Load Cell Adaptive Computer Mouse Tutorial

Mae Hubel
Washington University in St. Louis

Follow this and additional works at: https://openscholarship.wustl.edu/mems500

Recommended Citation
https://openscholarship.wustl.edu/mems500/192

This Final Report is brought to you for free and open access by the Mechanical Engineering & Materials Science at Washington University Open Scholarship. It has been accepted for inclusion in Mechanical Engineering and Materials Science Independent Study by an authorized administrator of Washington University Open Scholarship. For more information, please contact digital@wumail.wustl.edu.
INTRODUCTION

So much of life happens through a computer these days. We take for granted our mobility within the digital interface. But what if you have an impairment inhibiting your hand dexterity? Or you are hoping to incorporate full body movement into your stagnant desk job? You may want a mouse that you can stand on and operate through shifting your weight—and not have to pay an arm and a leg for it.

Sure, maybe you aren’t interested in integrating this option into your daily desktop computer habits, but perhaps into a video game? Replace your Xbox joystick with a standing balance board using this simple tutorial—and pick up some handy maker skills along the way.

Read time: 30 min
Build time: 2 hr

Required Materials:
1. Electronics
   a. HX711 Load Cell Amplifier
   b. Ethernet port
   c. RJ45 Cat-6 Ethernet cable
   d. Microcontroller
   e. Aux port
   f. Potentiometer
   g. Female header pins
   h. PCB
   i. Force sensors
1. Platform and casing
   a. Board
   b. 8 Screws (½”-long #8 or #6 wood screws)
   c. Screwdriver and drill bits (½” twist drill for pilot, phillips head PH2 for screwing)
   d. 3D printed load cell casings
2. Tools
   a. Wire cutters/strippers
   b. Soldering iron and fan, solder
   c. Screwdriver
   d. Electrical tape
   e. Suggested reading
      i. How to Solder
      ii. More information about load cells

ACCESSIBLE CONTROLLERS

Over the last decade or so, a couple major tech companies have started incorporating alternative controllers into their repertoire. One of the leading products is the Xbox Adaptive Controller. It is a main
controller hub that includes options to add on additional switches, buttons, mounts, or joysticks. Its modular design allows for maximum flexibility but it is primarily to be used in conjunction with an Xbox.

![Image of XBox Adaptive Controller](xbox.com)

**Figure 1:** XBox Adaptive Controller (Source: [xbox.com](http://xbox.com))

Some other products used as alternative computer mice or video controllers are shown below.

![Alternative controllers: Smashmouse, Boomer Mouse, and 3DRudder](coolthings.com, boomerthefootmouse.com, 3drudder.com)

**Figure 2:** Alternative controllers: Smashmouse, Boomer Mouse, and 3DRudder (Sources: [coolthings.com](http://coolthings.com), [boomerthefootmouse.com](http://boomerthefootmouse.com), [3drudder.com](http://3drudder.com)).

**FORCE SENSORS**

There are a few different ways to quantify an applied force, but one of the simplest methods is with a strain gauge. A strain gauge usually consists of a fork shaped piece of metal (as shown below) that receives forces from either side in designated locations. In receiving this force, the strain fork deforms slightly and the material’s resistance changes. When connected to a microcontroller, a computer can detect the change in electrical resistance and based on known material properties of the fork, compute the experienced strain. From here, the force reading is calibrated using a known weight.
A Wheatstone bridge is an electrical circuit that was developed to get the most accurate measurement possible of an unknown resistance. It incorporates three resistors of known resistance and one unknown resistance in a diamond shape. Learn more about how Wheatstone bridges work [here](800loadcel.com). In our case, each load cell is a “half-bridge force sensor” and essentially includes half of a wheatstone bridge. What this consists of is two strain gauges that change resistance when a load is applied—one gets stretched and increases in resistance while the other gets compressed and decreases in resistance. Measuring the voltage difference between the red wires of two load cells can combine two “half-bridges” into a “full” wheatstone bridge. A common bathroom scale will combine these in opposite directions (as shown in the left figure below) and therefore measure the total weight applied. In our case, we will combine them in the same direction (as shown in the right figure below) to measure the relative difference in weight applied to each load cell. This wheatstone bridge setup helps to detect the very small changes to resistance.
that occur when a load is applied.

**Figure 4:** Symbolic comparison of load cell connections for normal and modified systems.

To easily convert the electrical signal from this Wheatstone bridge into usable information, Sparkfun has developed a Load Cell Amplifier (HX711), an integrated circuit (IC) chip that has a 24-bit analog-to-digital (ADC) converter. As opposed to an Arduino’s built-in 10-bit ADC, the HX711 can check for over 1,000 times as many values over the same voltage range, leading to much higher precision.

Inside a standard bathroom scale there are four load cells on each corner. The four applied forces read from these load cells are added together to find a total applied force, or in the case of a scale, weight. In our case, information from the strain gauges will be interpreted slightly differently. We care not only about magnitudes of input but also how each cell’s magnitude relates to the other’s. As shown in the figure above, we will be measuring the relative load between the cells. The overall magnitude will still be important in order to define how much the relative difference between magnitudes gets magnified. The overall magnitude will also help define a threshold of force under which we consider input signals to be “noise.”

You may want to double check that your load cells are color coded in the same way as we have shown here. To check this, check the resistance of the connection between different pairs of wires. Follow the process shown at the end of the Hardware Hookup section of [this article](#) for more details.

**ARDUINO BASICS**

If you aren’t already familiar with Arduino, it’s a great tool to have under your belt and will open up a world of opportunities. It’s an open-source electronics platform that can be used for projects of a wide range of complexity. You can program the Arduino microcontroller using the Arduino programming language on your computer. Programming is essentially writing a list of instructions for how it should
convert a given input (finger on a button, light on a sensor, force on a load cell, etc.) into an output (turn on a light, activate a motor, move a computer mouse right 100 pixels).

There’s a lot you can do with an Arduino. Here we are focusing on defining the input as signals from the load cells made readable through the load cell amplifiers (HX711). We will also have the option to add in other inputs using an aux plug and the option to modify the load cell input using potentiometers.

One option for the output is using it to control the mouse of a computer. The Keyboard and Mouse Control package can be used to convert Arduino inputs into basic computer mouse controls. Some of the most common mouse outputs we are aiming to emulate are left, right, middle click, and a scroll wheel. See the Arduino documentation for details on these and other common functions here.

The Arduino can also function as a joystick. Using the external library from mhieron shown here, inputs will be converted into directional controls similar to that of a joystick. As a joystick, the Arduino can then be plugged into the Xbox Adaptive Controller as one of the analog sticks. This expands options to maneuver directional controls on the Xbox using different inputs than the standard hand manipulation—for instance, the balance of your bodyweight through a standing board. There is also an Xbox library for the Arduino that allows it to itself emulate an Xbox controller. This is a really great tool for further exploration but it doesn’t work great on Macs.

BUILD THE BASE AND CASE

For the physical board itself, you’ll want to find a relatively square piece of sturdy wood that can be drilled into. This could be a chopping board, a game board (as shown in this tutorial), or simply a piece of lumber that you can cut to shape. Rummage around your local thrift store and get creative!
As explained earlier, the load cells you’re using measure the difference between force exerted on one side (through the outer edge of the fork) to the force exerted on the other side (through the center of the load cell). A special casing is needed to hold the load cells in place while still directing all forces through the desired locations. We’ve designed a housing for the load cells that is included with this report, in both SolidWorks (.SLDPRT) and stereolithography (.STL) file formats. The STL file can be printed with no support material if the rectangular face (the back of the C-shape) is placed on the print bed with the round foot and mounting slots facing upward. See the image below for print orientation.

LoadCellHousing.SLDPRRT
LoadCellHousing.STL
Once you have your load cell housings printed and your board prepared, you’re ready to start putting it all together. Place the housings on the corners of your board as shown below, lining up the “arms” of the housings so that they are in line with the edge of the board. These “arms” should be flush with the board while the “head” is hovering a distance away. Note that you are attaching these load cell housings to the bottom of your board, so the designated locations will be flipped.
Use a pencil to mark the holes. Using your \( \frac{1}{8} \)" twist drill drill bit, drill pilot holes at all of these pencil markings. Using the PH2 driver, screw one screw into each housing. This will hold the housing in place while remaining just flexible enough to slip each load cell into place.

**Figure 8:** Layout of load cell housings on base of board.
Figure 9: Marking location, drilling pilot hole, and screwing in first screw of housings.

The load cells should slide easily into place. Once all load cells are in, screw the second screw into each housing to secure them into place.

Figure 10: Inserting load cell into housing.

BUILD THE ELECTRONICS
The following diagram shows the main components and connections of the electronics system we will be constructing.

**Figure 11:** Schematic of electronics parts and their connections.

There are a couple options for how to put this system together. One is to use a printed circuit board, also known as a PCB. A PCB is a layered board of conductive and insulating materials that maps electrical connections between components mounted to the board. If you were to take apart many household appliances, you’d find a PCB connecting their various electrical components. One way to make a PCB is to send in a custom file (.gbr) to a PCB manufacturing company, such as JLCPCB. Attached is a
zip file that you can submit directly to their order form. All automatic presets should be fine. Another way to make a PCB is to upload a custom file (.brd) to a Bantam PCB Milling Machine to carve electronic channels into a copper covered resin. Attached is a .brd file that you can upload directly to the Bantam Milling software.

ScaleMouse_final.zip
ScaleMouse_final.brd

Once you have your PCB ready to go, you can start connecting your components. Before connecting the HX711 components to the board, you’ll need to change the sampling rate from 10 Hz to 80 Hz. On the bottom of each HX711 component, there is a little connection that says “RATE.” Use a utility knife to sever this connection to set the module to high-speed sampling mode.

Figure 12: Severing connection on HX711 to change rate.

We recommend soldering female header pins onto the PCB for plugging in the Pro Micro and the Load Cell Amplifiers (HX711) to allow for easy removal and replacement if any of these components malfunction. To cut female header pins to the right appropriate size, account for one extra hole and cut along that extra hole using the cutting part of wire strippers. Aux ports, potentiometers, and the ethernet port can be soldered directly to the board. All components and header pins should be soldered to the side of the board without visible wiring to ensure the wire mappings are correct.
The location of each load cell on the final product is important to keep track of for once we get to the Arduino code. We will keep track of the location by designating them as northwest (NW), northeast (NE), southwest, (SW), and southeast (SE). Note how the bottom of the board will have east and west designations flipped to that of the top of the board.
Now it’s time to connect the load cells in your board to the electronics system. The goal here is to splice wires from the load cells to wires in the ethernet cable so that they can be easily plugged into your electronics system. Carefully strip the plastic casing off all load cell wires. Cut one side off the ethernet cable and similarly strip the inner wires.

Figure 14: Location designations on the top and bottom of the board.

Figure 15: Cutting and stripping main ethernet cable.
Figure 16: Separating and stripping individual cables within ethernet cable.

Twist the appropriate bare wire of the load cell with appropriate bare wire of the ethernet cable. Now it becomes important to reference back to location designations we made for the load cells earlier. You’ll need to connect the load cell wires to the ethernet according to the table below:

<table>
<thead>
<tr>
<th>Load Cell Wire(s)</th>
<th>Ethernet Wire</th>
</tr>
</thead>
<tbody>
<tr>
<td>NE White &amp; SW White</td>
<td>Blue Striped</td>
</tr>
<tr>
<td>NE Black &amp; SW Black</td>
<td>Green</td>
</tr>
<tr>
<td>NE Red</td>
<td>Brown Striped</td>
</tr>
<tr>
<td>SW Red</td>
<td>Brown</td>
</tr>
<tr>
<td>NW White &amp; SE White</td>
<td>Orange Striped</td>
</tr>
<tr>
<td>NW Black &amp; SE Black</td>
<td>Orange</td>
</tr>
<tr>
<td>NW Red</td>
<td>Green Striped</td>
</tr>
<tr>
<td>SE Red</td>
<td>Blue</td>
</tr>
</tbody>
</table>
Figure 17: Combination process of load cell and ethernet wires.

Solder the twisted wires thoroughly. Wrap electrical tape around the connection to secure and protect it. Once you have twisted, soldered, and taped all connections, you may want to tape the jumble of wires more neatly to the underside of the board to keep it all contained and relatively untangled.
You might be wondering at this point—what are these potentiometers for? When should I twist each dial? What are the direct and effective consequences?

The top potentiometer controls the overall load **magnitude**, and the lower one controls the **gain** of the mouse movement. Together, these potentiometers determine how the control box processes inputs and converts it into mouse movement. You’ll want to adjust these depending on your specific use case.

For example, if you are standing on the device, you will want the magnitude to be quite high, so you should rotate the potentiometer clockwise until it feels comfortable. If you are sitting to operate the device, you may want the magnitude fairly low, and you should turn it counterclockwise. If you want the mouse to move at a high velocity when you give it commands, then turn up the gain potentiometer (again, rotating clockwise). For a slower mouse, turn the gain down.
Now for an explanation of what’s actually happening. The “magnitude” potentiometer determines what the scale sees as the “maximum” load. If the actual load is greater than the magnitude defined by this potentiometer, the code will limit it to the magnitude value. In other words, the load saturates at this maximum magnitude value. The maximum load determines the other load values in the figure below – the deadzone (where mouse command is zero), saturation (where mouse command is maximum), and the range where the mouse command is linearly proportional to the load value between the deadzone and saturation values.

This variable range determines the “sensitivity” of control over the mouse. Currently, sensitivity must be changed in the program itself, although it would be easy to reassign one of the potentiometers to adjust sensitivity. Sensitivity is the fraction of the overall range of input values (from 0 to the magnitude) that will have the variable minimum to maximum output. Within this range (shaded gray in the figure below), there is a direct correlation between the input (load applied) and output (mouse command). Between an input of 0 and the start of this range, there is a “dead zone” in which any input is considered noise and thus the output is 0. Outside of the range, the output is saturated and will be the maximum output value regardless of varied input. The graph visually depicts the function made up by the “magnitude” and “sensitivity” that will define how the device converts a given input (force applied) into an output (mouse control).

![Graph of Mouse Command vs. Load Cell Value](image)

**Figure 19:** Plot of Mouse Command vs. Load Cell Value, incorporating potentiometer controls of “magnitude” and “sensitivity” values.

The other potentiometer adjusts the mouse “gain.” The mouse command (between -1.0 and 1.0 in the horizontal and vertical directions) is multiplied by the “gain” and rounded to the nearest integer. This determines how many pixels to move the mouse cursor during this single loop through the program (the program loops about 66 times per second). As an example, if the mouse command is 1.0 in the horizontal
and vertical directions, and the gain value is set to 8, then the cursor will be moved 8 pixels to the right and 8 pixels upward during this loop.

We have added three aux ports to allow additional functionalities (by default, these act as mouse left-, middle-, and right-click buttons). To use these ports, solder a normally-open switch to a 3.5-mm audio jack so that the T (tip) and S (sleeve) contacts are connected when the switch is triggered. The controller treats a closed circuit as a pressed mouse button.

PROGRAM THE MICROCONTROLLER

Now grab your microcontroller. If you want to use the provided PCB design, then it must be a Sparkfun Pro Micro 5V development board. If you are not using the PCB (for example, if you want to make your own circuits on a solderless breadboard), then you may use a different microcontroller if it uses an AVR processor. This is because the Joystick and Mouse libraries require an AVR-based board. Models such as the Arduino Micro or Leonardo, Adafruit Metro, or Pololu A-Star work well.

Upload the provided Arduino program. This program uses 3 libraries:

- **Joystick.h** allows the microcontroller to act as a USB joystick.
- Mouse.h (included by default on AVR boards) allows the microcontroller to act as a USB mouse.
- **HX711-Multi** enables the board to read results from more than one load cell amplifier

```c
// Libraries
#include "Joystick.h"  // https://github.com/MHeironimus/ArduinoJoystickLibrary
#include "HX711-multi.h" // https://github.com/compugian/HX711-multi
#include <Mouse.h>

// Settings
boolean is_joystick = false; // true if you want this to be a 2-axis joystick instead of mouse
boolean use_nesw = false; // if true, vertical axis controlled by NW/SE, horizontal by NE/SW
boolean read_pot_values = true; // adjust magnitude and mouse speed (gain) with potentiometers
boolean send_serial_output = true; // send data with serial communication
```

**Figure 20:** Arduino code used to import necessary libraries and establish settings.

After importing libraries, the program has a few settings to be chosen by the user. To use the device as a joystick instead of a mouse (to use it as a thumbstick with an Xbox Adaptive Controller, for example), then set `is_joystick` to true.

By default, the load cell readings are cross-coupled/combined to get the mouse directional command. If you would like to decouple the load cells so that one pair directly controls up/down and another pair controls left/right, then set `use_nesw` to true. The program will use the NW and SE load cells for up/down and NE and SW for left/right.

To set the magnitude and gain values at the beginning of the program and not use the potentiometers to change them, change `read_pot_values` to false.

To send data back from the microcontroller through the serial port, set `send_serial_output` to true. In the Arduino IDE, you can see this data by opening Tools>Serial Monitor after uploading the program.
Figure 21: Arduino code used to establish mouse commands and movement.

A few more blocks of code brings you to the mouse control variables. A few notes:

- Do not change the value of `max_pot` unless you have a good reason to.
- The upper-limit of mouse speed (in pixels per refresh frame) can be changed with `max_mouse`.
- The values of `gain` and `magnitude` are immediately overwritten if you set `use_pots = true;`

```cpp
void loop() {
    read_pots();
    read_load_cells();
    process_data();
    mouse_or_joystick();
    wait_for_next_frame();
}
```

Figure 22: Arduino code for the main program loop.

The main program loop is shown above. It runs a little more than 60 times per second. The name of each subfunction describes what it does, and comments inside each subfunction describe the operations fairly well. Please see the program itself for more details.

MORE APPLICATIONS

Congratulations! You have created your very own alternative directional controller! You have also picked up some skills that will open a new world of possibility. While these load cells are most commonly found in a traditional bathroom scale, people have gotten creative in using them for all sorts of things. Some have used it to monitor the weight of their beehives to reduce maintenance; others have used it to monitor the weight of their cat’s food bowl to refill automatically. The arduino Keyboard and Mouse Control package can be used for any number of keyboard or mouse alterations you can think of. The possibilities are endless! Get creative!