WATT BALANCE

By

Julian O’Hern
Justin Ansell
John Howard

Final Report for ECE 445, Senior Design, Spring 2024
TA: Abhisheka Sekar

1 May 2024
Project No. 76

Abstract

Our senior design project focused on the development of a Kibble balance, also known as a watt balance. This is an instrument capable of precise measurements based on electromagnetic forces and the definition of a kilogram based on fundamental constants. Our design is focused on the two main working modes of a watt balance, velocity mode in which a coil constant is calculated, and force mode in which that coil constant is used to realize some mass. We designed a custom 3D printed balance, along with a custom PCB to drive the measurements. Once assembled we found that velocity mode on our balance was inaccurate, but our force mode performed as expected. With a manual estimation of the coil constant, we were able to get accurate mass readings from our balance. The designs, failures, and successes are all documented in the following report.

Contents

1. Introduction

1.1 Problem

Since 1879, the universal standard of a kilogram was derived from a physical mass made of a platinum-iridium alloy stored away in a vault. However, even with careful preservation in optimal conditions, over time the mass of the object drifted slightly, which is problematic for use as a constant standard. In 2019, as part of a sweeping change to the SI system, the kilogram was redefined to depend on Planck’s constant rather than the mass of a physical object as described by Boutin [2]. Researchers at the National Institute of Standards and Technology (NIST) developed a device called a watt balance (or Kibble balance, after its inventor Dr. Bryan Kibble) that uses induced magnetic fields to precisely determine the mass of a given object by measuring the rotational velocity and force generated by the balance. Following the creation of the device, NIST also released a paper [4] detailing how to make a simple version of the balance at home using LEGOs. A group of graduate researchers in the ABE department of UIUC followed this paper to make their own LEGO version of the balance, but it is imprecise in its measurements, often off by 20-30%. This is problematic for a device whose goal is to create very precise measurements.

1.2 Solution

Our solution for this problem involved iterating on the design created by the UIUC graduate researchers and creating a more precise and easier to use watt balance. We planned to improve the sensing capabilities of the device and make mechanical improvements to help support those changes. By replacing the 3d printed pointed fulcrum of the balance with a smooth axle and bearing, we could not only reduce friction in the balance, but also use a more accurate position sensor on the axle than the ultrasonic sensor the previous design used. We planned to improve the sensing of the induced current in the coil using a discrete current sensor rather than the Arduino ports as it was previously measured to further increase the accuracy of the mass calculation. We also planned to update their existing MATLAB-based software implementation to a more powerful language to allow for more efficient processing of the data and create an easier to use user interface to match.

1.3 Operation

1.3.1 Visual Aid

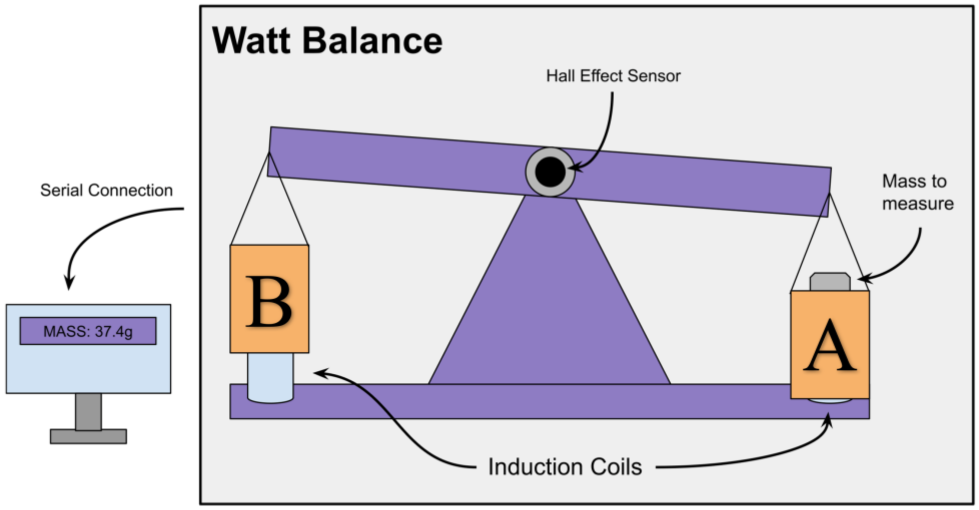


Figure 1: The high-level design of the balance

1.3.2 Velocity Mode

The first mode that the balance operates in is velocity mode. This mode is used to calculate the coil constant (also known as the flux integral) for the coil that has the mass above it, which we refer to as coil A. To achieve this, the opposite coil, coil B, is driven by a sinusoidal wave. We then measure the velocity of bucket A as it moves up and down through the coil and the voltage in coil A that is created by the induced current. Using Faraday's laws, we can find coil A's respective BL which we call $BL_{velocity}$ as seen in Equation 1.

$$v(BL)_{velocity} = V \tag{1}$$

1.3.3 Force Mode

The second operating mode is force mode. In force mode the balance drives coil A until it finds the current I through the coil that creates an upwards force counteracting the force of gravity, resulting in a leveled position. Then once this current is found, the mass of the object can be calculated. To do this we use the $BL_{velocity}$ found in Equation 1 and the local gravitational constant to solve for the mass in Equations 2 and 3.

$$F = mg = I(BL)_{velocity} \tag{2}$$

$$m = \frac{I(BL)_{velocity}}{g} \tag{3}$$

1.4 High Level Requirements

1.4.1 Accurate Position Measurements

To meet the requirements for accurate position measurements, the device must be able to correctly measure the rotational velocity of the balance as it rotates around the fulcrum. Given that the error when using a known mass in the original ABE device was between 20-30%, we targeted <5% error for our measurements to be considered accurate, and <1% error as a stretch goal.

1.4.2 Accurate Force Measurements

To meet the requirements for accurate force measurements, the device must be able to correctly measure the force generated by induction in the coil as the magnet moves through it. Like the rotational velocity, we targeted <5% error for our measurements to be considered accurate, and <1% error as a stretch goal.

1.4.3 Communication

To meet the requirements for communication, the device must be able to communicate with the control software via a serial connection and accept input calibration parameters given by the user of the device.

1.5 Subsystems

1.5.1 Block Diagram

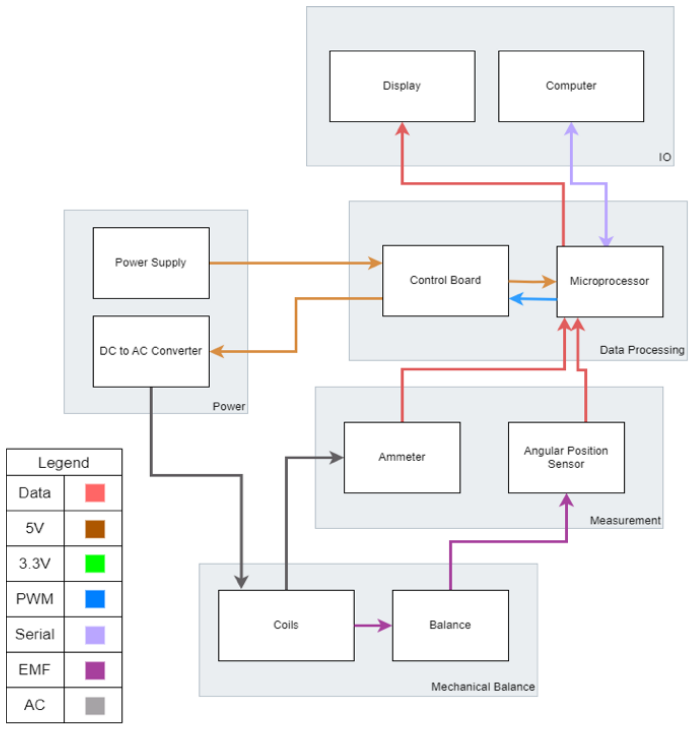


Figure 2: The subsystems of the balance and how they interconnect

1.5.2 Power Subsystem

The power subsystem plays the role of delivering the correct power and waveforms throughout the system. The primary focus of the power subsystem is to generate controllable waves to each coil. For coil B our goal was to generate a smooth bipolar sin wave between two and negative two volts with a controllable period to drive it in velocity mode. Then for coil A we needed to generate a current controlled supply. To accomplish this, we used pulse width modulation (PWM) and have a feedback mechanism that updates the duty cycle based on the current sensor.

1.5.3 Measurement Subsystem
The measurement subsystem encompasses all the sensors which collect the data for our calculations. It should both be able to measure the voltage induced in coil A in velocity mode and the current through coil A in force mode, along with the movement and velocity of the balance. The data from these measurements is sent to the data processing subsystem for both control feedback, tuning, and for calculating the coil constant and mass of the object.

1.5.4 Mechanical Subsystem
The mechanical subsystem is the physical balance itself and how it moves. It allows for the manipulation of the balance through the electromagnetic force generated by the coils. It also encapsulates the changes to the fulcrum of the balance to allow its angular position to be read by the angular position sensor.

1.5.5 Data Processing Subsystem
The data processing system acts as the brain for our project. The goal is it should be able to take in commands from the input output and measurement subsystems. Then with the data it should calculate and update the feedback mechanisms and send the data to the input output subsystem. The data processing subsystem should also be tunable, allowing for users to change constants to improve the performance of the balance.

1.5.6 Input Output Subsystem
The IO subsystem is the primary way for interacting with the balance. The goal is to have a display to read out reading from the sensors and the calculated mass and constants. It also needs to take in tuning variables to be able to change the constants, such as gravity, that are used in the data processing subsystem.

2 Design

2.2 Physical Balance

2.2.1 Base

There were several main design considerations that went into the base of the balance. The first was that it should be large enough to house our PCB, Hall effect sensor, and IO, as well as the associated wiring for those components. The second was that it needed to be stable enough to not move around while the arms are in motion. The third consideration was that it needed to have enough space around the fulcrum for the arms to swing without obstruction during operation. Overall, our final design for the base (Figure 3) accomplished all of these goals in a fashion that could be 3d printed easily.

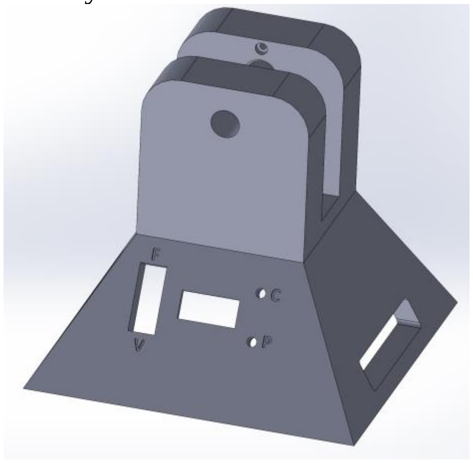


Figure 3: The CAD model of the base of the balance

2.2.2 Arms

The primary design factor on the arms of the balance was their length. They needed to be long enough to have a measurable range of motion, but short enough to be rigid and remain within the build volume of the 3d printer. Our final design for the arms (Figure 4) features a balanced length of 200mm, as well as small cutouts for the D-shaft, bearings for the buckets, as well as screw holes to attach the two halves of the arm.

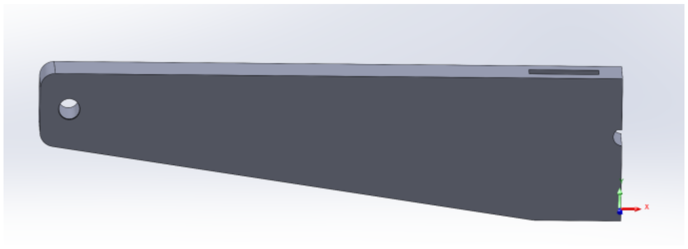


Figure 4: The CAD model of one side of the balance arm

2.2.3 Buckets

The buckets did not have as many design factors as the other parts of the balance, but considerations did have to be made for the bottom part that houses the magnets. It needed to be large enough to hold the magnets in, while being as slim as possible to avoid contacting the coils as it moved through them. Our final design of the bucket (Figure 5) features a long stalk to hold the magnets, a cup wide enough to hold different sizes of objects, as well as attachment points onto the arms.

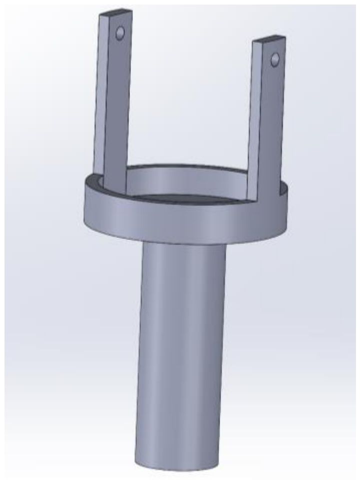


Figure 5: The CAD model of the final bucket design

2.2.4 Full Assembly
The full assembly of the balance features all of the components listed above, as well as bearings to ensure smooth motion of the arms and buckets, the two coils to be driven by the device, as well as internal mounting for the PCB.

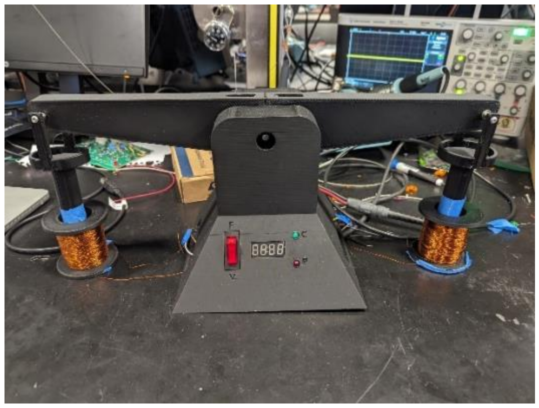


Figure 6: The completed balance assembly

2.3 PCB

2.3.1 Microcontroller
We chose to use the ATMega328PB as the microcontroller for our system. We were initially planning to use the ATMega328, as that is the one that we had practice using in class, however the PB revision contained support for two serial peripheral interface (SPI) connections, while the base 328 only has one. This would be useful to have both the SPI connection to the in-system programming (ISP) and the Hall effect sensor. We paired the ATMega with a 16 MHz crystal oscillator to allow for higher clock speeds, allowing for faster sampling of our sensors and for faster control over our voltage delivery. To connect the ADC pins on the ATMega with the coils we used a voltage shifter as pictured in Figure 7. This allowed us to read a bipolar voltage between -1.65V and 1.65V. We used one of the ADC pins for the connection to the current sensor, which delivered an analog voltage based on the read current. Two of the general pin input output (GPIO) pins we used to deliver the PWM wave to the half bridge, and another two were used to interface with an I2C connection to our DAC. Finally, we had external connectors out for both the SPI connections, and for all the additional unused GPIO pins.

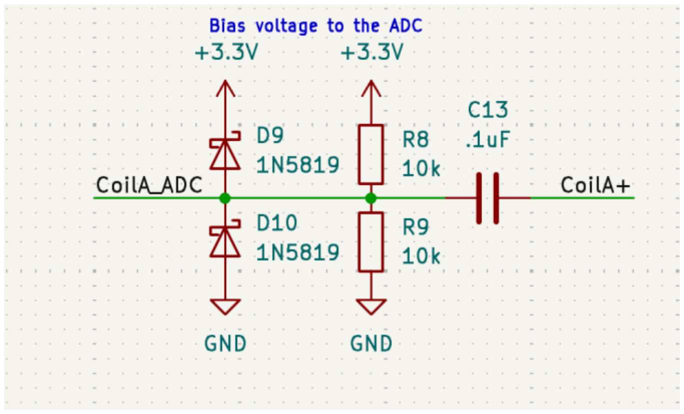


Figure 7: The voltage shifter to convert 0V to 3.3V into -1.65V to 1.65V

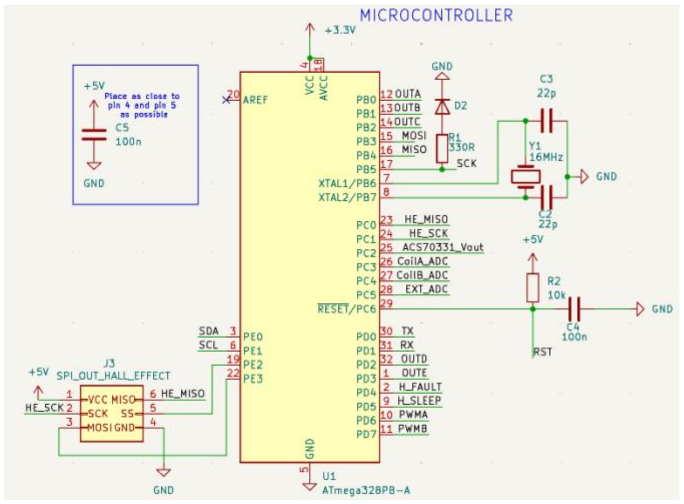


Figure 8: The ATMega328PB microcontroller and its connections

2.3.2 Power Delivery

For the power delivery we needed three voltage rails. A 3.3V rail to power the microcontroller and some ICs and a 5V rail to power the rest of the ICs. We also need a -5V rail to easily be able to send a bipolar voltage signal to our coils. The 5V supply is provided to us from the USB connection, we added a 1-amp glass fuse and an electrostatic discharge protection chip, the TPD3E001DR on the 5V rail from the USB connector to protect our device. The USB to UART chip we used, the FT232RL, had a built-in voltage step down to 3.3V we could use to create our 3.3V rail. Finally, we used the ICL7660 to create the -5V rail as shown in Figure 9.

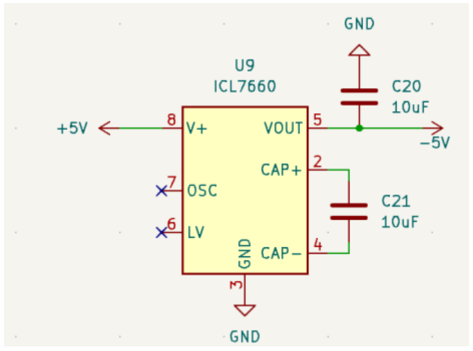


Figure 9: The ICL7660 CMOS voltage converter

2.3.3 Digital to Analog Converter

The digital to analog converter is a critical part of the system used for driving coil B in force mode. The MCP chip is controlled through an I2C connection to the ATMega. It is used to create a sinusoidal output between 0V and 5V. One we have our wave signal we pass it through the OP-AMP configuration seen in Figure 10. This OPA188AID shifts the wave into a range of between 1.25V and -1.25V. This can then be sent to the coil to create the sinusoidal motion.

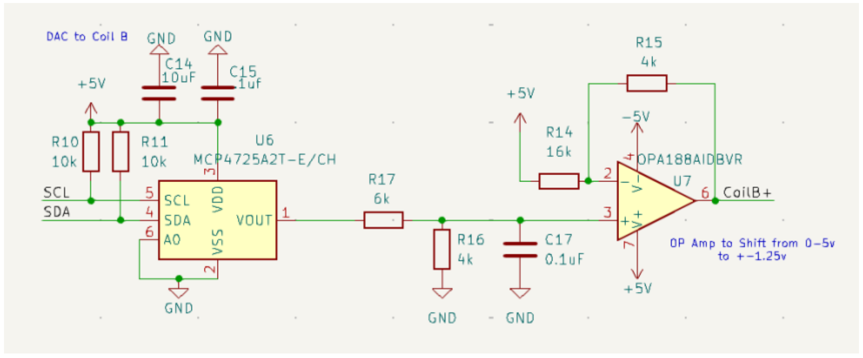


Figure 10: The DAC and OP-AMP circuit to drive the coils

2.3.4 Half Bridge

The half bridge, a DRV8848, is used as the current controlled voltage output to coil A in force mode. It is controlled by the ATMega through a PWM signal. The half bridge is capable of outputting a voltage between 0 and 5V with a maximum current of 2A as specified by [3]. This is useful as when in force mode, coil A requires an ample amount of power, roughly one watt, to be able to raise the mass to a balanced position.

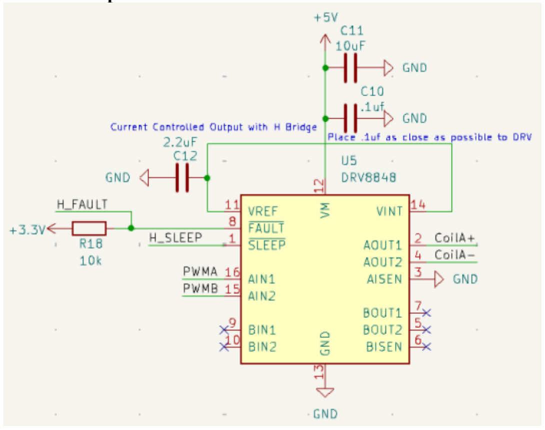


Figure 11: The DRV8848 H-Bridge for driving force mode

2.3.5 USB Connection

The USB connection to our PCB is key point in the design, as the USB both supplies the power and the communication with the host for our design. We included a 1-amp glass fuse on the power line along with an ESD protection chip on both the data and power line on the connector. The goal of this was to keep our design up with the best practices and protect our board from shorts and static discharges. We put a 5.1kΩ resistors on both CC1 and CC2, selecting to use the base power setting specified in the USB 2.0 specification seen in [5]. We routed the data lines into the FT232RL, which converts the USB signals into serial UART which we can then pass along into the UART RX and TX data lines of our ATMega. The FT232RL also has the benefit of providing a 3.3V line to use for logic ICs on the rest of our PCB. This setup allows for the optimal control of our PCB while maintaining simplicity and following the best practices to ensure safe operation.

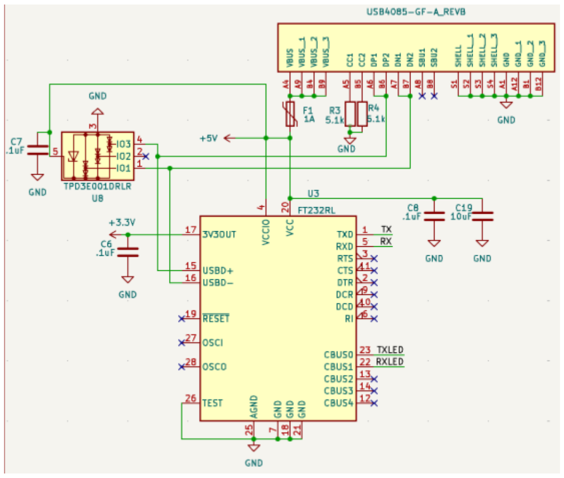


Figure 12: The USB-C connector and UART converter for power and software input to the balance

2.4 Arduino Uno

2.4.1 New Power Delivery

Due to issues in getting our ATMega to program we had to switch to using an Arduino Uno. While this made programming much simpler it limited our ability to deliver ample power to the components. In the new design we used two of the pins to create a bipolar digital wave out to coil B for velocity mode. This was achieved by switching the between high/low and low/high on the two pins out to coil B. To drive coil A, we bit banged a PWM wave through a pin to coil A and connected the other side of coil A to ground. These methods are effective but due to the Arduino's power limitations in the new design our current maxed out at roughly 40mA at 5V.

2.4.2 New Current Sensing

The Arduino had no available built in current sensor, which resulted in us calculating the current through the coil based on the voltage through the coil and the known impedance of the coil. While this measurement was sure to be less accurate, it would still allow us to operate force mode under our new circumstances.

2.5 Software

2.5.1 Control Flow

The control flow is illustrated by the flowchart in Figure 13. When the system boots, we initialize all the pins and communication interfaces. Then we enter the main loop, the first step is to read the state of the switch to understand which mode the balance is in. Then run through the respective mode operations. This cycle is continuously repeated while the balance is on.

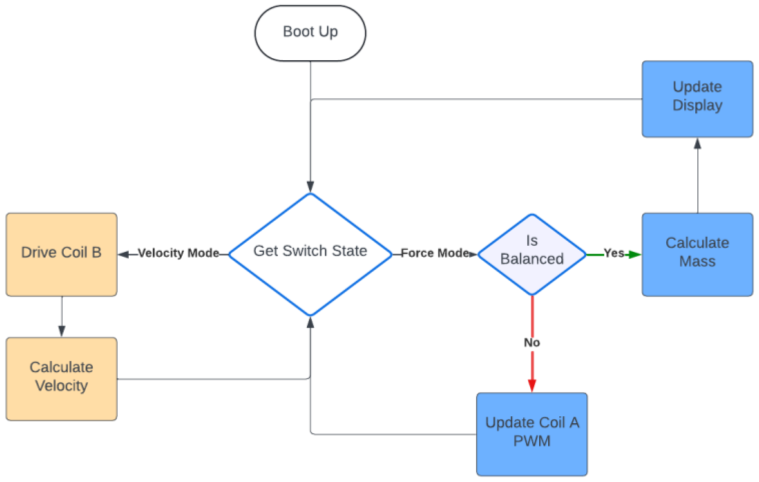


Figure 13: The high-level control logic flowchart

2.5.2 Velocity Mode

In velocity mode, we make sure that coil B is the only coil being driven, then we take two position samples, and two voltage samples of coil A to update our running average of BL. This loop continues and updates the calibration LED once enough samples of BL have been taken, in our case we went with 100 samples of BL.

2.5.3 Force Mode

In force mode, first we ensure that coil B is not being driven. Then we look at the position of the balance. If the position is less than the balanced position then we know we need to increase the duty cycle for our PWM to coil A, meanwhile if the position is greater than the balanced position we need to decrease our duty cycle. If the balance is detected to be leveled then the current through coil A, the gravitational constant, and the calculated BL from velocity mode are used to calculate the mass. Once the mass is calculated, the system turns on the calibrated LED and displays the mass on the 4x7 segment display.

3. Design Verification

3.1 Power Subsystem

3.1.1 Inverter

Our goal for the inverter was to have a smooth sinusoidal output. While this was not possible due to the hardware limitations of the Arduino, we were still able to generate a bipolar square wave to drive coil B, as is visible in Figure 14. Also pictured is the power limitations of the Arduino, where each leading edge of the wave the voltage spikes to 5V then back down to roughly 3.5V. This is due to the power limitations of the Arduino Uno, which we found can supply a maximum of roughly 40mA which is more than suggested in the datasheet [1]. Given that the resistance of our coil is 84Ω it is expected that the maximum voltage would be roughly 3.36V which is what is shown in Figure 13. The solution to this would be to get a DAC that is controllable by the Arduino with a much higher maximum current, which would allow us to get both the wave form and the power we wanted. Despite this, we believe that this result was good enough to meet are expectations for this verification.

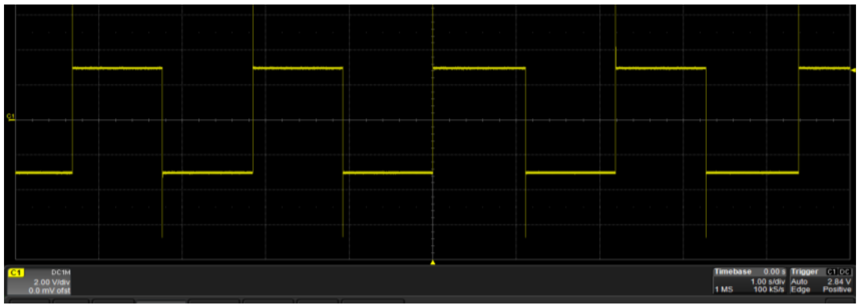


Figure 14: The square wave output from the Arduino to drive coil B

3.1.2 Power Supply

For the power supply to the board our verification was to make sure we were able to generate the appropriate voltages; 3.3V, 5V, and -5V. On our PCB we had probe points for each voltage rail and we found that all the required voltages were being produced. However, when we switched to use the Arduino we found that the 40mA provided by the pins were not sufficiently powerful to drive coil A in force mode, as we only had roughly one fifth the lifting power we expected. This can be seen in the PWM signal below as the Arduino pins initially try to hold 5V but the current limit means they must drop down. We believe that this verification was not adequately met in our final design.

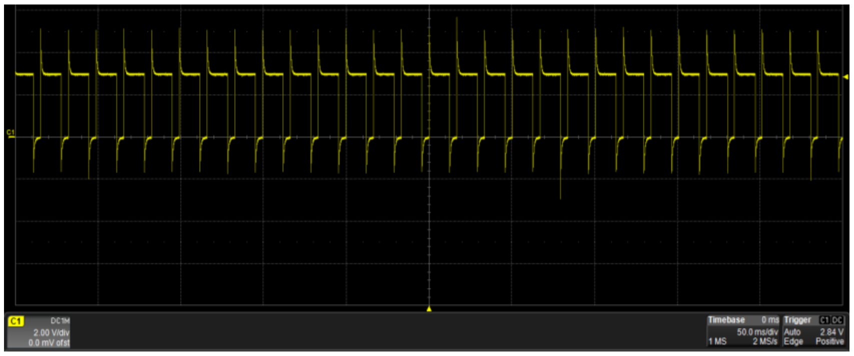


Figure 15: The PWM signal generated by the Arduino to drive coil A in force mode

3.2 Measurement Subsystem

3.2.1 Position Sensing

The position sensing in our design was done with the Hall effect sensor fixed around the fulcrum shaft. This sensor gave us an initial trouble when programming its SPI interface. The documentation on our sensor, the ERCK 05SPI 360, was sparse, and its data sheet did not thoroughly explain its operation. We ended up needing to iterate through all possible parameters until we found some that returned the correct position data. Once we had the position data from the sensor, we could see that the sensor was very accurate with only a small amount of jitter. The accuracy of our position sensor can be seen in Figure 16, where the angle of the balance oscillates and decays in amplitude as it targets being fully balanced at a position of 0m, with a small error bound to prevent it from oscillating infinitely. We found this to be within the 5% error described in the verification and consider the position measurement to be successful.

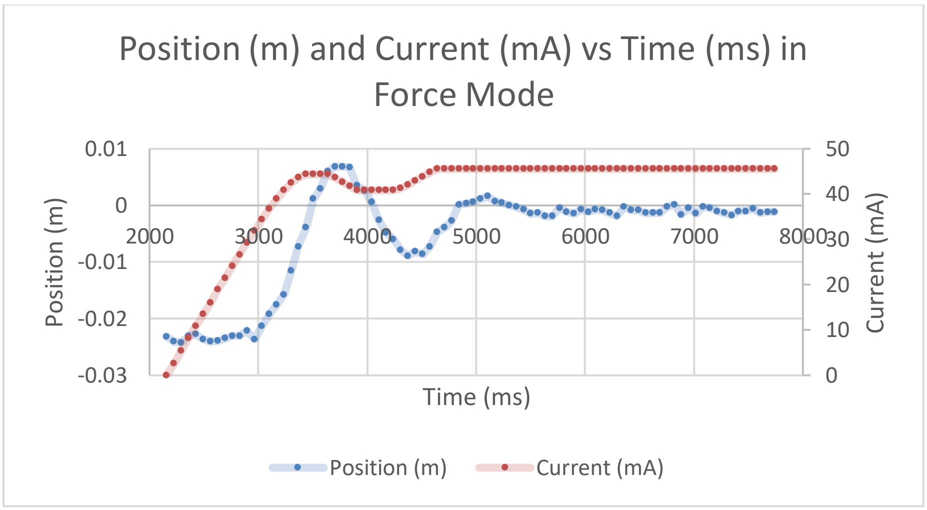


Figure 16: Position and Current over time in force mode operation

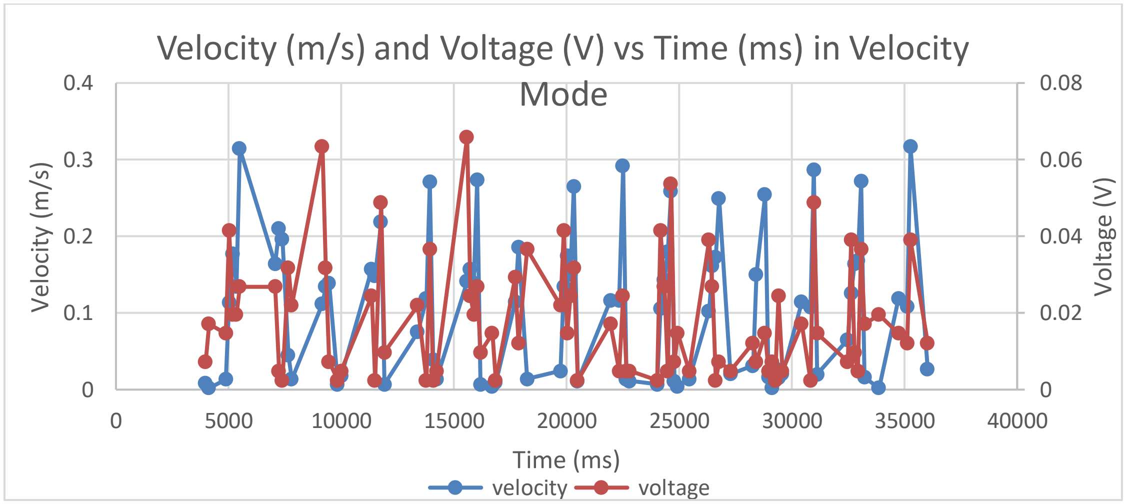


Figure 17: Velocity and voltage over time in velocity mode operation

3.2.2 Voltage Sensing
Although not in our verifications, we found that verifying the ability to read the voltage caused by the induced current in coil A was pivotal in the correct operation of the balance. Figure 18 shows an example oscilloscope reading of this induced voltage waveform. While we were able to read this voltage and correlate it with the velocity as shown in Figure 18, we believe that an improved ADC with higher resolution and a higher possible sampling rate would allow us to get better results.

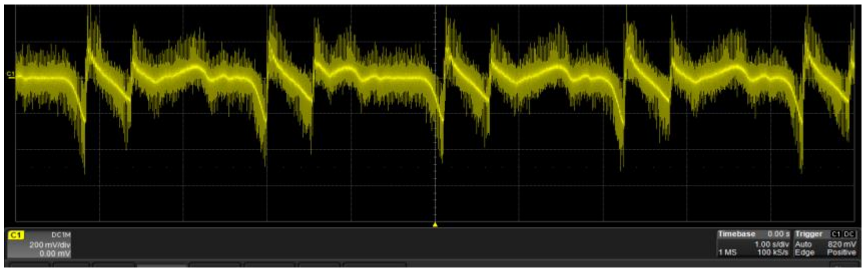


Figure 18: The induced current waveform in coil A during velocity mode operation

3.2.3 Current Sensing
In our final design, we were not able to use our current sensor due to using the Arduino instead of the PCB. Because of this, we had to estimate the current through coil A. We could do this reasonably as we knew that the pin would always supply 40mA of current when set high, and the duty cycle for the PWM corresponded the percentage of time the pin was high, using equation 4, we were able to get an accurate estimation of the current. While we don't believe that this satisfied our verification as it's only an estimate and therefore could is at times inaccurate with fluctuations in the supply current from the Arduino, it worked well enough to be able to accurately read a mass in force mode.

$$current = duty\_cycle * \ 40mA \tag{4}$$

3.3 Mechanical Subsystem

3.3.1 Movement from Coil
We were able to verify that the coils moved properly simply by running the balance in velocity mode, in which the arms of the balance oscillated up and down as expected. We did have a small issue with the buckets on the ends of the arms bottoming out and bumping into the coils at the bottom of their arc, but we do not believe this caused any major issues with the operation of the device.

3.3.2 Electromagnetic Interference
We did not have a Gaussmeter in the lab to verify there was no electromagnetic interference, and such a device was outside the budgetary range of this project, but driving the coils at full strength did not lead to any change in the measurements from the Hall effect sensor (the most electromagnetically sensitive component), so we can be reasonably confident that there is no significant interference between the coils and any other electrical component within the project.

3.3.3 Friction

Due to the current restrictions through the Arduino, the 10-gram test mass we had planned to be among the smallest measurable values was beyond the lifting capabilities of the coils. However, in the force mode tests with varying masses, we were able to see distinct and uniform measurements from changes in mass of less than a third of a gram, which is an indicator that friction in the bearings is even less of a concern than we originally thought.

3.4 Data Processing Subsystem

3.4.1 Receiving Measurements

For the data processing subsystem, it was important that we were able to interface with all of our sensors and receive data into the program. We initially found an issue when we were connecting our Hall effect sensor, as the documentation did not thoroughly lay out the parameters for connecting it through SPI. The special SPI interface worked by combining the MISO and MOSI lines into a single data line, this was done by bridging the two with a 10kΩ resistor. Once this was completed, we ran a suite of tests trying different clock speeds, delays between packets, and SPI modes until we found a combination that reliably delivered us accurate and consistent results. To receive measurements from the analog pins on the Arduino we just had to make sure that the coils were connected and that the pin modes were set to input. We are then easily able to read the voltages and use them in software, which meets the requirement to get measurements into our data processing subsystem.

3.4.2 Intermediate Calculations

The main intermediate calculation was to find the value of the flux integral, which is our coil constant. This is the value we get as a result of Equation 1 where we divide the voltage by the velocity. To so the accuracy of our calculation we can plot voltage against velocity in velocity mode and it should give a straight line with a slope of BL. However, as pictured in Figure 18, our data is highly nonlinear, which leads to poor accuracy in the BL calculation. This, combined with the jitteriness of our estimated current measurements due to the lack of a dedicated current sensor, leads to inaccurate final mass calculations. For the data processing system, we found it was easier to keep a running average to estimate BL. Figure 19 illustrates the inaccuracy of this running average, and that the average is raised by the high amount of error and noise in the voltage and velocity measurements. While this subsystem is successful at calculating an intermediate value as described in the verification it follows the adage of garbage in, garbage out as our data contains no clear way of finding an accurate BL.

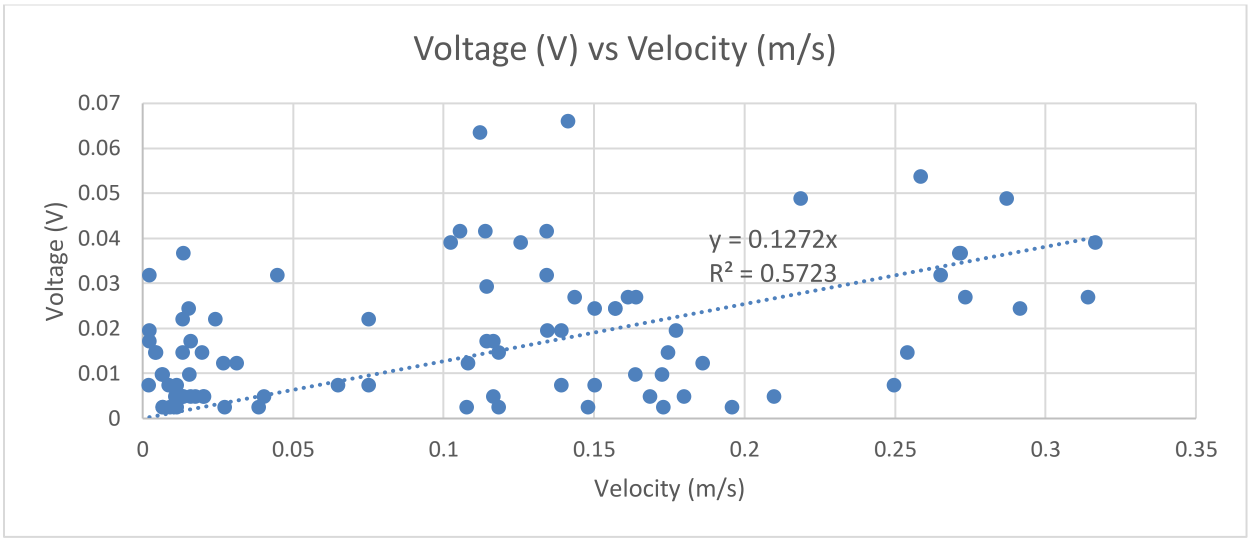


Figure 19: Voltage vs velocity in velocity mode operation, note that the slope should equal BL

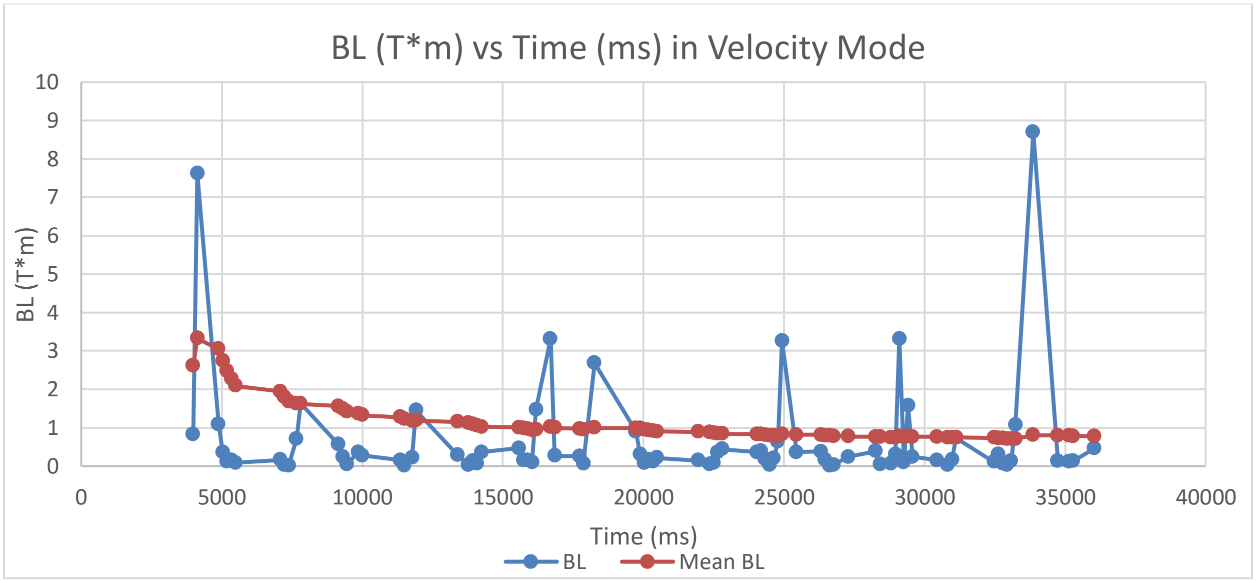


Figure 20: BL and Mean BL over time in velocity mode

3.4.3 Mass Calculations

For calculating the mass of the object in the bucket we use Equation 3. With the BL we collected in force mode we can use our current estimation, along with the gravitational constant to figure out the mass of the object. As seen in Figure 21 were successfully able to calculate masses up to roughly 4.5 grams although we were off by a linear factor. As we know our BL is highly inaccurate, this makes sense, given that the BL term in the formula is linear. Our mass measurement does increase linearly with an increase in mass, with little error. This shows that with a more accurate velocity mode and BL calculation our mass measurements would be correct. We believe that our mass calculations successfully demonstrate the balance's ability and pass the verification.

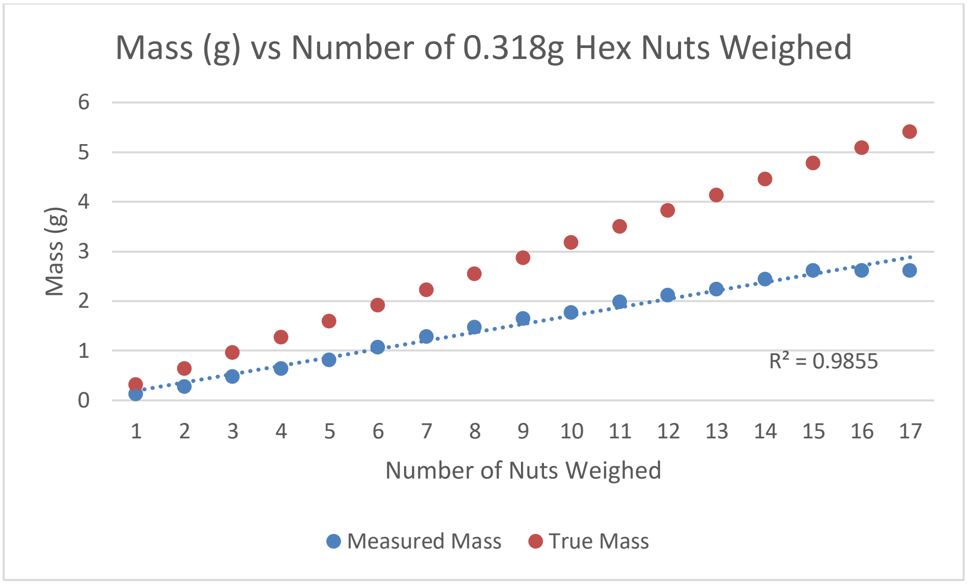


Figure 21: Incrementally adding mass and comparing measured vs known mass

3.5 Input Output Subsystem

3.5.1 User Interface Software

The goal for the user interface software was to verify user input of parameters and run the measurement program once using the control program. Before we switched to the Arduino, we had a small python program written to communicate through serial with the PCB. Once we switched to the Arduino, we found it was easier to use the built in Arduino serial library along with updating the parameters in the sketch. We had a section of definitions for the tuning parameters, this included gravity, the current feedback constant, and the zero position of the balance. The balance also prints out through serial over the USB the readings of every active sensor depending on the mode. We believe that the verbosity and the ease of use of the Arduino software help us pass this verification.

3.5.2 Front Display

Our front display was relatively simplistic, containing a switch to alternate between force and velocity modes, an indicator LED for power status, a second indicator LED for calibration status, and a digital readout to display the calculated mass once it has been calculated. It was integrated directly into the base of the balance for ease of use and proximity to the PCB housed within.

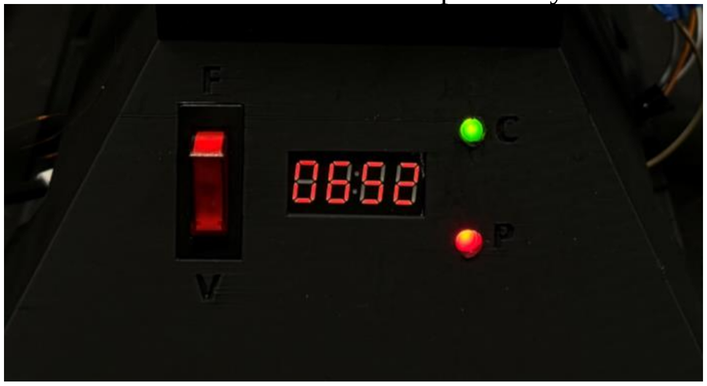


Figure 22: The front display of the balance

4. Costs

4.1 Parts

Physical Part	Unit Cost	Quantity	Subtotal Cost
Arduino Uno	$27.60	1	$27.60
ERCK 05SPI 360	$43.00	1	$43.00
Current Sensor	$1.57	1	$1.57
LED Screen	$3.50	1	$3.50
PLA filament	$20.00	1	$20.00
Copper Magnet Wire	$9.99	2	$19.98
Bearings	$0.91	2	$1.82
TRYMAG Small Strong Magnets	$15.99	1	$15.99
D-Shaft	$1.67	1	$1.67
ATMEGA328PB-AU	$1.47	1	$1.47
MCP2200-I/SO	$2.35	1	$2.35
ACS70331EOLCTR-2P5U3TR-ND	$2.71	1	$2.71
USB4085-GF-A	$0.85	1	$0.85
LED 4x7 Segment Display (Pack of 6)	$8.71	1	$8.71
ACS70331EOLCTR-2P5B3	$3.00	1	$3.00
DRV8848PWP	$2.95	1	$2.95
ICL7660CBAZ-T	$2.03	1	$2.03
OPA188AIDBVT	$2.86	1	$2.86
TPD3E001DRLR	$0.52	1	$0.52
FT232RNL	$4.80	1	$4.80
IA0505S	$5.82	1	$5.82

4.2 Labor

Labor Type	Number of Workers	Hourly Cost	Total Hours	Subtotal Cost
ECE Students	3	$50	4hr* x 15 weeks	$9,000

4.3 Total

The total cost of our project including parts and labor was $9173.20.

5. Conclusion

5.1 Accomplishments

There are many areas of the project in which we are quite proud of our accomplishments. The first such area is our PCB design. Our group members had no prior experience designing a PCB from scratch, and only limited experience populating and soldering premade PCBs in other contexts. With only a few weeks of learning and testing, we were able to design and populate multiple revisions of our PCB, and despite a few small oversights and issues with the bootloader, we were able to make a final design that we were happy with. We were also particularly enthusiastic about the performance of the Hall effect sensor and force mode overall, since it was able to provide very good results and linearity, even with its limited range of test masses.

5.2 Uncertainties

Although we were overall happy with the output and results of our project, there are still several areas in which we could have improved throughout the process. One such area is doing more quantitative analysis about the currents and voltages required for operation during the design phase, which would better inform our component and coil choices. Another area we felt lacked a little bit was rigorous testing and simulation of the PCB, as both major iterations of the PCB design had small issues, such as missing grounding pins or components having the wrong footprints. Catching these issues before we sent the board designs out for production would have helped our workflow a lot and limited the number of modifications we had to do later in the assembly process. Additionally, picking a more commonly used Hall effect sensor would have saved a lot of development time trying to guess at settings based on limited information available in the documentation. Lastly, starting on the code earlier in the development process would have allowed us to catch some of the sensor issues earlier, as well as give us more time to tune the different parameters.

5.3 Ethical considerations

We had a few safety and ethical considerations that we had to take into consideration while designing this device. One of the safety concerns that arose is as with any electrical system making sure that we use safe and proper power equipment to make sure that we are always operating within a safe voltage. We also consider accurate measurements an important safety factor, as inaccurate measurements can lead to accidents and failures when needed for a critical system. Because of this, we designed the balance to have an indicator for whether the balance has been calibrated and is ready to measure. This lets the user know that either the balance is accurate or not, improving the reliability and safety of the instrument.

5.4 Future work

While we feel comfortable with the amount of features we chose to tackle with this project, there are many other potential inclusions that could be incorporated into a future version. The simplest of these features would be a dedicated power button, as our device can only be powered off by unplugging it. Another such improvement would be designing a more compact PCB, through minimizing gaps as well as hardware choices, which would allow our printed base to be smaller as well. Additionally, one of the uses for the original NIST watt balance was to determine a precise value for Planck’s constant, which our device should be able to do with a few software changes, using the difference between unit definitions across the two modes. Next, we would include a more robust balanced level detection sensor, rather than relying on the zero position of the Hall effect sensor, to be more precise with our measurements. Another change that would have allowed us to skip many of the issues we encountered with programming our PCB would be to pre-flash the bootloader onto the microcontroller before soldering it on, then program it directly through the USB-C port, eliminating the need for the ISP programmer connection. Additionally, we could have more detailed feedback from the device using a full LCD screen that would be able to show the calculated values for the coil constants, as well as display sensor information that could previously only be read through an oscilloscope or a debugging program. Lastly, there are still plenty of improvements to be made to the physical design, removing the bottoming out issue with the buckets, and having a more robust mounting system for the coils.

5.5 Acknowledgements

We would like to thank the researchers in the Agricultural and Biological Engineering for pitching the project and guiding us through their work.

References

[1] *Arduino Uno Rev3 SMD*, datasheet, Arduino, Accessed April 2024. [Online], Available: https://media.digikey.com/pdf/Data%20Sheets/Arduino%20PDFs/A000073_Web.pdf

[2] C. Boutin, "NIST's Newest Watt Balance Brings World One Step Closer to New Kilogram" June 2016. [Online], Available: https://www.nist.gov/news-events/news/2016/06/nists-newest-watt-balance-brings-world-one-step-closer-new-kilogram

[3] *DRV8848 Dual H-Bridge Motor Driver*, datasheet, Texas Instruments, Oct. 2014. [Online], Available: https://www.ti.com/lit/ds/symlink/drv8848.pdf

[4] L. S. Chao, S. Schlamminger, D. B. Newell, J. R. Pratt, F. Seifert, X. Zhang, G. Sineriz, M. Liu, D. Haddad, "A LEGO Watt balance: An apparatus to determine a mass based on the new SI," Am. J. Phys., Nov. 2015. [Online], Available: https://doi.org/10.1119/1.4929898

[5] *Universal Serial Bus 3.0 Specification*, Rev 1.0, Nov. 2008. [Online], Available: https://www.maxrev.de/files/2014/08/usb_3_0_english.pdf

Appendix A Requirement and Verification Tables

Power Subsystem

Requirement	Verification
DC to AC converter (inverter) to the power coils.	Use an oscilloscope on the waveform sent to the coils to make sure they are consistent, and the desired waveform.
5v to the control board from the power supply.	Probe the voltage out of the power supply to make sure the voltage is correct and will not damage the components.

Measurement Subsystem

Requirement	Verification
Accurate sensing of angular position to within 5% error.	Begin with the balance at 0 degrees, then place a heavy object in the measurement area to ensure that it rotates as far as it can go. Compare the difference between the two measured values of the position sensor to the known maximum travel distance and see whether it is within 5% error.
Accurate sensing of current to within 5% error.	To verify, place the leads of an oscilloscope across the current sensor, then run the measurement program. Measure the current using the oscilloscope and make sure it matches with the measured value to within 5% error.

Mechanical Subsystem

Requirement	Verification
Ability to move the balance using the force from an induced magnetic field in the coil	To verify, apply varying currents to one coil, and observe whether the balance moves as a result. Repeat with the other coil to make sure both coils are capable of moving the balance.
Minimal interference with other electronic components (shielding if necessary)	To verify, use a gaussmeter to measure the emf at the coils and an ammeter to measure the induced current. When each electrical system is activated (apart from the inverter) measure the EMF and the induced current to be sure that no interference exists.
The device must have minimal friction in the pivot such that a 10	To verify, place a known 10 gram object on the measurement side of the balance. Then, run the force and velocity measurements on the balance using the control

gram weight is measurable to within the 5% accuracy tolerance.	software and see if the calculated mass value displayed is within 5% of the known mass.

Data Processing Subsystem

Requirement	**Verification**
Read in measurements from position and current sensors	To verify, run the measurement program and check that the measurements from both the position and current sensors are displayed in the control program.
Convert position and current data into angular velocity and force data	To verify, run the measurement program and check that the calculated values displayed in the control program match up with those calculated by hand given the measurements from the sensors.
Calculate the mass of the object based off the angular velocity and force data	To verify, run the measurement program and check that the calculated mass displayed in the control program matches with the value calculated by hand given the measurements from the sensors.

IO Subsystem

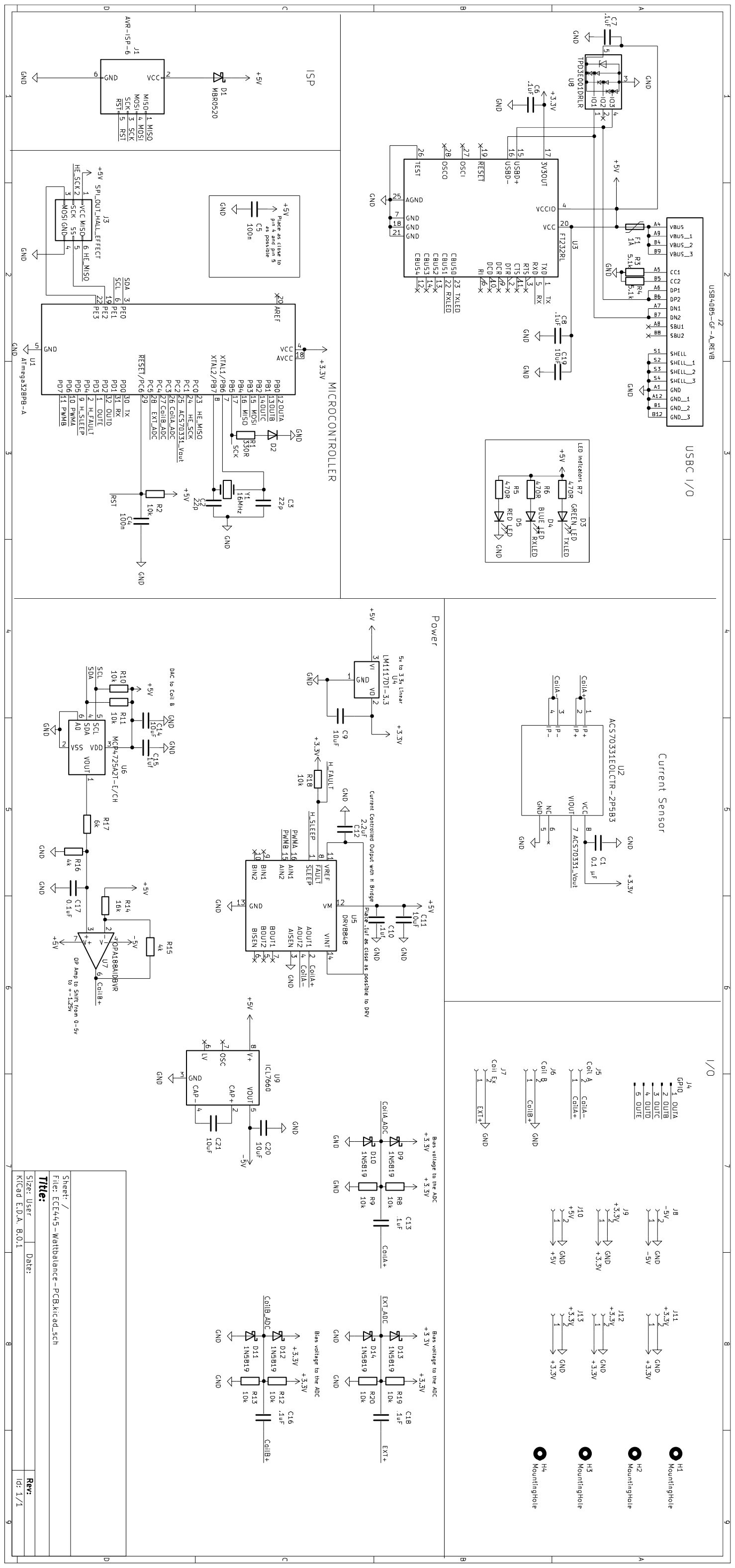
Requirement	**Verification**
User interface software to input control parameters for the balance	To verify user input of parameters, run the measurement program once using the control program. Then, edit the configuration values in the control program. Finally, run the measurement program again and see if the resulting measurements have changed from their original values based on the new tuning parameters.
Display to read out the mass of the object on the balance	To verify the display of data, run the measurement program using the control program, then check both the control software and the LED screen to confirm that they both display the same measured mass

Appendix B Schedule

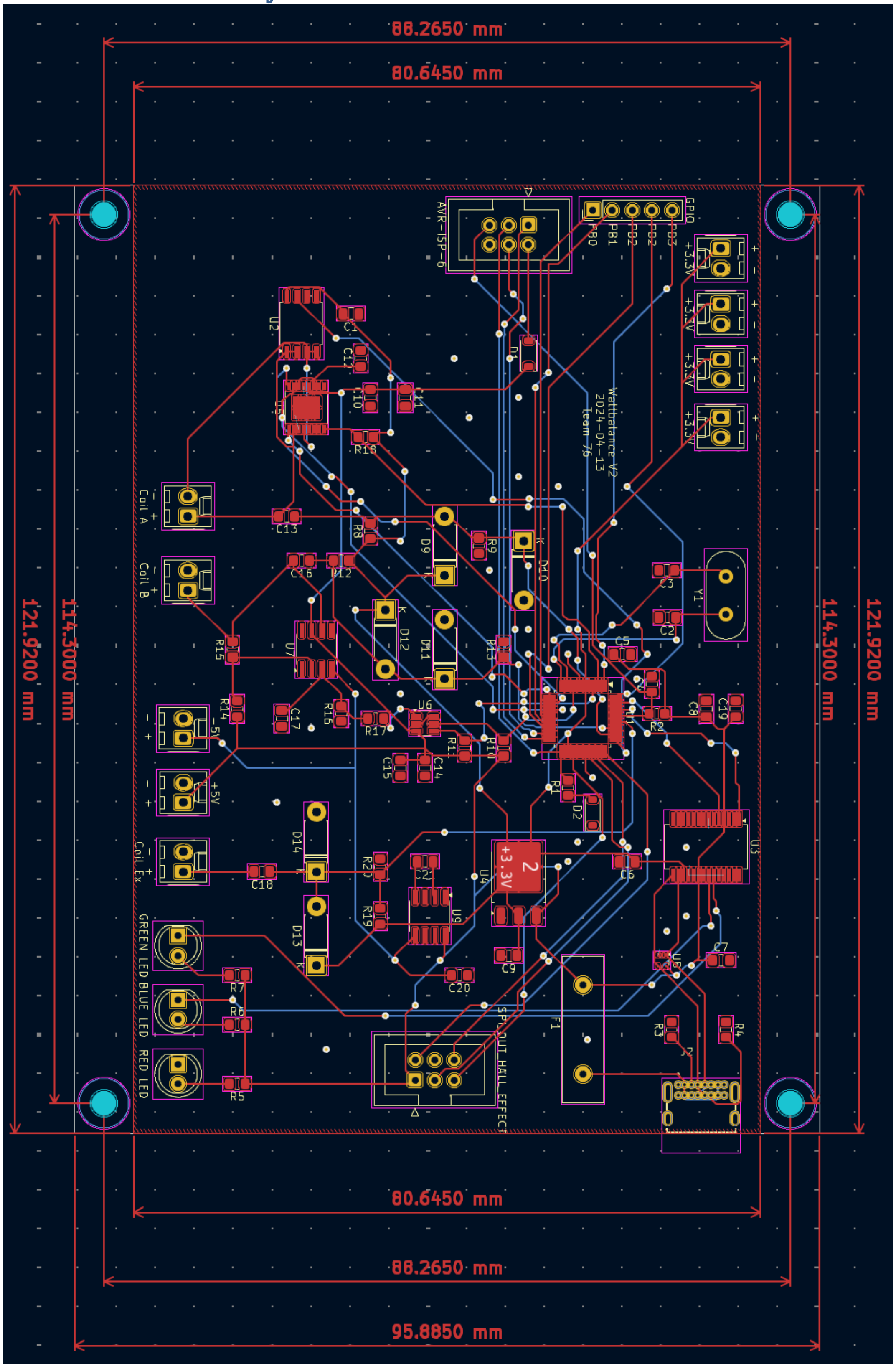
Week	Goals	Team Members
2/19	Fix and submit Proposal for Regrade Complete and Submit Design Document	Everyone Everyone
2/26	Design Review	Everyone

	Make revisions to design document Design PCB PCB Review	Everyone Everyone Everyone
3/4	Teamwork Evaluation PCB revisions	Individual Everyone
3/11	Spring Break	
3/18	Start to design software to work with new PCB and sensors Start to construct and iterate designs of balance	Julian John, Justin
3/25	Test to make sure balance fits within expected tolerances Revisions to Software Revisions to Balance	John, Justin Julian John, Justin
4/1	Make sure the balance and software interact correctly Revisions to Software Revisions to Balance	Everyone Julian John, Justin
4/8	Small refinements and tuning to the final version of the balance. Make sure software is stable and accurate.	John, Justin Julian
4/15	Mock Demo	Everyone
4/22	Final Demo Mock Presentation	Everyone Everyone
4/29	Final Presentation	Everyone

Appendix C PCB Schematic



Appendix D PCB Layout



88.2650 mm
80.6450 mm
121.9200 mm
114.3000 mm
95.8850 mm
AVR-ISP-6
GPIO
PB0
PB1
PB2
PB3
+3.3V
Wattbalance V2
2024-04-13
Team 76
Coil A
Coil B
Coil Ex
-5V
+5V
GREEN LED
BLUE LED
RED LED
SPI OUT HALL EFFECT
Y1
F1
U2
U3
U4
U6
U7
U9
+3.3V