Infinity Control Gauntlet

Ashish Pabba (apabba2), Chris Schodde (schodde3) and Ramakrishna Kanungo (kanungo3)
Team 25
TA: Yifan Chen

Table of Contents

Introduction

Objective

For certain applications, the issues with conventional input devices such as the mouse and keyboard are evident. There are numerous such fields that lack a sufficiently intuitive and natural input method, including but not limited to VR technology, video games, CAD/3D modelling, and even basic menu navigation. For example, in the context of 3D modeling and CAD applications, rotation, zooming, and translation are extremely inconvenient actions. Such a lack of intuitiveness can seriously hamper workflows and negatively affect efficiency and accuracy. Therefore, an advancement or breakthrough in this area would be analogous to the invention of the mouse as a point and click device.

Moreover, another source of inconvenience is that most input devices are external and secondary instruments as opposed to extensions of the body, the latter being overwhelmingly preferable. This is evident in the complete replacement of stylus-based touch screens by capacitive touch screens that can be used with fingers.

With these aspects of the problem space in mind, our proposed solution is to engineer a wearable glove that can be used to recognize the hand-based gestures of the wearer to serve as an input device. Such a solution would enable the user to translate specific and precise gestures to a directive/input command. As previously discussed, the applications for this method of input would be significant. Therefore, our objective is to essentially develop a glove that can serve as an immersive and intuitive input instrument with applications in several fields.

Background

Our goal is to attempt to develop a more intuitive, immersive, and natural method of input in the form of a glove that is capable of recognizing gestures. While there are similar products in development from companies such as HTC and Sony, such products are either nascent, niche, or prohibitively expensive. In order to distinguish itself, our glove solution will have to be both reliable and inexpensive. Another possible argument against the pursuit of the development of such a glove is the alternative possibility of using computer vision to track hand/finger movements. However, this solution can be easily dismissed when the field-of-vision requirements, camera costs, and necessary computer vision hardware and software are contrasted with the simplicity of the glove.

High-level requirements list:

• Sensor data is transmitted to the microcontroller without any significant signal noise addition.

• Control loop on microcontroller should be able to detect a move right, move left, zoom in, zoom out, and thumbs-up gestures.

• False positive gestures i.e. recognition of a gesture when a gesture is not actually performed, should be ignored.

Design

Block Diagram

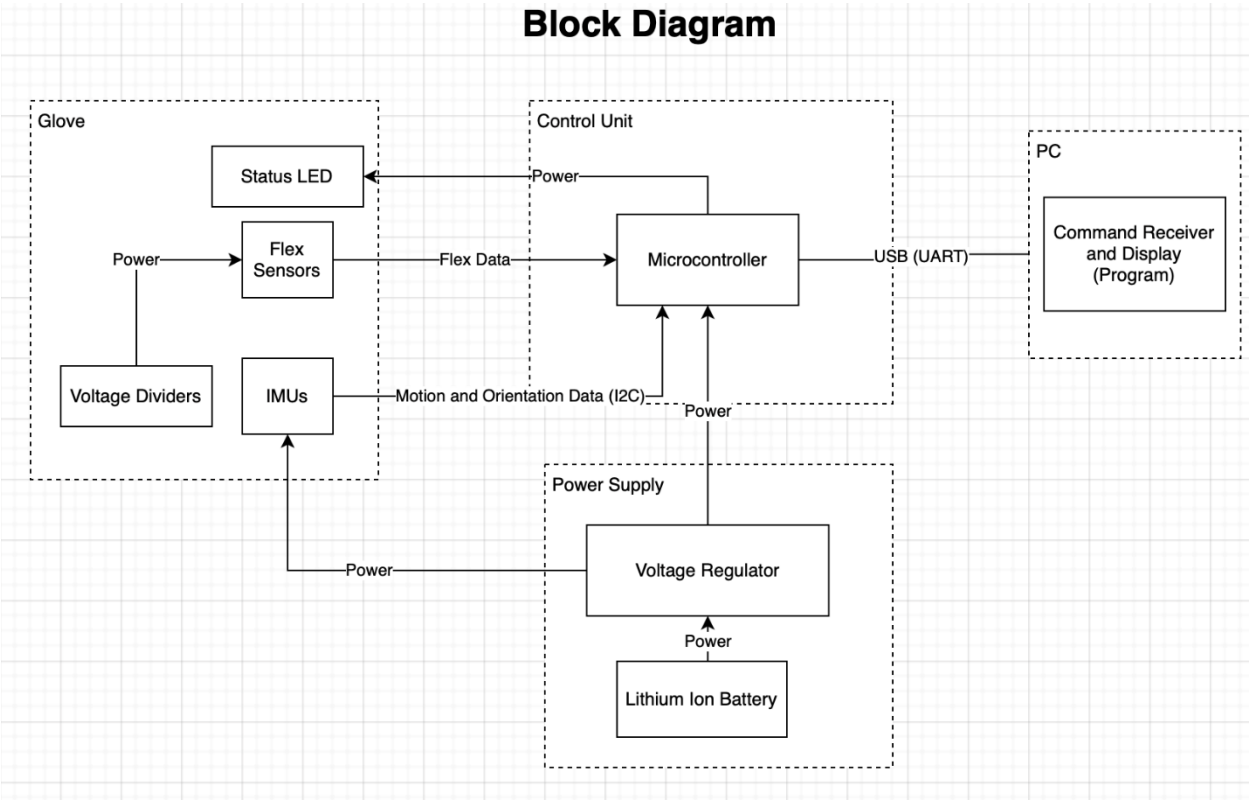

Figure 1: Block Diagram

Functional Overview

The ***glove subsystem*** encompasses the sensors mounted on the glove that are used to help detect gestures. And since a gesture is just a combination of different motions happening in parallel the aim of the glove subsystem is to provide the control unit, with the microcontroller, data from different types of sensor values that aim to detect a certain motion.

The ***control unit*** consists of the microcontroller and the analog to digital converter. The ADC is used to convert the voltage divider circuit value from the glove subsystem and send the digital output to the microcontroller. The microcontroller is used to interface with the various sensors and detect gestures, these include rotational motion, translational motion and hand contractions.

The ***power supply unit*** is used to provide power to the various sensors and the microcontroller.

The ***PC unit*** is used to interact with the microcontroller and report the gestures sensed.

Physical Design

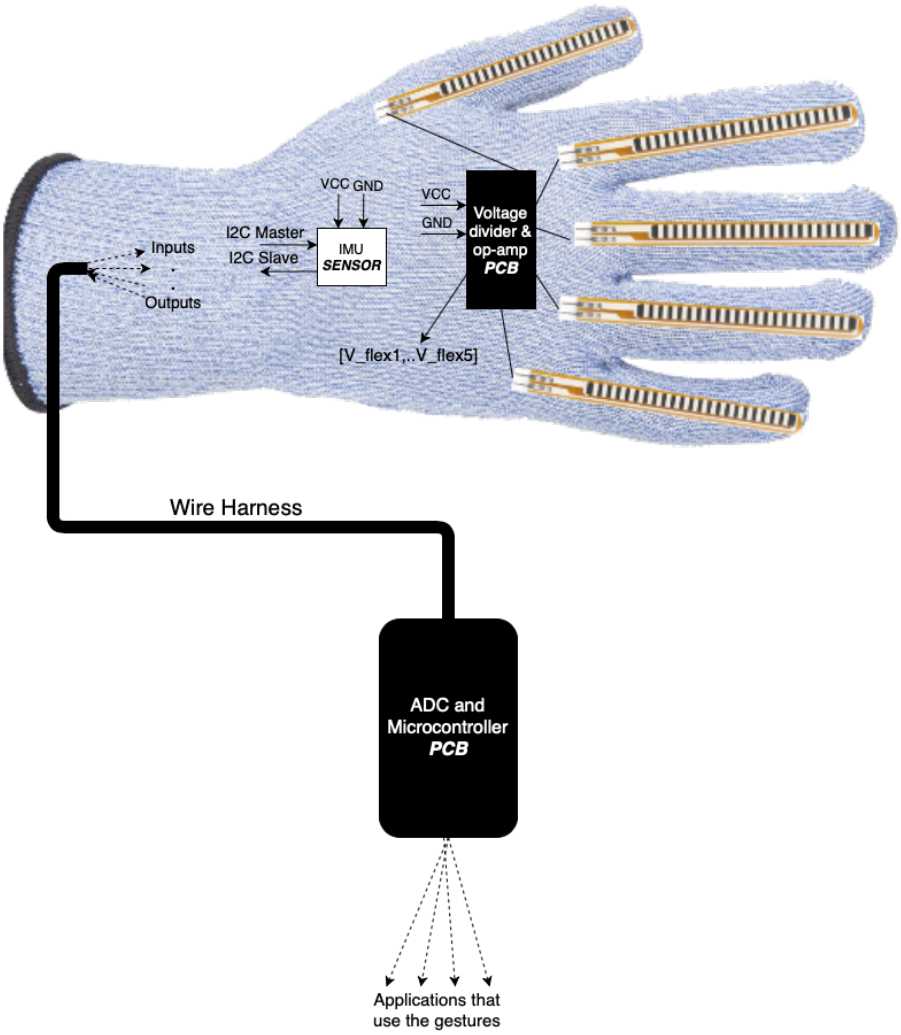

Figure 2: Physical Design

Glove Subsystem

Flex Sensors

The flex sensors [1] are placed to capture the contraction of each of the fingers. The flex sensors are connected to another set of resistors in series to form voltage divider circuits. The voltage drop is fed to a voltage follower op amp [2] circuit to avoid source impedance. Then this is routed to the control unit. The routing to the control unit is done using 20 AWG wires that have a low resistance of 10.5Ω/1000 feet [3]. This is done to avoid noise gained during transmission.

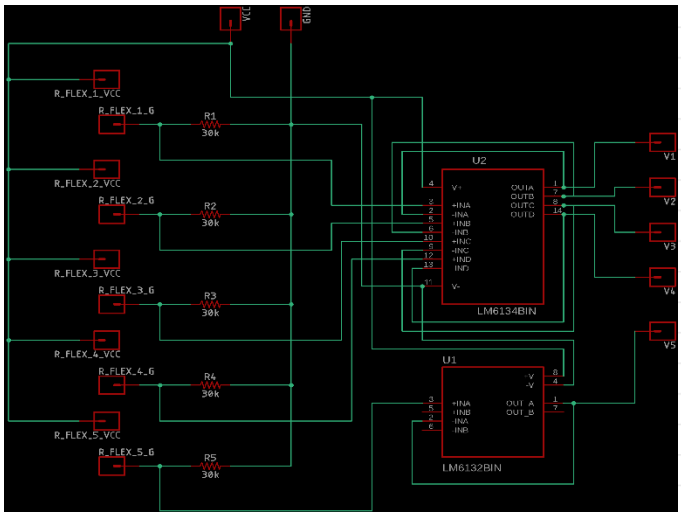

Figure 3: On hand PCB circuit

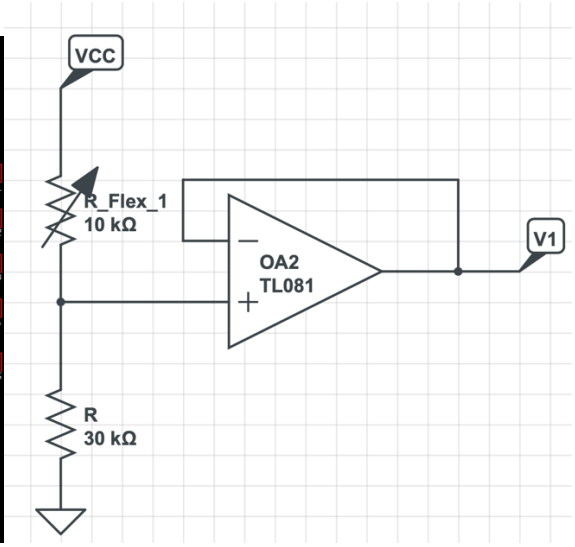

Figure 4:Individual flex sensor circuit

Requirement	Verification
- Resistance needs to lie between 7kΩ and 13 kΩ - Needs to output voltages between 2.5-4.5V based on finger orientation - Consumes ~2 mA across sensors and op-amps	• Measure resistance of each resistor • Create resistor divider circuit on a breadboard and measure output of flex sensors using voltmeter and power supply • Recreate unity gain buffer circuit on op amp and check if requirements are met • Vary length of 20 AWG wire and confirm low resistance

Table 1: Requirements & Verification: Flex sensors

IMUs

The BNO055, our chosen breakout-board-mounted IMU, is used to detect rotational and translational motion across nine degrees of freedom. It does so using a gyroscope and an accelerometer. Moreover, it can be configured to interface with the control unit's microprocessor using the I^2C standard.

More specifically, the BNO055 can provide the following relevant sensor data:

- Orientation across 3 axes
- Angular velocity across 3 axes
- Linear acceleration across 3 axes

Given this sensor data profile, we are confident that we will be able to accurately recognize our chosen set of gestures.

Requirement	Verification
- Stationary and level IMU should output zeros across output as expected. - Needs to be calibrated to $\pm 2g$ range (or $\pm 4g$ based on actual values obtained) for precision - Must consumes manufacturer specified amount of ~12.3 mA - Must communicate orientation and motion data reliably over I^2C	• Verify zero output while IMU is level and stationary • Measure operating current draw of sensor • Monitor output to microprocessor while in different positions and ranges of motion

Table 2: Requirements & Verification: BNO055 IMU

LED

The LED is a component that is added to see which mode the microprocessor is in. This will help determine gestures and is elaborated on in the next section. It is connected to the control unit using a 20 AWG wire again to increase efficiency.

Control Unit

The microcontroller chosen for this project is the 28-pin dsPIC33CH128MP508 [4]. It supports I^2C interface to collect data from the IMU and internally contains 4 ADCs, with more than the 5 conversion channels required in this project. The ADC converts the analog voltage values from the flex sensor into digital ones. Since the output expected from the flex sensor is primarily binary, the digital signal doesn't need to have a high resolution. Although since the values of the voltage divider circuit vary from 2.5 to 4.5V, the 12-bit resolution provided is more than necessary. A control loop needs to be able to constantly analyze sensor data and output which gesture is being performed. Details about the control loop are found below. Further, the clock speed needs to be kept in mind since a relatively fast clock is required to do computationally heavy tasks to detect gestures while missing minimal sensor information between two computation cycles. The microcontroller supports a 200Mhz crystal that will enable it to meet this requirement. Hence the primary requirements for the PCB in this unit are the microcontroller, a clock generator and a push button to calibrate the sensors (elaborated below).

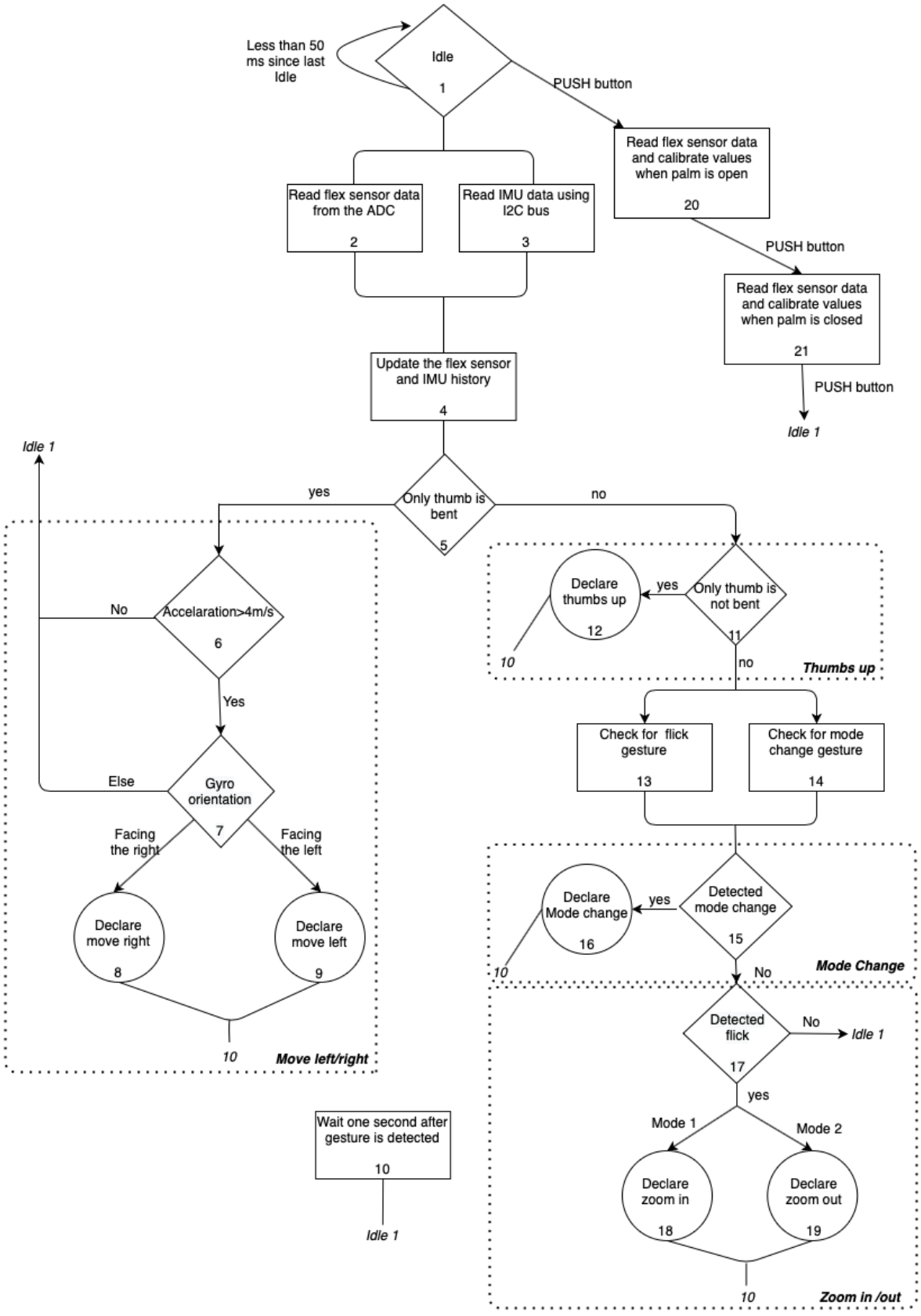

Figure 5: Control Loop flowchart in Microcontroller

The diagram above shows the software flowchart for the control loop on the microcontroller. The idle state (state 1) is where the control reaches after every event. The idle state has a condition where it checks whether it has been 50ms since the previous idle state. That ensures a total of 20 loops over 1 second. This is more than enough data to detect gestures.

A push button on the PCB with the microcontroller is used to move into the calibration state (state 20). The flex sensors have different values in resistance and hence need to be calibrated before use. When the button is pushed the user is expected to keep their fingers spread out and palm flat. The voltage value is noted. Then the push button is again clicked, and a state is entered where the flexed voltage sensor values are noted by making the user make a fist. Then another push on the button leads us back to the idle state where gesture detection begins again.

After waiting in idle the control loop moves into both the read for flex sensors and read from IMU state. This is possible since the microcontroller chosen has multiple cores. The data is read from these sensors in parallel. Further, due to the presence of multiple ADC's on the microcontroller, state 2 will have some internal parallelism and read values from multiple ADCs at once. This parallelism helps with more synchronized values from the sensors. This data is then loaded into a history table where the last 40 values for each sensor are stored. With a 16kB RAM in the microcontroller, this is easily accomplishable.

Move left & move right: These two gestures have common features and are seen towards the bottom right of the control flowchart. If the palm of the person wearing the device is facing the left, which means the IMU is facing the right, and there is some translational acceleration detected towards the left, then we detect move to the right. Vice versa for move to the left. Also, the persons thumb needs to be kept bent at all times for this gesture to be detected. The bent thumb is detected using the flex sensor.

Thumbs up: This is the simplest of the gestures. If the thumb is not bent and all other fingers are bent a thumbs up is detected.

State 13 and 14 are computationally heavy states and are done in parallel. State 13 checks for a flick that is used in zoom in and zoom out. And state 14 checks for mode change.

Mode change: The flick gesture described in the next paragraph is used to zoom in or zoom out. But zoom in or out is declared based on a mode variable. The mode variable is toggled between one and two if the mode gesture is detected, ie, if mode was one it is set to two when this gesture is detected. The gesture is detected if a fist is made quickly and released. If all the fingers are spread out and ones palm is flat, and then in quick succession a fist is made followed by spread out fingers and flat palm. This gesture is detected by processing the data in the history table which stores the data of all the sensors. The current mode is also displayed by lighting up the LED on the glove in mode 1 and not lighting it in mode 2.

Zoom in/ Zoom out: The flick gesture is used to declare a zoom in our out. A flick is detected if all the fingers are bent like a fist and then the pointer finger and thumb suddenly extend. It is similar to the action when a carrom striker is hit with the index finger while playing carrom. This is detected again by looking at the history data of all the sensors. Once a flick is detected, a zoom in or a zoom out is declared based on the current mode.

After a gesture is detected, the control loop pauses for a second so as to avoid constant detection of the same gesture multiple times. After every gesture control is passed to state 10, where we wait for one second before proceeding to idle again.

Requirement	Verification
- Microcontroller draws a maximum of 24mA at 3V - Interface with the IMU using I^2C protocol and store this data - Interface with flex sensor unit, convert data using ADC and store the data. - Processing is fast enough to allow for at least 20 loops per second. - Able to detect the gestures reliably with minimal false positives.	• Prototype the microcontroller on the bread board with the push button and the clock generator. • Program microcontroller to do Dijkstra algorithm, which is computationally intensive and check current draw on breadboard. • On a breadboard, verify the I^2C interface. • Write functions for various gestures and verify them individually. • Then construct the control loop to detect all the gestures

Table 3:Requirements & Verification: Control Unit

Power Unit

This consists of a 3V battery and a step-up linear voltage regulator. The step-up regulator powers the resistors in the flex sensor system, which requires 5V. The battery can directly power the microcontroller and the IMU at 3V with some safety measures incorporated to prevent overcurrent damages.

Requirement	Verification
- Output of 3V for microprocessor and IMU (current draw of 24 mA) - Output of 5V for flex sensor system (current draw of 2 mA) - Sufficient capacity for a reasonable duration of operation.	• Probe step up output voltage and confirm 5V output • Probe battery output and confirm 3V output • Discharge power supply unit at 3V and 5V (through step up linear regulator) with appropriately sized resistors to observe discharge characteristics and duration.

Table 4: Requirements & Verification: Power Unit

Tolerance Analysis:

Having flex sensors that produce output voltages within our specified range for input into the ADC is critical to the correct detection of finger position. The value of the resistor used in conjunction with the flex sensor to produce a voltage divider must be such that the input voltage into the ADC neither falls below 2.5 V nor exceeds 4.5 V. The flex sensor is listed as 10 kΩ, with a tolerance of ±30%, resulting in the range of values 7 kΩ to 13 kΩ. At 180° flexure, the flex sensor produces a resistance not less than twice that of the flat position. To ensure that there is no overlap between the upper bound of the flat resistance and the lower bound of the flex resistance, we seek a position that results in the greatest difference between the two ranges. With the flex sensors mounted to the Distal bone of the hand, the 180° position can be reached with full closure of the fingers. With $Vdd = 5\,V$, we seek the upper and lower bound on the resistor R from figure 3. Solving

$$\frac{R}{R + R_{flat,min}} V_{in} \leq 4.5V$$

We find that our upper bound is $R \leq 63000\ \Omega$.

Our lower bound is found through solving

$$\frac{R}{R+R_{flex,max}} V_{in} \geq 2.5V$$

We find that our lower bound is $R \geq 26000\ \Omega$

Using a worst-case tolerance of $\pm 20\ \%$ on the resistor value to search for the cheapest part on Digikey, the bounds for R become $32.5\ k\Omega \leq R \leq 52.5\ k\Omega$. A $47\ k\Omega$ resistor was then chosen. Using this value, we can find the smallest gap between a flat and a flexed hand, to determine if the resolution of our ADC will be enough to distinguish between a flat and flexed hand. The minimum values for the flat and flexed resistance of our flex sensor resistor give the smallest gap in the output voltage for the flat and flexed position. Knowing that,

$$\frac{R}{R+R_{flat,min}} V_{in} = V_{out,flat,min} = 4.352V \text{ and } \frac{R}{R+R_{flex,min}} V_{in} = V_{out,flex,min} = 3.852V$$

The difference between a flat and flexed hand in the worst-case scenario is 0.5 V. The resolution of the ADC using the 8-bit mode is 0.02 V, well below the difference in the worst-case scenario.

COVID-19 Contingency Planning:

Assuming all parts are soldered before UIUC transitions to online classes, then the project is all but completed, and no major changes must be made. However, if the transition occurs before this, lab equipment sufficient to solder components and test the device after completion must be acquired. The most critical part, if an unexpected transition to remote learning occurs, is the PCB on the hand since this has to be soldered. The microcontroller can be put onto a breadboard using a breakout board along with the other control unit components. After this, the project is primarily software driven and can be easily completed at home. Advanced notice from the university is expected, such that either the equipment will arrive in time before the deadline, or the transition will occur after completion of the device.

Cost and Schedule

Cost Analysis:

Description	Part Number	Quantity	Cost (USD)
Flex Sensor	FS-L-095-103-ST	2	21
Resistor 47 kΩ	CF14JT47K0	2	0.1
Microcontroller	dsPIC33CH64MP502	1	3.95
IMU Breakout	BNO055	1	34.95
Op Amp	LM741CN/NOPB	2	0.88
Battery Holder	Rechargeable 3V battery	1	4.79
Gloves	Cevapro Running Gloves	1	14.99

Table 4: Cost breakdown

The total cost of the parts as listed in the parts table is $102.62. From UIUC's own data, the average salary of EE and CE majors is $88,000 per year [5]. Assuming 52 work weeks, 5 days a week, 8 hours a day, we arrive at an hourly rate of $42.31. Average partner contribution can be assumed at 10 hours per week. From the given formula for the cost of labor, we find that labor will cost 2.5 X 30 hours/week X 7 weeks X 42.31 $/hour = $22,212.75.

The **total cost** of both labor and parts is then $102.62 + $22,212.75 = **$22,315.37**

Schedule:

Week	Ashish	Chris	Ramakrishna
10/5	Decide on and order parts for power unit, begin to test parts	Decide on and order parts for control unit, begin to test parts	Decide on and order parts for glove, begin to tests parts
10/12	Start working on microcontroller programming based on sensor tests, help with PCB design	Work on PCB Eagle Schematic, submit PCB order	Work on PCB Eagle Schematic, submit PCB order
10/19	Program microcontroller and test motion and flex recognition	Program microcontroller and test motion and flex recognition	Program microcontroller and test motion and flex recognition
10/26	Test power units, explore computer interface and how to display recognized gesture	Test control unit in conjunction with glove, verify microcontroller program and modify as necessary.	Test glove unit, log sensor data for gestures and identify usable trends/patterns
11/2	Complete assembly, test gesture recognition with integrated prototype and modify microcontroller program as necessary	Complete assembly, test gesture recognition with integrated prototype and modify microcontroller program as necessary	Complete assembly, test gesture recognition with integrated prototype and modify microcontroller program as necessary
11/9	Prepare for mock/final demos, grace time in case of delays in earlier steps	Prepare for mock/final demos, grace time in case of delays in earlier steps	Prepare for mock/final demos, grace time in case of delays in earlier steps

Table 5: Schedule

Discussion of Ethics and Safety:

The weight and positioning of the sensors and boards may over time, if care is not taken, cause strain on the operator's hand or abrasions on the skin, potentially resulting in permanent injury. To comply with rule #1 of the IEEE Code of Ethics [6], that we may preserve the health and safety of our users, attention will be given to ergonomic placement of the said components on the hand and wrist, to achieve a balanced, comfortable experience.

Additionally, because the user is intended to wear a mounted electrical system, care will be taken to ensure proper insulation of the mounted electrical components and their associated wires, so that no part of the operator physically encounters these components.

For the battery, there must also be a consideration regarding the temperature, as there exists the possibility of overheating. However, since the battery is not mounted on the glove in our design, the risk of a burn is rather low.

Finally, the design of this system is open source. This is good ethical practice because it allows for adoption and modification of the system without additional expense by the community that utilizes the design.

Works Cited

[1] Spectrasymbol, "Sparkfun," [Online]. Available: https://cdn.sparkfun.com/datasheets/Sensors/ForceFlex/FLEXSENSORREVA1.pdf.

[2] T. Instruments, "Texas Instruments," [Online]. Available: https://www.ti.com/lit/ds/symlink/lm6134.pdf?ts=1601536549483&ref_url=https%253A%252F%252Fwww.google.com%252F.

[3] Hyperphysics, "Hyperphysics," [Online]. Available: http://hyperphysics.phy-astr.gsu.edu/hbase/Tables/wirega.html.

[4] MicroChip, "MicroChip," [Online]. Available: https://ww1.microchip.com/downloads/en/DeviceDoc/dsPIC33CH128MP508-Family-Data-Sheet-DS70005319D.pdf.

[5] UIUC, "courses.engr.illinois.edu," 2020. [Online]. Available: https://ece.illinois.edu/admissions/why-ece/salary-averages.

[6] IEEE, "ieee.org," 2020. [Online]. Available: http://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed 16 Sep 2020].

[7] Sparkfun. [Online]. Available: https://cdn.sparkfun.com/datasheets/Sensors/ForceFlex/FLEXSENSORREVA1.pdf.