# BICI July update

**Brackets**

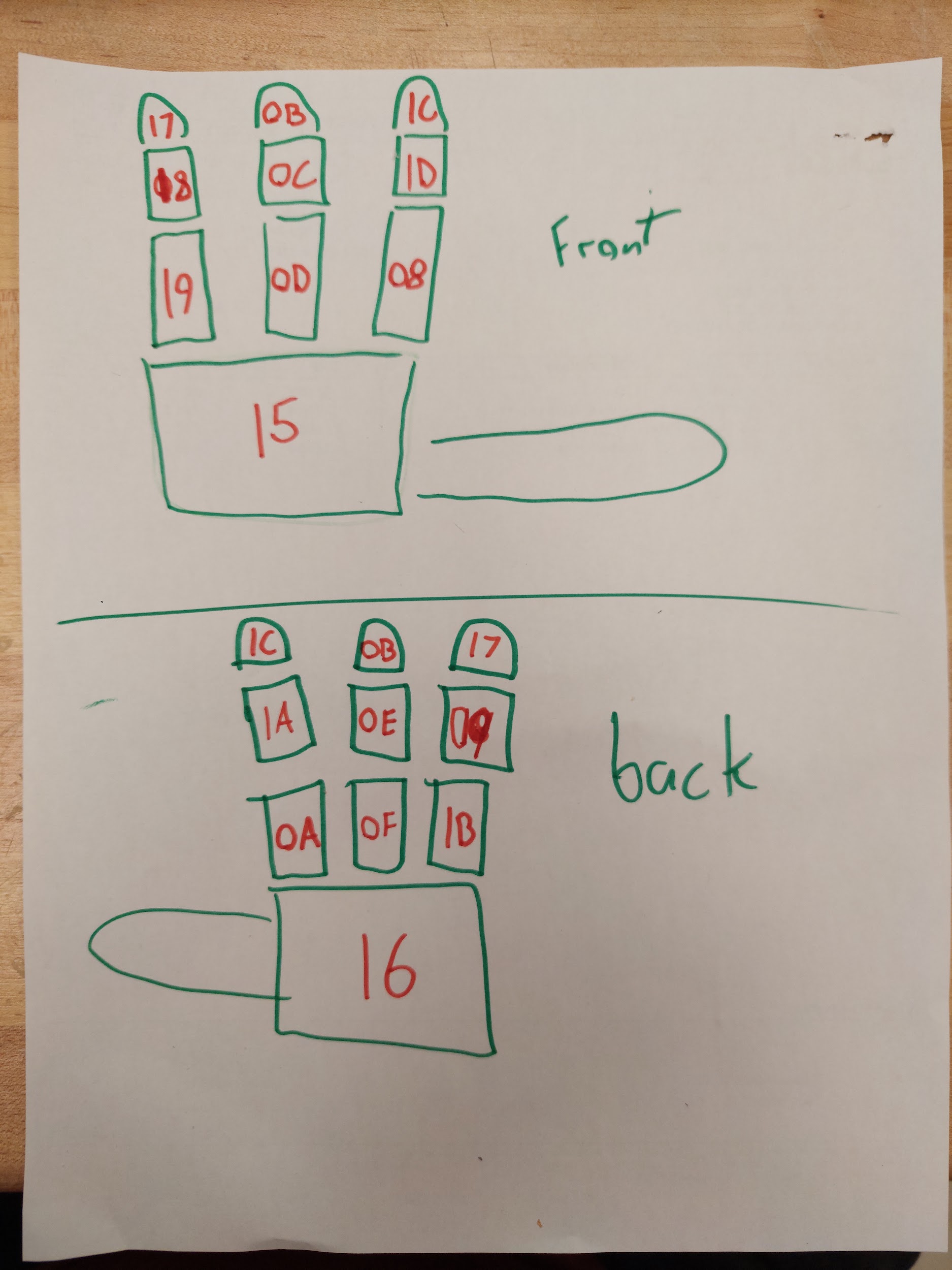
* The brackets were all printed using the filament printer instead of the resin printer. Unfortunately, this makes some parts a little more susceptible to breaking (especially the medial front/back bracket).
* The screws used for installing the brackets are M2. They can be found in two plastic boxes in the fabrication lab, on the mechanical hardware table.
* I figured too late that the brackets for the thumb did not fit properly on the Allegro hand. They fit OK, but then don't fit when the PCBs are added to the bracket. You can still maybe just install the fingertip of the thumb to get some data, but the other parts would need significant work to be used.

**Dielectrics**

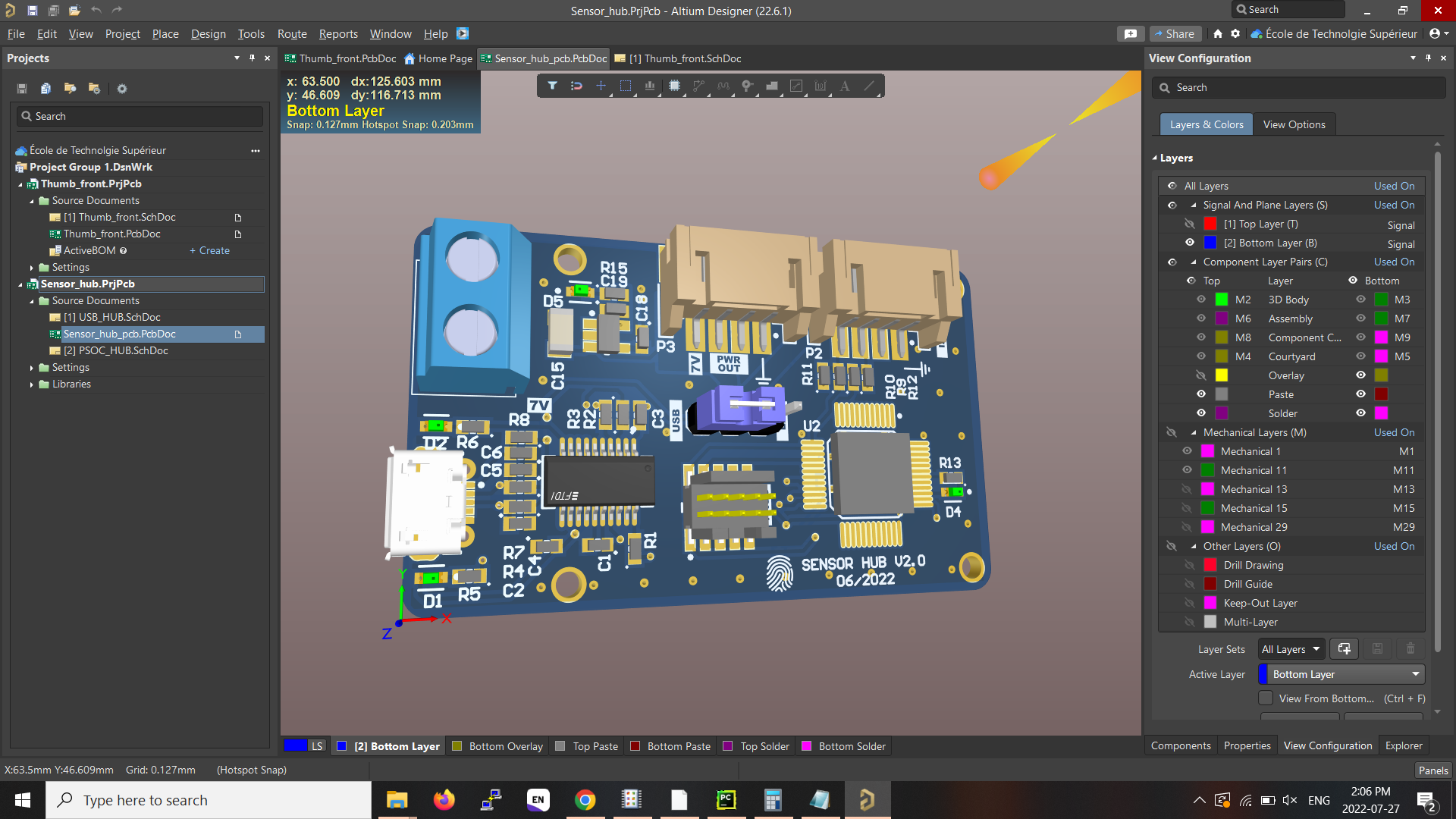
* Enough dielectrics were molded to be used on 2 fingers (and the thumb, but thumb brackets don’t fit..). Some of them are on the hardware table in the fab lab and some are still in the molds next to the vent. If you need help molding more, Berith was informed of the procedure. Until I come back, he can help you with that.
* Unfortunately, we still don’t have a proper way of encapsulating the dielectrics on the sensors. I would either tape them on or use silicone-based glue to bond the sides of the flex pcbs with the urethane.

**PCBs**

* A complete set of PCBs were made. All of them are already installed on the hand in their respective brackets (except the thumb…)
* If a PCB breaks, there is enough material to make a few (very few) more. See Altium 365 workspace for plan on how to assemble new one.
* I2C addresses have changed since the last document that was produced. See figure below to have the active addresses.



* The sensor hub PCB was modified to accommodate the new PCBs. There is now a jumper to choose the power source between USB power or external power. When using the hub with 1 to 6 sensors, USB power can be used. External power must be used beyond that number of sensors since power from the USB port will be dropping and funky behavior can happen from either the sensors or your USB port. Jumper switch is labeled on the PCB as well as the polarity of the external power port (see figure below). There is no inverted polarity protection, inverting the power cables will result in damaging the boards. This goes for cables that link all other sensors together, inverting flat cables will cause damage.



**PSoc programs**

* Code for the PSocs can found here: <https://github.com/etienne-roberge/BICI_Psoc_projects>
* Sensors programs were changed to have their own timer to get the time at which the data were acquired (instead of doing it on the Hub). This makes the time more accurate to the time the values were acquired.
* Sensor programs were changed to now have 2 I2C addresses. The first one (addresses showns in the image earlier in this documents) are used for communication between the Hub and the sensors. The second addresses are used for capsense tuner application in Psoc creator. This way, there is no more needs to reprogram the PSocs to test them in capsense tuner. The second address of all PCBs is their actual address+ 0x40. (For example, the BOH add#1 is 0x16, then the add#2 is 0x56).
* Bootloaders used to program the PCBs through I2C were removed from the sensors. This was done to have a consistent experience between sensors, since some of the programs were too big to fit in the flash of the PSocs.
* The proximity taxel was removed from all Psoc code, since it is unusable.
* I added a python script in the git folder that can be used to test the communication with the sensors. This was tested only on Windows, but should probably also work on Ubuntu. To use it properly on Windows, you need to modify the generic usb driver to have 1ms latency instead of 16ms. Refer to Jean-Philippe for help with that.
* Since code on the Sensor hub was modified, there is a good chance that the communication parser that was made in the ROS package might be outdated (for example: no more proximity taxel, means the message is 2 bytes shorter). The parser will need to be updated to work with the new generation. The current message structure is:

| StartByte | msgLen | sensorAddress | dataTime | VALUES | EndByte |
| --- | --- | --- | --- | --- | --- |
| **0x01** | 1 byte | 1 byte | 4 bytes | Nbtaxel \* [2bytes] | **\n** |