Contents

1 - Introduction ........................................................................................................................................... 3
   1.1 - Overview .......................................................................................................................................... 3
   1.2 - Concept ........................................................................................................................................... 3
2 - Design Development .............................................................................................................................. 5
   2.1 - Crank Arm Design .......................................................................................................................... 5
   2.2 - Strain Gauge Circuit ........................................................................................................................ 6
      2.2.1 – Selecting Parameters .............................................................................................................. 6
      2.2.2 – Bridge Circuit ............................................................................................................................ 7
   2.3 – Load Cell Amplifier ........................................................................................................................ 9
   2.4 – Gyroscope ..................................................................................................................................... 10
   2.5 – Arduino ......................................................................................................................................... 10
   2.6 – Power Supply ............................................................................................................................... 11
   2.7 - Enclosure ..................................................................................................................................... 12
      2.7.1 - Design ..................................................................................................................................... 13
      2.7.2 - SolidWorks ............................................................................................................................. 13
3 – Final Design ....................................................................................................................................... 17
   3.1 – Hardware ....................................................................................................................................... 18
   3.2 – Software ....................................................................................................................................... 19
      3.2.1 – Arduino .................................................................................................................................. 19
      3.2.2 – Python .................................................................................................................................... 20
   3.3 – Data Analysis ................................................................................................................................ 21
      3.3.1 – Gyroscope Numbers .............................................................................................................. 21
      3.3.2 – Power Numbers ...................................................................................................................... 23
4 – Conclusion .......................................................................................................................................... 25
Appendix A: Bill of Materials .................................................................................................................. 26
References .................................................................................................................................................. 28
1 - Introduction

1.1 - Overview

A power meter is a cycling training tool used to record the power a rider is outputting. This is very useful to athletes who regularly do bike workouts, because the power output is a consistent measure of the rider’s effort level, and is not affected by outside factors such as wind or road gradient. If a cyclist does a workout with the intent to carry a certain speed for a certain amount of time, a strong headwind will slow them down and make them work harder to maintain the same speed, defeating the goal of the workout. When a power meter is used, the rider can try to maintain a certain number of watts across a period of time. When the same headwind affects them their speed will decrease, but if the power output stays consistent then the effort level will also be consistent. Power meters combine analog circuits with microcontroller technology and digital filtering, making this an ideal project for cycling-inclined computer engineers.

1.2 - Concept

Power is a rate of energy expenditure, which is useful for an athlete looking to manage their effort level. In physics, work is found by multiplying an applied force over the distance that force is applied, and power can be found by dividing work by the time the force is applied. Alternatively, the force applied by an object multiplied by the velocity of the object will also result in power. Regardless of the design chosen, two main things have to be calculated: the force’s magnitude, and the velocity of the applied force.

Power meters are already sold by a variety of companies in an array of design choices. A power meter can be mounted on the bike in any place that experiences force from the rider’s pedal stroke, such as the chain rings, crank arm, or rear wheel hub. All design types measure the force produced by the rider and use this number to calculate power. The crank arm design is the simplest one to implement, but has the disadvantage of being able to only measure the power output by one leg. Both the crank arm and wheel hub implementations measure the output from two legs, but the chain rings are subject to compatibility issues when swapped between bikes, and the hub is subject to lost energy due to friction between the chain and drivetrain interfaces.
A crank arm design was chosen for this project. Figure 1 shows an existing crank arm power meter design by the company Stages. Products like this served as the inspiration for this final design. The actual power meter is the rectangular insert placed on the inside of the crank arm, bonded to the material using an adhesive. It must be mounted in a very precise orientation to ensure the strain gauges read force in the correct direction, and calibrated for the specific model of crank arm.

To obtain the power numbers for a crank arm design, two measurements need to be made: the force the cyclist outputs on the crank arm, and the circular velocity of the rider’s foot moving through the pedal stroke. A set of strain gauges measure the strain induced on the crank arm, used to calculate force. The circular velocity is measured by a gyroscope that detects the angular velocity of the crank arm. Angular velocity $\omega$ is a measure of the movement through an angle over time, either in degrees/second, or radians/second. Circular velocity is a measure of the speed an object is moving around the circumference of a circle, given in meters/second. An object moving in a circle with a large radius will have a greater circular velocity than one moving with the same $\omega$ but a smaller radius. The resulting calculation for circular velocity is shown in Equation 1, where $t$ is the time period measured over, $r$ is the radius of the circle the rider’s foot moves around, $\omega$ is the angular velocity, $\theta$ is the angle in radians through the circle the foot has moved in time period $t$, and $d$ is the distance in meters the foot travels through the time period.

$$v = \omega * r = \frac{\theta * r}{t} = \frac{d}{t}$$

Equation 1: Circular velocity calculation.
The readout from the strain gauge correlates with a force vector. The relationship between the gauge output and the force’s magnitude is linear. A calibration constant scales the gauge output into the force magnitude. Once the force $F$ is found, it can be multiplied by $v$ to obtain the power $P$ in watts, as shown in Equation 2.

$$P = F \times v = \frac{F \times d}{t}$$

*Equation 2: Power calculation.*

## 2 - Design Development

### 2.1 - Crank Arm Design

Not all power meters are built into a bicycle’s crank arm. Although it is a very common design to see, it is not the only option. Some other popular locations to place the device include the rear wheel hub, front chainrings, and pedals. All locations rely on strain gauges to measure the rider’s force exerted on the respective bicycle part as the base of the power calculation. The problem with many of the designs mentioned, though, is the mechanical knowledge needed to build the electronics while making a stable part. With a wheel hub, for example, much of the product would have to be built from scratch to house the electronics inside the hub’s shell. The construction process would include manufacturing a shell out of a strong enough material to withstand the rider’s forces, mounting bearings to this shell, and installing the freehub with the ratcheting mechanism that allows the rider to coast, all while leaving space for the strain gauge circuit and microprocessor.

The crank arm design is appealing because the electronics can be added on to a production part with little modification. The crank arm itself is a simple piece that does not need to be rebuilt, and it also has a wide flat space that electronics can be easily mounted to. A crank arm’s strain is also easier to detect using strain gauges, and a simple circuit can be set up to achieve accurate measurements.

The design of a crank arm power meter breaks down into six main components that need to be designed individually. The strain gauge circuit is the network of four strain gauges connected in a Wheatstone bridge configuration [2]. The output of the strain gauge circuit feeds to a load cell amplifier to boost the signal and convert it from analog to digital. A gyroscope is used to gather the velocity of the rider’s foot. A microcontroller is needed to manage the data collection and calculation, and transmit the results via Bluetooth. A battery is needed to supply power to the device. Finally, an enclosure wraps up the electronics and attaches them to the crank arm.
2.2 - Strain Gauge Circuit

The strain gauge circuit requires the most precision out of any individual component of this project. Strain gauges have to be bonded to a very clean surface in a specific orientation, or the data they read will be inaccurate. Strain gauges have a number of important characteristics that need to be considered to find the best match for the project at hand. Strain gauges require weeks of research before they could be selected and installed with confidence.

2.2.1 – Selecting Parameters

Strain is the measure of a material’s deformation compared to its resting length, and strain gauges measure the magnitude of this deformation. Strain gauges are composed of a network of wires that stretch or compress depending on the motion of the material they are bonded to. As the gauge is stretched, the internal wires stretch also, increasing the device’s resistance. The opposite effect is had when the device is compressed [3]. Bending strain is experienced by the crank arm. This happens when a structure such as a straight bar experiences force at one end, causing it to bend into an arc [4]. An example of this can be seen in Figure 2. Strain gauges are placed on top and below the bar. When the bar experiences bending strain, the top gauge expands, while the bottom one contracts.

Like all electrical components, strain gauges have different qualities that tailor them for specific applications. Cal Poly Mechanical Engineering professor John Ridgely gave advice on what options to consider when choosing strain gauges. He recommended to look at the gauge length, resistance, and grid pattern when selecting a device. The gauge length is the length of the sensitive area of a strain gauge. The shorter the gauge length, the smaller the area the strain will be measured across. A longer gauge length is desired for this project, because the crank arm is expected to flex roughly equally along its entire length, so it is advantageous to measure across a larger area. Gauges with a length of 1-6 mm are recommended for metals, and if it is intended to
be stiff like a bicycle crank arm is, then one on the lower half of the spectrum should be chosen [5]. Using this information, gauges with a length of 3 mm were chosen for this project. The gauge resistance is important to reduce the current draw of the device. The higher the gauges’ resistance is, the less current the circuit will draw, and the longer the battery will last. Gauges are usually sold in values of 60, 120, 350, and 1000 Ω. Gauges with a higher resistance are also more expensive, so a set of 350 Ω gauges was chosen. Finally, the grid pattern refers to the alignment of the wires inside the gauge, which determines the direction the gauge will be sensitive to strain in. A uniaxial pattern is optimal for this project because it measures strain in only one direction, as opposed to a biaxial pattern, which measures strain in two directions offset by 90°. Figure 3 is a diagram of a uniaxial strain gauge. The entire wire mesh is aligned facing one direction, so this gauge will be most sensitive to strain along the length of the mesh.

![Figure 3](image_url)

*Figure 3: An example of a uniaxial strain gauge. The wires are aligned facing in one direction, which means that the gauge will be sensitive to strain in the direction parallel to the wires. Image from Tacuna Systems [6].*

### 2.2.2 – Bridge Circuit

Once the gauges are chosen, they have to be placed in a specific way to obtain meaningful results. First, the gauges have to be placed parallel to the direction of strain. In this case, the grid pattern is pointing down the length of the crank arm. The crank arm bends along its length as the rider puts force on it, so a gauge placed in this configuration will detect the maximum amount of strain. To better model the movement of the crank arm, a gauge is placed on both the top and bottom of the crank arm, to record the compression and extension. An example of this configuration can be seen in Figure 2. The gauges in that figure are configured in a circuit called a Wheatstone half bridge. Wheatstone bridges and half bridges are common circuits used to measure the resistance of a circuit element, and are the standard configuration for strain measuring applications.

A schematic for a Wheatstone bridge is shown in Figure 4. It consists of two parallel branches that each have a pair of resistors in series. The output voltage of the Wheatstone bridge is measured across the middle point of the two branches. If \( R_1 = R_4 \) and \( R_2 = R_3 \), the bridge is considered balanced, which means \( V_{\text{out}} \) is 0. If any of the resistance values change, however, then the bridge will be unbalanced, and \( V_{\text{out}} \) will change [2].
An easy way to configure a strain gauge without using a bridge is in a simple voltage divider with a resistor. As the gauge flexes, its resistance changes, making the output voltage of the divider change as well. The disadvantage of this configuration is its sensitivity to temperature. As the temperature changes, a strain gauge’s resistance changes as well, which would cause the circuit to register a change in strain. Wheatstone bridges are used to offset this problem. In Figure 4, changing $R_3$ and $R_4$ to strain gauges creates a Wheatstone half bridge. As long as both gauges are in the same environment, they will undergo the same thermal expansion. The ratio of their resistances will not change, and the output voltage will not change [7]. This entire circuit is referred to as a load cell, because it’s output varies depending on the mechanical load.

For small amounts of strain, the half bridge does not have a large output voltage. The Wheatstone full bridge, or simply the Wheatstone bridge, solves this problem. By replacing all four resistors with strain gauges, each one’s resistance will vary depending on the strain. A load cell requires the gauges to be placed in a specific way. Gauges on opposite sides of the bridge (e.g. $R_1$ and $R_3$) have to be placed on the same side of the crank arm. Figure 5 shows a 3D mockup of a crank arm, with the strain gauges mounted on the correct side of the crank arm. $R_1$ and $R_3$ (blue) share one side of the crank arm, and $R_2$ and $R_4$ (red) share the other.
Configuring the gauges in this manner results in twice the sensitivity of a half bridge circuit. This is because gauges on the opposite side of the Wheatstone bridge are experiencing the same type of strain, either compression or extension. The concept is illustrated in Figure 6. As the arm bends, $R_1$ and $R_3$ experience compression, lowering their resistance. Simultaneously, $R_2$ and $R_4$ experience extension, increasing their resistance. Assuming the compression and extension is symmetric, the voltage on $V_0-$ will rise the same amount that the voltage on $V_0+$ will fall, resulting in double the voltage swing. This makes Wheatstone full bridges ideal for high-sensitivity strain measurements.

![Wheatstone Bridge Diagram](image)

*Figure 6: Demonstration of a Wheatstone bridge under strain. Resistor numbering, starting from top left and proceeding counter-clockwise, is $R_1$, $R_2$, $R_3$, and $R_4$. Image from National Instruments [7].*

### 2.3 – Load Cell Amplifier

Despite the amplification qualities of the Wheatstone bridge, the circuit’s output voltage will still be on the order of tens or hundreds of microvolts. For many analog to digital converters, this is too weak of a signal to convert with reasonable resolution. To solve this problem, the signal was run through an amplifier before being fed into an ADC. This is accomplished by a chip made by Avia, the HX711, which provides both amplification and digital conversion [8].

![HX711 Breakout Board](image)

*Figure 7: An image of SparkFun’s HX711 breakout board. Image from SparkFun [9].*

Figure 7 contains an image of the HX711 breakout board produced by SparkFun [9]. This board is an HX711 chip installed on a PCB, with through-hole mounts connected to the chip’s pins. The pins on the left edge interface with the strain gauge circuit. The RED and BLK connections are the excitation+ and excitation- (E+/E-). They provide the positive and ground reference for the strain gauge circuit, and act in place of $V_{ex}$ in Figure 4. WHT is output+, and
GRN is output. These two read $V_{\text{out}}$, the voltage difference between the two branches in the bridge circuit. YLW is an optional ground that some systems make use of to discharge built up electromagnetic interference. On the right side of the board, VCC and VDD provide power for the chip and a digital signal reference, respectively. DAT and CLK are the data and clock lines for communication with the microcontroller. GND is the board’s ground reference.

2.4 – Gyroscope

To calculate the cyclist’s power through a crank arm, the rider’s cadence needs to be determined, or more specifically, the circular velocity of their foot as it moves through its pedal stroke. To gather this data, an inertial measurement unit (IMU) was used. An IMU is a chip that contains an accelerometer and gyroscope and combines the data from both for further processing. The accelerometer function was not needed for this project, because the gyroscope itself can measure the angular velocity of the crank arm. The device used for this project is an MPU6050, an IMU chip produced by InvenSense [10]. The chip was built onto a breakout board by DFRobot [11].

The breakout board contains input voltage and ground pins, serial data and clock pins, and an interrupt pin if the user chooses to have the IMU notify the microcontroller whenever it has a new set of samples ready. For this project, the interrupt pin was not used; instead, the microcontroller requests a new value from the MPU6050 on its own schedule. The reduced amount of wires also helps with the size considerations of the system.

2.5 – Arduino

The microcontroller used for this project needed to be something small. Modern microcontroller chips are incredibly compact. However, the prototyping process requires easily available IO, serial, or power pins, which limits the selection to pre-built microcontroller boards, such as an Arduino, Raspberry Pi, or the Texas Instruments MSP series. Built-in Bluetooth capability is a good quality, but not a requirement.

The Bluno Beetle BLE was chosen for this project. The Bluno series is a line of breakout boards produced by DFRobot that are built around an Arduino Uno chip, and adds built-in Bluetooth capability. The Beetle is the smallest Bluno board, measuring only 29x33 mm, and has Bluetooth BLE communication capability. Because of its size, it has a limited number of digital IO ports. Figure 8 depicts the Beetle and its available pins. There is one $V_{\text{in}}$ pin and one 5 V out pin, two ground connections, as well as 4 digital IO pins and 4 analog pins. On the backside are pads for TX/RX and SDA/SCL connections. Processing capability of the microcontroller is not a large concern. It only has a 16MHz clock, but the calculations needed for a power meter are not complex. The Beetle is built around the Arduino Uno’s ATmega328 processor, it should be more than capable of handling the needs of the program [12].
The Beetle controls two peripherals: the load cell amplifier and the IMU. The load cell amplifier communicates with a custom 2-wire interface which requires a clock and data line. Both clock and data can be assigned to analog pins. The IMU communicates via I2C, and the Beetle has the SCL and SDA pads on the bottom of the board, which are connected to I2C hardware. Outside of the serial communication wires, both devices have to be powered and connected to ground. Both peripheral devices are powered by 5 V, and connected to the same 5 V output pin. There are two ground connections, making it easy to ground both the IMU and load cell amplifier. The battery input connects to V\textsubscript{in} and one of the two ground pins. This still leaves four digital pins, two analog pins, and the TX/RX pads available for further connections, making this board more than capable of handling all the connections required.

2.6 – Power Supply

The entire system is powered by one battery. The power supply needs to provide at least 5 V to properly power the Arduino and the peripherals. Additionally, it has to be able to supply the required amount of current for a reasonable amount of time. In this case, the goal was to allow the device to run for at least five hours. This is not a reasonable number for a production power meter, but for this project it is sufficient.

Each component will have a typical current draw listed, and adding them together yields the estimated total current draw. The strain gauge circuit’s current draw can be determined using Ohm’s law. The circuit itself is a parallel combination of two branches, each branch with a resistance of 700 Ω. This makes the circuit’s total resistance equal to 350 Ω. The HX711 amplifier outputs 4.3 V to the gauge circuit. Using Ohm’s law, the current through the circuit is calculated to be 12.3 mA. The current draws of the other components had to be looked up through their respective datasheets. The Arduino draws 12.3 mA at a normal idle state. During Bluetooth transmission it will draw more, so this number is a lower estimate. The HX711 amplifier draws 1.5 mA [8], and the gyroscope draws 3.6 mA with the accelerometer disabled [10]. This results in 21.6 mA drawn by the whole system. If the system has to be active for at least five hours, then a power supply with at least 108 mAh will suffice.

The first design made use of a pair of CR2032 batteries, produced by Panasonic. A CR2032 outputs 3 V, so two were connected in series to supply 6V, which is above the 5 V minimum required by the Arduino. According to the datasheet, 6 V is safe for the Beetle’s V\textsubscript{in}. 
Initially, the batteries were bought assuming the battery would operate at the nominal capacity of 225 mAh [13]. However, when the system was built the first time, the batteries were only able to keep all the peripherals running for less than ten minutes. Upon closer inspection of the datasheet, it shows that the effective capacity is much lower than 225 mAh when the current draw is this high. Figure 9 is a graph included in the datasheet of this battery, indicating the load the battery is experiencing. The chart’s range does not even cover a load of 21.6 mA, which indicates that a pair of these batteries would power the system for a matter of minutes. A better solution is needed.

![Figure 9: Capacity vs. Load Resistance graph of the Panasonic CR2032 battery. Image from Panasonic [13].](image)

A rechargeable lithium-ion battery serves as a good replacement for the pair of CR2032 batteries. The battery has two benefits. It is rechargeable, which reduces waste and cost. Also, these batteries can have a much higher capacity than a CR2032, so it should have a better duration. A 3.7 V rechargeable battery with a 500mAh capacity was chosen. To boost the 3.7 V battery to a 5 V level, an Adafruit PowerBoost charger was used. This small chip boosts the battery’s output to 5.2 V, and also allows it to be recharged while in use. The PowerBoost also has an enable pin that, when connected to ground, shuts the device off. Attaching a slide switch between those two connections provides an easy way to turn the system on and off.

2.7 - Enclosure

The electronics require a container to organize the wiring, protect the components from damage, hold them in place, and provide a cleaner look. The enclosure has to be relatively small, no wider than the crank arm itself, and short enough to fit inside the area between the crank arm and the bike frame. It can be no wider than 35 mm and no taller than 30 mm to meet these constraints. It also has to be able to hold the parts firmly in place, so clamps or posts have to be added to hold them steady. It also must come apart easily and allow each part to be easily accessed, since the device will undergo many revisions through the development process.
2.7.1 - Design

To start, each component’s dimensions were measured to figure out which ones would be the limiting factors in the size of the enclosure. The largest ones are the Arduino Beetle board and the HX711 load cell amplifier, measuring at 33.1x28.3 mm and 31.0x23.5 mm, respectively. The Beetle has a micro-USB port for programming which needs to be exposed, and the HX711 has to be able to easily reach the strain gauge circuit, which is mounted to the crank arm but outside the enclosure. The Arduino’s serial clock and data pads are on the bottom of the board, so it should be mounted upside-down so the IMU can be easily wired to it. At this time, a pair of CR2032 batteries were still the main power supply, so there had to be space to accommodate for their holders.

The enclosure was designed to have two stacked levels to create a compact package. Figure 10 and Figure 11 are sketches of the layout design for the two layers. The HX711 has the pins facing the edge of the device, where wires are run through a hole in the enclosure and to the strain gauge circuit. The wires on the right of it interface with the Arduino. The second layer is stacked on top of the first. Wires connecting the IMU and the batteries to the first layer are run through holes cut in layer 2’s base.

![Figure 10: Layout of the enclosure’s first level.](image)

![Figure 11: Layout of the enclosure’s second level.](image)

2.7.2 - SolidWorks

To keep each component fixed in place, the holes cut into the PCB of each board slide over a set of pegs extruding from the base of the layer. The Beetle has no such holes, so a slot was cut into the wall of the enclosure to keep it steady. The enclosure was 3D printed, because of its small size and required tolerances. The models were created in SolidWorks.
Figure 12: Enclosure level 1 model, in SolidWorks.

Figure 12 shows layer 1’s model in SolidWorks. To support the Beetle, layer 1 includes a rectangular hole for the USB on the front side, as well as a slot cut in the back to hold the board in place. The left side has four pegs on the ground to hold the HX711, as well as a hole cut in the left wall to allow the wires to exit the enclosure and connect to the strain gauges. The rectangle and circle holes on the back wall are for a switch and a power indicator LED, respectively. The ledge protruding from the front and back walls allow layer 2 to rest above layer 1’s electrical components. The two rectangular cutouts on the bottom allow zip ties to pass through, to mount the enclosure on the crank arm.

Figure 13: Enclosure level 2 model, in SolidWorks.
Figure 13 shows layer 2’s model in SolidWorks. To hold the IMU, layer 2 has a pair of pegs to mount with the holes in its board. The rectangle cut in the bottom allows the wires leading from the IMU to pass through the base and connect to the Arduino. The circles cut on the right match with the CR2032’s battery mounts, with holes to allow the pair of wires to pass to layer 1. The large pegs extruding between the two batteries press on the ceiling of the cap, to hold layer 2 in place.

![Figure 14: Enclosure cap model, in SolidWorks.](image)

Figure 14 shows the cap’s SolidWorks Model. The cap sits on top of layer 1, with enough space to enclose layer 2 and its components. It features a space for a screw to hold the cap onto the hex bolt mounted in layer 1. Figure 15 is a model of the layers joined together, one with the cap removed and one with the cap installed.

![Figure 15: The assembled enclosure. The left image is with the cap removed. The right image is with the cap in place.](image)

The change from replaceable batteries to rechargeable requires a change in the enclosure shape as well. The cutouts for the battery holder on level 2 were no longer needed, and instead room was made to hold the PowerBoost 5 V battery adapter. Figure 16 is the revised model in SolidWorks.
An issue with the size of the USB port on the Arduino required the first level’s model to be updated as well. The USB port was wider and longer than anticipated, and it would not fit inside the enclosure with the original design. As a temporary fix some of the wall was cut out to allow the Arduino to fit inside, but for version 2 the cut was built into the design. The hex bolt attachment is also removed from the side, and tape is used to hold the two layers together. Figure 17 shows the updated level 1 model.
3 – Final Design

Figure 18 is a picture of the first iteration of the final design, with all the electrical components mounted inside the enclosure. The second level is removed, exposing the Beetle and the HX711. The MPU6050 can be seen on the top, and the boost converter is seen tucked behind the enclosure. Figure 19 shows the device closed and mounted on a bicycle. When mounted, the enclosure clears the frame by roughly 2 mm, not leaving much room for error. Because of this, the battery and power regulator cannot be mounted on the top, and must be tucked around the side to keep them out of harm’s way.

Note that the final models shown in Figure 16 and Figure 17 were never implemented. The design was made as an improvement on the current design by removing extraneous parts and adding mounting points for new hardware, but it was not able to be 3D printed before the end of the project. Figure 18 and Figure 19 are pictures of the final prototype reached in the duration of this project. This includes the electronics of the final schematic, but housed in the enclosure of the first iteration.
3.1 – Hardware

The final hardware design for this project is a result of multiple design iterations. Changing the power source is the biggest single change made to the schematic. Each part was developed and tested on its own before integrating it into the entire design. Figure 20 is the final schematic for the completed project.

![Schematic](image)

**Figure 20: Schematic for the final hardware design.**

The Wheatstone bridge is the strain detecting circuit. It connects to the HX711 load cell amplifier via four wires. Positive and negative excitation, or E+ and E-, supply an input voltage and ground for the bridge. Amplified positive and negative (A+ and A-) span the bridge’s output voltage and feed it back to the HX711, where the small signal is amplified and converted to a digital number.

The converted digital number is read from the HX711 by the Arduino. The HX711 communicates with a custom serial protocol. It still uses two wires, data and clock, like I2C. The HX711 holds the data line high during normal operation. When the HX711 has a new value converted, it stops pulling data high, which serves as a signal to the Arduino to read a new value. The Arduino pulses the clock line 24 times, and on each pulse the HX711 sends a new bit out the data line. When the process is complete, the data line is pulled high again.

The IMU’s purpose is to sense the angular velocity of the crank arm, which is a necessary metric for calculating the rider’s power output. It communicates with the Arduino via I2C, and is powered by the 5 V output.

Power for the entire circuit is supplied through the TPS61090 PowerBoost amplifier. This piece of hardware allows for two functions: it amplifies the voltage of a 3.7 V battery to 5.2 V, and it allows the attached battery to be recharged while in operation. It interfaces to the battery through a plastic JST connector matching the 3.7 V and ground connections. It has an attached micro-USB female port which attaches to any supply in the range of 4.75-5.25 V, which means
any computer and most cell phone wall adapters will suffice. There are holes in the board for the boosted output voltage of 5.2 V. A USB connection can be soldered here, but in this case a pair of wires were connected, which lead to the Arduino’s V\textsubscript{in} and ground pins. The PowerBoost also has an enable pin, which is normally pulled to 5.2 V but will shut down the board if connected to ground. As seen in the schematic, a slide switch was attached between EN and GND. When open, the PowerBoost board supplies power to the rest of the system, but when closed it shuts off.

3.2 – Software

There are two main software components to this project. The Arduino has a routine that gathers and processes data and sends it to a computer, while a Python program receives and displays the numbers sent by the power meter. To interface between the two, the Arduino sends data to the serial port via Bluetooth, which the program running on the computer is able to read and print. This allows for two places where the required numbers can be calculated. Although the computer will be able to process the numbers more efficiently than the Arduino will, the hardware development is the main focus for this project, not the design of the user interface on the computer. Thus, all the number calculations were implemented on the Arduino, and the results were streamed over to the computer.

3.2.1 – Arduino

All of the firmware was written in the Arduino IDE. Although the Bluno Beetle is not directly produced by Arduino, the Arduino IDE can compile and upload code for it because it shares the same chip as the Uno. To build the code correctly the compile target must be set for an Arduino Uno.

A big advantage of using an Arduino-based microcontroller is the access to the number of Arduino libraries that exist. Both the HX711 and the MPU6050 have Arduino libraries that were used in the software development of this project. Data from the HX711 is read using a custom two-wire protocol. The clock pin is pulsed 24 times, and on each clock pulse a bit from the ADC output is sent over the data line. The HX711 library [14] provides a read function that manages this operation. It also provides functions that return an average across a specified number of samples, one that returns the value in a specific data unit depending on a predetermined scale value, and one that resets the output to zero by changing an offset value. The MPU6050 library [15] provided a setup and calibration function that automatically generated offset values for the gyroscope and wrote them to the correct registers. It also has a function that returns the normalized angular velocity which automatically adjusts for small errors and scales down the output value.

The Beetle operates in a main loop, periodically polling the HX711 and MPU6050 to request new values. The main routine calls a series of functions that either gather or process data. A sample is first taken from both the gyroscope and load cell. The samples from both sensors are prone to noise or instantaneous spikes. Both power and cadence are rate measurements, so instantaneous values are not very valuable. To get rid of the spikes, a rolling average is used to calculate both values. A rolling average is found by using a FIFO buffer to store the most recent values read, and then periodically calculating the average across them.
When a cyclist is looking at their power numbers, they usually are looking at the 1 second or 5 second rolling average. This power meter is designed to output both numbers to the computer program. The FIFO buffer is sized to hold 5 seconds worth of data. The user display is designed to update every 500 ms, so to accommodate this every index of the FIFO represents half a second of data. An index of the force buffer represents the sum of each force reading taken in that 500 ms period. Parallel to that are buffers that represent the sum of every circular velocity measurement and the total number of samples taken in that period. To find a power calculation in that time interval, the total force sum is divided by the number of samples.

To calculate a 1 second number, the total average across the current index and previous index is found. To calculate a 5 second number, the total average across the entire FIFO is found. Because there are three FIFOs, the program results with an average force and circular velocity over the 1 second or 5 second time period. The average power is found by multiplying the force and circular velocity numbers together, and the cadence is found by running the average circular velocity through Equation 3.

When the 500 ms timer is up, the new averages are calculated and sent to the computer via Bluetooth. The print operation is triggered by an interrupt set on a timer using the Arduino TimerObject library, which generates an interrupt on a specified time interval. For this program, every 500 ms the TimerObject triggered a function that set a flag to ‘true’. This would allow a segment of code which calculates and outputs the rolling averages in the main loop to execute.

One issue found with the readings from the HX711 is the tendency to drift. When the program was left to run over a period of time, the values would slowly increase without any movement or force on the crank. To offset this issue, the tare() function from the HX711 library was used. A tare function on a bathroom or scientific scale works by adjusting its zero value, offsetting the output value by the weight on the scale. This works the same way, by setting an internal offset variable that is subtracted from every scale reading. The tare() function is called whenever the circular velocity is below 0.1 m/s, and the force reading is above 10 N.

3.2.2 – Python

On the computer side, a simple user interface was written in Python to display the numbers sent over from the Arduino. Figure 21 is a screenshot of this interface. The user selects their serial port using a drop down menu, containing the ports available on that machine at the moment. The refresh button updates the list of available serial ports. The baud rate can be selected using a drop down populated with the most common baud rates. The open and close buttons change the status of the serial port. When a port is open, data appears on the console.
Data has to be formatted in a certain way to appear on this console. The raw number is sent from the Arduino, but prefixed with a character specifying what type of data it is. Cadence is prefaced with a ‘c’, 1 second power with a ‘o’, and 5 second power with an ‘f’. This helps the Python program identify what data is read. If data does not have a tag, it is ignored.

3.3 – Data Analysis

Data collection and calibration is crucial to the accuracy of this project. The output of the load cell is a meaningless number that needs to be correlated with a real life number for it to carry any weight. The MPU6050 returns the angular velocity in degrees per second. This value is a good start, but it needs to be adjusted to account for the length of the crank arm before it is useful for this application.

3.3.1 – Gyroscope Numbers

The gyroscope detects its angular velocity in three dimensions, and returns the value in degrees per second. It is a simple calculation to convert the angular velocity to circular velocity and cadence. Angular velocity has the units degrees/second, and circular velocity is in meters/second. As shown in Equation 1, the circular velocity is found by multiplying the angular velocity by the radius of the circle the rider’s foot follows through its pedal stroke. The radius is measured from the center of the pedal spindle, to the center of the bottom bracket spindle. The angular velocity must first be converted from degrees/second to radians/second, through multiplying by \( \frac{2\pi}{360} \). The radius was measured to be 17.5 cm.

One problem with using the angular velocity as a measurement is its tendency to jitter. One instant to fix is to normalize the velocity reading from the gyroscope. The MPU6050 library contains a function that reads the normalized velocity instead of the raw velocity. The normalizing function first scales the readings to be in degrees per second, instead of an arbitrary number. Then, it checks to see if the number is below a certain threshold, and if it is, it will return zero instead of a number that was most likely generated due to gyroscope drift. Angular velocity has to be converted to circular velocity to be used for a power calculation. In the process, a 1 second rolling average is applied to the data to smooth out the waveform and reduce noise. Figure 22 depicts the distance between the raw circular velocity and the averaged output. The averaged value is smoother and more useful for a metric like cadence and power, which are more useful as rates, not instantaneous values.
The rider’s cadence can also be easily found using the circular velocity readings. Cadence is not necessary for the power calculation, but it is a useful metric for a cyclist to have. Cadence is the rate the rider’s legs are moving at, given in rotations per minute (rpm). Equation 3 is the derivation of cadence in rpm from the circular velocity in m/s. Figure 23 is a graph of the cadence calculated from the averaged circular velocity, across the same data sample as Figure 22.

\[
\text{Cadence } \frac{\text{rotations}}{\text{min}} = \nu \frac{\text{m}}{\text{s}} \cdot \frac{1 \text{ rotation}}{2\pi r} \cdot \frac{60 \text{ s}}{1 \text{ min}} = \frac{30}{\pi r}
\]

*Equation 3: Calculation of cadence from angular velocity*

Figure 22: Normalized angular velocity vs. elapsed time.

Figure 23: Cadence value calculated from the averaged circular velocity.
3.3.2 – Power Numbers

The strain gauge circuit is responsible for gathering the force data. The output from the HX711 is a 24-bit number representing the voltage across the strain gauges. As the force on the crank arm increases, this number increases linearly. The number has to be scaled by a constant found by calibration, or else it is just an arbitrary value. To find this constant, a known weight was hung from the end of the crank arm. The output number from the HX711 divided by the known weight is the calibration constant. The HX711 library provides a function that automatically divides values read by this calibration constant. In this case, an object of 5 pounds (22.24 newtons) resulted in an average ADC output of 44,802. The ADC was scaled to newtons, so 44,802 was divided by 22.24 newtons, leading to a constant of 2014.9.

The scaled output of the ADC represents force, in newtons. When force is multiplied by the rotational velocity, found by the gyroscope, it results in the instantaneous power. Figure 24 displays the relationship between the force and power. A negative force represents the segment of the pedal stroke when the rider pulls up on the pedals, creating a negative voltage difference across the strain circuit. Power uses the absolute value of the force number, because any force output is still produced by the rider and should be included in the calculation. Power will usually be slightly larger than the force, because the pedal’s rotational velocity usually lies around 1-2 m/s.

Looking at Figure 24, the instantaneous power varies rapidly. When the rider is pushing down on the pedal, they are able to generate more power than they can when they are pulling up. This explains the periodic nature of the power and force waveform. However, this makes the instantaneous power not very useful to a cyclist, because they want to know how much power they are outputting over a complete pedal stroke or across multiple seconds, not at specific points in time. A very common function for power meters is to output power in rolling averages. This takes the average power across the respective time intervals and averages it, to make a smoother output.
This project implemented 1 second and 5 second averages. The method of finding the rolling average is described in section 3.2.1 – Arduino. Both averages were computed and output every 500ms, creating a smoothed signal that is more useful to a cyclist. The effects of the averages is shown in Figure 25. The 1 second average is still subject to random spikes and drops. The rider’s cadence was roughly 70 rpm through the entire test, which means that 1 second covers slightly more than one full pedal rotation. If the start and end of the one second sample happened to occur on an upstroke, then the 1 second power will be lower than expected. The 5 second power is a more stable waveform, increasing and decreasing at a more reasonable pace. A relationship exists between the 1 second and 5 second power; they experience the same slow rise to begin the test, crest around the middle, and spike at the end.

![POWER AND CADENCE VS TIME](image)

*Figure 25: Instantaneous power, along with the 1s and 5s averages, shown on the same plot as the cadence.*
4 – Conclusion

A power meter is a useful cycling training tool that informs the rider of their current energy expenditure rate. The cyclist can use this to specialize their training further. This design builds one mounted on the crank arm of the bicycle, where it measures the amount the arm flexes. This is a metric called strain, which is directly correlated to the force the rider is outputting. The force is multiplied by the circular speed of the pedals to achieve power. The device is designed sufficiently small to be mounted on the inside of the crank arm and not interfere with the frame. It is designed to be completely wireless, because running a power or data connection to a rotating crank arm is not feasible. The device communicates with a computer via Bluetooth BLE, and it sends its calculated power and cadence data to a computer program which displays it for the rider to see.

There are a few features that could be added on to this project. Instead of receiving the data on a computer program, a phone app should be written that receives and displays the data. Having this data mobile would make testing much easier, and potentially lead to a marketable product in the future. As of now, the system has to be within Bluetooth range of a computer, which is a limiting factor. Another improvement on this project is to attempt to reduce the size of it. Most of the space comes from the large breakout boards and the wires connecting them. Somebody with PCB design experience could purchase the bare chips and solder them to a custom board. This would eliminate the need for wires running through the enclosure and greatly reduce the device’s size. Both would add a brand new dimension to this project, and may have been explored if this was a group effort instead of an individual.

If this project was going to be replicated, I would recommend working with more precision than I did. There were many connections that had to be re-soldered, which damaged the serial data pad on one Arduino. The strain gauges have very thin wires leading out of them, and they broke multiple times, which led to them having to be re-soldered. There are terminals that exist that are meant specifically for these gauges, to hold the wires in place. I would recommend investing in those for a future project. Faulty connections plagued this project, and loose wires would make one component work one day and be unresponsive the next day. With some better planning, this project could produce better data and a more polished final product.
Appendix A: Bill of Materials
<table>
<thead>
<tr>
<th>Name</th>
<th>Supplier</th>
<th>Part No.</th>
<th>PPU</th>
<th>Qty</th>
<th>Subtotal</th>
<th>Shipping</th>
<th>Tax</th>
</tr>
</thead>
<tbody>
<tr>
<td>22 AWG Copper Wire, Striveday</td>
<td>Amazon</td>
<td>stranded-wire</td>
<td>$16.99</td>
<td>1</td>
<td>$16.99</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6 DOF Sensor IMU - MPU6050</td>
<td>DFRobot</td>
<td>SEN0142</td>
<td>$9.20</td>
<td>2</td>
<td>$18.40</td>
<td>$12.00</td>
<td>-</td>
</tr>
<tr>
<td>Arduino Bluno Beetle BLE</td>
<td>DFRobot</td>
<td>DFR0339</td>
<td>$14.50</td>
<td>2</td>
<td>$29.00</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Strain Gauge 5 pack, 350 Ohm</td>
<td>Omega Strain</td>
<td>350-TY13</td>
<td>$36.75</td>
<td>1</td>
<td>$36.75</td>
<td>$12.00</td>
<td>$3.68</td>
</tr>
<tr>
<td>Strain Gauge Adhesive</td>
<td>Omega Strain</td>
<td>SG401</td>
<td>$10.70</td>
<td>1</td>
<td>$10.70</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>HX711 Load Cell Amplifier</td>
<td>Sparkfun</td>
<td>SEN-13879</td>
<td>$16.11</td>
<td>1</td>
<td>$16.11</td>
<td>$6.16</td>
<td>-</td>
</tr>
<tr>
<td>CR2032 Batteries, Panasonic - 24 pack</td>
<td>Amazon</td>
<td>B01881CTKQ</td>
<td>$8.90</td>
<td>1</td>
<td>$8.90</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CR2032 holder, uxCell, 10 pack</td>
<td>Amazon</td>
<td>a13102800ux0447</td>
<td>$6.32</td>
<td>1</td>
<td>$6.32</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>IL081 Assorted LED, 25 pack</td>
<td>Amazon</td>
<td>IL081</td>
<td>$5.49</td>
<td>1</td>
<td>$5.49</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SPDT Slide Switch with Solder Lug, 5 pack</td>
<td>Amazon</td>
<td>a12022800ux0258</td>
<td>$5.50</td>
<td>1</td>
<td>$5.50</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>8&quot; Cable Ties, Black Nylon, 100 pcs</td>
<td>Amazon</td>
<td>43237-2</td>
<td>$5.99</td>
<td>1</td>
<td>$5.99</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Scotch Super33 Electrical Tape, Black</td>
<td>Amazon</td>
<td>6132-BA-10</td>
<td>$3.98</td>
<td>1</td>
<td>$3.98</td>
<td>-</td>
<td>$0.33</td>
</tr>
<tr>
<td>Lithium Ion Polymer Battery, 3.7V, 500mAh</td>
<td>Amazon</td>
<td>1578</td>
<td>$14.06</td>
<td>1</td>
<td>$14.06</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>PowerBoost Recharger and 5v regulator</td>
<td>Amazon</td>
<td>1944</td>
<td>$15.95</td>
<td>1</td>
<td>$15.95</td>
<td>$1.09</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$14.14</td>
<td></td>
<td>$14.14</td>
<td>$30.16</td>
<td>$5.10</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Total</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$229.40</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
References


