Design of Engineering Systems I

Rogowski Coil Precision Current Sensor

Team #2
Wei Loon Chim
Patrick Johnson
Haley Welch

Spring 2018

Table of Contents:

Requirements: **2**
- Rogowski coil requirements: 2
- Instrument requirements: 2
- Other: 2

Specifications: **3**
- Rogowski Coil: 3
- Filter System: 4
- Analog to Digital Conversion: 5
- Instrument: 5

Design: **6**
- Filter System: 6
- Instrument: 7
- Equipment Enclosure: 9

Performance: **9**
- Rogowski Coil: 9
- Filter System: 9
- Microcontroller: 10

Calibration and Results: **10**

Volume Production: **11**
- Instrument Cost: 11
- Coil Cost: 11
- Beta Cost: 11
- Production Run Cost: 11

Conclusion: **11**

Appendix A **13**

Appendix B **14**

Appendix C **15**

Appendix D **16**

Appendix E **17**

Appendix F **21**

I. Requirements:

A. Rogowski coil requirements:

- Measurement range: 10 mA to 10 A rms
- Accuracy: ±1% of true or better
- Bandwidth: -3 dB bandwidth, 30 Hz to at least 3 kHz (normalized to 60 Hz)
- Reference frequency: 60 Hz, all measurements
- Self-resonance: 30 kHz or higher
- Diameter: Outside diameter no greater than 4" (102 mm)
- Attachment: Capable of installation without disturbing power conductors
- Must be securely shut when in place
- Coils must be freely interchangeable, at least three per instrument
- Coils will contain internal ID code
- Calibration parameters, if needed, will be stored in the coil assembly

B. Instrument requirements:

- Battery type/number: To be selected by design team
- Power consumption: Can operate from a single USB 3.0 port
- Instrument capable of operating as USB-connected laptop accessory
- Instrument connections will define primary and secondary coil ID
- Instrument will calculate:
    - primary and secondary rms current
    - turns ratio based on current readings.
    - transformer operating efficiency from 0% to 100% load in 10% increments (resistive load)
- Maximum transformer load will be an input parameter
- Instrument will display primary and secondary rms current at all times
- Instrument will plot/display transformer turns ratio and efficiency curve on command
- Instrument will sample both coils simultaneously at 10 ksps for transient studies when initiated by remote trigger
- The instrument will store, graphically display, and transfer 5 seconds of transient data centered on the trigger to an external computer (file format .csv)

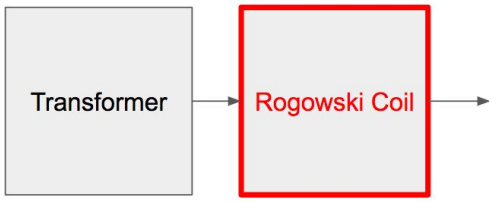
C. Other:

- Measure each coil for gain and phase performance (relative to 60 Hz) from 30 Hz to 50 kHz. This information must be available as a unique file stored in the instrument
- Each team will develop their own coil calibration apparatus and technique.
- Operating temperature range is room temperature (23±10 ^{0}C)
- Cost estimates must address
    - 10 instruments for beta test

- 100 instruments total production run
- 30 Rogowski coil assemblies for beta test
- 600 Rogowski coil assemblies for total production run

II. Specifications:

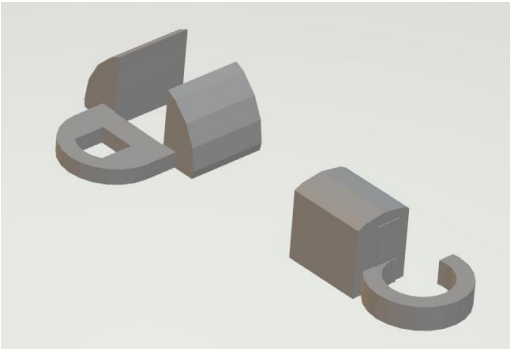
A. Rogowski Coil: Haley



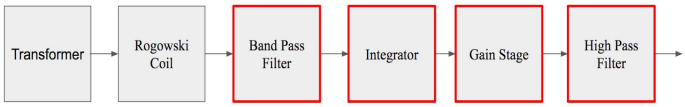
The Rogowski coils made use clear vinyl tubing to keep the shape, and magnet wire is wrapped around the tubing. The constants used for the coil calculations are $\mu_0 = 1.25664*10^{-6}$ H/m and wire diameter = $431.8*10^{-6}$ m. The equation for mutual inductance for a circular cross-section is $M_{21} = \frac{\mu_0 * N_2}{2}(a + b - 2\sqrt{ab})$ H where N_2 is the amount of turns in the coil, a is the inner radius of the coil, and b is the outer radius of the coil. The equation for self inductance for a circular cross-section is $L_2 = \frac{\mu_0 * N_2^2}{2}(a + b - 2\sqrt{ab})$ H where all of the variables are the same as before. The equation for the signal at I current and f frequency is $V = (2\pi f M_{21} I)$ volts.

- The first coil made has 480 turns, an outer radius of $38.1*10^{-3}$ m, an inner radius of $31.75*10^{-3}$ m, a mutual inductance of $87.231*10^{-9}$ H, a self inductance of $41.871*10^{-6}$ H, and a signal of $328.854*10^{-6}$ V at 10A and 60Hz. This coil was not used as one of the final three coils because the signal is too low, and the wire around the coil is too loose. The diameter of the tubing used in the center of the coil was doubled for the good coils.
- The first good coil made has 504 turns, an outer radius of $50.8*10^{-3}$ m, an inner radius of $38.1*10^{-3}$ m, a mutual inductance of $288.748*10^{-9}$ H, a self inductance of $145.528*10^{-6}$ H, and a signal of $1.089*10^{-3}$ V at 10A and 60Hz.
- The second good coil made has 468 turns, an outer radius of $50.8*10^{-3}$ m, an inner radius of $38.1*10^{-3}$ m, a mutual inductance of $268.123*10^{-9}$ H, a self inductance of $125.482*10^{-6}$ H, and a signal of $1.012*10^{-3}$ V at 10A and 60Hz.
- The third good coil made has 478 turns, an outer radius of $50.8*10^{-3}$ m, an inner radius of $38.1*10^{-3}$ m, a mutual inductance of $273.852*10^{-9}$ H, a self inductance of $130.901*10^{-6}$ H, and a signal of $1.032*10^{-3}$ V at 10A and 60Hz.

The coils had 3D printed hooks and loops glued into the tubing to keep the coils securely shut when in place. Three increasingly better designs were used for this. The first design was used to see if the basic idea was plausible. The next two designs each added more points of contact for security. The final design is shown below.



B. Filter System: Patrick



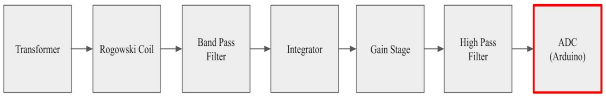
As seen in the diagram, the filter will be receiving a differential term of the current carried by the wire, scaled by some M value supplied by the parameters of the coil. The job of the filter is four-fold.

1. Band pass filter to eliminate DC signal noise as well as high frequency noise created through the self inductance of the coil. For this stage the cut off frequencies selected were 30 and 4000 HZ to meet the specifications supplied to us. The gain at this stage was selected to be 3.25 V/V or 10.23 dB.
2. An integrator to turn the differential current term into a regular term for current. This can be modeled as a low pass filter to set the gain at lower frequencies to be more reasonable. The cut off frequency was selected to be at 32.88 HZ. The gain of the low pass filter was selected to be 10 V/V or 20 dB. The integration takes place at the falling edge of the bode plot, so our integrable range was estimated to be between 40-450 HZ. Because 60 HZ was the calibration frequency, the gain through this stage was less than the expected 20 dB, but this was taken care of in the calibration process.
3. A gain stage to ensure the output signal has a high enough amplitude to be accurately sampled by the analog to digital converter. The trick here is selecting a high enough gain to be accurate through the ADC, but not too much to go over the VREF of 5V given by the ADC. We treated these numbers as calibration constants more than theoretical values, and therefore they were obtained through empirical studies. The gain for the first channel was selected to be 151 V/V or 43.52 dB. The gain for the second channel was selected to be 69.18 V/V or 36.67 dB.
4. A passive high pass filter on the output of the system to ensure the removal of any DC offset or noise going into the analog to digital converter. Ast his stage is passive, it does not provide any gain to the overall system and will not be included in the gain calculations. This stage came about as a result of experimentation in order to fix the DC offset seen by the ADC.

The total gain of this system for the first channel at 30 HZ was 4,875 V/V or 73.76 dB. The total gain of this system for the

second channel at 30 HZ was 2215.9 V/V or 66.91 dB. The operational amplifier selected for this design was the single channel LM-741 chip which had a gain-bandwidth product of 1 MHZ, but was cascaded through four chips, so the GBP was increased to allow for the gain we needed over the entire range of 400 HZ.

C. Analog to Digital Conversion: Patrick



The Arduino Uno used to collect and display the data has an internal analog to digital converter with a resolution of 10 bits. The reference voltage supplied by the Arduino is 5V, giving us a bit resolution of 4.88 mV per bit. Because the specifications stated the instrument must measure the current within 1% accuracy, the minimum required voltage coming from the filter system is 488 mV. The expected minimum voltage coming from the 1.098*10^-6 V, meaning the minimum amount of gain required for accuracy is 444,444.4 V/V. Because this is too much for our op amps to handle, the minimum measurable current was shifted from 10 mA to 1 A. This brings the minimum required gain down to 4,444.4 V/V which is below our gain stage amount. Raising the minimum required measurable current also means the necessity of a variable gain stage is deprecated, as the entire spectrum of possible frequencies will be measurable through the system.

D. Instrument: Wei Loon

Arduino UNO R3 is an open-source microcontroller board based on the ATmega328P microcontroller and developed by Arduino.cc. This board is equipped with sets of digital and analog input/output pins that can be interfaced to various expansion boards and other circuits. It is programmable with the Arduino Integrated Development Environment via a type-B USB cable. It can be powered by a USB cable or by an external 9 volt battery, though it accepts voltages between 7 and 20 volts. The ATmega328 on the Arduino UNO R3 comes preprogrammed with a bootloader that allows to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol.

With the calibration constant determined, both primary current and secondary current can be determined by

$$I_{primary,f} = c_{1,f} V_{primary,f} + c_{2,f}$$

$$I_{secondary,f} = c_{3,f} V_{secondary,f} + c_{4,f}$$

Where f is frequency, $c_{1,f}$ and $c_{2,f}$ are the determined calibration constants at different frequency for primary coil calculation, and $c_{3,f}$ and $c_{4,f}$ are the determined calibration constants at different frequency for secondary coil calculation.

To determine the maximum current of both primary and secondary, the instrument is capable to capture 5,000 samples and pick

the largest value among these 5,000 samples before show any value on the LCD display.

The turns ratio can be calculated with the following formula:

$$Turns\ Ratio = \frac{I_{max,\ secondary}}{I_{max,\ primary}}$$

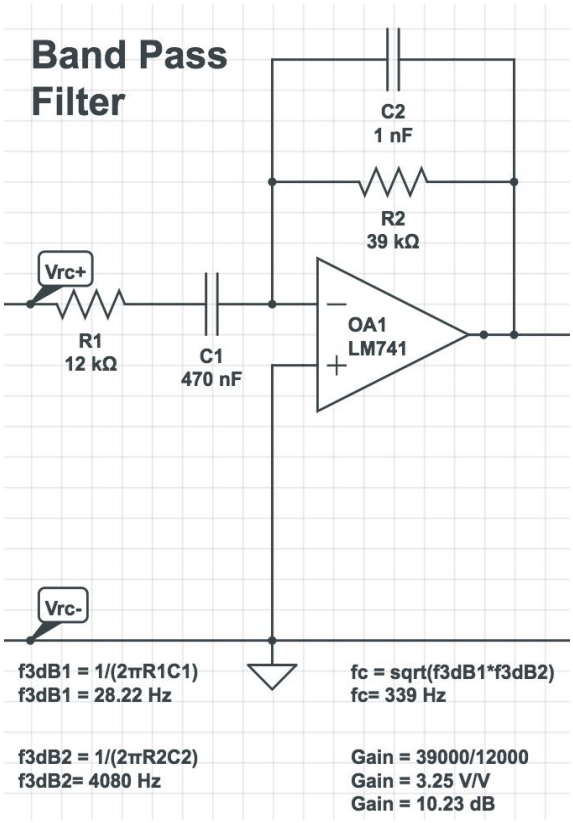
The efficiency can be calculated with the following formula:

$$Efficiency(\%) = \frac{R_{load}}{V_{primary}} \times \frac{I^2_{secondary}}{I_{primary}} \times 100\%$$

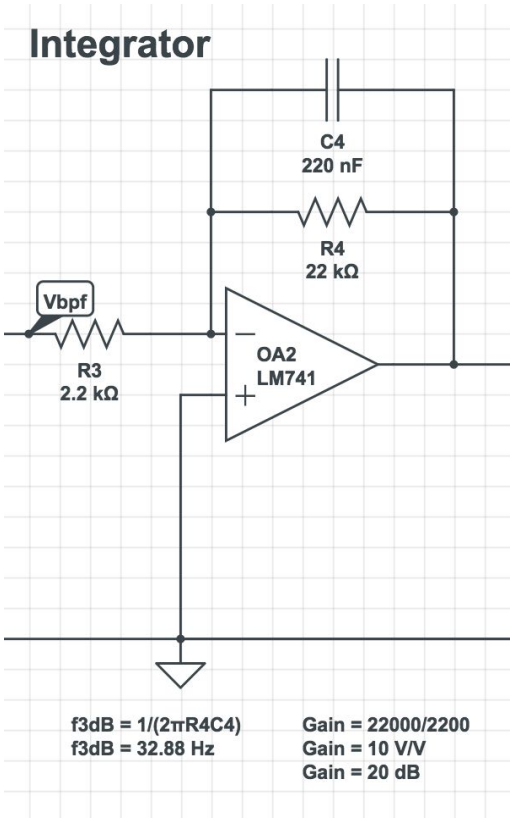
III. Design:

A. Filter System: Patrick

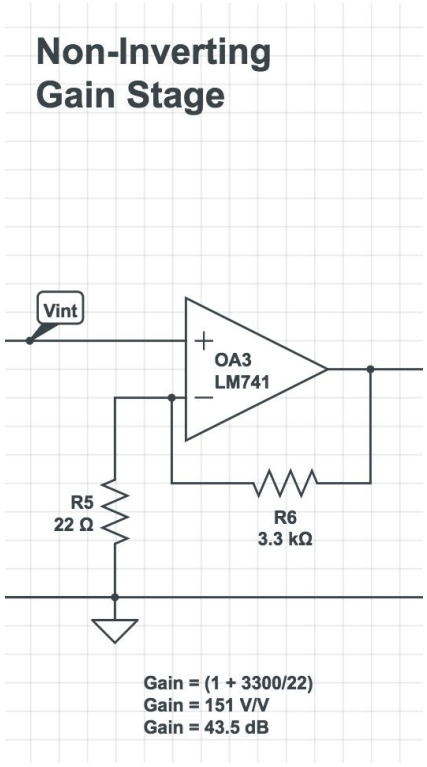
1.



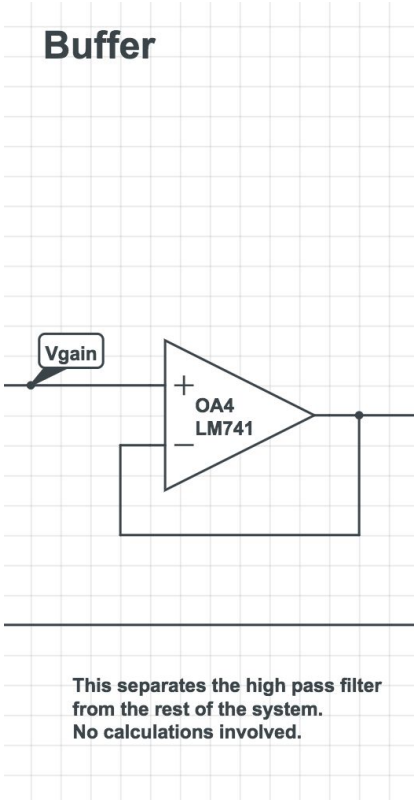
2.



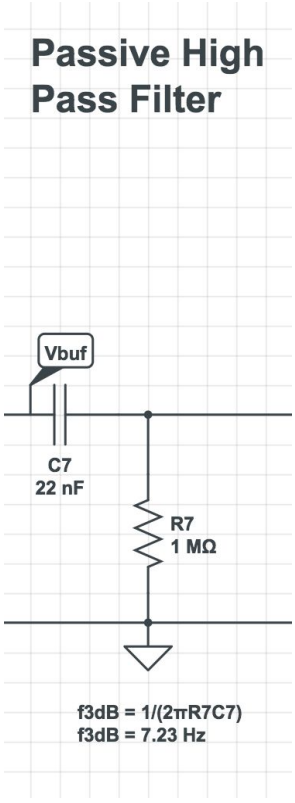
3.



4.



5.



For full system schematic, see appendix F.

B. Instrument: Wei Loon

The Arduino UNO R3 can interface with 16 by 2 LCD display with the following circuit setup in schematic form:

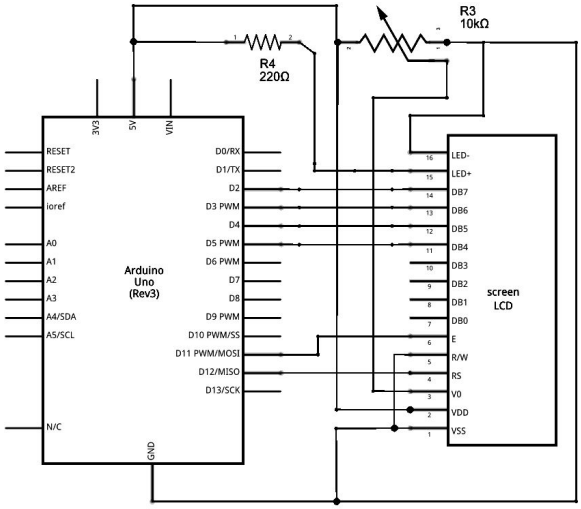


Figure 1 shows schematic of Arduino UNO R3 with 16 by 2 LCD display

The code along with its description, see Appendix A.

For full codes for this project, see Appendix E.

The Arduino UNO R3 can read analog value from an analog signal circuit and convert it into digital reading with the built-in Analog to Digital Converter (ADC). The ADC has a bit resolution of 4.9 mV per bit, as it is a 10-bit ADC, and the reading range is from 0 to 5 V. With that say, the Arduino UNO R3 can read analog signal from instrumental amplifier and integrator circuit, and with conversion and current calculation, the maximum current value at certain period can be determined.

The circuit setup in schematic form is shown below:

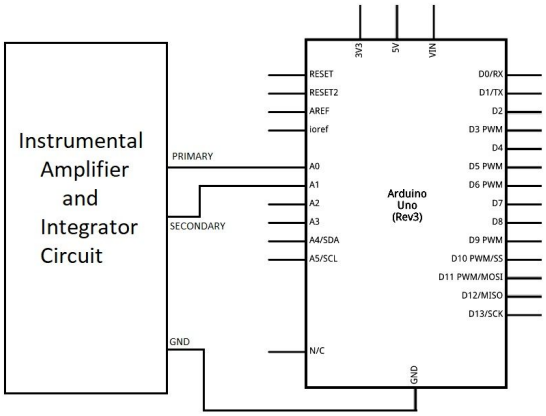


Figure 2 shows schematic of Arduino UNO R3 with instrumental amplifier and integrator circuit

The code along with its description, see Appendix B.

For full codes for this project, see Appendix E.

To control the readings shown on display, a circuit with buttons is required without re-upload code with different variables everytime. In this project, three buttons are used to change calibration constants and frequency, change screen content, and start recording value and save it into file in Excel format.

Each button performed the following actions:

- Calibration constants and frequency
    - When frequency of the input signal is given, toggle the button and change the frequency mode between 50 Hz, 60 Hz, and 400 Hz.
    - With the given frequency mode, the calibration constants will also change accordingly with the frequency mode, so that the calculation would be accurate.
- Switching screen content
    - When the Arduino UNO R3 boot up, the LCD display would show the primary and secondary current in ampere.
    - When the button is triggered, the LCD display would show coil turn ratio and efficiency in percentage. When the button is triggered again, the LCD display would show the primary and secondary current in ampere again.
- Start recording value and save it into file in Excel format
    - When the Arduino UNO R3 boot up, recording button is disabled until any value shows on the LCD screen.
    - When the button is triggered at the first time, the Arduino UNO R3 will send an instruction to a program call

gobetwino, (see details in Appendix C) which will open an existing format Excel file and ready to write value on it.

- When the button is triggered again, the Arduino UNO R3 will send analog values with calculation result, along with other constant (such as input voltage, load resistor) to the Excel file and save it.
- The circuits is shown below:

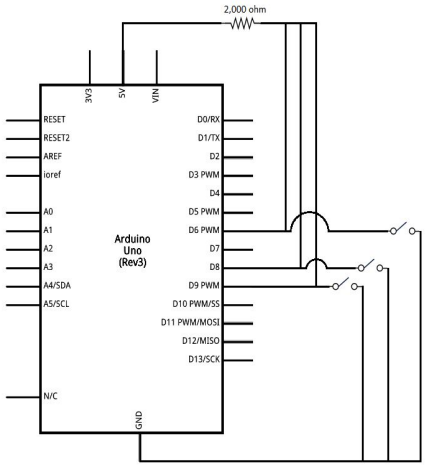
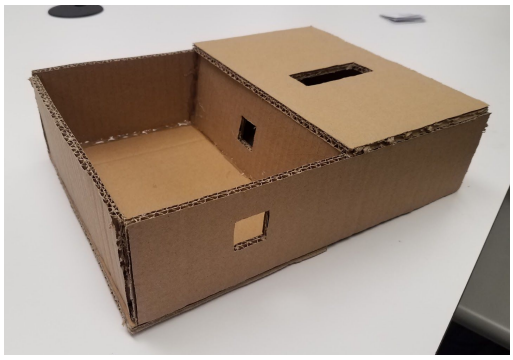


Figure 3 shows schematic of Arduino UNO R3 with buttons circuit

- The code along with its descriptions, see Appendix D.
- For full codes for this project, see Appendix E.

C. Equipment Enclosure: Haley

An enclosure was created to house the equipment. It was made of cardboard and hot glue. Its dimensions are 9in x 12in x 3in. A picture of this is shown below. The rectangular cutout on the top is for the LCD display. The two square cutouts on the sides are for the cord and wires to fit through. This enclosure helps keep everything together.



IV. Performance:

A. Rogowski Coil: Haley

Coil was constructed to be within 4" diameter and had a self resonance of higher than 30 kHz. Coils had their own ID code given by the color of hooks attached on the end to indicate different parameters. Calibration parameters were stored in the microcontroller, not the assembly. Coil remained securely fastened throughout the duration of the test.

B. Filter System: Patrick

Had a experimentally obtained bandwidth of 33-4020 Hz, only a small amount off of the theoretical bandwidth in the schematic. This could be attributed to many factors, most notably the hardware tolerances. DC noise

was almost completely attenuated through the use of the final high pass filter stage. Two 9 V batteries were used to obtain ±9 V as the rails for the system. Output voltages ranged from 0 V to 3.83 V based on an input of 500 mA to 8 A to be fed into the microcontroller.

C. Microcontroller: Wei Loon

Able to be powered by 8 AA batteries held together by a battery pack. Power consumption did not exceed USB 3.0 interface. Instrument was able to calculate primary and secondary rms current and be within 5% true accuracy for a range of 500mA - 8A. Instrument was able to display turns ratio and efficiency on command by way of pressing an external button to switch frames on the LCD module. Instrument was able to sample at the required 10ksps by way of scheduled interrupts through the adc. Instrument was able to write data (efficiency or transient) to excel file upon trigger via an external button and plot the data as well.

V. Calibration and Results: Haley

In order to take precise measurements, each coil had to be calibrated due to the slight differences between the coils. First, each coil's readings at different voltages and frequencies were compared to the actual currents at those voltages and frequencies by plotting them. This created nine graphs with the actual currents at one frequency along the x axis and the measured values along the y axis. Then the trendline for each graph was found. These equations were programmed in so that the correct equation is used for each coil for each frequency. For each equation, x is the raw reading, and y is the calibrated reading. The equations are below.

Coil 1:

Frequency (Hz)	Calibration Equation
50	y = 2.289x + 0.4144
60	y = 2.0201x + 0.4078
400	y = 1.4094x + 0.4491

Coil 2:

Frequency (Hz)	Calibration Equation
50	y = 4.4448x + 0.7846
60	y = 4.0824x + 0.6871
400	y = 2.8814x + 0.733

Coil 3:

Frequency (Hz)	Calibration Equation
50	y = 3.4108x + 0.6616
60	y = 3.046x + 0.6262
400	y = 2.173x + 0.5586

After this calibration, the competition was completed. A table showing the results is below.

Reference (A)	Reading (A)	Error (%)
6.65	6.74	1.35
3.57	3.67	2.80
5.45	5.66	3.85
2.67	2.7	1.12
4.18	4.31	3.11
1.844	1.94	5.21

VI. Volume Production: Patrick

A. Instrument Cost:

1x Arduino Uno - $22

1x USB cable - $3

1x 16 by 2 LCD Module - $10

4x Switch Buttons - $0.6

1x 10k Potentiometer - $1.11

10x LM741 Op-Amp - $6.02

6x Mini Breadboards - $10

2x 9v Battery - $5

8x AAA Battery - $3.20

Cardboard - $1.25

Wires - $2

Total : $64.18

B. Coil Cost:

Magnet Wire - $3

Tubing - $1.50

Plastic Hooks - $0.35

Super Glue - $0.20

Total - $5.05

C. Beta Cost:

10x Instruments - $641.80

30x Coils - $151.50

Total - $793.30

D. Production Run Cost:

100x Instruments - $6418.00

30x Coils - $3030.00

Total - $9448.00

VII. Conclusion:

The coil worked better than we had anticipated at outputting the expected voltages for the instrument to measure. However, we encountered a large amount of pesky DC noise that would not seem to go away. If we had to do the project again, an instrumentation amplifier would be a great way to get rid of the common mode noise in the system. Unfortunately in the beginning the wrong instrumentation amplifier was ordered and that was no longer an option. This made things difficult when it came time to send the signal over to the arduino's built

in ADC because it was picking up the DC offset that we could not see. Another thing that went well was connecting the common grounds, since nothing was destroyed in the process. The microcontroller was a huge success for us, but if we had to do the project again we would be more inclined to use a raspberry pi for the development kit as it has more features that can be utilized by the client. The arduino did everything we needed it to, but the ability to remotely graph data on a touch screen LCD module is very enticing.

The cost calculations were made simply with our raw design and no other enhancements made outside. Some tweaks that would bring down the cost would be to use printed circuit boards as they are much cheaper than mini breadboards, as well as smaller op amp chips. Nothing can really be done about the biggest costs in the system, the arduino and LCD module. However, reducing the costs of the other components will greatly affect the production run cost.

Appendix A

This appendix describes the Arduino UNO R3 with 16 by 2 LCD display. It requires a $10\ k\Omega$ potentiometer and Arduino UNO R3 to complete the setup.

Here are the pin description of LCD display

- D4 to D7 pins
  - Control content shows on 16 by 2 LCD display.
- Enable pin
  - Enable features on LCD display, so that the display performed action (such as read and write).
- V_{SS} and V_{DD} pins
  - Power source for LCD display
- V0 pin
  - Connect with $10\ k\Omega$ potentiometer for adjusting contrast of the LCD screen.
- RS pin
  - Select registers between instruction register and data register
- R/W pin
  - Read or write content from or to LCD display, must be work with enable pin.
- LED+/LED-
  - Power for backlit screen.

Note: The LCD screen must be compatible with Hitachi HD44780 controller.

Here is some of the code for LCD

```
#include <LiquidCrystal.h>
const int rs = 12, en = 11, d4 = 5, d5 = 4,
d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5 , d6 , d7);
…

void setup() {
  lcd.begin(16, 2);
}

void loop() {
  ...
}
...
```

Appendix B

This appendix describes how the Arduino UNO R3 handle analog signal from instrumental amplifier and integrator circuit.

Arduino UNO R3 features 6 channel 10 bit ADC, however, in this project, only 2 channels are necessary.

Here are the pin required on Arduino UNO R3:

- A0 pin
  - The first analog pin for reading primary coil voltage, after process through Arduino UNO R3 and calculation, the maximum current on primary coil can be determined.
- A1 pin
  - The second analog pin for reading secondary coil voltage, after process through Arduino UNO R3 and calculation, the maximum current on secondary coil can be determined.
- GND pin
  - Connect the analog circuit ground with Arduino UNO R3 ground, so that they have common ground.

Here is some of the code for analog reading

```
…
int analogValueZero = 0;
int analogValueOne = 0;
…

void loop() {
  …
  calculation();
  ...
}

void calculation() {
  …
  analogValueZero =
analogRead(pinAnalogZeroOutputSignal);
  analogValueOne =
analogRead(pinAnalogOneOutputSignal);
  …
}
```

Appendix C

This appendix describe an external program *gobetwino*.

This program is kind of a “generic proxy” for Arduino, which only run on Windows operating system. It will act on behalf of Arduino and do some of the things that Arduino can’t do on its own.

It can listen on the serial port, for commands coming from Arduino, and in response it will perform action for Arduino and possibly return action to Arduino.

It defines a set of command types that can be used as templates to create actual commands. Arduino can as *gobetwino* to execute these commands, and return something to Arduino.

Here is the list what *gobetwino* can do in this project:

- Start Excel program on the PC.
- Send data to Excel from Arduino, like it was typed on the keyboard.

Here is some of the code where *gobetwino* takes place:

```
void efficiencyTableButton() {
  /* Start Excel for efficiency table */
  if(!flag) {
    Serial.println("#S|SPTXT|[]#");
    delay(1000);
    flag = true;
  }
  else {
    char buffer[15];

    Serial.print("#S|SENDK|[");
    Serial.print(itoa((pID), buffer, 10));
    Serial.print("&");
    Serial.print(itoa((trial + 1), buffer, 10));
    Serial.print(" {TAB} ");
    Serial.print(dtostrf((voltageP), 1, 3,
buffer));
    Serial.print(" {TAB} ");
    Serial.print(dtostrf((loadR[trial]), 1, 3,
buffer));
    Serial.print(" {TAB} ");
    Serial.print(dtostrf((currentP), 1, 3,
buffer));
    Serial.print(" {TAB} ");
    Serial.print(dtostrf((currentS), 1, 3,
buffer));
    ...
```

Appendix D

This appendix describe the button circuits. It requires resistor and button with Arduino UNO R3 to complete the setup.

Here are the pins required from Arduino UNO R3:

- Digital pin 6
  - This pin is set to control frequency mode with associated calibration constants when the button is triggered.
- Digital pin 8
  - This pin is set to control the screen content when the button is triggered.
- Digital pin 9
  - This pin is set to control the command is sent to program *gobetwino* and perform data record and save.

Here is some of the code of button circuit

```
…
if(digitalRead(pinEfficiencyTable) ==
LOW) {
   lcd.clear();
   lcd.setCursor(2, 0);
   lcd.print("Write data to");
   lcd.setCursor(5, 1);
   lcd.print("table");
   delay(500);
   lcd.clear();
   efficiencyTableButton();
  }
...
if(digitalRead(pinSwitchFrequency) ==
LOW) {
   delay(50); // software debounce
   frequency = (frequency + 1) % 3;

   lcd.clear();
   lcd.setCursor(3, 0);
   lcd.print("Frequency");
   lcd.setCursor(5, 1);
   if(frequency == 0) {
    lcd.print("50 Hz");
   }
...
```

Appendix E

This appendix shows all the source code for Arduino UNO R3. Comments are attached on necessary line for better understanding of each function.

```
#include <LiquidCrystal.h>

const int rs = 12, en = 11, d4 = 5, d5 = 4,
d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5 , d6 , d7);

const int pinEfficiencyTable = 6; // write
all data to file in table form
const int pinDataRecordButton = 7; // start
or stop record transient data, directly write
to file
const int pinSwitchFrame = 8; // show
between turn ratio/efficiency and currents
const int pinSwitchFrequency = 9; //
toggle three constants between frequency
50Hz, 60Hz, and 400Hz
const int pinAnalogZeroOutputSignal = 0;
// A0
const int pinAnalogOneOutputSignal = 1;
// A1
const int arrayLength = 128;
const int pID = 0;
const float voltageP = 120.0;
const float loadR[11] = {5.0, 2.5, 1.667,
1.25, 1.0, 0.8333, 0.714, 0.625, 0.556, 0.5,
0.333};
const float constFrequencyP1[3] = {2.289,
2.0201, 1.4094};
const float constFrequencyP2[3] =
{0.4144, 0.4078, 0.4491};
const float constFrequencyS1[3] =
{4.4448, 4.0824, 2.8814};
const float constFrequencyS2[3] =
{0.7846, 0.6871, 0.733};
```

```
float currentP = 0.0;
float currentS = 0.0;
float turnsRatio = 0.0;
float efficiency = 0.0;
float dataCurrent[arrayLength];
int analogValueZero = 0;
int analogValueOne = 0;
int frequency = 0;
int frame = 0;
int trial = 0;
bool pushed = false;
bool flag = false;
bool flag2 = false;

void setup() {
  /* 57600 Serial for data transfer purpose
*/
  Serial.begin(57600);

  /* Initialize necessary I/O */
  pinMode(pinEfficiencyTable, INPUT);
  pinMode(pinDataRecordButton,
INPUT);
  pinMode(pinSwitchFrame, INPUT);
  lcd.begin(16, 2);

  /* Initialize array */
  for(int i = 0; i < arrayLength; i++) {
    dataCurrent[i] = 0.0;
  }
}

void loop() {
  calculation();

  if(digitalRead(pinEfficiencyTable) ==
LOW) {
    lcd.clear();
    lcd.setCursor(2, 0);
    lcd.print("Write data to");
    lcd.setCursor(5, 1);
    lcd.print("table");
    delay(500);
```

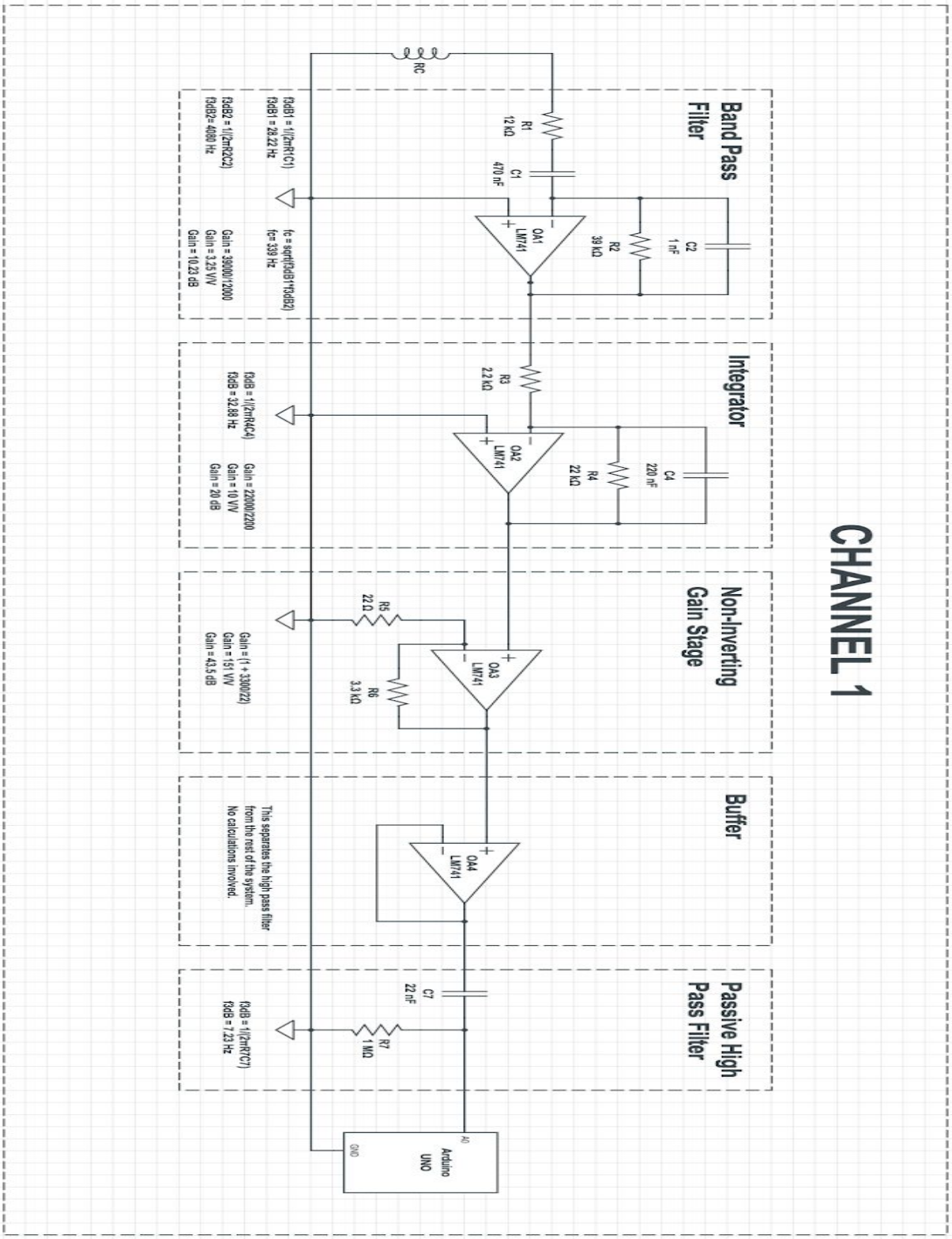
```
    lcd.clear();
    efficiencyTableButton();
  }

  // Start record once button is pushed, and
stop record once button is pushed again
  if(digitalRead(pinDataRecordButton) ==
LOW) {
    delay(10); //software debounce
    pushed = !pushed;
    lcd.clear();
    lcd.setCursor(5, 0);
    lcd.print("Record");
    delay(500);
    recordButton();
  }

  // Switch display from current to ratio &
eff
  if(frame == 0) {
    frameOne();
    if(digitalRead(pinSwitchFrame) ==
LOW) {
      delay(10); // software debounce
      frame = 1;
      lcd.clear();
    }
  }
  else if(frame == 1) {
    frameTwo();
    if(digitalRead(pinSwitchFrame) ==
LOW) {
      delay(10); // software debounce
      frame = 0;
      lcd.clear();
    }
  }

  // Toggle constant associated with
respective frequency
  if(digitalRead(pinSwitchFrequency) ==
LOW) {
    delay(50); // software debounce
    frequency = (frequency + 1) % 3;
    lcd.clear();
    lcd.setCursor(3, 0);
    lcd.print("Frequency");
    lcd.setCursor(5, 1);
    if(frequency == 0) {
     lcd.print("50 Hz");
    }
    else if(frequency == 1) {
      lcd.print("60 Hz");
    }
    else {
      lcd.print("400 Hz");
    }

    delay(500);
    lcd.clear();
  }
}

void frameOne() {
  // Display primary and secondary current
  lcd.setCursor(0, 0);
  lcd.print("Primary: ");
  lcd.print(currentP);
  lcd.setCursor(0, 1);
  lcd.print("Secondary: ");
  lcd.print(currentS);
  delay(250);
}

void frameTwo() {
    // Display turn ratio and efficiency
    lcd.setCursor(0, 0);
    lcd.print("Ratio: ");
    lcd.print(turnsRatio);
    lcd.setCursor(0, 1);
    lcd.print("Efficient: ");
    lcd.print(efficiency);
    lcd.print("%");
    delay(250);
}

void efficiencyTableButton() {
```

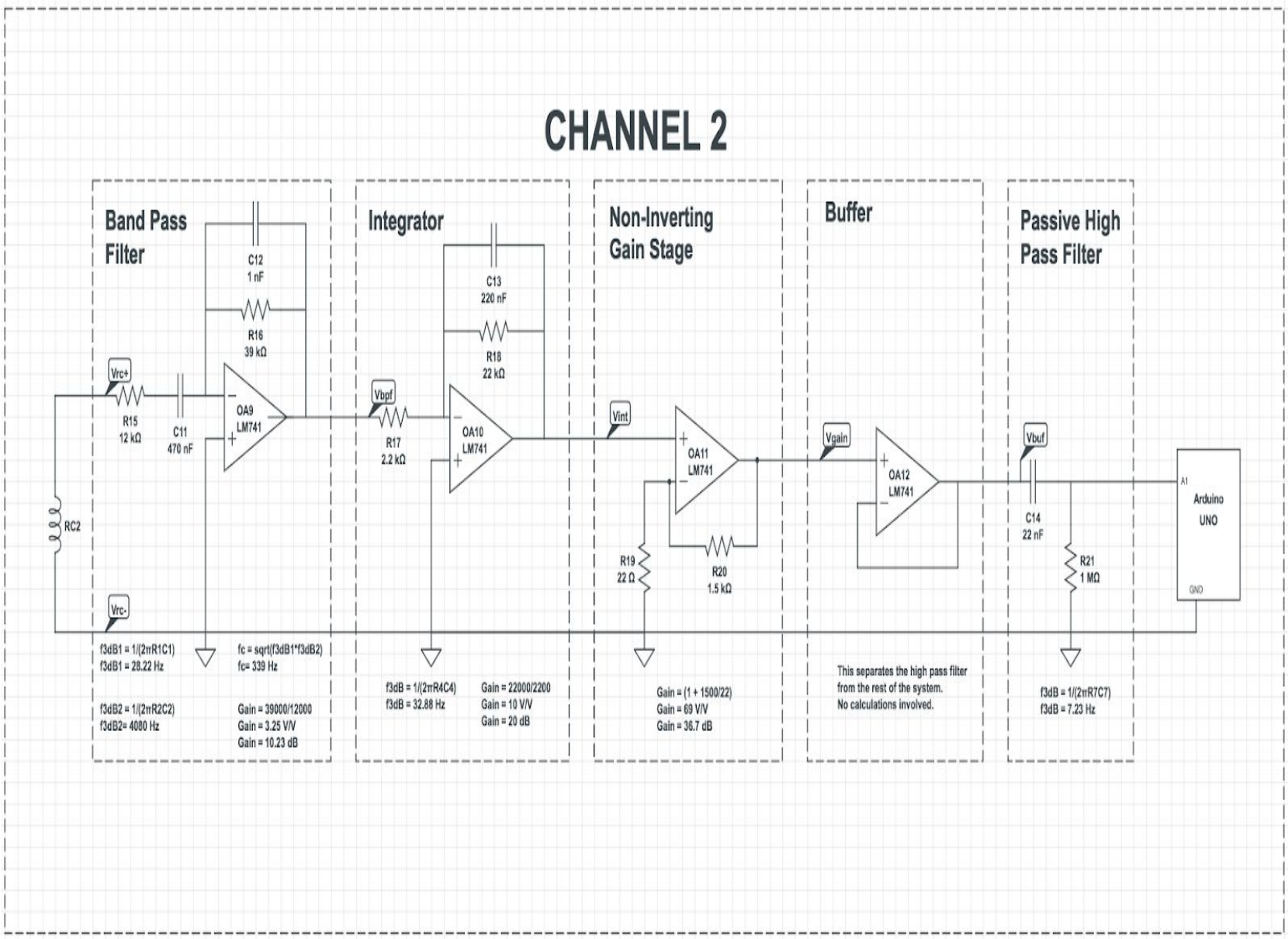
```
  /* Start Excel for efficiency table */
  if(!flag) {
    Serial.println("#S|SPTXT|[]#");
    delay(1000);
    flag = true;
  }
  else {
    char buffer[15];

    Serial.print("#S|SENDK|[");
    Serial.print(itoa((pID), buffer, 10));
    Serial.print("&");
    Serial.print(itoa((trial + 1), buffer, 10));
    Serial.print(" {TAB} ");
    Serial.print(dtostrf((voltageP), 1, 3,
buffer));
    Serial.print(" {TAB} ");
    Serial.print(dtostrf((loadR[trial]), 1, 3,
buffer));
    Serial.print(" {TAB} ");
    Serial.print(dtostrf((currentP), 1, 3,
buffer));
    Serial.print(" {TAB} ");
    Serial.print(dtostrf((currentS), 1, 3,
buffer));
    Serial.print(" {TAB} ");
    Serial.print(dtostrf((efficiency), 1, 3,
buffer));
    Serial.print(" {DOWN} ");
    Serial.print(" {LEFT}  {LEFT}
{LEFT}  {LEFT}  {LEFT} ");
    Serial.println("]#");
    delay(100);

    Serial.print("#S|SENDK|[");
    Serial.print(itoa((pID), buffer, 10));
    Serial.print("& ");
    Serial.print("%Fs");
    Serial.println("]#");
    delay(750);

    trial = trial + 1;
  }
}
```

```
void recordButton() {
  if(!flag2) {
    /* Start Excel for transcient data */
    Serial.println("#S|SPXL|[]#");
    delay(1000);
    flag2 = true;
  }
  else {
    char buffer[32];

    for(int i = 0; i < arrayLength; i++) {
      dataCurrent[i] =
(analogRead(pinAnalogOneOutputSignal)
* (5.0 / 1023.0)) *
constFrequencyS1[frequency] +
constFrequencyS2[frequency];
    }

    for(int i = 0; i < arrayLength; i++) {
      Serial.print("#S|SENDK|[");
      Serial.print(itoa((pID), buffer, 10));
      Serial.print("&");
      Serial.print(itoa((i), buffer, 10));
      Serial.print(" {TAB} ");
      Serial.print(dtostrf((dataCurrent[i]), 1,
3, buffer));
      Serial.print(" {DOWN} ");
      Serial.print(" {LEFT} ");
      Serial.println("]#");
      delay(10);
    }

    Serial.print("#S|SENDK|[");
    Serial.print(itoa((pID), buffer, 10));
    Serial.print("& ");
    Serial.print("%Fs");
    Serial.println("]#");
    delay(750);
  }
}

void calculation() {
```

```
  /* First sample 5000 of voltage value for
primary and secondary,
     then compare with previous value to
determine maximum value. */

  int count = 5000;
  float maximumCurrentZero = 0.0;
  float maximumCurrentOne = 0.0;

  while(count > 0) {
    /* Read data from circuit */
    analogValueZero =
analogRead(pinAnalogZeroOutputSignal);
    analogValueOne =
analogRead(pinAnalogOneOutputSignal);
    float sampleZero =
constFrequencyP1[frequency] *
(analogValueZero * (5.0 / 1023.0)) +
constFrequencyP2[frequency];
    float sampleOne =
constFrequencyS1[frequency] *
(analogValueOne * (5.0 / 1023.0)) +
constFrequencyS2[frequency];

    /* Compare current maximum value
with current analog value */
    if(sampleZero > maximumCurrentZero)
{
      maximumCurrentZero = sampleZero;
    }

    if(sampleOne > maximumCurrentOne)
{
      maximumCurrentOne = sampleOne;
    }

    /* Reduce count by 1 */
    count = count - 1;
  }

  /* Record the maximum current value
and reset count value */
  currentP = maximumCurrentZero;
  currentS = maximumCurrentOne;

  /* Calculate efficiency */
  efficiency = ((loadR[trial] / voltageP) *
((currentS * currentS) / currentP)) * 100;

  /* Calculate turn ratio */
  turnsRatio = currentS / currentP;
}
```

Appendix F


CHANNEL 1
Band Pass Filter
Integrator
Non-Inverting Gain Stage
Buffer
Passive High Pass Filter
RC
R1
12 kΩ
C1
470 nF
C2
1 nF
R2
39 kΩ
OA1
LM741
f3dB1 = 1/(2πR1C1)
f3dB1 = 28.22 Hz
f3dB2 = 1/(2πR2C2)
f3dB2= 4080 Hz
fc = sqrt(f3dB1*f3dB2)
fc= 339 Hz
Gain = 39000/12000
Gain = 3.25 V/V
Gain = 10.23 dB
R3
2.2 kΩ
C4
220 nF
R4
22 kΩ
OA2
LM741
f3dB = 1/(2πR4C4)
f3dB = 32.88 Hz
Gain = 22000/2200
Gain = 10 V/V
Gain = 20 dB
R5
22 Ω
R6
3.3 kΩ
OA3
LM741
Gain = (1 + 3300/22)
Gain = 151 V/V
Gain = 43.5 dB
OA4
LM741
This separates the high pass filter from the rest of the system.
No calculations involved.
C7
22 nF
R7
1 MΩ
f3dB = 1/(2πR7C7)
f3dB = 7.23 Hz
A0
Arduino
UNO
GND


CHANNEL 2
Band Pass Filter
C12
1 nF
R16
39 kΩ
Vrc+
R15
12 kΩ
C11
470 nF
OA9
LM741
RC2
Vrc-
f3dB1 = 1/(2πR1C1)
f3dB1 = 28.22 Hz
f3dB2 = 1/(2πR2C2)
f3dB2= 4080 Hz
fc = sqrt(f3dB1*f3dB2)
fc= 339 Hz
Gain = 39000/12000
Gain = 3.25 V/V
Gain = 10.23 dB
Integrator
C13
220 nF
R18
22 kΩ
Vbpf
R17
2.2 kΩ
OA10
LM741
f3dB = 1/(2πR4C4)
f3dB = 32.88 Hz
Gain = 22000/2200
Gain = 10 V/V
Gain = 20 dB
Non-Inverting Gain Stage
Vint
OA11
LM741
R19
22 Ω
R20
1.5 kΩ
Gain = (1 + 1500/22)
Gain = 69 V/V
Gain = 36.7 dB
Buffer
Vgain
OA12
LM741
This separates the high pass filter
from the rest of the system.
No calculations involved.
Passive High Pass Filter
Vbuf
C14
22 nF
R21
1 MΩ
f3dB = 1/(2πR7C7)
f3dB = 7.23 Hz
A1
Arduino
UNO
GND