expected, larger source voltages yield larger voltage responses. Both the front to back and left to right range tests display this characteristic. This suggests that larger source voltages are beneficial for increasing the sensitivity of the system, as illustrated by Figure 5-6a. On the other hand, it is clear from the front to back derivative plot that the increased source voltage is not strongly connected to the detection range of the system. As illustrated by Figure 5-8, doubling or tripling the signal source voltage adds only a marginal increase in detection range. It can then be inferred that physical constraints limit the detection range of the capacitively coupled occupancy sensor.

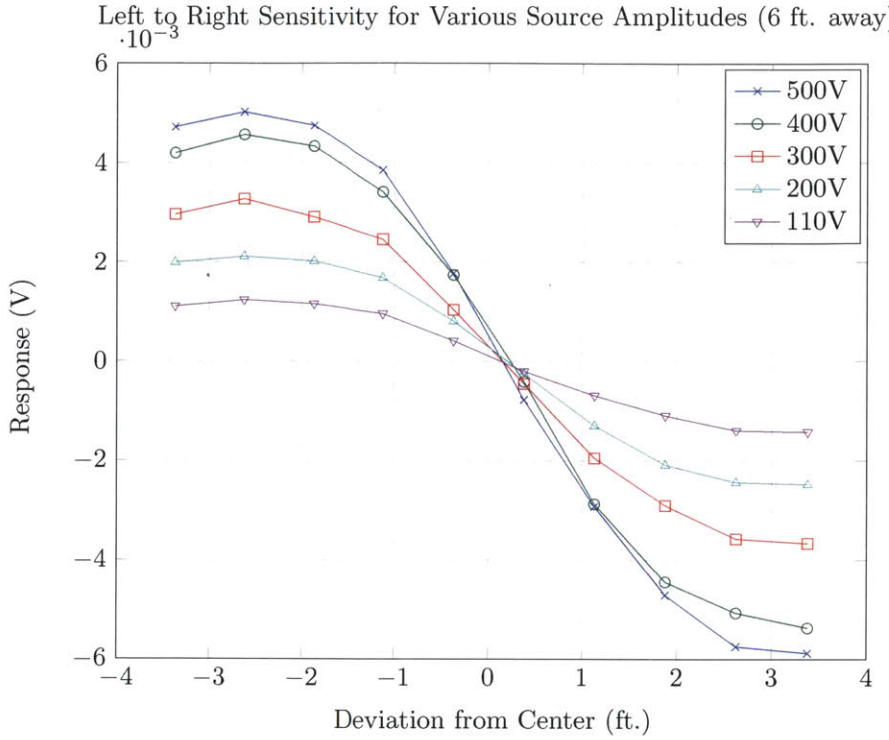


(a) Measured responses for front to back range tests with varied source amplitudes.

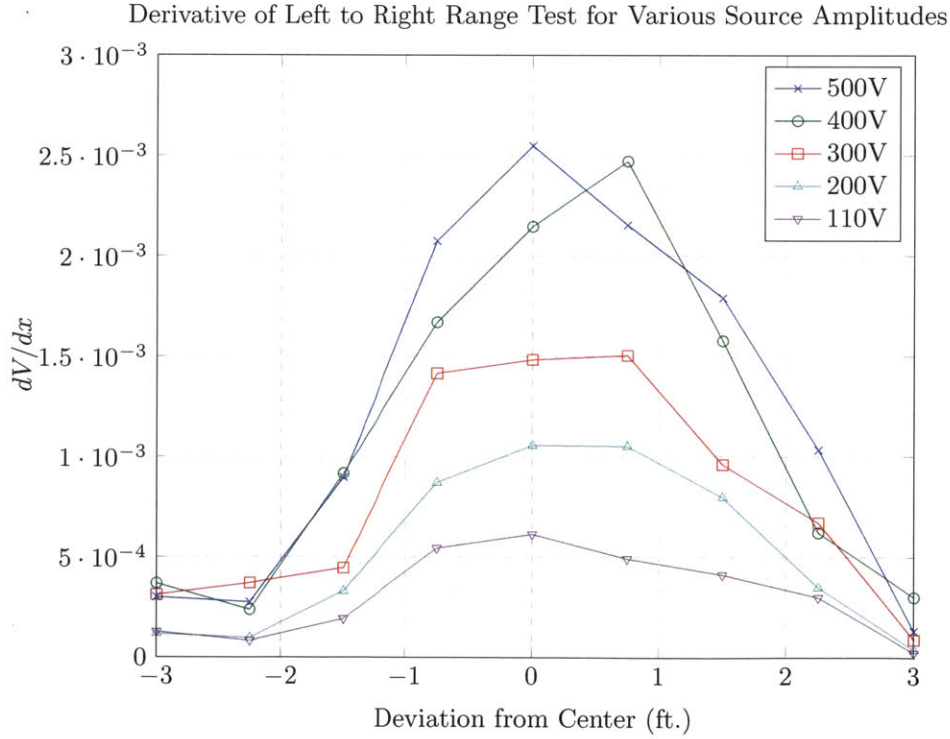


(b) Derivatives of the measured responses for front to back range tests with varied source amplitudes.

Figure 5-6: Measured responses and their derivatives for front to back range tests with varied source amplitudes.



(a) Measured responses for left to right range tests with varied source amplitudes.



(b) Derivatives of the measured responses for left to right range tests with varied source amplitudes.

Figure 5-7: Measured responses and their derivatives for left to right range tests with varied source amplitudes.

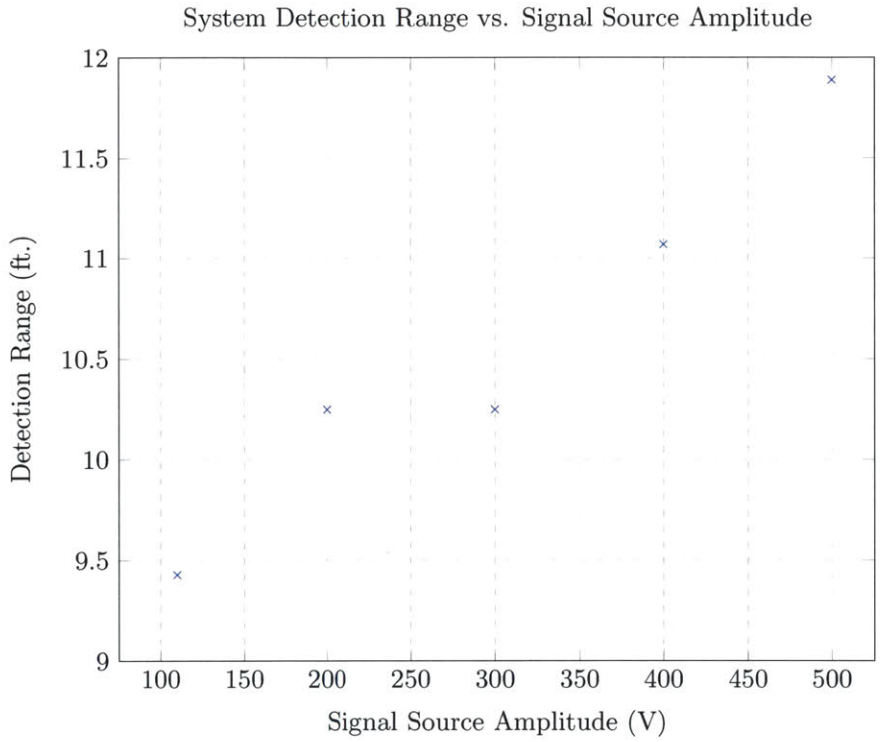


Figure 5-8: The system detection radius, based on front to back range tests, as a function of the system signal source amplitude.

The derivative of the front to back response, seen in Figure 5-6b, highlights another feature of the standalone sensor. The graph shows three distinct regions of operation, labeled as the near field region, the linear far field region, and the end of the detection range where the system noise is dominant. The noise boundary is approximate since each configuration has a varying noise profile. The near field region is located in the area where the occupant is close enough to the electrodes to act as a conduit. In such close proximity, the electric fields cannot be shunted to ground without interfering with the occupant.

The most important detail to note in far field region is its linearity. This implies that the environment is not a significant source of interference. Furthermore, to approximate the drop off of the electric field, this region may be fit and the non-linear roll off of the field modeled. For the far field region, the derivative of the response is approximately linear and thus can be modeled

$$y' = \frac{k}{r^n}, \tag{5.1}$$

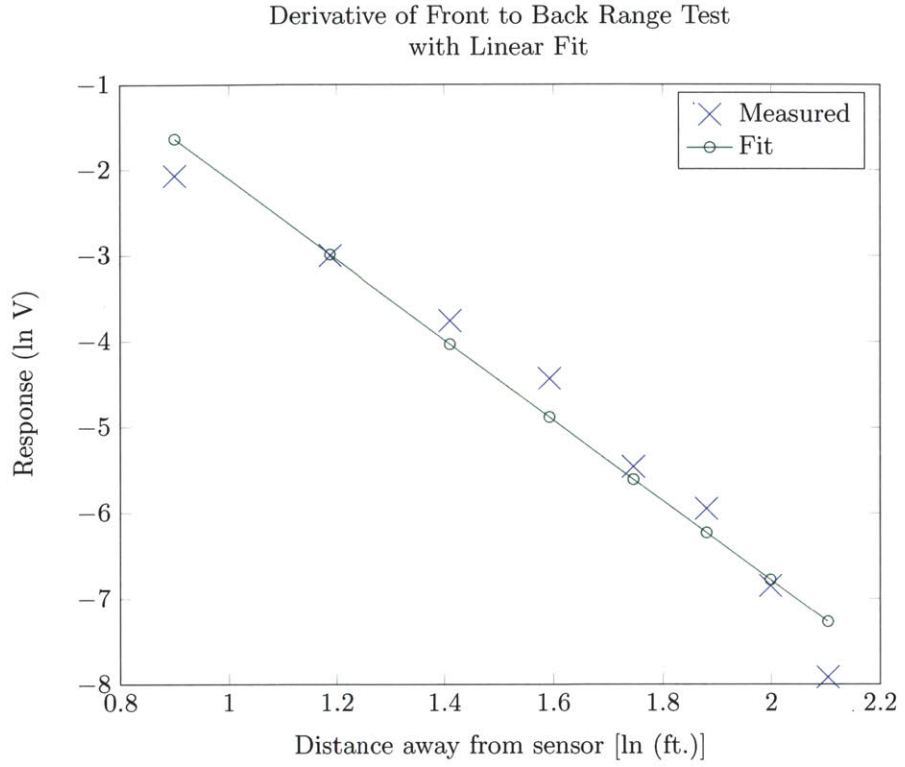
where y is the measured response, r the distance from the sensor, k a constant scaling factor, and n the dropoff of the response. By fitting this linear region, k and n can be approximated. The approximated values may then be evaluated with the integral of equation (5.1)

$$y = \frac{k}{1-n} \cdot \frac{1}{r^{n-1}}. \tag{5.2}$$

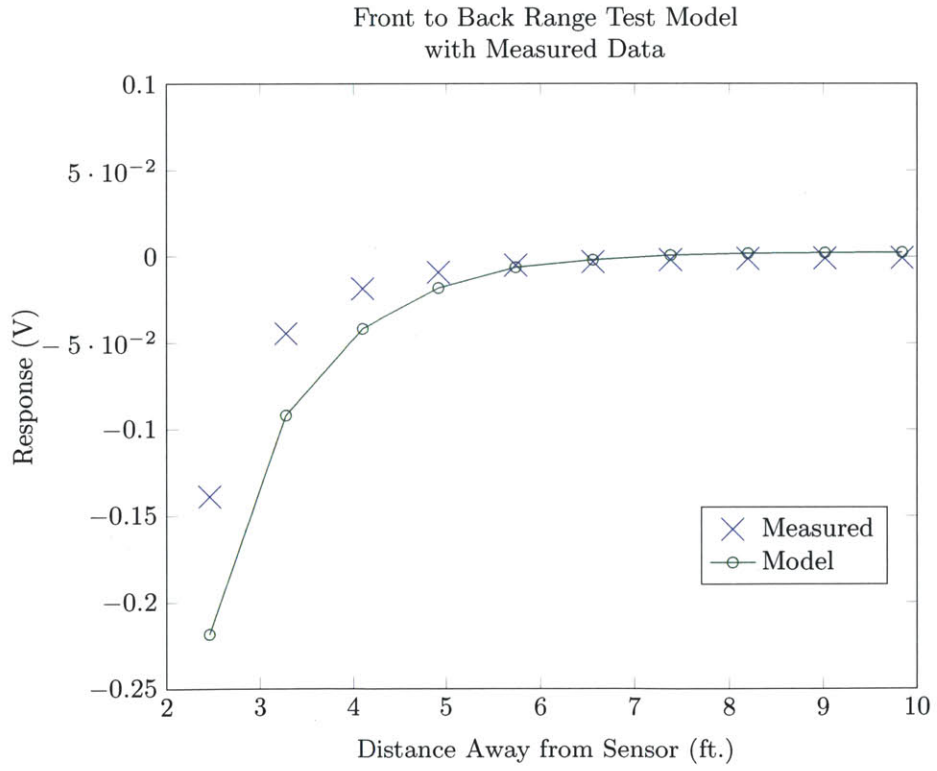
Figure 5-9 presents the linear fit of the derivative and the modeled sensor response, each with their respective data for the 10 MΩ, 180 pF plate. Table 5.3 present the constants and exponents for each signal source amplitude. The results of the fit show that the system response will, on average, drop off according to $\frac{1}{r^{3.6}}$.

Table 5.3: Linear Fit Parameters

V_s	# of Points	Fit Distance	k	n	R^2
110 V	8	6 ft.	3.2	4.66	.9787
200 V	8	6 ft.	4.23	4.48	.9766
300 V	8	6 ft.	7.65	4.62	.9834
400 V	8	6 ft.	9.06	4.55	.9837
500 V	8	6 ft.	13.16	4.68	.9640



(a) Linear fit of the derivative of the front to back range test response with measured data.



(b) Modeled response of front to back range test with measured data.

Figure 5-9: Front to back range test linear fit and modeled response.

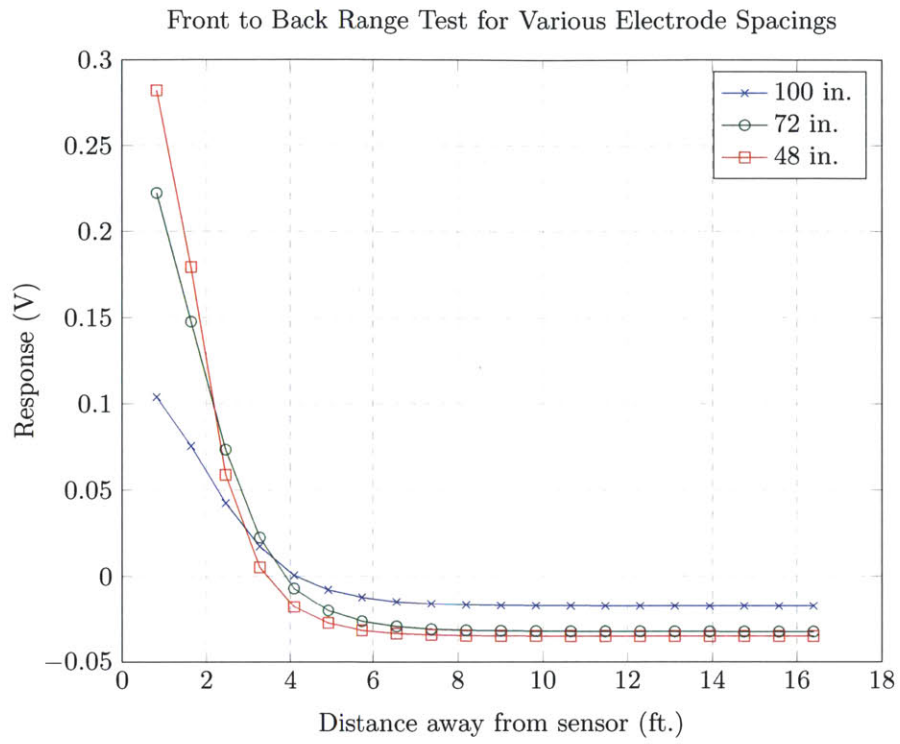
Electrode Geometry vs. Range

The geometry of the electrodes is also tied to the performance of the sensor. Two different separations are of particular interest: the spacing between receive electrodes and the spacing between the receive electrode plane and the source electrode plane. The interest again is the effect of the geometry on the range of the system. The parameters for the tests performed are listed in Table 5.4.

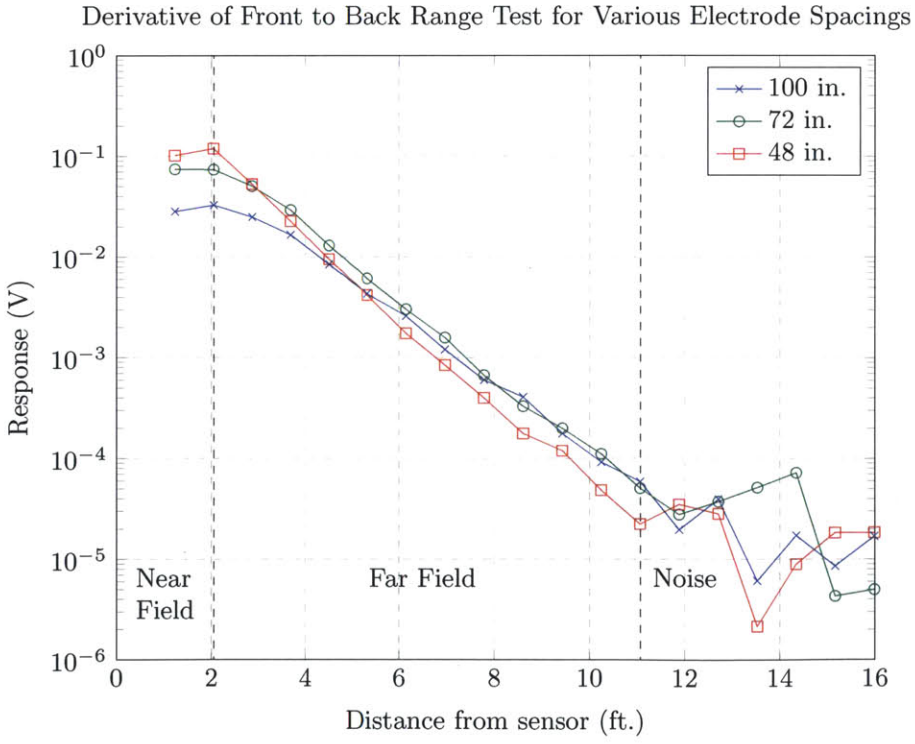
Table 5.4: Varied Geometry Parameters

Parameter	Value
Source Voltage Amplitude	300V
Receive Electrode Separation	4 ft, 6 ft, 8 ft
Operating Frequency	100 kHz

The same three operating regions can be seen, albeit to a lesser degree, in the derivative of the front to back range test. The first important feature is the near field sensitivity. As the separation decreases, the sensitivity close to the sensor increases. However, as the distance from the sensor increases, there is a crossover point at which the larger separation configuration becomes more sensitive. This implies that the sensor configuration may be tailored for a specific application. Bringing the plates closer together may be most beneficial for uses that do no require a large detection range and instead prioritize local sensitivity.

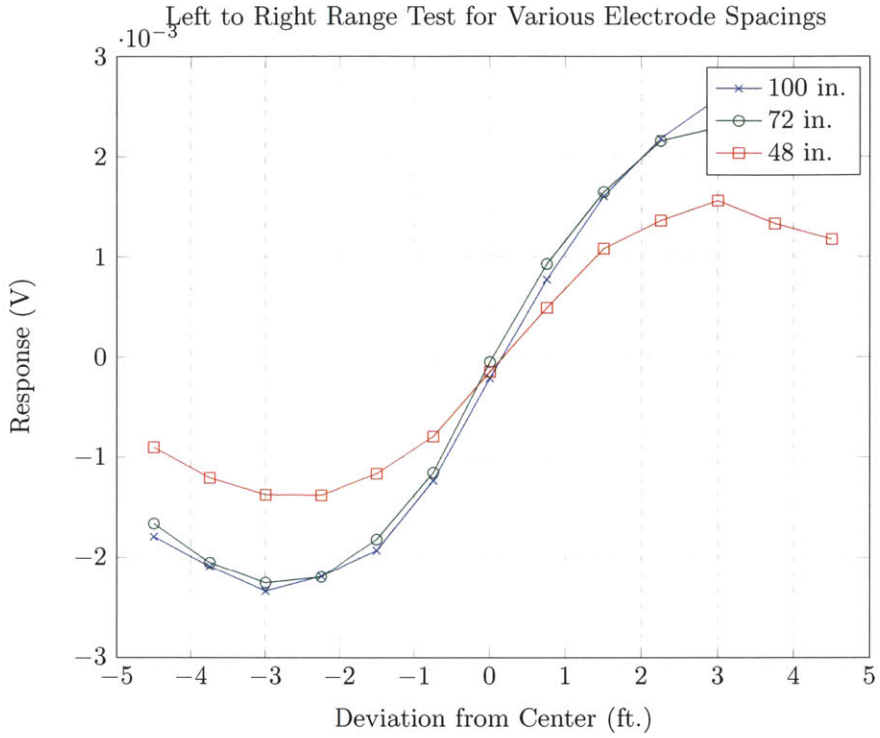


(a) Measured responses for front to back range tests with varied electrode spacing.

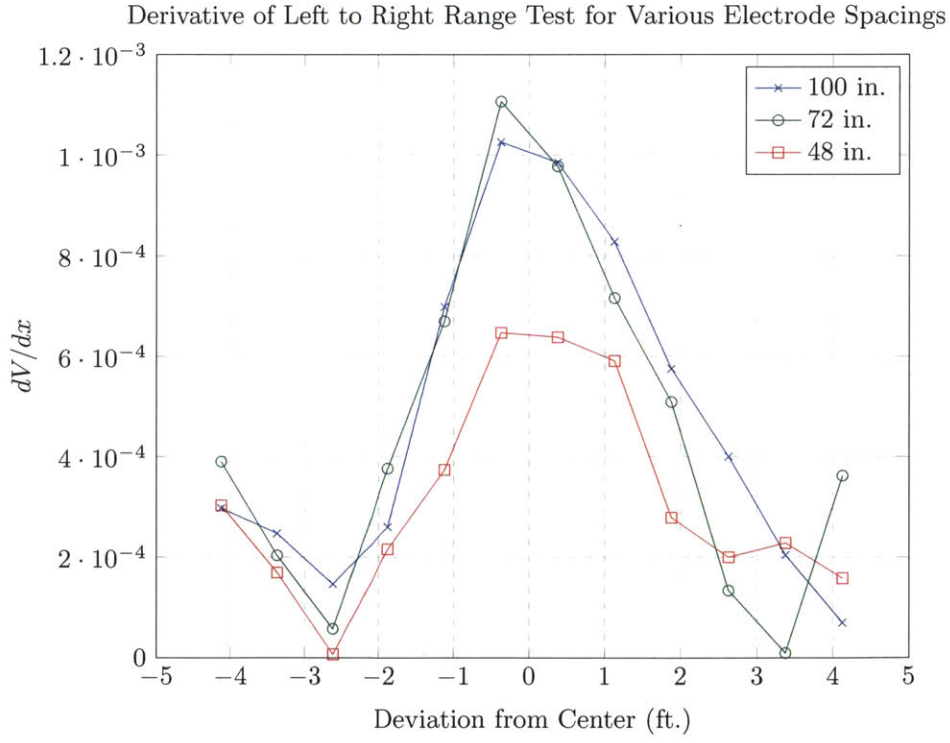


(b) Derivatives of the measured responses for front to back range tests with varied electrode spacing.

Figure 5-10: Measured responses and their derivatives for front to back range tests with varied electrode spacing.



(a) Measured responses for left to right range tests with varied electrode spacing.



(b) Derivatives of the measured responses for left to right range tests with varied electrode spacing.

Figure 5-11: Measured responses and their derivatives for left to right range tests with varied electrode spacing.

Chapter 6

Object Detection

Based on the discussion in Chapter 5, an occupant, p, within the detection range of the sensor may be mapped to some response, a_p, by a function f,

$$p \xrightarrow{f} a_p. \tag{6.1}$$

This concept is not restricted to people and in fact may be applied to other objects in the detection range, such as a metal object, by a second mapping

$$m \xrightarrow{g} a_m. \tag{6.2}$$

Both m and p represent electric field profiles, which are based on properties, such as volume, permittivity, conductivity, shape, and orientation, among others. Independent information through f and g allows an occupant with an object to be resolved. This is shown experimentally in Figure 6-1, which details the frequency response of the conductivity and permittivity of human muscle. For comparison, the frequency response of aluminum is shown in Figure 6-2. The relative permittivity of metals is just 1.

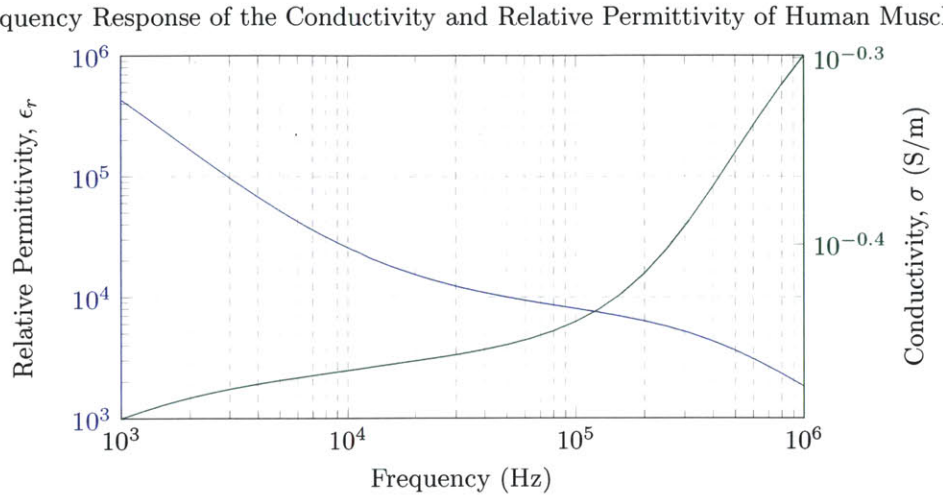


Figure 6-1: The conductivity and permittivity of human muscle as a function of frequency.

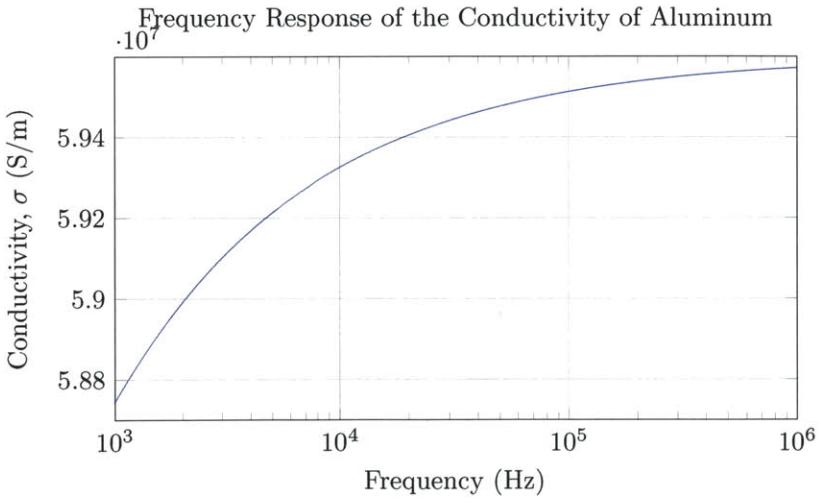


Figure 6-2: The conductivity of aluminum as a function of frequency.

6.1 Linear Mixing Detection Algorithm

In order to explore this, a linear mixing assumption is made, which provides

$$cp + dm \xrightarrow{f,g} ca_m + da_m. \tag{6.3}$$

The current system allows for four signals to be measured. These signals correspond to the in phase and quadrature measurements at two separate frequencies, 10 kHz and 100 kHz. Arranged in a matrix, they are

$$\begin{gathered} \mathbf{a} = \mathbf{K}\begin{pmatrix} \mathrm{P} \\ \mathrm{M} \end{pmatrix} \\ \begin{pmatrix} a_{I,10} \\ a_{Q,10} \\ a_{I,100} \\ a_{Q,100} \end{pmatrix} = \begin{pmatrix} k_1 & k_2 \\ k_3 & k_4 \\ k_5 & k_6 \\ k_7 & k_8 \end{pmatrix} \begin{pmatrix} \mathrm{P} \\ \mathrm{M} \end{pmatrix}, \end{gathered} \tag{6.4}$$

where $\mathbf{a}_{i,j}$ is the measured response, P and M are scalar electric field cross-sections, and k_i are the gains mapping the cross-sections to the data. Equation (6.4) can be rewritten as

$$\mathbf{a}_{\mathrm{pm}} = \mathbf{k}_{even}\mathrm{M} + \mathbf{k}_{odd}\mathrm{P}. \tag{6.5}$$

In the context of a detection algorithm, an object factor, ρ, is defined as the ratio of metal to person cross-sections

$$\rho = \frac{\mathrm{M}}{\mathrm{P}}. \tag{6.6}$$

The object factor is the metric by which person and object detection will be based. By normalizing so that $\mathrm{P} = 1$, equation (6.5) becomes

$$\mathbf{a}_{\mathrm{pm}} = \mathbf{k}_{even}\rho_o + \mathbf{k}_{odd}(1), \tag{6.7}$$

where ρ_o is the calibration factor. Subtracting the person response, $\mathrm{P}|_{\mathrm{P}=1}$, from the measured response removes the influence of the person on the measurement. To best approximate the true metal gains, ρ_o can be found by

$$\rho_o = \frac{\mathbf{k}_{even}^T\big(\mathbf{a}_{\mathrm{pm}} - \mathbf{k}_{odd}(1)\big)}{\mathbf{k}_{even}^T\mathbf{k}_{even}}, \tag{6.8}$$

and subsequently multiplied with the original $\mathbf{k}_{even}$ such that

$$\mathbf{k}_{even,o} = \rho_o \cdot \mathbf{k}_{even}, \tag{6.9}$$

and the new metal gains are compensated. Finally, with all of the gains known, the object factor for a given experiment may be determined. Using least squares, the electric field cross-sections are found by

$$\begin{pmatrix} \mathrm{P} \\ \mathrm{M} \end{pmatrix} = \left(\mathbf{K}^T\mathbf{K}\right)^{-1}\mathbf{K}^T\hat{\mathbf{a}}_{\mathrm{pm}}, \tag{6.10}$$

where $\mathbf{K}$ is now comprised of $\mathbf{k}_{odd}$ and $\mathbf{k}_{even,o}$. The detection metric is then

$$\hat{\rho} = \frac{\mathrm{M}}{\mathrm{P}}. \tag{6.11}$$

6.2 Multifrequency Tests

In practice, this algorithm is first implemented with three calibration stages in order to obtain the components necessary to determine $\mathbf{K}$. The vector $\mathbf{a}$ is the response measured from each test. All tests performed were similar to the left to right range tests. The elements that comprise $\mathbf{a}$ are defined as the peak to peak deltas of the in phase and quadrature response curves, as shown in Figure 6-3. This places emphasis on the sensed changes during a test. Any long term effects due to offsets present in the response are ignored. This removes any potential error in the data caused by drift in the system.

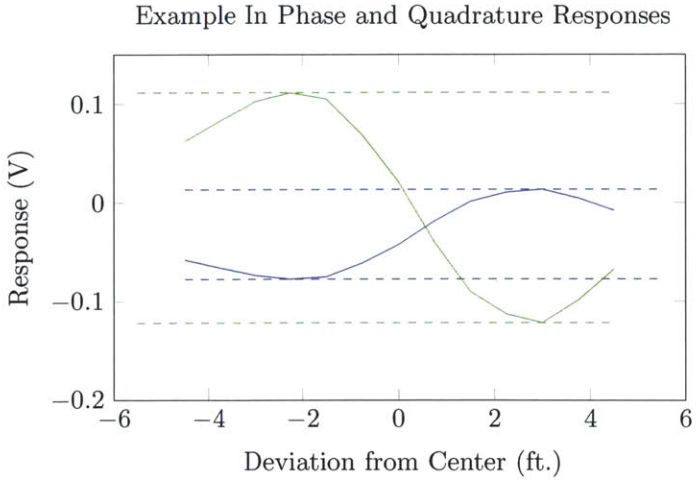


Figure 6-3: Example in phase and quadrature response curves and the measurements of interest.

This method presumes that the deltas measured for every experiment are equivalent measurements, which means that the peaks of the responses occur at the same point for each test. Figure 6-4

shows the derivatives of two pairs of in phase and quadrature curves from two different tests. The zero crossing then, corresponds to the peak of the measured response. This illustrates that while the peaks of the same test are perfectly matched, the peak position from test to test varies. However, as noted by the scale, the change in peak position is small and thus may be ignored.

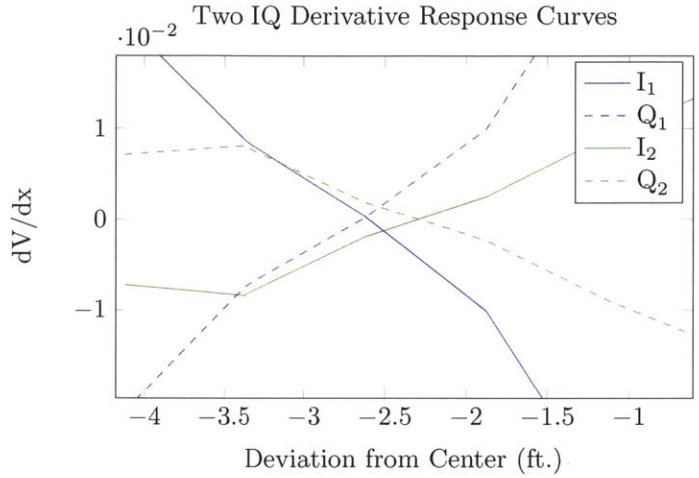


Figure 6-4: Mismatch between zero crossings of the derivative response curves.

The calibration tests include a person-only run to find $\mathbf{k}_{odd}$, a metal-only test for the initial $\mathbf{k}_{even}$, and finally a test with both person and metal. The final calibration run yields the set of measured $\mathbf{a}_{\text{pm}}$ needed to find the true $\mathbf{k}_{even,o}$. With the three calibration tests complete, ρ may be found from equation (6.8) and finally the adjusted $\mathbf{k}_{even}$ from equation (6.9).

6.3 Object Detection Implementation

Preliminary data collected using the detection algorithm outlined here is shown in Table 6.1. For these tests, the system was driven with a summed 10 kHz and 100 kHz sinusoid with an amplitude of 300 V. The left to right tests were performed 1 m away from the sensor over a span of 10 ft. The metal-only calibration was performed with the aluminum cylinder, while the person-metal calibration done with an aluminum toolbox.

Table 6.1: Object Detection Results

Object	$\frac{\text{M}}{\text{P}}$
Stool	1.97
Aluminum Cylinder	0.26
Aluminum Bracket	0.81

The three objects held during the tests were, from largest to smallest, a metal stool, an aluminum

cylinder, and an aluminum bracket. The results of applying the algorithm suggest that the stool had the largest cross-section of the three, followed by the bracket, and lastly the cylinder. Furthermore, though the bracket was physically smaller, other properties of its electric field cross-section are such that it has a larger effect than the cylinder.

While these are preliminary results, they suggest that linear mixing may provide a good approximation. Further testing must be done in order to confirm statistical significance.

Chapter 7

Conclusions and Future Work

7.1 Summary

Chapter 2 laid out the design parameters and requirements for the standalone capacitively coupled occupancy sensor.

Chapter 3 discussed the high voltage signal source designed to replace the fluorescent lamp. Removing the dependency on the fluorescent lamp as the signal source required the standalone system to provide an alternative. The fluorescent lamp was used as a guideline for the minimum operating requirements. The design expands on these requirements to increase the breadth of possible operating configurations. The input to the signal source is digitally synthesized and amplified as much as possible before finally driving a step-up autotransformer. The final implementation is capable of generating a single, or summed, sinusoid with an amplitude of up to 500 V over a wide frequency range from tens to hundreds of kilohertz.

Chapter 4 described the design of the standalone sensor front end amplifier and signal conditioning circuitry. A new approach was taken to design and implement the receive electrode. The front end amplifier was placed on the electrodes themselves and the sensed signal then sent on a driven cable to the signal conditioning electronics. This method is in contrast to previous iterations, where undriven cables were used to connect the inputs of the front end amplifier to passive electrodes. The new method confines the sensing node to a well controlled area and limits pickup of extraneous signals and noise. The signal conditioning circuitry synchronously detects the in phase and quadrature components of the sensed signal, which is then read by an ADC. Finally, the on board microcontroller sends the data to a computer for analysis.

Chapter 5 presented the results of the system performance tests. The detection radius was tested for various system configurations, including front end amplifier compensation, electrode spacing, and signal source amplitude. The detection range of the system experienced diminishing returns for

increases in the signal source amplitude and receive electrode spacing. Ultimately, 11 ft. was found to be the maximum detection range of the system. Sensitivity was found to scale with the signal source amplitude in the short range. Furthermore, the spacing between receive electrodes gives an indication of the tradeoffs between sensitivity and range. Overall the sensor was found to be capable of attofarad measurements.

Lastly, in Chapter 6 material detection was shown to at least be a feasible application via the linear mixing detection algorithm, though further testing is necessary for statistical value.

7.2 System Improvements and Alternatives

Though functional, the implementation of the standalone sensor system revealed limitations of the design. Improvements to these concerns are outlined here.

7.2.1 Digital Synthesis

The AD9837 direct digital synthesis chip, used to generate the input sine wave to the high voltage signal source, is not designed for use in precision systems. The output voltage temperature coefficient is the largest in the high voltage signal chain. Thus it is the weakest link regarding signal source drift. The alternative to using an all-in-one solution is to break out the individual components and reimplement them individually more accurately.

7.2.2 Wide Rail Power Op-Amp

While the op-amp used in the last stage of the high voltage signal source is designed for wide rail operation, its gain bandwidth product and slew rate leave a little to be desired. As such, the OPA452 cannot meet all of the demands of the standalone sensor. It is not able to be driven as hard as the rest of the high voltage circuitry. Instead a push pull output stage specifically can be designed to handle the wide rails at the operating frequencies of the sensor. This would allow the system to operate at even higher source amplitudes. Though it was showed that increases in the amplitude of the signal source have diminishing returns for the detection range, the system sensitivity would benefit greatly.

7.2.3 Upstream ADC

An alternative to the existing signal conditioning circuitry is to recreate it digitally. Moving the ADC upstream in the chain would allow for multiplier and filters to be completely digital. Concerns regarding impedance mismatches, precision, and noise may be ignored. Instead the difficulty involves

driving the upstream ADC without degrading the sensed signal. The ADC chosen must have a sufficient sample rate and resolution in order to accurately measure the incoming sensed signals.

7.3 Future Work

7.3.1 Differential Signal Source

One key aspect that the standalone sensor does not exactly replicate from the fluorescent lamp versions is the fully differential signal source. Instead it relies on stray coupling to provide a return path to the signal source reference. The two bulbs of the fluorescent lamp, with which previous sensors were integrated, served as a differential drive with an explicit return path. The subtleties between the two drivers and their tradeoffs will be explored in the future.

7.3.2 Multiple Sensor Applications

Implementing multiple sets of receive electrodes opens the possibility of imaging a space. A version of the standalone sensor, capable of multiplexing several inputs, may be employed as a form of an electrical impedance tomographer. By relying on the differing frequency responses of various materials, in a similar fashion to Chapter 6, an object may be imaged.

7.3.3 Large Scale Implementation

Because the sensor is not tied to a lamp or any other type of infrastructure, it may be scaled freely. A logical next step would be to drastically increase the area of the electrodes. This may provide a method of combating the dropoff of the electric field. If such large electrodes, like a room sized installation, are indeed able to do so, then the door to large scale occupancy detection will be opened. Alternatively, the geometry of the electrodes may be configured in such a way as to aim the electric field and tune the detection range to a specific area.

Appendix A

Code

A.1 Sensor Interfacing Software in Matlab®

A.1.1 readADC.m

```
function val = readADC(s, chan)

%% Send the command to the sensor if necessary
fprintf(s,'%s',strcat('A',chan));

%% Read from the board

% matrix = fscanf(s,'%s');
raw = fscanf(s,'%02x');
matrix = rot90(raw);

%% Print channel and received data

fprintf('%c: ',chan);

for k=1:4
    fprintf('%02X ',matrix(k));
end

%% Interpret data received from the board and print and return read value

if bitget(matrix(1),6)
    result = matrix(1)*2^24+matrix(2)*2^16+matrix(3)*2^8+matrix(4);
    result = bitshift(result,-5);
    result = bitand(result,16777215);
    val = result*5/(2^24-1);
    fprintf(', %f\n', val);
elseif bitget(matrix(1),6) == 0
```

```
    result = matrix(1)*2^24+matrix(2)*2^16+matrix(3)*2^8+matrix(4);
    result = bitshift(result,-5);
    result = bitand(result,16777215);
    val = -(16777215-result)*5/(2^24-1);
    fprintf('␣%f\n', val);
end

end
```

A.1.2 recorder.m

```
function recorder(port,pts,ADC,file)

%% Open Data Link to Board
s = open_ser(port);

%% Initialization
% port: port to collect data from
% pts: number of points to record
% ADC: array argument to select ADCs to read
%      should be in the form: ['0','1']
% file: target file to save data to

% Set drive frequencies
% - x1: 10.416 kHz
% - x2: 83.333 kHz
% fprintf(s,'%s ',' p2'); % PWM frequency for demodulation
% fprintf(s,'%s ',' d1'); % DDS source drive frequency

%% Open Target Save File
fid = fopen(file,'w'); % Open file with write access
fprintf(fid,'%s\r\n',datestr(now)); % Record starting time and date

% Write column headers
for k = 1:2:length(ADC)
    fprintf(fid,'%c,',ADC(k));
    fprintf(fid,'%c,',ADC(k+1));
    fprintf(fid,'iq%d,',k);
end

fprintf(fid,'\r\n'); % New line for the recorded data

%% Main plot loop
for l = 1:pts
    % Print current point to screen
    fprintf('%d:␣\r\n',l)
```

```
    % Read one channel from board
    for k = 1:2:length(ADC)
        % Read a pair of ADCs and find sqrt of the sum of the squares
        sample_i = readADC(s,ADC(k));
        sample_q = readADC(s,ADC(k+1));
        sample = sqrt(sample_i^2 + sample_q^2);

        % Print results to screen and write to file
        fprintf('%2.6f, %2.6f, %2.6f\r\n',sample_i,sample_q,sample);
        fprintf(fid,'%10.18f,%10.18f,%10.18f,',sample_i,sample_q,sample);
    end

    % Create new line for after each datapoint
    fprintf(fid,'\r\n');

    % Wait sufficient time to resample
    pause(1/6.7);
end

fprintf(fid,'%s',datestr(now)); % Record time and date at the end of recording

% Close file and serial port upon termination
fclose(fid);
stopasync(s);
fclose(s);

end
```

A.1.3 liveGraph.m

```
function liveGraph(port,pts,avgs,ADC)

%% Open Data Link to Board
s = open_ser(port);

%% Initialization
% port: port to collect data from
% pts: number of points in plot buffer
% avgs: number of points to average over
% ADC array to choose which ADCs to read from

% Set drive frequencies
% - x1: 10.416 kHz
% - x2: 83.333 kHz
fprintf(s,'%s','p3'); % PWM frequency for demodulation
fprintf(s,'%s','d3'); % DDS source drive frequency
```

```
% Initialize sample and plotting arrays
samples_iq = zeros(length(ADC),pts);
samples_iq_avg = ones(length(ADC),1);

samples = zeros(length(ADC)/2,pts);
samples_avg = ones(length(ADC)/2,1);

% fig = figure(1);

%% Main plot loop
while(1)
    % Shift sample arrays left and append placeholder 0
    samples_iq = [samples_iq(:,2:length(samples_iq)) zeros(length(ADC),1)];
    samples = [samples(:,2:length(samples)) zeros(length(ADC)/2,1)];

    % Read one channel from board
    for k = 1:2:length(ADC)
        % Read a pair of ADCs and find sqrt of the sum of the squares
        sample_i = readADC(s,ADC(k));
        sample_q = readADC(s,ADC(k+1));
        sample = sqrt(sample_i^2 + sample_q^2);

        % Place results in appropriate row in the last column of each array
        % (this should be in place of the placeholder 0)
        samples_iq(k:k+1, length(samples_iq)) = [sample_i; sample_q];
        samples(ceil(k/2), length(samples)) = sample;

        % Print results
        fprintf('%2.6f, %2.6f, %2.6f\r\n',sample_i,sample_q,sample);
    end

    % Get the average value of each raw channel for the last "avgs" samples
    for k = 1:length(ADC)
        samples_iq_avg(k,1) = mean(samples_iq(k,length(samples_iq)-avgs:end));
    end
    for k = 1:length(ADC)/2
        samples_avg(k,1) = mean(samples(k,length(samples)-avgs:end));
    end

    % Populate plotting and average buffers
    plotbuf = samples;
    avgbuf = samples - (samples_avg*ones(1,length(samples)));

    % Plot the sampled data from each ADC
%       ylabel('Volts');
%        title(['Chan ' int2str(k)]);
%       axis([1 pts -.002 +.002]);
%        axis([1 pts 0 .05])
```

```
    for k = 1:length(ADC)/2
        subplot(2,length(ADC)/2,k)
        plot(plotbuf(k,:))
        title(['Chan ' int2str(k)]);
        axis([1 pts 0e-3 600e-3]);
%         axis([1 pts 1 1.2]);
        subplot(2,length(ADC)/2,k+length(ADC)/2)
        plot(avgbuf(k,:))
        title(['Avg. of ' int2str(k)]);
        axis([1 pts avgbuf(k,pts)-1e-3 avgbuf(k,pts)+1e-3]);
    end

    % Quit the loop if 'q' is pressed, close figure as well if 'Q'
    if strcmp(get(1,'currentcharacter'),'q') || strcmp(get(1,'currentcharacter'),'Q')
        % Close serial port
        stopasync(s);
        fclose(s);
        delete(s);
        clear s;
        if strcmp(get(1,'currentcharacter'),'Q')
            close all;
            break
        else
            break
        end
    end

    drawnow
    pause(1/6.7);
end

end
```

A.1.4 rangeExpt.m

```
function response = rangeExpt(port,pts,avgpts,ADC,file)
% port: port to collect data from
% pts: number of range points to record
% avgpts: number of samples to average at each point
% ADC: array argument to select ADCs to read
%      should be in the form ['0','1']
% file: target file to save data to

%% Open Data Link to Board
s = open_ser(port);
```

```
%% Initialize
% Preallocate space for result
response = zeros(length(ADC),pts);

% Set drive frequencies
% - x1: 10.416 kHz
% - x2: 83.333 kHz
%fprintf(s,'%s ',' p1'); % PWM frequency for demodulation
fprintf(s,'%s','d3'); % DDS source drive frequency

%% Open Target Save File
fid = fopen(file,'w'); % Open file with write access
fprintf(fid,'%s\r\n',datestr(now)); %Record starting time and date

% Write column headers
for k = 1:2:length(ADC)
    fprintf(fid,'%c,',ADC(k));
    fprintf(fid,'%c,',ADC(k+1));
    fprintf(fid,'iq%c,',ADC(k));
end

fprintf(fid,'\r\n'); % New line for the recorded data

%% Record from board
waitforbuttonpress; % Press button to start recording

for k = 1:pts
    % Press button before each point
    waitforbuttonpress;
    fprintf('%s',repmat(sprintf('\n'),1,10));
%     pause(8);

    % (Re)-Initialize sample array
    pt_vals = zeros(length(ADC),avgpts);

    % Poll ADCs and discard result to flush
    % each ADC before recording.
    for l = 1:length(ADC)
        readADC(s,ADC(l));
    end
    % Pause sufficient time before beginning to sample
    pause(1/6.7);

    for l = 1:avgpts
        % Sample each ADC once
        for n = 1:length(ADC)
            pt_vals(n,l) = readADC(s,ADC(n));
        end
```

```
        % Pause before resampling
        pause(1/6.7);
    end

    for m = 1:length(ADC)
        % Average sampled values
        response(m,k) = mean(pt_vals(m,:));

        % Write sampled values to file
        fprintf(fid,'%c,',ADC(m));
        fprintf(fid,'%10.18f,',pt_vals(m,:));
%           fprintf(fid,'%10.18f', response(m,k));
        fprintf(fid,'\r\n');
    end
    fprintf('%s\n',repmat(sprintf('='),1,50));
    fprintf('%s\n',repmat(sprintf('='),1,50));
    fprintf('%s\n',repmat(sprintf('='),1,50));
    fprintf('%s\n',repmat(sprintf('='),1,50));
    fprintf('%s\n',repmat(sprintf('='),1,50));
end

%% Write averaged samples (response) to file
fprintf(fid,'%s\r\n','Average(s)');
for k = 1:length(ADC)
    fprintf(fid,'%10.18f,',response(k,:));
    fprintf(fid,'\r\n');
end

%% Close file and serial port
fclose(fid);
stopasync(s);
fclose(s);
delete(s);
clear s;
close all;

end
```

A.1.5 gridExpt.m

```
function response = gridExpt(port,xpts,ypts,avgpts,ADC,file)
% Inputs:
%     port: port to collect data from
%     xpts: number of x points to record
%     ypts: number of y points to record
%     avgpts: number of samples to average at each point
%     ADC: array argument to select ADCs to read
```

```
%           should be in the form  ['0','1']
%     file : target  file  to save data to
% Output:
%       response:


%% Open Data Link to Board
s = open_ser(port);

%% Initialize and Set Up Experiment
% Preallocate space for  result
response = zeros(xpts,ypts,length(ADC));

% Set drive and chop frequencies
% - x1: 10.416 kHz
% - x2: 83.333 kHz
% - x3: 10.416 & 83.333 kHz Drive
%       Ch1: 10.416 kHz, Ch2: 83.333 kHz Chop
% fprintf(s,'%s ',' p1'); % PWM Chop
% fprintf(s,'%s ',' d1'); % DDS

%% Open Target Save File
fid = fopen(file,'w'); % Open file with write acces
fprintf(fid,'%s\r\n',datestr(now)); % Record starting time and date
fprintf(fid,'%s\r\n',ADC); % Record ADC
fprintf(fid,'Grid: %d x %d\r\n',xpts,ypts); % Record grid size

%% Record Data from Board
waitforbuttonpress; % Press button to start experiment

% Iterate through each column of every row in grid
for k = 1:ypts
    for l = 1:xpts
        % Press button before each point
        waitforbuttonpress;

        % (Re)-Initialize sample array
        pt_vals = zeros(length(ADC),avgpts);

        % Flush ADCs before each point
        for m = 1:length(ADC)
            readADC(s,ADC(m));
        end
        pause(1/6.7)

        % Sample each ADC at each point
        for m = 1:avgpts
            for n = 1:length(ADC)
```

```
                pt_vals(n,m) = readADC(s,ADC(n));
            end
            pause(1/6.7);
        end

        for m = 1:length(ADC)
            % Average the sampled values for each point
            response(l,k,m) = mean(pt_vals(m,:));

            % Write each sampled value to file
            % This will write length(ADC) rows per point.
            % There will be xpts*ypts*length(ADC) rows total,
            %  as there will be a new row for each point.
            fprintf(fid,'%c,',ADC(m));
            fprintf(fid,'%10.18f,',pt_vals(m,:));
            fprintf(fid,'\r\n');
        end
    end
end

%% Write final (averaged) respones to file
fprintf(fid,'%s\r\n','Response');
for k = 1:ypts
    for l = 1:length(ADC)
        fprintf(fid,'%10.18f,',response(k,:,m));
        fprintf(fid,'\r\n');
    end
end

%% Close file and serial port
fclose(fid);
stopasync(s);
fclose(s);
delete(s);
clear s;
% close all;

end
```

A.2 Sensor Board Software

The software below can be compiled from a fresh installation of Ubuntu with the following commands.

```
sudo apt-get install build-essential subversion git gcc-avr avr-libc dfu-programmer
cd $HOME
svn checkout https://bucket.mit.edu/svn/Standalone_Sensor/
mkdir $HOME/atmega32u4/
```

```
cd $HOME/atmega32u4/
git clone https://github.com/abcminiuser/lufa-lib.git
cd lufa-lib/
git checkout acd92983
cd $HOME/Standalone_Sensor/Sensor\ and\ Code/code/Fully_Diff/
```

The command

```
make
```

will compile the software. The command

```
make dfu
```

will program the ATmega32U4 microcontroller breakout board with the makefile included here. The breakout board must be reset with the HWB button held down, in order to enter the bootloader, and program the microcontroller.

A.2.1 makefile

```
# Hey Emacs, this is a -*- makefile -*-

# Fully Differential Standalone Sensor
# MIT 04/2012
#
#
# Makefile edited for use with Atmega32U4 and
# Fully Differential Standalone Sensor by:
#  BJ Thompson
#
#
# NOTE: LUFA path must be edited if not built on <sensor>
#
# Adapted from (below):

# WinAVR Makefile Template written by Eric B. Weddington, Jrg Wunsch, et al.
# >> Modified for use with the LUFA project. <<
#
# Released to the Public Domain
#
# Additional material for this makefile was written by:
# Peter Fleury
# Tim Henigan
# Colin O'Flynn
# Reiner Patommel
# Markus Pfaff
# Sander Pool
```

```
# Frederik Rouleau
# Carlos Lamas
# Dean Camera
# Opendous Inc.
# Denver Gingerich
#

# On command line:
#
# make all = Make software.
#
# make clean = Clean out built project files.
#
# make coff = Convert ELF to AVR COFF.
#
# make extcoff = Convert ELF to AVR Extended COFF.
#
# make program = Download the hex file to the device, using avrdude.
#                Please customize the avrdude settings below first!
#
# make dfu = Download the hex file to the device, using dfu-programmer (must
#            have dfu-programmer installed).
#
# make flip = Download the hex file to the device, using Atmel FLIP (must
#             have Atmel FLIP installed).
#
# make dfu-ee = Download the eeprom file to the device, using dfu-programmer
#               (must have dfu-programmer installed).
#
# make flip-ee = Download the eeprom file to the device, using Atmel FLIP
#                (must have Atmel FLIP installed).
#
# make doxygen = Generate DoxyGen documentation for the project (must have
#                DoxyGen installed)
#
# make debug = Start either simulavr or avarice as specified for debugging,
#              with avr-gdb or avr-insight as the front end for debugging.
#
# make filename.s = Just compile filename.c into the assembler code only.
#
# make filename.i = Create a preprocessed source file for use in submitting
#                   bug reports to the GCC project.
#
# To rebuild project do "make clean" then "make all".



# MCU name
```

```
MCU = atmega32u4


# Target architecture (see library "Board Types" documentation).
ARCH = AVR8


# Target board (see library "Board Types" documentation, NONE for projects not requiring
# LUFA board drivers). If USER is selected, put custom board drivers in a directory called
# "Board" inside the application directory.
BOARD = USER


# Processor frequency.
#     This will define a symbol, F_CPU, in all source code files equal to the
#     processor frequency in Hz. You can then use this symbol in your source code to
#     calculate timings. Do NOT tack on a 'UL' at the end, this will be done
#     automatically to create a 32-bit value in your source code.
#
#     This will be an integer division of F_USB below, as it is sourced by
#     F_USB after it has run through any CPU prescalers. Note that this value
#     does not *change* the processor frequency - it should merely be updated to
#     reflect the processor speed set externally so that the code can use accurate
#     software delays.
F_CPU = 16000000


# Input clock frequency.
#     This will define a symbol, F_USB, in all source code files equal to the
#     input clock frequency (before any prescaling is performed) in Hz. This value may
#     differ from F_CPU if prescaling is used on the latter, and is required as the
#     raw input clock is fed directly to the PLL sections of the AVR for high speed
#     clock generation for the USB and other AVR subsections. Do NOT tack on a 'UL'
#     at the end, this will be done automatically to create a 32-bit value in your
#     source code.
#
#     If no clock division is performed on the input clock inside the AVR (via the
#     CPU clock adjust registers or the clock division fuses), this will be equal to F_CPU.
F_USB = $(F_CPU)


# Output format. (can be srec, ihex, binary)
FORMAT = ihex


# Target file name (without extension).
TARGET = FullyDiff
```

```
# Object files directory
#     To put object files in current directory, use a dot (.), do NOT make
#     this an empty or blank macro!
OBJDIR = .


# Path to the LUFA library
LUFA_PATH = ../../../../atmega32u4/lufa-lib/trunk


# LUFA library compile-time options and predefined tokens
LUFA_OPTS = -D USB_DEVICE_ONLY
LUFA_OPTS += -D FIXED_CONTROL_ENDPOINT_SIZE=8
LUFA_OPTS += -D FIXED_NUM_CONFIGURATIONS=1
LUFA_OPTS += -D USE_FLASH_DESCRIPTORS
LUFA_OPTS += -D USE_STATIC_OPTIONS=\
"(USB_DEVICE_OPT_FULLSPEED | USB_OPT_REG_ENABLED | USB_OPT_AUTO_PLL)"


# Create the LUFA source path variables by including the LUFA root makefile
include $(LUFA_PATH)/LUFA/makefile


# List C source files here. (C dependencies are automatically generated.)
SRC = $(TARGET).c                                                    \
          Descriptors.c                                                \
          $(LUFA_SRC_USB)                                              \
          $(LUFA_SRC_USBCLASS)


# List C++ source files here. (C dependencies are automatically generated.)
CPPSRC =


# List Assembler source files here.
#     Make them always end in a capital .S.  Files ending in a lowercase .s
#     will not be considered source files but generated files (assembler
#     output from the compiler), and will be deleted upon "make clean"!
#     Even though the DOS/Win* filesystem matches both .s and .S the same,
#     it will preserve the spelling of the filenames, and gcc itself does
#     care about how the name is spelled on its command-line.
ASRC =


# Optimization level, can be [0, 1, 2, 3, s].
#     0 = turn off optimization. s = optimize for size.
#     (Note: 3 is not always the best optimization level. See avr-libc FAQ.)
```

```
OPT = s


# Debugging format.
#     Native formats for AVR-GCC's -g are dwarf-2 [default] or stabs.
#     AVR Studio 4.10 requires dwarf-2.
#     AVR [Extended] COFF format requires stabs, plus an avr-objcopy run.
DEBUG = dwarf-2


# List any extra directories to look for include files here.
#     Each directory must be seperated by a space.
#     Use forward slashes for directory separators.
#     For a directory that has spaces, enclose it in quotes.
EXTRAINCDIRS = $(LUFA_PATH)/


# Compiler flag to set the C Standard level.
#     c89   = "ANSI" C
#     gnu89 = c89 plus GCC extensions
#     c99   = ISO C99 standard (not yet fully implemented)
#     gnu99 = c99 plus GCC extensions
CSTANDARD = -std=c99


# Place -D or -U options here for C sources
CDEFS  = -DF_CPU=$(F_CPU)UL
CDEFS += -DF_USB=$(F_USB)UL
CDEFS += -DBOARD=BOARD_$(BOARD) -DARCH=ARCH_$(ARCH)
CDEFS += $(LUFA_OPTS)


# Place -D or -U options here for ASM sources
ADEFS  = -DF_CPU=$(F_CPU)
ADEFS += -DF_USB=$(F_USB)UL
ADEFS += -DBOARD=BOARD_$(BOARD)
ADEFS += $(LUFA_OPTS)

# Place -D or -U options here for C++ sources
CPPDEFS = -DF_CPU=$(F_CPU)UL
CPPDEFS += -DF_USB=$(F_USB)UL
CPPDEFS += -DBOARD=BOARD_$(BOARD)
CPPDEFS += $(LUFA_OPTS)
#CPPDEFS += -D__STDC_LIMIT_MACROS
#CPPDEFS += -D__STDC_CONSTANT_MACROS
```

```
#---------------- Compiler Options C ----------------
# -g*:          generate debugging information
# -O*:          optimization level
# -f ...:       tuning, see GCC manual and avr-libc documentation
# -Wall...:     warning level
# -Wa,...:      tell GCC to pass this to the assembler.
#   -adhlns...: create assembler listing
CFLAGS = -g$(DEBUG)
CFLAGS += $(CDEFS)
CFLAGS += -O$(OPT)
CFLAGS += -funsigned-char
CFLAGS += -funsigned-bitfields
CFLAGS += -ffunction-sections
CFLAGS += -fno-inline-small-functions
CFLAGS += -fpack-struct
CFLAGS += -fshort-enums
CFLAGS += -fno-strict-aliasing
CFLAGS += -Wall
CFLAGS += -Wstrict-prototypes
#CFLAGS += -mshort-calls
#CFLAGS += -fno-unit-at-a-time
#CFLAGS += -Wundef
#CFLAGS += -Wunreachable-code
#CFLAGS += -Wsign-compare
CFLAGS += -Wa,-adhlns=$(<:%.c=$(OBJDIR)/%.lst)
CFLAGS += $(patsubst %,-I%,$(EXTRAINCDIRS))
CFLAGS += $(CSTANDARD)


#---------------- Compiler Options C++ ----------------
# -g*:          generate debugging information
# -O*:          optimization level
# -f ...:       tuning, see GCC manual and avr-libc documentation
# -Wall...:     warning level
# -Wa,...:      tell GCC to pass this to the assembler.
#   -adhlns...: create assembler listing
CPPFLAGS = -g$(DEBUG)
CPPFLAGS += $(CPPDEFS)
CPPFLAGS += -O$(OPT)
CPPFLAGS += -funsigned-char
CPPFLAGS += -funsigned-bitfields
CPPFLAGS += -fpack-struct
CPPFLAGS += -fshort-enums
CPPFLAGS += -fno-exceptions
CPPFLAGS += -Wall
CPPFLAGS += -Wundef
#CPPFLAGS += -mshort-calls
#CPPFLAGS += -fno-unit-at-a-time
```

```
#CPPFLAGS += -Wstrict-prototypes
#CPPFLAGS += -Wunreachable-code
#CPPFLAGS += -Wsign-compare
CPPFLAGS += -Wa,-adhlns=$(<:%.cpp=$(OBJDIR)/%.lst)
CPPFLAGS += $(patsubst %,-I%,$(EXTRAINCDIRS))
#CPPFLAGS += $(CSTANDARD)


#---------------- Assembler Options ----------------
# -Wa,...:   tell GCC to pass this to the assembler.
# -adhlns:   create listing
# -gstabs:   have the assembler create line number information; note that
#            for use in COFF files, additional information about filenames
#            and function names needs to be present in the assembler source
#            files -- see avr-libc docs [FIXME: not yet described there]
# -listing-cont-lines: Sets the maximum number of continuation lines of hex
#       dump that will be displayed for a given single line of source input.
ASFLAGS = $(ADEFS) -Wa,-adhlns=$(<:%.S=$(OBJDIR)/%.lst),-gstabs,--listing-cont-
    lines=100


#---------------- Library Options ----------------
# Minimalistic printf version
PRINTF_LIB_MIN = -Wl,-u,vfprintf -lprintf_min

# Floating point printf version (requires MATH_LIB = -lm below)
PRINTF_LIB_FLOAT = -Wl,-u,vfprintf -lprintf_flt

# If this is left blank, then it will use the Standard printf version.
PRINTF_LIB =
#PRINTF_LIB = $(PRINTF_LIB_MIN)
#PRINTF_LIB = $(PRINTF_LIB_FLOAT)


# Minimalistic scanf version
SCANF_LIB_MIN = -Wl,-u,vfscanf -lscanf_min

# Floating point + %[ scanf version (requires MATH_LIB = -lm below)
SCANF_LIB_FLOAT = -Wl,-u,vfscanf -lscanf_flt

# If this is left blank, then it will use the Standard scanf version.
SCANF_LIB =
#SCANF_LIB = $(SCANF_LIB_MIN)
#SCANF_LIB = $(SCANF_LIB_FLOAT)


MATH_LIB = -lm
```

```
315
# List any extra directories to look for libraries here.
#     Each directory must be seperated by a space.
#     Use forward slashes for directory separators.
#     For a directory that has spaces, enclose it in quotes.
320 EXTRALIBDIRS =



#---------------- External Memory Options ----------------
325
# 64 KB of external RAM, starting after internal RAM (ATmega128!),
# used for variables (.data/.bss) and heap (malloc()).
#EXTMEMOPTS = -Wl,-Tdata=0x801100,--defsym=__heap_end=0x80ffff

330 # 64 KB of external RAM, starting after internal RAM (ATmega128!),
# only used for heap (malloc()).
#EXTMEMOPTS = -Wl,--section-start,.data=0x801100,--defsym=__heap_end=0x80ffff

EXTMEMOPTS =
335



#---------------- Linker Options ----------------
# -Wl,...:     tell GCC to pass this to linker.
340 #   -Map:     create map file
#   --cref:    add cross reference to map file
LDFLAGS = -Wl,-Map=$(TARGET).map,--cref
LDFLAGS += -Wl,--relax
LDFLAGS += -Wl,--gc-sections
345 LDFLAGS += $(EXTMEMOPTS)
LDFLAGS += $(patsubst %,-L%,$(EXTRALIBDIRS))
LDFLAGS += $(PRINTF_LIB) $(SCANF_LIB) $(MATH_LIB)
#LDFLAGS += -T linker_script.x

350



#---------------- Programming Options (avrdude) ----------------

# Programming hardware
355 # Type: avrdude -c ?
# to get a full listing.
#
AVRDUDE_PROGRAMMER = jtagmkII

360 # com1 = serial port. Use lpt1 to connect to parallel port.
AVRDUDE_PORT = usb
```

```
AVRDUDE_WRITE_FLASH = -U flash:w:$(TARGET).hex
#AVRDUDE_WRITE_EEPROM = -U eeprom:w:$(TARGET).eep


# Uncomment the following if you want avrdude's erase cycle counter.
# Note that this counter needs to be initialized first using -Yn,
# see avrdude manual.
#AVRDUDE_ERASE_COUNTER = -y

# Uncomment the following if you do /not/ wish a verification to be
# performed after programming the device.
#AVRDUDE_NO_VERIFY = -V

# Increase verbosity level. Please use this when submitting bug
# reports about avrdude. See <http://savannah.nongnu.org/projects/avrdude>
# to submit bug reports.
#AVRDUDE_VERBOSE = -v -v

AVRDUDE_FLAGS = -p $(MCU) -P $(AVRDUDE_PORT) -c $(AVRDUDE_PROGRAMMER)
AVRDUDE_FLAGS += $(AVRDUDE_NO_VERIFY)
AVRDUDE_FLAGS += $(AVRDUDE_VERBOSE)
AVRDUDE_FLAGS += $(AVRDUDE_ERASE_COUNTER)



#---------------- Debugging Options ----------------

# For simulavr only - target MCU frequency.
DEBUG_MFREQ = $(F_CPU)

# Set the DEBUG_UI to either gdb or insight.
# DEBUG_UI = gdb
DEBUG_UI = insight

# Set the debugging back-end to either avarice, simulavr.
DEBUG_BACKEND = avarice
#DEBUG_BACKEND = simulavr

# GDB Init Filename.
GDBINIT_FILE = __avr_gdbinit

# When using avarice settings for the JTAG
JTAG_DEV = /dev/com1

# Debugging port used to communicate between GDB / avarice / simulavr.
DEBUG_PORT = 4242

# Debugging host used to communicate between GDB / avarice / simulavr, normally
```

```
#     just set to localhost unless doing some sort of crazy debugging when
#     avarice is running on a different computer.
DEBUG_HOST = localhost


# Define programs and commands.
SHELL = sh
CC = avr-gcc
OBJCOPY = avr-objcopy
OBJDUMP = avr-objdump
SIZE = avr-size
AR = avr-ar rcs
NM = avr-nm
AVRDUDE = avrdude
REMOVE = rm -f
REMOVEDIR = rm -rf
COPY = cp
WINSHELL = cmd


# Define Messages
# English
MSG_ERRORS_NONE = Errors: none
MSG_BEGIN = --------- begin ---------
MSG_END = --------- end ---------
MSG_SIZE_BEFORE = Size before:
MSG_SIZE_AFTER = Size after:
MSG_COFF = Converting to AVR COFF:
MSG_EXTENDED_COFF = Converting to AVR Extended COFF:
MSG_FLASH = Creating load file for Flash:
MSG_EEPROM = Creating load file for EEPROM:
MSG_EXTENDED_LISTING = Creating Extended Listing:
MSG_SYMBOL_TABLE = Creating Symbol Table:
MSG_LINKING = Linking:
MSG_COMPILING = Compiling C:
MSG_COMPILING_CPP = Compiling C++:
MSG_ASSEMBLING = Assembling:
MSG_CLEANING = Cleaning project:
MSG_CREATING_LIBRARY = Creating library:


# Define all object files.
```

```
OBJ = $(SRC:%.c=$(OBJDIR)/%.o) $(CPPSRC:%.cpp=$(OBJDIR)/%.o) $(ASRC:%.S=$(OBJDIR)/%.o)

# Define all listing files.
LST = $(SRC:%.c=$(OBJDIR)/%.lst) $(CPPSRC:%.cpp=$(OBJDIR)/%.lst) $(ASRC:%.S=$(OBJDIR)
    /%.lst)


# Compiler flags to generate dependency files.
GENDEPFLAGS = -MMD -MP -MF .dep/$(@F).d


# Combine all necessary flags and optional flags.
# Add target processor to flags.
ALL_CFLAGS = -mmcu=$(MCU) -I. $(CFLAGS) $(GENDEPFLAGS)
ALL_CPPFLAGS = -mmcu=$(MCU) -I. -x c++ $(CPPFLAGS) $(GENDEPFLAGS)
ALL_ASFLAGS = -mmcu=$(MCU) -I. -x assembler-with-cpp $(ASFLAGS)




# Default target.
all: begin gccversion sizebefore build sizeafter end

# Change the build target to build a HEX file or a library.
build: elf hex eep lss sym
#build: lib


elf: $(TARGET).elf
hex: $(TARGET).hex
eep: $(TARGET).eep
lss: $(TARGET).lss
sym: $(TARGET).sym
LIBNAME=lib$(TARGET).a
lib: $(LIBNAME)



# Eye candy.
# AVR Studio 3.x does not check make's exit code but relies on
# the following magic strings to be generated by the compile job.
begin:
        @echo
        @echo $(MSG_BEGIN)

end:
        @echo $(MSG_END)
```

```
        @echo


    # Display size of file.
510 HEXSIZE = $(SIZE) --target=$(FORMAT) $(TARGET).hex
    ELFSIZE = $(SIZE) $(MCU_FLAG) $(FORMAT_FLAG) $(TARGET).elf
    MCU_FLAG = $(shell $(SIZE) --help | grep -- --mcu > /dev/null && echo --mcu=$(MCU)
        )
    FORMAT_FLAG = $(shell $(SIZE) --help | grep -- --format=.*avr > /dev/null && echo
        --format=avr )

515

    sizebefore:
        @if test -f $(TARGET).elf; then echo; echo $(MSG_SIZE_BEFORE); $(ELFSIZE); \
        2>/dev/null; echo; fi

520 sizeafter:
        @if test -f $(TARGET).elf; then echo; echo $(MSG_SIZE_AFTER); $(ELFSIZE); \
        2>/dev/null; echo; fi



525

    # Display compiler version information.
    gccversion :
        @$(CC) --version

530

    # Program the device.
    program: $(TARGET).hex $(TARGET).eep
        $(AVRDUDE) $(AVRDUDE_FLAGS) $(AVRDUDE_WRITE_FLASH) $(AVRDUDE_WRITE_EEPROM)

535 flip: $(TARGET).hex
        batchisp -hardware usb -device $(MCU) -operation erase f
        batchisp -hardware usb -device $(MCU) -operation loadbuffer $(TARGET).hex
            program
        batchisp -hardware usb -device $(MCU) -operation start reset 0

540 dfu: $(TARGET).hex
        -dfu-programmer $(MCU) erase
        dfu-programmer $(MCU) flash $(TARGET).hex
        dfu-programmer $(MCU) reset
        -dfu-programmer $(MCU) reset
545

    flip-ee: $(TARGET).hex $(TARGET).eep
        $(COPY) $(TARGET).eep $(TARGET)eep.hex
        batchisp -hardware usb -device $(MCU) -operation memory EEPROM erase
        batchisp -hardware usb -device $(MCU) -operation memory EEPROM loadbuffer $(
            TARGET)eep.hex program
```

```
	batchisp -hardware usb -device $(MCU) -operation start reset 0
	$(REMOVE) $(TARGET)eep.hex

dfu-ee: $(TARGET).hex $(TARGET).eep
	dfu-programmer $(MCU) eeprom-flash $(TARGET).eep
	dfu-programmer $(MCU) reset


# Generate avr-gdb config/init file which does the following:
#     define the reset signal, load the target file, connect to target, and set
#     a breakpoint at main().
gdb-config:
	@$(REMOVE) $(GDBINIT_FILE)
	@echo define reset >> $(GDBINIT_FILE)
	@echo SIGNAL SIGHUP >> $(GDBINIT_FILE)
	@echo end >> $(GDBINIT_FILE)
	@echo file $(TARGET).elf >> $(GDBINIT_FILE)
	@echo target remote $(DEBUG_HOST):$(DEBUG_PORT) >> $(GDBINIT_FILE)
ifeq ($(DEBUG_BACKEND),simulavr)
	@echo load >> $(GDBINIT_FILE)
endif
	@echo break main >> $(GDBINIT_FILE)

debug: gdb-config $(TARGET).elf
ifeq ($(DEBUG_BACKEND), avarice)
	@echo Starting AVaRICE - Press enter when "waiting_to_connect" message displays.
	@$(WINSHELL) /c start avarice --jtag $(JTAG_DEV) --erase --program --file \
	$(TARGET).elf $(DEBUG_HOST):$(DEBUG_PORT)
	@$(WINSHELL) /c pause

else
	@$(WINSHELL) /c start simulavr --gdbserver --device $(MCU) --clock-freq \
	$(DEBUG_MFREQ) --port $(DEBUG_PORT)
endif
	@$(WINSHELL) /c start avr-$(DEBUG_UI) --command=$(GDBINIT_FILE)



# Convert ELF to COFF for use in debugging / simulating in AVR Studio or VMLAB.
COFFCONVERT = $(OBJCOPY) --debugging
COFFCONVERT += --change-section-address .data-0x800000
COFFCONVERT += --change-section-address .bss-0x800000
COFFCONVERT += --change-section-address .noinit-0x800000
COFFCONVERT += --change-section-address .eeprom-0x810000

```

```
coff: $(TARGET).elf
	@echo
	@echo $(MSG_COFF) $(TARGET).cof
	$(COFFCONVERT) -O coff-avr $< $(TARGET).cof


extcoff: $(TARGET).elf
	@echo
	@echo $(MSG_EXTENDED_COFF) $(TARGET).cof
	$(COFFCONVERT) -O coff-ext-avr $< $(TARGET).cof



# Create final output files (.hex, .eep) from ELF output file.
%.hex: %.elf
	@echo
	@echo $(MSG_FLASH) $@
	$(OBJCOPY) -O $(FORMAT) -R .eeprom -R .fuse -R .lock $< $@

%.eep: %.elf
	@echo
	@echo $(MSG_EEPROM) $@
	-$(OBJCOPY) -j .eeprom --set-section-flags=.eeprom="alloc,load" \
	--change-section-lma .eeprom=0 --no-change-warnings -O $(FORMAT) $< $@ ||
	    exit 0

# Create extended listing file from ELF output file.
%.lss: %.elf
	@echo
	@echo $(MSG_EXTENDED_LISTING) $@
	$(OBJDUMP) -h -S -z $< > $@

# Create a symbol table from ELF output file.
%.sym: %.elf
	@echo
	@echo $(MSG_SYMBOL_TABLE) $@
	$(NM) -n $< > $@



# Create library from object files .
.SECONDARY : $(TARGET).a
.PRECIOUS : $(OBJ)
%.a: $(OBJ)
	@echo
	@echo $(MSG_CREATING_LIBRARY) $@
	$(AR) $@ $(OBJ)
```

```
645
    # Link: create ELF output file from object files.
    .SECONDARY : $(TARGET).elf
    .PRECIOUS : $(OBJ)
    %.elf: $(OBJ)
650         @echo
            @echo $(MSG_LINKING) $@
            $(CC) $(ALL_CFLAGS) $^ --output $@ $(LDFLAGS)


655 # Compile: create object files from C source files.
    $(OBJDIR)/%.o : %.c
            @echo
            @echo $(MSG_COMPILING) $<
            $(CC) -c $(ALL_CFLAGS) $< -o $@
660

    # Compile: create object files from C++ source files.
    $(OBJDIR)/%.o : %.cpp
            @echo
665         @echo $(MSG_COMPILING_CPP) $<
            $(CC) -c $(ALL_CPPFLAGS) $< -o $@


    # Compile: create assembler files from C source files.
670 %.s : %.c
            $(CC) -S $(ALL_CFLAGS) $< -o $@


    # Compile: create assembler files from C++ source files.
675 %.s : %.cpp
            $(CC) -S $(ALL_CPPFLAGS) $< -o $@


    # Assemble: create object files from assembler source files.
680 $(OBJDIR)/%.o : %.S
            @echo
            @echo $(MSG_ASSEMBLING) $<
            $(CC) -c $(ALL_ASFLAGS) $< -o $@


685
    # Create preprocessed source for use in sending a bug report.
    %.i : %.c
            $(CC) -E -mmcu=$(MCU) -I. $(CFLAGS) $< -o $@


690
    # Target: clean project.
    clean: begin clean_list end
```

```
clean_list :
	@echo
	@echo $(MSG_CLEANING)
	$(REMOVE) $(TARGET).hex
	$(REMOVE) $(TARGET).eep
	$(REMOVE) $(TARGET).cof
	$(REMOVE) $(TARGET).elf
	$(REMOVE) $(TARGET).map
	$(REMOVE) $(TARGET).sym
	$(REMOVE) $(TARGET).lss
	$(REMOVE) $(SRC:%.c=$(OBJDIR)/%.o) $(CPPSRC:%.cpp=$(OBJDIR)/%.o) $(ASRC:%.S=$(
	    OBJDIR)/%.o)
	$(REMOVE) $(SRC:%.c=$(OBJDIR)/%.lst) $(CPPSRC:%.cpp=$(OBJDIR)/%.lst) $(ASRC:%.S
	    =$(OBJDIR)/%.lst)
	$(REMOVE) $(SRC:.c=.s)
	$(REMOVE) $(SRC:.c=.d)
	$(REMOVE) $(SRC:.c=.i)
	$(REMOVEDIR) .dep

doxygen:
	@echo Generating Project Documentation \($(TARGET)\)...
	@if ( doxygen Doxygen.conf 2>&1 | grep ": warning:" ;); then \
	  exit 1; \
	fi;
	@echo Documentation Generation Complete.

clean_doxygen:
	rm -rf Documentation

checksource:
	@for f in $(SRC) $(CPPSRC) $(ASRC); do \
		if [ -f $$f ]; then \
			echo "Found Source File: $$f" ; \
		else \
			echo "Source File Not Found: $$f" ; \
		fi; done


# Create object files directory
$(shell mkdir $(OBJDIR) 2>/dev/null)


# Include the dependency files.
-include $(shell mkdir .dep 2>/dev/null) $(wildcard .dep/*)


# Listing of phony targets.
```

```
.PHONY : all begin finish end sizebefore sizeafter gccversion \
build elf hex eep lss sym coff extcoff doxygen clean      \
clean_list clean_doxygen program dfu flip flip-ee dfu-ee \
debug gdb-config checksource
```

A.2.2 FullyDiff.c

```
/*
  Fully Differential Standalone Sensor Code
  MIT 04/2012
  BJ Thompson
  Rev. 1
*/

/*
  Adapted from:

  LUFA Library
  Copyright (C) Dean Camera, 2012.

  dean [at] fourwalledcubicle [dot] com
  www.lufa-lib.org
*/

/*
  Copyright 2012 Dean Camera (dean [at] fourwalledcubicle [dot] com)

  Permission to use, copy, modify, distribute, and sell this
  software and its documentation for any purpose is hereby granted
  without fee, provided that the above copyright notice appear in
  all copies and that both that the copyright notice and this
  permission notice and warranty disclaimer appear in supporting
  documentation, and that the name of the author not be used in
  advertising or publicity pertaining to distribution of the
  software without specific, written prior permission.

  The author disclaim all warranties with regard to this
  software, including all implied warranties of merchantability
  and fitness. In no event shall the author be liable for any
  special, indirect or consequential damages or any damages
  whatsoever resulting from loss of use, data or profits, whether
  in an action of contract, negligence or other tortious action,
  arising out of or in connection with the use or performance of
  this software.
*/

/** \ file
```

```
 *
 *      Main source file for the Fully Differential Standalone Sensor board.
 *      This file contains the main resources for using the board:
 *       - hardware configuration and initialization
 *       - USB and virtual serial communication
 *
 */

#include "FullyDiff.h"
#include <stdlib.h>

/** LUFA CDC Class driver interface configuration and state information. This structure is
 *      passed to all CDC Class driver functions, so that multiple instances of the same class
 *      within a device can be differentiated from one another.
 */
USB_ClassInfo_CDC_Device_t FullyDiff_CDC_Interface =
  {
        .Config =
        {
          .ControlInterfaceNumber              = 0,

          .DataINEndpointNumber       = CDC_TX_EPNUM,
          .DataINEndpointSize         = CDC_TXRX_EPSIZE,
          .DataINEndpointDoubleBank           = false,

          .DataOUTEndpointNumber              = CDC_RX_EPNUM,
          .DataOUTEndpointSize        = CDC_TXRX_EPSIZE,
          .DataOUTEndpointDoubleBank         = false,

          .NotificationEndpointNumber       = CDC_NOTIFICATION_EPNUM,
          .NotificationEndpointSize           = CDC_NOTIFICATION_EPSIZE,
          .NotificationEndpointDoubleBank = false,
        },
  };

/** Standard file stream for the CDC interface when set up, so that the virtual CDC COM port
    can be
 *      used like any regular character stream in the C APIs
 */
static FILE USBSerialStream;

/* Global Variables */

char ADC_buf[5]; //Array to consolidate ADC conversions

char lastPress = 0;
bool debugFlag = true;
```

```
// Generic bad input message
char *ErrStr = "Bad Input.\r\n";

/** Main program entry point. This routine contains the overall program flow, including initial
 *      setup of all components and the main program loop.
 **/
int main(void)
{
  SetupHardware();

  /* Create a regular character stream for the interface so that it can be used with the stdio.h
       functions */
  CDC_Device_CreateStream(&FullyDiff_CDC_Interface, &USBSerialStream);

  //LEDs_SetAllLEDs(LEDMASK_USB_NOTREADY);
  sei();

  for (;;)
       {

         CheckPE2();

         /* Must throw away unused bytes from the host, or it will lock up while waiting for
              the device */

         uint16_t BytesReceived = CDC_Device_BytesReceived(&FullyDiff_CDC_Interface
              );
         if (BytesReceived)
       {

         int16_t recByte = CDC_Device_ReceiveByte(&FullyDiff_CDC_Interface);
         switch (recByte)
              {
              case 'A': // ADC Command
                {
              int16_t ADC_rec = 0;

              if (debugFlag)
                {
                     // Echo received command byte
                     CDC_Device_SendByte(&FullyDiff_CDC_Interface,recByte);
                     CDC_Device_SendString(&FullyDiff_CDC_Interface,": \r\n");
                }

              // Receive channel byte and check if valid
              ADC_rec = waitforByte();

              // Select a single ADC
```

```
        if (ADC_rec >= '0' && ADC_rec <= '6')
          {
                uint8_t chan = ADC_rec-'0';
                ADC_flush();
                ADC_read(chan);
                ADC_response(); // Responds with 4 bytes
          }

        // Select all ADCs
        else if (ADC_rec == 'a' || ADC_rec == 'A')
          {
                for (int i = 0; i < 6; i++)
                  {
                ADC_flush();
                ADC_read(i);
                ADC_response(); // Responds with 4*6=24 bytes
                  }
          }

        // (Re)Configure ADCn
        else if (ADC_rec == 'w')
          {
                // placeholder for writing to ADC
          }

        // Invalid input
        else
          reportError();
          }
          break;

        case 'd':
          {
        int16_t DDS_rec = 0;

        DDS_rec = waitforByte();
        if (checkHex(DDS_rec))
          {

                if (DDS_rec == '1')
                  {
                DDS_Config(6,0x00100000);
                DDS_Config(7,0x00100000);
                  }
                else if (DDS_rec == '2')
                  {
                DDS_Config(6,0x00800000);
                DDS_Config(7,0x00800000);
```

```
            }
          else if (DDS_rec == '3')
            {
          DDS_Config(6,0x00100000);
          DDS_Config(7,0x00800000);
            }
          else if (DDS_rec == '0')
            {
          DDS_Config(6,0);
          DDS_Config(7,0);
            }
  }
else
  {
        reportError();
        break;
  }

/*
  CDC_Device_SendByte(&FullyDiff_CDC_Interface,DDS_rec);
  CDC_Device_SendString(&FullyDiff_CDC_Interface,"\r\n");
*/
  }
  break;

case 'D': // DDS Command
  {
// Initialize DDS byte buffer
// 1 Byte channel, 4 Byte data = 5 bytes + NULL
char DDS_bytes[6];
int16_t DDS_rec = 0;
uint8_t DDS_chan = 0;
uint16_t DDS_com = 0;

// Wait for 4 incoming bytes and check if input is valid
for (int i = 0; i < 5; i++)
  {
        DDS_rec = waitforByte();
        if (checkHex(DDS_rec))
          {
        // Save input if valid
        DDS_bytes[i] = DDS_rec;
          }
        else
          {
        // Terminate buffer and exit if not valid
        DDS_bytes[i] = '\0';
        reportError();
```

```
            break;
              }
    }

  // Terminate buffer
  DDS_bytes[5]= '\0';

  // Extract, convert, and write DDS channel and command from input
  DDS_chan = (uint8_t) (DDS_bytes[0] & 0x0F);
  DDS_com = (uint16_t) strtoul(&DDS_bytes[1],NULL,16);
  DDS_Set(DDS_chan,DDS_com);

  // Echo received DDS instruction
  if (debugFlag)
    {
          fprintf(&USBSerialStream,"DDS:_Chan:%X_Command:%X\r\n",
              DDS_chan,DDS_com);
    }
    }
    break;
  case 'p':
    {
  int16_t pwm_rec = 0;

  pwm_rec = waitforByte();
  if (checkHex(pwm_rec))
    {
          if (pwm_rec == '1') // Set PWM chans to 41.6kHz
            {
          OCR0A = 0x17;
          OCR1A = 0x0060;
          OCR4C = 0xC0;
          OCR4B = OCR4C/2;
            }
          else if (pwm_rec == '2') // Set PWM chans to 333kHz
            {
          OCR0A = 0x02;
          OCR1A = 0x000C;
          OCR4C = 0x18;
          OCR4B = OCR4C/2;
            }
          else if (pwm_rec == '3') // PWM1: 41.6kHz, PWM2: 333kHz
            {
          OCR0A = 0x02;
          OCR1A = 0x0060;
          OCR4C = 0x18;
          OCR4B = OCR4C/2;
            }
```

```
  }
else
  {
        reportError();
        break;
  }
  }
  break;
case 'P': // PWM Command
  {
CDC_Device_SendByte(&FullyDiff_CDC_Interface,recByte);
CDC_Device_SendString(&FullyDiff_CDC_Interface,"\r\n");

char PWM_bytes[4];
int16_t PWM_rec = 0;

for (int i = 0; i < 3; i++)
  {
        PWM_rec = waitforByte();
        if (checkHex(PWM_rec))
          {
        PWM_bytes[i] = PWM_rec;
          }
        else
          {
        reportError();
        break;
          }
  }
PWM_bytes[3] = '\0';

if (PWM_bytes[0] == '1')
  {
        OCR0A = (uint8_t) strtoul(&PWM_bytes[1],NULL,16);
        fprintf(&USBSerialStream,"%d\r\n",(uint8_t) strtoul(&PWM_bytes
            [1],NULL,16));
  }
else if (PWM_bytes[0] == '2')
  {
        OCR1A = (uint8_t) strtoul(&PWM_bytes[1],NULL,16);
        fprintf(&USBSerialStream,"%d\r\n",(uint8_t) strtoul(&PWM_bytes
            [1],NULL,16));
  }
else if (PWM_bytes[0] == '3')
  {
        OCR4C = (uint8_t) strtoul(&PWM_bytes[1],NULL,16);
        OCR4B = OCR4C/2;
```

```
                        fprintf(&USBSerialStream,"%d\r\n",(uint8_t) strtoul(&PWM_bytes
                            [1],NULL,16));
                    }
                  else
                    {
                        reportError();
                        break;
                    }
                    }
                  break;
                }
        }
          else if (BytesReceived > 1)
        {
          char recBytes[BytesReceived+1];

          for (int i = 0; i < BytesReceived; i++)
                {
                  recBytes[i] = CDC_Device_ReceiveByte(&FullyDiff_CDC_Interface);
                }

          CDC_Device_SendString(&FullyDiff_CDC_Interface,recBytes);
        }
          CDC_Device_USBTask(&FullyDiff_CDC_Interface);
          USB_USBTask();
        }
}

void delay_ms(uint8_t count)
{
  while (count--)
        {
          _delay_ms(1);
        }
}

bool checkHex(int16_t data)
{
  if (data >= '0' && data <= '9')
        {
          return true;
        }
  else if (data >= 'a' && data <= 'f')
        {
          return true;
        }
  else if (data >= 'A' && data <= 'F')
        {
```

```
            return true;
          }
370     else
          return false;
}

void reportError(void)
375 {
  debugCheck();

  //Clear CDC buffer and exit if input not valid
  CDC_Device_Flush(&FullyDiff_CDC_Interface);
380   CDC_Device_SendString(&FullyDiff_CDC_Interface,ErrStr);

}

int16_t waitforByte(void)
385 {
  // Wait for an incoming byte
  while (!( CDC_Device_BytesReceived(&FullyDiff_CDC_Interface))) {continue;}

  // Receive incoming byte
390   return CDC_Device_ReceiveByte(&FullyDiff_CDC_Interface);
}

/* Configures the board hardware and chip peripherals */
void SetupHardware(void)
395 {
  /* Disable watchdog if enabled by bootloader/fuses */
  MCUSR &= ~(1 << WDRF);
  wdt_disable();
  MCUCR = (1 << JTD);
400   MCUCR = (1 << JTD);

  /* Disable clock division */
  clock_prescale_set(clock_div_1);

405   /* Hardware Initialization */
  //LEDs_Init();
  USB_Init();
  SPI_Init();

410   /* Set up pin direction for button */
  DDRE &= ~(1 << PE2);
  PORTE |= (1 << PE2);

  /* Set up pin directions for CS and disable all chips */
415   DDRD = 0xFF;
```

```
    PORTD = 0xFF;

    /* Set up PWM for chopper channels */
    PWM_Init();

    /* Set up PWM on Status LEDs */
    LEDs_PWMInit();

    /* Set up Debug Mode */
    DDRF &= ~(1 << PF7);
    PORTF |= (1 << PF7);
    debugCheck();

    DDS_Init();

}

/* Set up PWM pins on board */
void PWM_Init(void)
{
    /* Set up directions for PWM pins */
    DDRB |= (1 << PB5)|(1 << PB6)|(1 << PB7);
    DDRC |= (1 << PC6);

    /* Set up PWM for 8-bit TC0
            OC0A on PB7 connected, toggle on compare match
            WGM = 1 (0b010), TOP=OCRA0
            Prescaler: CLK/8
            TCCR0A = 0b01000010;
            TCCR0B = 0b00000010; */
    /* No prescaler: 0x00 =      8MHz, 0xFF = 31.2kHz
     * CLK/8:       0x00 =      1MHz, 0xFF = 3.94kHz
     * CLK/8:       0x02 = 333kHz, 0x18 = 40kHz
     * CLK/8:       0x17 = 41.6kHz
     */
    TCCR0A = 0;
    TCCR0A |= (1<<WGM01);
    TCCR0B |= (1<<CS01);
    OCR0A = 0x02;
    TCCR0A |= (1<<COM0A0);

    /* Set up PWM for 16-bit TC1, OC1A on PB5
            Use frequency and phase correct mode:
            - Mode: Toggle OC1A on compare match (OC1x disconnected)
            - WGM: Mode 9 (1001)
            - Prescaler: CLK/1 (none)
            - TCCR1A = 0b01000001;
            - TCCR1B = 0b00010001; */
```

```
  /* OCR1A: 400kHz = 0x0A, 333.33kHz = 0x0C, 40kHz = 0x64, 41.6kHz = 0x60 */
  // Chan: 40kHz
  TCCR1A = 0;
  TCCR1A = (1<<WGM10);
  TCCR1B = (1<<WGM13)|(1<<CS10);
  OCR1A = 0x0060;
  TCCR1A |= (1<<COM1A0);

  /* Set up PWM for 10-bit TC4, OC4B on PB6
        Use frequency and phase correct mode:
        - Mode:
        - WGM:
        - Prescaler: CLK/1 (none)
        - TCCR4A = 0b00100000
        - TCCR4B = 0b00000001
        - TCCR4C = 0b00100000
        - TCCR4D = 0b00000001 */
  /* OCR4C (TOP): 400kHz = 0x14, 40kHz = 0xC8,
        0CR4C (TOP): 333.33kHz = 0x18, 41.6kHz = 0xC0,
        OCR4B (MATCH): 50% Duty Cycle = OCR4C/2 */
  // Chan: 400kHz
  TCCR4A = (1<<COM4B1);
  TCCR4B = (1 << CS40);
  TCCR4C = (1<<COM4B1S);
  TCCR4D = (1<<WGM40);
  TCCR4A |= (1 << PWM4B); //Enable PWM on OC4B
  OCR4C = 0x18;
  OCR4B = OCR4C/2;

  /* Set up PWM for 4MHz DDS clock on TC3,
   * Using OC3A on PC6 CTC mode,
   * OCR3A: 0x01 = 4MHz, 0x02 = 2.67MHz, 0x18 = 320kHz
   */
  TCCR3A = 0;
  TCCR3B = (1<<WGM32)|(1<<CS30);
  OCR3A = 0x02;
  TCCR3A |= (1<<COM3A0);
}

void SPI_Init(void)
{
  /* Set up SPI Data Directions:
        - !SS  (PB0): output
        - MOSI (PB1): output
        - SCK  (PB2): output
        - MISO (PB3): input
        explicitly  set  !SS as output to ensure SPI configuration */
  //DDR_SPI = (1<<DD_MOSI)|(1<<DD_SCK);
```

```
  DDRB = ((1 << PB0) | (1 << PB1) | (1 << PB2));
  PORTB |= (1 << PB0);

  /* Enable SPI as Master with clock rate fck/16
        Transmit MSB first (DORD=0), SCK low when idle (CPOL=0)
        Sample data on trailing ( falling ) edge (CPHA=1) */
  //SPCR = (1<<SPE)|(1<<MSRT)|(1<<CPHA)|(1<<SPR0);
  SPCR = 0b01010101;
}

uint8_t SPI_transmit(uint8_t data)
{
  /* Send data */
  SPDR = data;

  /* Wait for transmission complete */
  while (!(SPSR & (1<<SPIF)));

  /* Return received byte */
  return SPDR;
}

void ADC_read(uint8_t chan)
{
  /* Sample data on trailing edge (CPHA=1)
        May need to reconfigure sample edge depending on how MOSI data works */
  //SPCR = 0b01010101;
  SPCR = 0b01011101;

  /* Chip Select Enable */
  PORTD &= ~(1 << chan);

  /* Send dummy bytes over SPI and read back 4 byte conversion */
  for (int i = 0; i < 4; i++)
      {
        ADC_buf[i] = SPI_transmit(0x00);
      }

  /* Chip Select Disable */
  PORTD |= (1 << chan);
}

void ADC_flush(void)
{
  /* Clear ADC temp bytes and buffer*/
  for (int i = 0; i < 4; i++)
      {
        ADC_buf[i] = 0;
```

```
        }
}

void DDS_Set(uint8_t chan, uint16_t data)
{
  /* Ensure that only DDS can be selected */
  if ((chan == PD6) || (chan == PD7))
        {
          unsigned char dataH = (data >> 8);
          unsigned char dataL = (data & 0xFF);

          /* Shift data on FALLING edge of SCK
         SCK idles HIGH when FSYNC/CS goes low (mode 3) */
          SPCR = 0b01011001;

          /* Chip Select Enable */
          PORTD &= ~(1 << chan);

          /* Send DDS setup data (no data to read back) */
          SPI_transmit(dataH);
          SPI_transmit(dataL);

          /* Chip Select Disable */
          PORTD |= (1 << chan);
        }
}

/* Configures DDS with a 10kHz and 100kHz sine wave on Ch. A and B */
void DDS_Init(void)
{
  /* Reset and Initialize */
  DDS_Set(PD6, 0x2100);
  //DDS_Set(PD7, 0x2100);

  /* Write to frequency register (Fout,Fin):
   * Fmclk = 320kHz:
   *  10kHz = 0x00800000,
   *
   * Fmclk = 2.67MHz:
   *  10.416kHz = 0x00100000,
   *  83.3kHz   = 0x00800000,
   *  125kHz    = 0x00C00000,
   *
   * (~10khz, 4MHz) = 0x000A3D70, (~100kHz, 4MHz) = 0x00666666
   * (~10kHz, 5MHz) = 0x00083126, (~100kHz, 5MHz) = 0x0051EB85
   */
  DDS_Freq(PD6,0,0x00800000);
```

```
  /* Write PHASE0 Reg */
  DDS_Phase(PD6,0,0);

  /* Exit Reset and Begin Output */
  DDS_Set(PD6, 0x2000);
  //DDS_Set(PD7, 0x2000);
}

void DDS_Config(uint8_t chan, uint32_t freq)
{
  DDS_Set(chan, 0x2100);
  DDS_Freq(chan,0,freq);
  DDS_Phase(chan,0,0);
  DDS_Set(chan,0x2000);
}

void DDS_Freq(uint8_t en, uint16_t reg, uint32_t freq)
{
  /*
   *
   *
   * freg is the 28 bit result of the following:
   *  freg = (fout/fmclk)*2^28
   */

  reg = (reg + 1) & 0x03;

  uint16_t fregL = (reg << 14) | (0x3FFF & freg);
  uint16_t fregH = (reg << 14) | (0x3FFF & (freg >> 14));

  DDS_Set(en,fregL);
  DDS_Set(en,fregH);

  //DDS_Set(en,0x4000);
  //DDS_Set(en,0x4040);
}

void DDS_Phase(uint8_t en, uint16_t reg, uint16_t preg)
{
  /*
   * The phase registers are written to with a 16-bit word.
   * D15 and D14 are always 1, D13 selects PHASE0 or PHASE1
   * registers . D12 doesn't matter. The remaining 12 bits
   * are preg.
   *
   * preg is the 12 bit result of the following:
   *  preg = (phase_shift*2^12)/(2*pi)
   */
```

```
  reg = (reg & 0x01) << 13;
  preg = (3 << 14) | reg | (0x0FFF & preg);

  DDS_Set(en,preg);
}

void debugCheck(void)
{
  uint8_t PF7Stat = (PINF & (1 << PF7));

  if (!PF7Stat)
      {
        debugFlag = true;
      }
  else
      debugFlag =    false;
}

void CheckPE2(void)
{
  uint8_t PE2Stat = (PINE & (1 << PE2));
  char*   ReportString = NULL;
  static bool ActionSent = false;

  if (!PE2Stat)
      {
        ADC_flush();
        ADC_read(1);

        ReportString = "Button Down\r\n";
      }
  else
      ActionSent = false;

  if ((ReportString != NULL) && (ActionSent == false))
      {
        ActionSent = true;

        CDC_Device_SendString(&FullyDiff_CDC_Interface, ReportString);
        ADC_response();
      }
}

void ADC_response(void)
{
  char ADC_chars[5];
```

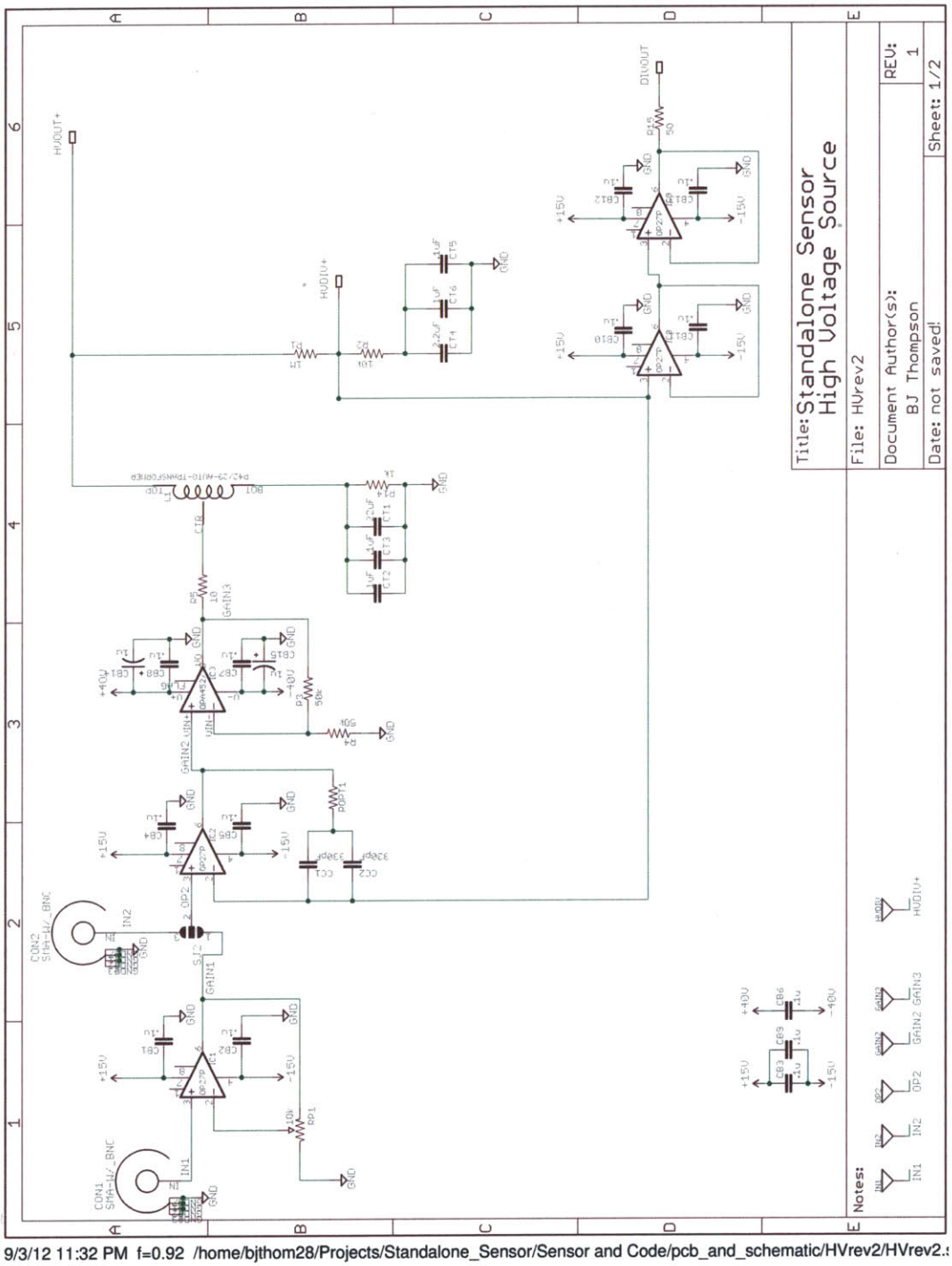
```
  // There are two methods for sending data over serial:
  // Method 1: Using the CDC_Interface
  for (int i = 0; i < 4; i++)
        {
          sprintf(ADC_chars,"%02X",ADC_buf[i]);
          CDC_Device_SendString(&FullyDiff_CDC_Interface, ADC_chars);
        }
  CDC_Device_SendString(&FullyDiff_CDC_Interface,"\r\n");

  // Method 2: Using the USBSerialStream
  /*for (int i = 0; i < 4; i++)
        {
        fprintf(&USBSerialStream,"%XX",ADC_buf[i]);
        }
        CDC_Device_SendString(&FullyDiff_CDC_Interface,"\r\n");*/
}

/** Event handler for the library USB Connection event. */
void EVENT_USB_Device_Connect(void)
{
  //LEDs_SetAllLEDs(LEDMASK_USB_ENUMERATING);
  LEDs_SetBrightness(LEDS_BLU,LOW);
}

/** Event handler for the library USB Disconnection event. */
void EVENT_USB_Device_Disconnect(void)
{
  //LEDs_SetAllLEDs(LEDMASK_USB_NOTREADY);
  LEDs_SetBrightness(LEDS_BLU,HALF);
}

/** Event handler for the library USB Configuration Changed event. */
void EVENT_USB_Device_ConfigurationChanged(void)
{
  bool ConfigSuccess = true;

  ConfigSuccess &= CDC_Device_ConfigureEndpoints(&FullyDiff_CDC_Interface);

  //LEDs_SetAllLEDs(ConfigSuccess ? LEDMASK_USB_READY : LEDMASK_USB_ERROR);
  LEDs_SetBrightness(LEDS_BLU,ConfigSuccess ? LOW : HIGH);
}

/** Event handler for the library USB Control Request reception event. */
void EVENT_USB_Device_ControlRequest(void)
{
  CDC_Device_ProcessControlRequest(&FullyDiff_CDC_Interface);
}
```

Appendix B

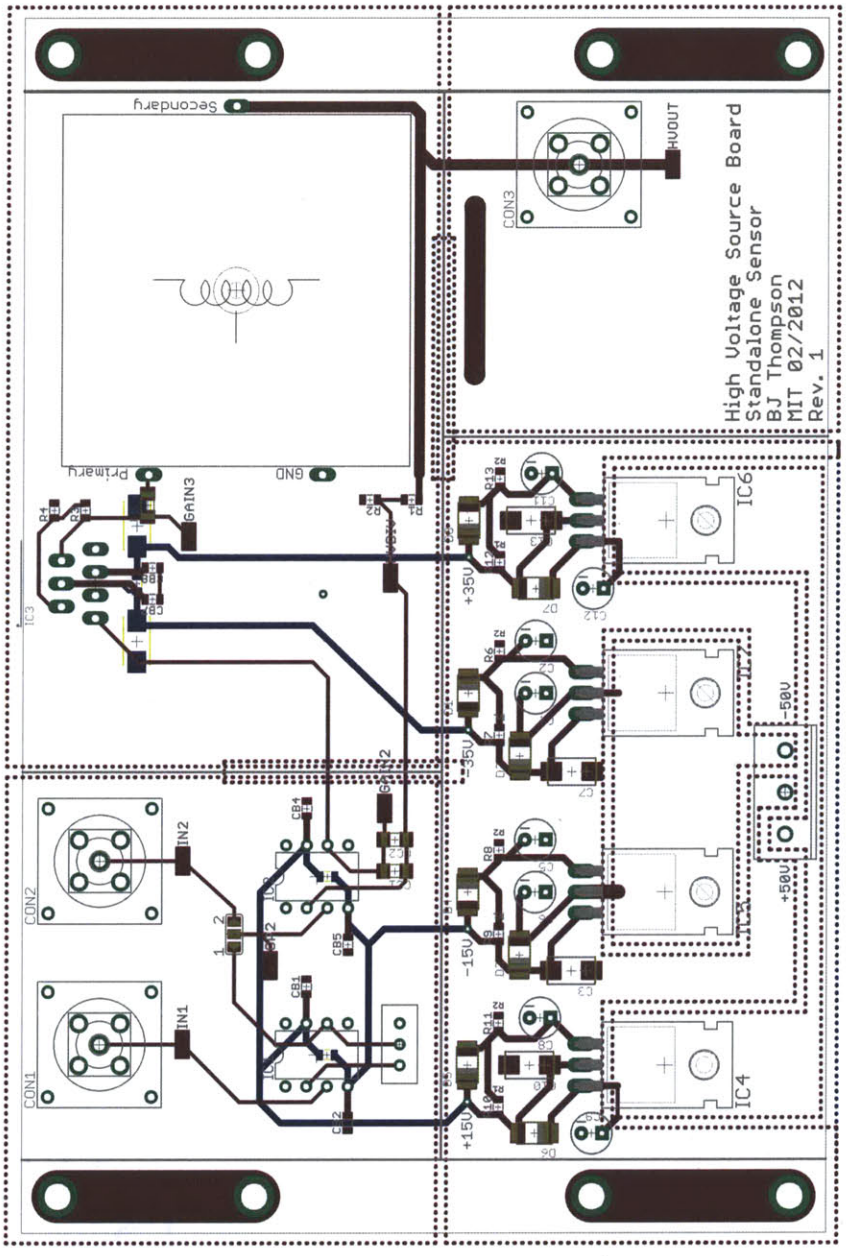
Schematics and Boards

B.1 High Voltage Signal Source

B.1.1 Full Schematic



B.1.2 PCB Layout



9/3/12 11:33 PM f=1.54 /home/bjthom28/Projects/Standalone_Sensor/Sensor and Code/pcb_and_schematic/HVrev1/HVrev1.l

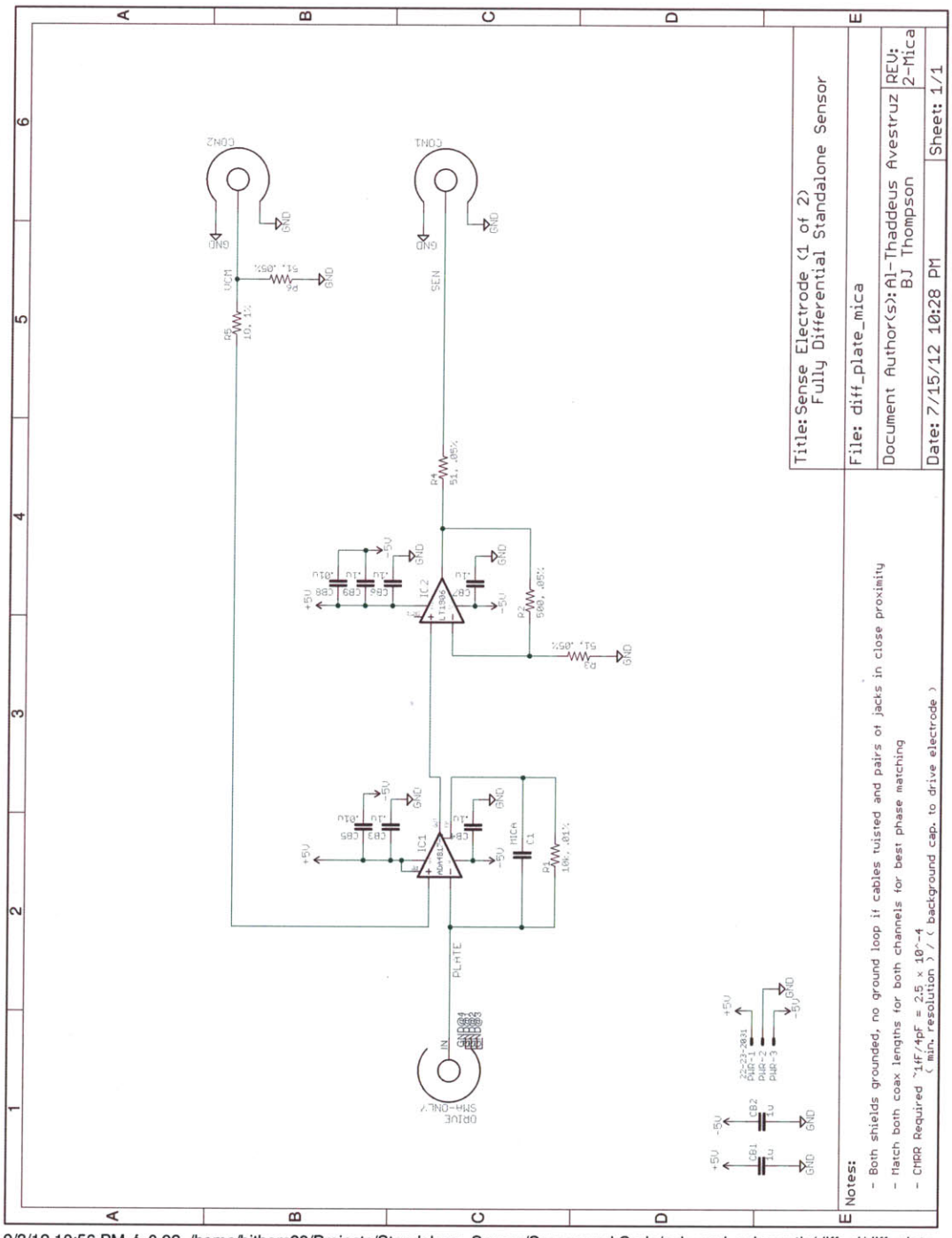
B.1.3 Bill of Materials

Qty	Value	Parts
1	15mH	L1
8	DIODE-DO-214AC	D1, D2, D3, D4, D5, D6, D7, D8
1	10	ROPT1
1	Jumper	SJ2
1	Terminal	PWR
13	.1u	CB1, CB2, CB3, CB4, CB5, CB6, CB7, CB8, CB9, CB10, CB11, CB12, CB13
2	.1uF	CT3, CT5
2	1M	R1, R16
1	1k	R14
6	1u	C3, C7, C10, C13, CB14, CB15
2	1uF	CT2, CT6
1	2.2uF	CT4
1	10	R5
1	10k	RP1
2	10k	R2, R17
8	10u	C2, C4, C5, C6, C8, C9, C11, C12
1	22uF	CT1
1	50	R15
2	50k	R3, R4
2	317T	IC4, IC6
2	330pF	CC1, CC2
2	337T	IC5, IC7
4	OP27P	IC1, IC2, IC8, IC9
1	OPA452/3S	IC3
4	R1	R7, R9, R10, R12
4	R2	R6, R8, R11, R13
2	BNC	CON1, CON2
6	TPT-SMD	GAIN2, GAIN3, HVDIV, IN1, IN2, OP2

B.2 Standalone Sensor Electrodes

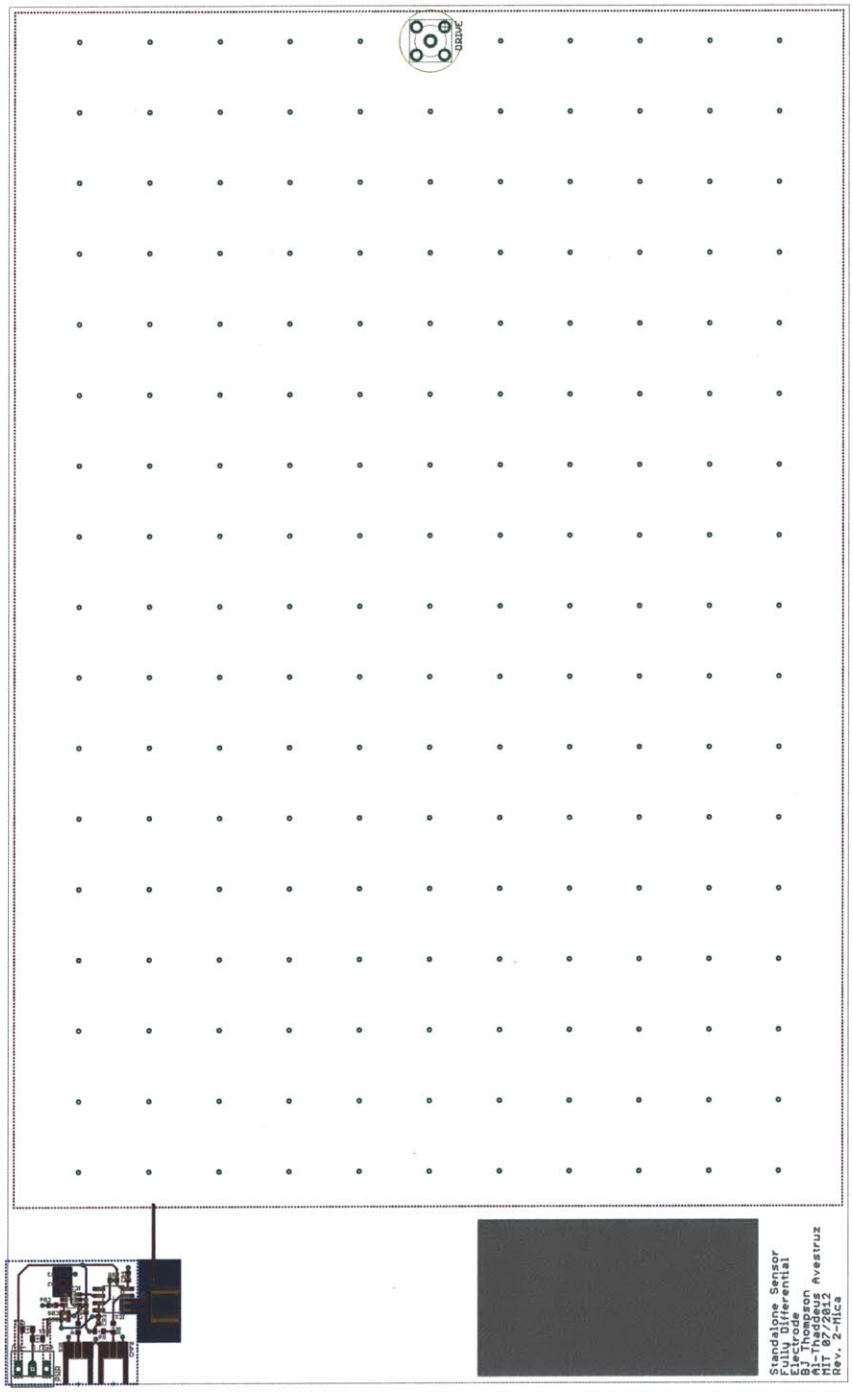
B.2.1 Capacitive Mode Electrodes

Full Schematic

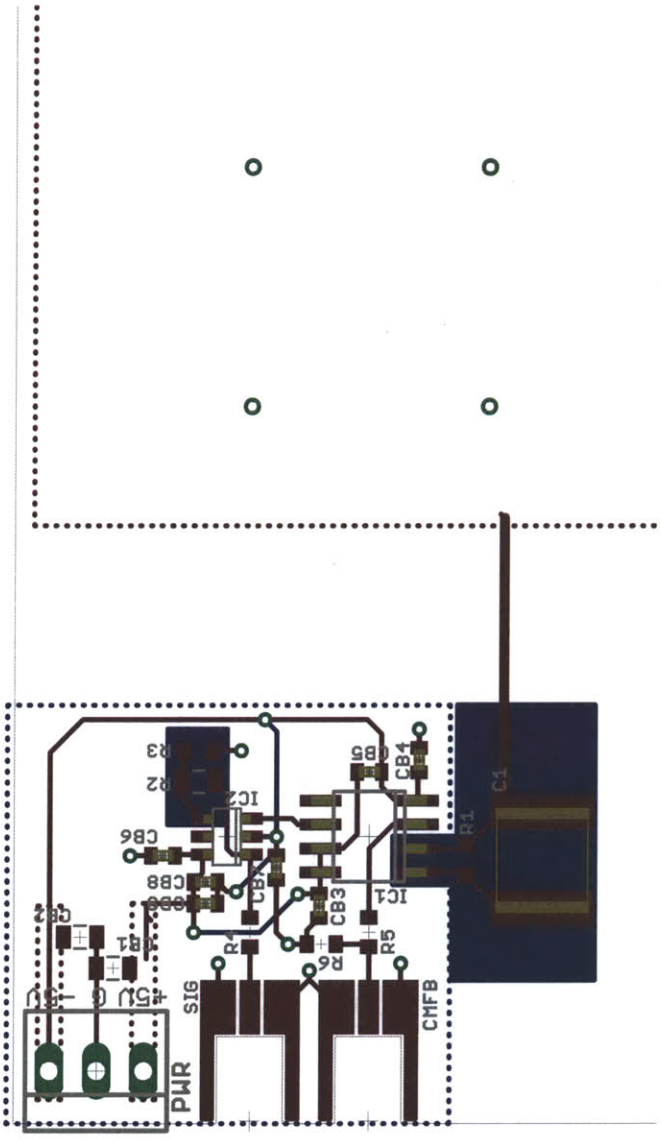


9/3/12 10:56 PM f=0.92 /home/bjthom28/Projects/Standalone_Sensor/Sensor and Code/pcb_and_schematic/diff_al/diff_plate_

PCB Layout



9/3/12 10:57 PM f=1.04 /home/bjthom28/Projects/Standalone_Sensor/Sensor and Code/pcb_and_schematic/diff_al/diff_plate_



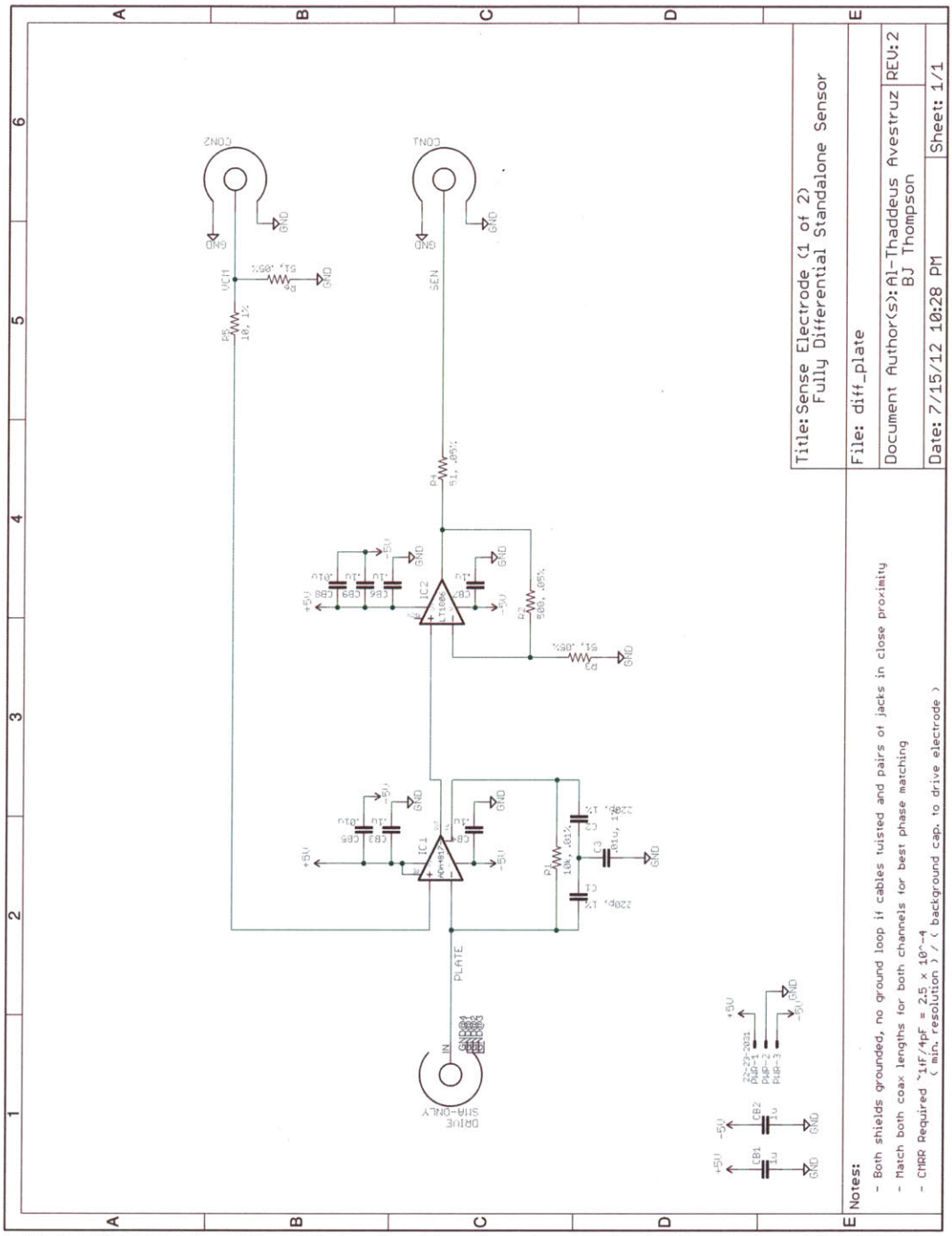
9/3/12 11:05 PM f=3.51 /home/bjthom28/Projects/Standalone_Sensor/Sensor and Code/pcb_and_schematic/diff_al/diff_plate_

Bill of Materials

Qty	Value	Parts
5	.1u	CB3, CB4, CB6, CB7, CB9
2	.01u	CB5, CB8
2	1u	CB1, CB2
1	10, 1%	R5
1	10k, .01%	R1
1	22-23-2031	PWR
3	51, .05%	R3, R4, R6
1	500, .05%	R2
1	ADA4817-1	IC1
1	LT1806	IC2
1	MICA	C1
2	MMXC-CUT	CON1, CON2
1	SMA-ONLY	DRIVE

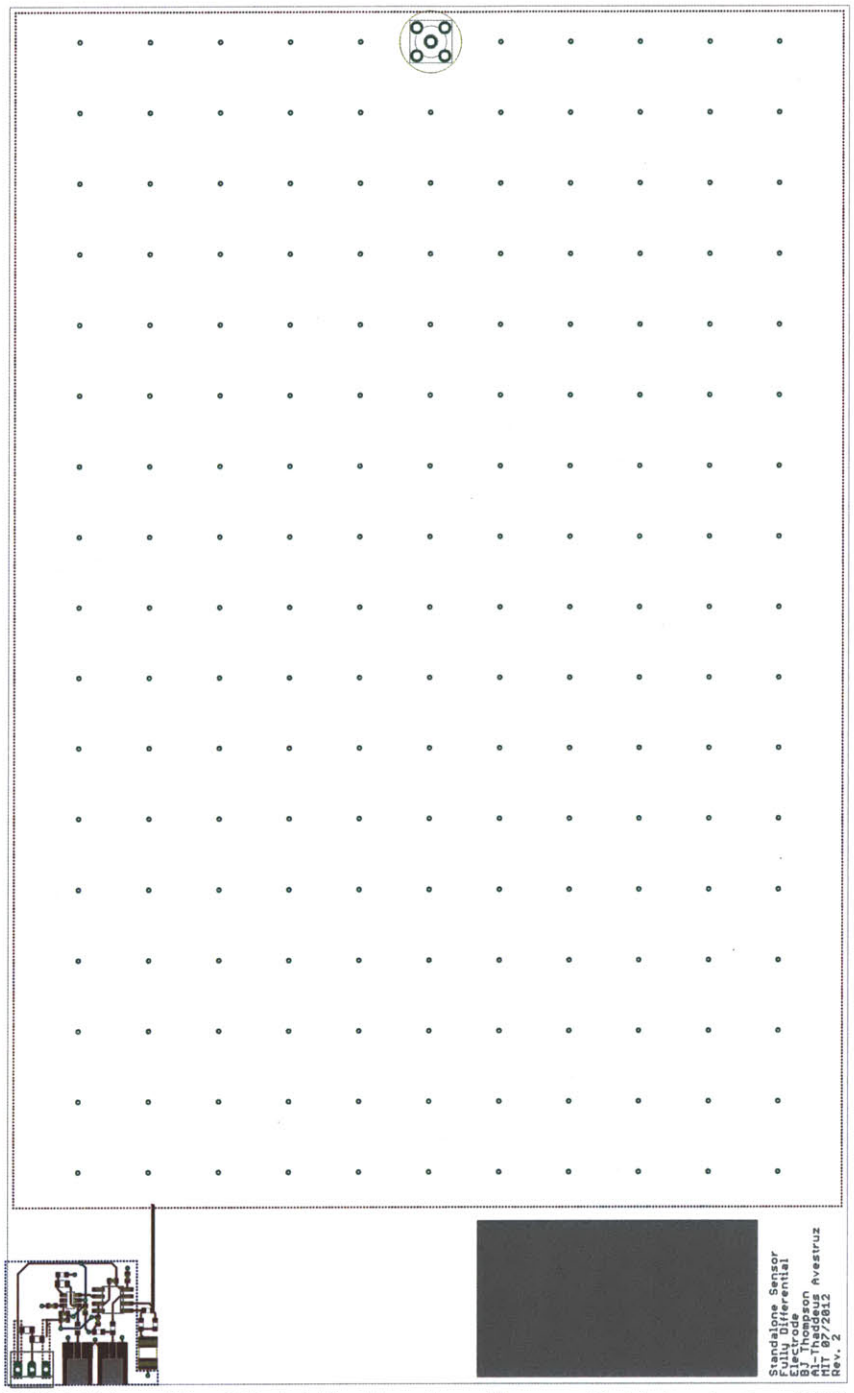
B.2.2 Resistive Mode Electrodes

Full Schematic

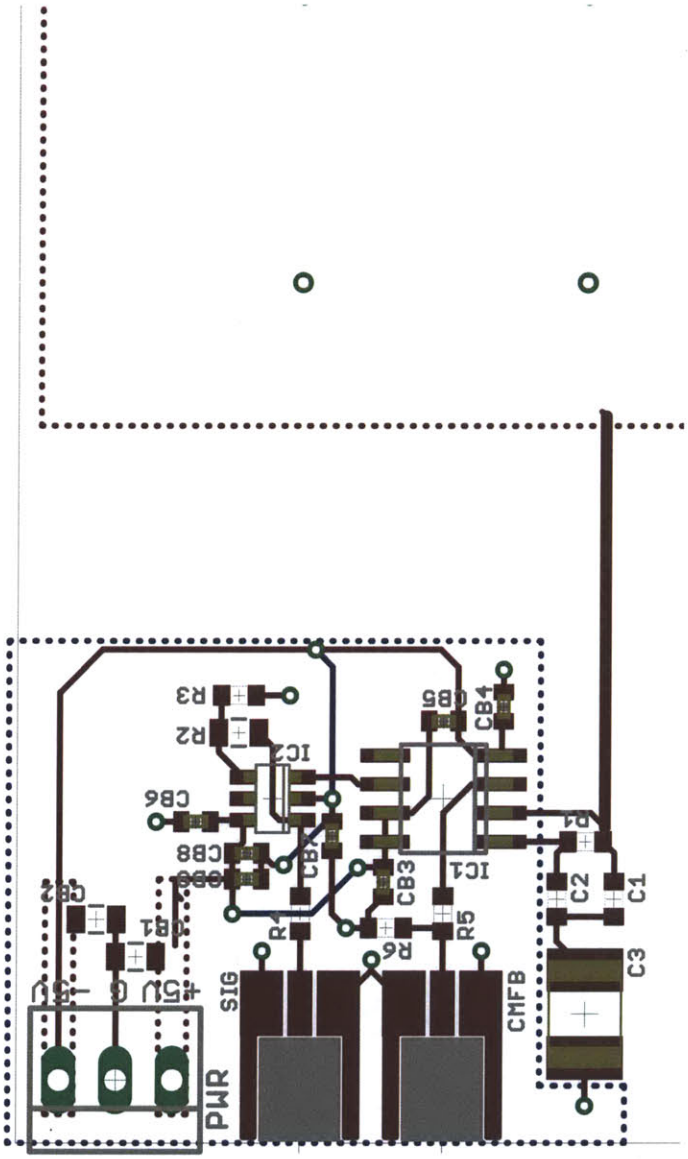


9/3/12 10:58 PM f=0.92 /home/bjthom28/Projects/Standalone_Sensor/Sensor and Code/pcb_and_schematic/diff_al/diff_plate.s

PCB Layout



9/3/12 10:58 PM f=1.04 /home/bjthom28/Projects/Standalone_Sensor/Sensor and Code/pcb_and_schematic/diff_al/diff_plate.l



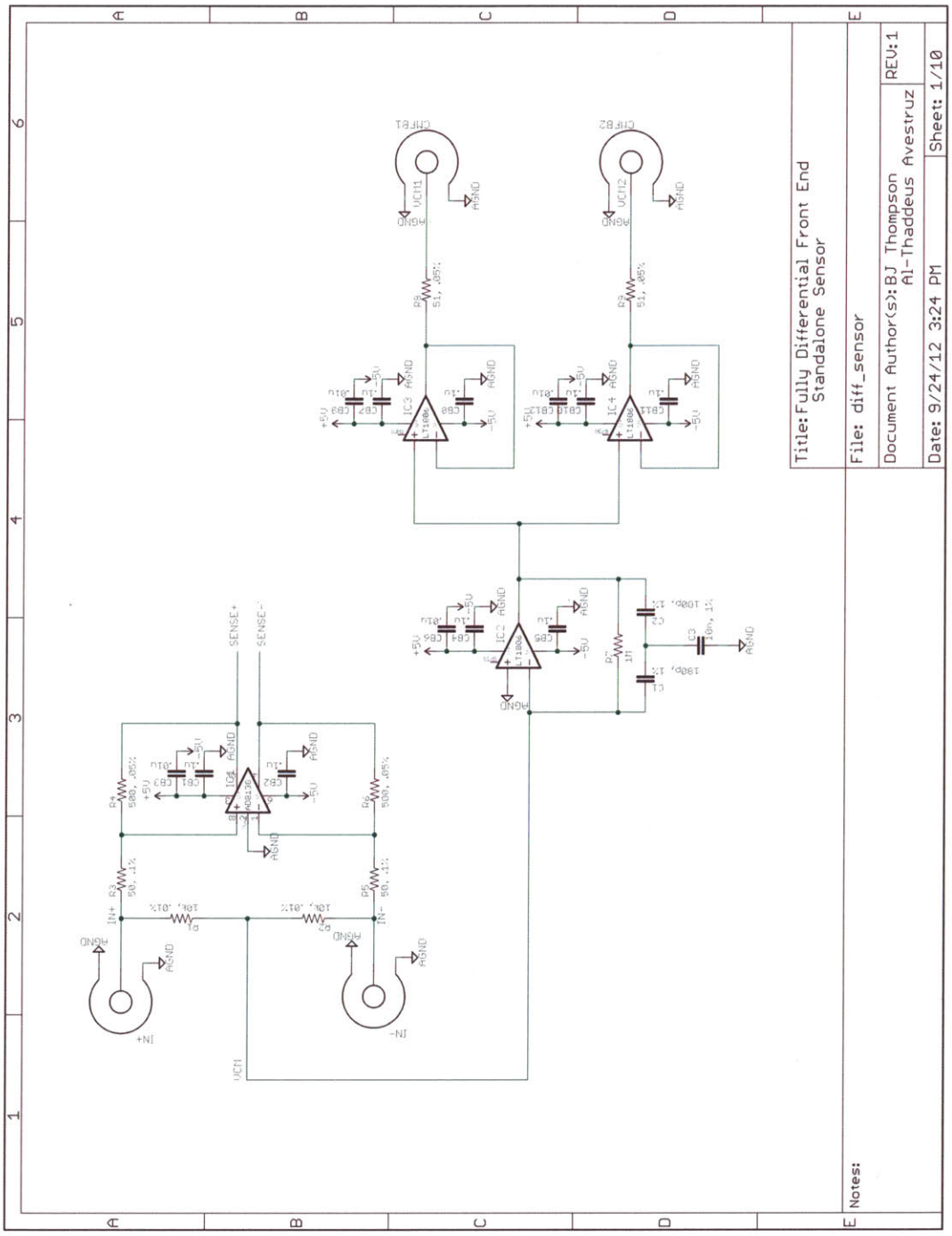
9/3/12 11:04 PM f=4.22 /home/bjthom28/Projects/Standalone_Sensor/Sensor and Code/pcb_and_schematic/diff_al/diff_plate.l

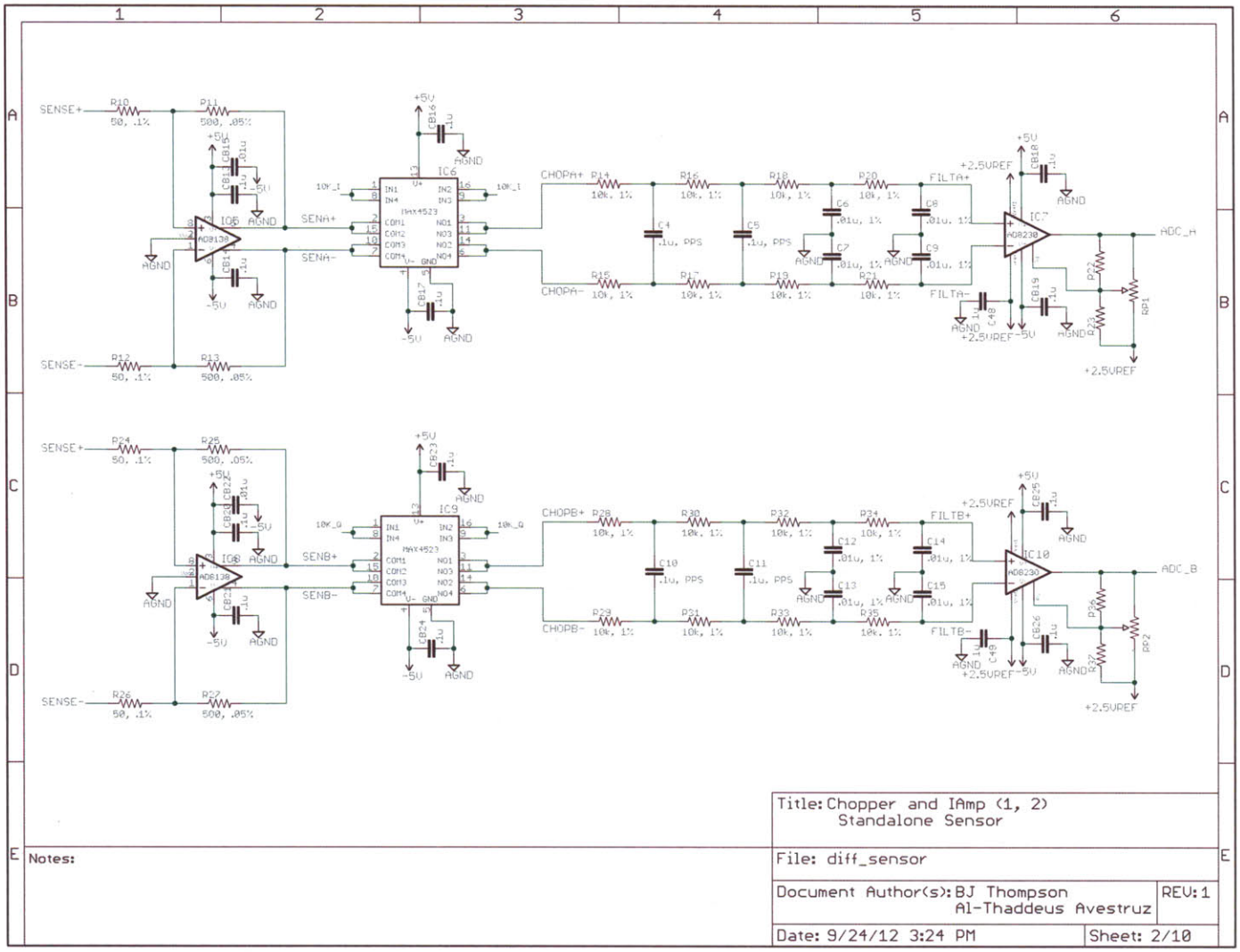
Bill of Materials

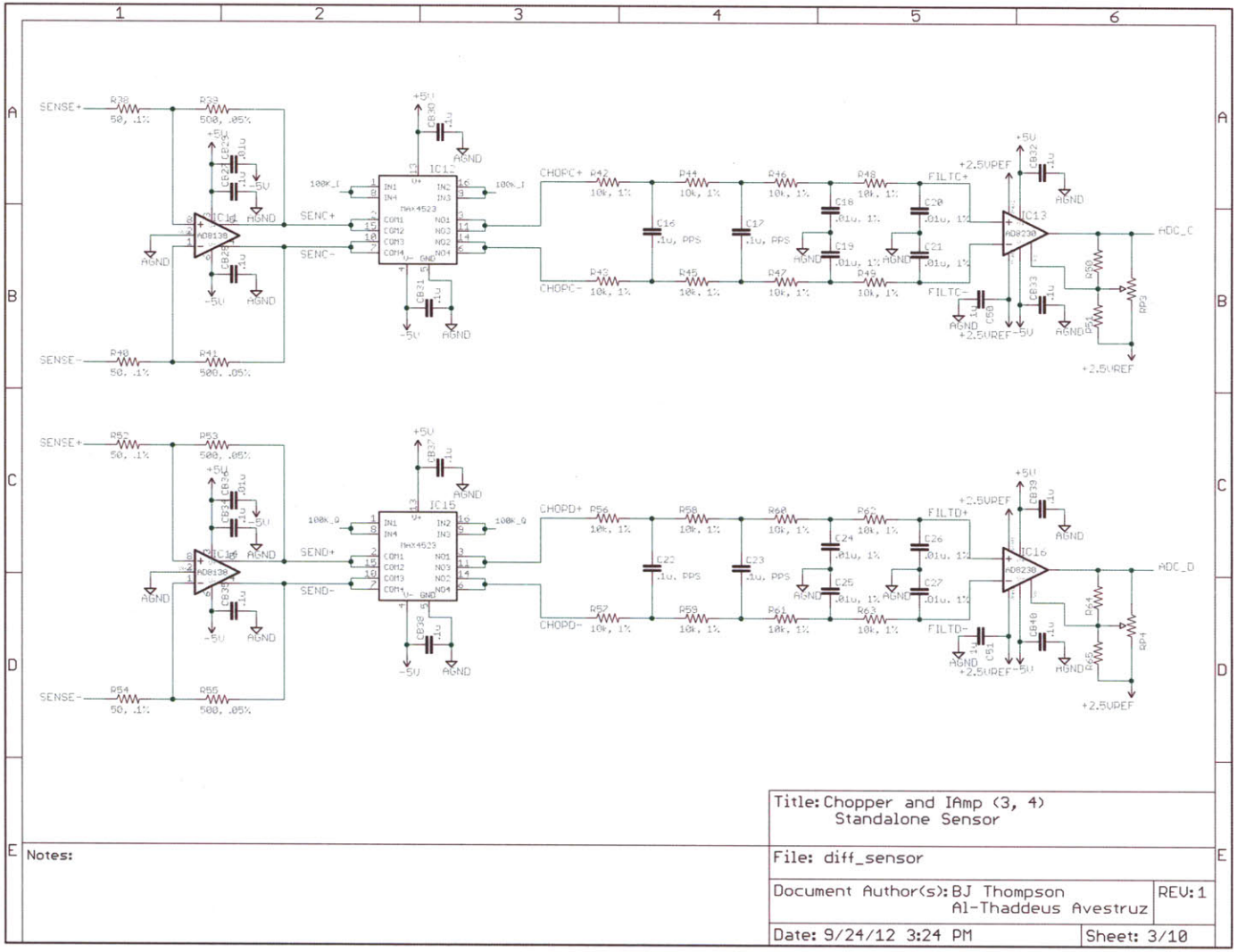
Qty	Value	Parts
5	.1u	CB3, CB4, CB6, CB7, CB9
2	.01u	CB5, CB8
1	.01u, 1%	C3
2	1u	CB1, CB2
1	10, 1%	R5
1	10k, .01%	R1
1	22-23-2031	PWR
3	51, .05%	R3, R4, R6
2	220p, 1%	C1, C2
1	500, .05%	R2
1	ADA4817-1	IC1
1	LT1806	IC2
2	MMXC-CUT	CON1, CON2
1	SMA-ONLY	DRIVE

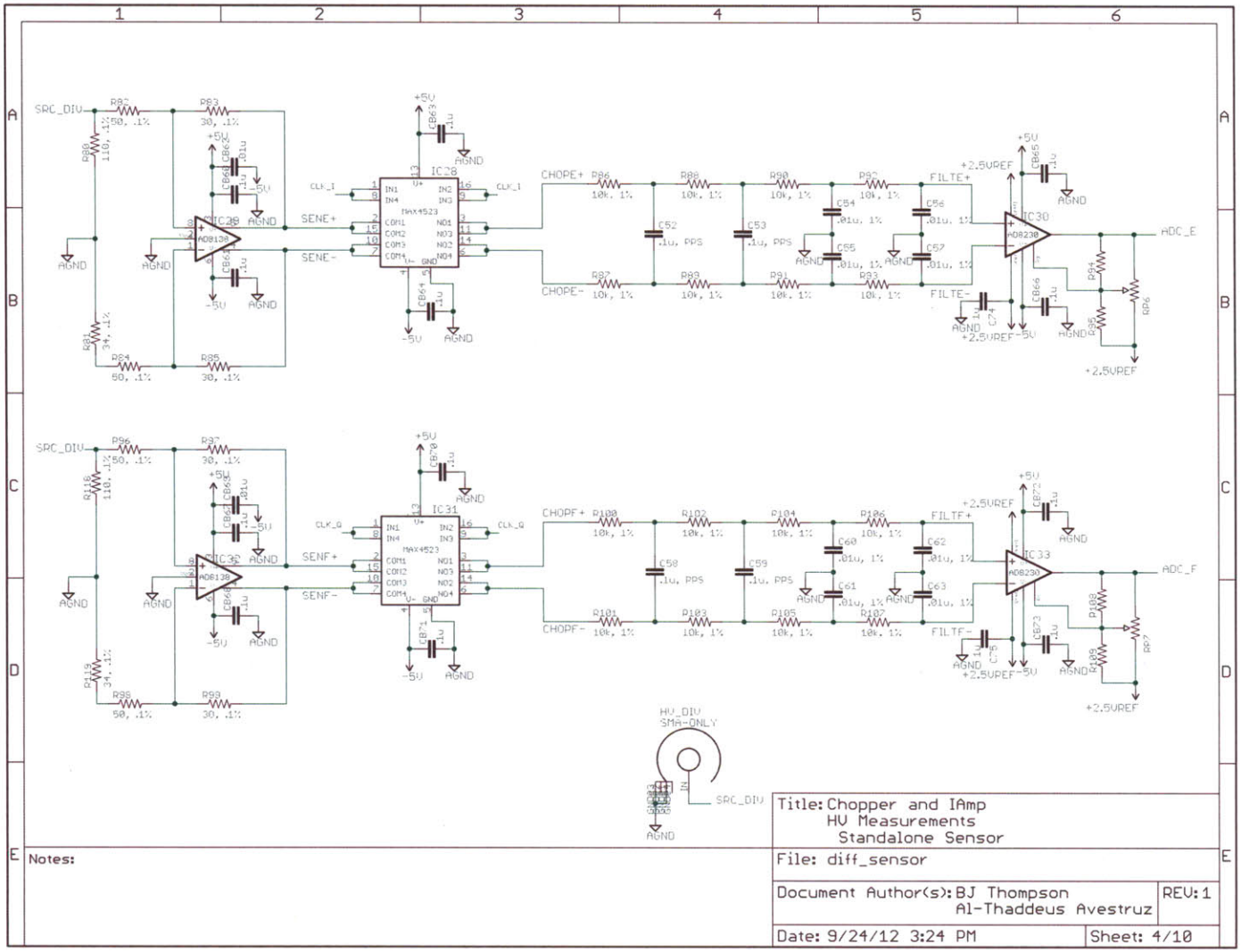
B.3 Standalone Sensor Signal Conditioning

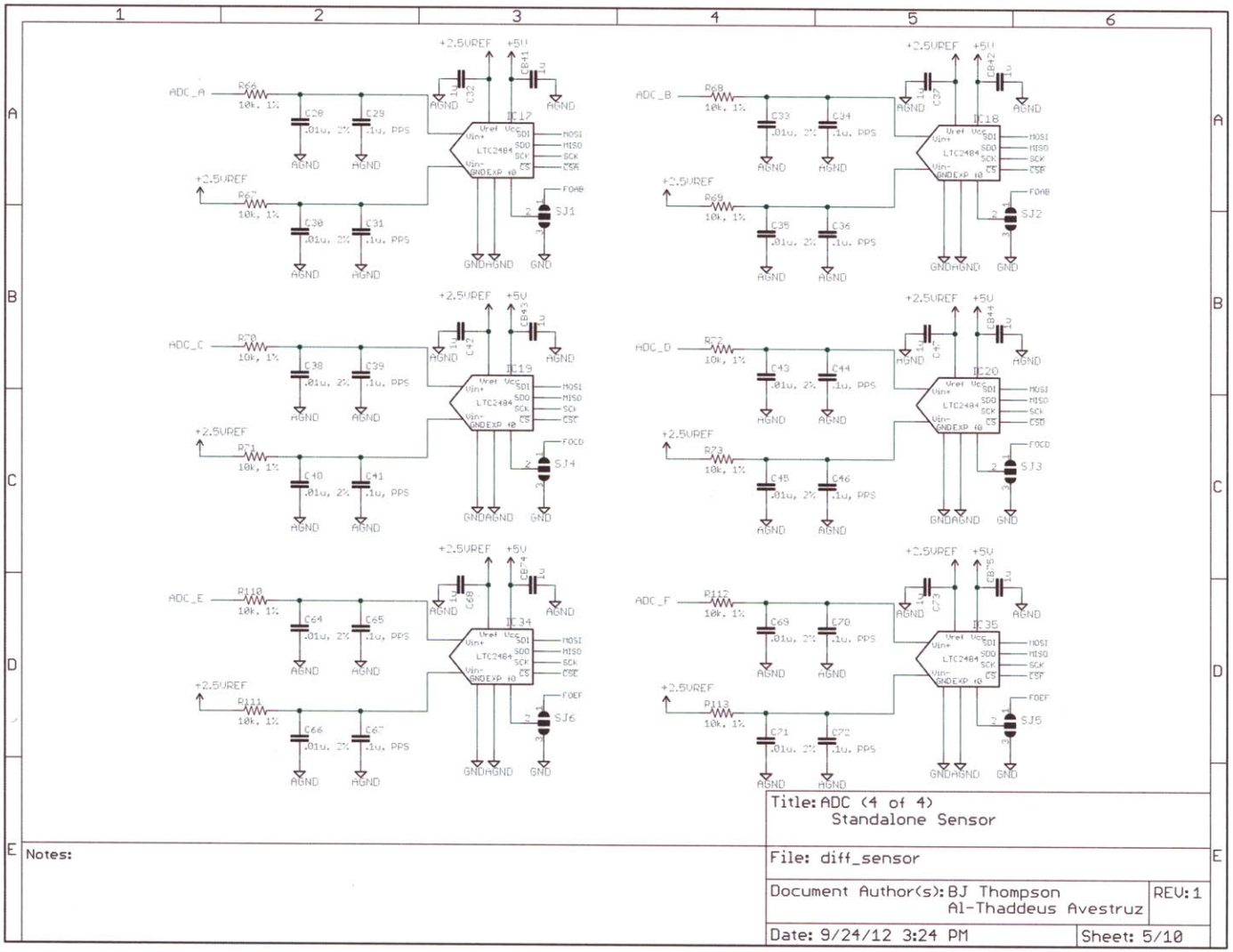
B.3.1 Full Schematic

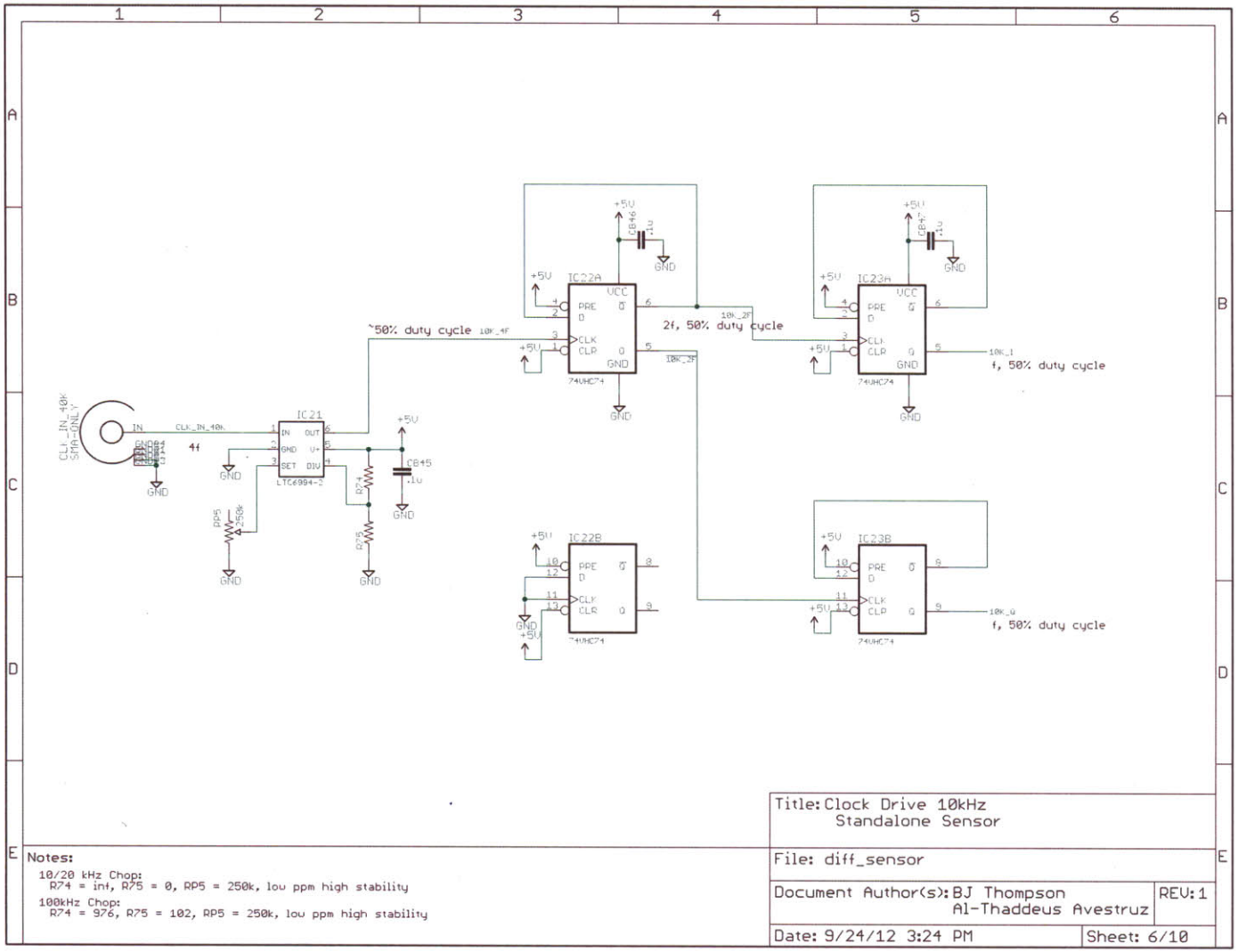


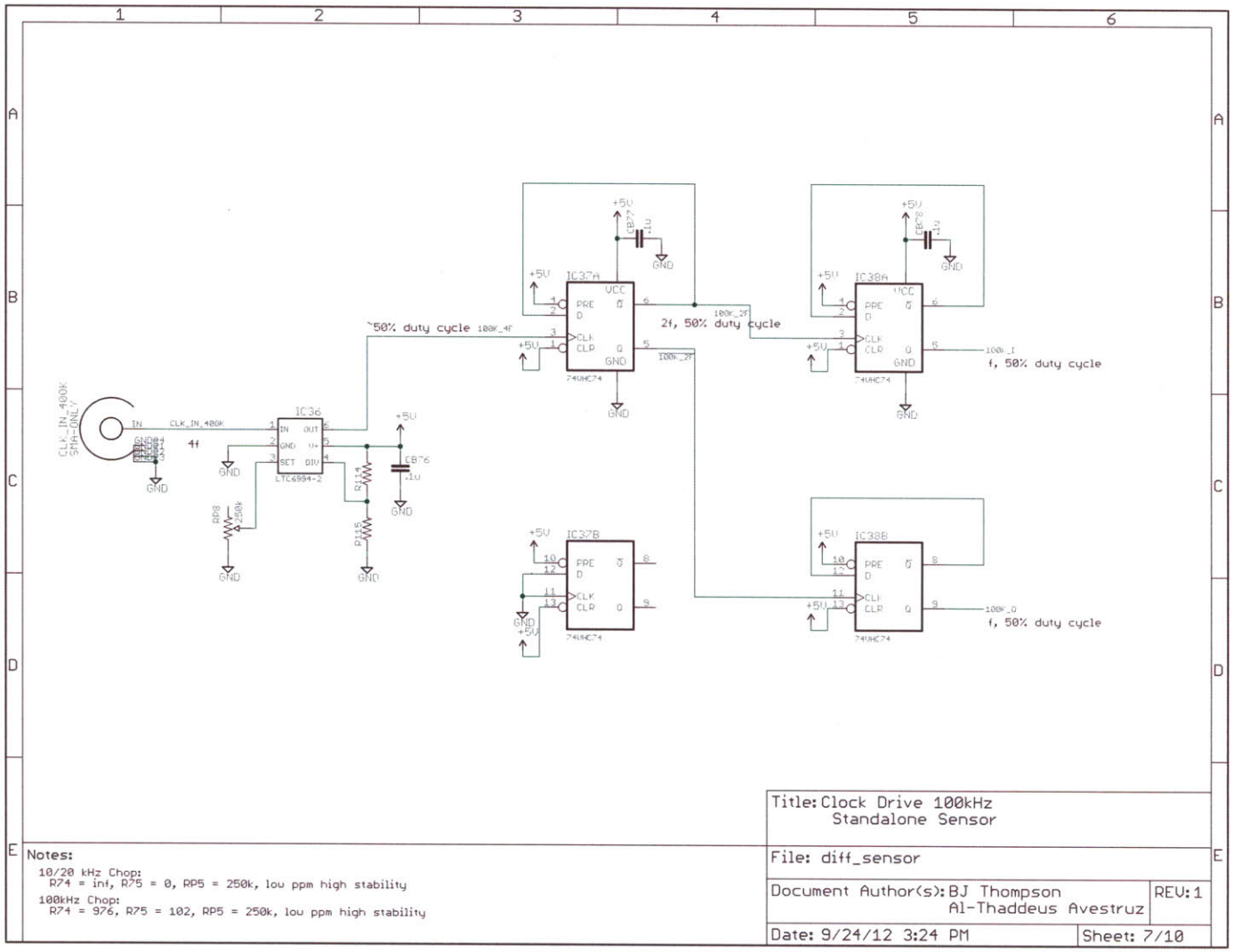

SENSE+
SENSE-
SENA+
SENA-
SENB+
SENB-
CHOPA+
CHOPA-
CHOPB+
CHOPB-
FILTA+
FILTA-
FILTB+
FILTB-
ADC_A
ADC_B
+2.5VREF
AGND
+5V
-5V
Title: Chopper and IAmp (1, 2)
Standalone Sensor
File: diff_sensor
Document Author(s): BJ Thompson
Al-Thaddeus Avestruz
REV: 1
Date: 9/24/12 3:24 PM
Sheet: 2/10
Notes:

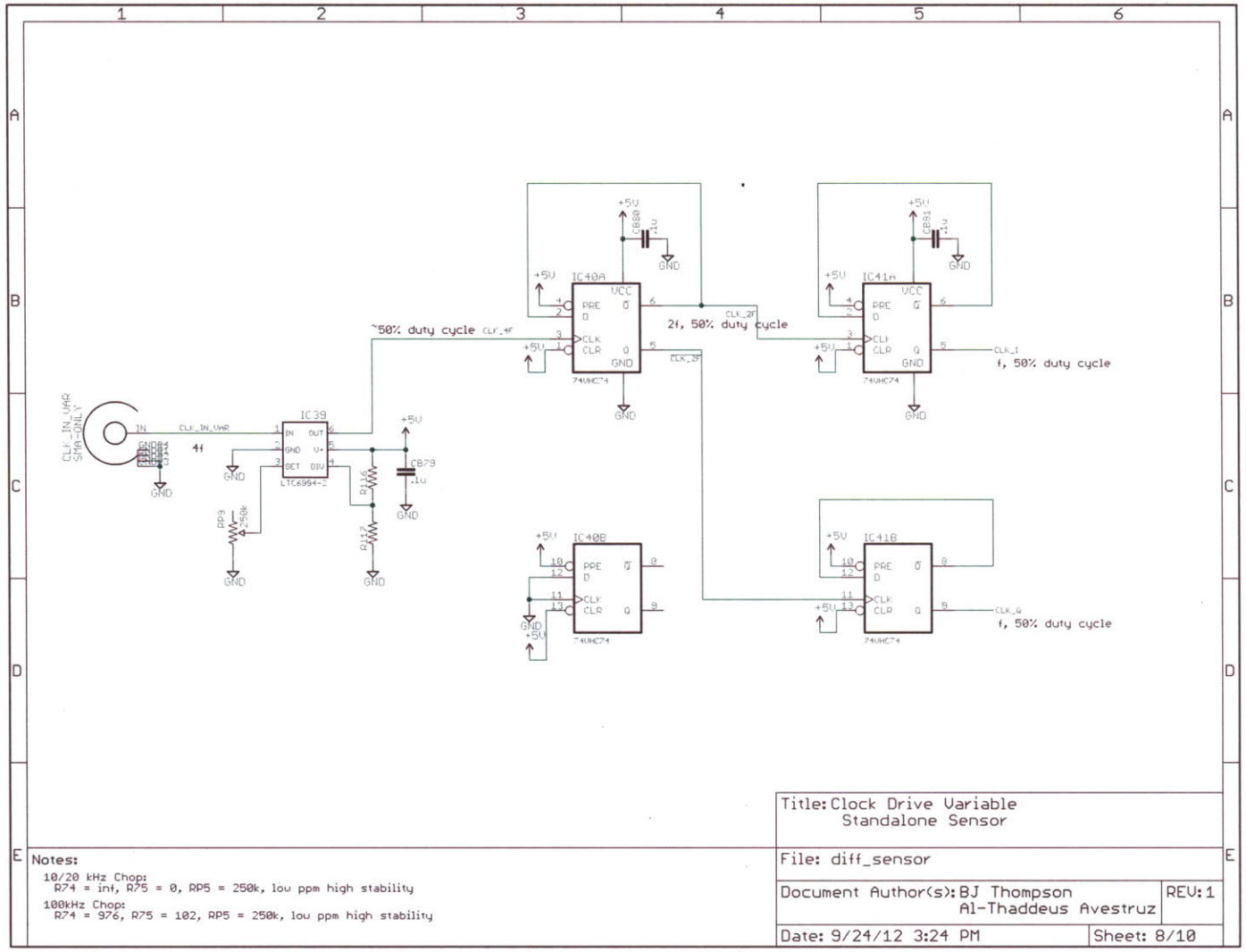

SENSE+
SENSE-
SENC+
SENC-
SEND+
SEND-
CHOPC+
CHOPC-
CHOPD+
CHOPD-
FILTC+
FILTC-
FILTD+
FILTD-
ADC_C
ADC_D
+2.5VREF
AGND
+5U
-5U
Title: Chopper and IAmp (3, 4)
Standalone Sensor
File: diff_sensor
Document Author(s): BJ Thompson
Al-Thaddeus Avestruz
REV: 1
Date: 9/24/12 3:24 PM
Sheet: 3/10
Notes:

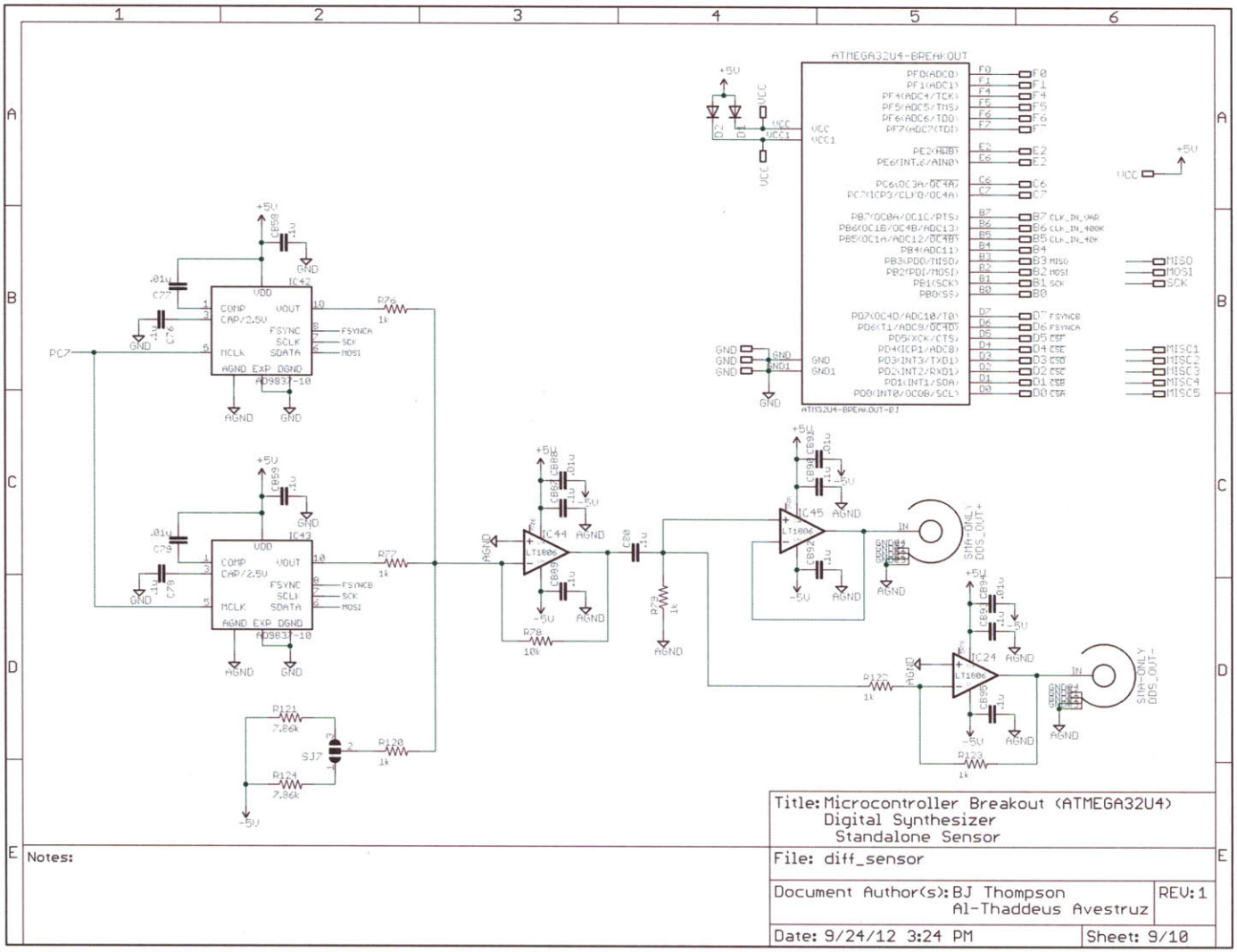

SRC_DIV
SENE+
SENE-
CLK_I
CHOPE+
CHOPE-
FILTE+
FILTE-
ADC_E
SENF+
SENF-
CLK_Q
CHOPF+
CHOPF-
FILTF+
FILTF-
ADC_F
HV_DIV
SMA-ONLY
Title: Chopper and IAmp
HV Measurements
Standalone Sensor
Notes:
File: diff_sensor
Document Author(s): BJ Thompson
Al-Thaddeus Avestruz
REV: 1
Date: 9/24/12 3:24 PM
Sheet: 4/10


ADC_A
ADC_B
ADC_C
ADC_D
ADC_E
ADC_F
+2.5VREF
+5V
AGND
GND
LTC2484
MOSI
MISO
SCK
CSA
CSB
CSC
CSD
CSE
CSF
FOAB
FOCD
FOEF
SJ1
SJ2
SJ3
SJ4
SJ5
SJ6
Title: ADC (4 of 4)
Standalone Sensor
Notes:
File: diff_sensor
Document Author(s): BJ Thompson
Al-Thaddeus Avestruz
REV: 1
Date: 9/24/12 3:24 PM
Sheet: 5/10

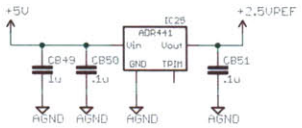

1
2
3
4
5
6
A
B
C
D
E
CLK_IN_40K
SMA-ONLY
IN
GND
CLK_IN_40k
4f
IC21
LTC6994-2
IN
GND
SET
OUT
V+
DIV
+5V
CB45
.1U
RP4
RP5
250k
GND
~50% duty cycle 10K_4F
IC22A
74VHC74
PRE
D
CLK
CLR
Q
VCC
CB46
CB47
10K_2F
2f, 50% duty cycle
10K_2F
IC23A
10K_1
f, 50% duty cycle
IC22B
IC23B
10K_Q
f, 50% duty cycle
Notes:
10/20 kHz Chop:
R74 = int, R75 = 0, RP5 = 250k, low ppm high stability
100kHz Chop:
R74 = 976, R75 = 102, RP5 = 250k, low ppm high stability
Title: Clock Drive 10kHz
Standalone Sensor
File: diff_sensor
Document Author(s): BJ Thompson
Al-Thaddeus Avestruz
REV: 1
Date: 9/24/12 3:24 PM
Sheet: 6/10

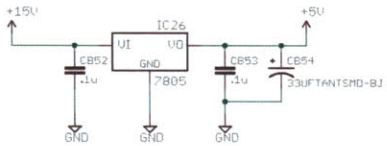

1
2
3
4
5
6
A
B
C
D
E
CLK_IN_400K
SMA-ONLY
IN
CLK_IN_400K
4f
GND
IC36
LTC6994-2
IN
GND
SET
OUT
DIV
+5V
RP4
250k
RP5
250k
CB76
.1U
~50% duty cycle 100K_4F
IC37A
IC38A
IC37B
IC38B
74VHC74
PRE
D
CLK
CLR
Q
VCC
100K_2F
2f, 50% duty cycle
100K_1
f, 50% duty cycle
100K_0
f, 50% duty cycle
Notes:
10/20 kHz Chop:
R74 = inf, R75 = 0, RP5 = 250k, low ppm high stability
100kHz Chop:
R74 = 976, R75 = 102, RP5 = 250k, low ppm high stability
Title: Clock Drive 100kHz
Standalone Sensor
File: diff_sensor
Document Author(s): BJ Thompson
Al-Thaddeus Avestruz
REV: 1
Date: 9/24/12 3:24 PM
Sheet: 7/10

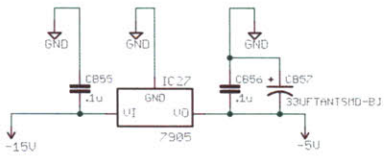

1
2
3
4
5
6
A
B
C
D
E
CLK_IN_VAR
SMA-ONLY
IN
GND
CLK_IN_VAR
4f
IC39
IN
GND
SET
OUT
V+
DIV
LTC6994-2
RP5
250k
R74
R75
CB79
.1u
+5V
~50% duty cycle CLK_4F
IC40A
IC40B
IC41A
IC41B
74VHC74
PRE
D
CLK
CLR
Q
VCC
CB80
CB81
CLK_2F
2f, 50% duty cycle
CLK_1
f, 50% duty cycle
CLK_0
f, 50% duty cycle
Notes:
10/20 kHz Chop:
R74 = int, R75 = 0, RP5 = 250k, low ppm high stability
100kHz Chop:
R74 = 976, R75 = 102, RP5 = 250k, low ppm high stability
Title: Clock Drive Variable
Standalone Sensor
File: diff_sensor
Document Author(s): BJ Thompson
Al-Thaddeus Avestruz
REV: 1
Date: 9/24/12 3:24 PM
Sheet: 8/10

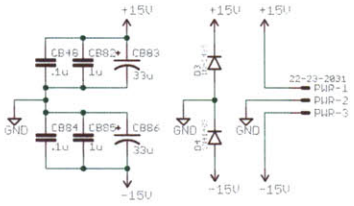

ATMEGA32U4-BREAKOUT
ATMEGA32U4-BREAKOUT-BJ
AD9833-10
LT1806
Title: Microcontroller Breakout (ATMEGA32U4)
Digital Synthesizer
Standalone Sensor
File: diff_sensor
Document Author(s): BJ Thompson
Al-Thaddeus Avestruz
REV: 1
Date: 9/24/12 3:24 PM
Sheet: 9/10
Notes:

Voltage Reference and Regulators

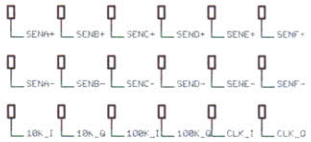


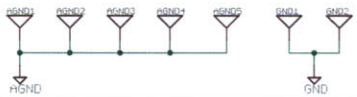






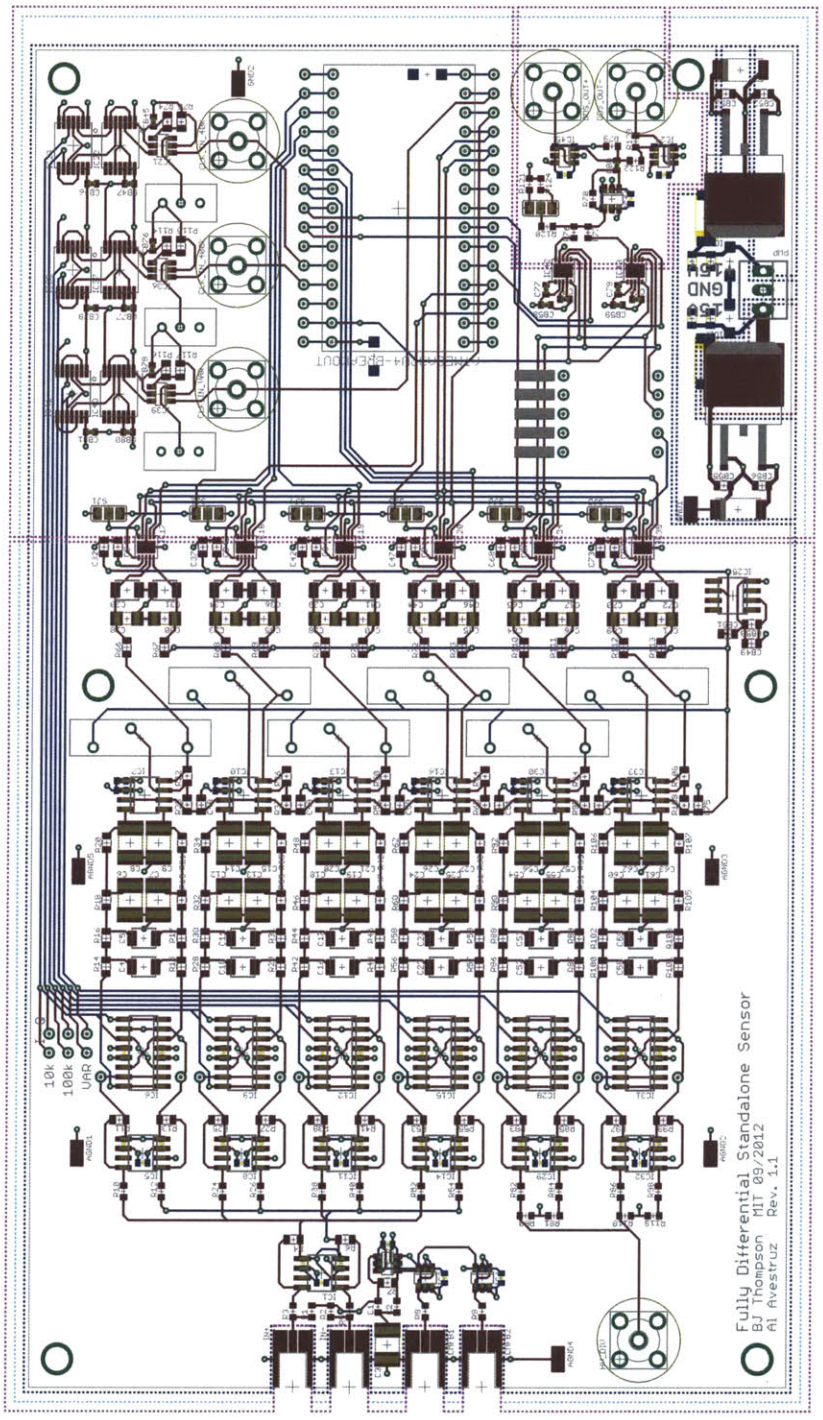
Test Points





Notes:	Title: Voltage Regulators and Reference Standalone Sensor
	File: diff_sensor
	Document Author(s): BJ Thompson, Al-Thaddeus Avestruz — REV: 1
	Date: 9/24/12 3:24 PM — Sheet: 10/10

B.3.2 PCB Layout



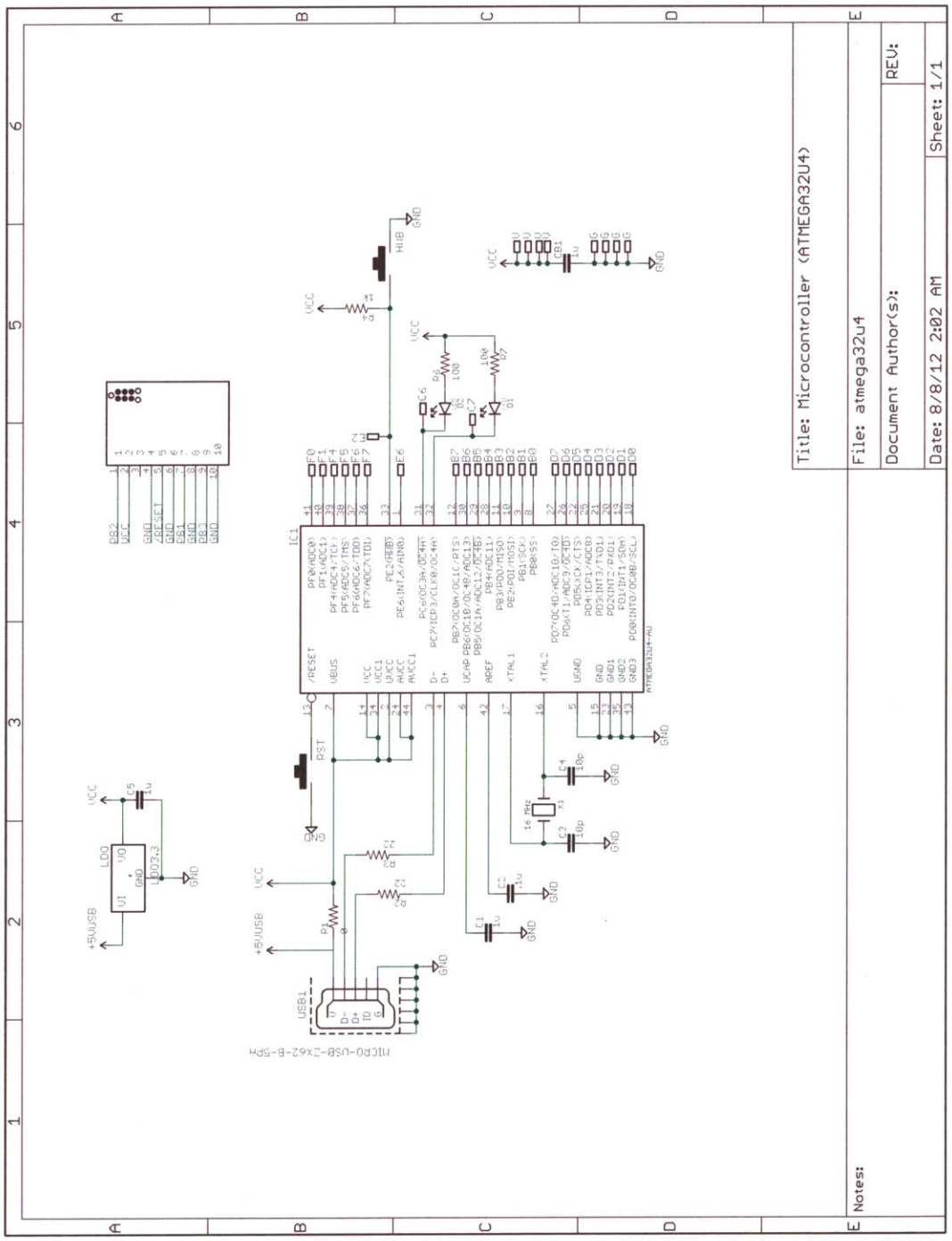
B.3.3 Bill of Materials

Qty	Value	Parts
6	10k Pot	RP1, RP2, RP3, RP4, RP6, RP7
18	Gain Adj.	R22, R23, R36, R37, R50, R51, R64, R65, R74, R75, R94, R95, R108, R109, R114, R115, R116, R117
1	5 MHz	X1
6	Jumpers	SJ1, SJ2, SJ3, SJ4, SJ5, SJ6
63	.1u	C76, C78, C80, CB1, CB2, CB4, CB5, CB7, CB8, CB10, CB11, CB13, CB14, CB16, CB17, CB18, CB19, CB20, CB21, CB23, CB24, CB25, CB26, CB27, CB28, CB30, CB31, CB32, CB33, CB34, CB35, CB37, CB38, CB39, CB40, CB45, CB46, CB47, CB58, CB59, CB60, CB61, CB63, CB64, CB65, CB66, CB67, CB68, CB70, CB71, CB72, CB73, CB76, CB77, CB78, CB79, CB80, CB81, CB87, CB89, CB90, CB92, CB93
8	.1u	CB48, CB50, CB51, CB52, CB53, CB55, CB56, CB84
24	.1u, PPS	C4, C5, C10, C11, C16, C17, C22, C23, C29, C31, C34, C36, C39, C41, C44, C46, C52, C53, C58, C59, C65, C67, C70, C72
14	.01u	C77, C79, CB3, CB6, CB9, CB12, CB15, CB22, CB29, CB36, CB62, CB69, CB88, CB91
24	.01u, 1%	C6, C7, C8, C9, C12, C13, C14, C15, C18, C19, C20, C21, C24, C25, C26, C27, C54, C55, C56, C57, C60, C61, C62, C63
12	.01u, 2%	C28, C30, C33, C35, C38, C40, C43, C45, C64, C66, C69, C71
1	1M	R7
2	1N4148S	D3, D4
4	1k	R76, R77, R78, R79
21	1u	C32, C37, C42, C47, C48, C49, C50, C51, C68, C73, C74, C75, CB41, CB42, CB43, CB44, CB49, CB74, CB75, CB82, CB85
2	10k, .01%	R1, R2
60	10k, 1%	R14, R15, R16, R17, R18, R19, R20, R21, R28, R29, R30, R31, R32, R33, R34, R35, R42, R43, R44, R45, R46, R47, R48, R49, R56, R57, R58, R59, R60, R61, R62, R63, R66, R67, R68, R69, R70, R71, R72, R73, R86, R87, R88, R89, R90, R91, R92, R93, R100, R101, R102, R103, R104, R105, R106, R107, R110, R111, R112, R113
1	10n, 1%	C3
1	22-23-2031	PWR

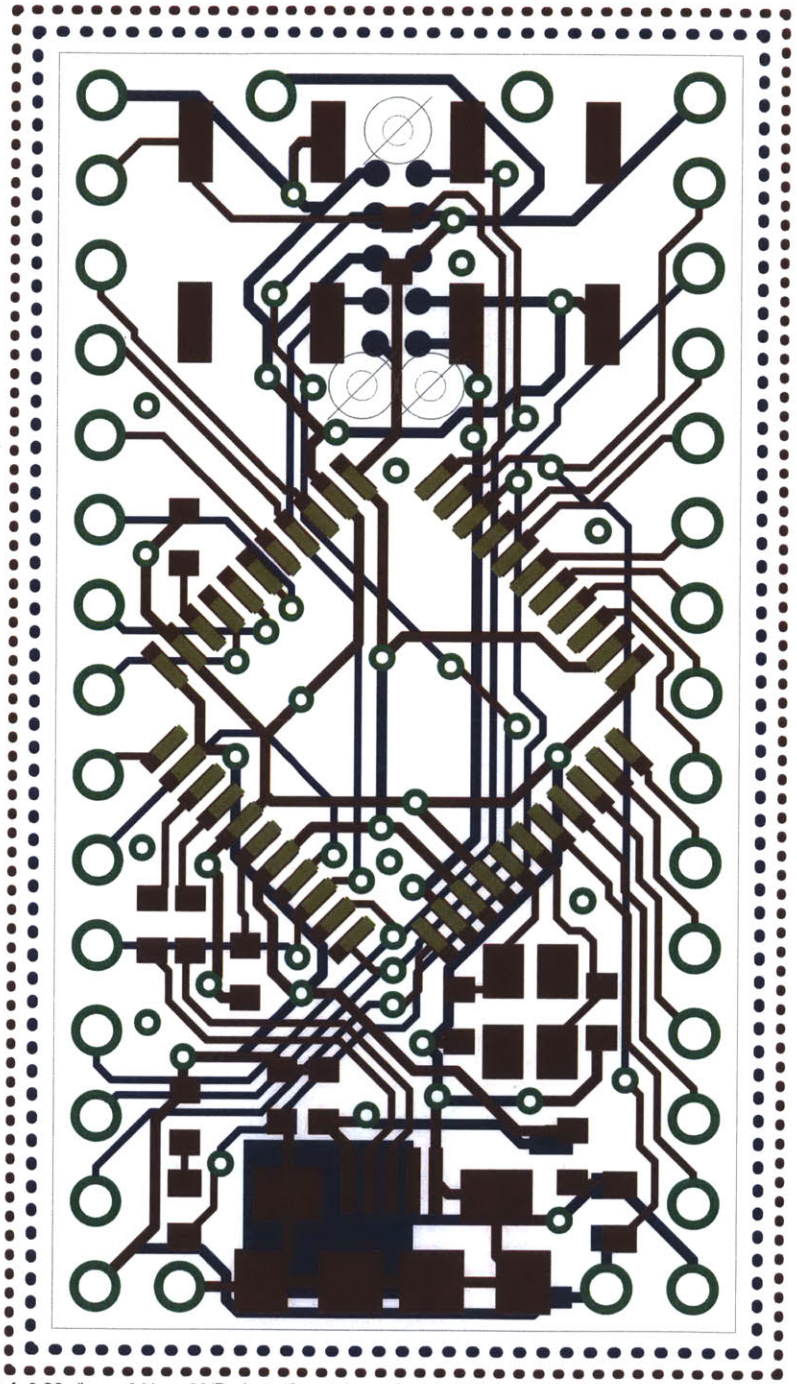
4	30, .1%	R83, R85, R97, R99
2	33u	CB54, CB57
2	33u	CB83, CB86
2	34, .1%	R81, R119
14	50, .1%	R3, R5, R10, R12, R24, R26, R38, R40, R52, R54, R82, R84, R96, R98
2	51, .05%	R8, R9
6	74VHC74	IC22, IC23, IC37, IC38, IC40, IC41
1	100p, 1%	C2
2	110, .1%	R80, R118
1	180p, 1%	C1
3	250k	RP5, RP8, RP9
10	500, .05%	R4, R6, R11, R13, R25, R27, R39, R41, R53, R55
1	7805	IC26
1	7905	IC27
7	AD8138	IC1, IC5, IC8, IC11, IC14, IC29, IC32
6	AD8230	IC7, IC10, IC13, IC16, IC30, IC33
2	AD9837-10	IC42, IC43
1	ADR441	IC25
1	ATM32U4	ATMEGA32U4-BREAKOUT
5	LT1806	IC2, IC3, IC4, IC44, IC45
6	LTC2484	IC17, IC18, IC19, IC20, IC34, IC35
3	LTC6994-2	IC21, IC36, IC39
6	MAX4523	IC6, IC9, IC12, IC15, IC28, IC31
4	MMXC	CON1, CON2, CON3, CON4
5	SMA	CLK_IN_40K, CLK_IN_400K, CLK_IN_VAR, DDS_OUT, HV_DIV
7	TPT	AGND1, AGND2, AGND3, AGND4, AGND5, GND1, GND2

B.4 Atmega32U4 Breakout Board

B.4.1 Full Schematic



B.4.2 PCB Layout



9/3/12 10:16 PM f=6.29 /home/bjthom28/Projects/Standalone_Sensor/Sensor and Code/pcb_and_schematic/atmega32u4/atm

B.4.3 Bill of Materials

Qty	Value	Parts
1	.1u	C2
1	0	R1
1	1k	R4
3	1u	C1, C5, CB1
2	10p	C3, C4
1	16 MHz	X1
2	22	R2, R3
2	100	R6, R7
1	ATMEGA32U4-AU	IC1
1	LDO3.3	LDO
2	LED	D1, D2
1	MICRO-USB-ZX62-B-5PA	USB1
2	SW-C&K-KSC-JWING	HWB, RST
1	TC2050-IDC-NL	ISP

Bibliography

[1] J.J. Cooley. Capacitive Sensing with a Fluorescent Lamp. M.eng. thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, February 2007.

[2] Analog Devices. AD9837 Low Power, 8.5 mW, 2.3 V to 5.5 V, Programmable Waveform Generator, 2011. Rev. 0.

[3] Analog Devices. AD8620 Precision, Very Low Noise, Low Input Bias Current, Wide Bandwidth JFET Operational Amplifiers, 2008. Rev. F.

[4] Texas Instruments. OPA452,OPA452: 80V, 50mA Operational Amplifiers, 2003. Rev. C.

[5] J.J. Cooley. *Analysis, Modeling and Design of Energy Management and Multisource Power Systems.* Ph.d. thesis, Massachusetts Institude of Technology, Cambridge, Massachusetts, June 2011.

[6] D. Vickery. A Retrofit Current Sensor for Non-Intrusive Power Monitoring at the Circuit Breaker. M.eng. thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, September 2011.

[7] M. Carminati, M. Sampietro, G. Ferrari, and F. Gozzini. Instrumentation with attofarad resolution for electrochemical impedance measurements on molecular biosensors. In *Research in Microelectronics and Electronics, 2009. PRIME 2009. Ph.D.*, pages 316–319, July 2009.

[8] G. Ferrari, F. Gozzini, A. Molari, and M. Sampietro. Transimpedance amplifier for high sensitivity current measurements on nanodevices. *Solid-State Circuits, IEEE Journal of*, 44(5):1609–1616, May 2009.

[9] L Fumagalli, G Ferrari, M Sampietro, I Casuso, E Martnez, J Samitier, and G Gomila. Nanoscale capacitance imaging with attofarad resolution using ac current sensing atomic force microscopy. *Nanotechnology*, 17(18):4581, 2006.

[10] A Heidary, R Taherkhani, and G.C.M. Meijer. A capacitance measurement system with milliwatt power and attofarad resolution. In *20th Iranian Conference on Electrical Engineering*, May 2012.

[11] S Wu and J Yu. Attofarad capacitance measurement with scanning microwave microscopy, April 2012. Agilent Technologies.

[12] Daniele Andreuccetti, Roberto Fossi, and Caterina Petrucci. Dielectric Properites of Body Tissues, 2010. http://niremf.ifac.cnr.it/tissprop/.

[13] Camelia Gabriel and Sami Gabriel. Compilation of the Dielectric Properties of Body Tissues at RF and Microwave Frequencies, 1997. mirrored at: http://niremf.ifac.cnr.it/docs/DIELECTRIC/home.html.

[14] Ferroxcube. 3C81 Material specification, 2008. Datasheet.

[15] Ferroxcube. P42/29 P cores and accessories, 2008. Datasheet.

[16] Linear Technology. LTC2484 24-Bit $\Delta\Sigma$ ADC with Easy Drive Input Current Cancellation, 2005. Rev. D.

[17] Analog Devices. AD8230 16V Rail-to-Rail, Zero-Drift, Precision Instrumentation Amplifier, 2007. Rev. B.

[18] Analog Devices. ADA4817-1/ADA4817-2 Low Noise, 1 GHz FastFET Op Amps, 2008. Rev. 0.

[19] Analog Devices. AD8131 Low Distortion Differential ADC Driver, 2006. Rev. F.

[20] Maxim. MAX4521/MAX4522/MAX4523 Quad, Low-Voltage. SPST Analog Switches, July 2007. Rev. 6.

[21] Linear Technology. LT1806/LT1807 325MHz, Single/Dual, Rail-to-Rail Input and Output, Low Distortion, Low Noise Precision Op Amps, 2010. Rev. C.

[22] KSM. RG178 Coaxial Cable to Mil-C-17 Specification, 2011.

[23] John Ardizzoni and Jonathon Pearson. AN-0990 Terminating a Differential Amplifier in Single-Ended Input Applications. Analog Devices, 2009. Rev. 0.