Design Document Check

1. Introduction

1.1. Contributors: Daniel Chen, Fangqi Han, Christian Held

1.2. Problem and Solution Overview:

A modern job-oriented citizen needs to have technology that lets them get their goals accomplished in a timely manner. In their office they have the space and comfort to use effectively a full size keyboard with all the amenities that allows them to work at maximum efficiency. Whilst traveling, they do not have such luxury. Additionally, left handed typers can find the numpad on keyboards to be awkward to use. The usual solution is to either buy additional, smaller keyboards for travels or to attach keyboard extensions through USB (Universal Serial Bus) at home. That means one will need either multiple keyboards or an undersized one. Furthermore, users of touchscreen devices may find the lack of numpad inconvenient while using a smaller keyboard.

Our solution to this problem is to make a keyboard with detachable modules that will allow users to conform to their working environment while still providing them with maximum utility. Users can simply add or remove modules to the keyboard by plugging or unplugging their TRRS(Tip/Ring/Ring/Sleeve) connectors. The keyboard will have adaptable firmware that allows the user to still have the keypresses available even when in a condensed mode.

1.3. Visual Aid

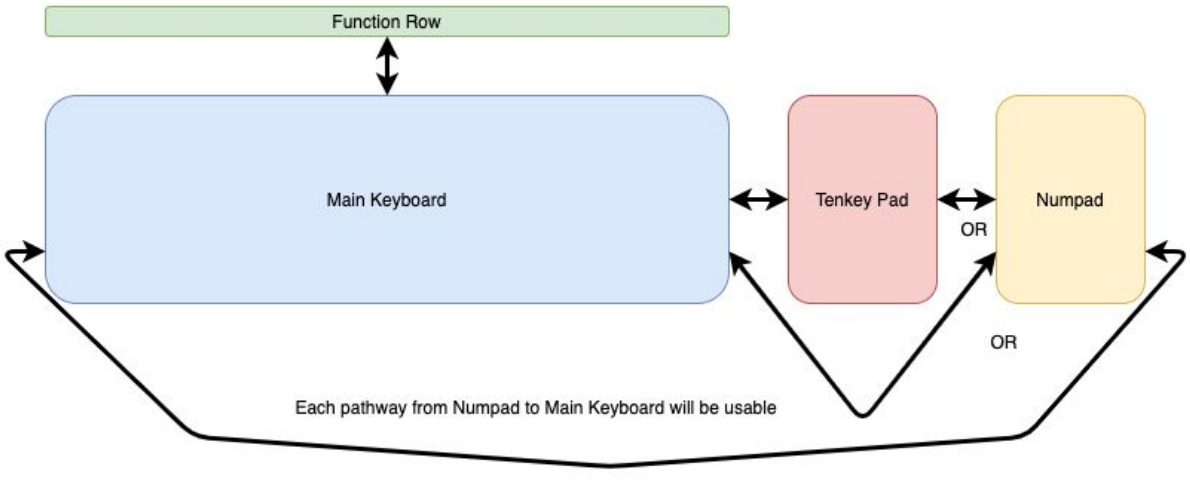

Figure 1. Visual Aid

1.4. High-level requirements list:

1.4.i. The Main Keyboard is functional with at least 61 usable keys.

1.4.ii. Auxiliary Modules are functional and able to plug into the main keyboard with at least 10 usable keys on each module.

1.4.iii. Firmware is workable and partially customizable with at least 20 changeable keys.

2. Design

2.1. Block Diagram:

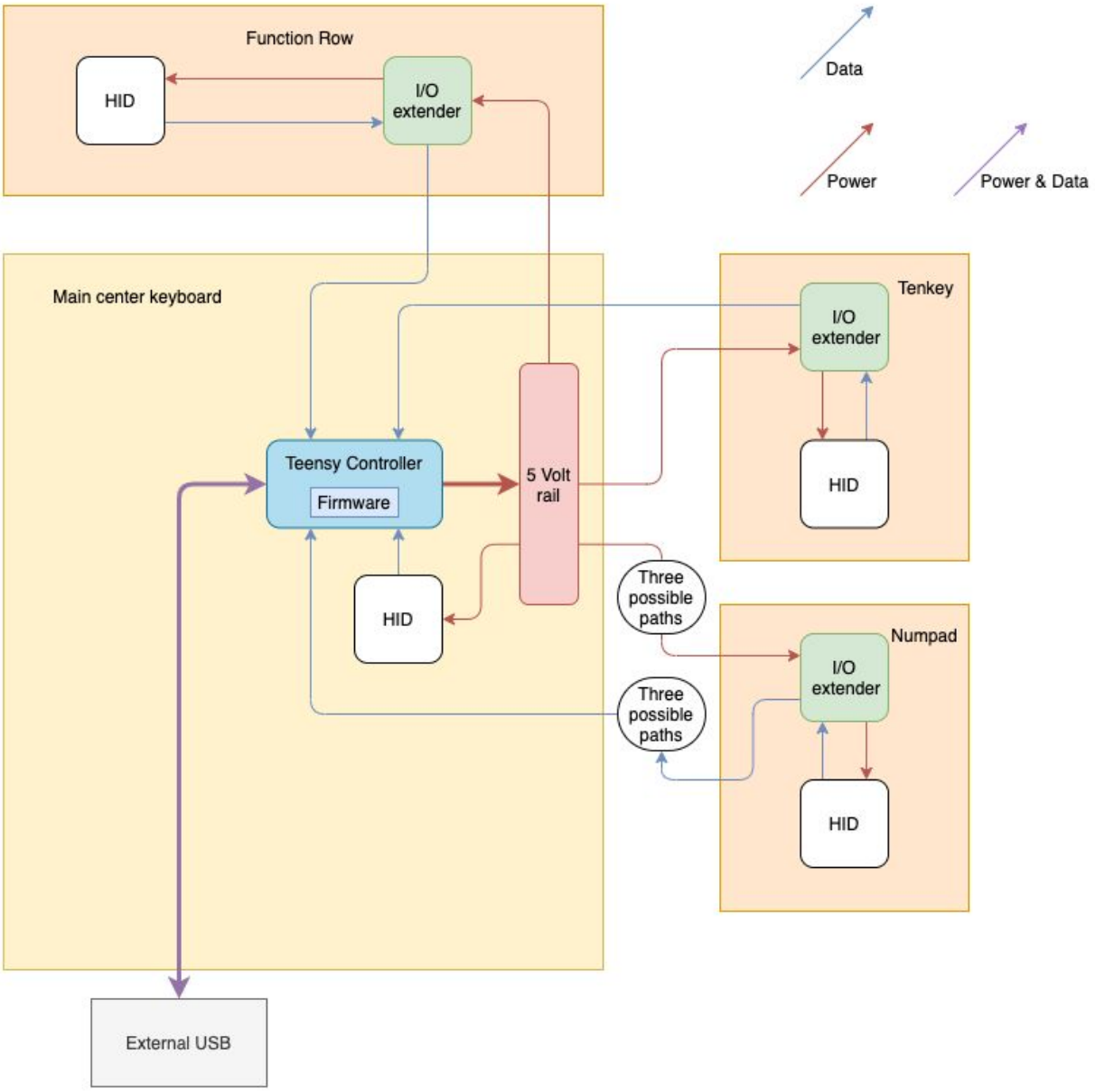

Figure 2. Block Diagram

An external USB powers the main microcontroller that is the operational key piece. With just the base keyboard connected solely the keyboard can still function. The other modules will take power and give data through a wire connector. When connected the microcontroller will recognize their data through an I/O extender from the respective module. The firmware on the microcontroller will be able to process any number of key presses, but the main limitation will be on the data feed back to the computer.

2.2. Schematic:

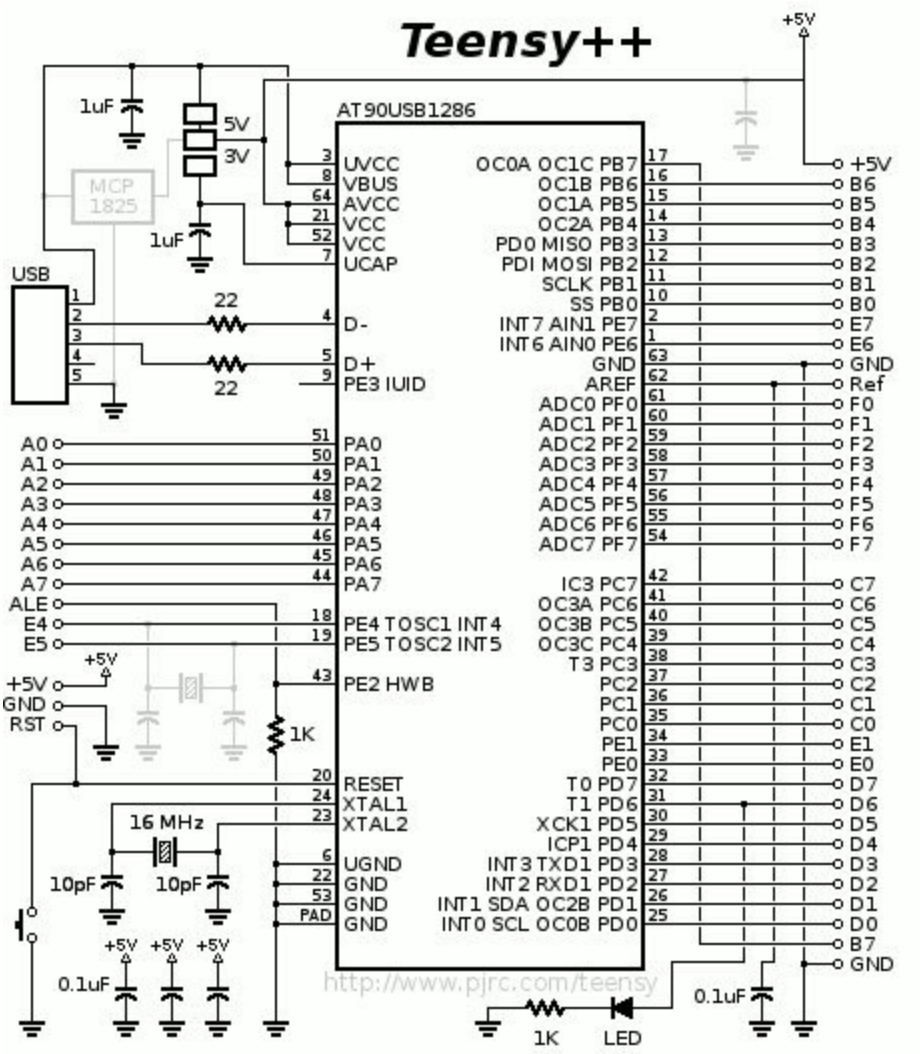

Figure 3. Provided Schematic for Teensy 2.0++[1]

[1] PJRC

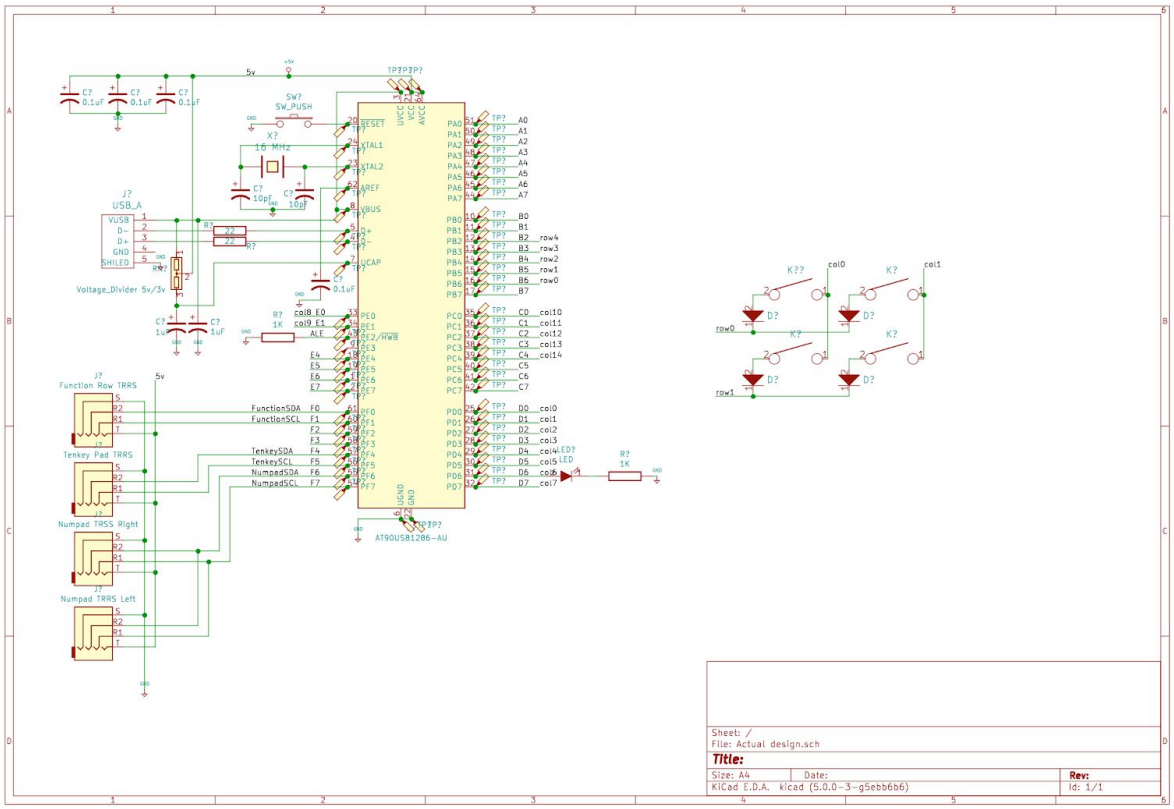

Figure 4. Kicad schematic.

Figure 4 shows the first steps to the keyboard PCB. The main challenge tackled in this figure is the microcontroller. Luckily, Teensy provides a picture of their schematic and so the difficulty is just attaching the right amount of keys and TRRS connections (commonly known as an audio jack; this has four channels in the Tip, two Rings and one Shield). Only an example of the key layout is shown due to the fact that it would dominate the whole image. The shown example is fully modular and expandable.

2.3. I/O Expander

2.3.i. These components on the exterior modules are the hub for power and data transfer on the modules. They receive power from the main 60% keyboard and use it to allow the HIDs to send data back to the I/O extender. It interprets the data and returns it back to the central keyboard via I2C connection.

Requirement	**Verification**
Have a polling speed that can interpret key presses and send an I2C signal to the microcontroller within 10 ms .	Give a unit step signal to both the I/O expander and oscilloscope. Verify that the response signal from the expander is within 10 ms.

2.4. Microcontroller

2.4.i. The microcontroller receives power from the computer and disperses it through the connected systems. This power is sent through the matrix that contains all the keyboard switches, as well as to the external components through the I/O extenders. The HID and I/O extenders will send back data they receive, which is organized and sent to the computer as typing inputs. The power and the thinking of the keyboard is performed by the microcontroller.

Requirements	**Verification**
Have a polling speed that allows a complete scan of the keyboard in around 100 ms. Each key should be able to be read (assuming the module is attached). Be able to run the firmware.	Give a unit step signal to both the controller and oscilloscope. Verify that the response signal from the expander is within 100 ms. Run the previous test additionally on all rows and columns in unit tests. Give an I^2C signal (provided from I/O expander) and verify it responds within 100 ms.

2.5. Key Switches (HID)

2.5.i. The human interface devices for our keyboard are mechanical key-switches which when pressed shorts a connection and completes a circuit. When these switches are combined into a matrix and hooked up to a microcontroller, firmware for the microcontroller can match these completed circuits to which keys were pressed and put letters from our fingers on to the screen. As the main component that interacts with the user, this is one of the most important components and provides most of the usability and with its specific configuration will do much of the organizing for the microcontroller.

2.6. Connectors

2.6.i. The wiring between the main hub keyboard and the modules will be a TRRS cable. This is commonly known as a 3.5 mm jack. The TRRS version has four separate wires associated with it. The Four things transferred will be five volts, ground, SDA, and SCL (the latter two are part of the I2C protocol and are the only lines that return from the modules to the main hub).

2.7. Firmware

2.7.i. The firmware is software coded in the microcontroller that will detect the attached-ness of modules, scan the keyboard matrix for pressed and released keys, interpret keys based on preset layouts or user determined values, and send out corresponding signals through anUSB connection.

The firmware also provides programmability, allowing users to modify their keyboard layout to produce user-determined characters or potentially commands. Examples include changing the function layers, having programmable keys, and other features that would help with user productivity and customization.

Requirement	Verification
1. The combined keypad matrix should have at least 101 entries, corresponding to 101 keys of a traditional keyboard. Keypad matrix for the default keyboard without extensions should cover around 60% of the keys, while each attached auxiliary module would extend the matrix by their size. The firmware needs to periodically scan pressed keys, map their coordinates to find their values, and send out signals accordingly. 2. The firmware also needs to periodically poll the status of attached auxiliary modules and adjust the scanning range accordingly. 3. The keypad matrix should be customizable through user input, allowing the user to change corresponding functions of programmable keys.	1. Produce input signals corresponding to each entry and verify that the firmware has sent out all signals correctly and in the correct order. 2. Produce signals corresponding to a matrix entry on an auxiliary module with and without sending an input indicating that the module is attached. Verify that the firmware only sends out a signal when the module is attached, and that the signal is correct. 3. Produce a set of signals that assign different interrupts to all programmable keys. Then send keypress signals for all programmable keys in random order. Verify that the output matches key customizations and is in correct order.

2.8. Tolerance Analysis:

2.8.i. According to Wyma et al, the human reaction time is about 210 ms from stimuli to response. Our goal is to have a keystroke event (between human interaction to sending the data to the host computer) last about 100 ms. This is so the host computer has enough time to process and

send the event to wherever it needs to go. The datasheets as shown by Cherry MX indicate that the average debounce time for a keystroke is about 5 ms. Assuming the longest pathway is from the exterior module keystroke, the path is as follows: Keystroke, I/O extender, Microcontroller, USB signal. Assuming we use the initial maximum speed for I^2C protocol of 100kHz (I2C INFO) This means that each 8 bit signal from the I/O extender will take approximately 8*1/100kHz = or 0.08 ms. The microcontroller we have preliminarily selected has a clock speed of 16MHz. This is a high enough value that we can assume that the time to process a signal from the I/O extender is just another round of 0.08 ms of processing time. USB 2.0 protocol indicates that the maximum speed for USB is 480Mbits/s; again, this is fast enough that we can add another 0.08 ms. This means that a signal can be gathered in about 5 ms + 3*0.08ms = 5.24ms. This can be made faster by using a higher I^2C protocol.

3. Cost and Schedule

3.1. Cost Analysis: Include a cost analysis of the project by following the outline below. Include a list of any non-standard parts, lab equipment, shop services, etc., which will be needed with an estimated cost for each.

3.1.i. Labor: (For each partner in the project)
Assume a reasonable salary
($/hour) x 2.5 x hours to complete = TOTAL
Then total labor for all partners. It's a good idea to do some research into what a graduate from ECE at Illinois might typically make.

3.1.ii. Parts: Include a table listing all parts (description, manufacturer, part #, quantity and cost) and quoted machine shop labor hours that will be needed to complete the project.

3.1.iii. Sum of costs into a grand total

3.2. Schedule:
Include a time-table showing when each step in the expected sequence of design and construction work will be completed (general, by week), and how the tasks will be shared between the team members. (i.e. Select architecture, Design this, Design that, Buy parts, Assemble this, Assemble that, Prepare mock-up, Integrate prototype, Refine prototype, Test integrated system).

4. Discussion of Ethics and Safety:

4.1. According to 1.2 of ACM code of conduct, we should design every part that could come in contact with the user to be safe to touch and interact with. Such contact points should not abrade the user in any way. With using ACM 2.8, any firmware or drivers we utilize will be our own construction or open source material. Our Design for this keyboard is intended to be touched and used directly by humans.This means that we should adhere to ACM 3.3 by creating designs that

are ergonomic and help the user have a more efficient experience, a crux of our background to this project.

5. **Citations:**

5.1. PJRC. (n.d.). Teensy Schematics. Retrieved February 22, 2020, from https://www.pjrc.com/teensy/schematic.html

5.2. L., D., Wyma, M., J., William, E., Herron, J., T., … Reed. (2015, February 26). Factors influencing the latency of simple reaction time. Retrieved from https://www.frontiersin.org/articles/10.3389/fnhum.2015.00131/full

5.3. CHERRY MX SPEED SILVER. (n.d.). Retrieved from https://www.cherrymx.de/en/mx-original/mx-speed-silver.html

5.4. I2C Info – I2C Bus, Interface and Protocol. (n.d.). Retrieved from https://i2c.info/