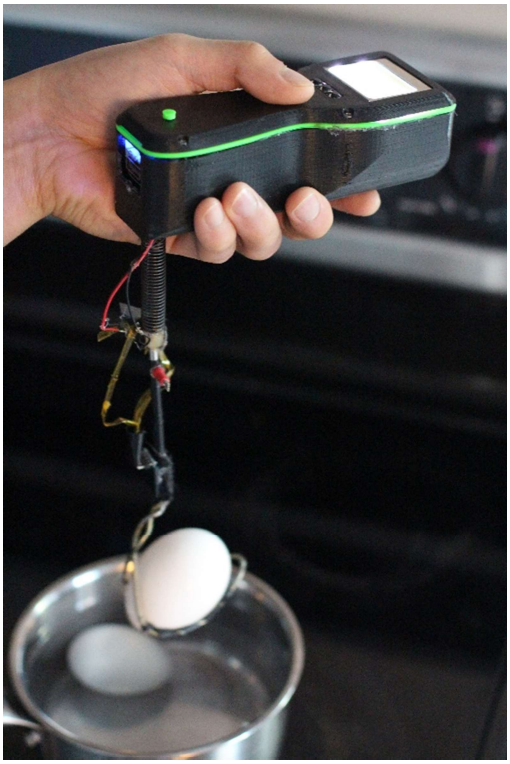
# MECH 423 Final Project Report

## Engineering the Perfect Soft-Boiled Egg



Cooking eggs are a ubiquitous activity in every household and commercial kitchen. Eggs are both inexpensive, healthy, and can be cooked by a multitude of ways. However, crafting the perfect soft-boiled egg becomes a non-trivial task due to the variability in egg size, starting temperature, stovetop thermal conductivity, and personal preference of yolk consistency. Thus, there presents a need for a method to determine yolk consistency without simply breaking the egg open.

We developed a hand-held device to analytically determine the consistency of egg yolks by measuring its oscillation response when suspended from the end of a spring. The device includes a load cell to measure the mass of the egg and an accelerometer to measure the oscillation response. The device uses an Arduino to record data in a buffer and send it through serial communication to a C# program to display the data. The C# program then feeds the data to MATLAB for analysis.

April 18, 2016

Justin Liang justin.jw.liang@gmail.com
Justin Lam justin.mk.lam@gmail.com

# Table of Contents

# 1 Introduction

## 1.1 Objective

The objective of this project is to develop a hand-held device to analytically determine the consistency of egg yolks by measuring the oscillation response of a suspended egg.

## 1.2 Rationale

A device that combines the culinary art of cooking eggs and physics of oscillatory second order systems does not exist. By taking this idea to fruition, we will learn about the design and implementation of a precision, handheld mechatronics device that may hold potential for consumer adoption. Due to this project's uniqueness, we plan to improve the design even after the completion of this course with the end goal of conducting a crowd-funded campaign to take our device to market.

# 2 Summary of Functional Requirements

A list of functional requirements of the device is presented below.

| Functional Requirements | Percent Effort | Assignee |
|---|---|---|
| FR1: Measure egg mass from strain gauge | 10 | Lam |
| FR2: Measure oscillation from accelerometer | 5 | Lam |
| FR3: Control LCD screen | 10 | Liang |
| FR4: Mechanical design and fabrication | 25 | Lam |
| FR5: Integration and user interaction | 10 | Lam |
| FR6: C# interface | 10 | Liang |
| FR7: MATLAB data analysis | 30 | Liang |

## 2.1 FR1: Measure egg mass from strain gauge

### *2.1.1 Approach and Design*

Use a strain gauge attached to both ends of a spring in order to measure the deflection of the spring. Interface with the MCU to read the resistor values. Calibrate the resistances to a corresponding mass.

#### *2.1.2 Inputs and Outputs*

Strain gauge takes 3V DC and outputs a resistance. This resistance will be converted to a mass and used for calculation of system's natural frequency.

#### *2.1.3 Parameters*

The metrics for the strain gauge are listed below.

- Dimensions: 45 x 9 x 6 mm
- Capacity: 5 kg
- Precision: 1g
- Operating voltage: 3V

#### *2.1.4 Development Plan*

The development plan for implementing the strain gauge is outlined below.

1. Read resistance of the strain gauge using ADC on the MCU
2. Calibrate the strain gauge resistance to mass

#### *2.1.5 Test Plan and Results*

The test plan for each of the development plans is outlined below.

1. Read resistance of the strain gauge using ADC on the MCU.
    a. Verify by displaying the results on the screen
2. Calibrate the strain gauge resistance to mass
    a. Attach the strain gauge to the spring setup
    b. Record resistances at varying known weights
    c. Determine the transfer function from resistance to mass

The result of this can be seen below:

### 2.2 FR2: Measure oscillation from accelerometer

#### *2.2.1 Approach and Design*

Use an accelerometer and attach it to the rotating shaft in order to measure the oscillation. Interface with the MCU to read the on and off output states from the sensor. Calibrate the decay and/or frequency to determine how hardboiled the egg is.

#### *2.2.2 Inputs and Outputs*

The accelerometer used is the MMA7361L. It measures the acceleration in 3 axis. For the purpose of this project, only the y-axis is used as an input for this system. This is because we are measuring the rotation of the shaft. Since the MMA7361L is analog, the y-axis will be in the form of a voltage that is measured by the Arduino. By measuring this, we can get a plot of acceleration versus time and we can correlate that to the frequency of oscillation and the decay in the oscillation.

#### *2.2.3 Parameters*

The metrics for the accelerometer sensor are listed below.

- XYZ measurements (+/- 1.5g, +/- 6g)
- Low voltage operation 2.2 V – 3.6 V
- Signal conditioning with low pass filter
- Robust design, high shocks survivability

The signal conditioning will be useful to reduce some of the noise. Also, the ability to select the measurement (1.5 vs 6g) will be helpful to attain higher sensitivity/resolution.

#### *2.2.4 Development Plan*

The development plan for implementing the accelerometer is outlined below.

1. Read the y-axis output of the accelerometer
2. Store the data in an array or circular buffer for post processing

#### *2.2.5 Test Plan and Results*

1. Read the y-axis output of the accelerometer
    a. Verify by transmitting the result through UART to a simple C# program (See C# section for results).
2. Store the data in an array or circular buffer for post processing
    a. Verify by transmitting the result through UART to a simple C# program (See C# section for results).

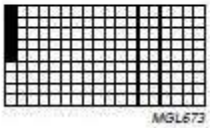
### 2.3 FR3: Control LCD screen

#### *2.3.1 Approach and Design*

Develop functions in C to convert decimal values to char and send it via UART to the LCD screen. This is done through Arduino.

### *2.3.2 Inputs and Outputs*

The input to the LCD is a single char byte. The output is the char byte displaying on the LCD screen. The LCD screen is a 48 x 84 pixel matrix. Each 8 bit column can be controlled by sending a serial bus byte, for example, take a look at the input and output in the image below:

| SERIAL BUS BYTE | | | | | | | | | DISPLAY |
|---|---|---|---|---|---|---|---|---|---|
| D/$\overline{C}$ | DB7 | DB6 | DB5 | DB4 | DB3 | DB2 | DB1 | DB0 | |
| start | | | | | | | | | |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |  |



Furthermore, the LCD has initialization inputs that initialize the screen and the brightness of the screen.

### *2.3.3 Parameters*

The LCD consists of a layer of molecules aligned between two transparent electrodes and two polarizing filters, thus, magnetic

fields can alter the results of the display. Need to be careful about magnet placement. As for some other relevant parameters:

- 48 x 84 pixel LCD
- Serial interface maximum 4.0 Mbits/s
- Low power consumption (suitable for battery operated systems)

The small screens means we are limited in how much we can display. The low power consumption is perfect for this device as the end goal of this device is to be portable, thus, saving power is essential for the success of this feature.

### *2.3.4 Development Plan*

The development plan for implementing the LCD is outlined below.

1. Use MCU to output char byte via UART

### *2.3.5 Test Plan and Results*

1. Use MCU to output char byte via UART
    a. Check to see if LCD displays the right values

As we can see in the image below, the LCD is correctly displaying the values that was sent to it by the Arduino:

## 2.4 FR4: Mechanical Design and Fabrication

### *2.4.1 Approach and Design*

The mechanical design will consist of an extension spring attached to the end of a handle. A proof of concept model is shown below.

Figure 1: Proof of concept model to measure oscillation. (*Image from woodgears.ca*)

Our objective is to digitize these movements. Our plan is to use a strain gauge to measure the mass and a spring for the oscillations. A sketch of the proposed device is shown in the figure below.

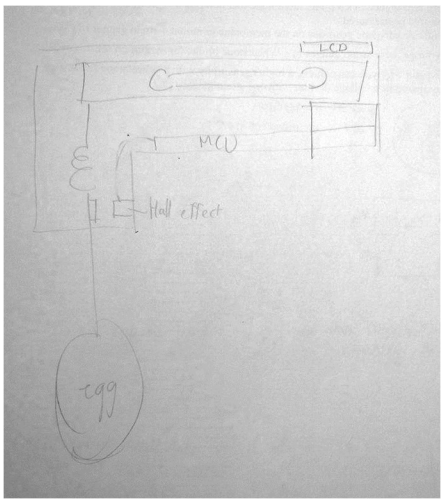



Figure 2: Sketch of proposed device.

The device was designed to be a handheld and battery operated. Thus, the handle body had to be designed to be as small as possible in order to be comfortable to hold with one hand. Since we were using headers to attach the Arduino Pro Mini and Nokia LCD screen, we had to find a way to minimize this footprint. We designed the LCD screen to fit on top of the Arduino on the protoboard, as shown in the figure below.

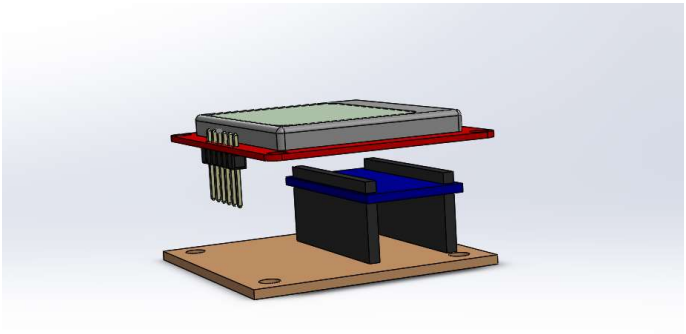


Figure 3: Determining spacing of electronics to minimize the footprint.

The rest of the electronic components were modeled in a similar method to determine the minimum volume that the handle would have to contain.

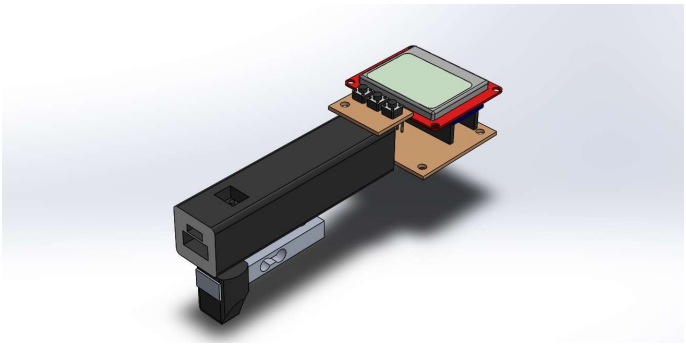


Figure 4: PCB module, buttons, USB battery pack, and strain gauge.

Once the minimum volume was determined, the handle body was designed around it with the intention of it being manufactured by 3D printing processes. This affects some design features (ie. the extrusions of the mounting holes) since 3D printers are best at reproducing features that are along the vertical axis. The region around the battery pack was slimmed down to indicate that is where user should grab the device. The handle body with the mounting features for electronic components are shown in Figure 5 below.

Figure 5: Handle body design

The fully integrated mechanical and electrical components are shown in Figure 6 and Figure 7. The electronic components are secured in the handle body using M2 and M3 fasteners. The USB port at the base of handle is from the USB battery pack, allowing users to charge their mobile phones while cooking eggs with our device.

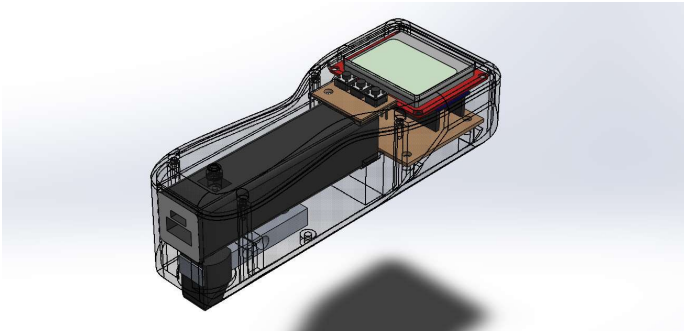


Figure 6: Handle body designed around the electronics.

Figure 7: Fully assembled device.

### 2.4.2 *Inputs and Outputs*

The inputs are the user placing the egg on the holder and shaking the device to start the oscillation. The outputs are the spring oscillating and the yolk consistency being measured, where the quantified result is displayed on the LCD screen at the top of the device.

### 2.4.3 *Parameters*

The spring constant was calculated to have required range between 0.09 lb/in to 0.28 lb/in. The operating range of the strain gauge will be 50g to 65g, which is the average weight of an egg. The device (which is essentially just a handle) must be comfortable and ergonomic to use.

### 2.4.4 *Development and Test Plan*

The development is outlined below:

1. Determine physical measurements of sensors and components
2. Develop solid model using CAD based on the physical dimensions
3. 3D print the parts and assemble

The tests for the mechanical design will simply be if the device operates as intended. If not, then we will iterate upon the design, revise, and re-print/make the parts for subsequent testing.

### *2.4.5 Test Results*

The mechanical components went through multiple revisions before the final design was satisfactory. The initial embodiment of the device incorporated a hall-effect sensor to measure the oscillation period. A column sleeve surrounding the spring (Figure 8) was required in order to constrain the oscillating shaft in order for the permanent magnet to be close enough to the hall-effect sensor for reliable readings. Two bearings were used to constrain the shaft with minimal friction (Figure 9).

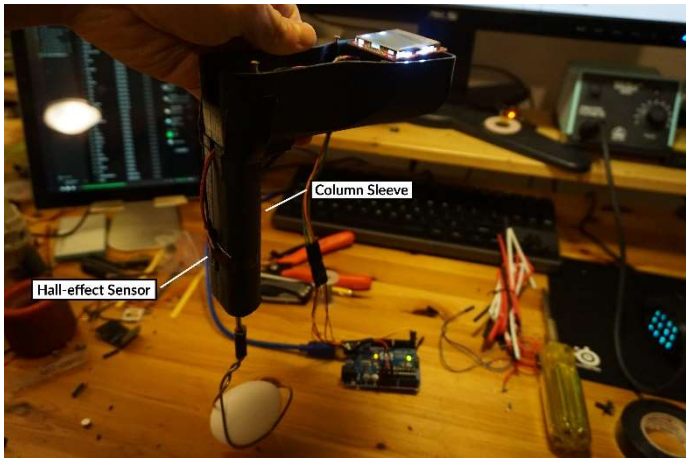



Figure 8: Revision using a hall-effect sensor to measure the oscillation period.

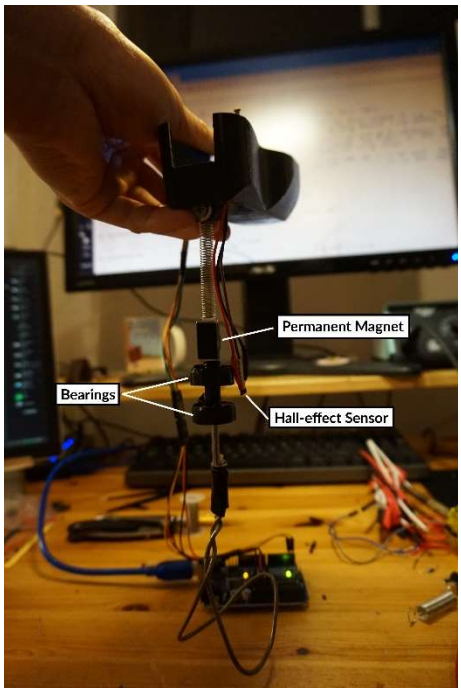



Figure 9: Mechanical components inside the column sleeve.

However, this design proved to be ineffective since the bearings and column sleeve contributed too much friction to the oscillations. We needed a method to measure unconstrained oscillations, so we turned to using a 3-axis accelerometer. Using acceleration to measure oscillations increases the complexity of analysis but radically simplifies the mechanical design (Figure 10).

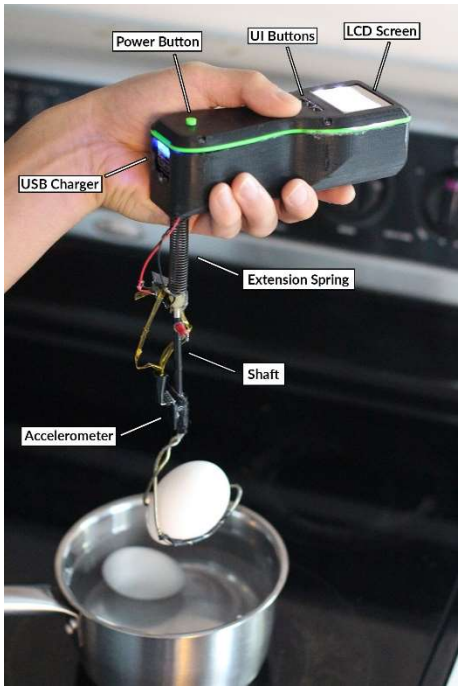



Figure 10: Revision using an accelerometer to measure oscillations

Achieving proper fit of the 3D printed parts was conducted through trial and error since 3D printed parts undergo thermal expansion, resulting in the dimensions of the final part slightly differing from the intended dimension. After a handful of iterations, the mechanical design was complete.

## 2.5 FR5: Integration and user interaction

### *2.5.1 Approach and Design*

Integrate the electrical and mechanical components by designing a 3D printed enclosure to secure the assembly. Ideally, all components will be enclosed in the handle unit such that no wires are exposed to the user. The device will be modeled in CAD to ensure proper fit.

User interaction will be determined by the on-board LCD screen and navigation buttons. Our goal will be to make the operation of the device as easy and seamless as possible. The screen will display instructions of operation and the measured egg yolk consistency once measurements are collected and analyzed. The buttons will be to navigate through the menu.

The PCB was designed to have a minimized footprint in order to fit on a 45mm x 45mm protoboard (Figure 11). Although the design process was rather painless, the actual assembly turned out to be rather tedious with respect to having to work with through-hole components in such a tight space (Figure 12). Clever soldering with somewhat janky jumper wires was required

for a number of connections due to the lack of available traces (Figure 13). The load cell, accelerometer, battery power, and button PCB are all connected with header pin connections.

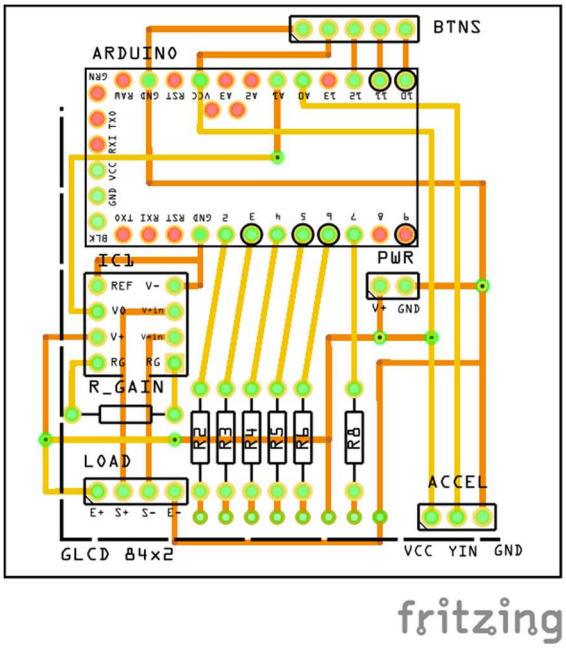



Figure 11: PCB design using Fritzing

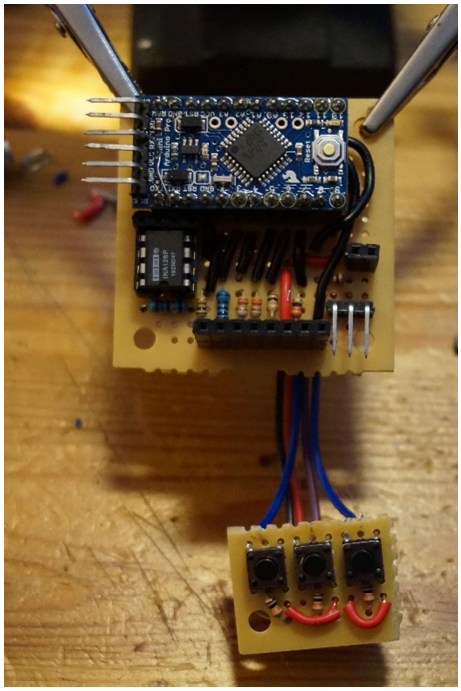


Figure 12: Top side of assembled PCB

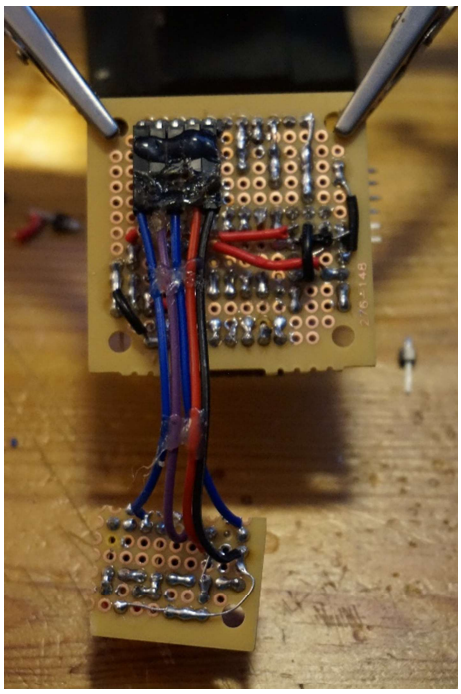


Figure 13: Bottom side of assembled PCB

### *2.5.2 Inputs and Outputs*

The inputs are described below.

- Button switch to start data collection
- Rotary lever to start spring oscillation

The output is described below.

- Egg yolk consistency as a percentage of done-ness (0% = raw, 100% = fully hard-boiled)

### *2.5.3 Parameters*

The parameters for integration and user interaction are listed below.

- Toggle button switch
- Rotary lever with $\geq 45^\circ$ sweep

### *2.5.4 Development Plan*

The development plan for this module is listed below.

1. Model prototype in CAD
2. Test mechanical operation of 3D printed parts
3. Integrate electronics

4. Test operation and ease-of-use of device

### *2.5.5 Test Plan*

The test plan for each of the development plan steps is outlined below.

5. Model prototype in CAD
    a. Review design with partner of mechanical operation, assembly, and integration with electrical components
6. Test mechanical operation of 3D printed parts
    a. Verify that the spring can freely oscillate without constraint by mechanical enclosure or strain gauge
7. Integrate electronics
    a. Verify assembly and fit of the electronic components in the 3D printed parts
8. Test operation and ease-of-use of device
    a. Conduct real-world testing of device with eggs and other users

### *2.5.6 Test Results*

The final integration of the mechanical and electrical components was successful since the electrical components fit inside the mechanical components. The series of images below show an overview of the assembly process.

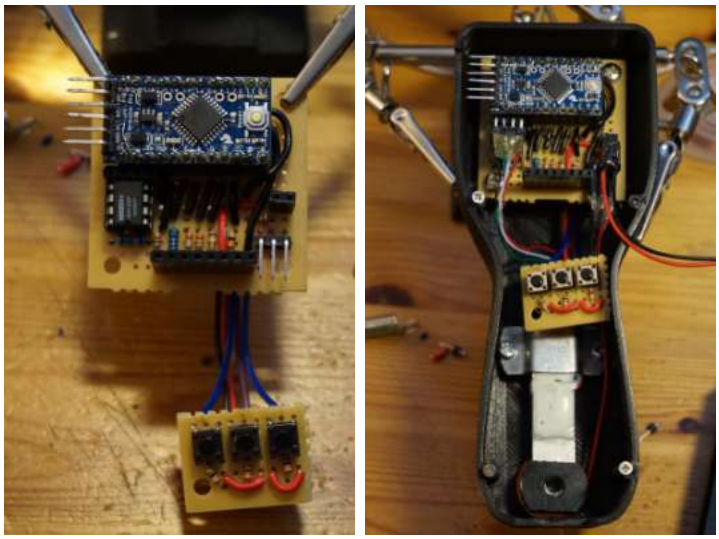


Figure 14: Assembly of the PCB and load cell into the handle body.

Figure 15: Assembly of the USB battery pack, LCD, and faceplate


Figure 16: Fully assembled device

Figure 17: Device in storage

In order to test the consistency and reliability of our device, we designed a jig to oscillate the egg with a standardized input. This was achieved by using a servo motor to create a torsional step input on the shaft (Figure 18, Figure 19). Through this, we were able to conclude that the mechanics of securing the egg while shaking it worked as intended.

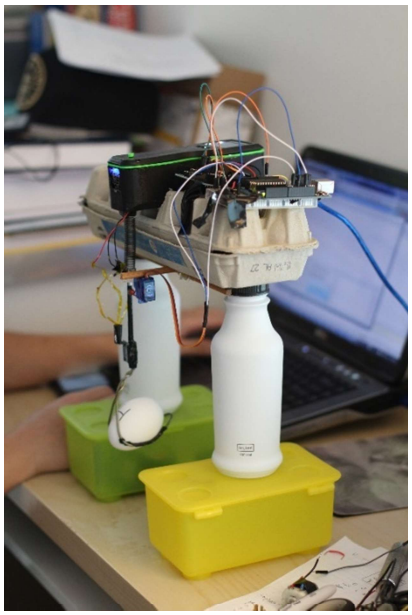


Figure 18: PerfEGGct development kit

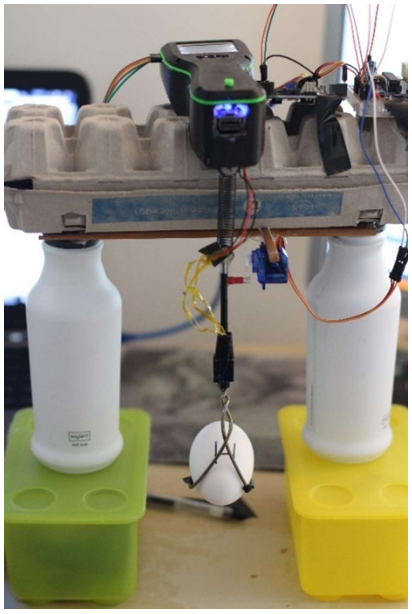


Figure 19: Servo motor arm swings 180° to rotate the shaft

## 2.6 FR6: C# Interface

### *2.6.1 Approach and Design*

The goal of the C# program is to display the accelerometer data for debugging purposes and also to have an interface to save this raw data to a text file. Furthermore, it interfaces with MATLAB and has a button to call the MATLAB code to post process the data. The interface can be seen below:

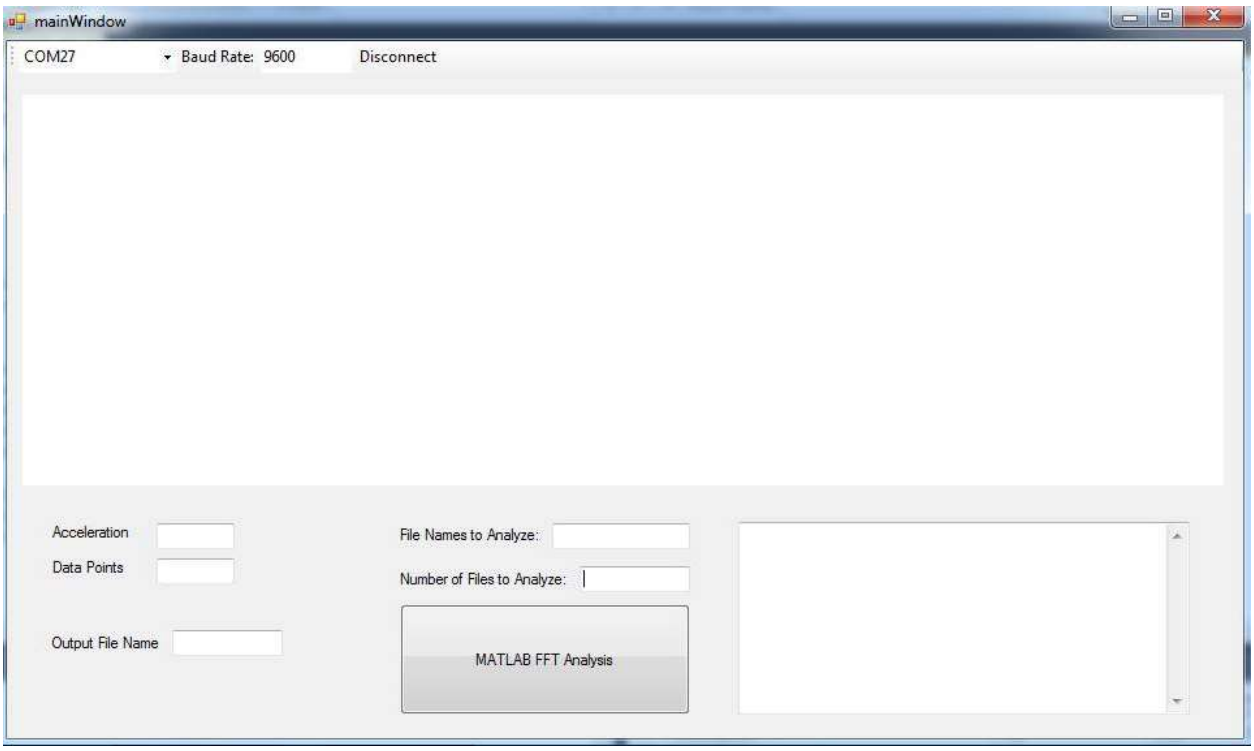



Figure 20 - C# Interface

### *2.6.2 Inputs and Outputs*

The input is the accelerometer data sent through the serial bus from the Arduino. It then outputs this data live onto the interface.

In terms of saving the files, the input is a file name and once the program receives 500 data points, it will output and save this data onto a text file under that name.

To interface with MATLAB, you input the name of the text file into the 'File Names to Analyze' textbox and also input how many files with the suffix to analyze. Then you click the 'MATLAB FFT Analysis' button to call the MATLAB function which processes the signal.

Furthermore, you can select the COM port and baud rate for the serial connection.

### *2.6.3 Parameters*

The parameters for this program are as follows:

- COM port number
- Baud rate
- Acceleration data
- Name of file to save
- Name of saved file to analyze
- Number of files to analyze

The acceleration data will be optimized through MATLAB by filtering the data to analyze only what is relevant, this will be described in the next section.

### *2.6.4 Development Plan*

The development plan for this module is listed below.

1. Build the GUI using the built in C# GUI building interface.
2. Develop a function called ComPortUpdate() to update the COM port when the device is connected to the computer.
3. Develop a function called serCOM_DataReceived() to read new data being sent to the COM port and store this in a circular buffer.
4. Develop a function called btnConnect_Click() to connect to the COM port when this button is clicked.
5. Develop a function called timer_Tick() to continuously display the acceleration data onto the chart and also to write this data into a text file.
6. Develop a function called matlabFFT_Click() to create a MATLAB instance and execute the MATLAB data analysis functions.

### *2.6.5 Test Plan and Results*

The test plan for each of the development plan steps is outlined below.

1. Build the GUI using the built in C# GUI building interface.
    a. Name the variables and edit the parameters to produce the correct interface blocks.
2. Develop a function called ComPortUpdate() to update the COM port when the device is connected to the computer.
    a. Check that the COM port is updated to the correct port by looking at the Device Manager.

3. Develop a function called serCOM_DataReceived() to read new data being sent to the COM port and store this in a circular buffer.
    a. Debug the program and look at the circular buffer and/or make sure the data on the C# program looks correct.
4. Develop a function called btnConnect_Click() to connect to the COM port when this button is clicked.
    a. Check that a connection has been made on the oscilloscope or by looking at the accelerometer data populating the C# chart.
5. Develop a function called timer_Tick() to continuously display the acceleration data onto the chart and also to write this data into a text file.
    a. Make sure that data is being written to the chart and the text file by looking at the chart and checking to see if the text file exists.
6. Develop a function called matlabFFT_Click() to create a MATLAB instance and execute the MATLAB data analysis functions.
    a. Check to see if the plots that pop up are consistent with the output of the MATLAB code (see next section).

The result of this is the following interface, we see that it correctly displays the oscillations in the accelerometer data.

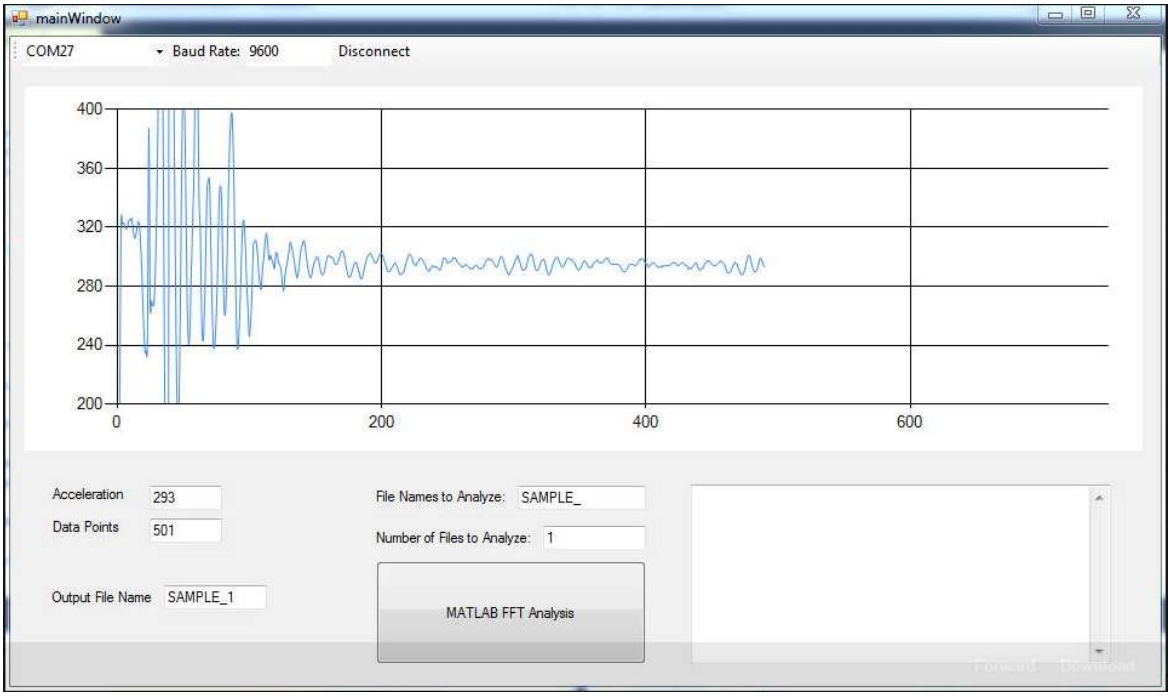



Figure 21 – C# interface with data

## 2.7 FR7: MATLAB Data Analysis

### *2.7.1 Approach and Design*

The objective of the MATLAB data analysis is to process the acceleration data to find a correlation between this data and the yolk consistency of the egg. The idea is to use MATLAB to test different signal processing methods and for future iterations, code them on the Arduino. This is because some of the signal processing algorithms require high processing power and may need to be coded in assembly for optimal performance. Thus, for the purpose of prototyping and proof of concept, MATLAB was used.

Two methods were tested for this project:

1. Analyzing the frequency
2. Analyzing the exponential decay rate

The first method stemmed from a quick analysis of the acceleration output as seen below:


Figure 22 – Hardboiled egg acceleration

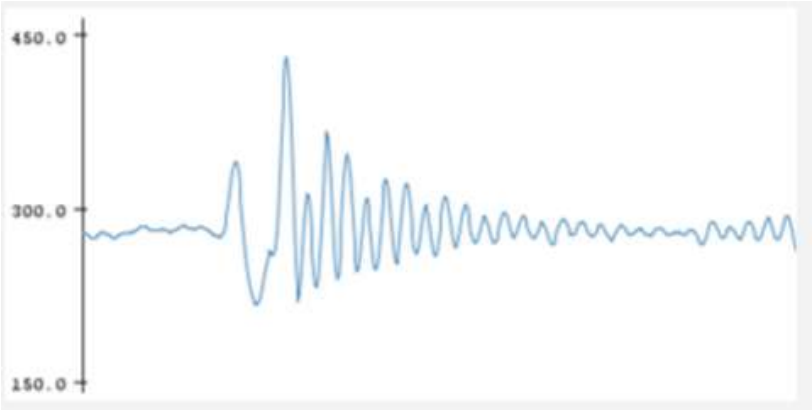



Figure 23 - Raw egg acceleration

Initially, based on these two graphs, it is obvious that the raw egg oscillates at a high frequency, hence, there must be a relationship between the frequency of oscillation and the yolk consistency.

The second method stems from the fact that if the yolk consistency is more raw, we can expect the system to damp faster, hence a larger decay rate.

### *2.7.2 Inputs and Outputs*

The input to the MATLAB code is the text file generated by the C# program containing the raw accelerometer data. The output of the code depends on which method. For the frequency method, the output is the frequency of oscillation of the wave. For the decay rate method, the output is the decay rate of the damped harmonic oscillation.

### *2.7.3 Parameters*

The relevant parameters for each method is as follows:

#### *2.7.3.1 Frequency Method:*

1. Frequency of oscillation
2. Period of oscillation

As seen from the quick visual analysis of the oscillation frequencies of the hard and raw eggs, it is clear that the frequencies are different. Thus, it will be necessary to correlate the rate of oscillation with the yolk consistency.

#### *2.7.3.2 Exponential Decay Rate Method:*

1. Rate of decay
2. Peak values of the curve

To get the exponential decay, an exponential curve will be fit to the peaks of the damped sinusoid.

### *2.7.4 Development Plan*

For testing, three eggs will be used. One is raw, one is boiled for 5 minutes and one is hardboiled. The development plan for this module is listed below.

#### *2.7.4.1 Frequency Method:*

1. Perform a Fast Fourier Transform on the data to determine the main frequencies.
2. Perform Butterworth filtering if necessary.
3. Correlate the frequency with the yolk consistency.

#### *2.7.4.2 Exponential Decay Rate Method:*

1. Fit the peaks of the sinusoid with a curve fitting method.
2. Find the decay rate of the fit.
3. Correlate the decay rate with the yolk consistency.

### *2.7.5 Test Plan and Results*

The test plan for each of the development plan steps is outlined below.

#### *2.7.5.1 Frequency Method*

1. Perform a Fast Fourier Transform on the data to determine the main frequencies

FFT is performed on 5 data sets for each yolk consistency of the egg. The frequency history and the peak frequency of this can be seen below:

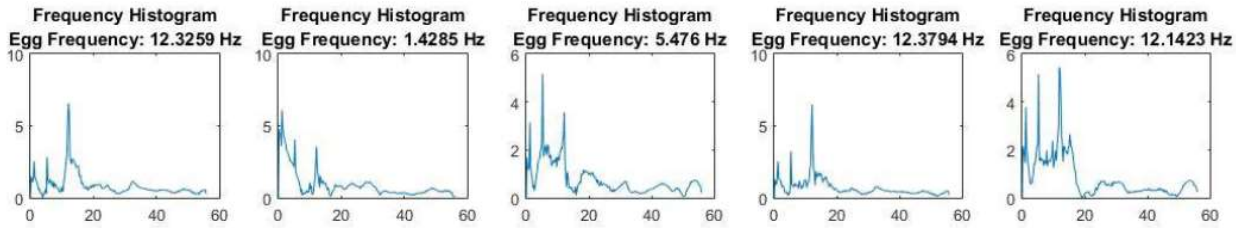



Figure 24 - Frequency Histogram (Hardboiled)

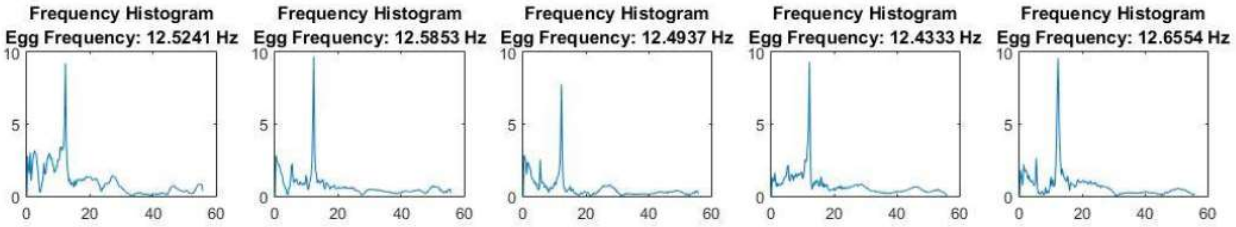



Figure 25 - Frequency Histogram (5 Min Boil)

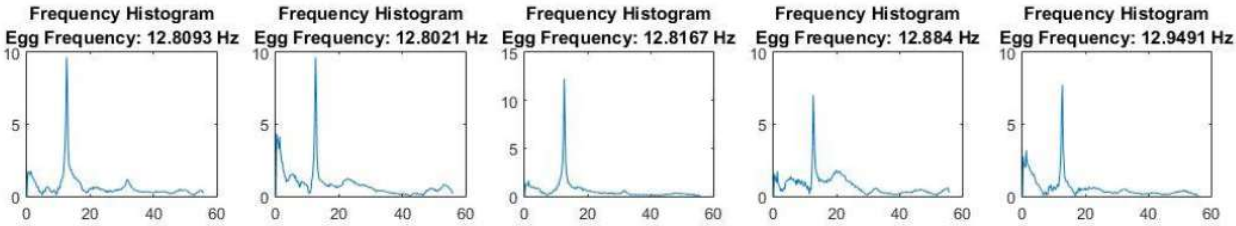



Figure 26 - Frequency Histogram (Softboiled)

2. Perform Butterworth filtering if necessary.

From the data above, it is clear that there are many different frequencies, thus, we tried filtering the data with the Butterworth filter. However, even after doing this, it was difficult to find which frequency is the one that correlates with the yolk consistency.

3. Correlate the frequency with the yolk consistency.

After extensive testing, we found that our original hypothesis is not true. The frequency we see in the hardboiled egg in Figure 22 is actually the frequency of the spring. The frequency we see in the raw egg in Figure 23 is actually the frequency of the vibrations in the spring (this is approximately 5.5 Hz). The problem is that there are too many mechanical vibrations that overshadow the actual frequency of oscillation of the egg (this one is approximately 12.8 Hz).

However, looking at all the frequencies, it is clear that each egg state has its own unique frequency histogram. Thus, one possible method is to use machine learning to find a correlation between the

different frequencies and the yolk consistency. However, since neither of us have experience in this area, this was not done and will be something that could be done for future iterations.

*2.7.5.2 Exponential Decay Rate Method:*

1. Fit the peaks of the sinusoid with a curve fitting method.
2. Find the decay rate of the fit.

A mixture of input signals were used. The data includes data with the raw egg, the 5 min egg, the hardboiled egg and no egg. This is to make sure that the exponential fit works for all cases. The result of this can be seen below:

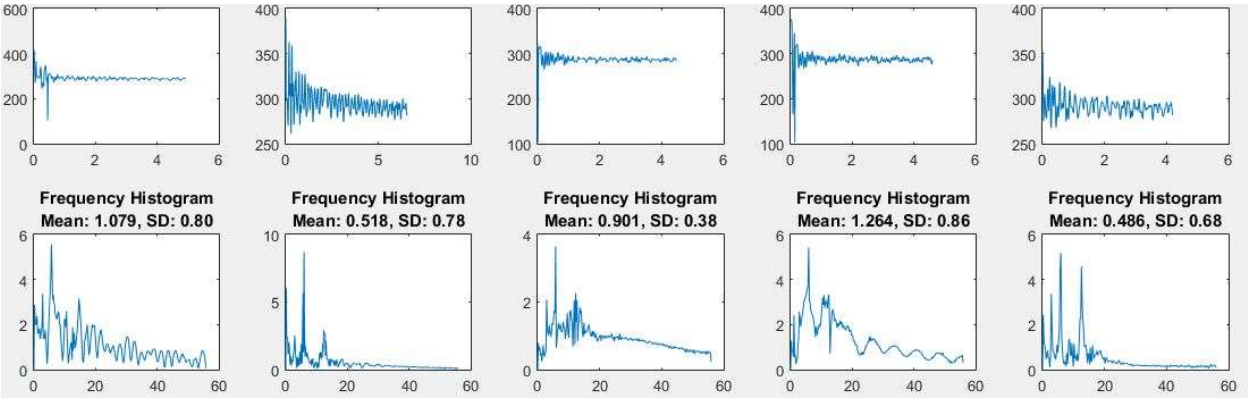



Figure 27 - Mixture of Input Signals

Multiple methods to generate an exponential fit was tested. This is necessary because the data is complicated by the varying sinusoidal frequencies, so certain methods may not work because of this. The methods listed below can be good depending on the data and the frequencies involved.

First off, it is assumed that the data has the form:

$$y = average + be^{-cx}$$

Here average is the average value at steady state.

Thus, for these methods, it is possible to linearize this equation by taking the log of the y values. And then it becomes a simple linear regression problem.

With that being said, let's dive into the different methods that were tested:

**findpeaks() Method**

This method finds all of the peaks in the data and performs an exponential fit on it. The method is a pretty good method for most data, but for this data, since there are multiple sinusoidal frequencies, it gets the wrong peaks. The red x's show the peaks.

```
% Find Peaks Method
[max_num,max_ind] = findpeaks(y(ind));
plot(max_ind,max_num,'x','Color','r'); hold on;
x1 = max_ind;
y1 = log(max_num-avg);
coeffs = polyfit(x1,y1,1)
b = exp(coeffs(2));
c = coeffs(1);
```

This results in the following fit:

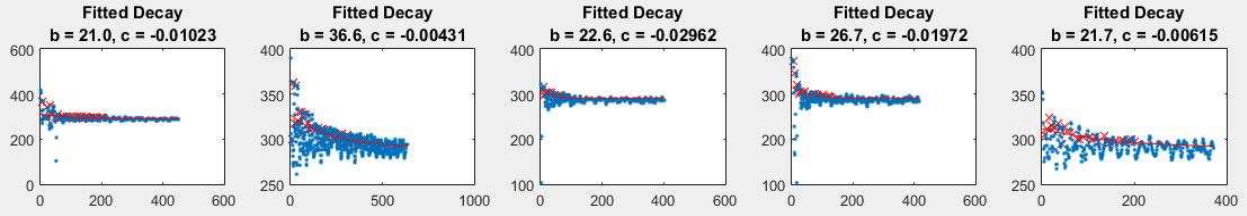



Figure 28 - findpeaks() Fit

### RANSAC

RANSAC is good if most of the data is at the peaks. As seen in the data, because of the multiple frequencies, more peaks exist near the top. Thus, RANSAC may be good in this situation. However, the problem with this data is that not all the data sets are like this. Hence, it occasionally worked.

```
% RANSAC Method
ind = (y > avg);
x1 = x(ind);
y1 = log(y(ind) - avg);
iterNum = 300;
thDist = 0.5;
thInlrRatio = .1;
[t,r] = ransac([x1;y1'],iterNum,thDist,thInlrRatio);
k1 = -tan(t);
b1 = r/cos(t);
% plot(x1,k1*x1+b1,'r'); hold on;
b = exp(b1);
c = k1;
```

This results in the following fit:

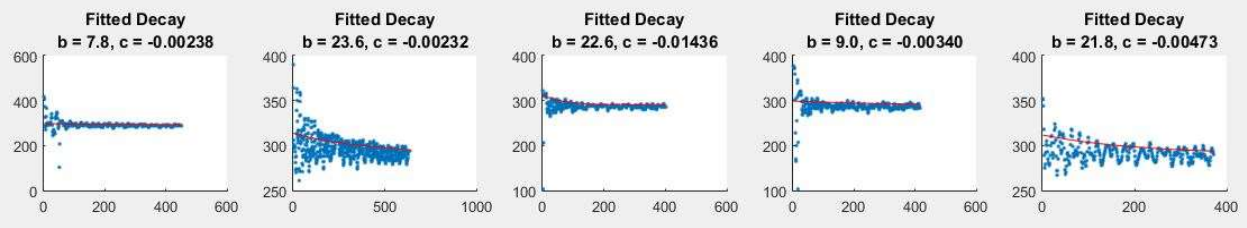



Figure 29 - RANSAC Fit

### lsqlin() Method

This method uses the lsqlin() to constrain the system. However, it ignores the data in the middle. Depending on the data set, this could work really well, though for this signal it did not fit the data very well.

```
% Lsqlin Method
avg = 290;
ind = (y > avg);
x1 = x(ind);
y1 = log(y(ind) - avg);
A = [ones(numel(x1),1),x1(:)]*1.00;
coeffs = lsqlin(A,y1(:),-A,-y1(:),[],[],[],[],[],optimset('algorithm','active-
set','display','off'));
b = exp(coeffs(2));
c = coeffs(1);
```

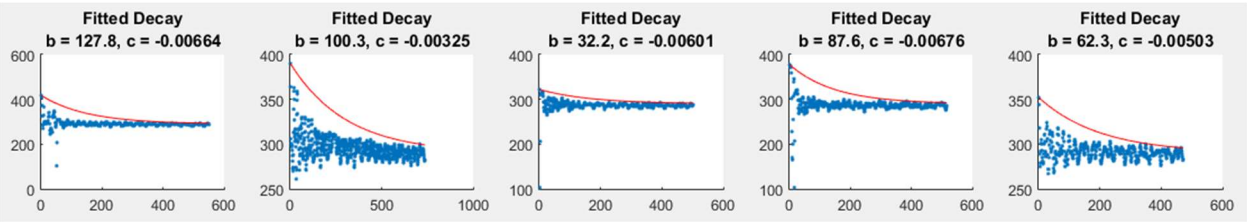



Figure 30 - lsqlin() Method

### Find Peaks in Period

In this method, the peak is found for a certain region and then a fit is found for all of these peaks. This is better than the findpeaks() method because it can help to reduce peaks that are created from the varying sinusoids. Thus, it gets the peaks that best fit the top boundary of the input data signal. From this, we realized that the data may not actually have a perfect exponential fit, hence the difficulty. We see that this method is unable to fit the large peaks at the beginning. This was fixed by only using the first 150 data points and ignoring the steady state data points. Here, the peak is found for every 25 data points.

```
% Incremental Method 2 Unknowns
x1 = [];
y1 = [];
max_num=[];
max_ind=[];
incr = 25;
for i=1:floor(size(y,1)/incr)
    [max_num(end+1),max_ind(end+1)] = max(y(1+incr*(i-1):incr*i));
    max_ind(end) = max_ind(end) + incr*(i-1);
    if max_num(end) > avg
        x1(end+1) = max_ind(end);
        y1(end+1) = log(max_num(end)-avg);
    end
end
plot(max_ind,max_num,'x','Color','r'); hold on;
coeffs = polyfit(x1,y1,1)
b = exp(coeffs(2));
c = coeffs(1);
```

This results in:

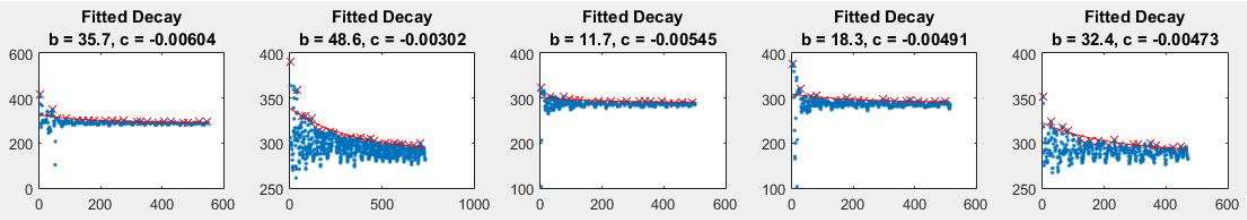



Figure 31 - Using All 500 Data Points

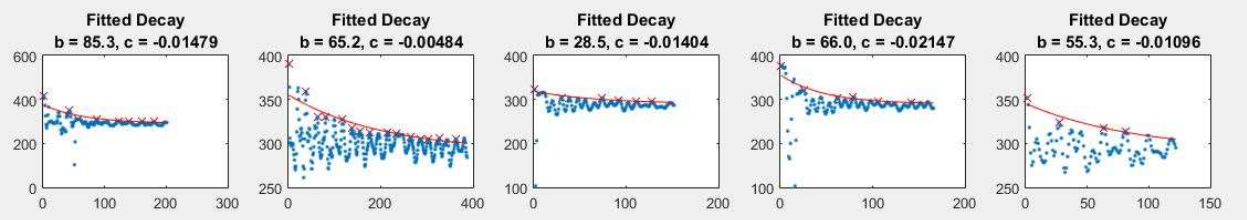



Figure 32 - Using the First 150 Data Points

### Find Peaks in Period With b Constrained

This method constrains the b value since we want it to start at the first peak. We know that the system is:

$$y = average + be^{-cx}$$

To make sure it starts at the first peak we can constrain the system:

$$b = y(1) - average$$

By doing so, we just need to solve for c where,

$$c = (\log(y - avg) - logb)/x$$

Next, we can take the average or median of c. This method is quite good as well, it doesn't fit the value near the end as well but that isn't as big of a deal since the change there is minimal. In addition to this, it is possible to set to data range to only fit the first 150 data points if this is a problem.

From this, it was found that a data point region of 25 data points produced the best fit.

```
% Incremental Method 1 Unknown (b is constrained y(1)-avg = b)
b = y(1) - avg;
c = [];
max_num=[];
max_ind=[];
incr = 25;
for i=1:floor(size(y,1)/incr)
    [max_num(end+1),max_ind(end+1)] = max(y(1+incr*(i-1):incr*i));
    max_ind(end) = max_ind(end) + incr*(i-1);
    if max_num(end) > avg
        c(end+1) = (log(max_num(end)-avg)-log(b))/max_ind(end);
    end
end
c = mean(c); % Or median(c) works just as good
```

This results in:

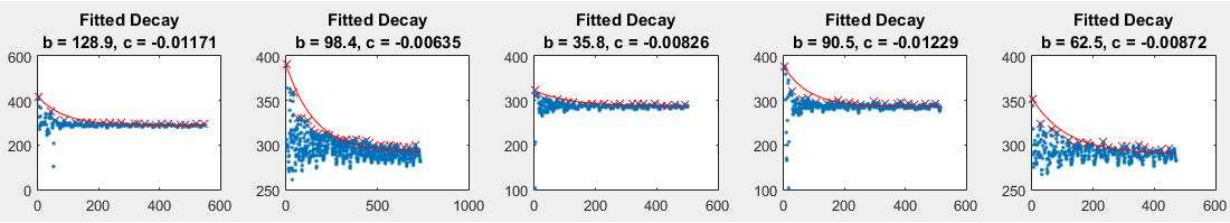



Figure 33 - Fit With Peak at Every 25 Data Points and Taking the Mean of c

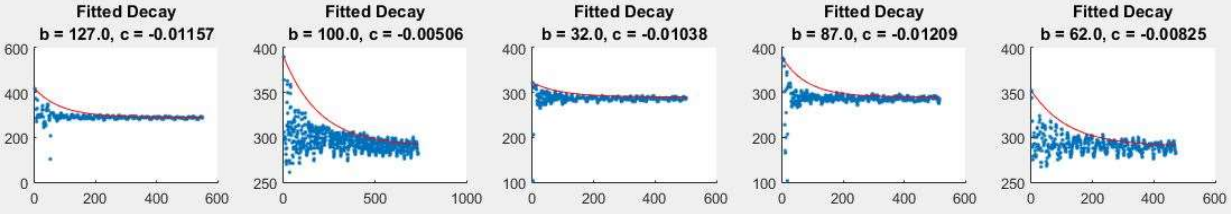



Figure 34 - Fit With Peak at Every 25 Data Points and Taking the Median of c

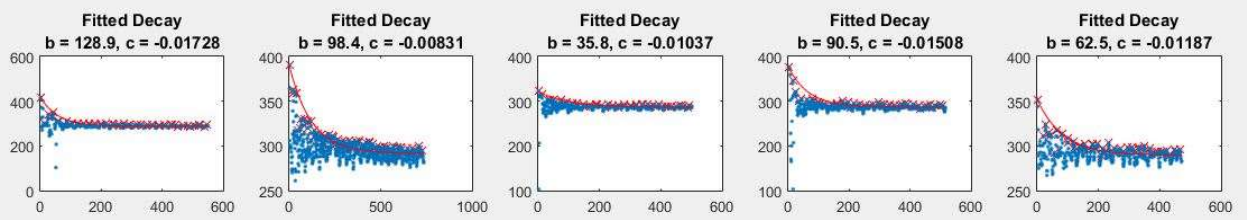



Figure 35 - Fit With Peak at Every 10 Data Points and Taking the Mean of c

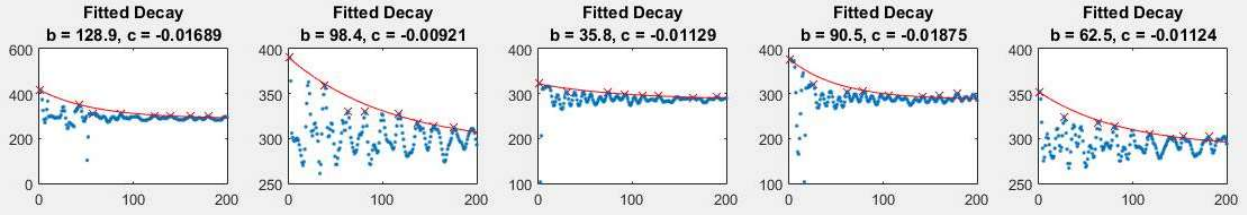



Figure 36 - Fit With Peak at Every 25 Data Points and Taking the Mean of c (First 200 Data Points)

From the plots, we see that the mean and median of c produces similar plots. And when the window is reduced, it produces a tighter fit to the data. The best fit can be produced with 25 data points, mean of c and taking only the first 200 data points.

3. Correlate the decay rate with the yolk consistency.

With the MATLAB code working, the next step is to correlate the decay rate with the yolk consistency. This is described in the system evaluation section.

# 3 System Evaluation

To test the entire system, we use three different eggs – raw, 5 min boil, hardboiled.


Figure 37 - Eggs Used for Testing

We then fit the data with an exponential curve. We ran hundreds of tests (all of which can be found on GitHub), 5 of these tests can be found below. These 5 tests were run consecutively.

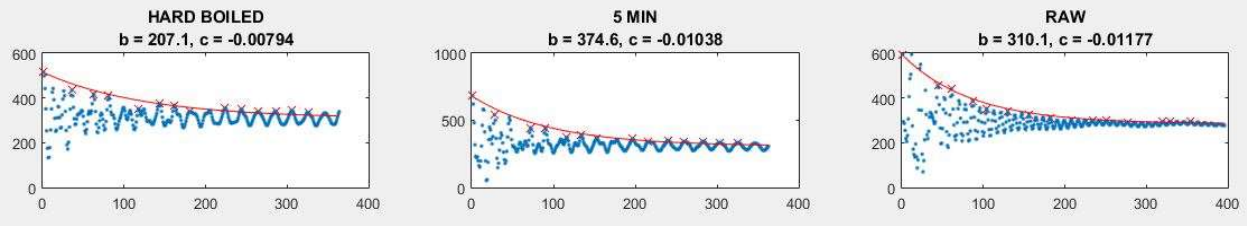



Figure 38 - Test 1

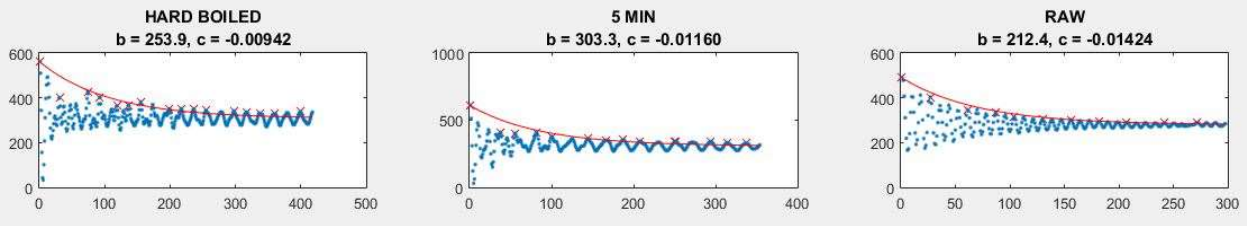



Figure 39 - Test 2

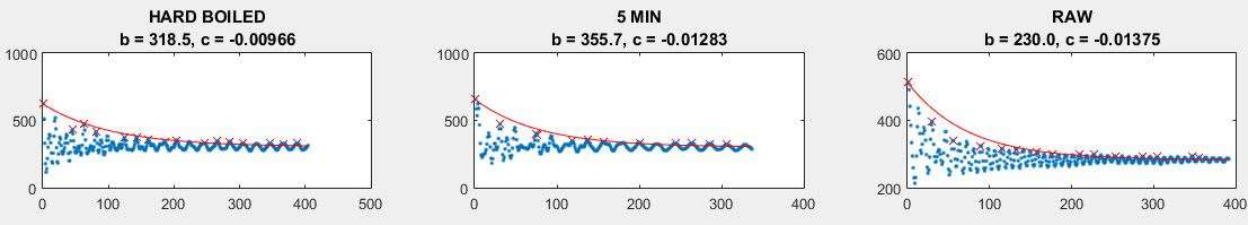



Figure 40 - Test 3

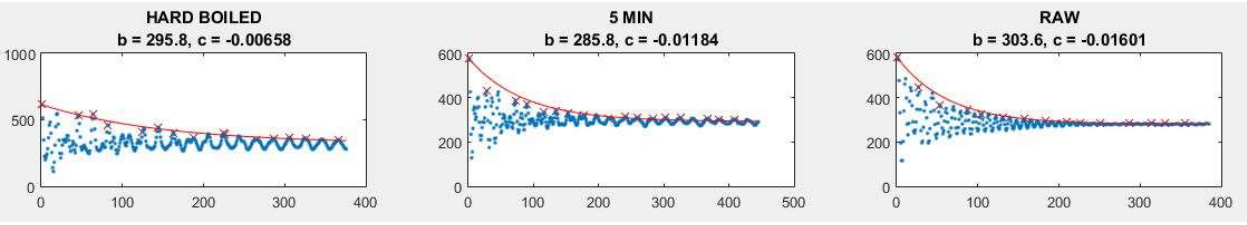



Figure 41 - Test 4

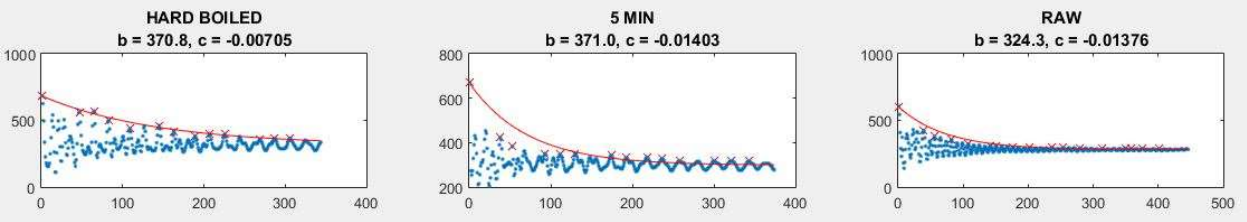



Figure 42 - Test 5

From the graphs above, there is a clear relationship between the three egg states. The average decay constant for each is:

$$c_{HARDBOILED} = -0.00813$$

$$c_{5MINBOIL} = -0.0121$$

$$c_{RAW} = -0.0139$$

Based on this data, there is a clear correlation between the three different eggs. However, we also notice that the constant can vary quite a bit. This is because of the mechanical vibrations and also because the decay is affected by the jerk motion of the user input. Thus, for the purpose of this project, we decided that the user has to test three eggs at a time. The egg with the largest decay is raw, the egg with the smallest decay is the hardboiled egg and the egg that has a decay in the middle will have a yolk consistency that is in between. The algorithm is not perfect, for example, in Test 5, it would have classified the raw egg as being the 5 min egg which is false. Because of this variability, we found that the repeatability of this device is approximately 80%.

# 4 Reflections

## 4.1 Reflections on the Mechanical and Electrical Design

Through this project, we learned many valuable lessons in hardware development. The integration of mechanical and electrical components is easy when one person is doing it, but in practice there are multiple people working on these design components. Being able to design mechanical components with electrical hardware in mind and vice versa is an integral benefit of being in mechatronics, as it reduces the amount of headache and back-and-forth that is usually required between the two divisions.

The main takeaway we found is that the mechanical design causes a heavy influence on the electronics and software. We thought it would be trivial to measure the spring oscillation and correlate it with yolk consistency, but the issue was that many other frequencies were being picked

up by the accelerometer (ie. jitter of the spring at rest). This made it difficult to reliably determine the yolk consistency since the measurements contained much more frequency information than just the effect of yolk consistency. In general, electrical and software systems cannot function effectively if the mechanical system is garbage. We also saw this in our MECH 421 controls course: if a mechanical system is not 100% rigid (ie. the shaft coupling for a motor), then even a well-tuned control system will only be able to react to instability and steady state error to a certain extent.

#### *4.1.1 What we learned from the mechanical/electrical component of this project*

1. How to design a PCB
2. How to integrate mechanical and electrical systems
3. How to use Windows Movie Maker

## 4.2 Reflections on the Software Algorithms

Initially, we tried to use the Arduino to calculate where the peaks are and also to perform the Fast Fourier Transform. However, due to the difficulty in debugging the data and the possibility that the Arduino cannot handle the vast number of calculations being performed, we decided to have MATLAB do the data analysis instead. However, since for the polished product we hope that it will be completely portable, it will be necessary to develop these algorithms in C and/or assembly to ensure that the performance is optimal.

In the MATLAB code, we initially tried to correlate the frequency of oscillation with the yolk consistency. However, it quickly became obvious that there is a significant amount of noise from the mechanical oscillations of the spring, the mechanical vibrations of the spring and the body and the variation in user input. As a result, the FFT showed that there are multiple peaks in the frequency histogram which made it difficult to determine which frequency was the correct one to determine the yolk consistency. Instead, we decided to use another method to correlate the yolk consistency. Here, we determined the exponential decay rate of the oscillations and used this to determine how cooked the egg was. This ended up working very well. However, since the user input can be different at times, it made result in a slightly different oscillation and a slightly different decay rate. Thus, for future iterations, it is necessary to use machine learning to properly correlate this data. This is because with machine learning, we can have all of the different parameters (frequency, egg size, input acceleration, output acceleration, egg mass) as inputs and use this to develop training data. Using this, we can correlate these parameters with the yolk consistency.

#### *4.2.1 What we learned from the software component of this project*

1. How to perform Fourier transforms to produce frequency histograms
2. Methods to fit an exponential curve to damped harmonic oscillations
3. How to interface Arduino to C# to MATLAB

## 4.3 What We Learned from MECH 423

The 3 most useful concepts that we learned were:

1. How to program embedded systems and read the associated datasheets
2. How to build a mechatronics device from scratch (the final project was extremely helpful and gave us learning experience in mechanical, electrical and software design)
3. How to totally over-engineer a simple everyday task

## 4.4 What We Would Like to Learn Going Forward

1. Preferably spend more time on a specific topic of circuit design, we found that the last few weeks of this class were too fast for us to fully understand all of the different concepts being taught
2. Maybe a bit more depth about how certain components in the microcontroller actually work. For example, how the ADC approximates continuous signals in the digital domain (successive approximation etc)
3. Refinement of mechanical design (ie. design for assembly and manufacturability)
4. Develop the electrical protoboard into a PCB with surface mount components to further minimize form factor