Smart Walker: Biometric Neurorehabilitation and Mobility Assessment System

A Sensor-Integrated walker for quantifying patient Load, Speed, and Distance

---

Final Report, December 10, 2025

BME 200/300

Client: Dan Kutschera, PT
Advisor: Dr. Duc-Huy Nguyen

Group members
Nicolas Maldonado (Team Lead)
Aidan Burich (Communicator)
Carolyn Randolph (BSAC)
Nial Donohoo (BWIG)
Henry Salita (BPAG)

Abstract

Patients undergoing rehabilitation from traumatic brain injuries (TBI) often rely on a walker for mobility, especially while recovering from experiencing trauma. Patients frequently face challenging recoveries, both emotionally and physically, and frequently have no way of feeling they are progressing. Practices also have difficulty demonstrating to insurance providers that they are making a difference. Therapy progress is inherently difficult to quantify with available resources, leaving room for error and forcing practitioners to rely on intuition. The client, Dan Kutschera, a physical therapist with Encompass Health, has requested the development of an innovative walker system that provides accurate, measurable feedback on patient performance. The system will consist of devices that attach directly to a standard walker without altering the safety or function. Weight sensors will be integrated beneath the walker to measure the support it provides to the patient, with sensors capable of accurately tracking weight up to 165 lb. The LiDAR sensors will measure speed and distance up to a range of 120 ft, with an average error rate of 1.24%. Together, these features will allow therapists to evaluate rehabilitation progress with clear metrics. Testing will involve applying known weights and measuring controlled distances and speeds to ensure accuracy and reliability. By developing a walker that provides both quantitative feedback and clinical reassurance, this project aims to enhance rehabilitation outcomes and equip therapists with data to support patient confidence and facilitate accurate insurance documentation.

Table of Contents

Introduction

Motivation

Patients with traumatic brain injuries often undergo traumatic events followed by intense rehabilitation to help them walk and return to everyday life as soon as possible. Acute stroke care clinics provide specialized care to initiate recovery following hospitalization [1]. During rehabilitation, physicians struggle to measure progress as patients gain strength, on the assumption that patients will place progressively less load on the walker as they regain strength. This makes it difficult to provide patients with tangible evidence of their improvement, and insurance companies require proof of improvement. Insurance companies use a process called "Medical Insurance Credentialing" to verify that medical providers are qualified to receive reimbursement for their services [2]. This necessitates that clinics document progress to be compensated for their hard work. Patients and practitioners alike need a way to track tangible data with ease. The smart walker will measure the pressure applied, speed, and distance walked of patients with neuro-rehabilitation needs. This data will be reported and displayed in real-time to help clinicians monitor progress and motivate patients. Ultimately, the device will reduce the time required to meet Medicare's documentation needs and increase objective markers of patient readiness for discharge [3].

Existing Devices

Currently, no commercially available devices fully meet these clinical and functional requirements. An example of an adapted smart walker is the Camino, which is an AI-equipped device that offers features such as automatic braking and power-assisted motion [4]. Several issues arise with this product. These issues stem from the fact that the Camino is not designed for use in clinical settings, limiting its applicability in structured rehabilitation environments. As a result, it is more focused on assisting ambulation than on monitoring progress. Therefore, the Camino does not measure the pressure the user applies to the walker. It tracks speed and distance and also uses cameras and AI to analyze gait, which exceeds the project's requirements. Lastly, the Camino far exceeds the budget at $2,999. The shortcomings of this commercially available device underscore the need for a cost-effective, clinically focused smart walker that provides accurate, real-time data to both patients and clinicians.

Problem Statement

Dr. Daniel Kutschera, a physical therapist specializing in neurorehabilitation, requires objective, real-time data on walker use to guide therapy and meet Medicare documentation requirements. Today, these metrics are gathered manually (using a wheel and a stopwatch), which does not quantify load, making measurements inconsistent and difficult to track. In previous iterations of the project, pressure-sensing sensors were integrated into the walker's frame by cutting the walker's legs, compromising its safety and usability. This demonstrates a need for a small, lightweight, clip-on module for standard walkers that displays speed, distance, and the amount of force the user applies to the walker in real time, saves a brief session summary after each use, and doesn’t alter how the walker is used or folded. Our main constraints are that the device must be low profile, accurate, and easily mounted/unmounted. The budget to complete the project is $500.

Background

Following a traumatic brain injury (TBI), walking speed is often referred to as the “sixth vital sign” because it provides an objective measure of functional recovery [5]. Gait speed encompasses the combined performance of multiple physiological systems, including balance, coordination, and muscle strength. Along with speed, the pressure applied to a walker is an essential indicator of patient stability, confidence, and weight-bearing ability, all of which are used to assess readiness for discharge from inpatient care [6]. Clinicians need to know how much weight patients can bear on a walker, especially if the patient is attempting to regain the ability to walk without one. When patients wish to discontinue use of a walker, clinicians must determine whether it is safe to do so. If patients discontinue use of a walker before they are ready, it may lead to reduced walking throughout the day, which can increase risks such as decreased blood flow, constipation, and stiffness [7].

Currently, the most effective methods for clinicians to measure progress are manual, using stopwatches and tape measures to record speed and distance; however, there is no reliable method to record pressure data. There are many problems with this method; it is time-consuming, subjective, and lacks real-time feedback. A system that continuously measures walking speed, distance, and applied pressure would allow for faster and more objective assessment of a patient’s progress during rehabilitation.

To address this clinical need, we have been tasked with developing a device that incorporates integrated sensors to measure pressure, speed, and distance without compromising the walker's structural integrity. Previous versions of this project at UW-Madison have failed due to instability in the custom walker's frame, creating liability and safety risks. Our new design will focus on non-invasive, removable attachments that are accurate, consistent, and readily sanitized for use in clinical settings. The attachments must be compact and lightweight, compatible with other existing two-wheeled walkers, and capable of repeated use.

Key Design Specifications (see full PDS in Appendix A):

- Tracks pressure, speed, and distance accurately and consistently
- Reliable for at least 10 meters of travel and 30 minutes of use
- Removable and compact attachments that do not interfere with the walker
- Easily sanitized between patients
- Compatible with existing two-wheeled walkers
- Supports up to 140 kg (≈300 lb) of patient weight [8]
- Follow all ISO/FDA legal standards [9][10]

Preliminary Designs

During the preliminary design phase, the team focused on two main aspects of the project: the load cell housing and the distance/speed sensor. The Tennis Baller, Endcap 2.0, and Handgripper are variations of encasing the load cells that will measure the pressure placed on the walker by the user. The following section provides an overview of the options the team considered for measuring speed and distance, including infrared, Rotary Encoder, and LiDAR sensors.

Load Cell Housings

Design 1: Tennis Baller

The Tennis Baller design utilizes a pancake load cell encased in a custom 3D-printed piece. It has a larger base for added stability during use and is installed by placing it on the end of one of the walker's back legs, replacing the existing end cap. Then, when the patient puts force on the walker, the load cell will measure the ground force as a way of measuring the weight being placed on the walker.

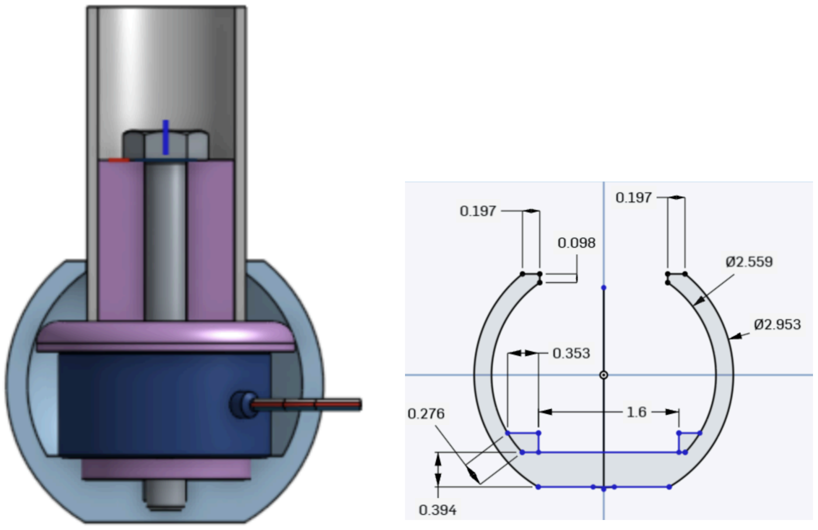


Figure 1: *Tennis Baller CAD Model and Dimensions*

Design 2: Endcap 2.0

The End-Cap design utilizes a load cell housed inside a custom 3D-printed end cap. It is used by placing the device on the end of one of the walker's back legs, replacing the existing end cap. When the patient puts weight on the walker, the load cell measures the ground reaction (vertical) force, which we use to determine the weight being applied to the walker. The purpose of this design is to differ only slightly from the current end caps already incorporated into the walker, keeping it as similar as possible, with the only modification being the addition of a load cell to the end cap. See Appendix B for specific measurements.

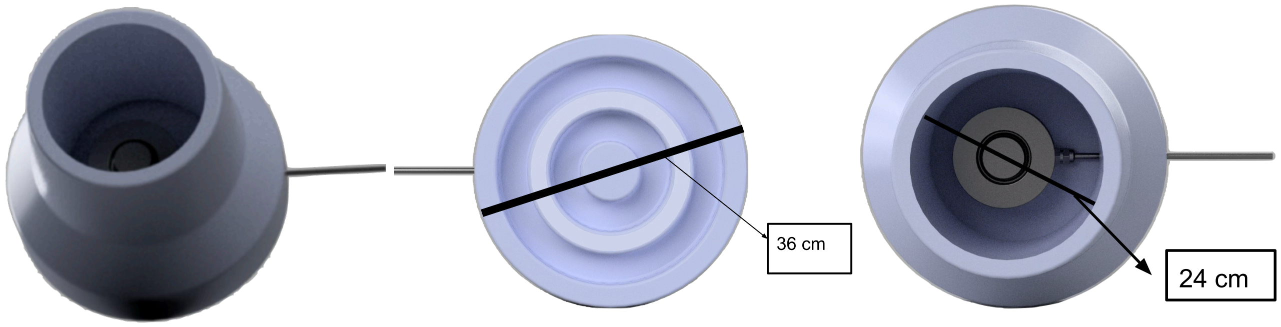


Figure 2: *End-Cap 2.0 CAD Models*

Design 3: Hand Gripper

The Hand Gripper design employs a load cell embedded within a 3D-printed double-shell clamp that encircles the walker’s grips. It installs on the handles (one unit per side). When the patient presses on the handles, the load cell measures the force applied at each hand. If we sum the two readings, we obtain the total weight the patient is bearing on the walker.

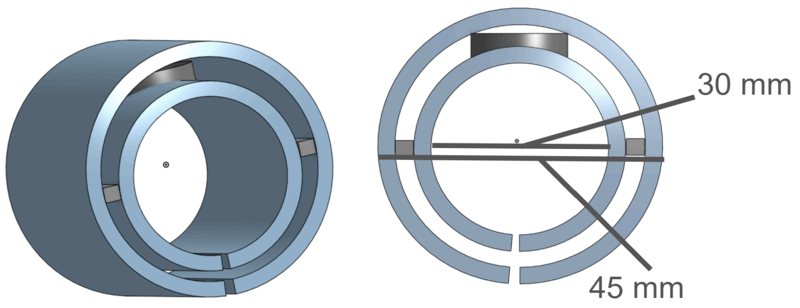


Figure 3*: Hand Gripper CAD Design and Dimensions*

Distance/Speed Sensors

Design 1: Infrared Sensor

The infrared sensor would be mounted at the bottom of one of the walker's legs and would track the rotation of the walker's wheel. The wheel will have a white tape strip, and each time the white strip passes the sensor, the sensor will track rotation and use this to calculate the walker's average speed.

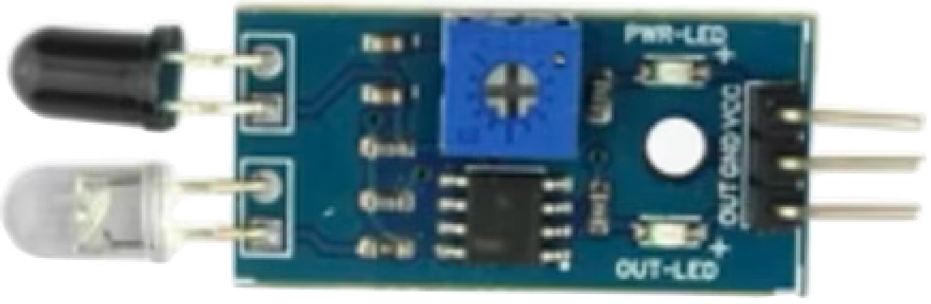


Figure 4: *Infared Sensor*

Design 2: Rotary Encoder

The rotary encoder sensor tracks the wheel's rotation. The encoder will be mounted on the walker's wheel and will track the wheel's axle rotations. The number of rotations can be used to determine the walker's distance and speed.

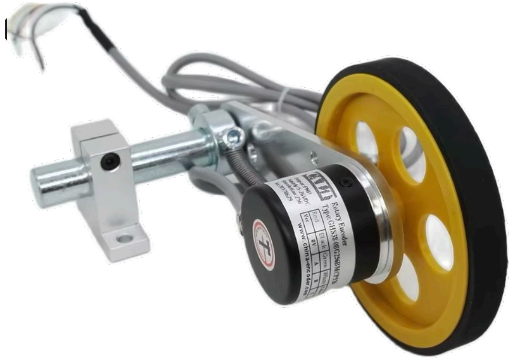


Figure 5: *Rotary Encoder Sensor*

Design 3: LiDAR Sensor

The LiDAR sensor will be placed in a box that will be mounted on the front of the walker. The LiDAR will measure the distance between the walker and the wall at a frequency of 2 MHz. It can track the rate of change in distance, which can be used to calculate speed and the distance traveled.

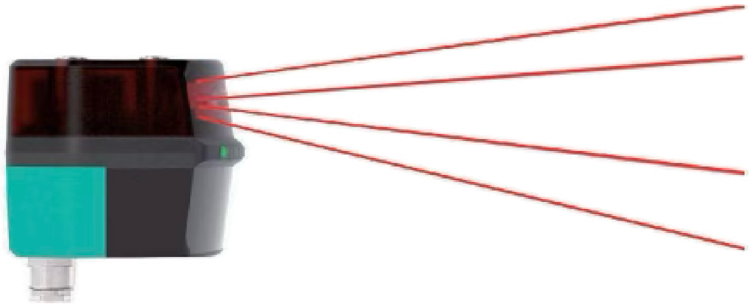


Figure 6: *LiDAR Sensor*

Preliminary Design Evaluation

Load Cell Housing Design Matrices

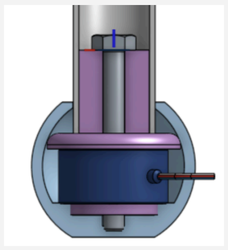
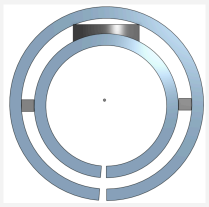
		 1. Tennis Baller		2. End-Cap 2.0		 3. Hand Gripper	
Criteria	Weight	Score	Weighted Score	Score	Weighted Score	Score	Weighted Score
Accuracy of Data	25	5/5	25	5/5	25	3/5	15
Simplicity	25	4/5	20	5/5	25	4/5	20
Usability	20	5/5	20	5/5	20	3/5	12
Safety	15	4/5	12	5/5	15	3/5	9
Ease of Set-up	15	3/5	9	3/5	9	5/5	15
Total (Out of 100):		86		94		71	

***Table 1**: Design Matrix Load Cell Housing*

Accuracy of Data (25):
Both the Tennis Baller and End-Cap 2.0 received a full score of 5/5, indicating that these designs can reliably measure weight distribution and walking metrics with minimal error. The Hand Gripper, however, received a 3/5 rating because its accuracy is compromised when the user squeezes the handle incorrectly. This reduces the Hand Gripper's reliability for collecting consistent rehabilitation data.

Simplicity (25):
The End-Cap 2.0 received the highest score of 5/5 due to its straightforward design and ease of integration onto the walker. The Tennis Baller also performed well, earning 4/5, though its slightly more complex structure makes it less seamless. The Hand Gripper also scored 4/5, but its added components make it less intuitive for consistent use.

Usability (20):
Both the Tennis Baller and End-Cap 2.0 scored perfectly (5/5), as they can be used naturally during walking without altering how a patient interacts with the walker. The Hand Griper scored only 3/5 due to its reliance on a squeezing action, which could reduce stability and make the walker less functional in real-world rehabilitation settings.

Safety (15):
The End-Cap 2.0 received the highest score of 5/5 because it maintains full surface contact with the floor and does not compromise the walker's stability. The Tennis Baller followed with a 4/5, as reducing the surface area could increase the risk of tipping. The Hand Gripper scored the lowest (3/5) because its design could compromise grip stability and raise safety concerns during patient use.

Ease of Set-up (15):
The Hand Gripper scored the highest (5/5), since it can be easily attached without extensive modifications. Both the Tennis Baller and End-Cap 2.0 received a score of 3/5 because they require more effort to install correctly on the walker's legs.

Total Scores (100):
The End-Cap received the highest score of 94/100, primarily due to its high accuracy, simple design, and ease of integration with the walker. The Tennis Baller was assigned a slightly lower score of 86/100 because, although it performed similarly in accuracy and usability, it scored

lower in safety due to its reduced bottom surface area, which could reduce stability. The Hand Gripper received the lowest score of 71/100 because even though it will be easy to attach, it will cause a decrease in grip stability and could produce inaccurate data readings if squeezed incorrectly, which lowered its safety and usability scores. Overall, the End-Cap design was our best choice due to its reliability, simplicity, and compatibility with existing walker components.

Distance/Speed Sensor Design Matrices

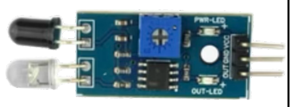
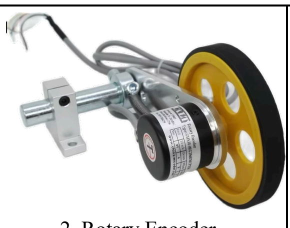
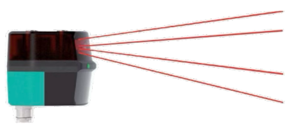
		 1. Infrared Sensor		 2. Rotary Encoder		 3. LiDAR Sensor	
Criteria	Weight	Score	Weighted Score	Score	Weighted Score	Score	Weighted Score
Accuracy of Data	25	3/5	15	5/5	25	4/5	20
Structural Impact	20	4/5	16	1/5	4	5/5	20
Safety	15	5/5	15	3/5	9	5/5	15
Reliability of Sensors	15	3/5	9	5/5	15	4/5	12
Ease of Set-up	15	3/5	9	1/5	3	4/5	12
Cost	10	5/5	10	3/5	6	2/5	4
Total (Out of 100):		74		62		83	

***Table 2**: Design Matrix Movement Sensors*

Accuracy of Data (25):
The rotary encoder scored the highest (5/5) for accuracy, as it provides precise measurement of distance and rotations. The LiDAR also performed strongly (4/5), offering accurate distance and speed tracking with minimal error. The infrared sensor scored the lowest (3/5), as it is less consistent and can be influenced by environmental conditions.

Structural Impact (20):
The LiDAR scored best (5/5) because it mounts externally and does not interfere with the walker's stability or structure. The Infrared sensor also performed well (4/5), being small and lightweight, although it still affected the structure more than the LiDAR, due to its position next to the wheels. The rotary encoder scored lowest (1/5) because it requires direct attachment to a wheel, thereby compromising structural simplicity.

Safety (15):
Both the LiDAR and infrared sensors received a score of 5/5 because they pose minimal risk to walker safety. Their designs permit minimal modification to the walker's critical areas. The Rotary Encoder received a score of 3/5 due to its mechanical integration, which introduces potential hazards if not properly secured.

Reliability (15):
The rotary encoder scored the highest (5/5) since it provides consistent, reliable readings. The LiDAR followed with a strong 4/5 rating, although it can be sensitive to reflective surfaces. The Infrared sensor scored the lowest (3/5) because it is prone to inaccuracies under varying lighting and surface conditions.

Ease of Set-up (15):
The LiDAR (4/5) and infrared (3/5) are both relatively simple to attach without significant modifications. The rotary encoder scored the lowest (1/5) because it requires precise mechanical integration, which complicates setup.

Cost (10):
The infrared sensor scored the highest (5/5) as the most affordable option. The rotary encoder received a moderate score (3/5), whereas the LiDAR received the lowest score (2/5), reflecting its significantly higher cost.

Total Scores (100):
Overall, LiDAR achieved the highest score of 83/100 due to its high accuracy/reliability, minimal structural impact, and straightforward setup. One of its main drawbacks is its high cost.

Infrared ranked second with a score of 74/100 because it is inexpensive, safe, and low-impact, but it has only moderate accuracy/reliability. Lastly, the rotary encoder received the lowest score, 62/100. This is because, although it offers the highest accuracy and reliability, its drawbacks include structural impact, setup complexity, and safety concerns. Overall, LiDAR provides the best total performance despite its high cost.

Proposed Final Design

The final prototype we have selected, after reviewing the design matrix, is a combination of the LiDAR sensor and a design modeled after End-Cap 2.0. The LiDAR will be mounted near the handlebars and will point forward to measure the distance to a wall or object. We calculate speed by measuring how the distance changes over time. The placement stays out of the user's way and does not change how the walker is used. The End-Cap 2.0 add-ons will wrap around the walker leg and tighten in place, ensuring secure placement and easy removal. A Lego-like insert fits within the leg and applies force to a load cell. This creates a direct force path, improves consistency, and keeps the footprint close to a standard tip.

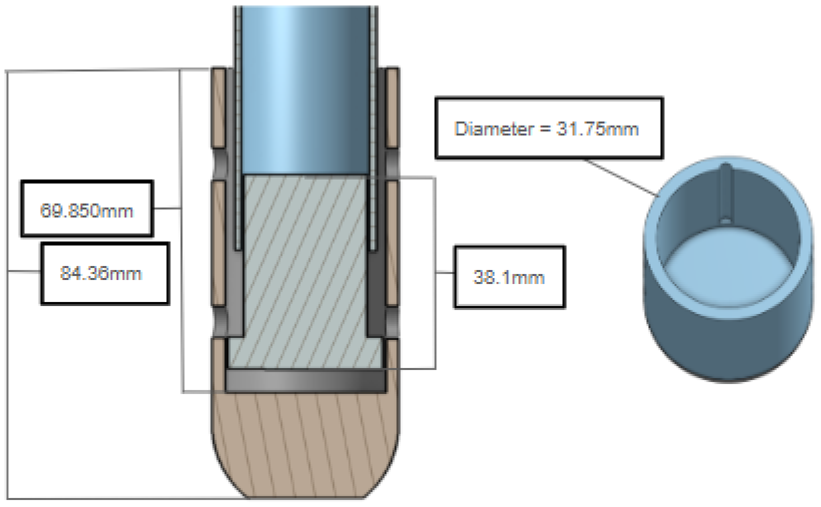


Figure 7: *End-Cap 2.0 CAD drawing with dimensions.*

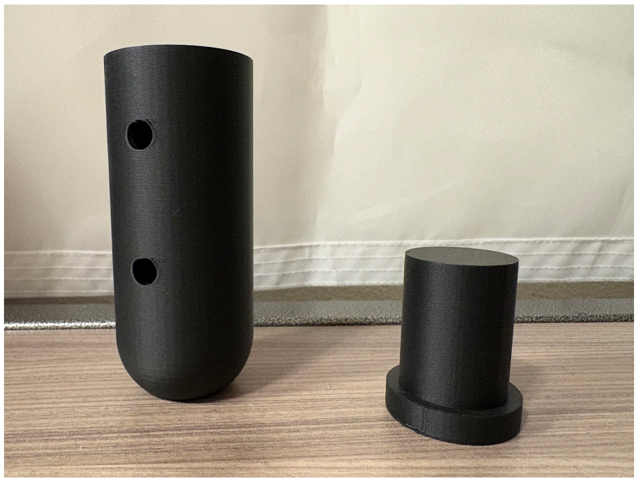


Figure 8: *End Cap 2.0 initial prototype*

Fabrication

Overview

The Smart Walker comprises two main categories of components and fabrication methods. The first being the circuitry components and circuitry design. The circuitry components include two sensor systems: one for the load cells to detect weight-bearing, and one for the LiDAR system to measure distance and speed. The second aspect is the 3D-printed components to house the sensor systems: one is the End Caps, which hold the load cells, and the other is the electrical housing box, which contains the LiDAR, Arduino microcontroller, and perfboard.

Materials

For the weight-tracking component of the design, two load cells will be used. The load cells work by tracking the displacement of the center region with the outside lip of the load cell, which changes the electrical resistance measured by the Wheatstone bridge circuit, producing a voltage proportional to the force. Figure 9 shows where the load cells get compressed and displaced. These load cells were chosen because they are very accurate for the price and can withstand up to 75kgs (165lbs) each for a total of 150kgs (330lbs) for two load cells, which meet the required weight of 140kgs. The load cells will be placed in the endcap design, which will be printed in Thermoplastic Polyurethane (TPU) with 50% infill. This material is flexible enough to fit the walker leg insert and the leg itself, but also durable enough to withstand the weight of the patient and walker. This material closely resembles the material of the end caps that a base walker comes with. So the load cell can be compressed properly, a walker leg insert will be made out of Polyvinyl Alcohol (PLA) with 50% infill. This is a hard, rigid material that is needed so the load cell is properly depressed by this piece. It is also a very durable material that can withstand constant pressure. To ensure the load cells send a strong enough signal, a load cell amplifier HX711 will be used. The amplifier takes in the weak or small voltage output from the load cells and amplifies the signal to a higher voltage. It also gets rid of electrostatic noise for a cleaner signal output. Because this design includes two load cells instead of four, a half-bridge configuration, seen in Figure 10, will connect them to the amplifier.

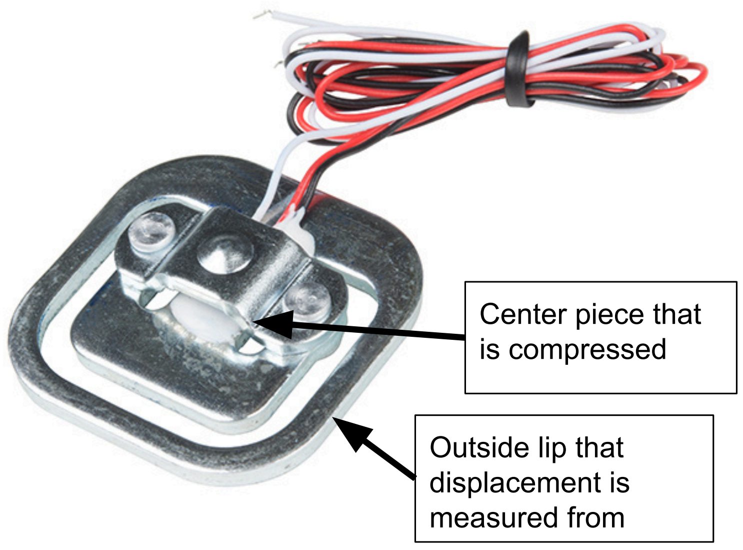

Center piece that is compressed
Outside lip that displacement is measured from

Figure 9: *Geekstory 50 kg Capacity Load Cells*

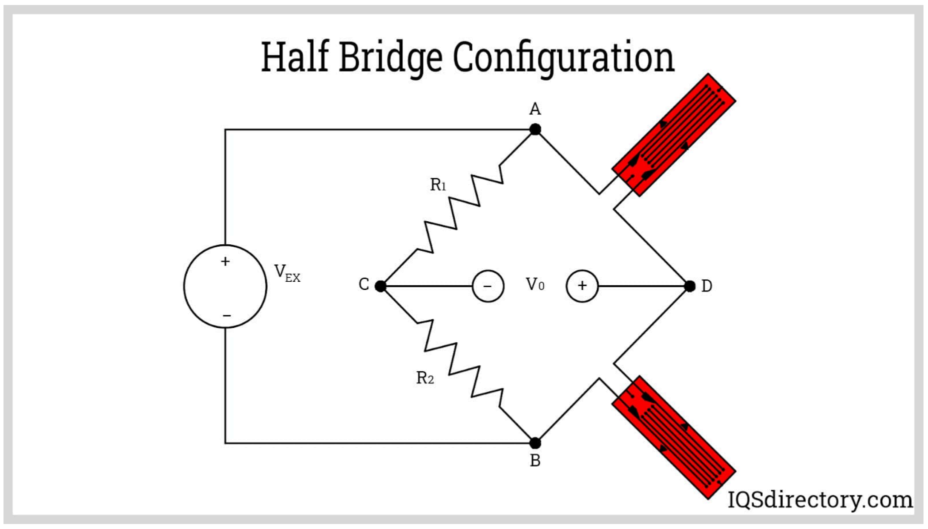


Figure 10: Half wheatstone bridge configuration showing strain gauges in red [11].

For the speed and distance portion of the design, the Garmin LIDAR-Lite Optical Distance Sensor - V3m, seen in Figure 11, will be used. The LiDAR emits a 905nm single-stripe laser that bounces off the wall/obstacle in front of it and back to the LiDAR. The time it takes for the laser to come back to the LiDAR is how it tracks distance, which can be used to compute speed. The LiDAR will be placed in the electrical component box. The component box will also be made out of PLA with 15% infill. This rigid nature of PLA will help protect the LiDAR and all the electrical components inside the component box. The box will also hold the perforated(perf) board and ARDUINO UNO WiFi REV2 microprocessor. See Appendix C thru E for detailed blueprints of all 3D prints. All 3D printed materials were designed in OnShape.

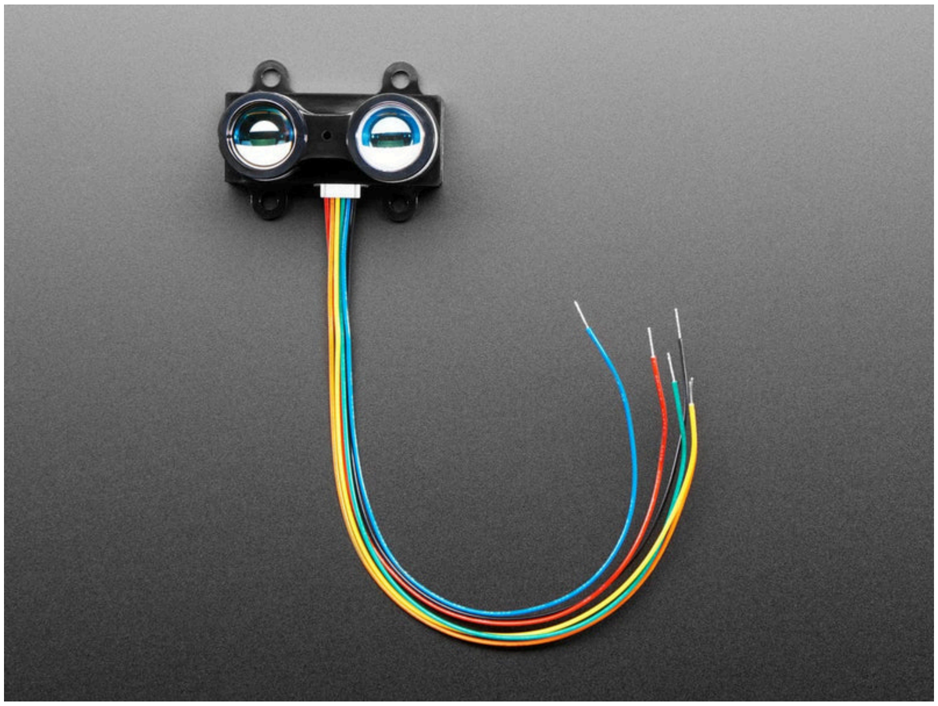


Figure 11: *LiDAR sensor*

A Duracell Procell 9V Alkaline Battery will serve as the power source for the device and be held in a Gikfun 9V Battery Holder with ON/Off Switch for Arduino. The battery housing has a barrel jack that plugs directly into the barrel jack on the Arduino. Figure 12 below shows a schematic of the circuit and components of the design.

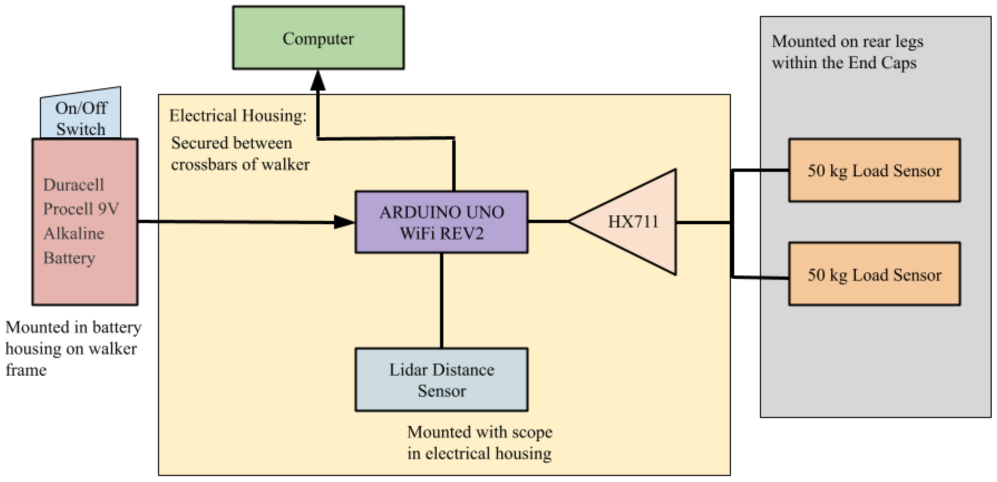

Computer
Mounted on rear legs within the End Caps
On/Off Switch
Electrical Housing:
Secured between crossbars of walker
Duracell Procell 9V Alkaline Battery
ARDUINO UNO WiFi REV2
HX711
50 kg Load Sensor
50 kg Load Sensor
Mounted in battery housing on walker frame
Lidar Distance Sensor
Mounted with scope in electrical housing

Figure 12: *Circuit block diagram of design and components.*

Methods

For the weight tracking circuit, the load cells will be connected directly to the HX711 load cell amplifier. The load cells have three wires coming from them: black, red, and white. The black wire is the negative wire, the white is the positive wire, and the red is the signalling wire. The HX711 amplifier board has two sides: one side that connects wires to the load cells and the other side that connects wires to the Arduino. The side that connects to the load cells has connection points E+, E-, A+, A-, B+, and B-. The side that connects to the Arduino has connection points GND, DT, SCK, and VCC. One load cell is connected to the HX711 amplifier by connecting the positive wire to the E+ connection, the negative wire to the E- connection, and the signalling wire to the A+ connection. The other load cell has the positive wire connected to the E- connection, the negative wire connected to the E+ connection, and the signalling wire connected to the A- connection. For the connection from the HX711 amplifier to the Arduino, the VCC connection is connected to the 5V port on the Arduino, the GND connection is connected to the GND port, the DT connection is connected to the 3 Digital port, and the SCK connection is connected to the 2 Digital port. See Figure 13 for a visual representation of the circuit. The load cells were calibrated using the code "SparkFun_HX711_Calibration.ino" from Sparkfun's HX711 Breakout Hookup Guide [12] found in Appendix G. Seeing as the load cell provides output based on the deformation of the center square of the load cell from the outer rim, while calibrating, they were attached to wooden boards with 3D printed mounting pieces from Thingiverse [13] that provide a spacer between the outer rim and the board. This way, the center piece was free to deform [14].

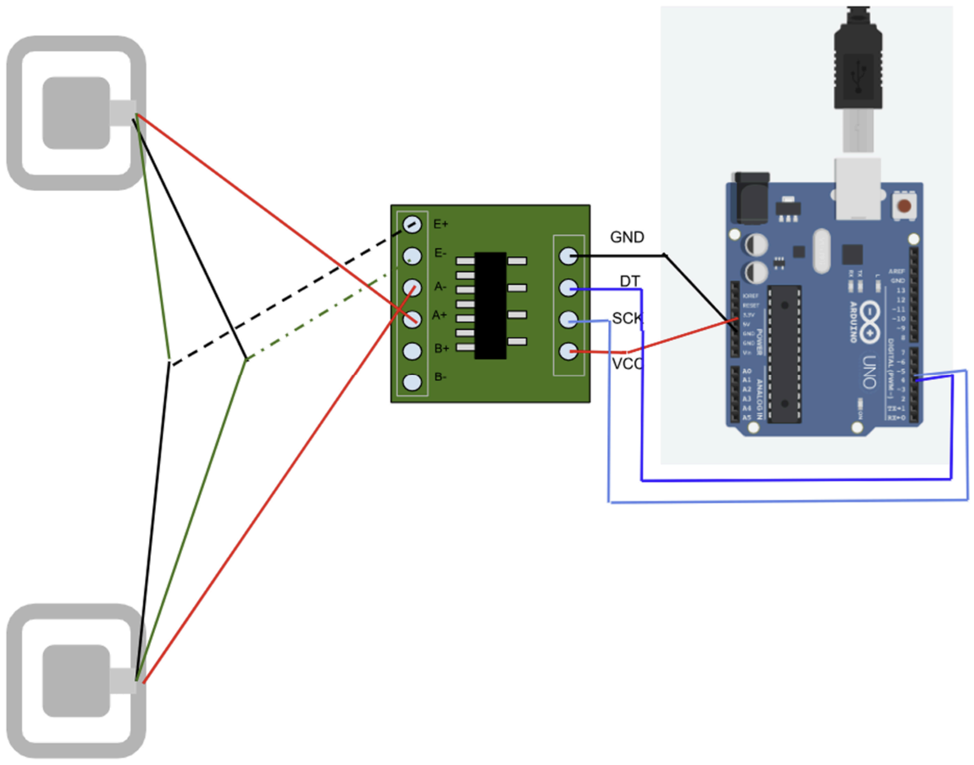


Figure 13: *Load cell circuit to Arduino [15]. The green wire is the white wire.*

For the speed and distance tracking circuit, the LiDAR wires are connected directly to the Arduino. The LiDAR has a power wire (red), a power enable wire (orange), a mode control wire (yellow), a clock power wire (green), a data power wire (blue), and a ground wire (black). The orange and yellow wires are not connected to any part of the circuit. A 1000 µF capacitor and two 4.7 kΩ resistors are needed in the circuit. The hookup guide for the LiDAR circuit was found on Sparkfun's LIDAR-Lite v3 Hookup Guide [16]. See Figure 14 for the full speed and distance tracking circuit.

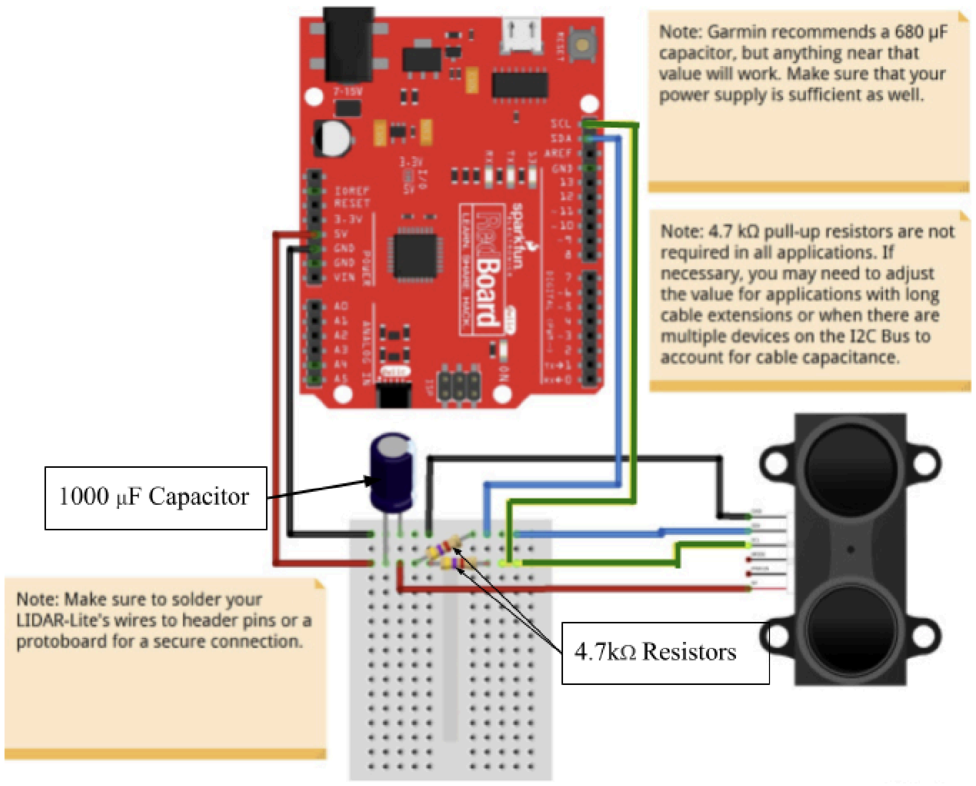

Note: Garmin recommends a 680 µF capacitor, but anything near that value will work. Make sure that your power supply is sufficient as well.
Note: 4.7 kΩ pull-up resistors are not required in all applications. If necessary, you may need to adjust the value for applications with long cable extensions or when there are multiple devices on the I2C Bus to account for cable capacitance.
1000 µF Capacitor
4.7kΩ Resistors
Note: Make sure to solder your LIDAR-Lite's wires to header pins or a protoboard for a secure connection.

Figure 14: *LiDAR circuit to Arduino [16]*

The combined circuit is soldered together on a perf board and then connected to the Arduino. The two circuits are completely separate on the perf board except for each circuit's power wire that connects to the 5V port on the Arduino board. Each circuit's power wire connects to a single wire that is connected to the 5V Arduino port. The complete circuits on the perf board connected to the Arduino can be seen in Figure 15.

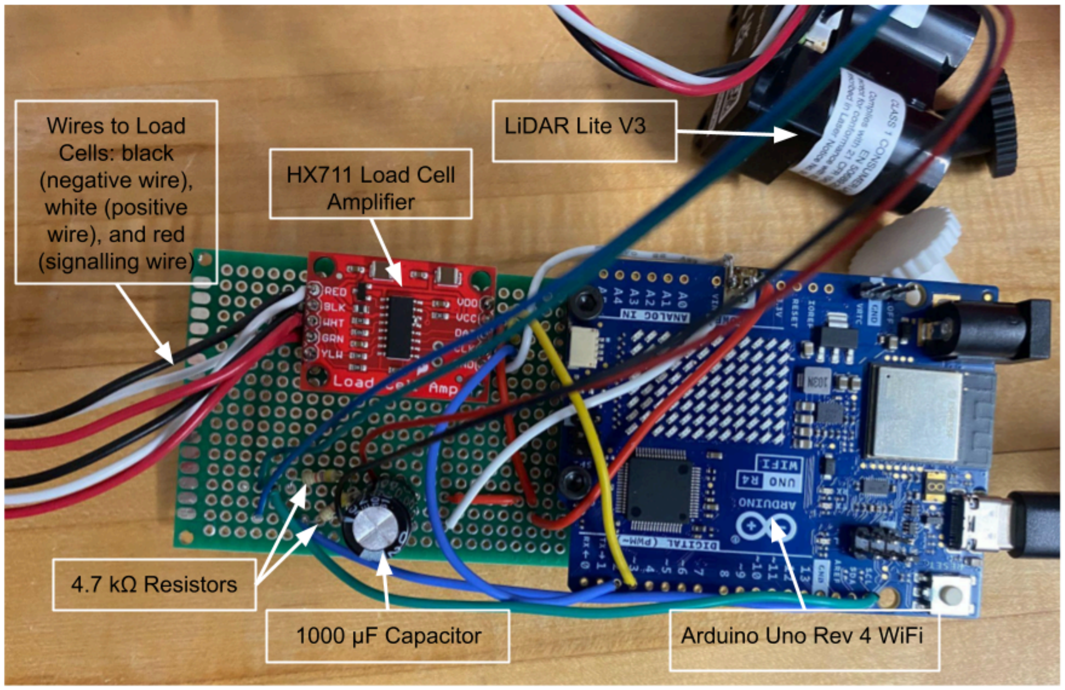
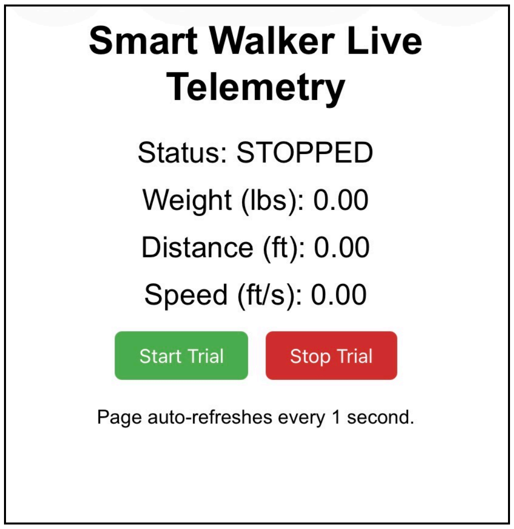

Wires to Load Cells: black (negative wire), white (positive wire), and red (signalling wire)
HX711 Load Cell Amplifier
LiDAR Lite V3
4.7 kΩ Resistors
1000 μF Capacitor
Arduino Uno Rev 4 WiFi

Figure 15: *Complete combined circuit on perfboard*

Software

The team first considered using MIT's App Inventor, but upon further research discovered that Arduino Uno R4 WiFi transmits Bluetooth Low Energy as opposed to classic Bluetooth (HC-05/06); therefore, the two are not compatible without additional hardware. The team then pivoted to a simple WiFi-based web browser. Code for the web page can be found in Appendix C. The webpage url is http://192.168.4.1/ and the device being used must be connected to the Smart_Walker wifi. See Appendix G for a more detailed connection guide. The webpage includes outputs of pressure/weight placed on the walker in pounds, the distance from the nearest surface in feet, and the speed at which the walker is moving in feet per second. The visual display can be seen in Figure 16Due to the large volume of data coming from the serial output of the Arduino, the webpage automatically refreshes every second.

The code for the load cells was also found on Sparkfun's HX711 Breakout Hookup Guide [12]. See Appendix H for the full code. This code was used to test the load cells and confirm that they functioned properly. The code for the LiDAR was found on the Garmin LIDARLite_Arduino_Library GitHub [17]. See Appendix I for the entire code. This code was also used for testing and to confirm the LiDAR worked correctly. Both of these codes were later adapted and combined along with the code for the webpage to make the final code that was uploaded to the Arduino. This code can be found in Appendix C.


Smart Walker Live Telemetry

Status: STOPPED

Weight (lbs): 0.00

Distance (ft): 0.00

Speed (ft/s): 0.00

Page auto-refreshes every 1 second.

Figure 16: *Webpage for data collection*

Final Prototype

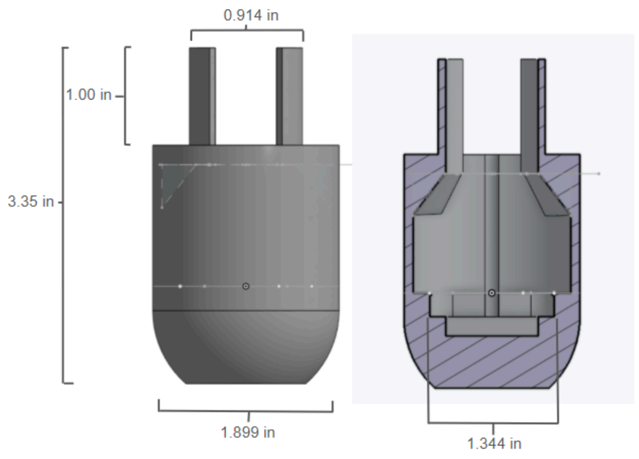


Figure 17: *Final CAD End Cap 3.0 Design*

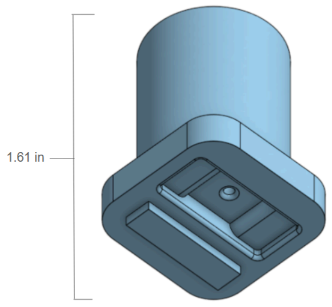


Figure 18: *Final CAD Cork Piece Design*

Load Cell Housing Explanation

The End Cap 3.0 design (Figure 17) securely holds the 50-kg load cells on the back two legs of the walker. Each load cell snaps into the bottom of the 3D-printed end cap and rests on an internal ridge, which ensures that the center sensing region remains free to deform properly when force is applied. The design also incorporates the extruded, trunk-like features at the top of the end cap, which wrap around the walker leg. These features provide surfaces for an external clamp, allowing the end cap to be fastened tightly to the leg. Although the end cap fits securely on its own, the area to add a clamp adds an extra layer of stability and helps ensure that the load cell and its housing remain firmly attached during repeated clinical use

To compress the load cell in a consistent and repeatable way, a 3D-printed cork piece (Figure 18) fits into the bottom of the walker leg and presses directly onto the center pad of the load cell. The geometry of the cork piece is specifically shaped so that the applied force is distributed only to the central sensing region (the part of the load cell designed to deform) while avoiding pressure on the outer frame. This design is intended to create a controlled force path, allowing the load cells to measure weight accurately and consistently during walker use.

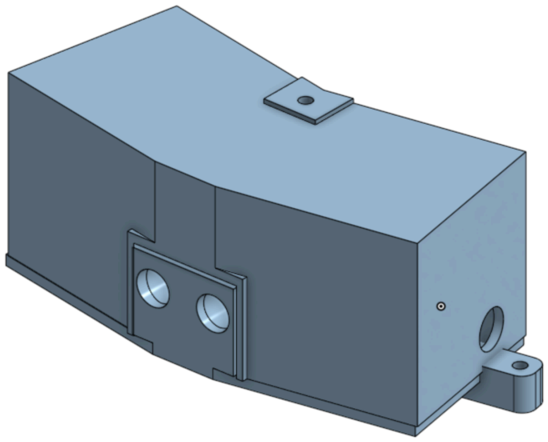


Figure 19: *Final CAD Electronic Housing + LiDAR Box*

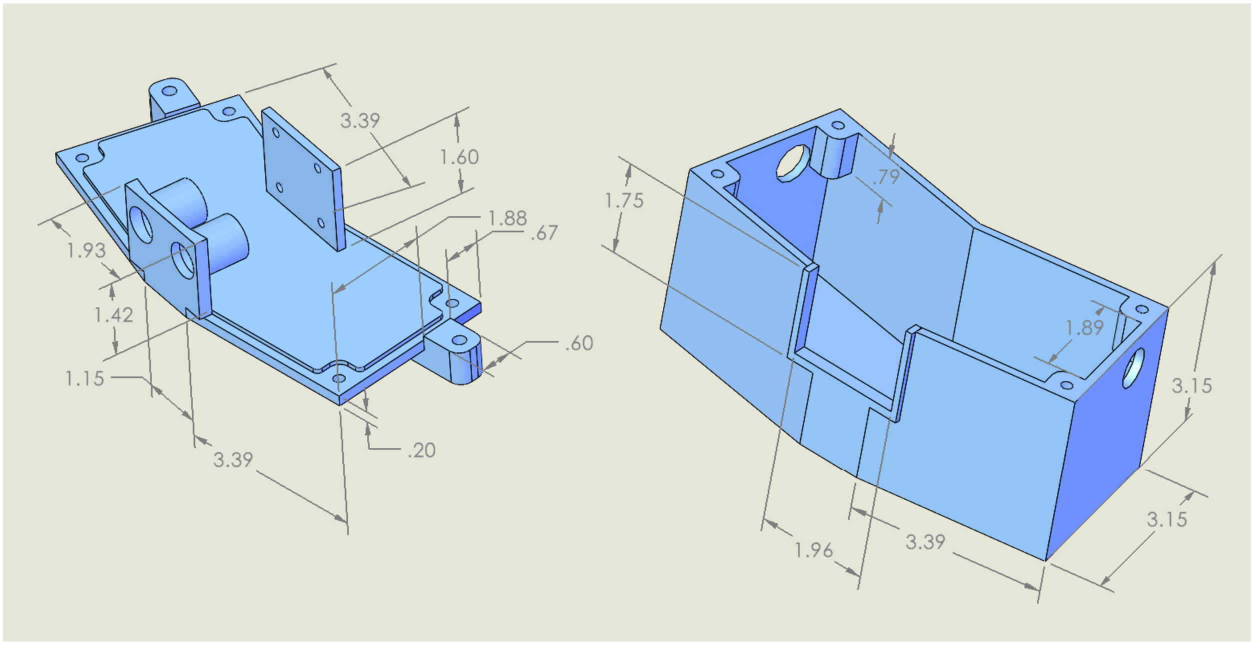


Figure 20: *Final CAD Electronic Housing + LiDAR Box Design with Dimensions in inches*

Electrical and LiDAR Housing Explanation

The electrical and LiDAR housing design (Figure 17) is mounted to the front of the walker and serves as an enclosure for the Lidar sensor and all other electrical components. The housing includes 3 mounting holes that allow external attachment clamps to secure the housing box to the frame of the walker. Inside, the electronic components are supported by a base piece (Figure 18). The top cover attaches to this base with screws at each corner creating a fully enclosed and protected compartment. The electrical housing is angled at 158.9 degrees along its longer sides to match the geometry of the walker frame and be as noninvasive as possible.

The LiDAR sensor is held in place through 4 screws that attach the LiDAR's rear mounting points to the vertical bracket shown in Figure 18. When installed, the LiDAR's lenses line up with the two cylindrical tubes to allow it to look out the front of the housing box. LiDAR sensors work by emitting laser pulses in a cone like shape, so the tubes help focus the scope of our sensor allowing it to emit a more focused beam. This eliminates unwanted peripheral detections and improves performance in hallways or areas with many other objects.

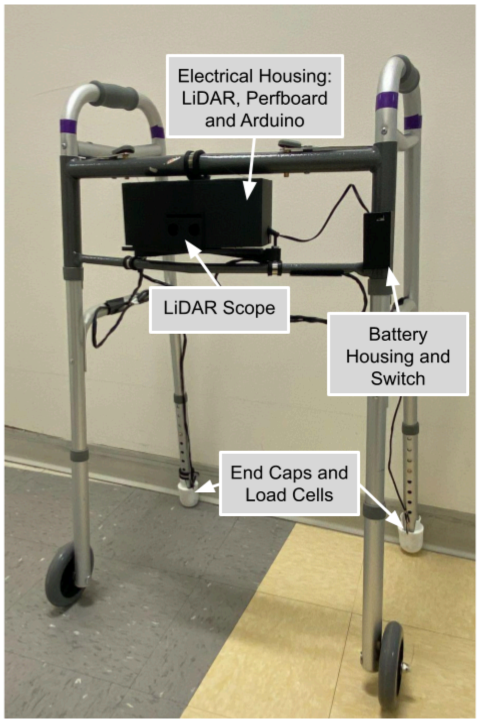

Electrical Housing: LiDAR, Perfboard and Arduino
LiDAR Scope
Battery Housing and Switch
End Caps and Load Cells

Figure 21: *Final Assembled Prototype*

Testing and Results

Load Measurement Testing:

To verify the accuracy of our calibrated load-cell system, outputs were compared with a set of standard weights of a known accuracy. The load cells were tested together using a calibration value of -18670 and placed flat on the ground on a hard, level surface to minimize variation due to uneven loading. Accuracy was assessed using linear weight increments of 5kgs (11.2 pounds), resulting in the following test loads: 22.05 lb, 33.07 lb, 44.09 lb, 55.12 lb, 66.14 lb, 77.16 lb, 88.18 lb, 99.21 lb, 110.23 lb, 121.25 lb, 132.28 lb, 143.3 lb, 154.32 lb, 165.35 lb, 176.37 lb, 187.39 lb, 198.42 lb, 209.44 lb, and 220.46 lb. For each load level, the appropriate combination of weight plates was carefully and centrally on the two sensors so that the applied force was evenly distributed across both sensors. After allowing several seconds for the load-cell reading to stabilize, the load cell's output was recorded. The paired load cell and reference readings from each trial were used to calculate absolute error, percent error, and variability across the full operating range, allowing us to determine how well the load-cell system performs under controlled loading conditions.

Once all measurements were collected, the accuracy of the load-cell system was evaluated by directly comparing the walker's recorded values to the corresponding known values. For every load level, we calculated the absolute error (walker reading minus reference reading) and the percent error relative to the true load. To visualize performance across the full range, we generated a plot comparing the walker's measured load to the reference values. The load cells remained accurate until 165 lbs when the readings leveled off. The result was an average error of 5.51%, but only 1.68% error up to 165 lbs.

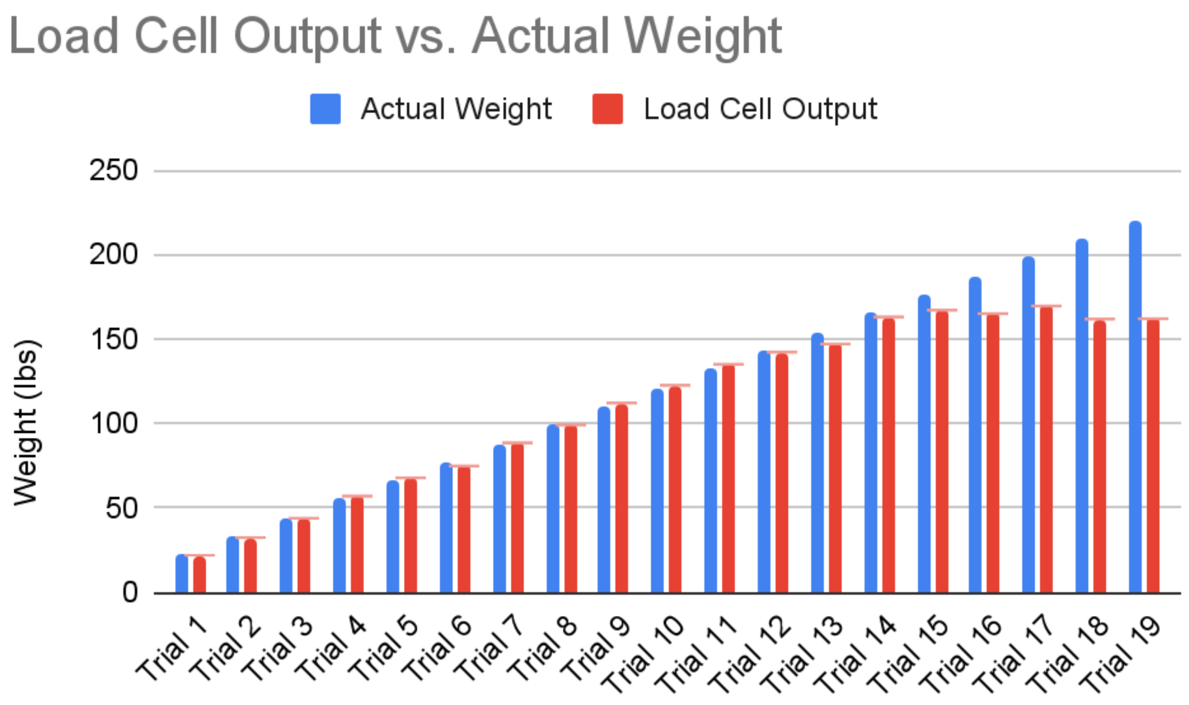


Figure 22: *Relationship between actual weight and load cell recorded weight, the blue bars represent the known weight placed on the load cells, the red bars are the corresponding readings given by the load cells, and the vertical error bars indicate the constant ± 0.056 lb potential error associated with the sensors.*

LiDAR Testing:

To evaluate the accuracy and reliability of the LiDAR distance and speed measurements, we performed a series of walking trials in a long hallway inside the UW–Madison Engineering Centers Building. This environment was not fully controlled, as students occasionally walked past or crossed the testing path, which reflects realistic clinical conditions in which the device may also be used. The LiDAR sensor remained mounted in its permanent configuration inside the electronic housing box, ensuring that all testing reflected its true operational setup rather than an idealized bench-test configuration.

Before testing, a long tape measure was used to mark reference distances on the hallway floor. Starting at 15 ft (4.57 m) from a flat wall, we placed tape marks at increments of 15 ft up to a maximum distance of 120 ft (36.6 m). These tape marks served as fixed starting positions for each trial. One team member pushed the walker while another monitored the Arduino data stream and managed timing. The walker was kept level, and the LiDAR remained pointed directly at the wall throughout each trial.

For each trial, the walker's front legs were placed exactly on the tape mark corresponding to the starting distance. The operator then pushed the walker normally toward the wall, without attempting to control or standardize walking speed. This approach allowed us to test the LiDAR under realistic variations in gait and movement. While the walker moved, the LiDAR continuously recorded the distance to the wall and calculated speed using the Arduino's internal clock.

At the end of each trial, we compared the final distance measured by the LiDAR to the known starting distance measured manually. This allowed us to compute absolute error and percent error for each of the eight trials. To evaluate speed accuracy, we compared the LiDAR-calculated speed with the manually calculated speed obtained by dividing the known travel distance by the stopwatch time. We then averaged the percent errors across all trials to determine the overall accuracy of the LiDAR system under typical operating conditions. These comparisons allowed us to assess whether the LiDAR performance met the PDS requirements for both distance- and speed-tracking accuracy.

Across all eight trials, the LiDAR distance measurements showed an average percent error of 1.24%, with the majority of trials remaining below 1% error. The two largest errors (2.64% at 105 ft and 2.97% at 120 ft) occurred at the longest distances, suggesting that accuracy decreases slightly as the LiDAR approaches the upper end of its effective range. Overall our average percentage error over 120ft is well within our guidelines for accuracy.

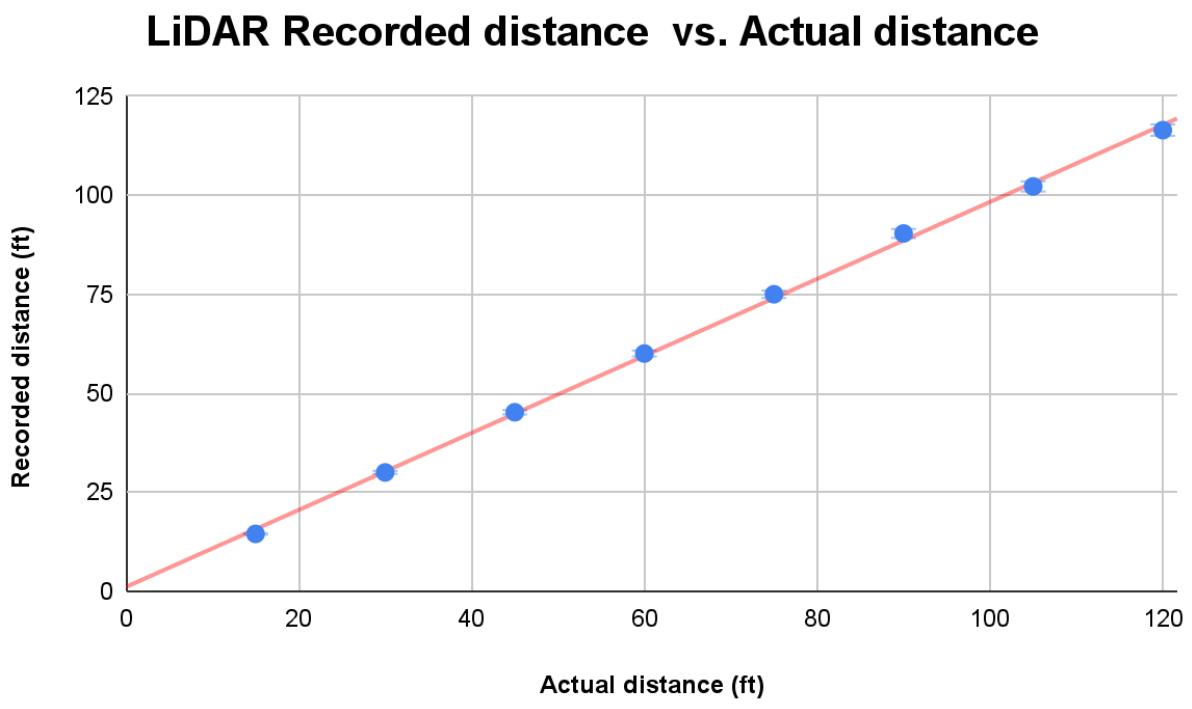


Figure 23: *Relationship between actual distance and LiDAR recorded distance, Blue dots represent individual LiDAR measurements, the red line is the linear best-fit trend, and the vertical error bars indicate the 1.24% distance measurement error associated with the sensor.*

Speed measurements showed similarly strong performance, with an average percent error of 1.38% when compared to manually calculated speeds. Most speed errors were below 1%, and the highest observed error was roughly 3.25% at 120 ft. This again shows that at larger distances accuracy may increase, but overall our average error over 120ft for speed data = is well within our guidelines for accuracy.

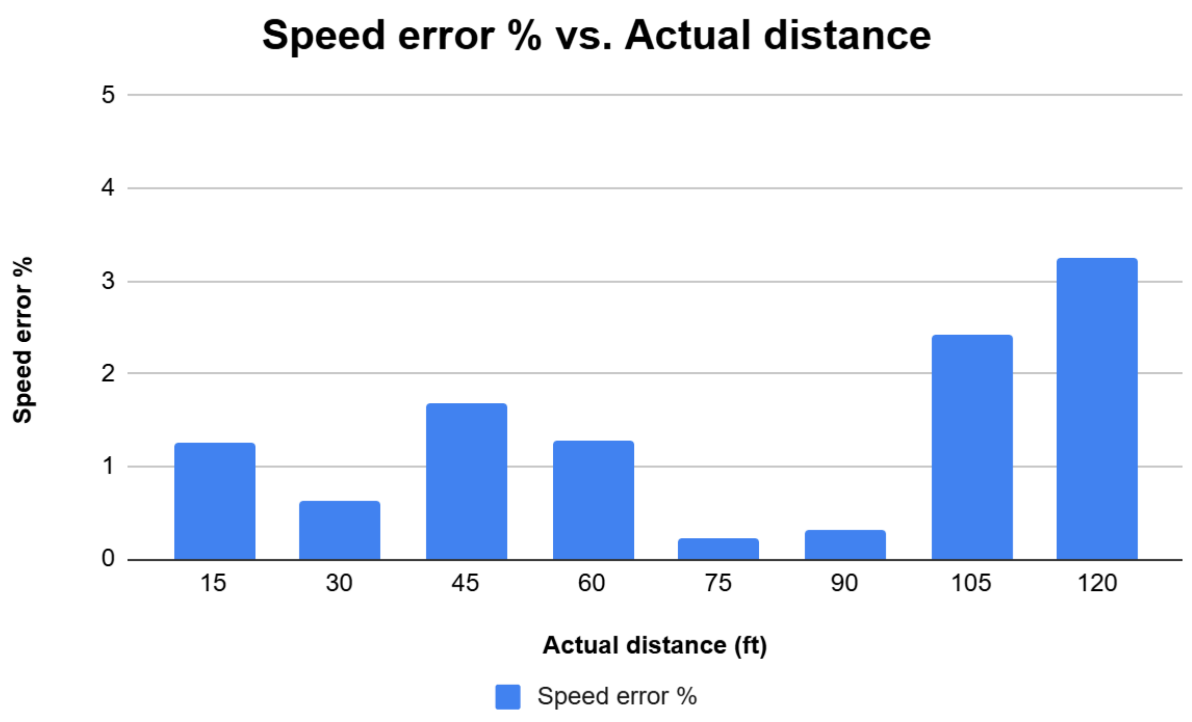


Figure 24: *Relationship between speed error % over certain distances. Blue bars represent error % and are organized into pre-set distances the speed was calculated over.*

Overall, the small magnitude and consistency of both distance and speed errors indicate that the LiDAR system provides stable and reliable measurements under typical indoor walking conditions.

Discussion

Our testing shows the Smart Walker prototype functions as intended and is capable of collecting the basic measurements required by the client. The LiDAR sensor consistently measured distance and produced speed values that matched the manually calculated speeds based on stopwatch timing. This confirms that the LiDAR system is suitable for tracking patient movement during indoor walking trials. Although our results do not attempt to validate clinical rehabilitation outcomes, they show that the sensing system operates reliably enough to support future clinical testing. Additionally, we have shown the accuracy of recording weight with two loads.

Ethically, the main considerations for this device involve patient safety and data handling. The walker add-ons must never compromise user stability, which guided our decisions to keep all attachments removable and be able to securely clamp to the walker frame. When used in a clinic, any recorded data such as speed, distance, or weight bearing would need to be handled in accordance with HIPAA, since it could be linked to patient identity or medical progress [18]. At this stage, our prototype only displays data locally and does not store it, minimizing privacy.

Our testing also identified several areas that require improvement. The LiDAR was accurate up to 120 ft, but our goal is 150 ft, so a higher-range sensor or stronger signal filtering may be needed. The load-cell system was not fully integrated due to a hardware failure in the amplifier, meaning weight measurements could not be tested on the walker itself. Before the device can be considered for clinical use, the load cells must be reinstalled, recalibrated with the additional walker frame weight, and tested under realistic loading conditions. The user interface should also be improved to make data easier for clinicians to interpret at a glance.

Several sources of error were observed. In the LiDAR trials, people passing through the hallway occasionally interfered with the sensor, causing brief spikes in the readings, as well as human error in measurement of pre-set distances, and human error in starting and stopping the stopwatch for timing trials. For the load-cell system, likely sources of future error include uneven weight distribution between load cells. In future testing of load cells, potential sources of error to watch out for could include 3D-printed tolerances that affect force transfer, and electrical noise in the strain-gauge amplifier.

Overall, while the prototype is not yet ready for clinical deployment, the results demonstrate that the core sensing components work as intended and provide a solid foundation for completing the design in future iterations.

Conclusion

The client requested the development of a device that could provide accurate, real-time measurements of patient performance during rehabilitation, specifically tracking pressure, speed, and distance without altering the safety or function of a standard walker. Through evaluation of multiple design options, a modified version of the End-Cap 2.0 load cell housing and LiDAR sensor combination emerged as the most effective solution. These components provide accurate, reliable data while minimizing structural impact and preserving the walker's usability and safety.

The team successfully produced a proof of concept prototype to meet the majority of the client's requirements while maintaining low error. One major improvement that could be made is increasing the range of the LiDAR sensor from 120 ft to the desired 150 ft. This could be accomplished in a multitude of ways, including buying a more expensive sensor with a greater range or turning the LiDAR towards the feet of the patient and employing the more advanced capabilities of the LiDAR to perform more advanced gait analysis in addition to step count and speed. Additionally, due to an error while soldering the circuit, the HX711 amplifier board shorted, and given the timeline constraints, the team was unable to reorder parts. Thus, functioning load cells were never integrated into the final design of the walker. Further testing with functioning load cells while within the walker must be conducted before the device is ready for official use. This is especially important because the load cells need to be recalibrated to account for the additional weight of the frame of the walker, as well as any weight displaced by the front two legs. Lastly, seeing as the device will be used in a clinical setting, a HIPAA-approved method of recording and storing data is needed for the device. With this, an improved user interface should be integrated, as this was a lesser focus during this design process.

The design will enable clinicians to measure patient progress objectively, thereby enhancing both patient motivation and compliance with Medicare documentation requirements. The device is compact, removable, and easily sanitized, making it appropriate for use in a clinical setting. Fabrication focused on lightweight, durable materials, and testing confirmed the accuracy of pressure, distance, and speed measurements under controlled conditions. This project has the potential to transform the way rehabilitation progress is tracked by providing clinicians with quantitative data, thereby reducing their reliance on subjective or manual methods. By providing therapists with a cost-effective and clinically focused tool, the smart walker aims to enhance rehabilitation outcomes and support both patient recovery and healthcare documentation needs.

Acknowledgements

The team extends special thanks to advisor Dr. Duc-Huy Nguyen for his advice throughout the design process, as well as the client Dan Kutschera for his ongoing support and the inspiration behind the design.

References

[1]"Choosing the Right Stroke Rehab Facility," www.stroke.org. Accessed: Oct. 09, 2025. [Online]. Available: https://www.stroke.org/en/life-after-stroke/stroke-rehab/choosing-the-right-stroke-rehab-facility

[2]"Medical insurance credentialing 101: Everything you need to know." Available: https://www.tebra.com/theintake/practice-operations/insurance/medical-insurance-credentialing-everything-you-need-to-know-to-avoid-losing-money-and-clients. [Accessed: Dec. 06, 2025]

[3]"What does Medicare cover after a stroke? | UnitedHealthcare." Accessed: Oct. 09, 2025. [Online]. Available:

https://www.uhc.com/news-articles/medicare-articles/what-does-medicare-cover-after-a-stroke?msockid=07aad215264f65222a5ec73327006463

[4] “Meet Camino,” Camino Mobility. Accessed: Sep. 19, 2025. [Online]. Available: https://caminomobility.com/pages/meet-camino

[5] S. Humphries, “A physical explanation of the temperature dependence of physiological processes mediated by cilia and flagella,” Proc. Natl. Acad. Sci. U.S.A., vol. 110, no. 36, pp. 14693–14698, Sep. 2013, doi: 10.1073/pnas.1300891110. [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC3767513/

[6]“Assistive Device Training: Walkers (Rehabilitation Therapy).” Accessed: Oct. 09, 2025. [Online]. Available: https://elsevier.health/en-US/preview/assistance-device-training-walkers

[7]“Frequently Asked Questions About Walking After Your Surgery | Memorial Sloan Kettering Cancer Center.” Available: https://www.mskcc.org/cancer-care/patient-education/frequently-asked-questions-about-walking-after-your-surgery. [Accessed: Dec. 06, 2025]

[8] Ladd, Kevin. “What Is the Maximum Weight Capacity for a Medical Walker?” TheMedSupplyGuide.Com - Medical Supply Company Directory. Accessed: Sep 24, 2025. [Online]. Available: www.themedsupplyguide.com/weight/.

[9] International Organization for Standardization, ISO 11199-1:2021 — Assistive products for walking — Walking frames — Requirements and test methods. Geneva, Switzerland: ISO, 2021. Accessed: Sep 24, 2025. [Online]. Available: https://www.iso.org/standard/76651.html

[10] International Organization for Standardization, ISO 14971:2019 — Medical devices — Application of risk management to medical devices. Geneva, Switzerland: ISO, 2019. Accessed Sep 24, 2025. [Online]. Available: https://www.iso.org/standard/72704.html

[11]“Principles, Types and Configurations of Strain Gauges.” Accessed: Oct. 09, 2025. [Online]. Available: https://www.iqsdirectory.com/articles/load-cell/strain-gauge.html

[12]“Load Cell Amplifier HX711 Breakout Hookup Guide - SparkFun Learn.” Available: https://learn.sparkfun.com/tutorials/load-cell-amplifier-hx711-breakout-hookup-guide#installing-the-hx711-arduino-library-and-examples. [Accessed: Dec. 09, 2025]

[13]“50kg Loadcell Bracket versionF by patrick3345 - Thingiverse.” Available: https://www.thingiverse.com/thing:2624188#google_vignette. [Accessed: Dec. 09, 2025]

[14]“50kg Load Cells with HX711 and Arduino. 4x, 2x, 1x Diagrams. - Circuit Journal.” Available: https://circuitjournal.com/50kg-load-cells-with-HX711. [Accessed: Dec. 09, 2025]

[15]"Smart Walker Fall 2023 Final Report." Available: https://bmedesign.engr.wisc.edu/projects/f23/smart_walker/file/view/2f76631a-d7b5-456e-933d-31e1f2adb6a5/Final%20Report.pdf. [Accessed: Dec. 07, 2025]

[16]"LIDAR-Lite v3 Hookup Guide - SparkFun Learn." Available: https://learn.sparkfun.com/tutorials/lidar-lite-v3-hookup-guide. [Accessed: Dec. 07, 2025]

[17]"garmin/LIDARLite_Arduino_Library." Garmin International, Sept. 25, 2025. Available: https://github.com/garmin/LIDARLite_Arduino_Library. [Accessed: Dec. 09, 2025]

[18]S. Alder, "Why is HIPAA Important? Updated 2025," The HIPAA Journal, Apr. 15, 2025. Available: https://www.hipaajournal.com/why-is-hipaa-important/. [Accessed: Dec. 09, 2025]

Appendix

Appendix A: Product Design Specifications

Function/Problem Statement:

Patients with TBI often undergo traumatic events followed by intense rehabilitation to help them walk and return to everyday life as soon as possible. During rehabilitation, doctors struggle to measure progress as patients gain strength, making it difficult for physicians to provide patients with tangible data of their improvement. The insurance companies also often do not feel that there is enough evidence of improvement, making it harder for the clinic to be paid for the services they provide. The smart walker will measure the pressure applied, speed, and distance walked of patients with neuro-rehabilitation needs. This data will be reported and displayed in real-time to help clinicians monitor progress and motivate patients. Ultimately, the device will reduce the time required to meet Medicare's documentation needs and increase objective markers of patient readiness for discharge.

Client requirements:

The device must provide real-time data on user pressure, speed, and distance. It must be compatible with the walkers currently being used without compromising the structural integrity of the walker. Data provided by the smart walker should be presented in a metrics report that is comprehensible to insurance companies. Furthermore, the client requested a compact design that is easy to use, accurate, and reliable. The budget for this project is ~$500.

Design requirements

1. Physical and Operational Characteristics
    a. *Performance requirements*
        i. The smart walker attachments will modify an existing clinical walker, which can support a patient weighing up to 140 kg.
        ii. The added attachments should not interfere with the original walker's function.
        iii. The metrics provided by the walker attachments will include distance, speed, and pressure, and should be recorded and displayed to both the user and clinician.
    b. *Safety*
        i. The structural integrity of the existing walker must not be compromised
        ii. The device must follow all of the neuro-rehabilitation facility's safety standards and regulations
        iii. ISO requirements must be followed for all electronic and battery components[4].
        iv. Clear instructions should be given on how to use our walker attachments
    c. *Accuracy and Reliability*
        i. The walker attachment should accurately measure the values of distance, speed, and pressure within 10% of absolute values.
        ii. The measurements should also not vary more than 5% from their measured values.
        iii. These metrics must be accurate over distances of 10 meters and over time periods of 30 minutes.
    d. *Life in Service*
        i. The walker should withstand up to 10 patients for up to 5 trials a day over a period of 10 years before requiring maintenance.
        ii. The battery should last for 1 year before needed to be replace
    e. *Shelf Life*
        i. The walker attachment is expected to last 10 years before requiring repair or part replacement.
    f. *Operating Environment*
        i. The walker will be used at the client's neurorehabilitation center, which will be at a temperature of 16-26 °C. It is designed for indoor use and should not be taken outside to prevent damage from outdoor conditions.

ii. The walker will be used by multiple people, which will require sanitation between each use. The walker should be able to withstand continuous use of alcoholic disinfectants.

iii. The walker attachments should be able to withstand up to 140 kgs (~300lbs) of pressure for up to 20 minutes at a time. The attached pressure devices should be able to withstand this pressure and accurately read a pressure of this magnitude [5].

iv. The attachments to the walker should be able to be moved to different walkers.

g. *Ergonomics*

i. The height of the walker should be adjustable to heights of 80-100 centimeters[6].

ii. The width of the walker should be 60 centimeters. This will not be adjustable; however, the attachments can be switched between walkers if needed [6].

h. *Size*

i. The size of the smart walker attachments should not impact the usability of the existing walker.

ii. The components should not protrude from the existing walker by more than 10 cm to ensure that the walker can still easily fit through doorways and can be stored effectively.

i. *Weight*

i. The walker attachments should not add significant weight to the preexisting walker.

ii. Clinical walkers typically weigh between 4.5 and 9 kg; therefore, the combined weight of the smart walker attachments and the walker itself should not exceed this range [7].

j. *Materials*

i. Walkers are typically made of Aluminum with vinyl handles, serving as the base for smart walker attachments.

ii. The attachments will include various electrical components and 3D printed materials.

k. *Aesthetics, Appearance, and Finish*

i. The smart walker attachments should be as non-invasive as possible to ensure the look is as close as possible to that of a regular walker. This will ensure that it is easy to use for patients and clinicians.

ii. All wires should be contained to protect the device's lifespan and improve patient usability.

iii. All data should be displayed in a way that is accessible to both patients and clinicians, and provide real-time updates to motivate improvement.

2. Production Characteristics
    a. *Quantity*
        i. There will be attachments for one walker, which include two pressure sensors, a device to measure speed and distance, a microcontroller. All attachments should be removable and switchable between walkers.
    b. *Target Product Cost*
        i. The Budget for this project is $500
3. Miscellaneous
    a. *Standards and Specifications*
        i. FDA 21 CFR Part 820 (Quality System Regulation / QMSR):
            1. Specifies quality system requirements for medical devices, including design control, production processes, and corrective actions. FDA's new QMSR aligns this regulation with ISO 13485:2016, which will govern the design and manufacturing of Class II medical devices such as the Smart Walker.
        ii. IEC 60601-1-2:2014 – Electromagnetic Compatibility (EMC):
            1. Specifies requirements to ensure the Smart Walker's sensors and circuits are immune to electromagnetic disturbances and do not emit interference that could affect other devices in home or clinical environments.
        iii. ISO 11199-2:2005 – Assistive products for walking, Part 2: Rollators:
            1. Specifies performance and safety requirements for walkers with wheels, including stability, braking systems, strength, fatigue resistance, and labeling. Ensures the device meets international expectations for durability and safety.
        iv. ISO 11199-1:2021: Assistive products for walking — Walking frames — Requirements and test methods
            1. This standard outlines the performance, safety, and durability requirements for walking frames, including our Smart Walker. This includes the load, fatigue, and stability testing. This is intended to ensure that walkers and any add-ons do not compromise user safety or functionality.
        v. ISO 14971:2019: Medical devices — Application of risk management to medical devices
            1. This standard defines a structured process for identifying hazards, estimating and evaluating risks, implementing control measures, and monitoring effectiveness. This standard is essential for documenting and managing risks associated with various components of our Smart Walker, including structural failure, inaccurate weight data, and electrical hazards related to the add-on.

b. *Customer*
    i. The metrics should be displayed live to motivate users and aid in recovery as efficiently as possible.
    ii. The smart walker attachments are designed to attach to a 2-wheeled walker, as most patients are already familiar with the operation of this type of walker.
    iii. Since the users of this device reside in the U.S., all units need to be reported in empirical units to make it easier for patients to understand
    iv. All wiring and battery components need to be enclosed to protect user safety.

c. *Patient-related concerns*
    i. The requirements must be removable and sanitizable, as a variety of patients will use them.

d. *Competition*
    i. Few walkers on the market have similar features to what Mr. Kutschera is looking for in this walker. Some designs record speed, but there is nothing on the market that effectively records the pressure exerted by the patient on the walker.
    ii. All known walkers are also extremely expensive and unreasonable for what Mr. Kutschera is using them for.
    iii. One device, called the Camino, uses multiple sensors to detect the walker's gait and changes in gait. It can also detect changes in terrain to help prevent falls in patients. Although this device has many good features, it does way more than Mr. Kutschera needs and is way too expensive. It also does not track the pressure the patient applies to the walker. [1]

References

[1] “Meet Camino,” Camino Mobility. Accessed: Sep. 19, 2024. [Online]. Available: https://caminomobility.com/pages/meet-camino

[2] “Recognized Consensus Standards: Medical Devices.” Accessed: Sep. 19, 2024. [Online]. Available:
https://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfStandards/detail.cfm?standard__identificati
on_no=4
1349

[3] “Recognized Consensus Standards: Medical Devices.” Accessed: Sep. 19, 2024. [Online]. Available:
https://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfStandards/detail.cfm?standard__identification_no=4
4029

[4]“ISO 9001.” Amtivo US, 20 Aug. 2025, amtivo.com/us/iso-certification/iso-9001/?utm_term=iso+9001+standard&utm_campaign=&utm_source=bing&utm_medium=ppc&hsa_acc=6120343225&hsa_cam=614127105&hsa_grp=1142394374372069&hsa_ad=&hsa_src=o&hsa_tgt=kwd-71400365327958%3Aloc-190&hsa_kw=iso+9001+standard&hsa_mt=p&hsa_net=bing&hsa_ver=3&msclkid=e1a25e0c0d901e2b67af36c72823d16b.

[5]Ladd, Kevin. “What Is the Maximum Weight Capacity for a Medical Walker?” TheMedSupplyGuide.Com - Medical Supply Company Directory, 24 May 2020, www.themedsupplyguide.com/weight/.

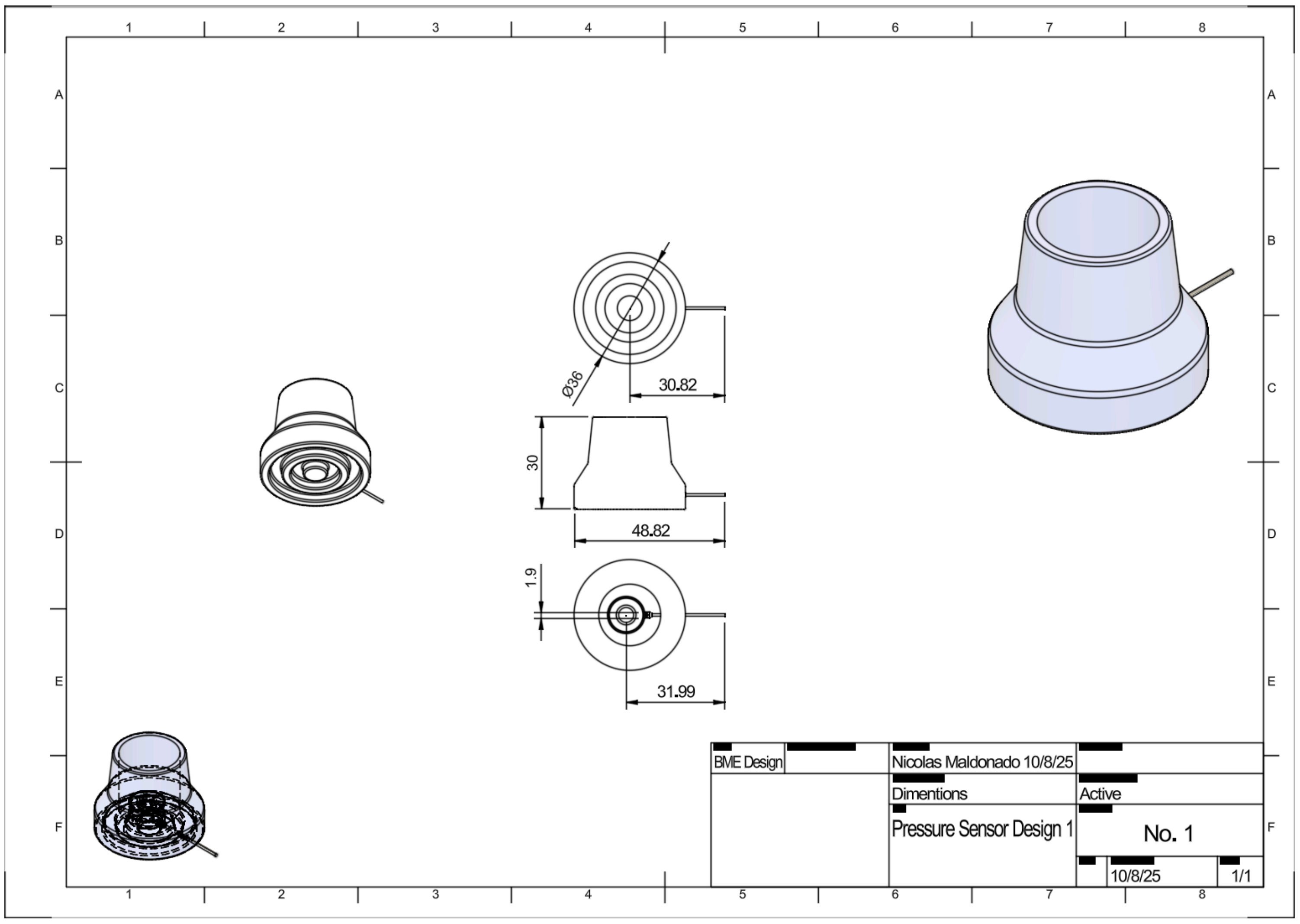
[6]Powell, Shanna. “How to Determine the Height of a Walker.” Senior Care Corner, 30 Nov. 2021, seniorcarecorner.com/how-to-determine-the-height-of-a-walker.

[7]Lars. “How Much Does a Walker Weigh? Is Lightweight Always the Best?” SeniorSupported, 16 Mar. 2023,
seniorsupported.com/how-much-does-a-walker-weigh/#:~:text=A%20standard%20walker%20%28no%20wheels%29%20weighs%20about%206,you.%20This%20is%20essential%20with%20a%20standard%20walker.

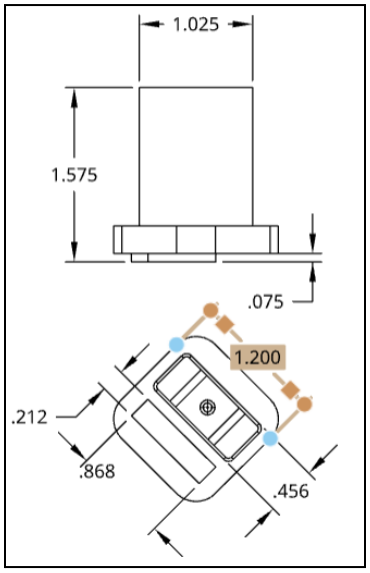
[8] International Organization for Standardization, ISO 11199-1:2021 — Assistive products for walking — Walking frames — Requirements and test methods. Geneva, Switzerland: ISO, 2021. [Online]. Available: https://www.iso.org/standard/76651.html

[9] International Organization for Standardization, ISO 14971:2019 — Medical devices — Application of risk management to medical devices. Geneva, Switzerland: ISO, 2019. [Online]. Available: https://www.iso.org/standard/72704.html

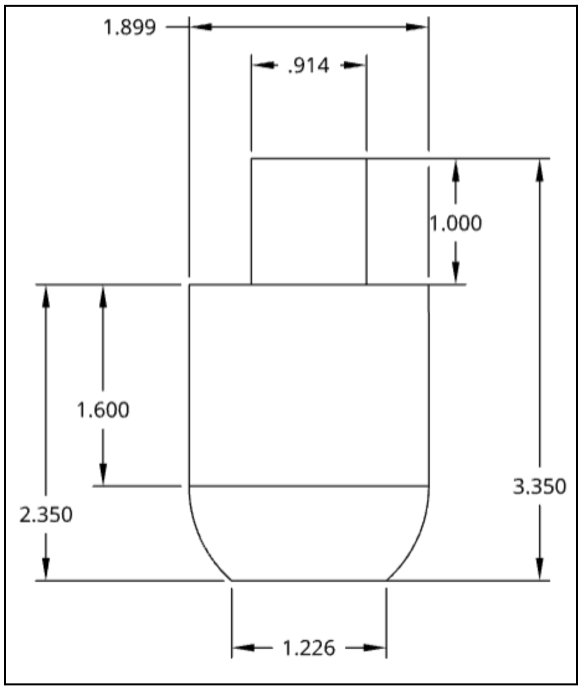
Appendix B: End Cap Design CAD Drawing



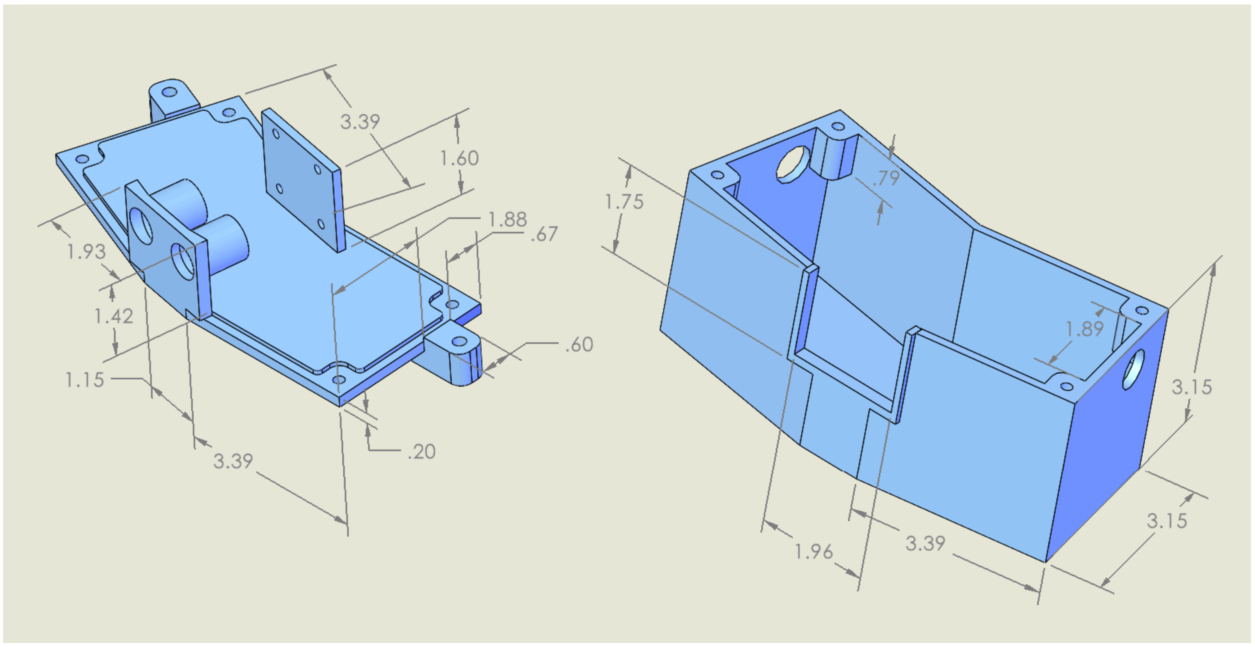
Appendix C: Cork Piece Design CAD Drawing



Appendix D: End Cap 3.0 Design CAD Drawing



Appendix E: Electrical Housing Design CAD Drawing



Appendix F: Final Code

```
#include <Wire.h>
#include <LIDARLite.h>
#include <WiFiS3.h>


// Load Cell
#define calibration_factor -18670.0
#define DOUT 3
#define CLK 2
HX711 scale;


// LIDAR
LIDARLite myLidarLite;


// WiFi Access Point
const char* ssid = "Smart_Walker";
const char* password = "12345678";
WiFiServer server(80);
```

```
// Update timing
unsigned long lastRead = 0;
const unsigned long sensorInterval = 50;   // 20 Hz
const unsigned long webpageInterval = 1000; // 1s refresh


// Speed State
float prev_ft = NAN;
unsigned long prev_ms = 0;
float speed_ema_ft_s = NAN;
bool ema_initialized = false;


// Latest values
float distance_ft = 0;
float speed_ft_s = 0;
float weight_lbs = 0;


// Trial Control
bool trialRunning = false;
float trialWeightSum = 0;
unsigned long trialWeightCount = 0;


// NEW: Trial metrics
float trialStartDist = NAN;
float trialEndDist = NAN;


// NEW: Avg speed during trial
float trialSpeedSum = 0;
unsigned long trialSpeedCount = 0;


// Convert LIDAR cm → feet
inline float cmToFeet(int cm) { return cm * 0.0328084f; }


// Calculate LIDAR distance + smoothed speed
void updateLidar(float &dist_out, float &speed_out) {
  unsigned long now = millis();
  int d = myLidarLite.distance(true);
  if (d <= 0) return;


  float dist_ft = cmToFeet(d);
  float instant = 0;


  if (prev_ms) {
    float dt = (now - prev_ms) / 1000.0;
    if (dt > 0) {
      // NOTE: speed = previous − current (walking toward sensor positive)
      instant = (prev_ft - dist_ft) / dt;


      const float alpha = 0.25;
      if (!ema_initialized) {
        speed_ema_ft_s = instant;
        ema_initialized = true;
      }
      else {
        speed_ema_ft_s = alpha * instant + (1 - alpha) * speed_ema_ft_s;
      }
    }
  }
  prev_ft = dist_ft;
  prev_ms = now;


  dist_out = dist_ft;
  speed_out = speed_ema_ft_s;
}


// Return HTML for 1s auto refresh
void sendWeb(WiFiClient &client) {
  client.println("HTTP/1.1 200 OK");
  client.println("Content-Type: text/html");
  client.println("Connection: close");
  client.println();
  client.println("<!DOCTYPE html><html><head>");
  client.println("<meta charset='UTF-8'>");
  client.println("<meta name='viewport' content='width=device-width'>");
  client.println("<meta http-equiv='refresh' content='1'>");
  client.println("<style>");
  client.println("body{font-family:Arial;text-align:center;}");
  client.println("h1{margin-top:18px;}");
  client.println("div{font-size:1.5em;margin:10px;}");
  client.println(".btn{padding:10px
20px;margin:5px;font-size:1em;border:none;border-radius:6px;}");
  client.println(".start{background:#4CAF50;color:white;}");
  client.println(".stop{background:#d32f2f;color:white;}");
  client.println("</style>");
  client.println("</head><body>");
  client.println("<h1>Smart Walker Live Telemetry</h1>");
  client.print("<div>Status: ");
  client.print(trialRunning ? "RUNNING" : "STOPPED");
  client.println("</div>");
  client.print("<div>Weight (lbs): "); client.print(weight_lbs, 2); client.println("</div>");
  client.print("<div>Distance (ft): "); client.print(distance_ft, 2); client.println("</div>");
  client.print("<div>Speed (ft/s): "); client.print(speed_ft_s, 2); client.println("</div>");
  client.println("<a href='/start'><button class='btn start'>Start Trial</button></a>");
  client.println("<a href='/stop'><button class='btn stop'>Stop Trial</button></a>");
  client.println("<p>Page auto-refreshes every 1 second.</p>");
  client.println("</body></html>");
}


// SETUP
void setup() {
  Serial.begin(115200);


  // Sensors
  Wire.begin();
  delay(200);
  scale.begin(DOUT, CLK);
  scale.set_scale(calibration_factor);
  scale.tare();


  myLidarLite.begin(0, true);
  myLidarLite.configure(0);


  // WiFi
  WiFi.beginAP(ssid, password);
  delay(500);
  Serial.print("WiFi AP IP: ");
  Serial.println(WiFi.localIP());
  server.begin();
}


// LOOP
void loop() {
  unsigned long now = millis();


  // SENSOR LOOP (20 Hz)
  if (trialRunning && now - lastRead >= sensorInterval) {
     lastRead = now;


     // Weight
     weight_lbs = scale.get_units(10);


     // Distance + speed
     updateLidar(distance_ft, speed_ft_s);


     // Track average weight
     trialWeightSum += weight_lbs;
     trialWeightCount++;


     // track speed average
     if (!isnan(speed_ft_s)) {
        trialSpeedSum += speed_ft_s;
        trialSpeedCount++;
     }


     // track end distance always
     trialEndDist = distance_ft;


     Serial.print("weight:"); Serial.print(weight_lbs,2);
     Serial.print("\tdist:"); Serial.print(distance_ft,2);
     Serial.print("\tspeed:"); Serial.println(speed_ft_s,2);
  }
```

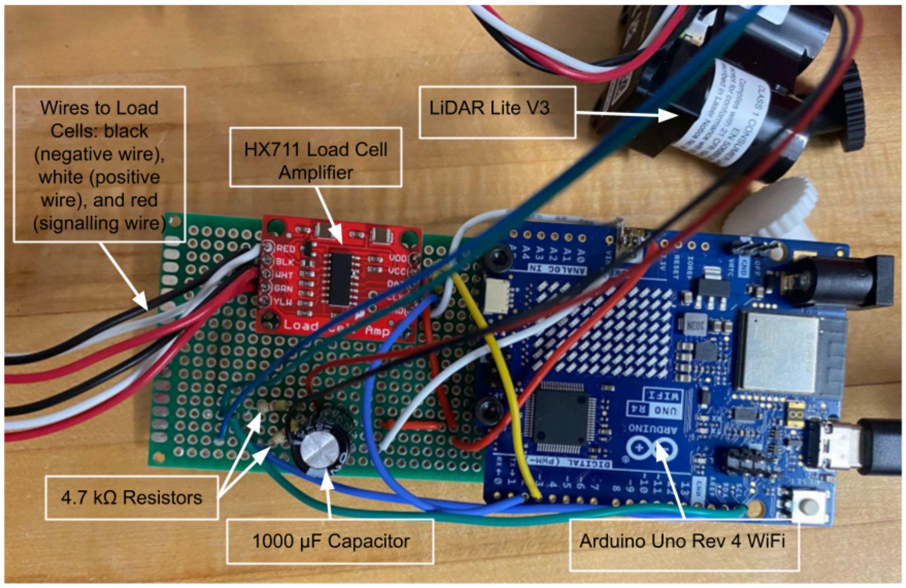
```
// WEB REQUEST HANDLING
WiFiClient client = server.available();
if (client) {
  while (client.connected() && !client.available()) delay(1);
  String req = client.readStringUntil('\r');
  client.flush();


  // Handle start/stop trial
  if (req.indexOf("GET /start") >= 0) {
    trialRunning = true;


    // reset metrics
    trialWeightSum = 0;
    trialWeightCount = 0;


    trialSpeedSum = 0;
    trialSpeedCount = 0;


    trialStartDist = distance_ft;
    trialEndDist   = distance_ft;


    Serial.println("=== Trial START ===");
  }


  if (req.indexOf("GET /stop") >= 0) {
    trialRunning = false;
    Serial.println("=== Trial STOP ===");


    // AVERAGE WEIGHT
    if (trialWeightCount > 0) {
      float avgWeight = trialWeightSum / trialWeightCount;
      Serial.print("Trial Average Weight (lbs): ");
      Serial.println(avgWeight, 2);
    }
```

```
      // AVERAGE SPEED
      if (trialSpeedCount > 0) {
        float avgSpeed = trialSpeedSum / trialSpeedCount;
        Serial.print("Trial Average Speed (ft/s): ");
        Serial.println(avgSpeed, 2);
      }


      // ABSOLUTE ΔDISTANCE
      if (!isnan(trialStartDist) && !isnan(trialEndDist)) {
        float deltaAbs = fabs(trialEndDist - trialStartDist);
        Serial.print("Absolute ΔDistance (ft): ");
        Serial.println(deltaAbs, 2);
      }
    }


    sendWeb(client);
    client.stop();
  }
}
```

Appendix G: Fabrication and Assembly Protocol

Soldering of Electrical Components

1. Take a perfboard and line up in parallel with an Arduino uno using the predrilled holes in the Arduino as a guide for the connection point.
2. Drill two 0.125 inch in diameter holes into the perfboard and connect with appropriate bolts and nuts.
3. Based on a breadboard or diagram of a breadboard place larger components first such as the HX711 load cell amplifier and the capacitor for the LiDAR leaving enough room open holes to make connections to wires.
4. Once layout is confirmed, solder pins to the perfboard.
5. From there solder connection wires directly to the Arduino
6. For reference to specific connections see figure below.


Wires to Load Cells: black (negative wire), white (positive wire), and red (signalling wire)
HX711 Load Cell Amplifier
LiDAR Lite V3
4.7 kΩ Resistors
1000 µF Capacitor
Arduino Uno Rev 4 WiFi

Endcaps and Loadcells

1. The wiring to the load cells is first wrapped in electrical tape
2. The load cells must then be clicked into place at the bottom of the endcaps
3. The direction can be ascertained by the wires as there are small grooves to indicate the direction of the wires out of the end caps.
4. Next, with a good amount of force, the cork pieces which push down on the load cells should be inserted into the end caps.
5. Once they are in the end caps they can be twisted until they click into place if they are not already properly aligned with the anatomy of the load cell.
6. After removing the current end caps on the walker, now both the 3D printed end cap-load cell systems are ready to be slid onto the ends of the back two legs.
7. Then the hook side of adhesive velcro was placed at several intervals on the walker frame to prevent the wires from drooping in the way of the walker user's path.
8. The loop side is attached to the wires in corresponding locations along their path to the electrical housing.

Attachment and Positioning of Electrical Housing

1. The electrical housing is placed centered between the two cross bars at the front of the walker
2. For the bottom cross bar the bottom half of the electrical housing was secured using two 1” long, ¼” diameter bolts and nuts.
3. The electronic components including the attached perfboard-arduino system and any extra wires are placed with the the housing and shut in by the top of the housing
4. A clamp of 1” diameter and 2”length was placed around the top crossbar with a ¼” diameter bolt through a hole in the top of the housing unit with a nut on top to secure them both together and to the walker frame.

Connecting to Webpage

1. Turn on battery to Arduino
2. Connect the wireless device to the “Smart_Walker” wifi network with the password “12345678”
3. Go to the webpage http://192.168.4.1/
4. Press “Start Trial” to begin the tral and “Stop Trial” to end it. The data that results is the average weight and speed and the total distance traveled.

Appendix H: Expense Table

Item	Description	Manufacturer	Part Number	QTY	Cost Each	Total	Link
Walker	2-wheel walker, **gifted** by client	Performance Health Supply, Inc.	081561703	1	$136.73	$0	Link
Load Cell initial 3D print	3D print of End-Cap 2.0 design **gifted** by friend with printer.	bambu lab a1 mini	N/A	1	$1.60	$0	N/A
Load Cells + HX711	4 50 kg load cells with HX711	Nextion	702795764555	1	$16.85	$16.85	Link
LiDar	Sensor Optical 3-200CM 12C	DigiKey	DigiKey part number : 1568-14032-ND	1	145.93	145.93	Link
Load Cell initial 3D print	3D print of End-Cap 2.0 design for show and tell	Bambu lab	NA	1	$1.04	$1.04	Link
Arduino Uno Rev 4	Arduino with wifi abilities for our code and electronics.	Arduino	Sku: ABX00087	1	29.21	29.21	Link
Additional HX711 Purchase	Extra HX711 Load Cell Amplifier because we were struggling to get the amazon one to function correctly	SparkFun	SKU: SEN-13879		11.50	11.50	Link

Electrical Component Prints	PLA print of electrical component box	Bambu lab	NA	1	4.56	4.56	Link
Battery Housing	9v Battery Holder with ON/Off Switch for Arduino	Gikfun	EK2107	1	19.28	19.28	Link
Velcro Straps	1.5ft of velcro to securely attach the wires along the walker leg	Wendt Commons	NA	1	5.19	5.19	NA
BME Design Account	This is a combination of shrink wrap for our prints, batteries, and a series of trial and final prints for the load cell and electrical box housing.	Wendt Commons	BME Design	NA	43.58	43.58	NA
TOTAL:						**$277.06**	

Appendix I: Load Cell Calibration Code

```
/*

   -------------------------------------------------------------------------------------

   HX711_ADC

   Arduino library for HX711 24-Bit Analog-to-Digital Converter for Weight Scales

   Olav Kallhovd sept2017

   -------------------------------------------------------------------------------------

*/
```

```
/*

  This example file shows how to calibrate the load cell and optionally store the calibration

  value in EEPROM, and also how to change the value manually.

  The result value can then later be included in your project sketch or fetched from EEPROM.



  To implement calibration in your project sketch the simplified procedure is as follow:

     LoadCell.tare();

     //place known mass

     LoadCell.refreshDataSet();

     float newCalibrationValue = LoadCell.getNewCalibration(known_mass);

*/



#include <HX711_ADC.h>

#include <EEPROM.h>



//pins:

const int HX711_dout = 3; //mcu > HX711 dout pin

const int HX711_sck = 2; //mcu > HX711 sck pin



//HX711 constructor:

HX711_ADC LoadCell(HX711_dout, HX711_sck);
```

```
}



const int calVal_eepromAdress = 0;

unsigned long t = 0;



void setup() {

  Serial.begin(57600); delay(10);

  Serial.println();

  Serial.println("Starting...");



  LoadCell.begin();

  //LoadCell.setReverseOutput(); //uncomment to turn a negative output value to positive

  unsigned long stabilizingtime = 2000; // preciscion right after power-up can be improved by adding a
few seconds of stabilizing time

  boolean _tare = true; //set this to false if you don't want tare to be performed in the next step

  LoadCell.start(stabilizingtime, _tare);

  if (LoadCell.getTareTimeoutFlag() || LoadCell.getSignalTimeoutFlag()) {

    Serial.println("Timeout, check MCU>HX711 wiring and pin designations");

    while (1);

  }

  else {

    LoadCell.setCalFactor(1.0); // user set calibration value (float), initial value 1.0 may be used for this
sketch

    Serial.println("Startup is complete");

  }
  while (!LoadCell.update());
  calibrate(); //start calibration procedure
}


void loop() {
  static boolean newDataReady = 0;
  const int serialPrintInterval = 0; //increase value to slow down serial print activity


  // check for new data/start next conversion:
  if (LoadCell.update()) newDataReady = true;


  // get smoothed value from the dataset:
  if (newDataReady) {
    if (millis() > t + serialPrintInterval) {
      float i = LoadCell.getData();
      Serial.print("Load_cell output val: ");
      Serial.println(i);
      newDataReady = 0;
      t = millis();
    }
  }
  // receive command from serial terminal

  if (Serial.available() > 0) {

    char inByte = Serial.read();

    if (inByte == 't') LoadCell.tareNoDelay(); //tare

    else if (inByte == 'r') calibrate(); //calibrate

    else if (inByte == 'c') changeSavedCalFactor(); //edit calibration value manually

  }


  // check if last tare operation is complete

  if (LoadCell.getTareStatus() == true) {

    Serial.println("Tare complete");

  }


}


void calibrate() {

  Serial.println("***");

  Serial.println("Start calibration:");

  Serial.println("Place the load cell an a level stable surface.");

  Serial.println("Remove any load applied to the load cell.");

  Serial.println("Send 't' from serial monitor to set the tare offset.");
```

```
boolean _resume = false;

while (_resume == false) {

  LoadCell.update();

  if (Serial.available() > 0) {

    if (Serial.available() > 0) {

      char inByte = Serial.read();

      if (inByte == 't') LoadCell.tareNoDelay();

    }

  }

  if (LoadCell.getTareStatus() == true) {

    Serial.println("Tare complete");

    _resume = true;

  }

}


Serial.println("Now, place your known mass on the loadcell.");

Serial.println("Then send the weight of this mass (i.e. 100.0) from serial monitor.");


float known_mass = 0;

_resume = false;

while (_resume == false) {

  LoadCell.update();
  if (Serial.available() > 0) {

    known_mass = Serial.parseFloat();

    if (known_mass != 0) {

      Serial.print("Known mass is: ");

      Serial.println(known_mass);

      _resume = true;

    }

  }

}


LoadCell.refreshDataSet(); //refresh the dataset to be sure that the known mass is measured correct

float newCalibrationValue = LoadCell.getNewCalibration(known_mass); //get the new calibration value


Serial.print("New calibration value has been set to: ");

Serial.print(newCalibrationValue);

Serial.println(", use this as calibration value (calFactor) in your project sketch.");

Serial.print("Save this value to EEPROM adress ");

Serial.print(calVal_eepromAdress);

Serial.println("? y/n");


_resume = false;

while (_resume == false) {
   if (Serial.available() > 0) {
    char inByte = Serial.read();
    if (inByte == 'y') {
#if defined(ESP8266)|| defined(ESP32)
      EEPROM.begin(512);
#endif
      EEPROM.put(calVal_eepromAdress, newCalibrationValue);
#if defined(ESP8266)|| defined(ESP32)
      EEPROM.commit();
#endif
      EEPROM.get(calVal_eepromAdress, newCalibrationValue);
      Serial.print("Value ");
      Serial.print(newCalibrationValue);
      Serial.print(" saved to EEPROM address: ");
      Serial.println(calVal_eepromAdress);
      _resume = true;


    }
    else if (inByte == 'n') {
      Serial.println("Value not saved to EEPROM");
      _resume = true;
    }
    }
   }

  Serial.println("End calibration");
  Serial.println("***");
  Serial.println("To re-calibrate, send 'r' from serial monitor.");
  Serial.println("For manual edit of the calibration value, send 'c' from serial monitor.");
  Serial.println("***");
}

void changeSavedCalFactor() {
  float oldCalibrationValue = LoadCell.getCalFactor();
  boolean _resume = false;
  Serial.println("***");
  Serial.print("Current value is: ");
  Serial.println(oldCalibrationValue);
  Serial.println("Now, send the new value from serial monitor, i.e. 696.0");
  float newCalibrationValue;
  while (_resume == false) {
    if (Serial.available() > 0) {
      newCalibrationValue = Serial.parseFloat();
      if (newCalibrationValue != 0) {
        Serial.print("New calibration value is: ");
        Serial.println(newCalibrationValue);
        LoadCell.setCalFactor(newCalibrationValue);
        _resume = true;
      }
    }
  }
  _resume = false;
  Serial.print("Save this value to EEPROM adress ");
  Serial.print(calVal_eepromAdress);
  Serial.println("? y/n");
  while (_resume == false) {
    if (Serial.available() > 0) {
      char inByte = Serial.read();
      if (inByte == 'y') {
#if defined(ESP8266)|| defined(ESP32)
        EEPROM.begin(512);
#endif
        EEPROM.put(calVal_eepromAdress, newCalibrationValue);
#if defined(ESP8266)|| defined(ESP32)
        EEPROM.commit();
#endif
```

```
        EEPROM.get(calVal_eepromAdress, newCalibrationValue);

        Serial.print("Value ");

        Serial.print(newCalibrationValue);

        Serial.print(" saved to EEPROM address: ");

        Serial.println(calVal_eepromAdress);

        _resume = true;

      }

      else if (inByte == 'n') {

        Serial.println("Value not saved to EEPROM");

        _resume = true;

      }

    }

  }

  Serial.println("End change calibration value");

  Serial.println("***");

}
```

Appendix J: Load Cell Code

```
/*

 The HX711 does one thing well: read load cells. The breakout board is compatible with any wheat-stone
bridge

 based load cell which should allow a user to measure everything from a few grams to tens of tons.

 Arduino pin 2 -> HX711 CLK

 3 -> DAT
 5V -> VCC

 GND -> GND


 The HX711 board can be powered from 2.7V to 5V so the Arduino 5V power should be fine.


*/


#include "HX711.h"


#define calibration_factor -7050.0 //This value is obtained using the SparkFun_HX711_Calibration sketch


#define DOUT  3

#define CLK  2


HX711 scale;


void setup() {

  Serial.begin(9600);

  Serial.println("HX711 scale demo");


  scale.begin(DOUT, CLK);

  scale.set_scale(calibration_factor); //This value is obtained by using the SparkFun_HX711_Calibration
sketch

  scale.tare(); //Assuming there is no weight on the scale at start up, reset the scale to 0
  Serial.println("Readings:");
}

void loop() {
  Serial.print("Reading: ");
  Serial.print(scale.get_units(), 1); //scale.get_units() returns a float
  Serial.print(" lbs"); //You can change this to kg but you'll need to refactor the calibration_factor
  Serial.println();
}
```

Appendix K: LiDAR Code

```
/*------------------------------------------------------------------------------

  LIDARLite Arduino Library
  v3HP/v3HP_I2C

  This example shows methods for running the LIDAR-Lite v3 HP in various
  modes of operation. To exercise the examples open a serial terminal
  program (or the Serial Monitor in the Arduino IDE) and send ASCII
  characters to trigger the commands. See "loop" function for details.

  Connections:
  LIDAR-Lite 5 Vdc (red) to Arduino 5v
  LIDAR-Lite I2C SCL (green) to Arduino SCL
  LIDAR-Lite I2C SDA (blue) to Arduino SDA

  LIDAR-Lite Ground (black) to Arduino GND


  (Capacitor recommended to mitigate inrush current when device is enabled)

  680uF capacitor (+) to Arduino 5v

  680uF capacitor (-) to Arduino GND


  See the Operation Manual for wiring diagrams and more information:

http://static.garmin.com/pumac/LIDAR_Lite_v3HP_Operation_Manual_and_Technical_Specifications.pd
f


------------------------------------------------------------------------------*/


#include <stdint.h>

#include <Wire.h>

#include <LIDARLite_v3HP.h>


LIDARLite_v3HP myLidarLite;


#define FAST_I2C


enum rangeType_T

{

   RANGE_NONE,
   RANGE_SINGLE,
   RANGE_CONTINUOUS,
   RANGE_TIMER
};


void setup()
{
   // Initialize Arduino serial port (for display of ASCII output to PC)
   Serial.begin(115200);


   // Initialize Arduino I2C (for communication to LidarLite)
   Wire.begin();
   #ifdef FAST_I2C
       #if ARDUINO >= 157
           Wire.setClock(400000UL); // Set I2C frequency to 400kHz (for Arduino Due)
       #else
           TWBR = ((F_CPU / 400000UL) - 16) / 2; // Set I2C frequency to 400kHz
       #endif
   #endif


   // Configure the LidarLite internal parameters so as to lend itself to
   // various modes of operation by altering 'configure' input integer to
   // anything in the range of 0 to 5. See LIDARLite_v3HP.cpp for details.
   myLidarLite.configure(0);
```

```
}



void loop()
{
  uint16_t distance;
  uint8_t  newDistance = 0;
  uint8_t  c;
  rangeType_T rangeMode = RANGE_NONE;

  PrintMenu();

  // Continuous loop
  while (1)
  {
    // Each time through the loop, look for a serial input character
    if (Serial.available() > 0)
    {
      //  read input character ...
      c = (uint8_t) Serial.read();

      // ... and parse
      switch (c)
      {
```

```
case 'S':
case 's':
   rangeMode = RANGE_SINGLE;
   break;

case 'C':
case 'c':
   rangeMode = RANGE_CONTINUOUS;
   break;

case 'T':
case 't':
   rangeMode = RANGE_TIMER;
   break;

case '.':
   rangeMode = RANGE_NONE;
   break;

case 'D':
case 'd':
   rangeMode = RANGE_NONE;
   dumpCorrelationRecord();
   break;
```

```
        case 'P':
        case 'p':
            rangeMode = RANGE_NONE;
            peakStackExample();
            break;

        case 0x0D:
        case 0x0A:
            break;

        default:
            rangeMode = RANGE_NONE;
            PrintMenu();
            break;
    }
}

switch (rangeMode)
{
    case RANGE_NONE:
        newDistance = 0;
        break;
    case RANGE_SINGLE:
        newDistance = distanceSingle(&distance);
        break;

    case RANGE_CONTINUOUS:
        newDistance = distanceContinuous(&distance);
        break;

    case RANGE_TIMER:
        delay(250); // 4 Hz
        newDistance = distanceFast(&distance);
        break;

    default:
        newDistance = 0;
        break;
}

// When there is new distance data, print it to the serial port
if (newDistance)
{
    Serial.println(distance);
}
```

```
        // Single measurements print once and then stop
        if (rangeMode == RANGE_SINGLE)
        {
            rangeMode = RANGE_NONE;
        }
    }
}

void PrintMenu(void)
{
    Serial.println("=====================================");
    Serial.println("== Type a single character command ==");
    Serial.println("=====================================");
    Serial.println(" S - Single Measurement");
    Serial.println(" C - Continuous Measurement");
    Serial.println(" T - Timed Measurement");
    Serial.println(" . - Stop Measurement");
    Serial.println(" D - Dump Correlation Record");
    Serial.println(" P - Peak Stack Example");
}

//------------------------------------------------------------------
// Read Single Distance Measurement
//
```

```
// This is the simplest form of taking a measurement. This is a
// blocking function as it will not return until a range has been
// taken and a new distance measurement can be read.
//---------------------------------------------------------------------
uint8_t distanceSingle(uint16_t * distance)
{
    // 1. Wait for busyFlag to indicate device is idle. This must be
    //    done before triggering a range measurement.
    myLidarLite.waitForBusy();

    // 2. Trigger range measurement.
    myLidarLite.takeRange();

    // 3. Wait for busyFlag to indicate device is idle. This should be
    //    done before reading the distance data that was triggered above.
    myLidarLite.waitForBusy();

    // 4. Read new distance data from device registers
    *distance = myLidarLite.readDistance();

    return 1;
}

//---------------------------------------------------------------------
```

```
// Read Continuous Distance Measurements
//
// The most recent distance measurement can always be read from
// device registers. Polling for the BUSY flag in the STATUS
// register can alert the user that the distance measurement is new
// and that the next measurement can be initiated. If the device is
// BUSY this function does nothing and returns 0. If the device is
// NOT BUSY this function triggers the next measurement, reads the
// distance data from the previous measurement, and returns 1.
//---------------------------------------------------------------------
uint8_t distanceContinuous(uint16_t * distance)
{
   uint8_t newDistance = 0;

   // Check on busyFlag to indicate if device is idle
   // (meaning = it finished the previously triggered measurement)
   if (myLidarLite.getBusyFlag() == 0)
   {
      // Trigger the next range measurement
      myLidarLite.takeRange();

      // Read new distance data from device registers
      *distance = myLidarLite.readDistance();
```

```
        // Report to calling function that we have new data
        newDistance = 1;
    }

    return newDistance;
}

//---------------------------------------------------------------------
// Read Distance Measurement, Quickly
//
// Read distance. The approach is to poll the status register until the device goes
// idle after finishing a measurement, send a new measurement command, then read the
// previous distance data while it is performing the new command.
//---------------------------------------------------------------------
uint8_t distanceFast(uint16_t * distance)
{
    // 1. Wait for busyFlag to indicate device is idle. This must be
    //    done before triggering a range measurement.
    myLidarLite.waitForBusy();

    // 2. Trigger range measurement.
    myLidarLite.takeRange();

    // 3. Read previous distance data from device registers.
    //    After starting a measurement we can immediately read previous
    //    distance measurement while the current range acquisition is
    //    ongoing. This distance data is valid until the next
    //    measurement finishes. The I2C transaction finishes before new
    //    distance measurement data is acquired.
    *distance = myLidarLite.readDistance();

    return 1;
}

//---------------------------------------------------------------------
// Print the correlation record for analysis
//---------------------------------------------------------------------
void dumpCorrelationRecord()
{
    myLidarLite.correlationRecordToSerial(256);
}

//---------------------------------------------------------------------
// Print peaks and calculated distances from the peak stack
//---------------------------------------------------------------------
void peakStackExample()
{
    int16_t   peakArray[8];
    int16_t   distArray[8];

    uint8_t   i;


    // - Read the Peak Stack.

    // - Peaks and calculated distances are returned in local arrays.

    // - See library function for details on the makeup of the stack

    //   and how distance data is created from the stack.

    myLidarLite.peakStackRead(peakArray, distArray);


    // Print peaks and calculated distances to the serial port.

    Serial.println();

    Serial.println("IDX PEAK DIST");

    for (i=0 ; i<8 ; i++)

    {

        Serial.print(i);

        Serial.print("   ");

        Serial.print(peakArray[i]);

        Serial.print("  ");

        Serial.print(distArray[i]);

        Serial.println();

    }

}
```