Data Logging System for a Synthetic Aperture Radar Unit

Nicholas J. Testin  
Kennesaw State University

Philip Davis  
Kennesaw State University

Ian Dorell  
Kennesaw State University

Alexander Gillespie  
Kennesaw State University

Follow this and additional works at: http://digitalcommons.kennesaw.edu/honors_etd

Part of the Electromagnetics and Photonics Commons, Other Electrical and Computer Engineering Commons, Signal Processing Commons, Systems and Communications Commons, and the VLSI and Circuits, Embedded and Hardware Systems Commons

Recommended Citation

Testin, Nicholas J.; Davis, Philip; Dorell, Ian; and Gillespie, Alexander, "Data Logging System for a Synthetic Aperture Radar Unit" (2016). Honors College Capstones and Theses. 7.  
http://digitalcommons.kennesaw.edu/honors_etd/7

This Capstone is brought to you for free and open access by the Honors College at DigitalCommons@Kennesaw State University. It has been accepted for inclusion in Honors College Capstones and Theses by an authorized administrator of DigitalCommons@Kennesaw State University. For more information, please contact digitalcommons@kennesaw.edu.
Data Logging System for a Synthetic Aperture Radar Unit

Submitted in partial fulfillment of the course requirements for EE 4800

Philip Davis
Ian Dorell
Alexander Gillespie
Nicholas Testin
1. Summary

This report details the design, implementation, and construction of a data logging system for a currently existing radar imaging unit constructed by a professor in the Electrical Engineering department at Kennesaw State University [1]. At the project’s inception, the unit worked completely with analog signals. For analysis on a computer, digital signals are required, and the interface currently being used to digitally convert this analog data is not adequate to take advantage of the full potential of this radar design. The work detailed here integrates microcontrollers and encoders which, combined with digital sampling, more accurately and consistently collects data from the radar unit. Furthermore, the end device allows for its collected data to be transferred to an external server or computer for processing. The end goal of this project was for the data collected by the radar unit to be processed into a two dimensional linear map of a static landmark.

This group consists of members Philip Davis, Ian Dorell, Alex Gillespie, and Nicholas Testin, who worked under the direction of faculty advisor Dr. Theodore Grosch. In the broadest sense, the division of labor was as follows: Philip took charge of programming the microcontroller, while Ian headed the designing of the wheel encoder and the circuit layout. Alex took lead on the .wav file formatting and the saving of data to the on-board SD card, and Nicholas acted as project manager and took lead on the design of the radar’s mobile platform.

The domains covered by this project are primarily embedded systems and digital signal processing, with some applied radio frequency theory integrated into the operation of the radar system itself. The budget for this project is $200, drawn from the $50 allocated to each group member.
2. Introduction

Radar systems are generally used in determining properties, most commonly distance from a reference point, of solid objects using single antennas or large antenna arrays. These antennas transmit and receive electromagnetic signals, which can be processed to obtain various relevant data. By using only one antenna and moving it along a linear axis to record an area of static objects, one can mimic a larger array of antennas to collect high-resolution data: this setup is known as a synthetic aperture radar system. The data collected from this type of radar configuration, after processing, is well-known for its detail and map-like quality, and can be used to render a two-dimensional or three-dimensional representation of the scanned area [2, pp. 19].

Radar systems almost exclusively have analog signal output, but digital data is required to perform analysis via a computer. By using a high-clock microcontroller with a comparably fast analog-to-digital converter (ADC), one can digitally store the analog signal as a digital file, which can be processed at any later time using an external computer. Combining this microcontroller with digital sensors that measure position and path deviation allows for accounting of the error present in a mechanical system and adjusting the collected data during processing.

The problem, for which the purpose of this project is to solve, is that an existing radar system used for imaging static objects needs improved performance while implementing few (if any) changes to the radar system itself. The objective of this project is to build a data logging system that can increase the performance of the radar unit by automatically collecting the radar data as well as other data, such as deviation, at precise intervals and storing it to an onboard memory storage unit for later processing.
3. Needs/Problems

The initial state of the project consisted of a working radar system, built by a EE professor, that collects incoming data to be processed by an external computer. However, the initial system did not perform to its full potential as noted below:

- The initial system did not have a structure unit dedicated to supporting the system while it moves and collects data. An improvised setup was used in the past, which reduces the precision of the data collected.

- Data is collected only when triggered manually by the system operator; as a result, the spacing between data sets is inconsistent and will result in a distorted image.

- The initial system only collects data based on a single height setting: as a result, the processed data will result in a “flat” image. A three-dimensional image could be constructed by taking consistent data from two or more different heights.

- There is not a subsystem in place to track the movement of the radar system. For this reason, the data collected by the system is inconsistent in heading and height, and appears “fuzzy” when processed into a visible image.

- The collected data was initially transferred to a computer for processing via an external sound card. The range of the radar system was limited by the sampling rate of the sound card, which is being used in a manner for which it was not originally designed.

- All movement of the radar system is induced and controlled manually by a human operator. While it is possible to collect precise data with manual controls, ensuring that data remains accurate during all collection times requires a great deal of effort and preparation.

4. Goals/Objectives

In order to maximize the potential of the initial radar system, some additions or modifications must be made. These modifications must be minimally invasive with respect to the radar system. They include:

- A dedicated platform or cart will be constructed to support the radar system during data collection. The platform will also have adjustable height options, to allow for the option of collecting data to aid in the construction of a three-dimensional model.

- A wheel encoder, which will track the distance that the platform has traveled, will be designed and installed on the platform. With the data from the encoder, the radar system can accurately track its movement and precisely capture data to minimize distortion.
• A microprocessor based on the radar system will store the collected data or stream the collected data over USB to a computer for processing.
5. Research Summary

Radar is defined by Skolnik as “an electromagnetic system for the detection and location of reflecting objects such as aircraft, ships, spacecraft, people, and the natural environment” [2, pp. 1]. The term originates as a contraction of the phrase “radio detecting and ranging”, which, while still representing the fundamental use of radio frequency (RF) waves to detect objects at a distance, no longer represents the full capabilities of modern-day radar systems [3, pp. 464]. A radar system functions by emitting electromagnetic frequency (EMF) waves from a transmitting antenna into a target area; the emitted waves will scatter upon making contact with an object, and the scattered waves reflected directly back can be collected by a receiving antenna. By comparing the changed properties, including but not limited to frequency, wavelength, phase shift, and amplitude, of the returning waves with the emitted waves, one can infer information about what objects might be present in the area. The value of a radar system comes from its ability to operate in varied weather conditions with little to no effects on its accuracy [2, pp. 2].

There are some fundamental equations to know before attempting to understand the workings of radar. The first, the relationship between the frequency and wavelength of electromagnetic waves, is given below:

\[ c = \lambda f \]

where \( c \) is the speed of light, \( \lambda \) is the wavelength of the EMF wave and \( f \) is the frequency of the EMF wave. (For ease of calculations, the value of \( c \) used here will be approximated to \( 3 \times 10^8 \) m/s.) In short, the wavelength and frequency are inversely related: as one increases, the other decreases, and vice versa [3, pp. 25]. The next is the basic range to target equation:

\[ R = \frac{cT_R}{2} \]

where \( R \) is the range to a target in the radar’s area of detection, \( c \) is the speed of light, and \( T_R \) is the time for the signal to travel to the target and back [2, pp. 2]. A variant of the above equation accounts for the waves from a secondary bounce being reflected back, which can cause an echoing effect; that equation, known as the maximum unambiguous range, is given as:

\[ R_{un} = \frac{cT_p}{2} = \frac{c}{2f_p} \]

where \( R_{un} \) is the value beyond which any returns would be from echoes, \( T_p \) is the repeating period of the radar output pulse, and \( f_p \) is the frequency of the radar output pulse [2, pp. 3]. The final, simplified fundamental radar range equation is given as:

\[ R^2_{max} = \frac{P_t A_e \sigma}{(4\pi)^2 S_{min}} = \frac{P_t G^2 \lambda^2 \sigma}{(4\pi)^3 S_{min}} = \frac{P_t A_e^2 \sigma}{4\pi \lambda^2 S_{min}} \]
where \( R_{\text{max}} \) is the maximum range of the radar, \( P_t \) is the peak power, \( S_{\text{min}} \) is the minimum detectable signal, \( G \) is the gain of the transmitting antenna, \( \lambda \) is the wavelength of the transmitted wave, and \( \sigma \) is the radar cross-section [2, pp.6-7]. Finally, it is important to keep in mind the ratio of signal power to noise power, also known as SNR, which can negatively affect the radar’s performance if too low. The equation is shown below:

\[
S_n = \frac{P_r}{P_n} = \frac{P_r \tau}{kT_{\text{sys}}}
\]

where \( P_r \) is the power of the received signal, \( P_n \) is the mean power of the input noise, \( \tau \) is the length between input pulse signals, \( k \) is Boltzmann’s constant, and \( T_{\text{sys}} \) is the system noise temperature [3, pp. 463, 467-468].

A radar system at its most fundamental level consists of a synchronization/modulation unit, an RF transmission unit, a transmit antenna, an RF receiver unit, a receive antenna, and a processor/display unit. (In the common case where the transmit and receive signals are broadcast over the same antenna, a duplexer is included to separate out the two signals and time the transmission and receiving functions; furthermore, a servo or motor to move the antenna is also often included.) The synchronizer/modulator generates a short timing pulse which is supplied to both the transmitter and the processor/display: the transmission unit, in turn, generates matching RF pulses and supplies the signal to the transmit antenna, which radiates the signal. The signal’s reflection off of the target is picked up by the receive antenna and supplied to the receiver unit, converting it to a lower frequency for analysis. The signal is finally supplied to the processor/display unit, which uses the signal to infer information about the target. The processor also places a human-readable version of the information on the display for the radar operator. By emitting multiple pulses continuously or in rapid succession, it is possible to determine information about a target, including velocity and location [3, pp. 464-465].

In order to function, a radar system requires the presence of an antenna, which is described by Fenn as a system of any number of “transducers that can convert the signal voltage on a transmission line to a transmitted electromagnetic (EM) wave”, or vice versa [4]. An antenna can be described as isotropic, or radiating equally in all directions, as well as omnidirectional, or radiating primarily along a single plane, and directional, or radiating primarily in a single direction [3; pp. 404]. The isotropic antenna is primarily a theoretical construct, and is used to describe the increase in signal output due to the antenna pattern, or gain, given by the formula

\[
G = \frac{4\pi A_e}{\lambda^2}
\]

where \( A_e \) is the antenna effective area and \( \lambda \) is the wavelength. Antennas are designed for their primary beam or main lobe to radiate in a particular direction for maximum gain, but virtually all have secondary beams, known as sidelobes, which radiate in divergent directions than the main beam and can cause unintended side effects such as interference when not accounted for [3, pp.
Finally, antennas can be moved to radiate in a different direction by mechanical means (i.e. by literally moving the physical antenna) or by electrical means (i.e. by varying the phase input of the transmit signal [3, pp. 436-437].

It is helpful to know several mathematical relationships. The power density of the antenna, which is the radiated power per unit area of coverage, decreases as the distance away from the antenna increases. In addition, antenna gain is typically measured and given as a value increase or decrease with respect to the isotropic antenna [4]. Finally, gain values are typically seen as logarithmic (as opposed to linear) values, with the rule being a gain of 10 dB representing a factor of 10 increase, and a gain of 3 dB approximately representing a factor of 2 increase. As an example, an antenna with a gain of 6 dBi on its main lobe would have a power output increase of approximately a factor of 4 over the isotropic antenna.

The design of the antennas for this radar system are primarily structured around the use of metal coffee cans as waveguides for each respective radiating element. The can dimensions specified in Fenn's original design are given below [4]:

<table>
<thead>
<tr>
<th>Metal can length:</th>
<th>5.25 inch</th>
</tr>
</thead>
<tbody>
<tr>
<td>Radiating monopole</td>
<td>1.2 inch</td>
</tr>
<tr>
<td>length:</td>
<td>(\approx \frac{\lambda}{4}) in free space</td>
</tr>
<tr>
<td>Metal can diameter:</td>
<td>3.9 inch</td>
</tr>
<tr>
<td>Spacing from monopole to back wall:</td>
<td>1.8 inch</td>
</tr>
</tbody>
</table>

In addition, the originally published design requirements of the radar system are given below [5]:

<table>
<thead>
<tr>
<th>Operating frequency band:</th>
<th>1.950 to 3.00 GHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bandwidth:</td>
<td>1 GHz</td>
</tr>
<tr>
<td>Waveform:</td>
<td>Continuous-wave ramp</td>
</tr>
<tr>
<td>Antenna isolation:</td>
<td>50 dB</td>
</tr>
<tr>
<td>DC Power:</td>
<td>&lt; 1W</td>
</tr>
<tr>
<td>RF Power:</td>
<td>&lt; 1W effective isotropic radiated power</td>
</tr>
</tbody>
</table>

Device is built from off-the-shelf parts.

Device uses connectorized components.

Minimal soldering is used in building the device.

One should note that the waveform being broadcast in the above design is a continuous-wave ramp function modulated with respect to frequency: this “FM-CW” radar configuration was chosen for this project due to its compatibility with off-the-shelf solid-state components, as well as its simplicity and versatility [6]. The broadcast waveform emitted out of the transmit antenna will be reflected off a target in the radar’s field of view and collected by the receive antenna; the waveform shape itself will not have changed, but it will have gained a frequency delay proportional to the target’s range. By comparing this received signal frequency to the initial transmitted signal frequency, the range of the target can be determined [7, pp. 185].

Equally important as the design of the radar is the algorithm used to process the collected data. Lipa and Barrick describe the optimal processing solution for an FM-CW radar to be a fast Fourier transform over the data collected in a radar sweep, with the signal power confined to a
single frequency pulse to maximize SNR [8, pp. 4-9]. This general method can be augmented by including techniques for accounting for sources of error, such as phase error from the FM signal modulation [9], as well as techniques for improving the radar’s ability to detect targets and, therefore, the amount of useful data it is able to collect [10].

The WAVE file type is a subset of the RIFF file types. The first 12 bytes in a WAVE are dedicated to specifying the file is RIFF and then the file is WAVE. The next sub-chunk is populated with general information including the number of channels, sample rate, byte rate, and bits per sample. The last subchunk is then populated with the samples expressed without any encoding. This file type does not use any footer. The last byte in a WAVE file is the last sample. The majority of the bytes used in a wave file are expressed in little endian, or where he most significant bit is written last and the zero fills are expressed first [11].

According to the PSoC documentation for the CY8C4247AZI-M485, the ADC analog pins operate in the range of +-1.024V or offset +-1V. It will also allow for a range of 0-3.3V with special settings [12]. The Radar outputs at +-5V but can be scaled down fairly easily. The different example code Cypress provides for the ADC uses interrupts to take in the information [13]. Using this method would mean setting interrupt handler bits high or low or else the chip will be constantly interrupting and the main loop will code will be staggered. So instead the better option would to make an infinite loop and check the ADC_EOS_MASK bit to determine whether the ADC conversion is done then break that loop. This will hopefully give a smoother tick lag than the interrupt method. This tick lag is also maintained due to the counter running in parallel [14].
6. Project Solutions

The primary option evaluation for this project concerns the chip used for processing the incoming data from the radar system. The ideal microprocessor platform for the data logger has the following properties:

- The platform must possess a robust ADC that can convert the incoming analog signals from the radar system into discrete and quantized digital signals.
- It possesses the ability to read and write to an SD card on its board.
- It must have a built-in camera system.
- It must have low power requirements.
- The platform must have an on-board debugging system or easy access to computer-based debugging.
- The platform must either have already been used in some capacity by group members, or be well-documented and straightforwardly easy to configure.
- It must be able to connect wirelessly to an external computer or server platform to transfer the collected data.
- The platform must be inexpensive.

Certain of the above properties are desired more than required: for instance, a robust ADC is required for the success of the project, but an SD card reader could be built or acquired. Considering there are 8 requirements, each alternative will be scored out of 8 points. The alternatives evaluated consist of the following platforms:

1. Arduino UNO with ATmega328P
2. Cypress CY8CKIT-043 with ARM Cortex M0
3. Field-programmable gate array (FPGA)

The first alternative, the Arduino UNO, was used by group members in a previous course and is, therefore, a familiar platform; for the same reason, it is also available for use without additional cost. It also possesses low power requirements and provides access to computer-based debugging through Atmel’s software. It possesses neither a built-in camera system, nor SD card read-write capabilities, nor the ability to connect wirelessly to an external computer. Most significantly, while the UNO possesses an ADC, the ADC is not especially robust and would not work especially well with the data collection that is required of this project. The Arduino UNO’s raw score is 4/8; however, because it does not possess an adequate ADC, it would not be an acceptable alternative for the data logger’s microprocessor.

The second alternative, the Cypress CY8CKIT-043, was recommended as an alternative by the project’s faculty advisor. It is an inexpensive prototyping board manufactured with an emphasis on debugging and low power requirements. Most notably, the platform possesses a robust on-board ADC. It does not possess an on-board SD card reader, a camera system, or wireless
connecting capabilities. No group members have prior experience with the platform. With a raw score of 4/8, the CY8CKIT-043 would be an acceptable option for use as the microprocessor.

The third alternative, the FPGA, is a powerful platform that essentially functions as a reconfigurable logic gate and circuit chip; for this reason, it could be programmed to function with a robust ADC. However, the chip and its functions are not well-known to the group members, and programming such a chip is not especially straightforward or easy to learn. The platform does not possess a built-in SD card reader, a camera, or wireless connection capabilities. Its power requirements are not-well defined, but its cost is significantly more expensive than other options available. With a raw score of (at best) 2/8, the FPGA would not be an acceptable alternative for the data logger’s microprocessor.

Based on the above analyses, the chosen chip is the Cypress CY8CKIT-043. The deciding factors were its robust ADC and debugging capabilities as well as its particularly low cost. Despite not having attributes such as a built-in SD card reader and wireless connection capabilities, it is possible to obtain the functionality elsewhere and integrate it into this chosen chip, allowing for a more modular amount of control over the capabilities of the final product.

<table>
<thead>
<tr>
<th>Solution</th>
<th>Score</th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arduino UNO</td>
<td>4/8</td>
<td>No cost, prior experience</td>
<td>Poor ADC, few features</td>
</tr>
<tr>
<td>Cypress CY8CKIT-043</td>
<td>4/8</td>
<td>Robust ADC, inexpensive</td>
<td>No experience, few features</td>
</tr>
<tr>
<td>FPGA</td>
<td>2/8</td>
<td>Powerful capabilities, robust ADC</td>
<td>No experience, difficult to use, expensive</td>
</tr>
</tbody>
</table>

7. Procedures/Scope of Work

The design of the platform focuses on cementing the stability of the data collected from the radar system, with respect to height and mechanical noise introduced from the traveled terrain. This involves working with some basic structures and mechanical devices such as wheels. The noise could also be recorded and saved for use in processing using electrical periphery devices such as accelerometers and gyroscopic sensors. All data will be recorded in a straight line; as a result, the cart has no need for steering or a system to account for taking data along a curved path.

The wheel encoder must track the distance that the cart has traveled and, upon moving a set interval, send a signal to the radar system to mark a set of the incoming data for processing. The output signal from this unit will be recorded for transmission with the collected RF data. This will involve programming a microcontroller to act as a DSP, store data to an onboard memory device, with the option of formatting it for use in MATLAB.

A unit was considered for tracking the deviation of the platform from a predetermined straight line as a function of \( m \) where \( m \) is the encoder data. This information would have also been recorded for transmission with the collected data and used to adjust the processing of the final
radar image for greater accuracy. *(Note: it was discovered during an analysis of the system’s design that the processing power needed for such a requirement would be prohibitive based on the budget of the project; for this reason, it was excluded from the final project scope of work, and is included here for continuity.)*

An on-board microprocessor is used to convert all measured analog data into digital data for analysis on an external computer or server. This involves programming a system on a chip (SoC) to function as an ADC. The digital data is written as a WAV file to a connected SD card or flash memory unit. Discrete channels in the file represent the radar system's data as well as ancillary data to aid in processing, including but not limited to the generated RF signal, the recorded RF signal, and the position of the measurement device.

### 8. Design Requirements

<table>
<thead>
<tr>
<th>Req. #</th>
<th>Requirement</th>
<th>Subcategory</th>
<th>Maturity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1A</td>
<td>A cart designed and built to support the radar system shall be constructed.</td>
<td>Feature Set</td>
<td>Alpha-1</td>
</tr>
<tr>
<td>1B</td>
<td>The cart shall, at minimum, support the radar system plus an additional 25 pounds.</td>
<td>Functional Req.</td>
<td>Alpha-1</td>
</tr>
<tr>
<td>2A</td>
<td>A system-on-a-chip (SoC) dedicated to the data logging shall be included with the existing radar circuitry.</td>
<td>Feature Set</td>
<td>Alpha-1</td>
</tr>
<tr>
<td>2B</td>
<td>The SoC shall be interfaced with the radar system for the purpose of reading its incoming data.</td>
<td>Software</td>
<td>Alpha-2</td>
</tr>
<tr>
<td>2C</td>
<td>The data from the radar system shall be constantly read by the analog-to-digital converter (ADC) on the SoC.</td>
<td>Software</td>
<td>Alpha-2</td>
</tr>
<tr>
<td>2D</td>
<td>The system shall store data from the ADC to an array when a counter interrupt is triggered.</td>
<td>Software</td>
<td>Alpha-2</td>
</tr>
<tr>
<td>2E</td>
<td>The SoC shall read data from the radar system at a minimum rate of 100 KHz.</td>
<td>Performance</td>
<td>Alpha-1</td>
</tr>
<tr>
<td>3A</td>
<td>A rotary encoder shall provide the distance the cart has traveled to the SoC.</td>
<td>Feature Set</td>
<td>Alpha-1</td>
</tr>
<tr>
<td>3B</td>
<td>The rotary encoder shall be interfaced with at least one wheel/axle assembly on the cart.</td>
<td>Functional Req.</td>
<td>Alpha-1</td>
</tr>
<tr>
<td>3C</td>
<td>The interface between the encoder and wheel/axle assembly shall consist of a 3D printed pulley.</td>
<td>Hardware</td>
<td>Alpha-1</td>
</tr>
<tr>
<td>3D</td>
<td>The encoder shall increment a counter with each pulse, directly indicating the distance the cart has traveled.</td>
<td>Operational Req.</td>
<td>Alpha-1</td>
</tr>
<tr>
<td>3E</td>
<td>The distance output of the encoder should possess error of no more than 0.5 cm per 10 cm of linear travel.</td>
<td>Operational Req.</td>
<td>Alpha-2</td>
</tr>
<tr>
<td>4A</td>
<td>The ADC shall collect data from the radar system upon receiving a signal from the encoder corresponding to a preselected distance traveled by the cart.</td>
<td>Feature Set</td>
<td>Alpha-2</td>
</tr>
<tr>
<td>4B</td>
<td>The ADC shall store the data currently collected from the radar system to a system memory device.</td>
<td>Output</td>
<td>Alpha-1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>4C</strong></td>
<td>The ADC shall store the data in Req. 4A when it receives a signal from a manually-triggered button.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>4D</strong></td>
<td>The ADC shall store the data in Req. 4A when it receives a signal from the rotary encoder.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>4E</strong></td>
<td>The data stored to the board or system memory shall be in the form of a 2D array.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>4F</strong></td>
<td>The minimum operational frequency for the data recorded shall be 50 KHz.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>4G</strong></td>
<td>The communications between the SoC and system memory shall permit, at minimum, simple read and write operations.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>5A</strong></td>
<td>An organic light-emitting diode (OLED) screen will display information relating to the status of the logging of data from the radar system.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>5B</strong></td>
<td>The OLED screen shall display whether the SoC is collecting data from the radar system at the moment.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>5C</strong></td>
<td>The OLED screen may display more detailed information gathered from the other components of the data logging system.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>6A</strong></td>
<td>The system memory, containing the collected data, shall be interfaced with an external computer so that the data can be transferred for processing.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>6B</strong></td>
<td>The final format of the collected data shall be a .wav file.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>7A</strong></td>
<td>A system for measuring the cart’s deviation from a set ideal line may be constructed.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>7B</strong></td>
<td>The cart’s deviation from a set ideal line may be recorded from visual samples taken by an on-board camera.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>7C</strong></td>
<td>The deviation, if implemented, shall be measured by displacement of laser dots along a two-dimensional plane.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>7D</strong></td>
<td>If implemented, the system should measure the displacement to a minimum accuracy of 5 cm.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>8A</strong></td>
<td>Prototype designate Alpha-1 shall consist of Requirement Numbers 1A, 1B, 2A, 3A, 3B, 3C, 3D, 5A, and 5B.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>8B</strong></td>
<td>Prototype designate Alpha-2 shall consist of all previous Requirement Numbers for Alpha-1 as well as Numbers 2B, 2C, 2D, 3E, 4A, 4D, 4F, 5C, 6A, and 6B.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>9A</strong></td>
<td>The budget for this system shall be $200.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>9B</strong></td>
<td>If needed and at the discretion of the group members, the budget may be increased to a maximum of $400.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 9. List of Deliverables

The following is the list of deliverables for the prototype designate Alpha-1:

- A stable mobile cart shall be built to support the radar unit.
- The waveform shall be converted by an onboard ADC and stored as a 2D array.
• The communication to the SD card must allow for simple read and write operations.

• A rotary encoder shall be connected to the SoC and trigger a counter for each pulse.

• The OLED display must be able to present information relevant to the system’s operation in a text based format.

• A custom on-board driver shall be used to drive the screen in parallel to the processor.

• Push buttons shall be connected to the SoC and be configured to trigger vital actions such as interrupts.

• The components shall be connected by a breadboard or similar prototyping board.

The following is the list of deliverables for the prototype designate Alpha-2. The deliverables for Alpha-2 will include or, where appropriate, supersede the deliverables of Alpha-1:

• All electrical components shall be mounted to the cart and must be powered with an onboard battery.

• A deviation system based upon the emission of 3 laser lines in perfect parallel could be constructed and included in the design.

• The SoC might communicate to a SPI camera, take images, and store said images to an SD card by way of a buffer.

• A pulley system shall be mounted on a wheel and successfully integrated with the rotary encoder. The pulley system shall be 3D printed and either belt or gear driven.

• The user interface must allow the system to start, stop, and control some basic variables.

• The system shall take data every 10 cm (with a tolerance of 0.5 cm) and store the collected data as a 2D array. The 2D array shall be stored on an SD card so that the data can be imported in MATLAB for processing.
10. Block Diagrams

Block diagram of prototype designate Alpha-1:
Block diagram of prototype designate Alpha-2:
### 11. Breakdown of Work

<table>
<thead>
<tr>
<th>Component Description</th>
<th>Type of Work</th>
<th>Percentage (est.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Building Mobile Platform</td>
<td>Implementation</td>
<td>25%</td>
</tr>
<tr>
<td>Integrating Wheel Encoder onto Platform</td>
<td>Implementation</td>
<td>5%</td>
</tr>
<tr>
<td>Designing Wheel Encoder-SoC Interface</td>
<td>Design</td>
<td>10%</td>
</tr>
<tr>
<td>Programming SoC</td>
<td>Design</td>
<td>30%</td>
</tr>
<tr>
<td>Interfacing SD card with SoC using FatFs</td>
<td>Design</td>
<td>10%</td>
</tr>
<tr>
<td>Circuit Prototype Layout</td>
<td>Design</td>
<td>5%</td>
</tr>
<tr>
<td>Circuit Prototype Building</td>
<td>Implementation</td>
<td>10%</td>
</tr>
<tr>
<td>Radar Integration</td>
<td>Implementation</td>
<td>5%</td>
</tr>
</tbody>
</table>

### 12. Analysis of Top 3 Risks

<table>
<thead>
<tr>
<th>Risk</th>
<th>Prob. of Occurring</th>
<th>Severity</th>
<th>Correction Time</th>
<th>Project Delay</th>
<th>Mitigation Plans</th>
<th>Action Plans</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard Libraries for SPI Digital Camera not adaptable to solution</td>
<td>10%</td>
<td>5</td>
<td>1 Week</td>
<td>2 Weeks</td>
<td>Research available libraries prior to camera arrival. Investigate feasible physical configurations.</td>
<td>Search and apply other available open source libraries to solution. Make modifications to library specific to solution</td>
</tr>
<tr>
<td>Non-negligible drift associated with mechanical configuration of prototype's wheels</td>
<td>20%</td>
<td>3</td>
<td>3 days</td>
<td>3 days</td>
<td>Autodesk modeling for mechanical parts. Initial testing with alpha-1 phase.</td>
<td>Reconstruct mechanical wheel assembly. Compensate for drift by alignment. Possible change to axle assembly</td>
</tr>
<tr>
<td>Encoder fails to provide accurate measurements to within 10 cm of actual position throughout prototype path.</td>
<td>33%</td>
<td>7</td>
<td>Up to part reorder, ~ 3 weeks</td>
<td>3 weeks</td>
<td>Pre-emptive coding and testing with encoder, 3D modeling mechanical assembly in Autodesk.</td>
<td>Additional encoders with averaging algorithm may be employed. More accurate encoder part may replace original. Replace mechanical encoder to wheel assembly. Abandon encoder feature if time restricted.</td>
</tr>
</tbody>
</table>
13. System Key Functional Requirements

<table>
<thead>
<tr>
<th>Req. #</th>
<th>Requirement</th>
<th>Subcategory</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The coffee-can synthetic aperture radar unit designed for imaging must be available for use.</td>
<td>Input</td>
</tr>
<tr>
<td>2</td>
<td>The analog signal output from the radar system that interfaces with the ADC should have a voltage range of ( \pm 1V ).</td>
<td>Input</td>
</tr>
<tr>
<td>3</td>
<td>The power supply to the board must be 3.3V; using the 12V battery power supply, the voltage must be regulated down.</td>
<td>Input</td>
</tr>
</tbody>
</table>

14. Timetable

<table>
<thead>
<tr>
<th>Description of Work</th>
<th>Start and End Dates</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Phase One</strong></td>
<td></td>
</tr>
<tr>
<td>Project Start</td>
<td>8/17/2016</td>
</tr>
<tr>
<td>Project Proposal</td>
<td>8/31/2016</td>
</tr>
<tr>
<td>Feasibility and Project Plan</td>
<td>9/14/2016</td>
</tr>
<tr>
<td>Design Requirements</td>
<td>9/21/2016</td>
</tr>
<tr>
<td>Preliminary Design Review</td>
<td>10/5/2016</td>
</tr>
<tr>
<td><strong>Phase Two</strong></td>
<td></td>
</tr>
<tr>
<td>Alpha-1 Prototype Demo</td>
<td>10/26/2016</td>
</tr>
<tr>
<td>Final Design</td>
<td>11/2/2016</td>
</tr>
<tr>
<td><strong>Phase Three</strong></td>
<td></td>
</tr>
<tr>
<td>Exam</td>
<td>11/14/2016</td>
</tr>
<tr>
<td>Alpha-2 Prototype Demo</td>
<td>11/23/2016</td>
</tr>
<tr>
<td>Final Presentation &amp; Exhibit</td>
<td>12/2/2016</td>
</tr>
</tbody>
</table>

15. Budget

<table>
<thead>
<tr>
<th>Description of Work</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Phase One</strong></td>
<td></td>
</tr>
<tr>
<td>Feasibility and Design</td>
<td>$160.00</td>
</tr>
<tr>
<td><strong>Phase Two</strong></td>
<td></td>
</tr>
<tr>
<td>Initial Implementation of Design</td>
<td>$20.00</td>
</tr>
<tr>
<td><strong>Phase Three</strong></td>
<td></td>
</tr>
<tr>
<td>Design Adjustments and Prototyping</td>
<td>$20.00</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>$200.00</td>
</tr>
</tbody>
</table>

16. Key Personnel

- **Professor**            | Dr. Hai Ho               |
- **Faculty Advisor**      | Dr. Theodore Grosch      |
- **Project Manager**      | Nicholas Testin          |
- **Team**                 | Philip Davis, Ian Dorell, Alexander Gillespie, Nicholas Testin |
17. Evaluation of Project Outcomes

The Phase 1 evaluation is considered successful with the submission of near-complete files detailing schematics and plans for the construction of the radar unit additions. These include, but are not limited to, wiring schematics, pseudocode, and block diagrams.

The Phase 2 evaluation is considered successful with the submission of finalized revisions of all of the above documents, along with the demonstration of an initial prototype based upon the designs.

The Phase 3 evaluation is considered successful with the collection and processing of data for the purpose of producing a two-dimensional and possibly a three-dimensional image of a static landmark on the Kennesaw State University campus, likely including the Globe on the Marietta campus and various other KSU landmarks. The evaluation will also include the demonstration of a second prototype iteration.
18. Circuit Schematic
19. Code Flowchart

Initialize Before main

Main.c

Delay then Initialize variables → Start Components → Create Interrupt Vectors → Initialize LCD screen

Operational Loop

Call User Menu method → Enable Interrupts → Set Counter Values → Mount SD is SPI mode → Fail

Write to LCD screen → Scan and create new file and Directories → reset file name → Success

Check input

If back is pressed

If back is not pressed → Wait for encoder distance → Wait for clock for radar

Write to LCD

Write data to wave file → Take Radar data
20. Alpha-1 Deliverables

The requirements for Alpha-1 are included below, along with a rating of their acceptability towards the initially proposed requirements. In summary, all of the proposed requirements for Alpha-1 were either met or exceeded.

<table>
<thead>
<tr>
<th>Req. #</th>
<th>Requirement</th>
<th>Subcategory</th>
<th>Meets?</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>1A</td>
<td>A cart designed and built to support the radar system shall be constructed.</td>
<td>Feature Set</td>
<td>Yes</td>
<td>9/10</td>
</tr>
<tr>
<td>1B</td>
<td>The cart shall, at minimum, support the radar system plus an additional 25 pounds.</td>
<td>Functional Req.</td>
<td>Yes</td>
<td>10/10</td>
</tr>
<tr>
<td>2A</td>
<td>A system-on-a-chip (SoC) dedicated to the data logging shall be included with the existing radar circuitry.</td>
<td>Feature Set</td>
<td>Yes</td>
<td>10/10</td>
</tr>
<tr>
<td>2E</td>
<td>The SoC shall read data from the radar system at a minimum rate of 100 KHz.</td>
<td>Performance</td>
<td>Yes</td>
<td>10/10</td>
</tr>
<tr>
<td>3A</td>
<td>A rotary encoder shall provide the distance the cart has traveled to the SoC.</td>
<td>Feature Set</td>
<td>Yes</td>
<td>8/10</td>
</tr>
<tr>
<td>3B</td>
<td>The rotary encoder shall be interfaced with at least one wheel/axle assembly on the cart.</td>
<td>Functional Req.</td>
<td>Yes</td>
<td>8/10</td>
</tr>
<tr>
<td>3C</td>
<td>The interface between the encoder and wheel/axle assembly shall consist of a 3D printed pulley.</td>
<td>Hardware</td>
<td>Yes</td>
<td>10/10</td>
</tr>
<tr>
<td>3D</td>
<td>The encoder shall increment a counter with each pulse, directly indicating the distance the cart has traveled.</td>
<td>Operational Req.</td>
<td>Yes</td>
<td>7/10</td>
</tr>
<tr>
<td>4B</td>
<td>The ADC shall store the data currently collected from the radar system to a system memory device.</td>
<td>Output</td>
<td>Yes</td>
<td>10/10</td>
</tr>
<tr>
<td>4C</td>
<td>The ADC shall store the data in Req. 4A when it receives a signal from a manually-triggered button.</td>
<td>Input</td>
<td>Yes</td>
<td>10/10</td>
</tr>
<tr>
<td>4E</td>
<td>The data stored to the board or system memory shall be in the form of a 2D array.</td>
<td>Software</td>
<td>Yes</td>
<td>10/10</td>
</tr>
<tr>
<td>4G</td>
<td>The communications between the SoC and system memory shall permit, at minimum, simple read and write operations.</td>
<td>Software</td>
<td>Yes</td>
<td>10/10</td>
</tr>
<tr>
<td>5A</td>
<td>An organic light-emitting diode (OLED) screen will display information relating to the status of the logging of data from the radar system.</td>
<td>Feature Set</td>
<td>Yes</td>
<td>9/10</td>
</tr>
<tr>
<td>5B</td>
<td>The OLED screen shall display whether the SoC is collecting data from the radar system at the moment.</td>
<td>Output</td>
<td>Yes</td>
<td>9/10</td>
</tr>
<tr>
<td>8A</td>
<td>Prototype designate Alpha-1 shall consist of Requirement Numbers 1A, 1B, 2A, 3A, 3B, 3C, 3D, 5A, and 5B.</td>
<td>Schedule</td>
<td>Yes</td>
<td>130/140</td>
</tr>
</tbody>
</table>
21. Changes from Alpha-1 to Alpha-2

With the Alpha-1 prototype completed, work transitions towards the completion of Alpha-2, a prototype demonstrating a more robust proof-of-concept for the data logging system. The primary goal of Alpha-2 is for the entirety of the data logging system to be integrated into the existing radar unit and for full functionality to exist between all components. The integration of the radar unit would make its output available for reading by the data logging system, instead of the simulated output from a potentiometer present in Alpha-1. These criteria are the core functionality of Alpha-2; their presence would indicate a successful Alpha-2 prototype.

However, several additional features, not included in the initial proposal, materialized which would provide additional functionality or ease-of-use, especially when considered as part of a potential final or production model. The most practical or useful of the features were selected for implementation with the data logging system, and are included below.

A system for allowing the collection of radar data at several fixed heights, known internally as the “tiering” system, is the most notable of additional requirements. Sets of data collected at the same linear locations, but at different heights, can be processed to produce a three-dimensional image of the target. The minimum distance for this feature to be viable to the end-user was given as 20 feet. This feature was requested after the submission of project requirements, and as such was not included as an Alpha-2 requirement.

During the construction and testing of Alpha-1 requirements, moving the cart proved to be awkward and challenging due to its short size, which requires the operator be bent over or on their knees in order to use the cart. This manner of operation serves as frustrating and difficult method of use by a hypothetical end-user. A handle, set at a comfortable height from the cart, would allow for easier cart mobility and a far more satisfying method of use.

The hypothetical end-user of the product might desire that the saved data sets contain more samples, or take data at a shorter interval; however, it would be difficult to modify the system’s code after the fact in order to account for such requests. Therefore, in order to provide more of such functionality, a menu providing the ability to select such features shall be created. The options included in the menu are interval distance, sample size, and a toggle switch for a debugging mode.

Finally, an internal desire of the group was for the components of the data logging system to be connected in a more permanent configuration using a copper-plated prototyping board. This would be a more professional and compact option than the solderless breadboard configuration currently used.

Overall, Alpha-2 should not only demonstrate the core functionality of the data logging system but also prove to be a more robust product presenting greater ease-of-use and end-user control.
# 22. Alpha-2 Deliverables

The requirements for Alpha-2 are included below, along with a rating of their acceptability towards the initially proposed requirements. In summary, all of the proposed requirements for Alpha-2 were either met or exceeded.

<table>
<thead>
<tr>
<th>Req. #</th>
<th>Requirement</th>
<th>Subcategory</th>
<th>Meets?</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>2B</td>
<td>The SoC shall be interfaced with the radar system for the purpose of reading its incoming data.</td>
<td>Software</td>
<td>Alpha-2</td>
<td>10/10</td>
</tr>
<tr>
<td>2C</td>
<td>The data from the radar system shall be constantly read by the analog-to-digital converter (ADC) on the SoC.</td>
<td>Software</td>
<td>Alpha-2</td>
<td>10/10</td>
</tr>
<tr>
<td>2D</td>
<td>The system shall store data from the ADC to an array when a counter interrupt is triggered.</td>
<td>Software</td>
<td>Alpha-2</td>
<td>10/10</td>
</tr>
<tr>
<td>3E</td>
<td>The distance output of the encoder should possess error of no more than 0.5 cm per 10 cm of linear travel.</td>
<td>Operational Req.</td>
<td>Alpha-2</td>
<td>10/10</td>
</tr>
<tr>
<td>4A</td>
<td>The ADC shall collect data from the radar system upon receiving a signal from the encoder corresponding to a preselected distance traveled by the cart.</td>
<td>Feature Set</td>
<td>Alpha-2</td>
<td>10/10</td>
</tr>
<tr>
<td>4D</td>
<td>The ADC shall store the data in Req. 4A when it receives a signal from the rotary encoder.</td>
<td>Input</td>
<td>Alpha-2</td>
<td>10/10</td>
</tr>
<tr>
<td>4F</td>
<td>The minimum operational frequency for the data recorded shall be 50 KHz.</td>
<td>Operational Req.</td>
<td>Alpha-2</td>
<td>10/10</td>
</tr>
<tr>
<td>5C</td>
<td>The OLED screen may display more detailed information gathered from the other components of the data logging system.</td>
<td>Interface</td>
<td>Alpha-2</td>
<td>10/10</td>
</tr>
<tr>
<td>6A</td>
<td>The system memory, containing the collected data, shall be interfaced with an external computer so that the data can be transferred for processing.</td>
<td>Functional Req.</td>
<td>Alpha-2</td>
<td>7/10</td>
</tr>
<tr>
<td>6B</td>
<td>The final format of the collected data shall be a .wav file.</td>
<td>Operational Req.</td>
<td>Alpha-2</td>
<td>10/10</td>
</tr>
<tr>
<td>8B</td>
<td>Prototype designate Alpha-2 shall consist of all previous Requirement Numbers for Alpha-1 as well as Numbers 2B, 2C, 2D, 3E, 4A, 4D, 4F, 5C, 6A, and 6B.</td>
<td>Schedule</td>
<td>Alpha-2</td>
<td>97/100</td>
</tr>
</tbody>
</table>
23. Conclusions

The platform, radar system, PSoC device, and encoder all combine to a successful system and greatly reduce the effort required to obtain data for mapping large 3D objects. This design effectively addresses the most critical shortcomings of the previous radar system, whose problems includes an unstable physical platform, inadequate ADC sampling rate, live sampling with a laptop, and manual positioning and triggering of the radar.

Most, if not all, systems and subsystems required work further than what was initially estimated. Due to the PSoC 4 lacking native SD card support, the SD card storage interface required additional hours past initial estimations to adapt an existing library to the correct configuration. Action items with the PSoC introduced difficulty above and beyond original expectations, including: the integration of a comparator for radar clock signal synchronization to appropriate clock voltage; the performing of encoder logic within system memory limitations; and satisfactory data upload speeds to write ADC results to the SD card.

Though the project fulfilled all requirements, including several additional quality of life changes, there are shortcomings to the design. The PSoC 4 is limited by a system memory of 16 Kbytes, making several operations performed by the PSoC complicated to implement. Additional system memory, as well as direct memory access (DMA) in the forms of a field-programmable gate array (FPGA), would help address these limitations of the PSoC 4. Self-propulsion, path deviation tracking systems, and further construction on the physical cart assembly are all additional ideas for future iterations of this design. Overall, these considerations would be welcome feature additions to the cart, but not necessary to its core operations.

With the conclusion of the project, the data logging system was delivered to the customer, ready for use in collecting radar data or for further expansion by a future senior design group.
24. Sources

25. Partial Annotated Bibliography


Dr. Skolnik’s textbook is the defining text for an introductory course on radar systems. It has been revised twice from its initial edition in 1962 to accommodate the rapidly changing radar industry, as well as to reflect Dr. Skolnik's vast experience. The book is straightforward in its language, and clearly combines the technical material and equations required for success in a radar course with a simple and direct style that appears to be easy to understand.

I foresee using this text as the predominant source for many of the equations that I will cite within my report. It will also be a helpful resource in deciphering much of the dense language in the other reports I will invariably be using. I would imagine, finally, that it would be helpful just in my own knowledge and understanding of the radar system's theory. For this reason, I can see this text as being valuable above all others for my final report.


This website is the origin of Dr. Grosch’s existing coffee can radar system. The authors, professors at MIT, created it as a sort of online “class” that is available for anyone to take, complete with lecture notes, project details, and lab exercises. The end of the “course” should result in a working radar system constructed from coffee cans.

The online course should serve as the predominant resource for the construction and behavior of the radar system itself. Considering its intended audience of hobbyists and students, it should also serve as a good foundational resource for the basics of RF engineering, in a similar manner to Dr. Skolnik’s textbook.


Stove’s article explores features of FMCW radar, including how reflected noise affects the system, as well as practical considerations such as methods of increasing FMCW’s strengths and offsetting its weaknesses in performance. The author also emphasizes FMCW radar’s usefulness due to its compatibility with solid state (as opposed to vacuum tube) amplifiers. Stove also includes application examples such as an automatic cruise control system and a “stealth” navigational radar.

While Stove chooses to focus on applications where the antenna is monitoring mobile targets (which is not within the scope of this project), the author’s details on FMCW will prove useful in the simplified case of collecting data of static objects at range.
Dr. Grosch writes about utilizing a reference target during data collection to correct for phase or modulation error. His method is a single-time “autofocus” technique that allows for precise modulation without a drifting depth of focus away from the calibration point, which results in a performance increase as well as a computational cost increase during the radar’s operation.

Since Dr. Grosch is the faculty advisor and sponsor of this project, it is immediately obvious how useful his research would be into its success. His paper also explains the origin of the radar system I am working with. Finally, it will probably help to explain some of his offsets and adjustments of the data that will be collected.

Ms. Parrish summarizes different frequency modulation signals in use with FMCW systems and their analysis within the program MATLAB. She includes derivations for several signal types including sawtooth and sinusoidal signals as well as respective methods for analysis by hand. She also discusses the use of different simulation schemes within MATLAB such as fast Fourier transform and the effects of error from noise and phase shifting.

The article should prove useful during the final leg of this project: the analysis of the signals collected using the radar system. While I have used MATLAB in some previous courses, I have never used it to analyze externally-sourced signals. The language is rather dense, but one could hardly ask for more detail in a report of a similar scope.

Griffiths’ article explores the use of frequency modulation in radar to obtain range data, in contrast with (at the time of the article) the popular method of pulse information. The author explores some of the advantages of FMCW radar, including a low probability of intercept (for military applications) and longer range (in FMICW form), as well as examples such as a radio altimeter. The author also discusses the effect of error on the radar system and the application of a single radar unit into a synthetic aperture radar system.

This article will prove useful explaining the theory behind the operation of the radar system. It is a fairly dense article, but its language is simple enough to suggest an easy approach for someone with some knowledge in the field.
The authors describe the analysis of continuous-wave signals and the results of fast Fourier transforms on the signal. They also describe the effects of gating, i.e. limiting saturation, on the radar transmitter and receiver and the different errors or artifacts which may occur. They also explain the design of a tracking system for an object moving at high speed, as well as the simulation of radar noise for analytic purposes. The authors conclude with a dissection of their process for analyzing a spectrum of measured data which includes noise from the radar (in the form of echos).

This article will prove useful in the processing of data collected from the radar system. It appears to be focused more on data collected from moving targets, and perhaps slightly large antennas or antenna arrays, but much of the theory appears to be applicable to our synthetic aperture radar system. The information in the article would become useful if we need to increase the signal-to-noise ratio of our antenna signal in order to get more useful data.


The authors present in this paper a mathematical model for defining how well a FMCW system performs, by measuring beat frequencies between transmitted and received signals using a fast Fourier transform. The authors explain some of the theory and benefits of processing data in this manner, as well as several of the drawbacks such as the detection ability of the radar system.

The method in this paper explains exactly how we will process the data collected by the radar system when it is operated. By comparing the beat frequencies of the sent and received signals, useful information (such as object density and thickness) can be obtained, and a two-dimensional model can be constructed of the measured range. This paper will be a vital part of explaining the actions performed during the processing of the data after collection.
26. Glossary

Amplitude: The distance between the maximum value and zero of a signal.
Antenna: A transducer used for emitting an electromagnetic signal into space.
Directional antenna: An antenna which radiates primarily in a single direction.
Duplexer: A device used to separate or combine two EMF signals.
Electromagnetic frequency (EMF) wave: a signal induced by alternating electric and magnetic fields and propagating in a perpendicular direction.
Frequency: The number of times a particular waveform in a signal occurs per second. Inverse of period. Inversely related to wavelength.
Gain: The increase in output power of an antenna. Measured with reference to the isotropic antenna.
Isotropic antenna: A theoretical antenna which radiated evenly in all directions.
Main lobe: The primary beam of a directional antenna.
Omnidirectional antenna: An antenna which radiates approximately evenly along a plane.
Period: The amount of time to complete a single waveform in a single. Inverse of frequency.
Phase: The offset of a repeating signal, measured in degrees or radians.
Radio frequency (RF) wave: A subset of EMF waves, possessing low frequencies and high wavelengths.
Range: The distance to a target in a radar’s field of measurement.
Sidelobe: A secondary, usually unintended beam pointed in a diverging direction on an antenna.
Signal to noise ratio (SNR): A ratio which details the power a useful signal has over the noise which the system detects. Higher is better.
Target: A detected object within a radar’s field of measurement.
Wavelength: The distance between peaks of a repeating waveform. Inversely related to frequency.
27. Appendix A: Drawing of Encoder-Wheel Interface Mount
# 28. Appendix B: Bill of Materials

<table>
<thead>
<tr>
<th>Qty</th>
<th>Part #</th>
<th>Description</th>
<th>Manufacturer</th>
<th>Cost</th>
<th>Extended Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CY8CKIT-043</td>
<td>CY8CKIT-043 PSoC® 4 M-Series Prototyping Kit</td>
<td>Cypress</td>
<td>$9.98</td>
<td>$9.98</td>
</tr>
<tr>
<td>1</td>
<td>UPC: 610708262626</td>
<td>HiLetgo Stackable SD Card Shield for Arduino</td>
<td>HiLetgo</td>
<td>$6.79</td>
<td>$6.79</td>
</tr>
<tr>
<td>1</td>
<td>N/A</td>
<td>Diymall 0.96&quot; Inch Yellow Blue I2C IIC Serial OLED LED Module 12864 128X64 for Arduino Display 5</td>
<td>Diymall</td>
<td>$9.99</td>
<td>$9.99</td>
</tr>
<tr>
<td>4</td>
<td>Harbor Freight Item #62388</td>
<td>10 inch Pneumatic Tire</td>
<td>Haul Master</td>
<td>$3.99</td>
<td>$15.96</td>
</tr>
<tr>
<td>2</td>
<td>204604770</td>
<td>5/8 in. x 48 in. Zinc Plated Round Rod</td>
<td>Home Depot</td>
<td>$8.27</td>
<td>$16.54</td>
</tr>
<tr>
<td>1</td>
<td>B00FR19WQA</td>
<td>15Pcs 40 Pin 2.54 mm Single Row (L 11MM) Male Header</td>
<td>Vlonfine</td>
<td>$4.46</td>
<td>$4.46</td>
</tr>
<tr>
<td>1</td>
<td>N04452</td>
<td>BQLZR 600P/R Incremental Rotary Encoder DC5-24V Wide Voltage Power Supply 6mm Shaft</td>
<td>BQLZR</td>
<td>$15.99</td>
<td>$15.99</td>
</tr>
<tr>
<td>1</td>
<td>6484K237</td>
<td>XL Series Timing Belt, Trade No. 280xL037</td>
<td>McMaster-Carr</td>
<td>$7.52</td>
<td>$7.52</td>
</tr>
<tr>
<td>2</td>
<td>57105K14</td>
<td>XL Series Lightweight Timing Belt Pulley, 1.13&quot; OD</td>
<td>McMaster-Carr</td>
<td>$7.51</td>
<td>$15.02</td>
</tr>
<tr>
<td>1</td>
<td>N/A</td>
<td>Estimated Shipping Cost</td>
<td>McMaster-Carr</td>
<td>$6.00</td>
<td>$6.00</td>
</tr>
<tr>
<td>4</td>
<td>N/A</td>
<td>Hex Bolt 5/16 x 5 1/2 in</td>
<td>Home Depot</td>
<td>$0.63</td>
<td>$2.52</td>
</tr>
<tr>
<td>12</td>
<td>N/A</td>
<td>Cut washers 1/4 in</td>
<td>Home Depot</td>
<td>$0.11</td>
<td>$1.32</td>
</tr>
<tr>
<td>12</td>
<td>N/A</td>
<td>Hex nuts 1/4 in</td>
<td>Home Depot</td>
<td>$0.06</td>
<td>$0.72</td>
</tr>
<tr>
<td>1</td>
<td>N/A</td>
<td>1 5/8 inch black all purpose screws</td>
<td>Home Depot</td>
<td>$2.97</td>
<td>$2.97</td>
</tr>
<tr>
<td>1</td>
<td>N/A</td>
<td>2x4 weatherproofed wood</td>
<td>Home Depot</td>
<td>$3.57</td>
<td>$3.57</td>
</tr>
<tr>
<td>1</td>
<td>N/A</td>
<td>2/4 sanded plywood sheet</td>
<td>Home Depot</td>
<td>$11.05</td>
<td>$11.05</td>
</tr>
<tr>
<td>6</td>
<td>N/A</td>
<td>Square end balusters</td>
<td>Home Depot</td>
<td>$0.74</td>
<td>$4.44</td>
</tr>
<tr>
<td>1</td>
<td>N/A</td>
<td>Coarse drywall screws</td>
<td>Home Depot</td>
<td>$6.47</td>
<td>$6.47</td>
</tr>
<tr>
<td>2</td>
<td>N/A</td>
<td>Angle slot sheet zinc 72 inch</td>
<td>Home Depot</td>
<td>$12.74</td>
<td>$25.48</td>
</tr>
<tr>
<td>20</td>
<td>N/A</td>
<td>Fender washer 3/16 inch zinc</td>
<td>Home Depot</td>
<td>$0.14</td>
<td>$2.80</td>
</tr>
<tr>
<td>1</td>
<td>N/A</td>
<td>Fiberboard panel</td>
<td>Home Depot</td>
<td>$7.42</td>
<td>$7.42</td>
</tr>
<tr>
<td>1.5</td>
<td>N/A</td>
<td>Copper-plated circuit prototyping board</td>
<td>N/A</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>N/A</td>
<td>2x40 Hitachi HD44780-compatible LCD screen</td>
<td>N/A</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>N/A</td>
<td>3.3V voltage regulator</td>
<td>N/A</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>N/A</td>
<td>5V voltage regulator</td>
<td>N/A</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>N/A</td>
<td>Solderless circuit prototyping breadboard</td>
<td>Jameco</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>N/A</td>
<td>22 AWG stranded copper cable, 50 ft</td>
<td>N/A</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>N/A</td>
<td>22 AWG stranded copper cable, 15 ft</td>
<td>N/A</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>N/A</td>
<td>24 AWG stranded copper cable, 100 ft</td>
<td>N/A</td>
<td>$12.99</td>
<td>$12.99</td>
</tr>
<tr>
<td>1</td>
<td>N/A</td>
<td>Electrical repair solder: Rosin core 60/40, 0.062 inch, 4oz</td>
<td>Alpha Fry</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>N/A</td>
<td>Electrical solder: Rosin core 60/40, 0.031 inch, 0.5oz</td>
<td>Kester</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>N/A</td>
<td>Heat shrink tubing 3/4 inch</td>
<td>NTE Electronics</td>
<td>$2.59</td>
<td>$2.59</td>
</tr>
<tr>
<td>1</td>
<td>N/A</td>
<td>Heat shrink tubing 1/8 inch</td>
<td>NTE Electronics</td>
<td>$1.49</td>
<td>$1.49</td>
</tr>
<tr>
<td>1</td>
<td>N/A</td>
<td>Through-hole circuit connector</td>
<td>N/A</td>
<td>$0.89</td>
<td>$0.89</td>
</tr>
<tr>
<td>1</td>
<td>N/A</td>
<td>Screw-on spacers</td>
<td>Micro Center</td>
<td>$7.47</td>
<td>$7.47</td>
</tr>
</tbody>
</table>
29. Appendix C: PSoC 4 Diagrams & Code

Pin Layout Diagram:

Logic Diagram for Encoder:
Logic Diagram for Button Interface:
main.c

/*
* Contributors: 
* -Philip Davis
* -Ian Dorell
* -Alex Gillespie
* -Nicholas Testin
*
* Peripherals:
* -SD Shield 3.0
* -UART to USB Silicon LABS
* -LCD Screen 40x2
* -Rotory Encoder 600ppr
*
* Purpose:
*
*/

#include <project.h>
#include <radar.h>
#include <lcd_com.h>
#include <Const.h>
#include <diskio.h>
#include <ff.h>

#define PPM 1600 //Pulses per Meter
#define PPcM 16 //Pulses per centiMeter

char8 Ready=0; //Encoder Check variable 0=not ready | 1=Ready
char8 RClock=0; //The variable for the Radar Clock 0=Low | 1=High
char8 MenuRet=0; //MenuReturn
char8 input=0; //variable for button register storage and check
char8 Display_Count=0; //variable for option to display the distance counter on the LCD

unsigned long CLOCK_TIME; //The Time Delay for Writing to the LCD min=37uS
unsigned int DATA_SIZE; //the amount of variable to store for a single radar pulse
unsigned long COUNT_LENGTH; //the distance interval between each recorded pulse
char name[25]={0}; //file name buffer

//Encoder Interrupt which changes Ready: 0->1
CY_ISR(Counterint)
{
    Ready=1;
    Counter_WriteCounter(COUNT_LENGTH); //reloads the counter with the set
main.c

travel distance before next shot
Counter_ClearFIFO(); //clears interrupt ready registers

//interrupt for the button
CY_ISR(ButtonInt)
{
    int inputBuff; //creates buffer state
    input=StatusReg_Read(); //reads in button input
    inputBuff=input; //stores in buffer
    while(input==inputBuff) //waits until the buffer does not equal the button input.
    {
        // |this waits until the button is not pressed anymore
        input=StatusReg_Read(); // |effectively pausing the code while the button is held down
    }
    input=inputBuff;
    StatusReg_WriteMask(0b1111); //resets the interrupt register
}

int main()
{
    CyDelay(2000); //initial startup delay to allow everything to startup before communications
    name[24]=0; //sets the last value to 0 so that there is always set stop point
    // sets the entire string for the file name to 0
    //prevent reading from random memory locations
    //Set Const.h Variables for global control
    //initial values
    CLOCK_TIME=100; //MicroSeconds
    DATA_SIZE=2048;
    COUNT_LENGTH=160;
    //Enable Global Interrupts
    CyGlobalIntEnable;
    //Local Variables for main
    char8 error_return; //return for the error in making the SD file
    int i; //incrementing variable
    int buff[2048]; //buffer for incoming radar data
    FATFS fatFs; //FATfs custom variable
    uint8 resultF; //result for communicating with the SD card
    for(i=0;i<25;i++)
    {
        name[i]=0; //sets the entire string for the file name to 0
    }
}
main.c

//Start Components
ADC_Start();
UART_Start();
SPI_Start();
Counter_Start();
DistTracker_Start();
Timer_Start();

//Set Up Local Interrupts
Counter_SetInterruptMode(Counter_STATUS_ZERO_INT_EN_MASK);
isr.CounterInt_StartEx(CounterInt);
isr.ButtonInt_StartEx(ButtonInt);

//disable button interrupt
isr.ButtonInt_Disable();
StatusReg_InterruptDisable();

//Initialize LCD
LCDInitialize();

㎝

/************************************************************
Operational Loop Start
************************************************************

for(;;)
{

//Main input loop
isr.ButtonInt_Disable();
StatusReg_InterruptDisable();

User_Menu(); //user menu function. doesnt need to be a method but makes main.c easier to follow

CyDelay(100); //debouncing delay 100mS
input=0; //resets button input variable

//Enables the button interrupts
StatusReg_InterruptEnable();
isr.ButtonInt_Enable();

//Set Initial Counter
Counter_WriteCounter(COUNT_LENGTH); //sets the counter that triggers data collection
DistTracker_WriteCounter(32768); //sets total distance tracker to
effectively 0

do{
    resultF = f_mount(&fatFs, "", 1); //Mounts the SD card in SPI
    if(resultF != RES_OK) //checks the error
    {
        FatFsError(resultF);
        while(input==0);
    }
}while(resultF != RES_OK);

//File name reset
for(i=0;i<25;i++)
{
    name[i]=0;
}

//Make the SD File and Pause if there is an error

do{
    error_return=sd_MakeFile(); //creates the file name based off of the parameters set
    if(error_return==0x00) //if something went wrong it will return a 0
    {
        UART_UartPutString("Making the File returned an error
        LCDWriteString("Something went wrong");
    }
}while(error_return!=1);

//Main Data acquisition and storage loop
ClearScreen(); //clears LCD screen
LCDWriteString("Writing To: "); //writes stuff to the screen
LCDWriteString(name);
LCDWriteString("Shots:0");

i=1;
while(input!=129)
{

}

//Waiting Cycle

while(Ready!=1) //waits for the encoder to reach the distance interval marke.
{
    if(Display_Count==1) //
    {
        //
        }
    }

Page 4 of 6
main.c
{
  //
  displayCount();  //
  //
  
  if(input==129)  //Breaks if the back button is pressed
  {
    //
    break;
    //
  }  //
  //
  if(input==129)  //breaks if the back button is pressed
  {
    break;
  }

  RClock=FreqClock_Read();  //reads the clock pin for the radar
  //
  //
  while(RClock!=0)  //wait for low
  {
  //
  //
  //
  //
  *****          ************          ************ *******
  RClock=FreqClock_Read();  //
  //
  //
  //
  while(RClock!=1)  //waits for high
  {
  //
  //
  //
  //
  ************          ************
  RClock=FreqClock_Read();  //
  //

  //Read and Write cycle
  //Stops the user interrupts
  isr_ButtonInt_Disable();

  ADC_StartConvert();
capture_Wave(buff);  //Takes so many data points and stores in buff
ADC_StopConvert();

  //LED_Write(1u);

  sd_WriteFile(buff);  //writes buffered data to wave file with number #

  //LED_Write(0u);

  isr_ButtonInt_Enable();

  Ready=0;

}
main.c

//prints to LCD screen
LCDGo(0x46);
LCDWriteInt(i); //writes the amount of shots currently taken
i++;
}
#include <project.h>
#include <diskio.h>
#include <ff.h>
#include <radar.h>
#include <Const.h>
#include <sprintf.h>
#include <lcd_com.h>

#define PPCM 16

void FatFsError(FRESULT result);

void displayCount()
{
    int j=0;
    int j_=0;
    float distance;
    int k;

    j=Counter_ReadCounter(); //reads counter
    k=DistTracker_ReadCounter()-32768; //converts to a signed integer

    if(j>2)
    {
        if(j!=j_)
        {
            LCDGo(0x4A); //writes tracking data to the LCD screen
            LCDWriteInt(j); //
            LCDWriteString(" "); //
            LCDGo(0x4F); //
            LCDWriteFloat(distance); //
            LCDWriteString(" "); //
            LCDGo(0x58); //
            LCDWriteInt(k); //
            LCDWriteString(" "); //
            j_=j; //
        }
    }
}
void User_Menu()
{
    char8 input=0;
    char8 inputBuff=0;
    char8 i=0;

    while(input!=0b0001) //Loops while the back button is not pressed
    {
        //print the main menu
        if(input==0b0100) //increments if the up button is pressed
        {
            i+=1;
            if(i>2)
            {
                i=0;
            }
        }

        if(input==0b1000) //decrements if down button is pressed
        {
            i-=1;
            if(i>100) //unsigned so the value will be 0-255 so when <
                       //unsigned so the value will be 0-255 so when <
            {
                i=2;
            }
        }

        if(input==0b0010) //if select button is pressed
        {
            switch(i)
            {
                case 0: //if its the first
                while(input!=0b0001)
                {
                    ClearScreen();
                    LCDWriteString("\tShot Int Size");
                    LCDWriteString("\n");
                    LCDWriteInt(DATA_SIZE);
                    LCDWriteString("\n");
                    inputBuff=input;
                    while(input==inputBuff)
                    {
                        input=StatusReg_Read();
                    }
                    switch(input)
                    {
                        case 0b0100: if(DATA_SIZE<2048) DATA_SIZE=
                                          DATA_SIZE*2;
                        break;
                        case 0b1000: if(DATA_SIZE>256) DATA_SIZE=
                                          DATA_SIZE/2;
                        break;
                    }
                }
            }
        }
    }
}
Radar.c

break;
}
}
break;
case 1:
while(input!=0b0001)
{
  ClearScreen();
  LCDWriteString("\tDistance To Travel");
  LCDWriteString("\n");
  LCDWriteFloat((float)COUNT_LENGTH/16);
  LCDWriteString(" cm");
  inputBuff=input;
  while(input==inputBuff)
  {
    input=StatusReg_Read();
  }
  switch(input)
  {
    case 0b0100: COUNT_LENGTH=COUNT_LENGTH+8;
      break;
    case 0b1000: COUNT_LENGTH=COUNT_LENGTH-8;
      break;
  }
  break;
}
case 2:
while(input!=0b0001)
{
  ClearScreen();
  LCDWriteString("\tDisplay Count?");
  switch(Display_Count)
  {
    case 0: LCDWriteString("\nNo");
      break;
    case 1: LCDWriteString("\nYes");
      break;
  }
  inputBuff=input;
  while(input==inputBuff)
  {
    input=StatusReg_Read();
  }
  switch(input)
  {
    case 0b0100: Display_Count^=1;
      break;
    case 0b1000: Display_Count^=1;
      break;
  }
Radar.c

break;
}
}
ClearScreen();
LCDWriteString("\t Begin or Change Values");
LCDWriteString("\n Shot Size Interval Counter");

switch(i)
{
    case 0: LCDGo(0x40);
            LCDWriteString(" >");
            break;
    case 1: LCDGo(0x4B);
            LCDWriteString(" >");
            break;
    case 2: LCDGo(0x55);
            LCDWriteString(" >");
            break;
}
LCDGo(0x00);
inputBuff=input;
while(input==inputBuff)
{
    input=StatusReg_Read();
}
inputBuff=input;
while(input==inputBuff)
{
    input=StatusReg_Read();
}

void capture_Wave(int *data)
{
    unsigned int i=0;

    for(i=0;i<DATA_SIZE;i++)
    {
        ADC_IsEndConversion(ADC_WAIT_FOR_RESULT);
        data[i]=ADC_GetResult16(0u);
    }
}

char sd_MakeFile()
{
    FIL file;

    Page 4 of 10
uint8 resultF;
uint8_t i;
uint8_t j;
unsigned int writefs;
char buf[20]={0};
char dir[5];
dir[4]=0;
float distance;
int buffer;
name[5]=0;
name[7]=48;
const char init[48]={0x52,0x49,0x46,0x46,0xD4,0xAD,0x01,
0x00,0x57,0x41,0x56,0x45,0x66,0x6D,
0x74,0x20,0x10,0x00,0x00,0x00,0x01,
0x00,0x01,0x00,0x20,0xA1,0x07,0x00,
0x40,0x42,0x0F,0x00,0x02,0x00,0x10,
0x00,0x64,0x61,0x74,0x61,0xB0,0xAD,
0x01,0x00,0x00,0x00,0x00,0x00};
name[8]=0;
sprintf(buf,"%d",DATA_SIZE);
i=0;

while (buf[i]!=((char) 0))
{
    i++;
}
switch(i)
{
    case 3: i=0;
    name[0]='0';
    dir[0]='0';
    for(i=0;i<3;i++)
    {
        name[i+1]=buf[i];
        dir[i+1]=buf[i];
    }
    i++;
    f_mkdir(dir);
    break;
    case 4: i=0;
    while (buf[i]!=((char) 0))
    {
        name[i]=buf[i];
        dir[i]=buf[i];
        i++;
    }
    f_mkdir(dir);
    break;
for(j=0; j<20; j++)
{
    buf[j]=0;
}
name[i]='/';
i+=2;
name[i]='_';
i++;
distance=(float)COUNT_LENGTH/PPcM;
buffer=(int)distance;
sprintf(buf,"%d",buffer);
j=0;
while(buf[j]!=((char) 0))
{
    name[j+i+2]=buf[j];
    j++;
}
i+=j;
name[i+2]='_';
i++;
buffer=(distance-buffer)*10;
for(j=0; j>20; j++)
{
    buf[j]=0;
}
sprintf(buf,"%d",buffer);
j=0;
while(buf[j]!=((char) 0))
{
    name[j+i+2]=buf[j];
    j++;
}
i+=j;
name[i+2]='.';
i++;
name[i+2]='w';
i++;
name[i+2]='a';
i++;
name[i+2]='v';
i++;
name[i+2]=0;
name[6]='0';
for(j=49; j<58; j++)
{
    for(i=48; i<58; i++)
    {
        name[7]=i;
        resultF = f_open($file, name, FA_CREATE_NEW);
    }
}
if(resultF == 0x00) break;
}

if(resultF == 0x00) break;
name[6]=j;
}
f_close(&file);

resultF = f_open(&file, name, FA_WRITE);
ClearScreen();
LCDWriteString(name);

if(resultF != RES_OK)
{
    UART_UartPutString("\ntest\n");
    UART_UartPutString(name);
    FatFsError(resultF);
    for(;;);
    return 0x00;
}

f_write(&file,init,44,&writefs);
f_close(&file);
UART_UartPutString("\nDONE MAKING FILE: ");
UART_UartPutString(name);
UART_UartPutString("\n");
return 1;

}

void sd_WriteFile(int *data)
{

    FIL file;
    uint8 resultF;
    uint32_t size;
    char buff[2];
    unsigned int writefs;
    uint16_t j;

    resultF = f_open(&file, name, FA_WRITE);

    size=f_size(&file);
    f_lseek(&file,size);
    if(resultF != RES_OK)
    {

void FatFsError(FRESULT result)
{
    switch (result)
    {
    case FR_DISK_ERR:
        UART_UartPutString("\n error: (FR_DISK_ERR) low level error.\n");
        LCDWriteString("\nerror: (FR_DISK_ERR) low level error.");break;
        
    case FR_INT_ERR:
        UART_UartPutString("\n error: (FR_INT_ERR)\n");
        LCDWriteString("\nerror: (FR_INT_ERR)\n");break;
        
    case FR_NOT_READY:
        UART_UartPutString("\n error: (FR_NOT_READY) sdcard not ready.\n");
        LCDWriteString("\nerror: (FR_NOT_READY) sdcard not ready.");break;
        
    case FR_NO_FILE:
        UART_UartPutString("\n error: (FR_NO_FILE) invalid file.\n");
        LCDWriteString("\nerror: (FR_NO_FILE) invalid file.");break;
        
    case FR_NO_PATH:
        UART_UartPutString("\n error: (FR_NO_PATH) invalid path.\n");
        LCDWriteString("\nerror: (FR_NO_PATH) invalid path.");break;
        
    case FR_INVALID_NAME:
        UART_UartPutString("\n error: (FR_INVALID_NAME) invalid name.\n");
        LCDWriteString("\nerror: (FR_INVALID_NAME) invalid name.");break;
        
    case FR_DENIED:
        
    
}
UART_UartPutString("\n error: (FR_DENIED) operation denied.\n");
LCDWriteString("\nerror: (FR_DENIED) operation denied.");break;

case FR_EXIST:
    UART_UartPutString("\n error: (FR_EXIST) it exists yet...");
    LCDWriteString("\nerror: (FR_EXIST) it exists yet...");break;

case FR_INVALID_OBJECT:
    UART_UartPutString("\n error: (FR_INVALID_OBJECT)\n");
    LCDWriteString("\nerror: (FR_INVALID_OBJECT)\n");break;

case FR_WRITE_PROTECTED:
    UART_UartPutString("\n error: (FR_WRITE_PROTECTED)\n");
    LCDWriteString("\nerror: (FR_WRITE_PROTECTED)\n");break;

case FR_INVALID_DRIVE:
    UART_UartPutString("\n error: (FR_INVALID_DRIVE)\n");
    LCDWriteString("\nerror: (FR_INVALID_DRIVE)\n");break;

case FR_NOT_ENABLED:
    UART_UartPutString("\n error: (FR_NOT_ENABLED) sdcard unmounted.\n");
    LCDWriteString("\nerror: (FR_NOT_ENABLED) sdcard unmounted.\n");break;

case FR_NO_FILESYSTEM:
    UART_UartPutString("\n error: (FR_NO_FILESYSTEM) no valid FAT volume.\n");
    LCDWriteString("\nerror: (FR_NO_FILESYSTEM) no valid FAT volume.\n");break;

case FR_MKFS_ABORTED:
    UART_UartPutString("\n error: (FR_MKFS_ABORTED)\n");
    LCDWriteString("\nerror: (FR_MKFS_ABORTED)\n");break;

case FR_TIMEOUT:
    UART_UartPutString("\n error: (FR_TIMEOUT)\n");
    LCDWriteString("\nerror: (FR_TIMEOUT)\n");break;

case FR_LOCKED:
    UART_UartPutString("\n error: (FR_LOCKED)\n");
    LCDWriteString("\nerror: (FR_LOCKED)\n");break;

case FR_NOT_ENOUGH_CORE:
    UART_UartPutString("\n error: (FR_NOT_ENOUGH_CORE)\n");
    LCDWriteString("\nerror: (FR_NOT_ENOUGH_CORE)\n");break;

case FR_TOO_MANY_OPEN_FILES:
    UART_UartPutString("\n error: (FR_TOO_MANY_OPEN_FILES)\n");
    LCDWriteString("\nerror: (FR_TOO_MANY_OPEN_FILES)\n");break;
case FR_INVALID_PARAMETER:
    UART_UartPutString("\n error: (FR_INVALID_PARAMETER)\n");
    LCDWriteString("\nerror: (FR_INVALID_PARAMETER)");break;
    
default: {}; break;
}
void LCDInitialize()
{
    LCDRegisterSelect_Write(0b0);
    LCDComm_Write(0x01);
    LongClockPulse();
    LCDComm_Write(0x38);
    ClockPulse();
    LCDComm_Write(0x0C);
    ClockPulse();
    LCDComm_Write(0x06);
    ClockPulse();
    LCDComm_Write(0x01);
    LongClockPulse();
}

void ClearScreen()
{
    LCDRegisterSelect_Write(0b0);
    LCDComm_Write(0x01);
    LongClockPulse();
}

void LCDGo(char pl)
{
    if(pl>127)
    {
        return;
    }
    LCDRegisterSelect_Write(0b0);
    LCDComm_Write(0x01);
    LongClockPulse();
}

void LCDWriteString(const char8 write[])
```c
uint32_t writeIndex=0u;
LCDRegisterSelect_Write(0b1);
while(write[writeIndex] != ((char8) 0))
{
    if(write[writeIndex] == '\n' )
    {
        LCDRegisterSelect_Write(0b0);
        LCDComm_Write(0b10000000+0x40);
        ClockPulse();
        LCDRegisterSelect_Write(0b1);
        writeIndex++;
    }
    else if(write[writeIndex] == '\t' )
    {
        LCDRegisterSelect_Write(0b0);
        LCDComm_Write(0b10000000);
        ClockPulse();
        LCDRegisterSelect_Write(0b1);
        writeIndex++;
    }
    else
    {
        LCDComm_Write(write[writeIndex]);
        ClockPulse();
        writeIndex++;
    }
}

void ClockPulse()
{
    LCDClock_Write(1u);
    CyDelayUs(CLOCK_TIME);
    LCDClock_Write(0u);
    CyDelayUs(CLOCK_TIME);
}

void LongClockPulse()
{
    LCDClock_Write(1u);
    CyDelay(8);
    LCDClock_Write(0u);
    CyDelay(8);
}

void LCDWriteInt(int var)
{
    char buf[20]={0};
    sprintf(buf,"%d",var);
    LCDWriteString(buf);
}

void LCDWriteFloat(float var)
```
Lcd_com.c

{
    int i;
    int buffer=var;
    char buf[20]={0};
    sprintf(buf,"%d",buffer);
    LCDWriteString(buf);
    LCDWriteString(".");
    buffer=(var-buffer)*100;
    for(i=0;i>20;i++)
    {
        buf[i]=0;
    }
    sprintf(buf,"%d",buffer);
    LCDWriteString(buf);
}
/* [] END OF FILE */
30. Appendix D: Gantt Chart
Data logging system for a synthetic aperture radar unit.
<table>
<thead>
<tr>
<th>Name</th>
<th>Begin date</th>
<th>End date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Milestone: Project Start</td>
<td>8/17/16</td>
<td>8/17/16</td>
</tr>
<tr>
<td>Project: Project Proposal</td>
<td>8/17/16</td>
<td>9/1/16</td>
</tr>
<tr>
<td>Meet with Advisor and Decide Topic</td>
<td>8/17/16</td>
<td>8/20/16</td>
</tr>
<tr>
<td>Draft and Submit Proposal</td>
<td>8/21/16</td>
<td>8/31/16</td>
</tr>
<tr>
<td>Project: Feasibility and Project Plan</td>
<td>9/1/16</td>
<td>9/17/16</td>
</tr>
<tr>
<td>Milestone: Feasibility and Project Plan</td>
<td>9/16/16</td>
<td>9/16/16</td>
</tr>
<tr>
<td>Parts List</td>
<td>9/1/16</td>
<td>9/6/16</td>
</tr>
<tr>
<td>Feasibility Document Draft</td>
<td>9/7/16</td>
<td>9/14/16</td>
</tr>
<tr>
<td>Feasibility Document Review</td>
<td>9/15/16</td>
<td>9/16/16</td>
</tr>
<tr>
<td>Project: Design Requirements</td>
<td>9/17/16</td>
<td>9/22/16</td>
</tr>
<tr>
<td>Milestone: Design Requirements</td>
<td>9/21/16</td>
<td>9/21/16</td>
</tr>
<tr>
<td>Requirements Document Draft</td>
<td>9/17/16</td>
<td>9/19/16</td>
</tr>
<tr>
<td>Requirements Document Review</td>
<td>9/20/16</td>
<td>9/21/16</td>
</tr>
<tr>
<td>Project: Preliminary Design Review</td>
<td>9/1/16</td>
<td>10/27/16</td>
</tr>
<tr>
<td>Milestone: Preliminary Design Review</td>
<td>10/5/16</td>
<td>10/5/16</td>
</tr>
<tr>
<td>Preliminary Design Draft</td>
<td>9/22/16</td>
<td>10/3/16</td>
</tr>
<tr>
<td>Preliminary Design Review</td>
<td>10/4/16</td>
<td>10/5/16</td>
</tr>
<tr>
<td>Encoder Research</td>
<td>9/1/16</td>
<td>9/15/16</td>
</tr>
<tr>
<td>Design Mechanical-to-Encoder Interface</td>
<td>9/16/16</td>
<td>9/2/16</td>
</tr>
<tr>
<td>Programming Encoder</td>
<td>9/16/16</td>
<td>10/2/16</td>
</tr>
<tr>
<td>Integrate Encoder into Wheel/Axle</td>
<td>10/8/16</td>
<td>10/21/16</td>
</tr>
<tr>
<td>Optimize Encoder Integration</td>
<td>10/22/16</td>
<td>10/26/16</td>
</tr>
<tr>
<td>Learn API/Program ADC</td>
<td>9/1/16</td>
<td>9/14/16</td>
</tr>
<tr>
<td>USB and SD Card Config/Intergr.</td>
<td>9/15/16</td>
<td>9/28/16</td>
</tr>
<tr>
<td>Project: Alpha-1 Prototype Demo</td>
<td>9/1/16</td>
<td>10/27/16</td>
</tr>
<tr>
<td>Milestone: Alpha-1 Prototype Demo</td>
<td>10/26/16</td>
<td>10/26/16</td>
</tr>
<tr>
<td>Alpha-1 Prototype Documentation</td>
<td>10/6/16</td>
<td>10/26/16</td>
</tr>
<tr>
<td>Interface</td>
<td>9/29/16</td>
<td>10/12/16</td>
</tr>
<tr>
<td>Build Platform</td>
<td>9/1/16</td>
<td>9/28/16</td>
</tr>
<tr>
<td>Debugging and Optimization</td>
<td>10/13/16</td>
<td>10/26/16</td>
</tr>
<tr>
<td>Find Wheel and Axle System for Cart</td>
<td>9/29/16</td>
<td>9/30/16</td>
</tr>
<tr>
<td>Implement Wheel/Axle System</td>
<td>10/1/16</td>
<td>10/7/16</td>
</tr>
<tr>
<td>Implement Tiering System</td>
<td>10/8/16</td>
<td>10/14/16</td>
</tr>
<tr>
<td>Optimizing Tiering System</td>
<td>10/15/16</td>
<td>10/26/16</td>
</tr>
<tr>
<td>Project: Final Design Review</td>
<td>10/27/16</td>
<td>11/4/16</td>
</tr>
<tr>
<td>Milestone: Final Design Review</td>
<td>11/2/16</td>
<td>11/2/16</td>
</tr>
<tr>
<td>Name</td>
<td>Begin date</td>
<td>End date</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>------------</td>
<td>------------</td>
</tr>
<tr>
<td>Final Design Draft</td>
<td>10/27/16</td>
<td>11/1/16</td>
</tr>
<tr>
<td>Final Design - Review Draft</td>
<td>11/2/16</td>
<td>11/3/16</td>
</tr>
<tr>
<td><strong>Milestone: Exam</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11/18/16</td>
<td>11/18/16</td>
<td></td>
</tr>
<tr>
<td><strong>Project: Alpha-2 Prototype Demo</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10/27/16</td>
<td>11/24/16</td>
<td></td>
</tr>
<tr>
<td><strong>Milestone: Alpha-2 Prototype Demo</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11/23/16</td>
<td>11/23/16</td>
<td></td>
</tr>
<tr>
<td>Alpha-2 Prototype Documentation</td>
<td>11/4/16</td>
<td>11/23/16</td>
</tr>
<tr>
<td>Integrate all Systems on Cart</td>
<td>10/27/16</td>
<td>11/2/16</td>
</tr>
<tr>
<td>Collect Data for Processing</td>
<td>11/3/16</td>
<td>11/5/16</td>
</tr>
<tr>
<td>Process Collected Data</td>
<td>11/6/16</td>
<td>11/8/16</td>
</tr>
<tr>
<td>Convert to Soldered Prototyping Board</td>
<td>11/3/16</td>
<td>11/23/16</td>
</tr>
<tr>
<td><strong>Project: Final Presentation and Exhibit</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11/24/16</td>
<td>12/3/16</td>
<td></td>
</tr>
<tr>
<td><strong>Milestone: Final Presentation and Exhibit</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12/2/16</td>
<td>12/2/16</td>
<td></td>
</tr>
<tr>
<td>Preparations for Final Presentation</td>
<td>11/24/16</td>
<td>12/2/16</td>
</tr>
<tr>
<td>Name</td>
<td>Default role</td>
<td></td>
</tr>
<tr>
<td>------------------</td>
<td>----------------</td>
<td></td>
</tr>
<tr>
<td>Philip Davis</td>
<td>developer</td>
<td></td>
</tr>
<tr>
<td>Ian Dorell</td>
<td>developer</td>
<td></td>
</tr>
<tr>
<td>Alex Gillespie</td>
<td>developer</td>
<td></td>
</tr>
<tr>
<td>Nicholas Testin</td>
<td>project manager</td>
<td></td>
</tr>
</tbody>
</table>
Gantt Chart

Name | Begin | End date |
--- | --- | --- |
Milestone: Project Start | 8/17/16 | 8/17/16 |
Project: Project Proposal | 8/17/16 | 9/1/16 |
Meet with Advisor and Decide Topic | 8/17/16 | 8/20/16 |
Draft and Submit Proposal | 8/21/16 | 8/31/16 |
Milestone: Feasibility and Project Plan | 9/1/16 | 9/1/16 |
Parts List | 9/1/16 | 9/6/16 |
Feasibility Document Draft | 9/7/16 | 9/14/16 |
Feasibility Document Review | 9/15/16 | 9/16/16 |
Milestone: Design Requirements | 9/17/16 | 9/17/16 |
Requirements Document Draft | 9/17/16 | 9/19/16 |
Requirements Document Review | 9/20/16 | 9/21/16 |
Milestone: Preliminary Design Review | 9/21/16 | 9/21/16 |
Preliminary Design Draft | 9/22/16 | 9/22/16 |
Preliminary Design Review | 9/23/16 | 9/23/16 |
Encoder Research | 9/23/16 | 9/23/16 |
Design Mechanical-to-Encoder Interface | 9/24/16 | 9/24/16 |
Programming Encoder | 9/25/16 | 9/25/16 |
Integrate Encoder into Wheel/Axle | 9/26/16 | 10/2/16 |
Optimize Encoder Integration | 9/27/16 | 10/3/16 |
Learn API/Program ADC | 9/28/16 | 9/28/16 |
Milestone: Alpha-1 Prototype Demo | 10/1/16 | 10/1/16 |
Build Platform | 9/29/16 | 9/30/16 |
Debugging and Optimization | 10/1/16 | 10/2/16 |
Find Wheel and Axle System for Cart | 10/3/16 | 10/4/16 |
Implement Wheel/Axle System | 10/5/16 | 10/6/16 |
Implement Tiering System | 10/6/16 | 10/7/16 |
Optimizing Tiering System | 10/8/16 | 10/9/16 |
Milestone: Final Design Review | 10/27/16 | 10/27/16 |
Final Design Draft | 10/27/16 | 10/27/16 |
Final Design - Review Draft | 11/1/16 | 11/1/16 |
Milestone: Exam | 11/1/16 | 11/1/16 |
Project: Alpha-2 Prototype Demo | 10/27/16 | 10/27/16 |
Milestone: Alpha-2 Prototype Demo | 11/2/16 | 11/2/16 |
Alpha-2 Prototype Documentation | 11/3/16 | 11/3/16 |
Integrate all Systems on Cart | 11/4/16 | 11/4/16 |
Collect Data for Processing | 11/5/16 | 11/5/16 |
Process Collected Data | 11/6/16 | 11/6/16 |
Convert to Soldered Prototyping Board | 11/7/16 | 11/7/16 |
Milestone: Final Presentation and Exhibit | 11/24/16 | 11/24/16 |
Final Presentation and Exhibit | 11/24/16 | 11/24/16 |
Preparations for Final Presentation | 11/24/16 | 11/24/16 |
### Resources Chart

<table>
<thead>
<tr>
<th>Name</th>
<th>Default role</th>
<th>2016</th>
</tr>
</thead>
<tbody>
<tr>
<td>Philip Davis</td>
<td>developer</td>
<td>50% 50% 50% 50% 75% 150%</td>
</tr>
<tr>
<td>Ian Dorell</td>
<td>developer</td>
<td>50% 50% 150% 145% 130%</td>
</tr>
<tr>
<td>Alex Gillespie</td>
<td>developer</td>
<td>50% 50% 125% 125% 75%</td>
</tr>
<tr>
<td>Nicholas Testin</td>
<td>project manager</td>
<td>50% 50% 125% 125% 125% 125%</td>
</tr>
</tbody>
</table>