VR Disability Accessibility

Team 27: Evan Miller (emm2), Justin Zhou (jezhou2), Vinith Raj (vinithr2)
ECE 445 Design Document: Fall 2020
TA: Dean Biskup

Abstract

This report highlights our mission to create an accessory for VR devices, built specifically for users that have upper body mobility issues. In order to accomplish this goal, we investigated ways to provide input using your feet, as well as determining the best method to convert this input into usable signals for VR software. The paper begins with a project overview, and continues with details about the design of our various components, the assembly of our hardware, and the frameworks we used to test the functionality of our subsystems. Verifications and results for the subsystems, as well as details about the costs incurred, are presented afterwards. We conclude with details of our accomplishments, uncertainties, and ethical considerations.

Table of Contents

1 Introduction

1.1 Problem and Our Solution

As the world advances, virtual reality (VR) devices are becoming increasingly popular. As VR becomes more advanced and consumer friendly, there is good potential for developers to make more gaming, resource, and teaching applications. Currently, however, VR is inhospitable for people with disabilities. This is especially true of people with disabilities of the arms or hands, because the devices are largely operated through hand-held controllers. There are already full body capture devices and haptic feedback devices available in the market, but nothing built specifically for people with upper body mobility issues.

There are a number of controllers in the market, like the QuadStick FPS game controller [1] and Microsoft Adaptive Controller [2], meant for people with disabilities. However, these controllers are expensive and not developed to be used in a VR environment. There is one "leg-based" controller for VR on the market, the 3D Rudder Foot Motion Controller, however, there are a number of complaints about it. Specifically, it is difficult to control and slide around on the ground. It should also be noted that there is praise, largely from people with upper body disabilities, but this support can be attributed to the fact that it is the only device that addresses their issues on the market [3], so there are few products to compare to.

Our design is broken up into three major sections: the Physical Design, the Electronics Hub, and the PC software. We use two load cell sensors for each input, with a combined maximum load of 150 kg per location with four locations. The physical input section is in the Physical Design block of Figure 1.1. The analog data is then digitally converted by two converters. These send their data to a microcontroller that parses and sends data to whatever personal computer (PC) the VR systems are connected via the USB-to-Serial converter. This section of the design is in the Figure 1 Electronics Hub block. The PC should then apply these signals as control inputs. We were unable to fully implement the bridge between PC inputs and a game system due to time limitations, but we were able to prove that the data could be parsed into different inputs and received by the PC for further processing, and believe that it would not take significant development to finish the software part of our design. The unfinished part is the xInput line of the PC block in Figure 1.1, but that specific signal type could be replaced with keystroke presses or any number of game controller protocols to be functional.

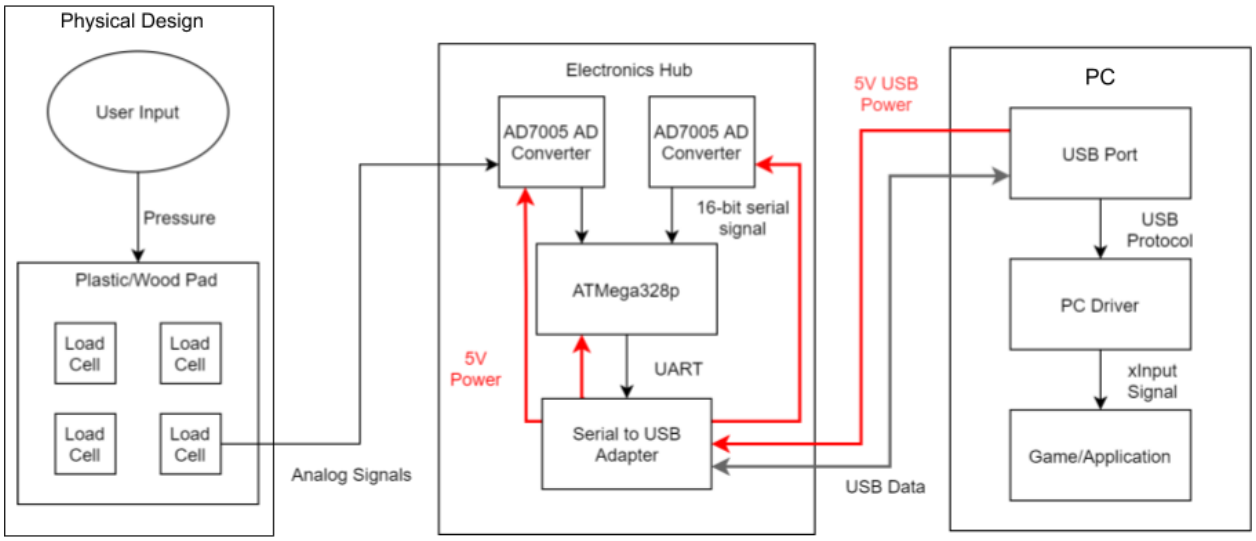


Figure 1.1: Block diagram of final design

1.2 High Level Requirements

The high level requirements for this project are as follows:

- **Latency**
  The input should be smooth, without volatile signals and latency should be at most 60 ms to ensure the user has an experience with minimal lag.
- **Compatibility**
  The inputs are compatible with most VR software on computers.
- **Comfort**
  Foot controls can be used without discomfort over a period of one hour.

1.3 Block Changes Over the Semester

We began the semester with a power system with a linear regulator that utilizes batteries to power our design, and a bluetooth connection that would wirelessly send data. We realized that it would be much simpler to use the 5 volt power provided by a USB port than it would be to make a power system with a 5 volt linear regulator, and since a USB to serial converter can provide both, we removed the power system and wireless bluetooth parts of our design, and powered it and transferred data via the USB port. In the final product, the voltage source is stable enough to provide power to all the components without initiating brown-outs, and we achieved stable voltage levels. However, there were variations in the smaller voltage levels recorded on our sensors, which could be potentially attributed to the +/- 5% on the voltage output of the USB while under load. This is the only change that was made to the block diagram.

2 Design

2.1 Design Procedure

In our design, we aimed for a simpler, more stable controller than the popular devices in the market. Instead of having the user sit down and rock their feet on an unstable ball, we wanted a design that used pressure sensors under the feet to allow for more comfort and accuracy for user input. This also allows for four inputs (front and back for both feet), which is three more than the one analog input from the gyro-based design that is currently sold in the market. This design also allowed for comfortable pads to be put on top of the sensors, improving the feel of the device over long term use. Since the controller is stable and not likely to move during use, it can also directly pull power and send data using a USB port instead of having to have its own power source and being forced to send data wirelessly.

2.1.1 Physical Design Procedure

For our physical design block, we brainstormed the different potential ways a person could make physical inputs into a program. We then came up with a number of basic designs, like wearable pressure pads, but then settled on a footpad controller, since it could take multiple inputs depending on how a foot's pressure is applied and seemed like it would be less uncomfortable to use. We then discussed different design options, like making the inputs pads that could attach to shoes in order to accommodate all foot sizes. We also looked at other tools that were on the market and eventually settled on the four input footpad design. The current design has a solid board on bottom, a ring around each sensor to facilitate sensing, and a pad on top of each set of two sensors for user comfort. Each pad is separated so that the pressure applied to the entire board doesn't get passed on to the incorrect controllers and also because it makes it easier for the user to determine where the input locations are so that they can navigate the inputs while they have their headset on.

2.1.2 Electronics Design Procedure

For our Electronics Hub block, we have two analog to digital converters, with two channels each. We made this decision, because it was the bare minimum number of converters necessary, and fewer parts means a smaller PCB and electronics box as well as a smaller chance of part failure. We found that swapping between channels on the converters to access both inputs increases latency from 10 ms to around 60 ms. While the final block design only has two converters, using four converters instead and not swapping between channels would allow for 10 ms response

times and still be within the accepted power range of 500 mW (5 V * 100 mA). Figure 2.1 shows how the final circuit for our PCB was designed, its main purpose being to hold each of the parts and chips for our design, and integrate them together.

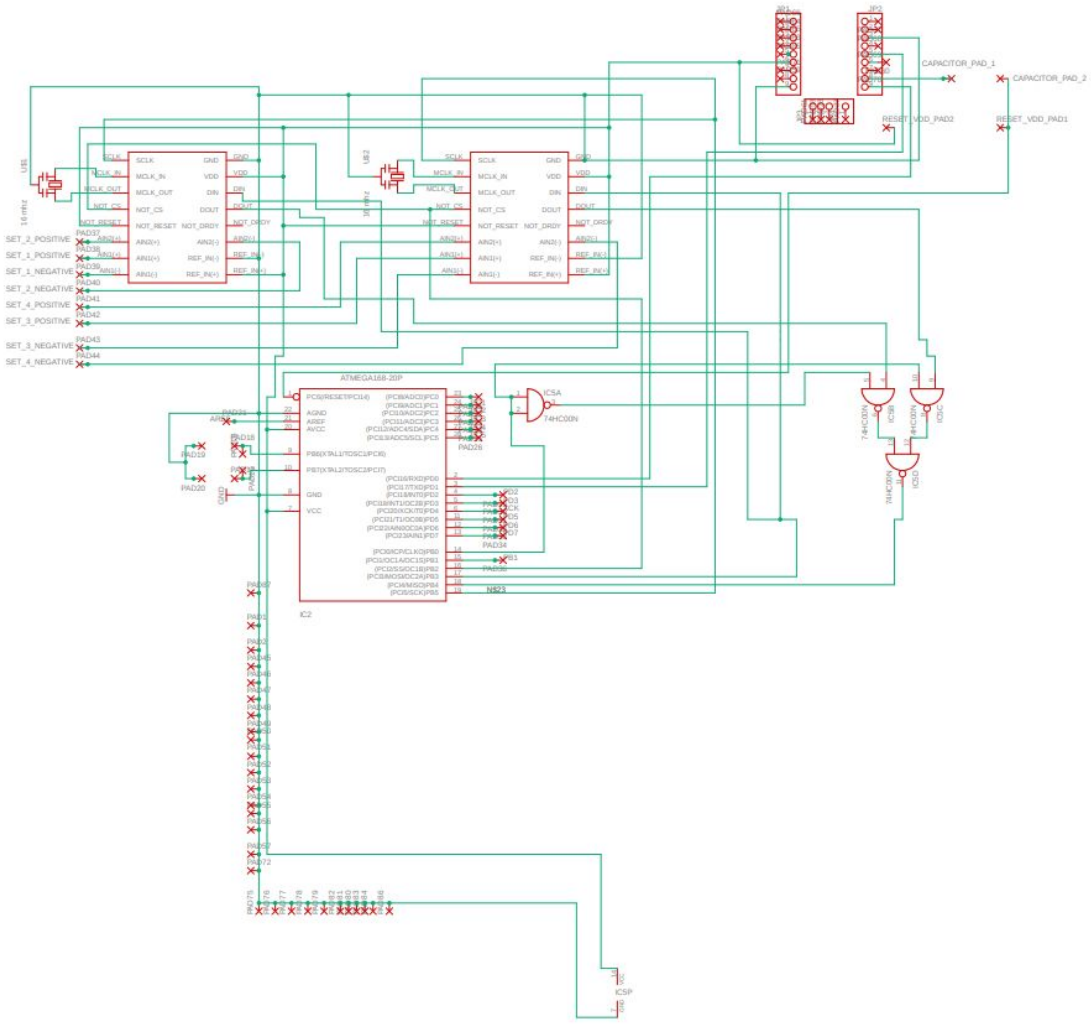


Figure 2.1: Full PCB schematic

Since power from the standard USB port and the power consumption of each of our parts is relatively low, there are no specific equations that went into determining if our power design was functional, other than adding together the maximum power draws of each part and ensuring that their sums were under the maximum power output of the USB port. We did have to determine the latency in milliseconds for verification, which is

$$1000 / IPS = LMS \quad (1)$$

where IPS is the inputs per second and LMS is the latency in milliseconds. To find a converter that could output at these speeds, we looked at a number of converters and calculated their operations per second, which is

$$CF/ (SC + NOB + EB) = OPS \quad (2)$$

where CF is clock frequency, SC is setup cycles, NOB is number of output bits, EB is number of end bits, and OPS is operations per second. If the LMS equation was below 60 ms where the IPS was substituted by the OPS, then it was a viable converter.

2.1.3 Software and Driver Design Procedure

For our PC block, we have code for our microcontroller that polls each of the converters in order to find their conversion values and output that data back to the PC via the USB-to-Serial adapter. We then use the Arduino drivers to pick up these signals. The microcontroller, based on it's analog to digital converter select at the time checks if the input value is outside of the threshold since the last input in order to determine if there was a change in input value, and if there was , it sends a signal back to the Arduino In System Programmer with the appropriate conversion value and from which sensor input it was obtained. The final part of our project, which was not implemented, would be to use a standard Arduino library to make the inputs act as keystrokes. The output could easily be demonstrated in a sample Unreal Engine program. This choice of programming tool was chosen, because if there is already a functional bootloader that comes with our microcontroller, then using it simplifies design and reduces potential bugs with our programs.

2.2 Design Details

2.2.1 Physical Design Details

The physical design, in Figure 2.2, was made with the comfort of the user in mind, with padding put on top of each of the sensors and a stable board acting as the bottom layer to prevent sliding and provide stability. Each of the different sensor pads are snugly placed next to each other in a raised area of the board to prevent them from falling out/off, with the sensors attached to the snugly fitted, free floating pads so that they can compress and decompress. A 3-D printed ring is attached between the sensor and top pad to help compress the edges of the sensor to increase sensitivity. The pads are made of a solid layer of wood for stable inputs, followed by a hard layer of foam to prevent tearing, a soft layer of foam for comfort, and a vinyl layer for endurance and to keep the foam in place. The electronics hub box was made separately, and it can be designed

to be smaller and fit in between the two sensors, keeping the hub and controller together, instead of forcing someone to walk around with both of them separately. Each of the pads have a number and direction on the bottom of them so that they can be placed back into their locations easily if they need to be removed for access to the sensors. In the side of each raised barrier that keeps in the pads are holes to allow wires to go through and prevent them from wearing down from the friction of the pads on them if they were draped over the sides instead.

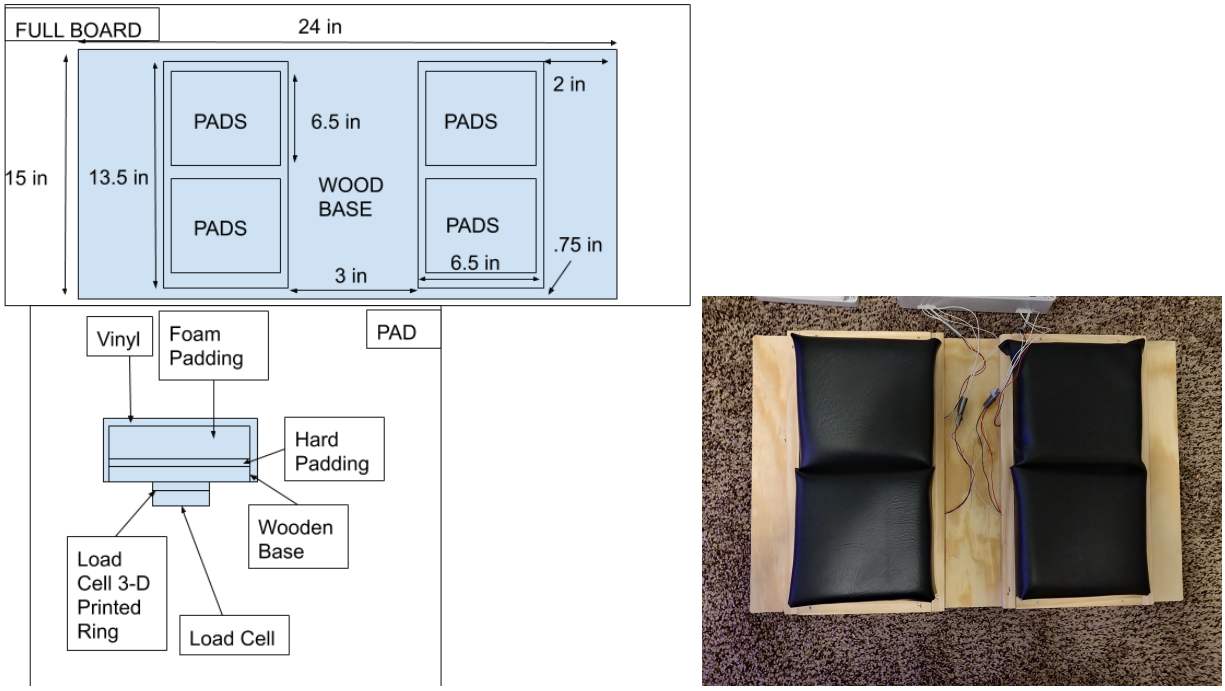


Figure 2.2: Final physical design schematic and controller board

2.2.2 Electronics Design Details

2.2.2.1 Microcontroller

For the programmable microcontroller, we used an ATMega328p-pu microcontroller to implement the controller logic required for the project.

We considered a number of different microcontrollers, such as the ATMega2560 or the Raspberry Pi. We ultimately chose the ATMega328p-pu because it had all the features we needed, with enough extra inputs to accommodate changes in our design or bug fixes that would need extra inputs/outputs.

We also chose the ATMega328p-pu because the chips that we could find online included Arduino bootloaders, which means that it did not require additional effort to program or bootload the chips. While bug fixing our microcontroller, we realized that both a ceramic oscillator and

the internal clock of the microcontroller were too inaccurate, so we settled on using a crystal oscillator for maximum accuracy and a location was left on the PCB for that part to be installed. Figure 2.3 shows its location on the schematic and the PCB board.

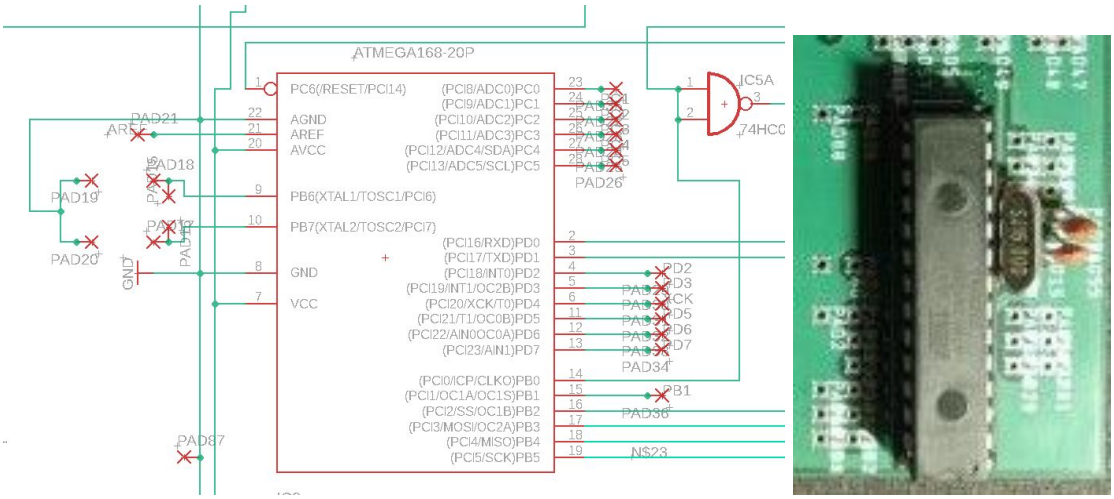


Figure 2.3: ATMEGA328p-pu in schematic (left) and final PCB (right)

2.2.2.2 AD Converters

We used AD7705 AD Converters for this project. We chose this chip because of its maximum serial output speed, its ability to take two inputs, and its 16-bit serial output (for accuracy of measurement). In the documentation, communication also required 4 cycles for startup polling, and there were no end bits. According to Equation 1 and Equation 2, the OPS is 500/(4 + 16 + 0) = 25 operations per second and LMS is 1000/25, which equals 40 ms latency. Ceramic oscillators are used to keep a 500 Hz speed. Figure 2.4 shows the location of the converters in our schematic, and where one of them is on the PCB.

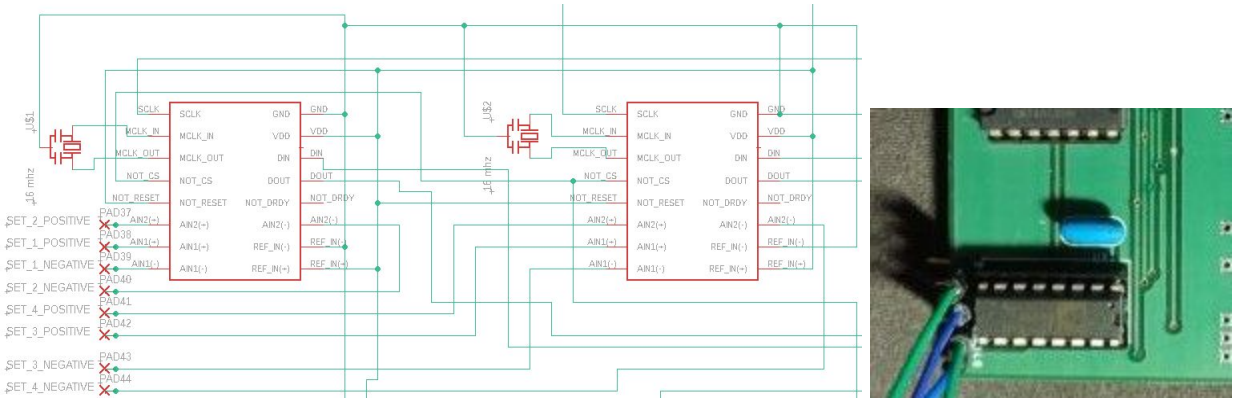


Figure 2.4: Converter locations in schematic (left) and one location on final PCB (right)

Another AD Converter that we considered using was an HX711 AD Converter. This AD Converter had a higher resolution of 24 bits, however the maximum output speed is only 80Hz, and because it needed to serially output even more data than the AD7705, the denominator of Equation 2 was much larger and the numerator was much smaller than with the AD7705, resulting in a significantly lower OPS. This OPS is much too slow for the latency requirements for our project, and the converter was subsequently rejected.

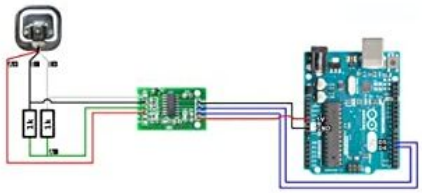
2.2.2.3 Load Cells

We use 50kg strain gauge load cells for this project. We chose these cells because they are able to support the required weight for this project and have a sufficiently high sensitivity voltage.

These load cells have a maximum excitation voltage of 10V, which is well above the 5V that the project runs off of. The sensitivity of the load cells are 1.0 +/- 0.1 mV/V at the maximum rated capacity of 50kg.

We also considered a number of different ways to detect if pressure is being applied to the pad by the user, such as using piezoresistive force sensors or spring-based potentiometer systems, however there were drawbacks and limitations of those designs compared to a strain gauge load cell-based design.

In our final design, we use one load cell per input pad on the controller, for testing's sake. The load cells were wired to the AD Converters based off of the reference image in Figure 2.5, however the AD converter pictured is different from the AD7705 we used.



Using a Single Load Cell 1x50kg by Adding 2 Resistors

-Connect outer wire(white and black) of load cell to E+ and E- outputs of hx711 module.

-Connect middle cable(red) of load cell to A+ input of hx711 module.

-Connect two 1k resistors to A- input of hx711 module. Then the other end of resistor to white and black wires.

-Connect GND of hx711 module to board GND, and VCC to board 5V pin.

Figure 2.5: Reference Wiring Diagram for Single Load Cell

2.2.2.4 NAND Mux

Since we use the SPI communication protocol to interact with each of the analog to digital converters, we multiplexed their outputs, so that when communicating to them, only the selected converter outputs it's data to the MOSI. While looking at MUXes, we realized that it would be simpler to implement a 2 to 1 mux using nand gates on our own, since there are very few alternatives on the market, and most muxes are larger or more complicated than our project required. Since one NAND chip can be used to act as a mux, we use one to select between the two converters. Figure 2.6 shows the simple mux design and its location on the PCB.

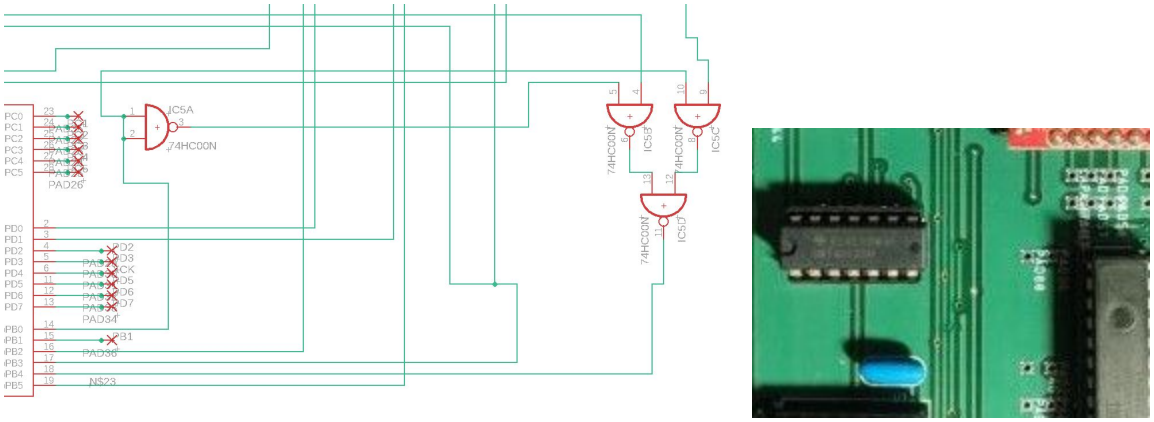


Figure 2.6: 2-1 NAND gate multiplexer schematic (left) with its location on the PCB (right)

2.2.2.5 USB to Serial Adapter

In order to send signals between the PC and the microcontroller, we use a USB to Serial Adapter. We used the SparkFun USB to Serial Breakout FT232RL which uses the FTDI FT232RL chip to perform serial to USB conversions.

The reason we chose to use a breakout board instead of the FT232 chip itself is because the breakout board comes with a micro-USB port included. This saves us from having to source and solder our own USB port, allowing us to get the project done within the timeframe. As shown in figure 2.7, we include a designated location for the FT232 board in our schematic. The microcontroller communicates with the FT232 board using the TXD and RXD pins. We also use the VCC provided by the USB port to power the rest of our board, eliminating the need for a dedicated power supply.

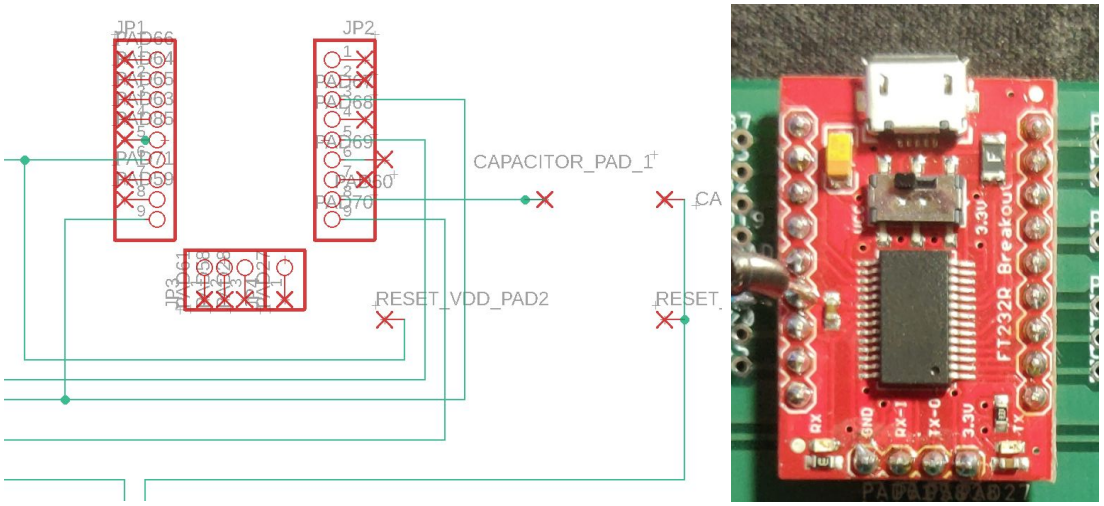


Figure 2.7: USB to Serial adapter slot in schematic (left), USB to Serial board on PCB (right)

2.2.3 Software and Driver Design Details

In terms of software design, we use code on the microcontroller to read the input from the AD Converters, check if the values reach a certain threshold, and output the proper controller input if it does reach the threshold. Figure 2.8 contains the high-level software flowchart for our project.

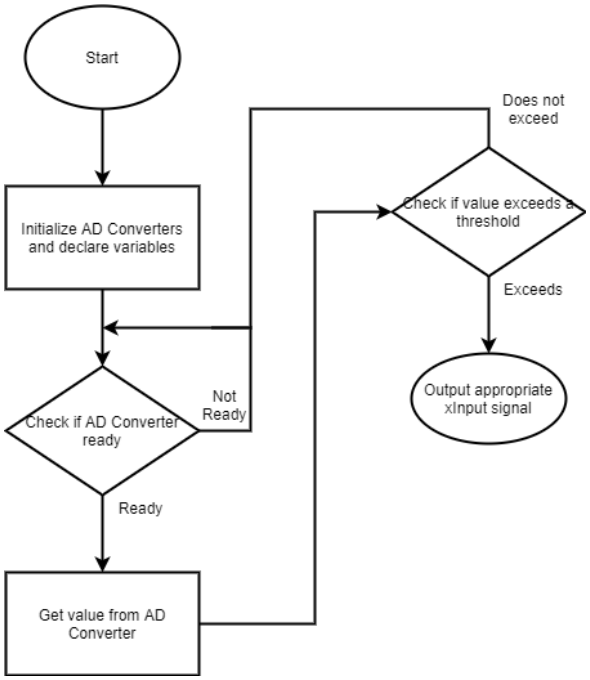


Figure 2.8: High-level software flowchart

Table 2.1 shows the mapping of pins from the microcontroller. In the final design of the schematic, we specify pin 11 as MOSI, pin 12 as MISO, pin 13 as SCK, and pin 10 as CS. MOSI goes to the DIN pin on both of the AD7705s; MISO goes to the output of the NAND MUX for the DOUT signals from both the AD7705s; CS goes to the CS pin on both AD7705s; SCK goes to the SCLK pin on both of the AD7705s; and we have pin 8 set as the MUX selector to choose which chip's data to read. With these pins, we initialize and poll data from either AD7705 chips.

Table 2.1: Microcontroller pin mappings

ATMega328p Pin (in software)	
Pin 11 (MOSI)	AD7705 DIN (both chips)
Pin 12 (MISO)	MUX Output
Pin 13 (SCK)	AD7705 SCLK (both chips)
Pin 10 (CS)	AD7705 CS (both chips)
Pin 8 (MUXSelect)	NAND MUX Selector

The final version of the software is very similar to the high-level flowchart, however there are a number of changes. One change includes throwing out any impossible values (e.g. the value from an AD converter spikes/dips very high/low) caused by interference or a bad connection. Another change is instead of checking for load cell values exceeding a set threshold, we check for an increase in value between two consecutive AD Converter reads. This indicates that the user has applied more pressure to an input and intends to activate it. This helps prevent accidental inputs from being sent when the user is not intending it. The following code snippet in Figure 2.9 shows an example of the finalized code:

```
void loop() {
  digitalWrite(MUXSelect, LOW);

  v1 = ad7705.readADResult(AD770X::CHN_AIN1);

  if(v1 - v1prev > pressLimit && abs(v1prev - v1) < range) {
    Serial.println("B");
    v1prev = v1;
  } else if (abs(v1prev - v1) < range) {
    v1prev = v1;
  }
}
```

Figure 2.9: Example snippet of finalized code with error checking

The code first stores the result of the AD Converter reading into v1. Then, if the new v1 increases enough (i.e. the user wants to send the input), but is not an erroneous value, then it outputs a signal to the PC, updates the previous value, and loops. Otherwise, as long as the value is not erroneous, it updates the previous value and loop.

3 Verification

3.1 Physical Design Verification

For the physical design, we had to verify that the controller is comfortable to stand on for half an hour - roughly the average amount of time a user spends in VR. We surveyed a number of people on their level of comfort after standing on the controller for half an hour, on a scale of 1 to 10. We expect a passing score to be a 7.5. This verification clearly passed, with an average comfort rating of 8.5. The controller must also not break when a person stands on the controller, which it did not break at all during testing. This shows that our physical design is fully functional and has met our requirements.

3.2 Electronics Design Verification

Our electronics requirements verify that the electronics hardware portion of our project works as intended and fulfills our high-level requirements for latency and smoothness of input.

To test most of the electronics requirements, we needed to ensure that all the different parts were functional. Unit testing each of the chips showed that every piece was functional. Then, to verify that the overall design is functional, we assembled and tested the circuit on a breadboard, and then the PCB. This verification was successful since behaviors between the breadboard circuit and PCB circuit were identical.

To test the latency requirement, we uploaded code to the microcontroller which polled the AD Converters and output the value to the PC. We then counted the number of inputs we received within one second to get the latency of the system. The final circuit did not meet this requirement, having a latency of 66 ms as opposed to our required 60 ms. There are a few design changes we could have made that would have allowed us to get down to as low as 10 ms of latency. Due to time constraints, however, we were unable to implement these changes.

Another requirement is that there must not be a greater than 1% error rate in the AD Converter readings. For one of the AD Converters, this requirement was met, however in the second AD Converter we were using, it would very regularly give false or erroneous data (large spikes/dips in the readings). We are unsure as to the exact cause of these errors, however, we have tried multiple solutions such as replacing the chip, replacing the ceramic oscillator, and changing the microcontroller code. None of these changes would fix the errors we were getting from our second AD Converter. However, some other changes would be to try to replace the ceramic

oscillator with a crystal oscillator or to try on a new PCB. However, due to time constraints we could not attempt other options.

3.3 Software Design Verification

To test that our software can properly parse the AD Converter values, we uploaded code to the microcontroller that would simply output the readings from the AD Converter directly to the PC. We can then test that the values output by the microcontroller changes appropriately when pressure is applied to the load cells. Figure 3.1 shows a labeled graph showing the AD Converter values changing when weight is placed on the load cell.

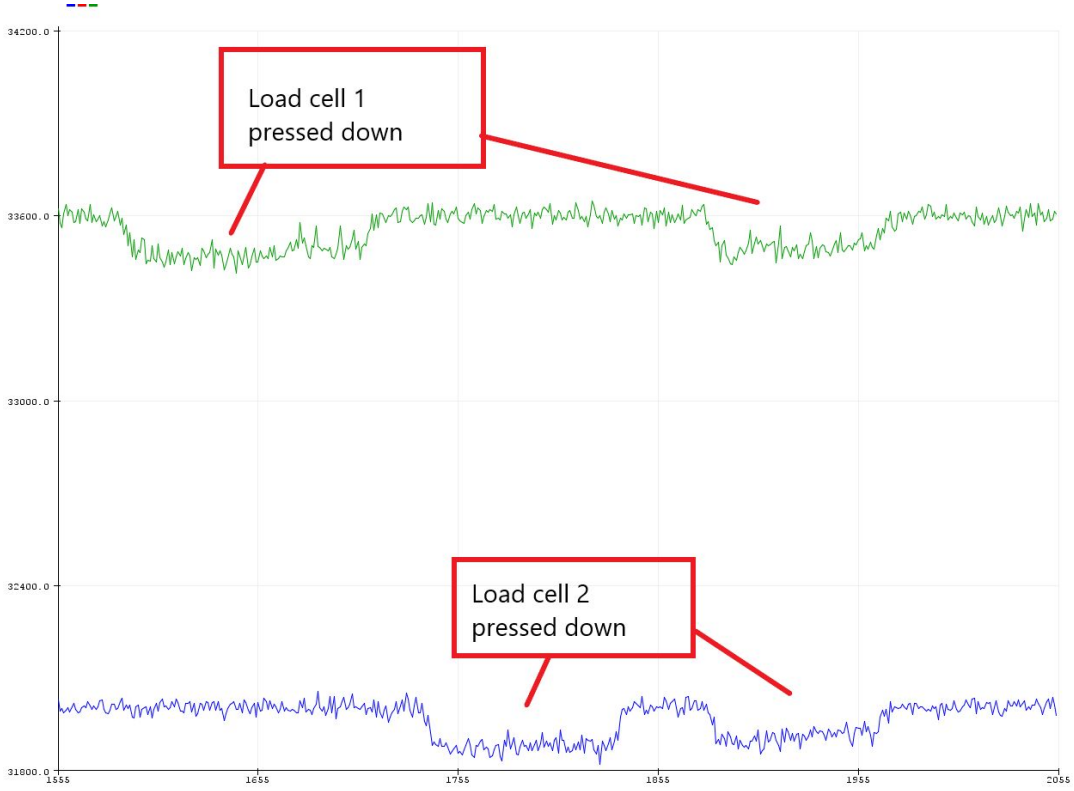


Figure 3.1: Graph of load cells being pressed down

Another software requirement is that it sends the appropriate input signal to the PC when the user steps on an input. Due to a firmware issue with the microcontroller, we were unable to directly send keyboard inputs to the PC. However, the microcontroller is still able to print to the computer, so we were able to print out the input in our demonstration. Additionally, there is a software library that allows the microcontroller outputs to be used in the Unreal Engine game engine, effectively fulfilling the requirement that the controller can be used in games or other software. Due to time constraints, this was unable to be included in the final product, however it would have allowed us to meet our requirements.

4 Costs

4.1 Parts

Table 4.1: Part Cost Analysis

Part	Part Number	Unit Cost	Quantity	Cost
ATMega328p Microcontroller	ATMega328p-pu	$7.50	1	$7.50
AD7705 AD Converter	AD7705BNZ	$11.19	2	$22.38
USB-to-Serial Adapter	FT232RL	$15.95	1	$15.95
SN75HC00N Quad 2-Input NAND Gate	SN75HC00N	$0.258	1	$0.258
50kg Load Cells	SEN-10245	$10.95	4	$43.80

4.2 Labor

Table 4.2: Labor Cost Analysis

Team Member	Hourly Rate	Total Hours	Total = Hourly Rate * 2.5 * Total Hours
Justin Zhou	$40.00	200	$20,000
Vinith Raj	$40.00	200	$20,000
Evan Miller	$40.00	200	$20,000

4.3 Total Cost

Table 4.3: Total Cost

Parts Subtotal	Labor Subtotal	Grand Total
$89.89	$60,000	$60,089.89

5 Conclusion

5.1 Accomplishments

In the end, our design accomplished much of what we set out to do, and provided an acceptable benchmark upon which we can improve our design in the future. We find that our design is comfortable and stable for users, and that the granularity of the responses we obtained from our sensors are sensitive enough to detect small differences in weight, potentially allowing for precise movements to be made. The code is also able to, mostly, mark outlier data, and the inputs (for one of the feet) are stable. The data attributed to each foot is correctly parsed and consistently attributed to the correct sensing location.

5.2 Uncertainties

Despite the successes, there are still some uncertainties with our design. We found that the latency is 6 ms above the design goal, however we know why the latency increased, and already have a solution to that issue. We will add two more AD converters and instead of swapping channels, we will just use one channel per chip and a 4-to-1 MUX to swap between converter outputs. This change will theoretically decrease the latency down to around the 10 ms range, which stays competitive with other game controllers. Another uncertainty with our design is in the extreme spikes in voltage data we occasionally see from one of our converters. Since the other converter is not showing this issue, we believe that the extreme spikes are caused due to a faulty connection or AD converter and that fixing that connection or replacing that converter would fix the issue of outlying data spikes. If the problem persists, then it could be an issue with noise in the PCB design and we will have to redesign our PCB. If none of these solutions fix the problem, then clever coding workarounds will be better used to deal with the voltage spikes. With regards to the PCB design, there is also an issue where there aren't enough VCC connections for clean REF + inputs to each of the converters and for each of the load cells to access the 5 V from the USB, so we are forced to use a breadboard to safely provide power to all of the parts. In a future redesign, all of the REF + inputs will be hardwired to VCC and a number of VCC outs will be lined up so that they can be easily accessed by the load cells. The final concern with our design is that the parsed signals are not used directly as controller inputs. This concern has already been addressed in Section 1. All of the issues that we have encountered with our project have prospective solutions, so we do not think that the specifications should be lowered or changed to accommodate a less robust product.

5.3 Ethical Considerations

Due to the project dealing with VR, some obvious health concerns are evident. Virtual reality headsets are known to cause nausea and anxiety, especially after extended usage. People with disabilities might be at a heightened risk. In addition, VR headsets can also cause eye strain, and even anxiety from all the stress of being in the virtual environment. These issues need to be addressed as we develop this technology, and make sure our project does not affect users through both accidental and intentional misuse. With regards to testing specifically, only the project members tested the technology, but we did survey willing participants on the overall comfort of the device. Since development time is relatively low, we are not certain that the project is ready for beta testing with the target audience, however, alpha level testing can be completed by the project team, as well as some able bodied participants that are aware of the minor risks. During assembly of the electronics, the main danger is soldering. To mitigate the dangers of accidental burns, we placed reminders around the soldering equipment to keep it unplugged when not in use, and we placed the tool in the holder to mitigate the risks such as burns and fire.

With regards to ethical issues relevant to the project, IEEE Code of Ethics Section 1.1 [4] discusses the importance of safety and health of the public, and we believe that we should ensure that the project is safe for any user. Considering our target audience is people with disabilities, for future testing, we need to be even more considerate of their needs to be safe and healthy while using this project. Furthermore, we need to be clear about the effects of the technology and the associated risks, and ensure we receive proper consent from testers via waivers. ACM Code of Ethics Section 2.7 [5] is also quite relevant to our project because it is important for the public to understand the technology we are using and how it works. This allows for projects like this to gain even more attention. Virtual reality has a lot of potential, and it could yet be expanded to help disabled individuals. By educating the public through proper conduct and safety precautions, as well as helping them to understand the consequences of this technology on society, we can collectively learn and continue our efforts in this field.

References

[1] “QuadStick FPS Game Controller.” *quadstick.com* QuadStick, 2020, https://www.quadstick.com/shop/quadstick-fps-game-controller. September 17th, 2020.

[2] “Xbox Adaptive Controller.” *microsoft.com* Microsoft, 2020, https://www.xbox.com/en-US/accessories/controllers/xbox-adaptive-controller. September 17th, 2020.

[3] “3D Rudder Foot Motion Controller for VR and PC Games and Applications with VR Mode, Foot Keyboard Mode, Foot Mouse Mode, Foot Joystick Mode.” *amazon.com* Amazon, December 29, 2016, https://www.amazon.com/3dRudder-Foot-Motion-Controller-Applications/product-reviews/B01MS26PFK. September 17th, 2020.

[4] “IEEE Code of Ethics” *IEEE,* https://www.ieee.org/about/corporate/governance/p7-8.html. September 17th, 2020.

[5] “ACM Code of Ethics and Professional Conduct” *ACM,* *https://www.acm.org/code-of-ethics*. September 17th, 2020.

Appendix

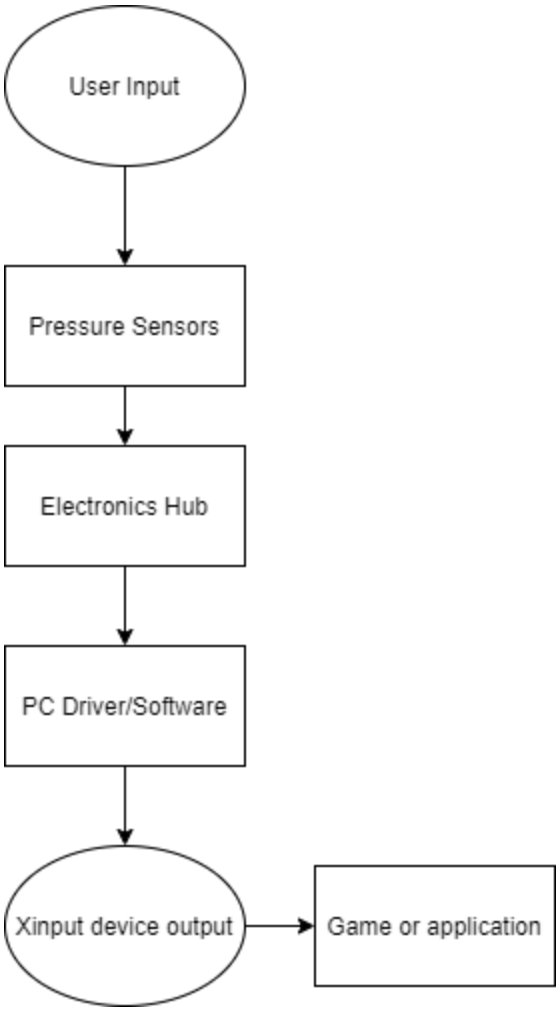


Figure ii.1: First iteration of block diagram

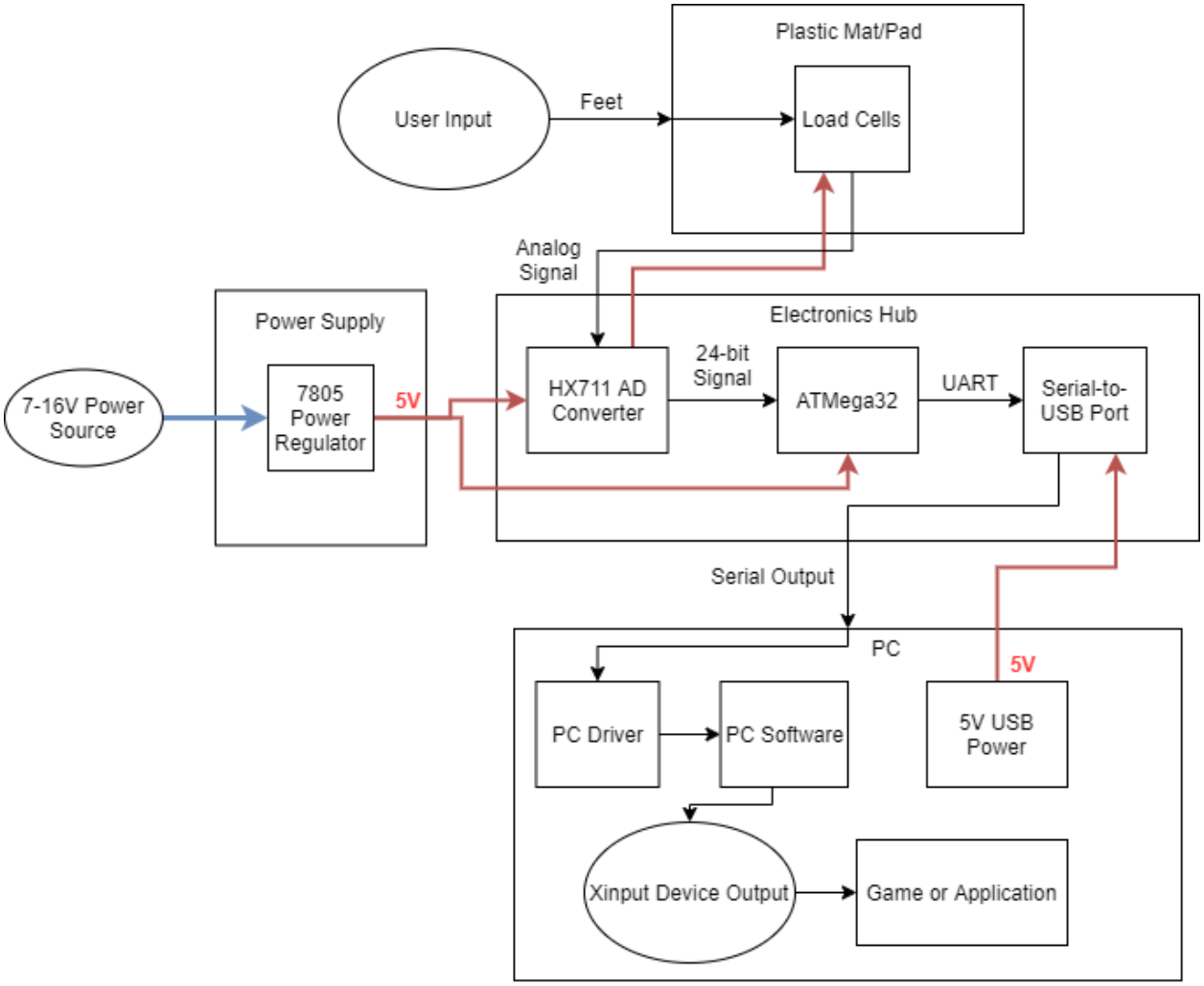


Figure ii.2: Second iteration of block diagram

Table ii.1: Physical Control Design Requirements

Requirements	Verification
1. When a 10kg load is applied to the top of the physical design foot pad, at least a .05 mV difference is measured.	Equipment: Multimeter Test Procedure: 1. Attach the load cell wires to appropriate locations according to schematic on a breadboard 2. Use a multimeter to measure the change in voltage 3. Record the average value output by the ATMega328 over 30 seconds with no weight on the foot pad 4. Place the 1kg weight onto the foot pad and record the average value output by

	the ATMega328 over 30 seconds 5. Repeat steps 1-4 with all load cells. Presentation of Results: - The results will be presented in a table which shows the difference in average values with and without the weight.
2. After standing on top of the board for 30 minutes, a user does not feel more uncomfortable than if they were standing on plain carpet.	Method: Ask roommates and friends to test Test Procedure: 1. Survey testers on scale of 1-10 2. Aggregate scores should average 7 or higher, with 6 being explicitly stated as the comfortability of carpet.
3. When a person of average American weight (+/- 20 kgs) steps onto the foot pads, they are able to stay standing on them without sliding or falling off for 10 minutes.	Method: Asked testers about comfortability Test Procedure: 1. Survey testers on scale of 1-10 at 10 minute mark 2. Aggregate scores should average 8 or higher

Table ii.2: Electronics Requirements

Requirements	Verification
1. The signals on the PCB arrive at the correct locations.	Equipment: Breadboard Test Procedure: 1. Wire up the breadboard according to the PCB schematic. 2. Confirm functionality of other requirements and expected outputs to PC 3. Rebuild schematic on PCB 4. Confirm that the functionality from the PCB matches the expected response from the breadboard Presentation of Results: - A binary confirmation from the product tester
2. The time from when a user applies pressure to the load cell to when the signal is received by the computer must be at or below 60ms.	Equipment: ATMega328, AD7705, Load Cell Test Procedure: 1. Connect the USB-to-Serial to computer 2. Restart the ATMega328 to begin

	polling 3. Use the ArduinoIDE and timestamps to count the number of average responses a second 4. Multiply that response time by 4, one for each analog-to-digital converter Presentation of Results: - The results will be presented as a chart of average timings Solution: - It slows down when we swap channels, so using 4 chips and 1 channel per would significantly decrease the latency. To further decrease latency, instead of using the polling protocol, an interrupt protocol can be used using the D_Ready pin on each chip.
3. Outputs from the microcontroller have an error rate of less than 1%	Equipment: ATMega328, AD7705, Load Cells Test Procedure: 1. Wire the ATMega328 and AD7705 according to schematic 2. Connect the circuit to the computer via USB 3. Let the ATMega328 run for 5 minutes with no weight on the load cells 4. Check that 99%+ of the outputs fluctuates between a range of +/- 0.05 mv from the baseline voltage 5. Apply a 10 kg weight to the attached load cell and record the new baseline voltage 6. Check that 99%+ of the outputs fluctuates between a range of +/- 0.05 mv from the baseline voltage Presentation of Results: - The results will be presented as an average voltage with its standard deviation Provisional Result: - Initially failed because of ceramic oscillators on the ATMega328.

	Succeeded when swapped to crystal oscillator

Table ii.3: Software Requirements

Requirements	**Verifications**
The ATMega328 polls the analog to digital converters correctly and passes that data along to the PC.	Equipment:PC Test Procedure: 1. Connect USB-to-Serial to PC 2. Press down on foot pad 3. Use ArduinoIDE to find voltage values of all inputs 4. Record data 5. Press down again with a different level of force and record data 6. Repeat for each pad Presentation of Results: Graph that show each sensor being pressed with varying levels of force. Each line represents voltage. X represents time. Y represents voltage level.
The software properly sends an input signal (key press) to the PC once a load cell has exceeded a set threshold (a.k.a the user steps on the pad)	Equipment: PC Test Procedure: 1. Plug in USB-to-Serial converter 2. Restart ATMega328 3. Open word processor 4. Step on foot pad 5. Record if a key was output to word processor 6. Repeat for each foot pad Presentation of Results: - Table comparing input values to output values and expected output values

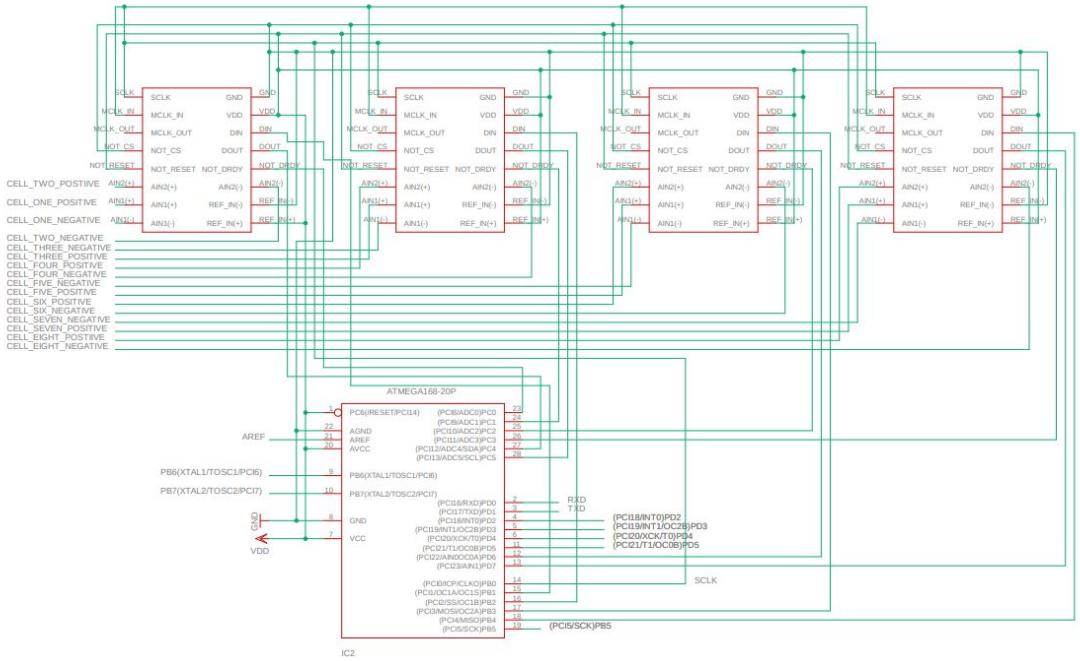


Figure ii.2: First Schematic Design

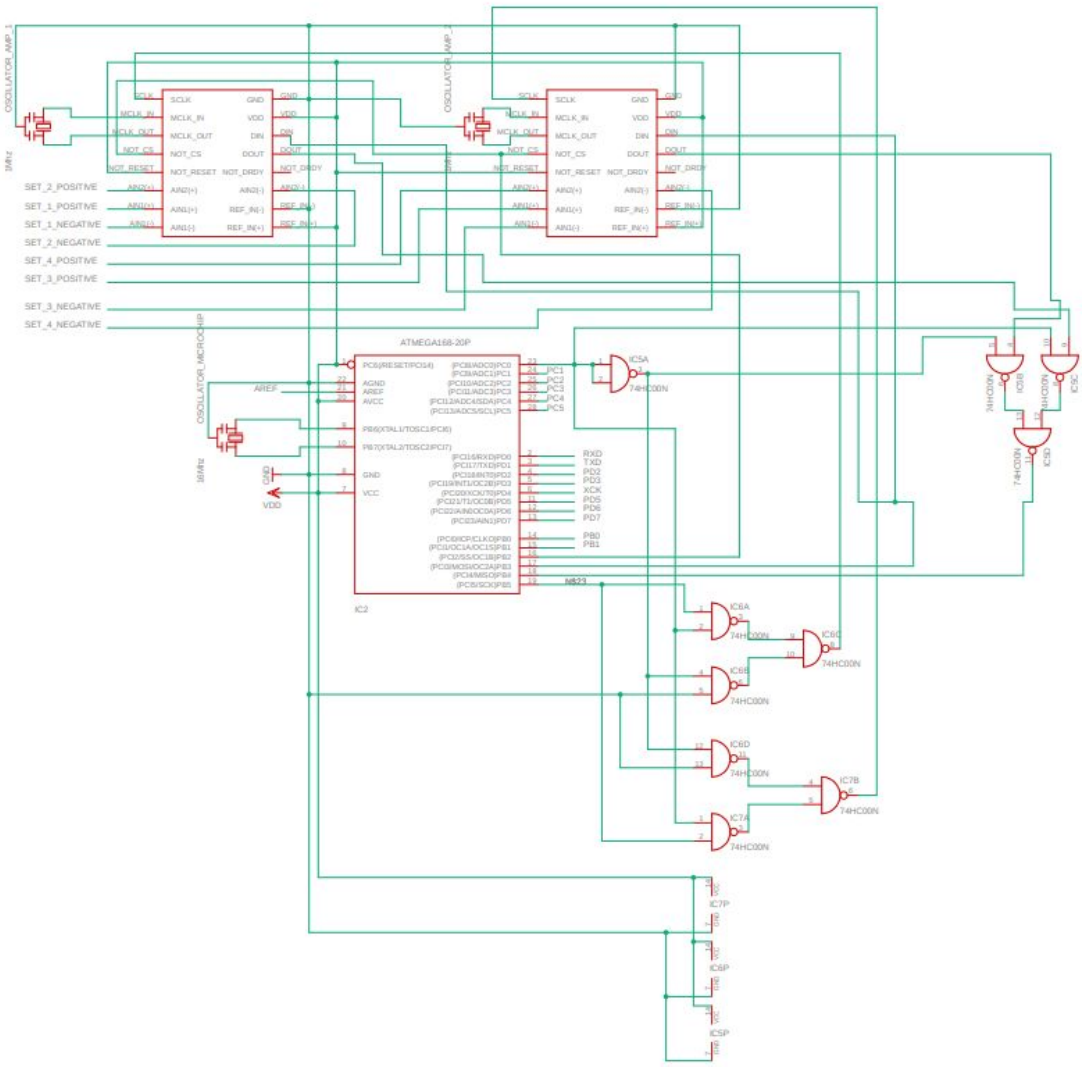


Figure ii.2: Second schematic design