Gest Final Report Postmortem

Joshua Feist	JoshuaFeist2023@u.northwestern.edu
Isabelle Johnson	IsabelleJohnson2023@u.northwestern.edu
Samin Kumra	SaminKumra2022@u.northwestern.edu
Haroon Nawaz	HaroonNawaz2022@u.northwestern.edu
Danche Smilkova	DancheSmilkova2023@u.northwestern.edu
Chris Uustal	ChristopherUustal2023@u.northwestern.edu

Northwestern University | CE347-2 | Spring ‘22

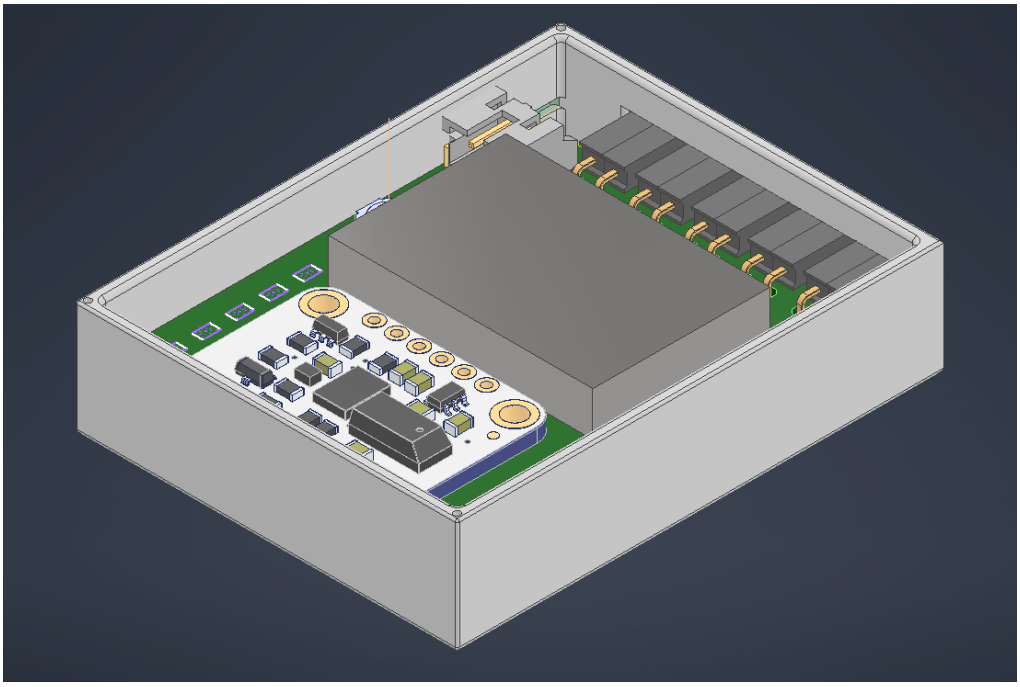
Design Overview

The finalized design of Gest is essentially the same design that was called out in the original design report. That is, it has the Wemos D1 Mini breakout board for the ESP32 at its core, and it utilizes an array of flex sensors and an absolute orientation sensor mounted at the base of the hand to measure linear acceleration as well as finger flexure. In particular, we used the same flex sensors as we originally used in our testing, the Adafruit long and short flex sensors, as well as the same Adafruit BNO055 absolute orientation sensor. Combined with a 400mAh battery we found on Amazon and some PCB components, these elements were all integrated together onto a custom PCB in a 3D printed enclosure we designed. Below is an image of the full Gest:

As is clearly the elephant in the room and what I think is our most impressive accomplishment is that we actually made a full set! We found very early on in the process that achieving the basic deliverables we originally set out to make would be 1) very and easy to achieve and 2) didn't give us the functionality we wanted to actually recommend using the device. How did we come to this conclusion, you may be asking? Well, using the mouse *sucked*. After a significant amount of effort was put into trying to smooth and process the data we were getting from the absolute orientation sensor, we came to realize that we weren't going to get a mouse that anyone was actually going to want to use. There was a similar experience with the zooming functionality we originally called out. While we did get those things working in our initial testing, we couldn't get them to be good enough that anyone would actually *want* to use them, so instead we pivoted and focused on making an experience you'd actually want rather than meeting our initial deliverable, and I'm really glad we did!

While you, the reader, will never be able to try our Gest in action, the members of our team have come to an agreement after using the Gest that it's actually *not terrible* and *can actually be really fun in certain applications*. In particular, *gaming*. As you can see from our hype video here, we quickly came to realize that (once we had access to the second hand), productivity was probably not the best use case for a device where using the mouse on the desktop sucks, so instead we turned it into a wireless gaming controller and used it to play Minecraft. While there's definitely a learning curve to getting used to it, none of us found it any worse than switching from a keyboard to a gaming controller or vice versa. If we had known this was the final direction we were going to end up going in from the beginning, we probably could have tuned the interface even more to make it perform really well for certain games.

Going into the hardware just a little bit, the custom PCB ended up being a really good choice for packaging. Since we ended up integrating a rechargeable battery, sensors, a microcontroller, a user interface button, feedback LEDs, and more all into our little enclosure with a height less than 15mm, the custom PCB was essential. The full design files can be found in the Hardware folder on our GitHub. From a high level, all the designs were done in Eagle, and the PCBs were manufactured by JLCPCB. We then ordered the components from a mixture of Mouser and DigiKey and assembled them in house. We ended up going through two board revisions because we had some issues with some of the original pins we selected on our microcontroller for some of the flex sensors. To meet the thickness requirements, we had to do a full CAD of the internals of the Gest to make sure it was possible to package it all, and we even had to order a custom ultra-thin PCB (0.6mm vs 1.6mm). You can see the CAD of the full design below:



An element of designing a wearable device that goes under-appreciated really is the thickness. The user generally doesn't care how large a device is as long as it doesn't protrude off their body by a lot, and every few mm counts here, so meeting that maximum thickness spec of 15mm called out in the original design report ended up being a difficult challenge to overcome, but the final results were definitely worth the effort. Due to stocking issues on the BNO055 chip, we were not able to order any more than the 2 we originally ordered for prototyping. As such, we chose to only install one on the main-hand which could be used for mouse movement and leave the second one for continued prototyping.

I'm not going to touch on the code too much here, but you can see the full code base on our GitHub here. Since we ended up going with two hands, we ended up having two very similar code bases--one for the main hand (right) and one for the off-hand (left). While these code bases shared a lot of similarities in the end, the main differences were that he off-hand didn't have any of the mouse movement code and was configured predominantly as just a keyboard input while the main-hand was mostly configured to act as a mouse. While we would have liked to make it support both modes, we ended up spending a significant amount of time attempting to design a fully custom BLE protocol for making them communicate to each other and show up to your device as a single peripheral. Sadly, we weren't able to get that working in time for the demonstration, so you currently have to pair both Gests to your device separately. One thing that ended up being a great addition we forgot to consider initially was calibration--since the gloves fit differently on everyone, that also meant the maximum flex reading from everyone ended up being different. We resolved this by taking a maximum flex reading on startup and again if the user chose to press the recalibration button and using that to define our thresholds for when the

user is clicking or not clicking a sensor. While a more advanced approach reading these sensors using an on-board AI, machine learning, or a deep neural network may have better results, this was simple and effective enough that we found it met our needs well.

While the predominant user experience exceeded our expectations, there are a few small things that we probably need to iron out before we'd be comfortable really bringing it to the mainstream. Outside of the physical assembly of the device (which obviously could use some care, but that's not really our field), the device also currently doesn't have any way of turning off. That should be a relatively trivial change, but we had some issues with it hanging and not re-pairing to a device when it turned back on, so we left that feature out. This leads to the somewhat annoying result of it just constantly being on then dying while it sets in your backpack after a few hours. While annoying, that's something we think can be fixed in the future. We already mentioned the small annoyance of having to pair them to your device separately, and this also led to some weird issues where the off-hand would randomly disconnect from our testing laptop far more often than the main-hand, but this could have been from a variety of factors that we weren't able to isolate. Recharging the device was really easy though since you just plug it in over microUSB and it just figures it all out--it even has an LED to indicate to the user that the battery is charging.

Changes

As mentioned, there ended up being a number of features we originally planned on implementing, while we did get them working, we just couldn't get them working well enough to actually make a product the user would want to use. Instead, we dynamically pivoted to continue focusing on making a product the user would actually want to use and dropped some of our initially called out functions. In particular, we originally said we were going to implement 3 functions: clicking, mouse movement, and zooming. Since we figured out that, while the mouse movement was functional, it wasn't particularly enjoyable to use on a desktop environment, we pivoted to mainly focusing on things like typing and gaming. As such, the focus shifted away from the zooming feature, since that isn't use much at all in those two spheres. Additionally, while we did get a zooming feature working, it fell into a similar camp as the mouse movement where, while it worked technically, we couldn't get it to a state where it was enjoyable to use. As such, we pivoted instead to add in the features of a full second hand and typing. If we had more experience with user interface design and had a more cohesive design team, we likely could have produced a product which included the zooming feature as a secondary option the user could switch to, but we changed the way our user interacted with the device so many times due to an inexperience in this space that it did not make it into the final product.

Another topic that went completely differently than expected was timelines. We essentially finished all of our initial deliverables weeks before anticipated (which was really good!), but we

quickly came to the realization we mentioned before that the features we were actually targeting just *weren't great*. Since we were lucky enough to be very ahead on our deliverables, we made the dynamic decision around week 6 to push forward with support of typing, gaming mode, and full two hand support. While this was not reflected in the initial timeline and we have no way of knowing that this would have been the direction we chose to go in, it is something that (had we somehow known to go in this direction) could have allowed us to produce an even better deliverable due to the increased development time which could have been directed towards the user experience.

Since we pivoted our design from a single hand to a two-hand design, our original cost estimates of approximately $115 for the full prototype obviously became invalid since we now required significantly more components. The total cost, including things just the custom PCB and its components as well as all the flex sensors, absolute orientation sensors, and microcontrollers ended up costing somewhere in the range of $300 total, but most of this was funded by the team directly out of pocket. While this significantly exceeds the original cost estimate, the majority of this was due to the cost of flex sensors ($13/each) and the new design now moving from only requiring 3 to requiring 10. We also had a fun *learning process* of how to accidentally break flex sensors, so we ended up having to replace a few of them before we understood how to properly reinforce them so they didn't keep breaking on our prototypes. Also, since we pivoted to using an off-the-shelf ESP32 solution instead of integrating the module directly into the board, that added unexpected costs as those are much more expensive than the module itself. Including other factors such as a second revision to the board, more PCB components than expected, and the cost of shipping, the significantly higher cost of the project becomes clear. While these are elements which are easy to highlight in hindsight as oversights from the initial scope of the budget, they are also the costs that come with prototyping and learnings, so it may be recommended in the future to just add a 2x factor on all prototyping costs to account for miscellaneous issues which may arise with the components which were originally selected.

Closing Thoughts

I think we're all really happy with the deliverables we were able to produce. While Gest definitely doesn't look exactly how we originally envisioned it looking or do all the things we originally envisioned it doing, it's actually fun to use and something that (if we have the time) some of us are interested in developing a little more in the future. We've already discussed ways that the flex sensors could be more effectively integrated into the hardware of the glove directly, the connection point to the processing unit could be much more robust, and the processing unit itself could be a lot smaller if we pivoted to a full custom PCB with no off-the-shelf development hardware (pending those items coming back in stock). We could get a lot closer to the size of a classic fitness tracker rather than the still somewhat large tile that we currently have, and that, combined with some minor user interface improvements, could make it a genuinely really nice

device to use. We might even throw some machine learning onto it and see if we can get even more useful data out of those flex sensors. All in all, we learned a lot, had a great time, and are really proud of the thing we got to make. We're sad we weren't able to take it in person and do a live demo with it, because the looks on all of our faces when we put a full assembled Gest on for the first time and starting using it was honestly priceless.