Passively Powered pH Sensor for Study of Gastric Disorders

Team Members:
Kevin Hart
Sean Hart
Tucker Lundgren
Nicholas Piaquadio

Faculty Advisor:
Dr. Reena Dahle

May 9, 2019
ABSTRACT

A fully passive wireless implantable pH sensor that is implantable in an animate silicone stomach environment has been developed for the study of gastric disorders. The system has applications in medical training and testing. It can be used for pH monitoring as well as testing the efficiency of antacid medication. To achieve this, wireless power is sent from a reading circuit, via a class E amplifier connected to an inductive coil. The implanted circuit harvests the energy sent with a charge pump, and returns the measured pH via Frequency Shift Keying (FSK) modulation. The electronic components were simulated using Keysight Advanced Design System (ADS), prototyped on breadboards, amended, and finally manufactured onto Printed Circuit Boards (PCBs). To make the stomach, an injection molding process was employed using a sacrificial wax inner core and a 3D printed mold. The completed stomach model features 5mm thick walls at life-size scale, and demonstrates realistic digestive motion. The current implant design uses a traditional pH probe for proof of concept, fits within the stomach at 2.5cm by 5cm, and is capable of returning readings at a distance of up to 4 inches with an accuracy within 0.2pH. The reader coil can read pH once every ten minutes for 43 hours on a single charge.

KEYWORDS:

Electrical Engineering Wireless Power, Inductive Coupling, 3-D Printing, Passive Sensing, Gastro-Intestinal Disorders, Implantable Electronics Injection Molding
## Contents

1 Introduction ........................................ 8  
   1.1 Motivation ....................................... 8  
   1.2 Project Description ............................... 8  
   1.3 Scope of this Report .............................. 9  

2 Project Overview .................................. 11  
   2.1 Functional Description ......................... 11  
   2.2 Engineering Constraints ....................... 12  
   2.3 Costs ............................................. 15  
   2.4 Implementation ................................ 15  
   2.5 Tools and Software .............................. 19  
   2.6 Relevant Courses ................................. 20  

3 Theory and Background ........................... 21  
   3.1 Passive Powering ................................ 21  
      3.1.1 Class E Power Amplifier .................... 21  
      3.1.2 Inductive Coupling .......................... 23  
      3.1.3 Charge Pumps ................................ 25  
   3.2 pH Measurement ................................... 26  
   3.3 Load Modulation .................................. 27  
      3.3.1 Relaxation Oscillators ....................... 28  
      3.3.2 FSK Modulation ............................... 30  
      3.3.3 Envelope Detection ......................... 31  
   3.4 Signal Conditioning ............................. 32  
      3.4.1 Sallen-Key Filters ........................... 32  
      3.4.2 Nyquist Sampling Theorem ................. 35  
      3.4.3 The Discrete Hartley Transform .......... 36  
   3.5 Stomach Properties .............................. 37  

4 Design and Simulation ............................ 38  
   4.1 Coupling Network ................................ 38  
   4.2 Transponder Electronics ....................... 40  
      4.2.1 Charge Pump ................................ 40  
      4.2.2 Relaxation Oscillator ....................... 42  
      4.2.3 Load Modulation Switching ................. 45  
   4.3 Reader Electronics .............................. 47
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.3.1 Amplifier</td>
<td>48</td>
</tr>
<tr>
<td>4.3.2 Envelope Detector</td>
<td>50</td>
</tr>
<tr>
<td>4.3.3 Signal Conditioning Chain (Initial Prototype)</td>
<td>51</td>
</tr>
<tr>
<td>4.3.4 Signal Conditioning Chain (Final Design)</td>
<td>53</td>
</tr>
<tr>
<td>4.4 PCB Layouts</td>
<td>55</td>
</tr>
<tr>
<td>4.4.1 Transponder Layout</td>
<td>55</td>
</tr>
<tr>
<td>4.4.2 Reader Layout</td>
<td>57</td>
</tr>
<tr>
<td>4.5 MATLAB GUI</td>
<td>59</td>
</tr>
<tr>
<td>4.6 Microcontroller Program</td>
<td>62</td>
</tr>
<tr>
<td>4.7 Stomach Modeling</td>
<td>64</td>
</tr>
<tr>
<td>4.7.1 Stomach Design</td>
<td>64</td>
</tr>
<tr>
<td>4.7.2 Fluid Simulation</td>
<td>67</td>
</tr>
<tr>
<td>5 Implementation</td>
<td>68</td>
</tr>
<tr>
<td>5.1 Coil Manufacturing</td>
<td>68</td>
</tr>
<tr>
<td>5.2 Breadboard Prototyping</td>
<td>71</td>
</tr>
<tr>
<td>5.3 Electronics Manufacturing</td>
<td>75</td>
</tr>
<tr>
<td>5.3.1 Transponder Packaging</td>
<td>75</td>
</tr>
<tr>
<td>5.3.2 Reader Packaging</td>
<td>77</td>
</tr>
<tr>
<td>5.4 Stomach Manufacturing</td>
<td>79</td>
</tr>
<tr>
<td>5.5 Gastric Movements</td>
<td>84</td>
</tr>
<tr>
<td>6 Measurement Results and Discussion</td>
<td>86</td>
</tr>
<tr>
<td>6.1 Charge Pump Evaluation</td>
<td>86</td>
</tr>
<tr>
<td>6.2 Coil Evaluation and System Read Range</td>
<td>87</td>
</tr>
<tr>
<td>6.3 System Measurements</td>
<td>91</td>
</tr>
<tr>
<td>7 Conclusion and Future Work</td>
<td>93</td>
</tr>
<tr>
<td>References</td>
<td>94</td>
</tr>
<tr>
<td>8 Appendix</td>
<td>96</td>
</tr>
<tr>
<td>8.1 Appendix 1: Full Gant Chart</td>
<td>96</td>
</tr>
<tr>
<td>8.2 Appendix 3: MATLAB GUI Program</td>
<td>98</td>
</tr>
<tr>
<td>8.3 Appendix 2: Microcontroller Program</td>
<td>118</td>
</tr>
</tbody>
</table>
List of Tables

1  Engineering Constraints .............................................. 13
2  Engineering Standards and Specifications .......................... 14
3  Component List .......................................................... 15
4  List of Project Phases ................................................... 17
5  System Measurements of Common Beverages ....................... 91
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>System Block Diagram</td>
<td>11</td>
</tr>
<tr>
<td>2</td>
<td>Work Breakdown Structure</td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td>Senior Design 1 Gantt Chart</td>
<td>17</td>
</tr>
<tr>
<td>4</td>
<td>Senior Design 2 Gantt Chart</td>
<td>18</td>
</tr>
<tr>
<td>5</td>
<td>Class E Amplifier Diagram</td>
<td>22</td>
</tr>
<tr>
<td>6</td>
<td>Mutual Coupling</td>
<td>24</td>
</tr>
<tr>
<td>7</td>
<td>4 Stage Dickenson Charge Pump</td>
<td>25</td>
</tr>
<tr>
<td>8</td>
<td>Charge Pump Functionality</td>
<td>26</td>
</tr>
<tr>
<td>9</td>
<td>Basic pH Measurement Setup</td>
<td>27</td>
</tr>
<tr>
<td>10</td>
<td>Relaxation Oscillator</td>
<td>28</td>
</tr>
<tr>
<td>11</td>
<td>Relaxation Oscillator Waveform</td>
<td>29</td>
</tr>
<tr>
<td>12</td>
<td>Voltage Modulated Relaxation Oscillator</td>
<td>30</td>
</tr>
<tr>
<td>13</td>
<td>FSK Modulated Waveform</td>
<td>31</td>
</tr>
<tr>
<td>14</td>
<td>Envelope Detector</td>
<td>32</td>
</tr>
<tr>
<td>15</td>
<td>Sallen Key Low-Pass Filter</td>
<td>33</td>
</tr>
<tr>
<td>16</td>
<td>Quality Factor Plot</td>
<td>34</td>
</tr>
<tr>
<td>17</td>
<td>Aliasing in Time Domain</td>
<td>35</td>
</tr>
<tr>
<td>18</td>
<td>Aliasing in Frequency Domain</td>
<td>35</td>
</tr>
<tr>
<td>19</td>
<td>Stomach Diagram</td>
<td>37</td>
</tr>
<tr>
<td>20</td>
<td>System Level Simulations</td>
<td>38</td>
</tr>
<tr>
<td>21</td>
<td>Schematic for Simulation of Coupling Network</td>
<td>39</td>
</tr>
<tr>
<td>22</td>
<td>Frequency Sweep Simulation of Coupling Network</td>
<td>40</td>
</tr>
<tr>
<td>23</td>
<td>ADS Charge Pump Implementation</td>
<td>41</td>
</tr>
<tr>
<td>24</td>
<td>Charge Pump Simulation Results</td>
<td>42</td>
</tr>
<tr>
<td>25</td>
<td>ADS Relaxation Oscillator Schematic</td>
<td>43</td>
</tr>
<tr>
<td>26</td>
<td>ADS Relaxation Oscillator Output</td>
<td>43</td>
</tr>
<tr>
<td>27</td>
<td>ADS Voltage Controlled Relaxation Oscillator</td>
<td>44</td>
</tr>
<tr>
<td>28</td>
<td>Voltage Controlled Relaxation Oscillator Outputs</td>
<td>45</td>
</tr>
<tr>
<td>29</td>
<td>Schematic for Simulation of Load Modulation Switching</td>
<td>46</td>
</tr>
<tr>
<td>30</td>
<td>Voltage Waveforms from Simulation of Load Modulation Switching</td>
<td>47</td>
</tr>
<tr>
<td>31</td>
<td>LTspice Model of Class E Amplifier</td>
<td>48</td>
</tr>
<tr>
<td>32</td>
<td>LTspice Waveforms for Class E Amplifier</td>
<td>49</td>
</tr>
<tr>
<td>33</td>
<td>ADS Envelope Detector</td>
<td>50</td>
</tr>
<tr>
<td>34</td>
<td>Envelope Detector Simulation Result</td>
<td>51</td>
</tr>
<tr>
<td>35</td>
<td>ADS Filtering Module</td>
<td>52</td>
</tr>
<tr>
<td>36</td>
<td>Frequency Simulation of Filtering Schematic</td>
<td>53</td>
</tr>
<tr>
<td>Page</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>Frequency Simulation of Filtering Schematic</td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>Frequency Simulation of Filtering Schematic</td>
<td></td>
</tr>
<tr>
<td>39</td>
<td>Transponder PCB Layout</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>Transponder Top Traces</td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>Transponder Bottom Traces</td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>Reader Board Layout</td>
<td></td>
</tr>
<tr>
<td>43</td>
<td>MATLAB Generated Graphical Display and Interface</td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>MATLAB Code for Alignment Feedback</td>
<td></td>
</tr>
<tr>
<td>45</td>
<td>Main Loop of Arduino Program</td>
<td></td>
</tr>
<tr>
<td>46</td>
<td>ADC Sampling Mechanism Function Definition</td>
<td></td>
</tr>
<tr>
<td>47</td>
<td>Compute FHT Peak Function Definition</td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>SolidWorks Stomach Core[1]</td>
<td></td>
</tr>
<tr>
<td>49</td>
<td>SolidWorks Stomach Cover Top[1]</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>SolidWorks Stomach Cover Bottom[1]</td>
<td></td>
</tr>
<tr>
<td>51</td>
<td>SolidWorks Simulation[1]</td>
<td></td>
</tr>
<tr>
<td>52</td>
<td>Manufactured Reader Coil</td>
<td></td>
</tr>
<tr>
<td>53</td>
<td>3-D Printed Coil Winding Stand</td>
<td></td>
</tr>
<tr>
<td>54</td>
<td>First Inductor Manufacturing Iteration using Coil Winding Stand</td>
<td></td>
</tr>
<tr>
<td>55</td>
<td>Second Inductor Manufacturing Iteration using Coil Winding Stand</td>
<td></td>
</tr>
<tr>
<td>56</td>
<td>Breadboard Implementation of Transponder</td>
<td></td>
</tr>
<tr>
<td>57</td>
<td>Breadboard Implementation of Excitation Circuit</td>
<td></td>
</tr>
<tr>
<td>58</td>
<td>Breadboard Implementation of Signal Conditioning network</td>
<td></td>
</tr>
<tr>
<td>59</td>
<td>Breadboard Implementation of Arduino Communication Circuit</td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>Manufactured Transponder PCB</td>
<td></td>
</tr>
<tr>
<td>61</td>
<td>Silicone Coated Transponder with pH Probe</td>
<td></td>
</tr>
<tr>
<td>62</td>
<td>Final Reader PCB</td>
<td></td>
</tr>
<tr>
<td>63</td>
<td>Reader Packaging</td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>Reader Being Worn</td>
<td></td>
</tr>
<tr>
<td>65</td>
<td>3D Printed Core</td>
<td></td>
</tr>
<tr>
<td>66</td>
<td>Core Mold Making</td>
<td></td>
</tr>
<tr>
<td>67</td>
<td>Wax Casting</td>
<td></td>
</tr>
<tr>
<td>68</td>
<td>Wax Core</td>
<td></td>
</tr>
<tr>
<td>69</td>
<td>3D Printed Cover</td>
<td></td>
</tr>
<tr>
<td>70</td>
<td>3D Printed Cover</td>
<td></td>
</tr>
<tr>
<td>71</td>
<td>Silicone Injection Setup</td>
<td></td>
</tr>
<tr>
<td>72</td>
<td>Silicone Stomach</td>
<td></td>
</tr>
<tr>
<td>73</td>
<td>Actuation Assembly</td>
<td></td>
</tr>
<tr>
<td>74</td>
<td>Charge Pump Read Range Comparison</td>
<td></td>
</tr>
<tr>
<td>Page</td>
<td>Section Description</td>
<td>Page</td>
</tr>
<tr>
<td>------</td>
<td>---------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>75</td>
<td>Reader Coil H-Field Pattern at 0 Mils</td>
<td>87</td>
</tr>
<tr>
<td>76</td>
<td>Reader Coil H-Field Pattern at 100 Mils</td>
<td>88</td>
</tr>
<tr>
<td>77</td>
<td>Tag Coil H-Field Pattern at 0 Mils</td>
<td>88</td>
</tr>
<tr>
<td>78</td>
<td>Tag Coil H-Field Pattern at 100 Mils</td>
<td>89</td>
</tr>
<tr>
<td>79</td>
<td>Coil Read Range Parameters</td>
<td>90</td>
</tr>
<tr>
<td>80</td>
<td>Two Dimensional Coil Range</td>
<td>90</td>
</tr>
<tr>
<td>81</td>
<td>Dissolving Rate of Antacids in System</td>
<td>92</td>
</tr>
<tr>
<td>82</td>
<td>Full Gantt Chart Sr. Design 1</td>
<td>96</td>
</tr>
<tr>
<td>83</td>
<td>Full Gantt Chart Sr. Design 2</td>
<td>97</td>
</tr>
</tbody>
</table>
1 Introduction

This design provides technology that allows doctors to monitor stomach pH of patients in a manner that is both non-invasive and affordable. The proposed system can also be employed for comprehensive testing of pharmaceutical antacids.

1.1 Motivation

The recent increase in Gastrointestinal (GI) issues in the United States calls for innovation in this field. Specifically, a recent poll found that 74 percent of Americans are living with GI discomfort [1]. The team aims to technologically aid the treatment of these issues, and develop a system that can potentially assist in research and development related to GI issues.

Conventional pH monitoring systems involve inserting a sensor via a catheter down a patient’s throat. To continuously monitor the patient’s pH, the tube must remain inside of the patient’s throat to maintain an electrical connection. The product presented in this paper will allow for non-invasive monitoring of a patient’s pH because it is wireless, and battery-less due to the fact that it is passively powered by an external reader. Following endoscopic insertion of the device, doctors can track pH over time, and synthesize the data with minimal discomfort in the patient. This will streamline the diagnosis of GI ailments.

Also included in this project is a realistic model stomach to act as a test environment. This model was implemented using 3-D printing and actuated to mimic a stomach environment, and is made to replace the necessity of harming animals during in-vivo testing. The model is also used to facilitate comprehensive tests on different types of antacids. The system designed lends itself to pharmaceutical testing and innovation in this branch of medicine which will further help combat GI issues.

1.2 Project Description

The purpose of this project is to design an implantable stomach pH sensor and its corresponding reader system. The functionality of the prototype will be tested within a realistic model stomach. This project represents a more sophisticated method of monitoring the symptoms of various Gastrointestinal (GI) disorders, which are very common among the American population [1]. This setup would, fully realized, allow real time pH readings to be taken wirelessly, assisting doctors in monitor diseases like Gastroesophageal Reflux Disease (GERD), stomach ulcers, and stomach cancer [3]. In addition, the modelled stomach setup provides a testing environment for antacid medications as well as a form of practice for professionals learning stomach surgery. For the patient, compared to conventional technologies,
the design provides a more comfortable experience in that it is both batteryless and wireless, making the implant process smoother, and reducing the overall footprint of the sensor.

The project team will deliver fully functional PCBs of both the reader and transponder circuits, a stomach model that incorporates realistic motion and the cycling in and out of fluids with different pHs, a Graphical User Interface (GUI) for the system, efficient transmitter and receiver coils, and a signal processing mechanism that corresponds the modulation of the signal to pH. Along the way, simulations of the stomach model and electrical circuits will also be produced.

1.3 Scope of this Report

This report will cover the following aspects of the project

- Transponder Electronics
  - ADS Design and simulation of transponder components including the charge pump, relaxation oscillator, and load modulation switching system
  - Optimization of resonant circuit for transponder coil
  - Implementation of breadboard model and issues encountered
  - Generation of PCB using Eagle Cad and manufacturing the final transponder package
  - Manufacturing the transponder coil

- Reader Electronics
  - ADS Design and simulation of transponder components including a power amplifier, envelope detector and signal conditioning network
  - Optimization of resonant circuit for reader coil
  - Implementation of breadboard model and issues encountered
  - Generation of PCB using Eagle Cad and manufacturing the final reader package
  - Manufacturing the reader coil

- Micro-controller and Software
  - Design of Matlab GUI to interpret user commands and provide an intuitive feedback and data display
- Implementation of Arduino Microcontroller Program to communicate between MATLAB and Reader
- Implementation of code to configure Arduino ADC and process the data to decode stomach pH via a Hartley Transformation

- Gastric Test Environment
  - CAD design and simulation for 3D manufacturing and design
  - A realistic stomach model that demonstrates realistic physical and electrical properties
  - A system that creates gastric movement on the stomach model
2 Project Overview

This section provides a high level overview of the project including an overall functional description and the factors evaluated and the planning carried out during the design process.

2.1 Functional Description

To better understand how the components of the system interact, a block diagram has been provided as reference in Figure 1 to accompany the explanation below it.

![System Block Diagram](image)

Figure 1: System Block Diagram

The model stomach shown in the left-hand portion of Figure 1 will be embed the pH sensor as well as the transponder PCB in order to take readings and relay the information. A fluid pump is connected to the stomach model in order to cycle fluids of different pH to show the response of the device to pH changes. The transponder and the reader will each be equipped with inductor coils for wireless communication through the model stomach via magnetic induction. The reader shown on the right of the above figure will power the transponder, and the transponder will communicate pH readings back to the reader by modulating the load at a frequency corresponding to the voltage supplied by the pH sensor.
The reader demodulates, and conditions the load modulated signal and then sends it to an Arduino Microcontroller for digital conversion and processing. Finally the Arduino sends the processed data to Matlab where the results are shown to a user via a Graphical User Interface (GUI). Additionally, this GUI also allows a user to control Readings taken by the system.

2.2 Engineering Constraints

Table 1 shows the engineering constraints that are applicable to this project.
The main design constraints include the low power requirements of the transponder electronics and considerations of the monitored patients’ comfort corresponding to the footprint of both the implant and reader. The implant must also be able to withstand highly acidic solutions, and the reader should be easy to use by non-engineering professionals such as doctors. The project will be considered to be successful under the condition that a transponder and reader have been produced on PCB, the transponder has been implanted into a functional

<table>
<thead>
<tr>
<th>Specification/ Constraint</th>
<th>Reason</th>
<th>Project Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low power requirements for transponder electronics</td>
<td>Transponder is powered wirelessly and maximum read range is needed</td>
<td>Incorporate low power electronics in the design, and compensate for their flaws</td>
</tr>
<tr>
<td>Small footprint allocation for implant</td>
<td>Implant needs to fit inside model stomach</td>
<td>Manufacture a proper PCB that optimizes real estate, use high frequency to shrink components</td>
</tr>
<tr>
<td>Read range must be as high as possible</td>
<td>The system must be usable by patients of various sizes, and reader to transponder alignment may not be perfect</td>
<td>Use RF design techniques to ensure perfect matching, optimize transponder power harvesting, focused antenna/ coil design</td>
</tr>
<tr>
<td>Chemical resistance of stomach model, and implant</td>
<td>Solutions with low pH values must be testable</td>
<td>Manufacture stomach model with resistant material, coat transponder electronics with resistant material</td>
</tr>
<tr>
<td>IEEE-C95.1-2005 (Regulation on Human Exposure to RF EM Fields)[2]</td>
<td>System must comply with Human Health Standards</td>
<td>Maximize read range efficiently, keeping total power under 1W and turn on only to take readings.</td>
</tr>
<tr>
<td>User friendliness of system interface</td>
<td>System must be usable by Non-Engineering Professionals (Doctors)</td>
<td>Make a graphical user interface that facilitates complete control over system</td>
</tr>
<tr>
<td>Implant needs to easily attach to internal of model stomach</td>
<td>Prototype must be able to be tested efficiently</td>
<td>Manufacture model stomach with internal affixture, and design transponder PCB accordingly</td>
</tr>
</tbody>
</table>
model stomach, and readings can be taken to an accuracy of 0.2 pH on a GUI.

Due to time and resource constraints, this project will not include design, and experimentation on medical implantation methodology for the device. Likewise, in vivo testing is deemed unreasonable due to the budget. The pH sensor will be purchased, as will all operational amplifiers (op-amps), and the microprocessor that is used for signal analysis.

The design will follow from prior work on a similar type of sensor, developed by Cao et al. [2], which measured pH and impedance of reflux in the esophagus, but seeks to make improvements to read range, overall circuit size, and to move the sensing into the stomach to look at other types of disorders. The modeled testing environment is also novel.

This design will be implemented under the assumption that medical professionals will use this product in a safe and useful manner. The team is also assuming that the biological stomach model is an accurate representation of human tissue, and that sufficient power can be transmitted wirelessly from the reader to the transmitter through human tissue. Finally, it is assumed that the pH sensor will remain stable and give reliable readings.

The prevailing risk in this project is that it may not be possible to transmit enough power through tissue to turn on the transponder. Following from this, there is some concern about the availability of low power op-amps and Metal Oxide Semiconductor Field Effect Transistors (MOSFETs) that may impact the power consumption of the finished product. The official engineering standards and specifications that needed to be followed in this project are listed in Table 2.

Table 2: Engineering Standards and Specifications

<table>
<thead>
<tr>
<th>Engineering Standard</th>
<th>Where it was used</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEEE C95.1 -2005</td>
<td>Wireless power transfer between Reader and Transponder that is intended to pass through the human body</td>
</tr>
<tr>
<td>FCC Policy on Human Exposure to RF EM Fields</td>
<td>Wireless power transfer between Reader and Transponder that is intended to pass through the human body</td>
</tr>
<tr>
<td>FCC ISM SRD Band Delegation</td>
<td>Frequency Band Delegation for Wireless Power Transfer, and Reader Telecommunications</td>
</tr>
</tbody>
</table>

The main standard that applies to this project is the IEEE C95.1 -2005 which regulates the amount of human exposure to high frequency radiation. We also would need to abide by the allocation of frequency bands by the FCC if this project were to be used in a clinical environment.
2.3 Costs

Table 3 shows all the expenditures necessary to complete this project.

Table 3: Component List

<table>
<thead>
<tr>
<th>Component</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>pH Probes</td>
<td>$98.58</td>
</tr>
<tr>
<td>Prototype 1 Transponder Electronics</td>
<td>$76.26</td>
</tr>
<tr>
<td>Prototype 1 Reader Electronics</td>
<td>$75.98</td>
</tr>
<tr>
<td>Transponder PCB Electronics</td>
<td>$35.70</td>
</tr>
<tr>
<td>Reader PCB Electronics</td>
<td>$92.80</td>
</tr>
<tr>
<td>Inductor Manufacturing Materials</td>
<td>$9.98</td>
</tr>
<tr>
<td>3-D Printed Stomach Molds</td>
<td>$126.54</td>
</tr>
<tr>
<td>Model Stomach Materials</td>
<td>$310.92</td>
</tr>
<tr>
<td>PCB Printing</td>
<td>$89.50</td>
</tr>
<tr>
<td>Microcontroller and Communication</td>
<td>$22.99</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>$938.63</strong></td>
</tr>
</tbody>
</table>

2.4 Implementation

This project is separated into four main modules including the design the implementation of the mechanical stomach model, realization of the transponder to be embedded in the stomach, the reader to power the transponder, and to obtain information, and finally the task of integrating all of the components together. Prior to prototyping each design will be simulated, the mechanical aspects using Solidworks [8], and the electrical aspects utilizing Keysight Advanced Design Systems (ADS) [9]. Additionally each component will be modeled and tested on a breadboard before printed circuit board (PCB) implementation.

The work breakdown structure is shown in Figure 2.
Each of the work packages will be assigned to a team member, however this should not manifest the misconception that each component will not be a team effort. Team members are encouraged to obtain assistance and consultation from teammates. Each macro design component will be broken up into separate modules of differing functionality. For example, the transponder system will be subdivided into the components: energy harvesting circuit, relaxation oscillator, and the transmitting/receiving coil. The functionality of each of these components and that of the pH sensor will be verified separately before bringing them together in order to avoid overloading any of the electronics and to simplify debugging. A list of the project iterations is shown in Table 4.
Table 4: List of Project Phases

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iteration 1</td>
<td>Breadboard Implementation of transponder, with bucket and pipe to test, and Oscilloscope to take/process readings</td>
</tr>
<tr>
<td>Iteration 2</td>
<td>Breadboard Implementation of transponder and reader circuit, with crude stomach model for testing, and matlab GUI with JTAG communication with controller of reader</td>
</tr>
<tr>
<td>Iteration 3</td>
<td>PCB Implementation of transponder and reader circuit, with sophisticated stomach model for testing and Bluetooth communication between MATLAB GUI and Reader.</td>
</tr>
</tbody>
</table>

The Gantt Chart used for Senior Design 1 is provided in Figure 3 in compacted form, while the full chart can be found in Appendix VI, Figure A1.

![Figure 3: Senior Design 1 Gantt Chart](image)

In Figure 8, the summary blocks and milestones are presented for the first semester, with blocks containing critical path tasks outlined in red. Each summary task includes the name of every member working on any task within it. Research proceeded for about three weeks after the signing of the project charter, followed by electrical and mechanical simulations and the manufacture and characterization of the pH probe. This has allowed the electrical parts to be ordered on 10/17, and inductor winding has had some progress made. Two important milestones have been passed, namely the completion of simulations and the ordering of the electrical components for the prototype. From there, it is anticipated that the silicone stomach will be ready on 11/6, the pH measurement system on 11/12, and
the first fully integrated prototype on 11/29. All of these will be first iterations, with a simple stomach model, breadboarded components, and pH sensor outside of a casing. The critical path stems from the one week delay in ordering components, and therefore follows the steps leading up to the order, research and simulation, through the electrical assembly and integration. While inductors are being made in the interim, the process doesn’t take up the whole gap, and therefore the parts still push the following tasks into critical. Because the parts are already ordered on time, the end date should not be pushed any further by this consideration.

As for Senior Design 2, the compact Gantt Chart is provided in Figure 4, while the full version can be found in Appendix VI Figure A2.

Figure 4: Senior Design 2 Gantt Chart

From Figure 4, the summary blocks and milestones are presented for the second semester, with blocks containing critical path tasks outlined in red. Each summary task includes the name of every member working on any task within it. As planned, the second semester begins with a few days to evaluate the performance of Senior Design 1 before moving on to iterate on the prototype. One of the large concerns is lead time on PCB manufacture, so, after making the second iteration of the electronics in the first week, Nick will immediately move to PCB planning to order the boards as soon as possible. This planning has kept the PCB from impacting the critical path. The major milestones include sending the PCB out for manufacture on 2/11, also marking the second iteration of the circuit design, and the second iteration of the stomach, expected to finish on 2/25. Kevin and Sean, in the meantime will optimize the coupling between the two inductors, potentially using antennas instead, meaning the improved coupling setup will be complete on 2/26. From there, it is anticipated that the microprocessor will be finished on 3/14, the electronics will be built on 3/21, the group will be able to measure through organic material on 4/14, and the second
version of the full setup will be ready for testing on 4/18. The critical path here stems from Kevin and Seans work schedule, since Nick and Tucker bounce between a few tasks each that allow their schedules some flexibility. Because the microprocessor, app, and electronics are critical, Kevin and Seans task mean that the critical path goes from evaluation, to range improvement, to microprocessor and app coding, to electronics testing, and last, of course to full system integration.

2.5 Tools and Software

The following tools and software were necessary for the successful development, testing and documentation of our project.

- Storage and Documentation
  - Google Drive
  - Microsoft Project Scheduling
  - Latex with Overleaf IDE
  - Microsoft PowerPoint

- Design Software
  - Keysight ADS
  - Solidworks
  - Eagle CAD
  - Arduino IDE

- Simulation Software
  - Keysight ADS
  - MATLAB
  - ANSYS

- Testing Equipment
  - Rigol Function Generator
  - Kesight Oscilloscope
  - LCR Meter
2.6 Relevant Courses

The following courses provided knowledge necessary for the completion of this project.

Relevant Electrical/Computer Engineering Courses:

- EGE200 Circuit Analysis
- EGE3XX Signals and Systems
- EGE320 Electronics I
- EGE321 Electronics II
- EGE331 Computer Simulation
- EGC331 Microprocessors
- EGE340 Applied Electromagnetics
- EGE412 Communication Systems Theory
- EGE493 Applied Digital Signal Processing
- EGE493 Introduction to Microwave Engineering

Relevant Mechanical Engineering Courses:

- EGM211 Statics
- EGM221 Engineering Materials
- EGM302 Finite Element Analysis
- EGM311 Kinematics of Machines
- EGM312 System Dynamics
- EGM322 Mechanics of Materials
- EGM332 Fluid Mechanics
3 Theory and Background

In this section includes an overview of the theoretical principles that are applicable to our project. The following descriptions are necessary to understand how the project functions and also to highlight the reasoning for the design choices that were made.

3.1 Passive Powering

Being that the implant must run without a battery, a variety of specialized circuitry was incorporated into the design in order to transmit power from the reading device and render it usable at the implant. This process runs through the amplification of the high frequency signal to power transmission by inductive coupling and rectification by charge pump.

3.1.1 Class E Power Amplifier

The square signal produced by a standard clock oscillator is, in general, too low in power for use in any kind of transmission. Furthermore, it contains unwanted odd-order harmonics of its fundamental frequency, whereas resonant circuits only accept a single carrier frequency. A class E amplifier, shown in Figure 5, solves both of these issues.
From Figure 5, the MOSFET switch at the left of the diagram toggles on and off, allowing a DC current, $I_{DC}$, to flow from a DC source at the top of the diagram. The choke inductor ensures only DC current can flow to or from the source. The only time the MOSFET has a current running through it is when it is closed, meaning there is no voltage drop across it, assuming its drain-source resistance is an ideal zero. Likewise, when the switch is open it has a voltage but no current. This means that the voltage is zero when the current is not and vice versa, giving a theoretical maximum efficiency of 100 percent [4]. The internal capacitance of the MOSFET $C_S$ charges and discharges during this process. The circuit labeled $f_0$ is called a series LC tank circuit [3]. The circuit only has zero impedance at resonance, and high impedance otherwise, so it selects for a particular frequency, and can pick the fundamental frequency of the square wave or any of its odd harmonics. At high frequency, a matching network is required to match the source impedance to the load, but in all cases the oscillations can reach 2.5-4 times the source voltage [4]. Equations 1 and 2 give the efficiency of a class E amplifier with a non-zero drain-source resistance, under the assumption of a matched load [3]:

$$\eta = \frac{P_{out}}{P_{in}} = 1 - \frac{P_{diss}}{P_{out}}$$

(1)
\[ P_{\text{diss}} = I_{RMS}^2 R_{\text{on}} + \frac{1}{4\pi} \frac{V_{\text{SW}}^2}{X_{CS}} \]  

In Equation 2, \( P_{\text{diss}} \) is the power dissipated in the MOSFET, where \( V_{\text{SW}} \) is the maximum voltage produced at the drain. This allows only two components of power loss in the class E amplifier, dissipation in the resistance of the MOSFET and in the parasitic capacitance, \( C_S \) down to ground. As such, the choice of MOSFET must minimize \( C_S \) and \( R_{\text{on}} \) while ensuring that \( V_{DS\text{max}} \) is greater than \( V_{\text{sw}} \). This usually necessitates a power MOSFET since the voltage swing at the drain can reach four times the DC voltage.

### 3.1.2 Inductive Coupling

A critical theoretical component of this design is wireless power transfer, specifically, inductive coupling. For a simple resonant LC circuit, the resonant frequency is given by Equation 3:

\[ \omega_0 = \frac{1}{\sqrt{LC}} \quad f_0 = \frac{1}{2\pi\sqrt{LC}} \]  

From Equation 3, the resonant radian frequency is simple one over the square root of \( LC \), and the frequency in Hertz is the same quantity divided by \( 2\pi \). This type of circuit can be used to sustain the oscillations used to drive inductive coupling.

The roots of inductive coupling itself are embedded in Ampere’s law and Faraday’s law of induction, the former dictates that when a time changing current is sent through a coil, it produces a time changing magnetic field which travels wirelessly through the medium that the coil is suspended in, the later allows the magnetic field to induce a time changing voltage on a second coil that is facing the transmitting coil [4]. This process is illustrated in Figure 6.
The frequency independent factor of Inductive coupling is quantified as the mutual inductance (M) in Henrys. The mutual inducance between two inductors can be found using Equation 4:

\[
M_{21} = k \sqrt{L_1 L_2}
\]

In Equation 4, the coupling coefficient (k), is the measure of the ability of a coupling network to wirelessly transfer power. The closer k is to unity, the more efficient a coupling network is. The coupling coefficient is a function of the distance between the two coils, the electromagnetic properties of the medium that the coils are in, and the characteristic impedances of both the transmitting coil, and the receiving coil. An engineer must take these factors into account while designing an inductive coupling network.

A more formal and descriptive definition of mutual inductance is the Neumann formula which is shown as Equation 5:

\[
M_{21} = \frac{\mu_0}{4\pi} \int_2 \int_1 \frac{\delta x_2 \cdot \delta x_1}{|x_2 - x_1|}
\]

Equation 5 is clearly not very useful for analytical calculations. However, it can easily be solved for any realistic situation using numerical methods, and also reveals some applicable properties of mutual inductance. First of all, it shows that the mutual inductance only depends on the size and shape of the two coils, and there relative positioning with respect to one another. For the purpose of our project, this shows that making the geometries of both coils as similar as possible optimizes the system’s read range if the coils are perfectly aligned. Also, it conveys the inherent reciprocity of mutual inductance because if the order of integration were reversed, such that \( M_{12} \) was calculated instead of \( M_{21} \), the same quantity
would result. Therefore, regardless of the geometries and positions of the two coils, the flux through coil 2 caused by a given current in coil 1 is the same as the flux through coil 1 caused by the same current in coil 2. This forms the basis from which our project function because it allows the wireless transfer of power and telemetry to occur simultaneously and as efficiently as possible. The energizing current in the reader coil induces a current in the transponder coil that is used to power the circuitry, while at the same time, the load modulation changes the current in the transponder coil, which couples back to change the current in the reader coil. [7]

3.1.3 Charge Pumps

Charge pumps are a key component in the energy harvesting module of this project and their function is twofold, they rectify an incident AC voltage into a DC output, and multiply the resulting DC output by an integer value. A four stage Dickinson charge pump is depicted in Figure 8

![Figure 7: 4 Stage Dickenson Charge Pump](image)

As shown in Figure 8 Dickinson charge pumps consist of diodes and capacitors that are arranged so that capacitors can be charged up without discharging, and the voltage across each capacitor adds constructively at the output. Ideally, the capacitors should be as large as possible so that they can store a lot of charge which maximizes efficiency. However there are limitations to this notion, as large capacitors can take up a big footprint and be expensive. Moreover each capacitors breakdown voltage should be significantly large than the supply
voltage (a larger capacitance is generally coupled with a smaller breakdown voltage). When selecting diodes for the charge pump, an engineer must also weight the trade off between a large enough reverse breakdown voltage and a sufficiently fast reverse recovery time. The reverse recovery time of the diodes must be significantly faster than the period of the source to conserve efficiency, and breaking down of the diodes would result in some unwanted capacitor discharge.

An explanation of the inner workings of a dickinson charge pump is accompanied with the diagram shown in Figure 8.

![Figure 8: Charge Pump Functionality](image)

As shown in Figure 8 at the negative cycle of the source D1 is forward biased, allowing C1 to charge up to the amplitude of the source minus the forward voltage of D1 with the polarity suggested by Figure 8. At the next positive cycle of the source, D1 prohibits C1 from discharging, maintaining the potential difference between its plates. Additionally, at the positive cycle of the source D2 is forward biased so that the voltage at each of the positive marked plates are the same. The fact that the other plate of C2 is grounded forces its potential drop to be twice that of the source minus the twice the forward voltage drop across each diode. The polarity of D2 blocks C2 from discharging so that it can maintain its voltage. Dickinson charge pumps with more than two stages extend these principles to facilitate constructive adding of multiple capacitors and embody a similar explanation.

### 3.2 pH Measurement

pH measurement is a unique chemical measurement founding on the difference in voltage between two electrodes [10], as shown in Figure 9.
In Figure 9, the reference electrode and the indicating electrode both have a half cell potential base on an oxidation-reduction half reaction. As the concentration of H+ ions increases, the reference electrode should experience a greater change in potential than the indicating electrode, meaning that different pH solutions will generate a different voltage between the two electrodes. In general, due to the complex nature of differing solutions and probe materials, the voltage-pH curve of an electrode pair is determined empirically via measurement against known buffer solutions [10].

The difficulty in this measurement is that there is very little available current, so the voltage must be measured by a system with a very high input impedance, usually on the order of several MΩ. This typically necessitates an op-amp circuit, since the receiving circuitry also needs to be sensitive enough to discern the small changes in voltage produced. For an Ag/AgCl cell, this comes out to 59.16 mV/pH using the Nernst Equation [11].

3.3 Load Modulation

The DC voltage from the pH probe needs to be converted into a signal that can be transmitted through the inductive link. The method used in this project is known as load modulation; switching an extra capacitor in and out of the circuit. This throws the coils in and out of resonance and produces a corresponding signal at the reader that can be read.
3.3.1 Relaxation Oscillators

Since the change in voltage due to the aforementioned modulation process is constant, the pH reading must be encoded as a square wave whose frequency is related to the pH. In order to achieve this, a special type of oscillator circuit known as a relaxation oscillator will be employed, as shown in Figure 10:

![Figure 10: Relaxation Oscillator [12]](image)

From Figure 10, the capacitor at the top of the circuit charges while the op amp is on until its voltage exceeds that provided by the division of R2 and R3, called the hysteresis resistors, switching the op amp to its minus voltage, and starting the process in reverse [12]. This results in a square wave output, whereas the voltage division between R2 and R3 determines how long the capacitor must charge for until the - voltage becomes more than the +. Figure 11 shows this process in more detail.
In Figure 11 the yellow waveform shows the charging and discharging of the capacitor at the - pin, where the blue waveform is the division of R2 and R3 at the + pin. As shown, the capacitor has to charge or discharge until it matches the division of R2 and R3 in order for the state of the oscillator to toggle. This allows control over both the frequency and duty cycle of the output by changing R2 and R3. A 50 percent duty cycle and a frequency given by Equation 6 are produced in the case that R2 and R3 are chosen to be equal:

$$f_{osc} = \frac{1}{2R_1Cln(3)}$$  \hspace{1cm} (6)

From Equation 6, the frequency varies linearly with the resistor and capacitor values chosen for the top half of the circuit. In order to make the oscillator’s frequency vary with voltage, a second op amp was added to change the voltage between the hysterisis resistors as shown in Figure 12:
The circuit in Figure 12 was developed by Cao et al. [13] in order to provide a voltage to frequency converter at reduced power and footprint for remote power applications. The voltage division between R3 and R4 now has a new reference, which is the output of the second op amp, U2. For example, if a voltage of 1V is at the output of U2 and a voltage of 5V is at the output of U1, R3 and R4 would divide to 3V assuming equal values have been chosen. This means the capacitor C1 only needs to discharge to 3V instead of 2.5V, raising the frequency of the oscillator as well as lowering the duty cycle.

This response is not perfectly linear, however, compared to a traditional VCO, this one is lower power in the sense that it only uses two op-amps. Since power is critical in this battery-less application, the trade off of linearity for lower power consumption was deemed worthwhile.

3.3.2 FSK Modulation

FSK Modulation is a form of telemetry that is commonly used in passive RFID systems to encode information in the backscattered signal. It can be thought of as a combination of conventional Amplitude Modulation (AM) and Frequency Modulation (FM). It parallels amplitude modulation because the desired information is extracted from the peak value of the modulated carrier using an envelope detector, which is found in conventional AM receivers. However, the information is not encoded in waveform of the envelope itself but instead in its frequency, just like in FM [14]. In Passive Radio Frequency IDentification (RFID) systems the excitation signal also acts as the carrier signal, which usually has a much higher frequency than the frequency of the envelope. The graph resulting from a MATLAB simulation used to derive the envelope of an FSK modulated waveform is shown in Figure 13 with a reduced
carrier frequency to improve the visibility of the concept.

In Figure 13 the blue waveform represents the carrier signal whose amplitude increases and decreases as a function of a square wave. This is purely an amplitude modulation and its frequency domain representation would not include a peak at the frequency that represents the information being transmitted. The extracted envelope is the orange signal which, in the frequency domain contains a peak at the desired frequency, despite containing higher order components due to the discharging ripples and spurious peaks. These effects are often reduced by low pass filtering.

### 3.3.3 Envelope Detection

Envelope Detectors are used in a variety of communication electronics in order to demodulate AM signals. As shown in Figure 11 they are simple devices, consisting of a diode biased towards ground leading into a shunt RC network. [16] A schematic of a basic envelope detector is shown in Figure 14.
The time constant of the RC network shown above is designated to have a moderate charge time such that it follows the lower frequency message signal, and loses the higher frequency of the carrier. Specifically, when the capacitor builds up charge during the high portion of the message signal, the resistor is large enough such that it blocks the capacitor from discharging during the relatively short cycles of the carrier signal. During low portions of the message signal, the capacitor will discharge and maintain the value of the reduced amplitude of the carrier. In addition, the diode ensures that only the positive portions of the modulated wave will reach the RC network. It is important to note that a designer can increase the resistance and decrease the capacitance to obtain a desired input impedance while maintaining the same time constant. As a result envelope detectors are robust in their ability to fit a system’s needs.[16]

3.4 Signal Conditioning

The hardware and software applications used to condition the signal so that data could be obtained during processing are explained below.

3.4.1 Sallen-Key Filters

Sallen Key filters are employed in an abundance of Electronic applications for signal conditioning. These devices are categorized as active filters, due to the fact that they require external power to attenuate targeted frequency ranges. The Sallen Key configuration can be utilized as any type of filter (low-pass, high-pass, band-stop, band-pass) and utilizes negative feedback to control gain within the passband, and positive feedback to control filter characteristics near the critical frequency. Moreover, by manipulating the values of elements in the filter, the response can be made to fit Butterworth, Chebyshev and
other filter prototypes. A low pass Sallen key filter schematic is presented in Figure 15.[18]

![Sallen Key Low-Pass Filter](image)

Figure 15: Sallen Key Low-Pass Filter[19]

The functionality of this device is best understood by observing Figure 15 in a unity gain configuration (R3 is open and R4 is shorted). In this configuration, at frequencies far away from the poles C1 sees a virtual ground (because of the large input impedance of the op-amp) and consequently the circuit appears as a 2 pole RC filter where only low frequency components will make it to the positive input of the op amp. The active nature of this configuration becomes apparent near the frequency of the poles, where positive feedback exists through the path of C1. at frequencies approaching the pole the circuit can be designed so that C1 allows power to pass, but appears small compared to C2, and a fraction of the output returns back to the input. This process maintains stability because of the negative feedback branch.[18]

The Magnitude of the Sallen Key Filter’s transfer function is represented by Equation 7. from which design equations 8 , and 9 are derived.[18]

\[
H(s) = \frac{1}{s^2(R_1R_2C_1C_2) + s(R_1C_2 + R_2C_2) + 1} \\
w_p = \frac{1}{\sqrt{R_1R_2C_1C_2}} \\
Q = \frac{\sqrt{R_1R_2C_1C_2}}{(R_1 + R_2)C_2}
\]

These equations allow an engineer to designate the frequency of the poles (assuming it is desired that both poles be the same frequency) and the quality factor by selecting discrete
components. In doing this, a designer should ensure that the resistors are large enough that the device has a significant input impedance, but not too large as to generate too much noise. The value of changing the quality factor is shown by plugging in equations 8 and 9 into equation 7 resulting in Equation 10[18]

\[ H(s) = \frac{w_p^2}{s^2 + s\left(\frac{w_p}{Q}\right) + w_p^2} \quad (10) \]

At the frequency of the pole, equation 10 simplifies to \( Q \), which shows that the gain at the pole can be dictated by the quality factor. It should be noted that at the pole, the output will be -90 degrees out of phase from the input resulting in some distortion in systems concerned with time domain signal integrity. Examples of Bode plots under different quality factors for this device are presented in Figure 16[18]

![Figure 16: Quality Factor Plot][20]

As suggested by the figure above, the quality factor a Sallen Key filter has a high impact on shaping its frequency response.
3.4.2 Nyquist Sampling Theorem

For a bandlimited signal, the Nyquist rate is represented by twice the signal’s bandwidth. When sampling a signal in real time, this is an important concern, because the rate at which the signal is sampled must exceed the Nyquist rate. Otherwise, the signal is considered under-sampled, and aliasing will occur. Aliasing is a type of distortion that causes the digital version of the signal to deviate from the analog signal that it represents. A comparison of a sufficiently sampled and an under-sampled sinusoid is shown in Figure 17. [16]

![Figure 17: Aliasing in Time Domain][21]

From Figure 17 it can be seen that in this basic example, the high frequency sinusoid will be misrepresented as a sinusoid of a much lower frequency. The general implications of this type of distortion can be explained by the frequency domain example shown in Figure 18.

![Figure 18: Aliasing in Frequency Domain][22]

In Figure 18 $f_{max}$ is the bandwidth of the signal and $F/2$ represents half of the sampling rate, known as the Nyquist frequency. The frequency spectrum of a sampled signal is a
symmetric function that repeats at intervals of the sampling frequency. Because half of the sampling rate is the highest frequency that can be detected, any frequency components of the signal that are higher than it fold around the Nyquist frequency and are therefore represented as lower frequency components. To prevent this from happening in this project, a low-pass anti-aliasing filter was implemented. [16]

3.4.3 The Discrete Hartley Transform

In this project, the sampled signal needs to be converted to the frequency domain to extract the pH measurement. Conventionally, the Fast Fourier Transform (FFT) algorithm is used to efficiently perform this conversion. However, in this project, the Fast Hartley Transform (FHT) algorithm was selected because of the limitation of the amount of RAM on the microcontroller that was selected. The FHT algorithm requires less RAM allocation because it is only valid for real inputs while FFT uses complex numbers. The Discrete Hartley Transform (DHT) and the inverse Discrete Hartley Transform (DHT) of a signal are given by Equation 11 and Equation 12 respectively. [23]

\[
H(k\Omega_v) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} h(nT) (\cos(k\Omega_v nT) + \sin(k\Omega_v nT)) \tag{11}
\]

\[
h(nT) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} H(k\Omega_v) (\cos(k\Omega_v nT) + \sin(k\Omega_v nT)) \tag{12}
\]

In Equation 11 and Equation 12 \(h(nT)\) is the n’th sample of the time domain signal, \(H(k\Omega_v)\) is the k’th element of the DHT, \(N\) is the total number of samples taken, \(T\) is the sampling interval which is the inverse of the sampling frequency, \(\Omega_v\) is the frequency domain resolution, which is equivalent to the sampling parameter relationship \(\frac{2\pi}{NT}\).

Observation of the DHT equations conveys the fact that the number of samples in the time domain and frequency domains are equivalent. Because of the symmetric nature of the transformation to the frequency domain, after shifting and scaling the frequency points to the proper values, they are distributed evenly between the negative and positive nyquist frequency. Therefore, the deviation in frequency between each of the DHT points is directly proportional to the sampling frequency and inversely proportional to the number of samples taken. When selecting the number of samples that will be operated on by the DHT, the tradeoff between taking more samples to increase resolution and taking less samples to speed up the acquisition and computation needs to be considered. Also, the sampling frequency should be chosen such that the frequency resolution is sufficient over the expected bandwidth of the signal, while also making sure to avoid aliasing. [23]
3.5 Stomach Properties

Stomachs are difficult to model due to the uniqueness to each person in regards to its size and its shape along with the fullness. We can use an average of several different dimensions in order to give us an average-sized human stomach. Figure 19 shows a diagram of a stomach with important modeling features such as the maximum curvature length, the widest section diameter and the pyloric ring diameter. An average-sized stomach has a maximum curvature length of 30 cm, its widest section is 10 cm, a Pyloric ring diameter of 1.1 cm and the wall of the stomach can vary from 2.35 and 5.43 mm. In addition to having the visual appeal of an actual stomach, the model needs to have certain properties to feel and act realistic. An average stomach has an average volume capacity of 0.94 liters and it can expand to hold up to 4 liters. Along with being able to expand, the model needs to be able to contract similar to a stomach churning food for mechanical digestion. The only electrical properties that are important to our application is that the material cannot be magnetic due to the inductive coupling.

![Figure 19: Stomach Diagram](image)
4 Design and Simulation

To verify the functionality of the project parts prior to construction, several simulation software suites were utilized. For the electrical circuits, Keysight ADS [17] was used for transient and steady-state solutions. For the stomach itself SolidWorks and ASNSY were used for measurements, system analysis and fluid analysis.

For the design of the PCBs, Autodesk Eagle [24] was used, and for the stomach molds SolidWorks was used.

All of the individual electrical components that were simulated were combined to form the schematic shown in Figure 20.

![Figure 20: System Level Simulations][17]

The simulations of each of the components in Figure 20 will be described in the following sections.

4.1 Coupling Network

In this project inductive coupling was used to transfer wireless power from the reader to the transponder, and relay information in the reverse direction. To optimize these processes coupling networks were designed for both the reader and the transponder. An ADS schematic used to simulate the coupling between the reader and transponder networks is shown in Figure 21.
In the reader coupling network, a capacitor was added in series with the coil so that at the resonant frequency of 1Mhz, the two reactive impedances would cancel each other out and allow the maximum amount of current to be drawn from the source and flow through the coil. The resistor was added in series with the reader source to account for its internal resistance. The transponder is matched in the same manner so that the coupled 1Mhz signal excites the maximum amount of current in the coil so that the voltage that drops across it is sufficiently charged to activate the charge pump. A frequency sweep simulation of the schematic in Figure 21 is shown in Figure 22.
It is clear from these results that 1MHz is the resonant frequency of the network, as intended. It also shows that there is a relatively strong signal at the resonant frequency despite the low value that was set as the coupling coefficient, the high value used for the internal resistance of the source and the insertion of the measured resistance values of the coils.

4.2 Transponder Electronics

This section gives an overview of the design and simulation of transponder electronic components including the charge pump, the relaxation oscillator and design of the load modulation switching circuit.

4.2.1 Charge Pump

An eight-stage charge pump schematic was implemented on ADS as per Figure 23.
Also included in this schematic is a voltage regulator in shunt with the output to compress the output under large input voltage. To ensure accurate simulation, Spice models of the small signal diodes and the zener diode used in this project were imported into ADS. A load resistor was integrated to approximate the load that the charge pump is biasing. Different stage orders were tested, and capacitor values were optimized to ensure desired functionality. It was found that an 8 stage charge pump was an ideal compromise between efficiency and output level. In addition, it was decided that 10uF capacitors would render optimum efficiency and not take up too much real estate. Finally due to the fact that the zener diode was regulating at very small currents it was found that the regulated voltage was roughly half of the rated voltage, thus a zener diode rated at 4.5V was elected for the final design.

The result of a transient simulation on the circuit is presented in Figure 24.
Ideally the charge pump would take 4 periods of the source to fully charge, however the simulation in Figure 24 has it taking roughly 42ms. Since the circuit is loaded, a significant fraction of the power that makes it to the final capacitor will discharge to the load and this will require more cycles to compensate. Also the switching of the diodes does not perfectly follow the negative and positive cycle change resulting in further unwanted capacitor discharge. Despite these factors, the voltage reached and maintained itself at the biasing voltage (2.5V) and the transient time observed is not nearly slow enough to compromise timing in the system.[18]

4.2.2 Relaxation Oscillator

To begin the design of the oscillator circuit, a regular relaxation oscillator was first created in Keysight ADS, shown in Figure 25:
Looking at Figure 25, the values of 22nF and 21kΩ result in a frequency of 1kHz using equation 6. This simple design produces the waveform shown in Figure 26:
In Figure 26, the blue waveform shows the capacitor voltage, while the red square wave is the output of the oscillator. As expected, the oscillator outputs a 1 kHz signal, and the capacitor charges and discharges almost linearly during each half cycle. With this basic design working, voltage control was added to the circuit, producing the more functional circuit shown in Figure 27:

![Figure 27: ADS Voltage Controlled Relaxation Oscillator][17]

From Figure 27, the frequency setting resistor at the top of the circuit has been lowered to 10kΩ in order to increase the frequency of the circuit. The second op amp has been added at the bottom of the circuit and the VDC component feeding into it represents the voltage produced by the pH probe. Here, the voltage output of the second oscillator is allowed to draw current across the 10kΩ resistor at its output to keep itself powered. Meanwhile, it
sets the voltage division between the 10kΩ R2 and the 8.2kΩ R3, modifying both the duty cycle and frequency of the final output waveform.

Since this circuit must vary its response with the voltage at the pH probe input, a tuning parameter was used in order to investigate the impact of changing the input voltage. As shown in Figure 28, below, the frequency of the oscillator goes up as pH decreases.

Figure 28: Voltage Controlled Relaxation Oscillator Outputs [17]

Figure 28 shows the square wave output of the oscillator in red in the small window, and its Fourier Transform in blue. The Fourier transform is centered on the fundamental component of the square wave, which is the frequency of the oscillation. Of note, there is a small amount of noise in the spectrum since not every cycle switches at exactly the same frequency. This noise level is far below the peaks, however, so it is mostly irrelevant for picking up the signal.

Starting with the left graph, at a voltage input of 0, which corresponds to a pH of 7, the oscillator has a frequency peak at 4.98 kHz. This is quite a bit different from the relaxation oscillator on its own, which would have a 2 kHz output for the 22nF capacitor and 10kΩ resistor. It is expected that the resting frequency will change with the addition of the new circuitry, so this isn’t an issue. At the right, for a 0.36V input, or pH 1, the oscillator’s frequency has been shifted up to 5.18 kHz, or a 200Hz change, for a slope of 33Hz/pH assuming a linear fit.

4.2.3 Load Modulation Switching

The circuit that was designed to accomplish the load modulation is shown in Figure 29 along with the two coupled resonant networks and sources that were needed for its simulation.
MOSFET 1 in Figure 29 performs the load modulation. The sinusoidal source connected to its gate represents the square wave output of the voltage to frequency converter. This causes this MOSFET to switch on and off at the particular frequency corresponding to the pH reading. This periodic switching on and off causes the transponder matching network to shift in and out of resonance, because the bypass capacitor connected to the source of MOSFET 2 is taken in and out of the circuit. This causes the necessary change in the current in the transponder coil to perform the modulation.

The purpose of MOSFET 2 in Figure 29 is efficiency and optimization of the read range. In the physical implementation of the circuit, the gate of this MOSFET is connected to the middle of a voltage divider between the Transponder’s DC source and ground. In the simulation schematic, the DC source at the gate represents this intermediate voltage. This MOSFET turns on when DC buildup reaches a threshold value determined by the ratio of the resistors in the voltage divider. Therefore, it prevent modulation from occurring until the transponder is sufficiently charged. This is necessary because when the system is modulating, it is only in resonance about half of the time, so it is not charging at maximum efficiency.

Some of the voltage waveforms generated from the schematic in Figure 29 are shown in Figure 30.
The transient voltage plots in Figure 16 show a proof of concept for the mechanism that is used to provide load modulation. The first graph shows the change in the voltage across the transponder voltage superimposed with the sinusoidal source used to represent the relaxation oscillator while the second graph shows the same response experienced by the reader coil. Both the transponder and the reader voltage plots appear thick because they are at 1MHz while the source labeled as the input is at 1kHz. This particular time scaling was chosen to highlight the variation in the amplitude of the high frequency carrier due to the networks shifting in and out of resonance. From this ideal simulation is clear that the voltage waveform is suitable for envelope detection.

4.3 Reader Electronics

The design of specific reader electronic components is provided below including the power amplifier, the envelope detector, and two iterations of the signal conditioning network.
4.3.1 Amplifier

In order to simulate the class E amplifier used in this project, LTspice [25] was used in place of ADS, since the student version of ADS lacks the packages for power MOSFETs. The modelled circuit in LTspice is shown in Figure 31:

![Figure 31: LTspice Model of Class E Amplifier](image)

From Figure 31, a DC voltage of 12V, maintained by a 10uF capacitor was selected as the DC source for the amplifier, being it is a relatively common voltage for batteries and allows more power than a 9V source. A 10mH inductor is used as the choke. For the MOSFET, the IRFP240 was used for simulation since it is a near equivalent to the on-hand IRF640. A voltage controller at the bottom left generates the input 1MHz square wave, simulating the output of the clock oscillator. The tank circuit of L2 and C1 resonates at 1MHz per equation 3. C3 and R1 are used as an extra DC block. While not strictly necessary, this adds assurance against grounding faults that may occur during assembly or measurement, protecting the coil and filter chain. Lastly, the resonant circuit for the transmitting coil is places as the circuit load at the right. Because the tank circuit and the transmitting circuit have no impedance at their resonance of 1MHz, there is no need for a matching network.

The simulated outputs of this circuit are shown in Figure 32:
In Figure 32, the light blue waveform is the current drawn from the source, which rests at a nearly constant 86 mA. The red waveform shows the current at the transmitting coil, a perfect sinusoid with a peak just under 1.5 Amps. The dark blue waveform is the voltage at the top of the resonant circuit vs ground, which is not quite sinusoidal, having a sudden spike at the start of each cycle. While not quite ideal, this isn’t enough of an issue to demand an additional tank circuit. The green waveform plots the instantaneous power available to the resonator, and can therefore be used to compute the amplifier efficiency. For 12V and 86 mA, the DC source supplies 1.032 W of power to the amp, while integrating the green waveform provides an average output power of 620 mW. This gives a simulated efficiency of 60%, which, while not close to the theoretical maximum, isn’t bad considering a real lossy MOSFET model was used and coil resistances were incorporated into the model, meaning the actual efficiency should be close.

Of note, changing R1 impacts the efficiency dramatically, which is expected since the high instantaneous voltages across it will draw substantial power unless it is chosen to be very large. What was less expected, however, was that a lower value of R1 actually increases the power available to the resonator somewhat. One explanation for this is that some of the current flowing into R1 through the above node comes from the coil on its negative half cycle, increasing the current through the coil and meaning more power is present overall. For example, lowering R1 to 1k increases the output power to 695 mW, but drops the efficiency to less than 30%. This is a slight gain in range at a heavy cost of power, however, since the device only needs to be active for short periods of time (about 5s out of every monitoring

Figure 32: LTspice Waveforms for Class E Amplifier
cycle, which could be as long as 10 minutes), the consumed power is not a major concern in regards to battery life. As such, lowering R1 can be beneficial to extend the device’s range in accommodation of larger users.

4.3.2 Envelope Detector

An Envelope detector is necessary in the circuit due to the fact that despite being modulated by the transponder load switching, it does not have a component at the switching frequency. It is necessary to extract the envelope waveform so that digital Fourier analysis can extract the frequency encoding the pH. The envelope detector used was designed and simulated using the parameters shown in Figure 33.

![Figure 33: ADS Envelope Detector [17]](image)

The 30pF capacitor and the 220k ohm resistor where selected such that their discharging time constant would be 6us which is notably longer than the 1us period of the 1MHz carrier signal and significantly faster than the message signal which has a period in the range of 125us to 300us. As a result the voltage at the node above the passive elements will track the 1Khz message signal while disregarding the fast shifts in the carrier signal voltage.

It is important to observe that in the original prototype, lower impedance were employed to realize envelope detection to minimize noise. However this had the negative effect of further loading the excitation circuit, which resulted in a decrease in power efficiency and a disruption of the resonance at the reader. To counter this, the large impedances shown in Figure 33 were used to achieve better results.
Simulation results of the ADS envelope detector are presented in Figure 34.

![Envelope Detector Simulation Result](image)

Figure 34: Envelope Detector Simulation Result [17]

As depicted in Figure 33 variable x was defined to periodically change the amplitude of the input signal to mimic a carrier with an envelope. The results in Figure 33 show that the voltage above the RC network tracks the envelope. It is noted that their is some small charging and discharging in the network at the carrier frequency (1Mhz) however it is observed that this is relatively small and consequently easy to filter out.

4.3.3 Signal Conditioning Chain (Initial Prototype)

The anti-aliasing filter chain was implemented and optimized on ADS. Many different filtering configurations were implemented and tested including passive RC filters, and Sallen Key as well as Multiple Feedback active filters of differing quality factors. In addition, different staging orders had to be tested in order to optimize the design. For instance, if filters are arranged in order of increasing Quality factor noise will be limited because this results in more attenuation at the end of the chain, however, this will push the DC biasing of the initial Op-amp due to the fact that significant amplification will occur near the pole frequency. These factors, as well as other considerations helped to realize the initial prototype shown in Figure 35. [18]
As depicted in Figure 35 the chain starts off with a descending Q order of 1.5 into 1 and then to a passive filter. This is because the incident envelope does not get larger than 1V in amplitude, thus an amplification of 1.5 will not saturate the op-amps, and the system is able to limit noise. A unity gain buffer is also included in this design to ensure that the input of the first stage does not load the reader’s resonant power transfer circuit. Values were chosen based off of design equations shown in section 3.4.1.

AC simulation was performed on this prototype of the filter chain resulting in the bode plot shown in Figure 36.
The frequency plot shown in Figure 36 shows a very steep cutoff as desired to filter out the harmonics of the square wave generated by the relaxation oscillator. Also the results shown amplification of frequencies that are within the output range of the relaxation oscillator (3-7kHz). The non-uniformity within this area is not an issue because the communication signal ideally has only one frequency.

4.3.4 Signal Conditioning Chain (Final Design)

In the final design more amplification was desired to increase the read range in place of the sharp filtering required in this design. This is because a significant noise issue was resolved by fixing a grounding error present in the breadboard which reduced the need to suppress the 1MHz carrier signal. To maintain the same real-estate and increase amplification the signal conditioning network shown in Figure 37 was designed.
Figure 37: Frequency Simulation of Filtering Schematic [17]

This filter design also includes a buffer to isolate it from the circuit. In addition a Sallen Key filter is added with a quality factor of 3 before a single feedback pole inverting amplifier with a gain of 10 with the goal of a very high gain within the message signal frequency range and moderate filtration of the residual carrier frequency. The inverting amplifier is designed such that the feedback impedance (with the addition of the 10pF shunt capacitor) will decrease as a function of frequency resulting in a pole that will significantly lower the gain at high frequencies. A simulation of this network is shown in Figure 37.
The results show that the circuit satisfies design criteria. A gain of up to 30 within the frequency range of the message signal is witnessed, which can be used to realize optimal read range above the noise floor. These results do not show as much noise suppression as shown in Figure 36, however this is not an issue because of grounding corrections and the high gain at the desired frequencies.

4.4 PCB Layouts

For the final version of the project, printed circuit boards were developed using Autodesk Eagle [24]. Each part used in the project had a layout defined, and the set of all parts was assembled into a parts library. On completion, Gerber files were generated for each board, and sent for manufacture at OSHPark in Oregon.

4.4.1 Transponder Layout

The primary concern for the transponder was size. Being that the project’s goal is to have the chip implanted endoscopically, the board was made double sided to cut its imprint by a factor of two. The full layout for the transponder board is shown in Figure 39.
Figure 39 shows the entire board with all its traces. Its overall footprint is 2.54 cm by 5.10 cm, meaning it is small enough to fit in the stomach, but not to be implanted endoscopically. Due to funding limitations, machine soldered parts were unavailable, and as such this board is as small as can be reasonably hand soldered, using a minimum package of SOD-323-2, which is only 1.75mm by 1.25mm. Figure 40 shows the traces on the front face.

In Figure 40, red traces show copper lines, while white writing is a layer of silkscreen, and
green pads are metallization on both sides of the board. The coil is placed in the two vias at the bottom right of the board with a tuning capacitor placed between them. From there, the charge pump with diodes and capacitors wraps around the central op amp. The pH probe is placed in the vias at the left, and the ground track runs along the bottom. Figure 41 shows the traces on the rear face of the PCB.

![Figure 41: Transponder Bottom Traces [24]](image)

From Figure 41 the back side of the board has blue traces marking the routing and surface mount pads for the resistors and capacitors that make up the oscillator circuit. This interfaces heavily with the central 8-pin op-amp. It also has the two MOSFETS and capacitor that allow for load modulation. Of note, the text is reversed since the board is a mirror image of the top face rather than a real view of the bottom. This allows for each point on the bottom board to correspond to the same place on the top. For example, the 8 pins of the op-amp fall on the same coordinates in both diagrams.

### 4.4.2 Reader Layout

For the layout of the reader board, space was much less of a concern, so a single-sided board using through-hole components was developed, shown in Figure 42:
Figure 42: Reader Board Layout [24]

Figure 42 has a large section of pin sockets for an Arduino ProMini Microcontroller. From there, the functional circuit moves counterclockwise to the class E power amplifier and the coil, placed between the two empty vias. Continuing to the to left, the filter chain moves through the op-amps, and the final signal comes pack to digital pin 12 on the Arduino. Because the board is only single-sided, each connection is clearly visible on the provided diagram, with blue traces being on the bottom side of the board, red traces on the top, and green pads on both sides.

All the used packaging types are standard except for the LFSPXO069854 clock oscillator. For this chip, a 14 pin Dual-in-Line (DIL) standard was adapted by removing the inner 10 pins, leaving only the corners.
The overall board measures 3in by 4in, making it an easy fit for a fanny pack or similar holder. Two extra LEDs have also been included for purposes of testing, and potential application in the alignment process. The largest vertical part is the 10 mH inductor, which protrudes 2cm from the board. As such, it is relatively flat, and not too troublesome to carry.

4.5 MATLAB GUI

The GUI that allows a user to control the system and view measurements is shown in Figure 43.

![Figure 43: MATLAB Generated Graphical Display and Interface](image)

The GUI allows a user to easily connect to the COM port that the reader is associated with. Once connection with the reader is established, two types of measurement are possible. Both of these types of measurements incorporate the user-entered text below the label "Number of Pts. Averaged" referring to each individual data point that is displayed to the user.
The first type of measurement is a single measurement that updates the "pH Level" text box in the upper right corner of the screen. This is activated when the user presses the push button labeled "Read pH". Once this button is pressed, MATLAB code is executed that commands the Arduino to turn on the clock oscillator that enables the power amplifier to allow excitation of the implant. Then a delay is necessary to allow the system to undergo a transient response before the Arduino is commanded to sample the ADC and return the signal strength and frequency corresponding to the pH reading. Once MATLAB receives the readings, the Arduino is commanded to take the same measurements until the desired number of data points for averaging are collected and averaged together. Once this is complete the clock oscillator is turned back off to conserve power.

The second method of measurement continuously updates the the plot in the center of the window to show how the pH measurement changes with time. This periodic measurement is activated by the toggle button labeled "Monitor pH" and allows the user to enter the period over which the pH will be monitored and how often a measurement is taken. The process of obtaining each data point is similar to that used in the aforementioned measurement method, but includes a mechanism that will turn the clock oscillator off between measurements if there is sufficient time between each measurement being in order to conserve the battery life. Otherwise, the clock oscillator is kept on throughout monitoring so that transient responses do not distort rapid monitoring.

To adjust the alignment of the reader coil with the transponder coil, the user can press the toggle button labeled "Align Reader" to have the reader constantly perform the necessary excitation to monitor the strength of the peak frequency of the received signal. The code used to generate this feature is shown in Figure 44, which is executed upon pressing the toggle button.
Figure 44: MATLAB Code for Alignment Feedback

The code in Figure 44 also commands the Arduino to measure the peak frequency and the signal strength around the peak. However it is only concerned with the signal strength. Conditional branching ensues based on this value to set the text in a text box that appears above the toggle button to indicate that status of the alignment. This text only has three different states, however the background color of the box was made to change continuously based on the signal strength by adjustment of the RGB coordinates. When the scaled signal strength is less than 10 the red coordinate is maintained as on while the green coordinate set to a scaled value that is propositional to the signal strength so that zero signal strength results in red while ten signal strength results in yellow, with shades of orange in between. For intermediate readings between 10 and 20 the color is shifted from yellow and green in a similar manner. Whenever an actual reading is taken, this text box is also updated in the
same way to provide more feedback on the alignment of the coils.

Additionally the GUI allows the user to save the data collected during pH monitoring as a .mat file, export it as a CSV file for use in Microsoft Excel, and open previously saved plots for viewing. It also has a working message box that communicates errors, successful connections and updates on file reading and writing. The full script for the generation of the GUI can be found in Appendix 2.

4.6 Microcontroller Program

The main loop of the Arduino code is shown in Figure 45.

```c
void loop()
{
  struct s:
  // Remain until it receives command from GUI
  if (Serial.available() > 0)
  {
    // Take reading from MATLAB GUI
    int f = Serial.parseInt();
    // Read ADC
    if (f == 1)
    {
      // Get ADC reading
      sampleADC();
      // Calculate peak of EMI result
      s = computeEMITrace();
      // Print the result back to the GUI
      Serial.println(s.peakFreq);
      delay(100);
      Serial.println(s.peakEnergy/100);
    }
    // Turn on Clock
    if (f == 2)
    {
      digitalWrite(VCO_PIN, HIGH);
    }
    // Turn off Clock
    if (f == 3)
    {
      digitalWrite(VCO_PIN, LOW);
    }
  }
}
```

Figure 45: Main Loop of Arduino Program

The loop in Figure 45 idles until it receives a command from the MATLAB GUI. Although the MATLAB GUI has many different capabilities, these features were able to be simplified to combinations of the three different commands present in the Arduino Code. The first
command completes 256 evenly spaced samples with the ADC, performs the Fast Hartley Transform (FHT) of these samples, interpolates to find the peak spectral frequency that represents the pH reading and also outputs the spectral energy density around the peak frequency to communicate the strength of the signal so coil alignment can be evaluated. The second command turns on the Clock Oscillator that excites the power amplifier while the third command turns it off. These last two commands were added to allow flexibility in the adjustment of the system’s power consumption. The function that contains the sampling mechanism is shown in Figure 46.

```c
void sampleADC()
{
    for (int i = 0; i < FHT_N; i++)
    {
        // save 256 samples
        while(!((ADCSRA & 0x10))); // wait for adc to be ready
        ADCSRA = 0x70; // restart adc
        byte m = ADCL; // fetch adc data
        byte j = ADCS;
        int k = (5 << 8) | m; // form into an int
        k = (k & 0x00FFFF) / k; // form into a signed int
        k <<= 6; // form into a 16 signed int
        FHT_input[i] = k; // put real data into bins
        delayMicroseconds(12); // Reduce Sampling Rate (delayMicroseconds only has a resolution of 4us)
    }
    return;
}
```

Figure 46: ADC Sampling Mechanism Function Definition

This user generated function is used in favor of the analogread function that is build in to the arduino IDE in order to speed up sampling to prevent spectral aliasing. The analogread function allots a longer settling time for the sample and hold mechanism before conversion for optimized accuracy. However for the purpose of this application the ADC was set to free running mode and manually restarted right before the data registers were accessed. A delay was added at the end of each iteration to decrease the sampling rate for higher bin resolution in the frequency domain. The sampling rate was measured through a software adjustment as 38.2kHz. The next function that is called, which finds the interpolated peak of the FHT of the samples and the spectral energy of in the frequency band of interest, is shown in Figure 47.
Figure 47: Compute FHT Peak Function Definition

First, the function calls the functions associated with FHT that are built into the Arduino IDE. The functions work with global variables and perform all of the necessary steps to perform zero padding and the adjustment order of the arrays to ultimately lead to a linear representation of the bins in the frequency domain. The subsequent loop finds the peak frequency by sweeping through the array starting at the index that contains the bin at the frequency corresponding to the maximum possible pH reading, and stopping at the minimum pH reading. Next, the peak is linearly interpolated with its surrounding values. The result of the interpolation, and the normalization factor, which represented the band energy near the peak were both used to update the data members of the structure that was returned after execution. The full Arduino Code can be found in Appendix 3.

4.7 Stomach Modeling

The following sections describe how Solidworks was used to design the model stomach, and perform a fluid simulation.

4.7.1 Stomach Design

The design of the stomach needs to have similar dimensions to the ones as stated in section 3.5. The process used for making the stomach is soluble core injection molding which uses a core and an external mold to from the stomach model. The design of the core of the stomach was modeled in Solidworks and can be seen in Figure 48. This model has an
extruded part at the base of the stomach for the pour spout when making a wax cast. In addition, there are two extruded parts at the inlet and outlet that were made for the covers to tightly fit over.

Figure 48: SolidWorks Stomach Core[1]

The next part to model was the covers for the mold. These can be seen in Figures 49 and 50. In Figure 49 the top cover has two pipes extruded near the inlet and outlet of the stomach. Initially this was designed to be the inlet for the silicone injection, but after complications with the first trial of injection, this section was switched to be the air hole for the silicone instead. This was due to cured silicone getting stuck when cleaning. In Figure 50 there is an extruded pipe that was initially designed to be the air hole. This was changed due to the injection hole failing in addition to more space being needed for silicone to settle.
Figure 49: SolidWorks Stomach Cover Top[1]

Figure 50: SolidWorks Stomach Cover Bottom[1]
4.7.2 Fluid Simulation

In addition to modeling the stomach, a simple fluid analysis of the stomach was created as shown in Figure 51. This was used to simply show the fluid movement through the stomach with a constant input velocity. The fluid flows fastest at the input and output due to the diameter being smallest at those sections and the slowest fluid flow takes place at the widest section of the stomach.

Figure 51: SolidWorks Simulation[1]
5 Implementation

In this portion of the report, physical implementation of the design is thoroughly covered.

5.1 Coil Manufacturing

The first step in the design of each coil was to use the measured value of the capacitor in the respective resonant network to calculate the inductance at which the resonant frequency was 1MHz. Once this was found the number of turns that lead to the desired inductance value was calculated given the cross-sectional area of each coil, and parameters of the specific magnet wire that was used. During each iteration, starting when two less than the calculated number of turns were completed, an LCR meter was used to measure the inductance after each loop, until the desired inductance value was reached. Once this point was reached, the wire was cut, and the epoxy was applied.

Creation of the reader coil was rather straightforward because the size constraints were somewhat relaxed. To ease manufacturing and make the alignment between the reader and transponder less susceptible to horizontal skew this coil was wound in a large circle that fit inside the footprint of the reader PCB. The finished reader coil is shown in Figure 52.

Figure 52: Manufactured Reader Coil
The transponder coil was designed to fit around the rectangular PCB of the implant. This was seen as the best way to minimize the overall footprint of the implant at the desired read range because the cross sectional area of the coil would be as large as possible in order to minimize the amount of turns needed for the proper inductance. To facilitate winding the inductor to the proper geometry, a rectangular stand that was slightly larger than the planar footprint of the PCB was designed in Solidworks and 3-D printed. This stand is shown in Figure 53.

![Figure 53: 3-D Printed Coil Winding Stand](image)

The widened base of the stand allows the bottom loop of the coil to be constrained while pressure is put on the top of the coil to minimize the spacing between loops. The sharp corners of the stand were maintained so that the coil would fit around the actual corners of the PCB. Additionally, the top of the stand in Figure 53 is hollow to conserve printing filament.

An image of the first iteration of the transponder coil manufacturing is shown in Figure 54.
The first attempt at inductor manufacturing involved stacking loops directly on top of one another to minimize the footprint in the plane of the cross section and simplify the application of epoxy to one layer. Once twelve loops were completed the desired inductance was measured and epoxy was applied to each of the four edges of the coil while pressure was maintained. This step resulted in a coil that functioned properly but extended that vertical footprint of the implant to an undesirable amount. The second winding attempt is shown in Figure 55.
The second attempt at manufacturing the inductor involved two layers along the radial direction of the cross-section. To begin, six loops, which represented half of the number of turns of the first iteration, were stacked directly upon one another and glued together in the same manner as the first iteration. Once the epoxy dried a second layer of another six loops was generated outside of the first layer starting from the top and travelling towards the base of the stand, until the end of the coil overlapped the beginning of the coil along one of the shorter edges of the stand. After the second layer of epoxy that was applied dried, the two overlapping leads were trimmed to the appropriate lengths so they could be soldered to the corresponding pins in the PCB. This attempt represented the appropriate trade off between the implant’s footprint and read range, so it was used in the final transponder iteration.

5.2 Breadboard Prototyping

The first physical implementations of the electronics in this project were implemented on breadboards. This streamlined the initial testing and allowed for easy changes to be made. At an operational frequency of 1MHz it is noted that the stray capacitances between
breadboard rows would have some effect, albeit small, on the performance the breadboarded prototype. Care was taken to ensure that this effect was mitigated by spacing out excited rows (within reasonable distances). Due to the complexity of the circuits and the presence of significant 1MHz back scatter from the coil coupling it was also vital to ensure that no ground loops were present to avoid magnetically induced voltages and currents between different segments of the ground reference. This was carried out by setting aside one of the long breadboard buses as a central ground row which all other ground connections were tied to directly. Despite these difficulties, breadboard testing facilitated easy error fixes, that would have been costly to fix after PCB implementation.

The transponder breadboard implementation is shown in Figure 56.

![Figure 56: Breadboard Implementation of Transponder](image)

During breadboard system tests, the transponder antenna was connected to the red and black wires shown in Figure 56. Using this setup, the optimum tradeoff between efficiency, and output voltage of the charge pump was experimentally found by tweaking the energy harvesting portion at the top of the image. Additionally, the resonant circuit had to be retuned in the breadboard because of the loading of additional components added. Finally, experiments were run with this setup in order to find an optimum value of the load modulation capacitor so that modulation had a dramatic effect on the voltage at the reader coil.

The breadboarded prototype of the Reader Excitation Circuity is shown in Figure 57.
In the circuit above, the reader antenna was attached to the left-handed red and black wires during testing. All of the grounding in the reader breadboard circuity was directly routed to the horizontal ground row shown in Figure 47 to avoid grounding error. Initially a class D amplifier was implemented on the breadboard for amplification of the power signal. During experimentation however, it was found that when the class D amplifier abruptly pulled on the DC supply for short intervals, the 1MHz component was also noticed in the grounding the circuit. To combat this the more stable class E amplifier was employed in the final design. Additionally, a large inductor was placed between DC power and ground to further stop the noise.

The breadboard created for the signal conditioning network of the reader is shown in Figure 58.
The signal conditioning network was frequently modified due to changes in the noise level of the circuit. Multiple DC blocks had to be added to the circuit because of the very high DC voltage level witnessed from the noisy envelope of the large voltage present at the reader inductor coils. These capacitors had to be loaded with large resistors so that they discharged when the system was turned off between readings. Furthermore, multiple voltage buffers had to be added as shown because to avoid significant loading changes on the resonant circuit. Finally, amplification and signal filtering was optimized on the breadboard to ensure maximum read range.

Figure 59 shows the Circuitry used to feed the Arduino Uno the demodulated signal for processing.
As is shown in the circuit above, two voltage regulators were added to ensure that the Arduino would not get a higher voltage than it could handle. Also, the DC voltage supply of 12V is regulated down to 5V in order to adequately power the Arduino.

5.3 Electronics Manufacturing

Below, the final packaging of the system’s electronic components is described.

5.3.1 Transponder Packaging

Figure 60 shows the transponder PCB after surface mount components were soldered on, and the rectangular coil was glued to the outside.
After the functionality of the manufactured transponder was verified, the leads of the pH probe were soldered unto the board, and it was coated in silicone. The final result is shown in Figure 61.

To coat the transponder in silicone, a cardboard box was constructed with the desired dimensions of the coated transponder. Then unused integrated circuits were glued to the bottom of the PCB to act as the mechanism by which the implant could be attached to the wall of the model stomach. The transponder was then placed inside the box with the leads of the added integrated circuits sticking into the bottom layer of the box, so that they would not be coated. Then, silicone was poured into the box while it was ensured that the pH
probe was sticking out. Finally, after the silicone hardened, the box was removed from the coated.

5.3.2 Reader Packaging

The reader PCB, shown in Figure 62, below, was easy to assemble, being that all the parts were through-hole. Some of the parts rise out of the board up to about 1”, however, it otherwise sits relatively flat.

![Figure 62: Final Reader PCB](image)

The only real stipulation with the reader board is that whatever comes into contact with it must be an insulator as to not short the exposed pins. As such, the reader, battery, and coil were packed into a clear fanny pack for demonstration purposes, as shown in Figure 63, on the following page.
Figure 64 shows the fanny pack being worn by a male model, who suffers from GERD. The fanny pack fits snugly around the torso, and keeps the coil fixed at a constant distance from the stomach. The clear fanny pack is not only a fashion statement, but also allows the device to travel through airport security without issue. The electronics are clearly visible, making it ideal for demonstration purposes.
5.4 Stomach Manufacturing

An important step in creating a model stomach is determining the material to use. Silicone Sorta-Clear 40 was chosen as it has a similar hardness and flexibility to an actual stomach along with not being magnetic which is important for our electrical purposes. After the material was chosen, we then needed to model the stomach and create a manufacturing process. The model of the stomach was created in SolidWorks along with a core and outside covers, which were shown in section 4.7.1. The core was 3D printed and can be seen in Figure 65.

![3D Printed Core](image)

Figure 65: 3D Printed Core

This core was then used to create and silicone mold as shown in Figure 66. This mold was made by hot gluing the core to the bottom of a foam core board. Then additional pieces of foam core board were arranged and hot glued in a way to minimize the amount of silicone needed for the mold while still having enough strength for several casts. After the silicone is cured the foam core board is pulled apart from the silicone as silicone does not stick to other material.
After the mold was cut, it can be setup for a wax cast as shown in Figure 67. The wax used for casting was Paraffin wax as it cheap and accessible. The silicone mold was heated to 100°F while the wax was melted at 140°F. This was done so that the wax cools at a slower rate and can cool evenly. Due to the wax settling when cooling down, the wax would need to be poured in several times to keep topping it off.
The wax is then taken out of the mold after it is all cooled down as shown in Figure 68. The wax is not ready to be used yet as the pour spout needs to be fixed. After post-processing the wax core by shaving down the outlet end and the pour spot, the wax is then ready to be used in the injection molding.

![Figure 68: Wax Core](image)

In addition to 3D printing the core, the mold covers were 3D printed as shown in Figures 69 and 70. These covers were used as a two-part mold for the silicone injection and hold the wax core in place. They are placed together with the wax core in place and clamped together as shown in Figure 71.
Figure 69: 3D Printed Cover

Figure 70: 3D Printed Cover
After the silicone stomach was cured, the wax core then needed to be removed. This was done by putting the stomachs in hot water so that the silicone will stay at the bottom and the wax will melt and rise to the top. After the wax was completely removed the silicone stomach if hollow and complete as shown in Figure 72.
5.5 Gastric Movements

In order to create gastric movements in the stomach, it needs to have forces around it acting like muscles that squeeze it together. Figure 73 shows the mechanism used to create realistic stomach motion.
The linear actuator, which is connected to a timer switch relay, produces a linear back and forth motion which enables a pulling action. The end of the actuator is attached to a ziptie, which was modified to move freely, that can squeeze the stomach in on itself. The stomach needs to have more than one contraction, so a system was setup so that it has a second linear motion moving in the opposite direction of the actuator.
6 Measurement Results and Discussion

This section presents the results obtained from experimentation on specific components of the system as well as the results of testing performed on the final product.

6.1 Charge Pump Evaluation

In order to optimize the charge pump, the load voltage outputted when loaded with a 100k ohm resistor is plotted versus separation distance for both a four stage and a six stage charge pump using the breadboard reader design. This is presented in Figure 74.

![Figure 74: Charge Pump Read Range Comparison](image)

The results show that despite having a lower initial output voltage, the 6 stage charge pump maintained a value above the threshold for a greater separation distance than the 4 stage charge pump. Specifically, the loading initially affected the 6 stage pump more, exposing its inefficiencies. Despite this, as the amplitude of the input voltage lowered due to attenuation with increased coil distance, the six stage charge pump’s additional DC multi-
plication was necessary to maintain an output level above the threshold between 1.5 and 2 inches. As a result of this, the six stage charge pump was elected for the final design.

6.2 Coil Evaluation and System Read Range

Near field radiation patterns of the transponder and reader coils were measured using an MDL technologies EMSCAN EHX+ near field scanner. The machine captured the data with a planar array of H-field probes, and interpolated values in between probes in order to realize the patterns of the reader coil presented in Figures 75 and 76.

![Figure 75: Reader Coil H-Field Pattern at 0 Mils](image-url)
Additionally, the transponder near field radiation patterns that were measured are mapped in Figure 77 and Figure 78.
The patterns presented by these results are not ideal in terms of uniformity, however they exhibit acceptable directivity in the desired radiation direction. It is important to note that each pattern is normalized with its own respective maximum value, and as a result patterns at 100 mils appear stronger than the radiation patterns measured at 10 mils. It is hypothesized that the lack of cylindrical symmetry present in the result is a consequence of inconsistencies present when the coils were hand wrapped.

Following individual coil characterization, the mutual inductance of the near field communication was tested for the system as a whole with respect to skew and coil separation. These parameters are visualized in Figure 79.
In Figure 79 displacement along the x dimension represents coil skew, and displacement along the z dimension was used to test coil separation. Using an X-Y table, the mutual coupling was tested with variation in the aforementioned variables to produce the display shown in Figure 80.

<table>
<thead>
<tr>
<th>Separation [in]</th>
<th>0.5</th>
<th>1</th>
<th>1.5</th>
<th>2</th>
<th>2.5</th>
<th>3</th>
<th>3.5</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>12.8</td>
<td>12.7</td>
<td>9.6</td>
<td>7.6</td>
<td>8.3</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>2.5</td>
<td>20.2</td>
<td>20.7</td>
<td>11.0</td>
<td>14.9</td>
<td>17.7</td>
<td>11.0</td>
<td>8.3</td>
<td>N/A</td>
</tr>
<tr>
<td>2</td>
<td>32.8</td>
<td>32.0</td>
<td>31.5</td>
<td>28.4</td>
<td>21.2</td>
<td>10.7</td>
<td>7.4</td>
<td>N/A</td>
</tr>
<tr>
<td>1.5</td>
<td>33.6</td>
<td>33.5</td>
<td>28.4</td>
<td>30.7</td>
<td>28.3</td>
<td>31.9</td>
<td>12.2</td>
<td>8.2</td>
</tr>
<tr>
<td>1</td>
<td>34.4</td>
<td>32.5</td>
<td>30.8</td>
<td>30.1</td>
<td>32.2</td>
<td>31.5</td>
<td>19.9</td>
<td>11.1</td>
</tr>
<tr>
<td>0.5</td>
<td>31.9</td>
<td>30.6</td>
<td>29.7</td>
<td>30.8</td>
<td>30.9</td>
<td>30.3</td>
<td>26.3</td>
<td>14.2</td>
</tr>
<tr>
<td>0</td>
<td>32.6</td>
<td>30.6</td>
<td>32.3</td>
<td>32.1</td>
<td>32.3</td>
<td>32.2</td>
<td>22.2</td>
<td>13.1</td>
</tr>
<tr>
<td>-0.5</td>
<td>32.6</td>
<td>30.8</td>
<td>29.9</td>
<td>31.4</td>
<td>31.3</td>
<td>30.4</td>
<td>25.0</td>
<td>14.0</td>
</tr>
<tr>
<td>-1</td>
<td>35.2</td>
<td>31.4</td>
<td>31.5</td>
<td>30.5</td>
<td>33.4</td>
<td>31.7</td>
<td>20.6</td>
<td>12.1</td>
</tr>
<tr>
<td>-1.5</td>
<td>33.8</td>
<td>31.7</td>
<td>29.0</td>
<td>30.9</td>
<td>28.6</td>
<td>32.4</td>
<td>12.4</td>
<td>8.4</td>
</tr>
<tr>
<td>-2</td>
<td>32.3</td>
<td>28.9</td>
<td>31.5</td>
<td>31.2</td>
<td>28.3</td>
<td>29.5</td>
<td>10.1</td>
<td>7.1</td>
</tr>
<tr>
<td>-2.5</td>
<td>29.2</td>
<td>28.7</td>
<td>31.0</td>
<td>14.9</td>
<td>17.6</td>
<td>11.0</td>
<td>8.2</td>
<td>N/A</td>
</tr>
<tr>
<td>-3</td>
<td>12.6</td>
<td>12.7</td>
<td>9.6</td>
<td>9.4</td>
<td>9.0</td>
<td>8.3</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Figure 80: Two Dimensional Coil Range
In Figure 80, the values shown represent the size of the peak frequency within the frequency range of interest after the discrete Hartley transform. This effectively characterizes the read range, because the level of the message signal over the noise floor directly depends on the impact that the transponder has on the reader coil during mutual coupling. Moreover, in Figure 80 the values are given a color scale so that the effective reading area can be visualized.

6.3 System Measurements

Once the entire system was complete and optimized its accuracy was first evaluated by measuring the pH of common beverages. The results of this test are shown in Table 5.

<table>
<thead>
<tr>
<th>Solution</th>
<th>A.D.A. pH [26]</th>
<th>System pH</th>
<th>Percent Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>pH 4 Buffer</td>
<td>4.00</td>
<td>4.00</td>
<td>0.12</td>
</tr>
<tr>
<td>Pepsi Original</td>
<td>2.40</td>
<td>2.46</td>
<td>2.57</td>
</tr>
<tr>
<td>Vitamin Water</td>
<td>3.05</td>
<td>2.88</td>
<td>-5.75</td>
</tr>
<tr>
<td>Grape Juice</td>
<td>3.30</td>
<td>3.22</td>
<td>-2.58</td>
</tr>
<tr>
<td>Grapefruit Juice</td>
<td>3.30</td>
<td>3.22</td>
<td>-2.58</td>
</tr>
<tr>
<td>Lemon Juice</td>
<td>2.25</td>
<td>2.46</td>
<td>8.69</td>
</tr>
</tbody>
</table>

Next, the monitoring feature of the MATLAB GUI was employed to obtain plots that compared the effectiveness of different antacids. Plots of pH versus time were obtained by monitoring the pH of a lemon juice solution after two crushed tablets of each of the test subjects were added, until the antacid was fully dissolved. The plots that resulted from a Generic Antacid, TUMs and Alka Seltzer Tablets are shown in Figure 81.
The plots in Figure 81 clearly show that the Alka seltzer tablets worked faster and raised the pH more when compared with the equivalent amount of the two other antacids as expected. It is interesting that upon comparison of the generic antacid and TUMS the former raised the pH to a higher value, while the rise of pH in the presence of the latter occurred slightly faster, indicating no clear superiority.
7 Conclusion and Future Work

Working on this project resulted in many new challenges and the accumulation of a large amount of practical experience. In particular, filtering the signal present at the reader coil required a substantial amount of design work and trouble shooting, from balancing the Sallen-Key active filter and properly isolating the loading impacts of each stage, to using the Hartley transform in order to fit the necessary code in the limited RAM provided by the microprocessor. Iterating the design into full-fledged printed circuit boards was a practical learning experience, from familiarity with the design software to surface mount soldering techniques. The development of the stomach required significant amounts of research into and experimentation with the injection molding process, as well as the properties and uses of silicone.

In its completed state, the project successfully demonstrates wireless pH readings at a range of up to 4 inches with a minimum error of 0.2pH, with a small implant footprint. The experimental data generated provides a useful comparison of beverage pH and antacid effectiveness. However, due to budget and time constraints, more development will be needed for full realization and bio-compatibility.

That development will involve several stages of future work. The primary focus will be to use micromachining techniques to manufacture a planar pH probe, and reduce the size of the transponder to be safe for implantation. Likewise, a paralyne coating can be applied to the outer layers as the last micromachining step, eliminating the need for the silicone coat on the transponder. The read range can be improved by increasing the available power, as well as using a coil geometry without a central null. As for the stomach, the accuracy of the model can be increased by adding inlets for peptic acid and other digestive enzymes to emulate the acid production that goes on in the stomach.
References


8 Appendix

8.1 Appendix 1: Full Gantt Chart

Figure 82: Full Gantt Chart Sr. Design 1
8.2 Appendix 3: MATLAB GUI Program

```matlab
function varargout = SeniorDesignControl1(varargin)

%GUI Control Program for the Senior Design project "Passively Powered pH
%Sensor for the Study of Gastric Disorders".
%Author: Kevin Hart
%Version: 1.1
%Date Modified: 2/15/19

%H = SENIORDESIGNCONTROL1 returns the handle to a new SENIORDESIGNCONTROL1
%or the handle to
%H = SENIORDESIGNCONTROL1('CALLBACK',hObject,eventData,handles,...) calls the
%local
%H = SENIORDESIGNCONTROL1('Property','Value',...) creates a new
SENIORDESIGNCONTROL1 or raises the
existing singleton*. Starting from the left, property Value pairs are
applied to the GUI before SeniorDesignControl1_OpeningFcn gets called. An
unrecognized property name or invalid Value makes property application
stop. All inputs are passed to SeniorDesignControl1_OpeningFcn via
varargin.

*See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
instance to run (singleton)".

See also: GUIDE, GUIDATA, GUIMANAGES

% Edit the above text to modify the response to help SeniorDesignControl1

% Last Modified by GUIDE v2.5 25-Mar-2019 16:25:42
```
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name', mfilename, ...
    'gui_Singleton', gui_Singleton, ...
    'gui_OpeningFcn', @SeniorDesignControl1_OpeningFcn, ...
    'gui_OutputFcn', @SeniorDesignControl1_OutputFcn, ...
    'gui_LayoutFcn', [], ...
    'gui_Callback', []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before SeniorDesignControl1 is made visible.
function SeniorDesignControl1_OpeningFcn(hObject, eventdata, handles, varargin)
    disp('Opening Function')
% Choose default command line output
handles.output = hObject;

% Query system for Serial Connections and save list returned
serialports = instrhwinfo('serial');
ports=serialports.AvailableSerialPorts;

% Populate GUI Pop-Up Menu with Available Serial Connections
portNames=ports;
set(handles.portMenu,'String',portNames);

% Initialize axes in GUI
axes(handles.axes1);
xlim([0 100]);
ylim([3 10]);
title('pH vs. Time');
xlabel('Time [sec]');
ylabel('pH');
grid on;

% Update handles structure
guidata(hObject, handles);

% --- Outputs from this function are returned to the command line.
function varargout = SeniorDesignControl1_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on selection change in portMenu.
function portMenu_Callback(hObject, eventdata, handles)
% hObject handle to portMenu (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns portMenu contents as
% cell array
% contents{get(hObject,'Value')} returns selected item from portMenu

% --- Executes during object creation, after setting all properties.
function portMenu_CreateFcn(hObject, eventdata, handles)
% hObject handle to portMenu (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
    get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end

%% --- Executes on button press in connectButton.
function connectButton_Callback(hObject, eventdata, handles)
    % hObject handle to btnConnect (see GCBO)
    % eventdata reserved - to be defined in a future version of MATLAB
    % handles structure with handles and user data (see GUIDATA)
    disp('Connect')
    % Get selected COM port from pop-up menu
    selection = get(handles.portMenu,'Value');
    portNames = get(handles.portMenu,'String');

    % Get serial port object.
    port=portNames(selection);
    port=char(port);
    global obj1 %global variable to store the serial object referencing the
               %connected Arduino
    % Instantiate serial object
    obj1 = instrfind('Type', 'serial', 'Port', port, 'Tag', '');

    % Create the serial port object if it does not exist
    % otherwise use the object that was found.
    if isempty(obj1)
        obj1 = serial(port);
    else
        fclose(obj1);
        obj1 = obj1(1);
    end
    % Set baudrate of serial connection to Arduino
    set(obj1,'BaudRate',9600);

    % Connect to instrument object, obj1.
    fopen(obj1)
    set(handles.connectButton,'String','Connected','enable','off')
    % Disable inputs for proper setup time
set(handles.readButton,'Enable','off')
set(handles.monitorButton,'Enable','off')
set(handles.disconnectButton,'Enable','off')
pause(2);
set(handles.readButton,'Enable','on')
set(handles.monitorButton,'Enable','on')
set(handles.disconnectButton,'Enable','on')

% --- Executes on button press in disconnectButton.
function disconnectButton_Callback(hObject, eventdata, handles)
    disp('Reached line to enable connect btn'); %%%% not reaching this line? bug
    hObject handle to btnDisconnect (see GCBO)
    eventdata reserved - to be defined in a future version of MATLAB
    handles structure with handles and user data (see GUIDATA)
    global obj1
    fclose(obj1);
    set(handles.connectButton,'String','Connect','enable','on')

% --- Executes on button press in resetButton.
function resetButton_Callback(hObject, eventdata, handles)
    global t;
    global pH;
    t = [];
    pH = [];
    hObject handle to resetButton (see GCBO)
    eventdata reserved - to be defined in a future version of MATLAB
    handles structure with handles and user data (see GUIDATA)

% --- Executes on button press in readButton.
function readButton_Callback(hObject, eventdata, handles)
    global obj1;
    ADCcmd = '1';
    ClockOnCmd = '2';
    ClockOffCmd = '3';
set(handles.errorBox,'String','Message Box')

% hObject handle to readButton (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
numAverages = str2num(get(handles.editAverage,'String'));

pHReading = [];

% Turn on VCO and give it time to charge up the system
fprintf(obj1,'%c',ClockOnCmd);
pause(2);

for i = 1:numAverages
    totSignal = 0;
    %Tell Arduino to execute one reading
    flushinput(obj1);
    fprintf(obj1,'%c',ADCcmd);
    %Get Reading from Arduino
    freq = fscanf(obj1);
    freq = str2num(freq)
    bandWeightStr = fscanf(obj1);
    bandWeight = str2num(bandWeightStr)
    if (bandWeight <= 20)
        red = 1;
        green = bandWeight/20;
        set(handles.alignText,'String','Re-Position Reader')
        set(handles.alignText,'BackgroundColor',[red,green,0])
    else
        green = 1;
        red = 1-(bandWeight-20)/20;
        if (red < 0)
            red = 0;
        end
    end
    if (bandWeight < 30)
        set(handles.alignText,'String','Intermediate Aligment')
        set(handles.alignText,'BackgroundColor',[red,green,0])
    else
        set(handles.alignText,'String','Alignment Good')
        set(handles.alignText,'BackgroundColor',[red,green,0])
    end
end
% Calculate pH Value corresponding to frequency
pHtemp = freq;
pHReading = [pHReading, pHtemp];
end

% Turn off VCO
fprintf(obj1, '%c', ClockOffCmd);

% Finds the average if the user enters a Value greater than 1
if (numAverages > 1)
    pHValue = mean(pHReading);
else
    pHValue = pHReading;
end

% Print the single result to the user
if (bandWeight <= 17)
    set(handles.pHDisplay, 'String', 'Invalid Reading');
else
    set(handles.pHDisplay, 'String', num2str(pHValue));
end

% --- Executes on button press in monitorButton.
function monitorButton_Callback(hObject, eventdata, handles)
if (get(hObject, 'value') == 1)
    % Disable Input While Monitoring
    set(handles.errorBox, 'String', 'Message Box')
    set(handles.readButton, 'Enable', 'off')
    set(handles.connectButton, 'Enable', 'off')
    set(handles.disconnectButton, 'Enable', 'off')
    set(handles.alignButton, 'Enable', 'off')
    % Variable in seconds that represents the maximum time interval allowed for
    % continuous powering for measurements
    timeThreshold = 2;
    % Time in seconds for power up to occur before taking measurements
    timeDelay = 2;
    global t;
    global pH;
    global obj1;
    ADCcmd = '1';
    ClockOnCmd = '2';
    ClockOffCmd = '3';
    interval = str2num(get(handles.editInterval, 'String'));
duration = str2num(get(handles.editLength,'String'));
numAverages = str2num(get(handles.editAverage,'String'));
axes(handles.axes1);
pHMax = str2num(get(handles.editpHMax,'String'));
pHMin = str2num(get(handles.editpHMin,'String'));
pH = [];
% Turn on VCO and give it time to charge up the system
fprintf(obj1,'%c',ClockOnCmd);
pause(timeDelay);
%This loop will run as long until the max duration
for i = 0:interval:duration
tic;
%End monitoring if the user toggles the button
if (get(hObject,'Value')==0)
    break;
end
pHReading = [];
for j = 1:numAverages
    %Tell Arduino to execute one reading
    flushinput(obj1);
    fprintf(obj1,'%c',ADCcmd);
    %Get Reading from Arduino
    freq = fscanf(obj1);
    freq = str2num(freq)
    bandWeight = fscanf(obj1);
    bandWeight = str2num(bandWeight)
    if (bandWeight <= 20)
        red = 1;
        green = bandWeight/20;
        set(handles.alignText,'String','Re-Position Reader')
        set(handles.alignText,'BackgroundColor',[red,green,0])
    else
        green = 1;
        red = 1-(bandWeight-20)/20;
        if (red < 0)
            red = 0;
        end
        if (bandWeight < 30)
            set(handles.alignText,'String','Intermediate Aligment')
            set(handles.alignText,'BackgroundColor',[red,green,0])
        end
    end
end
else
    set(handles.alignText,'String','Alignment Good')
    set(handles.alignText,'BackgroundColor',[red,green,0])
end
end

%Calculate pH Value corresponding to frequency
pHtemp = freq;
pHReading = [pHReading,pHtemp];
end

%Finds the average if the user enters a Value greater than 1
if(numAverages > 1)
    pHValue = mean(pHReading);
else
    pHValue = pHReading;
end

%Update Current Value and Plot
if (bandWeight <=17)
    set(handles.pHDisplay,'String','Invalid Reading');
else
    set(handles.pHDisplay,'String',num2str(pHValue));
end

pH = [pH,pHValue];
t = 0:interval:i;
plot(t,pH);
title('Ph vs. Time');
xlabel('Time [sec]');
ylabel('pH');
grid on
set(handles.axes1,'YLim',[pHMin,pHMax]);
if((interval-toc) < 0)
    set(handles.errorBox,'String','Error unable to keep up with sampling interval. Reduce sampling interval or number of pts. averaged')
    break;
end

%Pause for the remainder of the monitoring duration
if((interval-toc) > timeThreshold)
    fprintf(obj1,'%c',ClockOffCmd);
    %Pause for the rest of the interval minus the time delay
    pause(interval-toc-timeDelay);
    %Allow the transponder to charge back up
fprintf(obj1,'%c',ClockOnCmd);
pause(timeDelay);
else
    pause(interval-toc);
end
pause(1);
fprintf(obj1,'%c',ClockOffCmd);
% Re enable input from the user
set(handles.readButton,'Enable','on')
set(handles.connectButton,'Enable','on')
set(handles.disconnectButton,'Enable','on')
set(handles.alignButton,'Enable','on')
set(handles.monitorButton,'Value',0)
end
% --- Executes on button press in alignButton.
function alignButton_Callback(hObject, eventdata, handles)
% Disable Input While Monitoring
set(handles.errorBox,'String','Message Box')
set(handles.readButton,'Enable','off')
set(handles.connectButton,'Enable','off')
set(handles.disconnectButton,'Enable','off')
set(handles.monitorButton,'Enable','off')
% Variable in seconds that represents the maximum time interval allowed for
% continuous powering for measurements
timeThreshold = 2;
global obj1;
ADCcmd = '1';
ClockOnCmd = '2';
ClockOffCmd = '3';
duration = 100;
interval = 1.5;
% Time in seconds for power up to occur before taking measurements
timeDelay = 2;
% Turn on VCO and give it time to charge up the system
fprintf(obj1,'%c',ClockOnCmd);
pause(timeDelay);
% This loop will run as long until the max duration
for i = 0:interval:duration
    tic;
if (get(hObject,'Value')==0)
    break;
end

%Tell Arduino to execute one reading
flushinput(obj1);
fprintf(obj1,'%c',ADCcmd);

%Get Readings from Arduino
freq = fscanf(obj1);
freq = str2num(freq);
bandWeight = fscanf(obj1);
bandWeight = str2num(bandWeight);

%If signal strength is weak
if (bandWeight <= 20)
    %Set Red component of color to 1
    red = 1;
    %Adjust green so box appears more yellow when signal is stronger
    green = bandWeight/20;
    set(handles.alignText,'String','Re-Position Reader')
    set(handles.alignText,'BackgroundColor',[red,green,0])
else
    %Set Green component of color to 1
    green = 1;
    %Adjust green so box appears more yellow when signal is weaker
    red = 1-(bandWeight-20)/20;
    %Ensures the red value cannot be negative
    if (red < 0)
        red = 0;
    end

    %Intermediate Alignment
    if (bandWeight < 30)
        set(handles.alignText,'String','Intermediate Aligment')
        set(handles.alignText,'BackgroundColor',[red,green,0])
    %Good Alignment
    else
        set(handles.alignText,'String','Alignment Good')
        set(handles.alignText,'BackgroundColor',[red,green,0])
    end
end
%Pause for the remainder of the monitoring duration
pause(interval-toc);
end
pause(1);
fprintf(obj1,'%c',ClockOffCmd);
% Re enable input from the user
set(handles.readButton,'Enable','on')
set(handles.disconnectButton,'Enable','on')
set(handles.monitorButton,'Enable','on')
set(handles.alignButton,'Value',0)

% hObject handle to alignButton (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of alignButton

function editLength_Callback(hObject, eventdata, handles)
% hObject handle to editLength (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editLength as text
% str2double(get(hObject,'String')) returns contents of editLength as a double

% --- Executes during object creation, after setting all properties.
function editLength_CreateFcn(hObject, eventdata, handles)
% hObject handle to editLength (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
    get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end

function editInterval_Callback(hObject, eventdata, handles)
% hObject handle to editInterval (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editInterval as text
%       str2double(get(hObject,'String')) returns contents of editInterval as a
double

% --- Executes during object creation, after setting all properties.
function editInterval_CreateFcn(hObject, eventdata, handles)
% hObject handle to editInterval (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
    get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function editAverage_Callback(hObject, eventdata, handles)
% hObject handle to editAverage (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editAverage as text
%       str2double(get(hObject,'String')) returns contents of editAverage as a
double
% --- Executes during object creation, after setting all properties.
function editAverage_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editAverage (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
    get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in errorBox.
function errorBox_Callback(hObject, eventdata, handles)
% hObject    handle to errorBox (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns errorBox contents as
%        cell array
%        contents{get(hObject,'Value')} returns selected item from errorBox

% --- Executes during object creation, after setting all properties.
function errorBox_CreateFcn(hObject, eventdata, handles)
% hObject    handle to errorBox (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
    get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit8_Callback(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
111
% eventdata reserved – to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit8 as text
% str2double(get(hObject,'String')) returns contents of edit8 as a double

% --- Executes during object creation, after setting all properties.
function edit8_CreateFcn(hObject, eventdata, handles)
    % hObject handle to edit8 (see GCBO)
    % eventdata reserved - to be defined in a future version of MATLAB
    % handles empty - handles not created until after all CreateFcns called

    if ispc && isequal(get(hObject,'BackgroundColor'),
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

function editpHMin_Callback(hObject, eventdata, handles)
    % hObject handle to editpHMin (see GCBO)
    % eventdata reserved - to be defined in a future version of MATLAB
    % handles structure with handles and user data (see GUIDATA)

    % Hints: get(hObject,'String') returns contents of editpHMin as text
    % str2double(get(hObject,'String')) returns contents of editpHMin as a double

    % --- Executes during object creation, after setting all properties.
function editpHMin_CreateFcn(hObject, eventdata, handles)
    % hObject handle to editpHMin (see GCBO)
    % eventdata reserved - to be defined in a future version of MATLAB
    % handles empty - handles not created until after all CreateFcns called

    if ispc && isequal(get(hObject,'BackgroundColor'),
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
set(hObject,'BackgroundColor','white');
end

function editpHMax_Callback(hObject, eventdata, handles)
% hObject handle to editpHMax (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editpHMax as text
% str2double(get(hObject,'String')) returns contents of editpHMax as a
double

% --- Executes during object creation, after setting all properties.
function editpHMax_CreateFcn(hObject, eventdata, handles)
% hObject handle to editpHMax (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
    get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes when user attempts to close figure1.
function figure1_CloseRequestFcn(hObject, eventdata, handles)
% hObject handle to figure1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
global obj1
%Close the serial port communicating with the Arduino
if ~isempty(obj1)
    fclose(obj1);
end
% Hint: delete(hObject) closes the figure
delete(hObject);

% --- Executes on button press in exportSNPBtn.
function exportSNPBtn_Callback(hObject, eventdata, handles)
global t;
global pH;
saveAs = strcat('VNA_Data_',date,'.s2p');
try
    fileName = 0;
    path = 0;
    [fileName, path] = uigetfile( ... 
        {'*.mat','MATLAB Files (*.mat)';
        '.*', 'All Files (*.*)'});
    if ~isequal(fileName, 0) || ~isequal(path, 0)
        try
            addpath(path);
            load(fileName,'-mat');
            catch ME
                switch ME.identifier
                    otherwise
                        rethrow(ME)
                end
        end
end
rfwrite(t,pH,'fname.s2p','Format','RI')
name = strsplit(fileName,'.);
movefile('fname.s2p', char(strcat(path, strcat(name(1,1),'.s2p'))))
set(handles.errorBox, 'string', 'Touchstone file exported successfully.');
catch
    set(handles.errorBox, 'string', '*Error exporting touchstone file.*');
end

% hObject    handle to exportSNPBtn (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- Executes on button press in exportCSVBtn.
function exportCSVBtn_Callback(hObject, eventdata, handles)

% hObject    handle to exportCSVBtn (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global pH;
global t;
saveAs = strcat('pH_Measurement', date, '.txt');
try
  fileName = 0;
  path = 0;
  [fileName, path] = uigetfile(  
    {'*.mat', 'MATLAB Files (*.mat)'  
    '.*', 'All Files (*.*)'});
  if ~isequal(fileName, 0) || ~isequal(path, 0)
    try
      addpath(path);
      load(fileName, '-mat');
    catch ME
      switch ME.identifier
        otherwise
          rethrow(ME)
    end
  end
end
pH
csvwrite('fname.txt', transpose([t; pH]));
name = strsplit(fileName, '.');
movefile('fname.txt', char(strcat(path, strcat(name(1,1), '.txt'))))
set(handles.errorBox, 'string', 'File exported successfully.');
catch
  set(handles.errorBox, 'string', '*Error exporting file.*');
end

% hObject handle to exportCSVBtn (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% --- Executes on button press in saveDataBtn.
function saveDataBtn_Callback(hObject, eventdata, handles)
global pH;
global t;
try
  saveAs = strcat('pH_Measurement_', date);
uisave({'t','pH'},saveAs);
set(handles.errorBox, 'string', 'File saved successfully');
catch
    set(handles.errorBox, 'string', '*Error saving file.*');
end
% hObject handle to saveDataBtn (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% --- Executes on button press in togglebutton6.
function togglebutton6_Callback(hObject, eventdata, handles)
% hObject handle to togglebutton6 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns toggle state of togglebutton6

% --- Executes on button press in openButton.
function openButton_Callback(hObject, eventdata, handles)
global pH;
global t;
fileName = 0;
path = 0;
[fileName, path] = uigetfile( ...
    {'*.mat','MATLAB Files (*.mat)';
    '.*', 'All Files (*.*)'});
if ~isequal(fileName, 0) || ~isequal(path, 0)
    try
        addpath(path);
        load(fileName,'-mat');
        set(handles.errorBox, 'string', strcat(path, fileName, ' imported successfully'));
    catch ME
        switch ME.identifier
            case 'MATLAB:nonExistentField'
                set(handles.errorBox, 'string', 'Error loading file.');
            otherwise
                % Handle other exceptions
            end
    end
end
rethrow(ME)
end
end

plot(t,pH);
title('Ph vs. Time');
xlabel('Time [sec]');
ylabel('pH');
grid on

% hObject    handle to openButton (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
8.3 Appendix 2: Microcontroller Program

/* Arduino Control Program for the Senior Design Project, "A Passively Powered pH Sensor for the Study of Gastric Disorders".
 * Author: Kevin Hart
 * Version: 1.0
 * Date Last Modified: 2/10/19
 */
#define FASTADC 0
#define LIN_OUT 1
#define FHT_N 256 // Number of Sampled Points, Must be a power of 2
// Digital Pin Assignments
#define VCO_PIN 9
#define YELLOW_LED_PIN 10
#define GREEN_LED_PIN 11
#include <FHT.h>
#include <Wire.h>

// defines for setting and clearing register bits
#ifndef cbi
#define cbi(sfr, bit) (_SFR_BYTE(sfr) &= ~_BV(bit))
#endif
#ifndef sbi
#define sbi(sfr, bit) (_SFR_BYTE(sfr) |= _BV(bit))
#endif

double sampFreq;
double samplingInterval;
int i;
int j;
int maxValue;
int maxFreqInd;
int totalWeight;
int interpFreq;
int fHigh;
int fLow;
int start;
int peakFreq;
int bandWeight;
unsigned long totalSignalEnergy;
unsigned long bandEnergy;

struct freqSpectrum
{
    double totSignal;
    double bandEnergy;
    int peakFreq;
};

typedef struct freqSpectrum Struct;

void setup(){
    Serial.begin(9600);
    pinMode(VCO_PIN, OUTPUT);
    while (!Serial)
    {
        ; // wait for serial port to connect. Needed for native USB port only
    }

#if FASTADC
    // set prescale to 16
    sbi(ADCSRA,ADPS2) ;
    cbi(ADCSRA,ADPS1) ;
    cbi(ADCSRA,ADPS0) ;
#endif

    pinMode(VCO_PIN, OUTPUT);

    ADCSRA = 0xe5; // set the adc to free running mode
    ADMUX = 0x40; // use adc0
    DIDR0 = 0x01; // turn off the digital input for adc0
}

void sampleADC()
{
    for (int i = 0 ; i < FHT_N ; i++)
    {
        // save 256 samples
        while(!(ADCSRA & 0x10)); // wait for adc to be ready
        ADCSRA = 0xf5; // restart adc
byte m = ADCL; // fetch adc data
byte j = ADCH;
int k = (j << 8) | m; // form into an int
//k -= 0x0200; // form into a signed int
k <<= 6; // form into a 16b signed int
fht_input[i] = k; // put real data into bins convert to voltage value
delayMicroseconds(58); //Reduce Sampling Rate (delayMicroseconds only has a resolution of 4us). Results in sampling Rate of 16kHz
}
return;
}

Struct computeFHTPeak()
{
    Struct s;
    //Measured Frequency compensated for accurate measurement
    sampFreq = 16275;
    maxValue = 0;
    fHigh = 9000;
    fLow = 2000;
    fht_window(); // window the data for better frequency response
    fht_reorder(); // reorder the data before doing the fht
    fht_run(); // process the data in the fht
    fht_mag_lin(); // take the output of the fht
    maxValue = 0;
    maxFreqInd = 0;
    //Find Peak Spectral Frequency Starting at fLow and ending at fHigh
    for (i = floor(fLow/(sampFreq/FHT_N)); i < ceil(fHigh/(sampFreq/FHT_N)); i++)
    {
        if(fht_lin_out[i] > maxValue)
        {
            maxValue = fht_lin_out[i];
            maxFreqInd = i;
        }
    }
    bandWeight = fht_lin_out[maxFreqInd-2] + fht_lin_out[maxFreqInd-1] +
                fht_lin_out[maxFreqInd] + fht_lin_out[maxFreqInd+1] +
                fht_lin_out[maxFreqInd+2];
bandEnergy = (sq(fht_lin_out[maxFreqInd-2]*5.0/1024.0) +
  sq(fht_lin_out[maxFreqInd-1]*5.0/1024.0) +
  sq(fht_lin_out[maxFreqInd]*5.0/1024.0) +
  sq(fht_lin_out[maxFreqInd+1]*5.0/1024.0) +
  sq(fht_lin_out[maxFreqInd+2]*5.0/1024.0));
interpFreq = 0;
for (i = (maxFreqInd - 2); i <= (maxFreqInd + 2); i++)
{
  interpFreq = interpFreq +
    i*(sampFreq/FHT_N)*(float(fht_lin_out[i])/bandWeight);
}
s.peakFreq = interpFreq;
s.bandEnergy = bandEnergy;
return(s);

//Main Function
void loop()
{
  Struct s;
  //Remain until it receives command from GUI
  if (Serial.available() > 0)
  {
    //Take Reading from MATLAB GUI
    int f = Serial.parseFloat();
    //Read ADC
    if (f == 1)
    {
      digitalWrite(GREEN_LED_PIN, LOW);
      digitalWrite(YELLOW_LED_PIN, LOW);
      //Get ADC reading
      sampleADC();
      //Calculate peak of FHT result
      s = computeFHTPeak();
      //Print the result back to the GUI
      Serial.println(s.peakFreq);
      delay(100);
      Serial.println(s.bandEnergy/100);
      if((s.bandEnergy/100) >= 30)
      {
        digitalWrite(GREEN_LED_PIN, HIGH);
        digitalWrite(YELLOW_LED_PIN, LOW);
      }
    }
  }
digitalWrite(GREEN_LED_PIN, HIGH);
}
else if((s.bandEnergy/100) >= 15)
{
    digitalWrite(YELLOW_LED_PIN, HIGH);
}
//Turn on Clock
if (f == 2)
{
    digitalWrite(VCO_PIN, HIGH);
}
//Turn off Clock
if (f == 3)
{
    digitalWrite(VCO_PIN, LOW);
}