Capstone Project Name

Authors: Ben Sun, Korene Tu, Zhejia Yang
Affiliation: Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*—There are many macro keyboards on the market that have creative and useful layouts, but many of them are upwards of $100 each for a high-quality board. We aim to create a macro keyboard that has an infinite number of layouts that any user can configure to their heart's desires. It must have a long battery life to last for long work periods, still have the performance of regular wireless keyboards today, be convenient in terms of charging and portability and of course must be cost-efficient. Our FP(Key)A fulfills all of these specifications to a degree.**

***Index Terms*—3D Printing, BLE, Mechanical Keyboard, Wireless Charging**

1 INTRODUCTION

Over the past 5 years, the mechanical keyboard market has exploded in popularity, with year over year growth of upwards of 10% since 2018 [2]. Such keyboards provide an outlet for self expression, with the user having the ability to customize the switches, keybindings, keycaps, and more to match their use case and aesthetic preferences. However, one area of customization that has remained prohibitively expensive to explore has always been the keyboard layout itself. Often times, mainstream custom keyboards and macropads are limited to the typical staggered QWERTY layout or a grid-like ortholinear layout. The only way to explore other, more ergonomic or project-specific layouts is to design your own or spend money on many different keyboards. With each custom mechanical keyboard possibly costing between $200-600 factoring in switches, keycaps, and housing, many users are put off from exploring different layouts and simply learn to compromise with what they have [1].

To address this shortcoming in current keyboard offerings, our project is to create a keyboard whose keys are independently movable to any position the user desires. Each key on the keyboard is able to be programmed through a central configurator software which remembers the key assignments across devices. This project is aimed at video editors, streamers, gamers, and generally people who need flexibility in keyboard layouts, as the software they use has different keys for shortcuts and game hotkeys in particular locations. Thus, creating a layout that matches the onscreen UI ends up being more user-friendly than using a fixed staggered QWERTY keyboard. For example, for rhythm games that require you to hit or hold certain keys on time, the keys can be grouped by functionality and the most used keys can be kept close to your hand instead of scattered throughout the typical staggered QWERTY layout.

Due to time constraints, we created a proof of concept 16-key macropad. This will be sufficient to demonstrate the viability of BLE as a connection protocol between the different keys, the wireless charging capabilities, and individual configurability of the modules. Additionally, since 16 keys is enough keys to house at least 16 shortcuts, sufficient for most video editing uses. We also felt that 16 keys was a good breakdown for 4 player split screen games, where each player would have access to 4 movement keys, or for a single player to have movement, 4 action keys, and 8 hotkeys for other uses.

2 USE-CASE REQUIREMENTS

We have identified the following use case requirements:

1. Speed: Latency of less than 50 ms. Users such as gamers need fast response times, and 50ms is what is typical for wireless gaming keyboards currently.

2. Longevity: Battery life of over 2 days per key. Users want their keyboards to work at least for the longest possible use sessions.

3. Expense: Cost of less than $300. Users would not pay for a customizable keyboard if they can buy two separate keyboards with different layouts for less.

4. Portability: weight of less than 1 pound. For users to want to carry the keyboard around, it needs to weigh around the same as current ”portable” keyboards.

5. Convenience: mostly wireless operation. Users do not want to deal with the hassle of managing multiple wires on limited desk space.

6. Customizability: Includes layout customizability with independently adjustable keys and typical customizability of a mechanical keyboard like switches and keybinds (the letters bound to each key). This is our most important requirement, as having the ability to configure the keyboard to any layout the user allows them to tailor their keyboard to the shape of their hands, reducing typing fatigue and increasing comfort. Additionally, having the ability to reconfigure the layout on the fly means that the user needs to buy fewer keyboards to try different layouts, potentially saving multiple keyboards from becoming e-waste due to user dissatisfaction.

3 ARCHITECTURE AND PRINCIPLE OF OPERATION

Provided below are the block diagrams for our keyswitch module and the wireless charging module. The keyswitch modules each communicate with a central BLE receiver module (the controller) which is connected to the computer. The user sets keymaps by setting keybindings in the configurator software, and then flashing the resultant firmware to the controller board. Each keyswitch module is able to be recharged on the wireless charging module, which can charge a key to full overnight.

3.1 Keyswitch Module

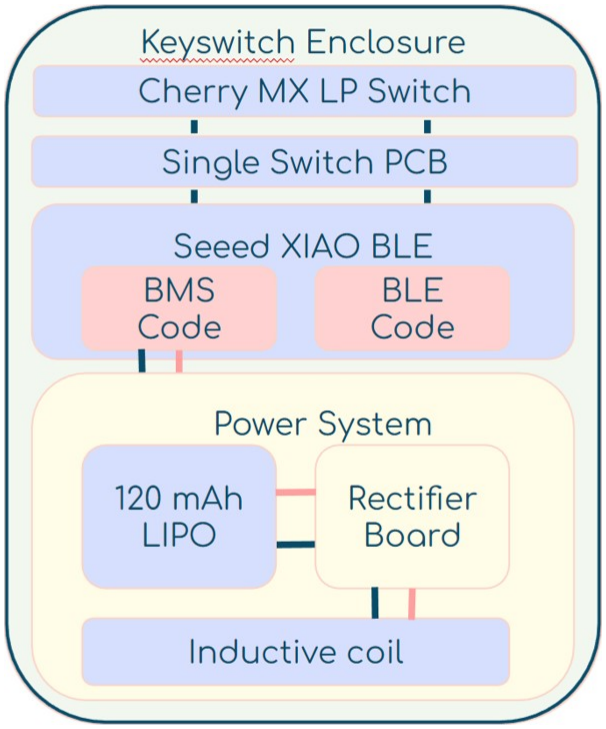

Figure 1: Keyswitch Module

The keyswitch module is the first half of our modular keyboard system. Each keyswitch module contains a Seeed XIAO nrf52840 (hereon referred to as the Seeed) board which has BLE capabilities and an integraded BMS chip. This board enables us to charge the battery both from the built in USB port and the wireless charging inductive coil. The keyswitch modules is identified uniquely by their MAC address, allowing them to be identified on the configurator. Each keyswitch module has a custom designed 3D-printed enclosure which keeps everything together. The switches are attached to a single switch PCB using Mill-Max 3305 sockets, which allows for hot-swap capabilities. The single switch PCB is then soldered to the Seeed board to allow the reading of keypress data. The switch pins are pulled up using the internal pull up resistors on the Seeed, and a keypress is registered if an interrupt is triggered upon pin state change.

The only update we have made to the architecture since the Design Report is adding a hotswap PCB between the switch and the Seeed board to allow users to swap in their own switches as they wish.

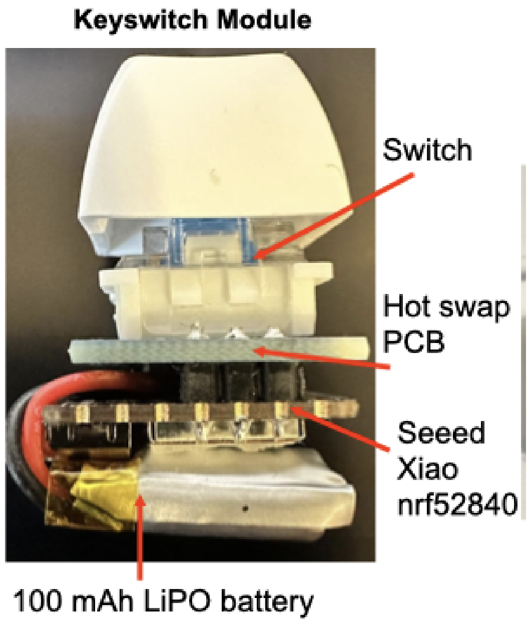

Figure 2: Physical Implementation

3.2 Wireless Charging Module

The wireless charging module consists of multiple inductive pads with markers over them delineating to the user where to place the individual keyswitch modules to charge the keyboard. Each inductive transmitter pad is an off-the-shelf coil with a corresponding receiver coil located inside each keyswitch module.

We bought two types wireless charging sets, which act as our evaluations boards that have a **XKT-412** oscillator and a **XKT-335** power amplifier as the wireless transmission chips and wireless receiver boards with a **T3168** receiver chip. The transmitter chips power the receiver chip through the inductor coils in order to supply energy. This way, the user is able to charge each keyswitch without plugging in wires. The circuit diagram of both wireless charging kit's transmitter board is found in Figure 4 and the circuit diagram of the receiver board is found in Figure 5. Our original goal was to have up to 4 keys able to be charged simultaneously or have a single wireless charging pad working as a proof of concept, but due to cost and shipping time concerns, we can only guarantee a single key as a demo for our wireless charging.

Our final proof of concept implementation was using the **Wireless Charging Set 10mm Coils** from **Amazon** [6]. We adapted the boards to use with the inductor coils that we bought from Digikey as that would minimize our key housing.

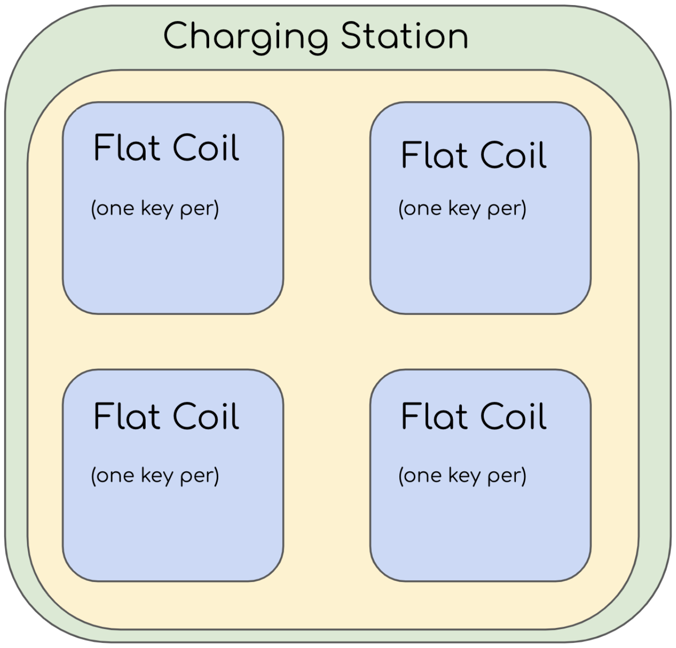

Figure 3: Charging Module

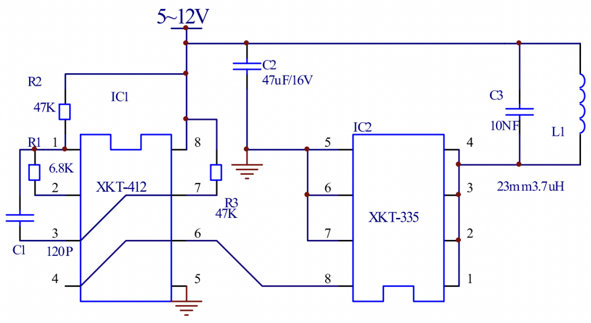

Figure 4: Transmitter Circuit Diagram

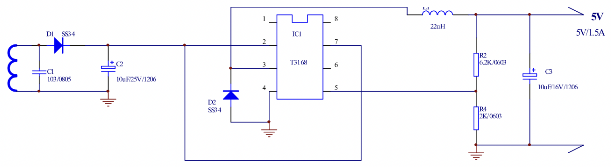

Figure 5: Receiver Circuit Diagram

4 DESIGN REQUIREMENTS

We have identified the following design requirements as a result of our use case requirements

1. Dimensions of less than 25x25x25mm for the base (not including switch height). This requirement comes from the comfort aspect, as if the keys themselves were any larger, then the keys would not be able to be placed side by side, thus limiting comfort to smaller hands. The height limit comes from the convenience factor. We want the keys to be easily gripped, but not so tall that the typing experience is hampered by key wobble. Note that this requirement has been updated from the Design Review to accommodate more sturdy housing while maintaining user comfort since the horizontal spacing is just as adjustable to different hand sizes. The vertical spacing of keys does not affect comfort as much as fingers can move vertically much more easily than they can move horizontally since human fingers are aligned vertically. Also, many typical keyboards space their keys out more vertically than horizontally. Furthermore, this study done at Berkeley in 2014 shows that while reducing key spacing beyond 16mm reduces accuracy and words per minute, there is no significant difference in performance for spacing above 16mm in both directions [3][4].

2. 100mAh battery capacity. This stems from the longevity requirement. Based on previous experience with LIPO batteries in wireless keyboards, 100mAh should be enough to last for 2 days of use without needing to be recharged since the active current consumption is 6.5mA while the sleep current consumption is 0.7mA. Assuming users are using their keyboards for 8 hours a day but only continuously typing for 70% of that time (taking breaks to think etc), users would consume 44.27mAh per day, leading to a little over 2 days of battery life. Additionally, 100mAh batteries are small enough to fit within the footprint of the dimensions [7].

3. BLE wireless capabilities. Based on the latency and customizability requirement, the keys must be able to communicate wirelessly to the central controller. BLE meets the requirements on latency and also has good library support with the Seeed boards that we are using. Additionally, the LE (low energy) part of BLE helps with extending the battery life even more.

4. Key stability. For the convenience factor, we need the keys to be stable while mounted to the baseplate. Any major wobble detracts from the user experience, and make it distracting to use.

5. Reliability: The keyboard must be able to register every key the user presses, regardless of the timing they press they keys. For this requirement, each key must be able to register at least 10 consecutive key presses correctly as that is more than the typical amount a single key will be used in rapid succession.

5 DESIGN TRADE STUDIES

5.1 I^2C

Originally, we wanted to have each key module connect to each other via I^2C using contact strips on the side of the module. Each key would have it's own address and the central device would poll each key to determine its state. However, we ultimately decided against this approach as it limited customizability too much and could hamper the user experience. Having the requirement for every key to be touching every other key does not allow for as many layouts as if each individual key was independent of the rest. However I^2C would have offered the ability to more easily identify the keys, as each key's address would be known to the central controller, and the general programming would have been easier as there would be no need to deal with wireless communication protocols. Additionally, each keyswitch module would have been cheaper to produce since it would require no active components such as microcontrollers, antennas, or receivers.

5.2 Creating or Buying: Microcontrollers, Keys, Charging

A large part of our design trade studies is understanding what comes with designing your own electronics versus when to buy the product on the market. A large part of decided what parts to buy and what to design ourselves was quite difficult to organize. First we were unsure how we wanted to implement BLE into the keys and whether we wanted to make our own PCBs. Ben had found the Seeed Xiao board which is perfect for our use case of having a small but reliable and convenient to insert into 3D housing. Buying 16 key microcontrollers would be much easier than first creating test boards and then figuring out if they could transmit or not. Another few things on debating whether to buy or not were flat coil inductors as well, as Korene had found a method of making flat coils and suggested making them could save money, however Zhejia suggested back that mass produced inductors would be much more consistent and stable in what the inductance values would be and would have less room for error. While it may save money to create coils, it saves a lot more time to buy the inductors instead.
The microcontrollers for the keys to relay data to and then for the microcontroller to relay to the computer were also debated on whether to buy an ESP32 or not. However, the main argument is that because we had decided on 4 client BLE chips to collect data from the 16 keys, we decided it would be much easier to lay out a microcontroller to have 4 receiver boards and then connect them directly to a USB-C connection for ease. Finally, the situation that our team had debated for a long time was whether or not to create the charging circuit or buy an IC chip that follows standard charging protocols and has an ecosystem. While creating the charging circuit was possible, especially with 18-220 knowledge, there are many more things to consider in the circuit such as how a sine wave could be generated or how to properly increase the current to have a sufficient power delivery and be consistent in its charging to ensure the safety of the LiPo battery as well. In the end, we've decided on using the wireless transmission chips of the XKT-510 and T3168 where it has it's own ecosystem and is already small enough to our advantage and the use case requirement of the keys to be small.

5.3 Keys

We had shifted our original plan of making a Ten Key Less (TKL, no number pad) keyboard which would have 87 keys to a 32-key keyboard with only the letters and a few special keys like Enter, Tab, and Spacebar to a 16-key macro keyboard. While having a full keyboard with all of the keys used in daily life would be ideal and incredible for the number of possible usages there could be, we had narrowed down that for a full keyboard, most people would not need to constantly change 87 keys on the regular and there are already difficulties in whether our connections could work properly with the BLE modules which are reported to support 8 devices. As a whole, it is also much more cost efficient and productive to ensure a set few keys are able to work properly to also be proof of concept that a full keyboard could be made in the future. In addition with a macro keyboard, while it is fewer keys to use, it is tailored towards the specific market of keyboard enthusiasts who are more willing to spend more for a keyboard and open to novelty ideas such as the infinite keyboard layout. The board is tailored to a more specific audience who would find use cases of a macro pad, which is commonly used in gaming and shortcut configurations.

6 SYSTEM IMPLEMENTATION

6.1 Power

In order to supply power to our switches, as seen in Figure 1, each keyswitch has a 18mm x 14mm x 7mm (length x width x height) rechargable 100mAh LIPO battery which acts as the power supply for the **Seeed Studio XIAO nrf52840 board**, which will be referred to as the seeed board from now on. We chose this LIPO since it has a small enough footprint to fit within our keyswitch housing as has a enough power to last an entire standard wireless keyboard about 36 hours under typical use, allowing our keyboard to meet our battery life requirements.

We use **inductive charging** which is a type of wireless charging to deliver rechargeable power to the keys wirelessly to reduce hassle for the user.

We bought Qi compliant transmitter and receiver coils from Digikey that have a metal sheet built in to prevent power loss through magnetic leakage, an image is show in Figure 6. Qi is a wireless charging standard commonly used for inductive charging.

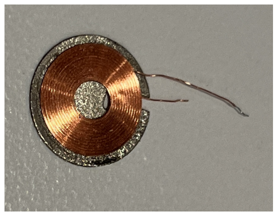
Figure 6: 8.62uH Inductor Coil

This coil with the metal ferrite is placed at the very bottom of the keyswitch housing as depicted in Figure 1, and connected to a rectifier board. These 3 things: *coil, sheet, and rectifier board* make up the **inductive power receiver**. This power receiver is connected to the Seeed board via the exposed V_{USB} and GND pins, which feeds into the Seeed board's built-in power management system and charges the 100mAh LIPO battery supplying power to the Seeed board.

For our **inductive power transmitter**, it will be part of power station where the user would put their keys in the station and leave them there to charge overnight. While ideally, the user would not have to take off the keys at all, for this first prototype, we are using the more well-researched and consistent inductive charging, which requires close coupling and has faster power delivery, and since the keyboard lasts though 2+ days of continuous use during waking hours (12 hours a day), taking off the keys before bed and charging them overnight should not be too much of an inconvenience.

The power station would be made of a transmitter board and transmitter coils with a metal sheet underneath the coils. We decided to use the **XKT-412** (an oscillator) and **XKT-335**(a power amplifier) for wireless transmission chips on our transmitter board and the **T3168** on our receiver board, which are simple and cheap Qi wireless charging compliant wireless charging chips. Ideally, in a finished charging station, there will be one transmitter for each key, making a 1:1 correlation between transmitters and receivers, but due to supply issues like cost and delivery time of wireless charging chips themselves, We have bought 2 types of evaluation board sets for these 3 chips instead and will only be demoing one transmitter-receiver pair for wireless charging.

Our first set was from **Adafruit** and made to use with a 30uH inductor coil. We tried to change the capacitances and resistances in order to have the chips work with our inductor coils that are **8.62uH** following the resonance frequency formula. Unfortunately, that plan would only work if the transmitter chips we had were a voltage-controlled oscillator or similar to a 555 timer. However, though adjusting the resistors connected to the XKT-412, we discovered the oscillator chip was an LC tank instead and the frequency generated was not high enough to function properly with our inductor coils. Additionally, adjusting the capacitance values did not affect the LC circuit as a whole and therefore was ineffective in allowing the inductor to resonate at the oscillator's set frequency. This forced us to move to plan B and we used the second set of evaluation boards from Amazon because it was a small coil that fit our sizing metrics and hoped that we would be able to adapt our own coils with the board. We only needed to make minor adjustments to the boards to work effectively with the coils we bought before.

6.2 KeySwitch Communication

To communicate our keyswitch states (pressed, unpressed) to our central micro-controller without using wires (since wires would overly complicate the process of moving switches around), we are using the Bluetooth Low Energy protocol or BLE, which is often used in smart home environments where many nonconnected sensors need to communicate wirelessly. BLE is both low power, helping us meet our battery life goals, and low latency compared to Bluetooth due to short data send preparation times at the cost of lower throughput. However, this fits our application since sending 1 binary state (on or off) per keywitch does not require high throughput.

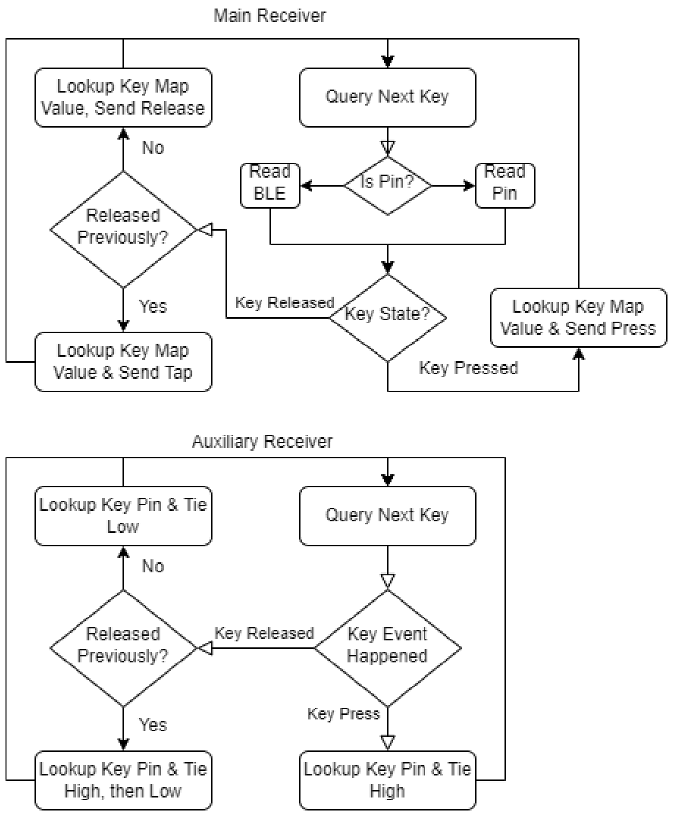

Figure 7: Key Receiver Software Flow

We are using the **Seeed Studio XIAO nrf52840 board** for its built in BLE and BMS capabilities. Furthermore, The Seeed XIAO is programmable using Ardunio C++, which allowed us to use **Arduino BLE library** for easy BLE connectivity as well. Meanwhile, the receiver setup consists of a Arduino Nano 33 IoT acting as the central receiver, with two Arduino Nano 33 BLEs acting as auxiliary receivers. Each of the receivers subscribes to a maximum of 7 switches at once, being alerted if their state changes. The limit of 7 seems to be a hardware limit, with this point only being discovered after 2 weeks. The straightforward idea of raising the defined value for **DMCONNMAX** did nothing to resolve the issue. Next, we tried changing the RTOS thread stack size, in case there was a memory limitation, but that also did not work. Finally, we attempted to change **DMCONNMAX** and recompile the RTOS, which did not end up helping either. Looking into the code, it turned on out that there was a status register that received the incoming BLE events and outputted a status. This status would always return in error whenever the 8th device tried to connect, so we concluded that this was most likely a hardware constraint. After making no progress in this department, we decided instead to figure out alternative solutions, which is where the idea of auxiliary receivers comes in. The auxiliary receivers take their received keypress data and write the recorded key value to the corresponding pin. The main receiver then takethe data from the keys that it's connected to, as well as reading the forwarded press data of the auxiliary receivers, and converts that into keypress data to be sent to the computer.

To act as a keyboard, the Arduino Nano 33 IoT uses the Arduino Keyboard and USB library, which allows it to be recognized as a keyboard by the recipient device. The receiver is connected to the target device via a **USB-C** connection just like a normal keyboard. Other options for the main microcontroller were considered such as another Seeed XIAO BLE, but the issue with these devices was that they are unable to connect more than 2 peripherals at once, meaning that we would need an prohibitively large (and expensive) number of receivers. We also considered other, cheaper BLE receiver boards such as the Adafruit BlueFruit, but these were not compatible with the ArduinoBLE library. As we did not want to completely rewrite the software around these boards, they were ruled out. Other Arduino BLE compatible boards such as the Arduino Uno Wifi Rev2 or Nicla Sense ME were either too big or too expensive to be practical as a receiver. The software flow diagram for the key receivers can be found in figure 7.

6.3 Keyboard Housing

The housing to hold both the components of each keyswitch together and the charging station was modeled using Solidworks and 3D-printed based on the dimensions of the components listed in 1. The baseplate that provides the freely adjustable keyswitch area and is made of a **thin acrylic plate** with a **3M Dual-Lock** layer on top for securing each keyswitch. We ended up using wood for the support plate since it is both light and strong, as well as being inexpensive (we picked one up for free), helping us satisfy the portability requirement. To prevent the sliding of the baseplate while typing due to it's light weight, we added *rubber feet* to the bottom of the baseplate.

Implementation was smooth, there were overall four iterations of housing design, one to test sizing and joints, one to implement the USB-C hole and key switch on top, one to readjust sizing again, and one to implement with the wireless charging PCB.
The final design of the 3D key housing is 22mm x 24mm x 20mm (WxLxH). There is a top and bottom portion with 2 cantilever joints to snap the two portions together to close the housing and there is one hole on top for the key switch to snugly snap into and there is a hole for the USB-C outlet and wireless charging PCB. See Figure 8 for the CAD image. We attached velcro onto the bottom of the keys and onto the wooden plate we picked up from TechSpark to finish off the exterior. We decided on wood for the reasons listed above, but also because we were unable to find a suitable acrylic plate, however it still serves the purpose of being lightweight and portable

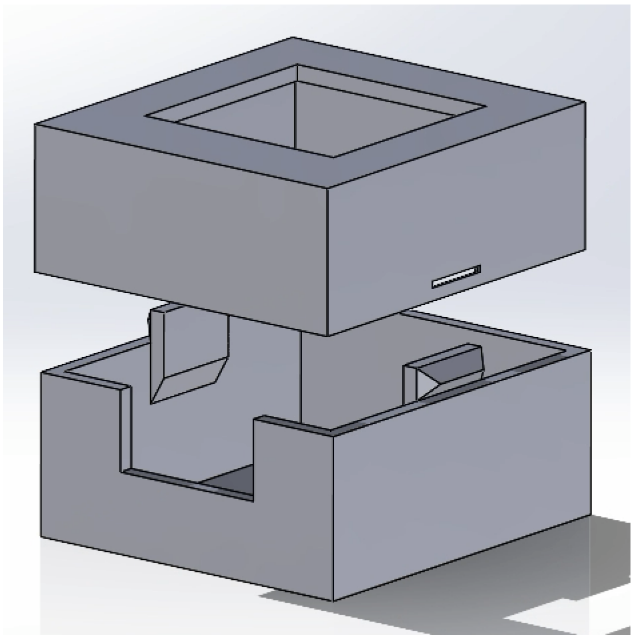

Figure 8: CAD of Housing

6.4 Keyboard Configuration

To make the keybindings - the letters and shortcuts associated with each key - customizable, we created a desktop application that allows the user to specify keymaps: maps of letters, symbols, and shortcuts to particular keys. This software is created using pyqt6 in python to utilize both the speed of C (pyqt's methods are implemented in C) and the convenience of python. Each key is a QLineEdit, a class (implemented in C) that allows users to click and type words and letters into it. We created a child class that allows the keys to be moved by clicking and dragging to mimic the layout freedom of the physical keys. During programming, which is done via the big "Configurate!" button seen in Figure 9, not only do the keymaps get sent to the main microcontroller, but they are also saved to text files along with the location of each key, so next time the user opens up the software, the previously programmed keymaps and key locations remain.

To program the firmware, the software generates a string containing the keymapping of each individual key. This string is then sent over UART to the main receiver, which interprets the string as a new keymap. The main microcontroller also has logic to parse special commands such as modifier keys and layer switching keys.

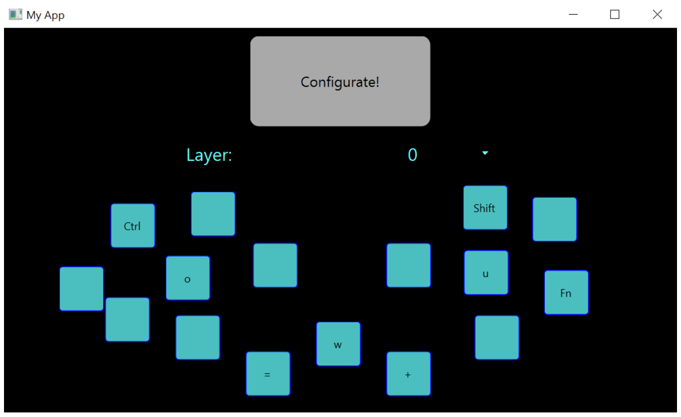

Figure 9: Keyboard Configurator Software

7 TEST & VALIDATION

Our testing methods evaluate how our final project holds against the set design requirements and use case requirements. In order to have a high-quality keyboard with novelty and practicality, we have noted previously that we need a high-latency keyboard that has a prolonged battery life with charging ease and it is cost-efficient. The keyboard should be stable to use, has easy portability, and fulfills its primary goal of being an infinite layout freedom keyboard.

7.1 Tests for Latency

Latency is the speed test of the keyboard where we observe how fast the response from the press of the wireless keyboard is registered on the computer. A fast response is necessary for applications like gaming and ensuring that the user will not see a significant delay in the usage of the keyboard.

Our use-case requirement for Latency was 50ms, which is the average latency of a Bluetooth keyboard. Our measured latency was 37.5ms < 50ms, which was obtained by counting the frames until the typed letter showed up on our phone's slow motion camera.

$$\frac{1000\text{ms}}{\text{s}} \times \frac{\text{s}}{240\text{frames}} \times 9\text{frames} = 37.5\text{ms}$$

Latency in BLE is theoretically limited by the minimum connection interval between transmitter and receiver devices. This can be as low as 7.5ms, but as a consequence, limits the number of BLE devices you are able to connect. We found that with a minimum connection interval of 31.25ms, all 7 BLE devices could be reliably recognized by a single receiver. However, this inevitably sets a lower bound on latency at 32ms, as we can only register a new event every 32ms per key. The additional 5ms of measured latency comes from a combination of display input latency, processing code, and driver overhead. While we are unable to quantify the exact impact of the screen input latency, 5ms is pretty typical for displays. The latency for the code is no more than a few hunderd microseconds, as with the processor running at 64MHz, five thousand instructions, which we figure is probably the amount it takes to process a BLE request, can be executed in 75 microseconds. The latency was improved mostly through the tuning of the connection interval. We originally started with a connection interval of 62.5ms, but this obviously led to us not meeting the 50ms latency target. Slowly, we decreased our connection interval by 6.25ms (which was the maximum granularity we could get), we eventually found that 31.25ms was the minimum we could push it before connections became unreliable. Thus, the overall worst case latency can be expressed as follows:

$$31.25\text{ms} + 5\text{ms} + 0.01\text{ms} = 36.25\text{ms}$$

Even though this is less than our measurement, it's important to remember that our 240fps slo-mo video only has an granularity of 4.17ms.

7.2 Tests for Battery Life and Charging

Battery life is crucial as it determines how long without interruptions can the user use the keyboard for a prolonged period of time and how convenient is the charging mechanism so the user would not be burdened with waiting a long time for their keyboard to charge. For this test, we use time as our metric to fulfill where the charging time for all of the keys must go from 0%to 100% in 8 hours or less, which was tested by observing the charging current supplied by the wireless charger. Specifically, we placed an ammeter in between the charger and the charger pin, ensuring that the green charging LED was lit up. We measured a charging current of 17mA being delivered, which is more than the 12.5mA required for the 8 hour charging time. In fact, the estimated charging time is now

$$\frac{100\text{mAh}}{17\text{mA}} = 5.88\text{h} < 8\text{h}$$

For how long our keyboard lasts, we recorded the current drawn by the processor from the battery while it was in the power on vs sleep mode. During power on, we found that the average current draw was 6.5mA while the key was transmitting keypress data, and 0.7mA when idling. This corresponds to a time on period of 10 hours transmitting data and 38 hours idle, which we feel closely approximates the average use time of 2 days, as it is unlikely for the user to sit there pressing an individual key for longer than 5 hours straight every day. This met our use case requirement of 2 day battery life, as even with 10 hours of on time and 38 hours of off time, we would still have 27.4 mA of battery remaining as a reserve. However, this is still slightly more than the minimum theoretical power draw achievable, which is close to $20\mu A$. The reason for this discrepancy is because to achieve this low of a power draw would require putting the processor into its deepest sleep state, resulting in a much longer time to wake that would negatively impact the user experience. Originally, our code used a polling approach, where we would poll the key pin many times a second in order to determine if a key was pressed or not. However, we found that this drew a large amount of current (9mA), and thus switched to using an interrupt based approach. However, we found that even with this interrupt based approach, the power draw was reduced, but only to the 6.5mA value we measured. To remedy this, we added in a sleep timer where if the key was idle, it would go into an idle state, saving on power until it needed to transmit again. This reduced the power draw to 0.7mA in the idle state, which we felt was sufficient to meet our 2 day battery life use case requirement.

7.3 Tests for Cost

This keyboard's goal was to be less than $300 in order to be worth less than buying 2 new macropads to test out a different layout. The cost calculation includes the total cost of the bill of materials used in the product plus an assembly fee.
Our resulting bill of materials is listed below, each key will be $29.70 per key with wireless charging (16 keys is $475.20) and $16.74 without wireless charging (16 keys is $267.84). A large component of the cost to wireless charging is because we bought a prebuilt board for it and thus costs quite a bit as these are not mass production quantities. In addition, the BLE boards and LiPO batteries are also quite costly when buying not mass-produced amounts. In order to demonstrate the mass production costs please refer to 2 where we show that each wireless charging key will be $15.57 per key, which is for all 16 keys $249.19.

$$\frac{\$249.19}{16\text{ keys}} + \frac{\$24}{2\text{ BLE Receivers}} + \frac{\$15}{\text{hour}} \times 1\text{hour} = \$288.19$$

So $288.19 is the final cost per keyboard for mass production (1 hour for assembly), which is below $300.

7.4 Tests for Stability

It's important that the keyboard feel steady when typing, especially at fast speeds, and for many different people to be able to type with ease on the keys. In order to test stability, we took videos of us typing on the keys while they were secured to the baseplate and measured the amount of visible deviation from side to side of the housing using a ruler. The end result was that we could see no perceptible wobble when measured to the ruler in the background. Through a combination of the wide base and the strong 3-M Dual Lock securing the key to the plate, we have met our use case requirement for stability. This is in line with what we predicted, as we specifically chose Dual Lock for its industrial grade grip strength and stability over velcro, which we feared would be too soft and cushy, leading to key wobble. We also iterated on the housing for this purpose, as originally we designed the housing to be 25mmx25mm, but found that our current 22mmx24mm was more than sufficient for stability.

7.5 Tests for Portability

Portability is important and the best thing about a wireless keyboard is that it could be used anywhere and is convenient to move around. The metric for portability was to have the weight of the total keyboard for the keys and board be less than a kilogram total, which is the typical weight of a mechanical keyboard. This weight was recorded on a scale to be 372g total, much less than our 1kg of required weight. While not specifically because of portability concerns, we also did shrink the size of the 3D printed key housings, which led to a minor reduction in weight. We did not really have a theoretical result for weight in mind, but based on the small number of keys we expected a value of less than 500 grams, as we were using all lightweight materials such as plastic and wood. As a side note, we found through experience that carrying around the keyboard prototype with the baseplate was not that much of a hassle, as the keys could be detached and thrown into the bag while the baseplate can be slid in to a computer pocket. The overall weight also did not feel very substantial from personal experience, although this part of the test is not exactly scientific.

7.6 Tests for Wireless Typing

The wireless typing test is necessary to ensure the keyboard can work wireless and without issue connecting to multiple BLE receivers. We ensured the whole system can support 16 keys to reach our goal of a successful macro pad with 16 usable keys. To verify this, we connected the receiver to the computer, waited for 30 seconds, and then tested every key was able to send key presses to the receiver. This was done in conjunction with the key reliability testing to verify both parameters at once. Originally, we found that there was an issue with connecting more than 7 devices to a single receiver, but we got around this limitation by having multiple receivers communicating their received keypress data to a central one. This matches our expectations as with 3 receivers, each receiver needs to handle less than the 7 connection limit to connect all 16 keys. Additionally, by spreading out the load over multiple receivers, we were able to reduce the time it took to connect all the keys, as each receiver would have to connect 1 fewer key.

7.7 Tests for Layout Freedom

This is crucial to the use case of having a keyboard to be used in many different configurations are a variety of locations and distances apart. We wanted each key to easily move in any direction while continuing to function properly. We tested this by putting the keyboard in a variety of common configurations of keyboards: split keyboard, circle cluster, arrow keys, and more configurations at a variety of distances as well to test the viability of spread out layouts. We found that the keys functioned correctly regardless of the key position, with key presses being able to be received even from 15 meters away. We expected this to work as, in theory, because the keys are wireless and communicate over BLE, there should really not be any limitations to where the keys can be placed.

7.8 Tests for Key Reliability

A keyboard should function seamlessly as an input device. For this to happen, the keyboard must register every single key pressed by the user. For this test, we pressed each key 10 times, sometimes in conjunction with other keys, and ensured that exactly 10 key presses showed up on the screen. Originally, we found that when pressing keys very quickly (<30ms between pressing and releasing), the key would only register the release and not the press, leading to some key presses being lost. We remedied this by adding a check to the last known key state, and if the key state was unpressed and another unpressed was received, then the software would interpret it as a tap, sending a single instance of the character onscreen. After performing the same tests again this time, we found that all 10 key presses were being reliably registered without issue.

8 PROJECT MANAGEMENT

8.1 Schedule

The updated schedule Gantt chart is shown in Fig. 11. The tasks are broken down by week starting from week 2 (Feb 06), and finish by week 13 (the week before finals week) and finish testing by week 14. We added in a section for testing, and dealing with the evaluation boards and poorly documented chips for wireless charging made getting it working take a longer time than expected.

8.2 Team Member Responsibilities

1. Ben - Programming each of the key modules and getting them to interface with a central controller via BLE.

2. Korene - Wireless charging controller research and prototyping. Design and test 3D printed keyswitch housings.

3. Zhejia - Wireless charging controller research and prototyping. Frontend app development of configurator.

4. All - Building and testing the keyswitch modules, integration of system components.

8.3 Bill of Materials and Budget

We have used most of our items purchased, except we did have some redirection with the inductive charging components and purchased a different kit that fits our specs with less modifications. Overall did not need to buy 40 inductor coils as we decided to only do a proof of concept. See Table 1 for Bill of Materials.

8.4 Risk Management

From the start, one of our worries was that BLE would not have adequate latency or bandwidth to support all the simultaneous connections required. Our first mitigation plan was to connect all the BLE devices up in a network, such that each device could forward values to other devices in a chain. However, this came with the risk of increased latency so we decided to not pursue it except as a last resort. After some experimentation, it turned out that we could only connect a maximum of 7 devices to a single BLE receiver. This seemed to be a hardware limit so our next plan was to figure out a way to get multiple receivers to communicate with each other. In the end, we settled on the current design where receivers communicate via tying pins high or low and connecting the pins of the receivers to the central microcontroller with headers.

The next issue we ended up needing to mitigate was the issue of wireless charging. We bought an inductive charging evaluation board for the chips we wanted, and we knew this would be an issue as the datasheet did not have very clear pinout diagrams or explanations for the functions of each pin. With little information on the internet as well, we conducted many tests to understand how the board worked. Specifically, the transmitter board gave us much more trouble than the receiver. Note that we needed to replace the coils on the evaluation boards with our own coils.

First, we tried to change the capacitor of the LC circuit to match our coil's inductance and the oscillator's original frequency in order to get the coil resonance. When that did not work we tried to adjust the oscillator's frequency to generate a resonant frequency. We iterated through each component and changed the values to see what the effect was on the output of the coils and then we would adjust the values to see if the capacitor changes would affect whether or not the coils were in resonance. Most of the changes resulted in no significant change. We concluded that the oscillator chip was an LC tank that had a working range of **70kHz to 110kHz** by measuring the output of the oscillator chip and using a potentiometer at the resistor. Beyond that range, the signal would deteriorate to be useless, so we would not be able to adjust the frequency to work with our inductor coils. In addition, when replacing the evaluation board coils with our coils, it also resulted in a short. That is primarily because of the 1Ω resistance of our coils compared to the 0.5Ω resistance of the board coils, which appears to affect the circuit to create a short. So because the board would not be able to change the frequency or change the capacitor that affects the LC circuit to generate the resonance frequency or even successfully attach our own coils to the board, we decided to look for a new board where the size of the coil was similar to our bought coils. This is to hope for a better chance of having similar inductances between our coils and the new board's coils and also as a back-up plan if we cannot fit our coils to the new board since they do fit in the key housing, albeit a bit pushing the height constraint. The new evaluation boards were measured to have an $8\mu H$ inductance which is very similar to our coils, and generated a frequency of **632kHz**, which was much higher than the first board's frequency. This new board worked quite well with our coils, which were $8.62\mu H$ and needed minimal adjustments.

9 ETHICAL ISSUES

While keyboard layout freedom is typically desired, if fallen into the hands of people who are not familiar with the ergonomics of keyboards or aware of the dangers of hand strain injuries, they may create a keyboard layout that is less ergonomic than a regular keyboard. Especially children or teenagers who may not have this awareness and buy the keyboard for it's "coolness factor", long term use of a non-ergonomic keyboard could increase their chance of hand injury later in life even if they notice no/minimal discomfort in the short term. Additionally, even after realizing the keyboard layout is causing discomfort, they may not have the knowledge to identify specifically which keys are causing it and fix the layout. To mitigate this risk, the keyboard could come with an info card that along with setup instructions, includes warnings on the dangers carpal tunnel and hand strain injuries along with some basic ergonomic layouts as a starting point.

Another possible issue is that this keyboard contains many small components that small children or pets might be tempted to swallow, which can cause choking or other issues if the keys get stuck in their bodies. To combat this, we can put warnings on the keyboard like "keep out of reach of small children: choking hazard" on the box like many other products with small pieces. Another possible mitigation strategy is to coat the keys in bitter material, but that is likely not necessary.

Thirdly, LIPO batteries are used to power each key, and LIPO batteries are notorious for their potential to explode and cause fires when they get overcharged or are dropped, which would pose danger to our users. To mitigate this issue, first of all, the LIPO batteries in the keys are small and don't contain enough energy to cause an explosion. If they were to malfunction, they would likely just smoke and die rather than cause any fires unless they were in a very oxygen rich enclosed environment. Additionally, the LIPOs are charged via the battery management system on the seeed board which prevents the LIPOs from being overcharged, further reducing the possibility of LIPO malfunction. Furthermore, this project uses many LIPOs which contribute to electronic waste, which is why we are using rechargeable LIPOs and preventing damage so they last as long as possible (3-5 years) to reduce waste.

10 RELATED WORK

Regular keybinding customizable keyboards with various fixed layouts have been on the market for a while and originated as an offshoot of the mechanical keyboard market, which was no longer being popularly manufactured with the new devices. More "ergonomic" layouts, however, did not really gain any traction until the introduction of the Planck keyboard around 2019. [5].

Bluetooth is a relatively new development in the keyboard space, with custom bluetooth keyboards not really existing until the introduction of bluetooth enabled Arduino pro micro devices such as the BlueMicro or nice!nano around 2020. Still today, however, the adoption and software support of custom Bluetooth keyboards is limited.

11 SUMMARY

Our system was able to meet both our use case specifications and our design specifications, although we did edit one of the design specifications through the process. Some limits include the battery life, which meets the specifications but would ideally be longer to reduce the hassle for the user. The battery life is limited by both the size of the LIPO, which given more time, we could get a slightly larger lipo like an 120mAh LIPO since our current LIPO is quite a bit smaller than the allowed footprint. Another way we could increase the battery life is to make the time waited for until sleep shorter so the Seeed boards sleep more often, but that would compromise on the user experience and latency.

Additionally, if we had more time, we could also improve the cost of our system, which is just barely in budget for mass production. First, we could designing our own wireless charging PCBs instead of modifying evaluation boards. Additionally, instead of using the Seeed Xiao boards, which are over-competent for our use case with many features we do not use like levels of computing power that can run machine learning, we can get a more specialized and cheaper board that specializes in just BLE.

Overall, our design uses Seeed XIAO boards with BLE capabilities and wireless individual power supply to each key to create a low latency, long battery life, freely layout customizable keyboard, where the keys can be placed in whatever layout and configuration and still deliver speedy keyboard responses to user devices. Custom keyboard users no longer need multiple different keyboards for their different setups and use cases.

Some of the lessons we learned were to not get too stuck on a single train of thought. Oftentimes, it appeared that we hit a dead end with one idea, but we still continued to pursue and debug until we realized that it would be much more efficient to work around the issue instead. For instance, we spent weeks figuring out why the wireless charging boards we bought were not working with our smaller coils before we had the idea to just buy boards with smaller coils. Similarly, we spent a lot of time figuring out how to connect more than 7 BLE peripherals to a single board, before considering the alternative of just using multiple receiver boards. In these kinds of time-limited projects, it makes more sense to fail fast, fail often, and stumble upon working solutions quicker than it is to endlessly pursue a single bug or issue only to have it end up going nowhere. Iteration and reevaluation is the best method of workflow.

Glossary of Acronyms

- BLE - Bluetooth© Low Energy
- BMS - Battery Management System
- LIPO - Lithium Polymer Battery
- mAh - Milliamp-hour
- UUID - Universally Unique Identifier

References

[1] Jake Harrington and Randall Jue. *How much does a custom keyboard cost?* 2021. URL: `https://switchandclick.com/what-is-the-average-cost-for-a-custom-mechanical-keyboard/`.

[2] Shubham Munde. "Mechanical Keyboard Market Report". In: *Market Research Future* (2020).

[3] L. Sadeeshkumar H. Laroche C. Odell D. Rempel Pereira A. Lee D. *The Effect of Keyboard Key Spacing on Typing Speed, Error, Usability, and Biomechanics: Part 1. Human Factors.* 2013. URL: `https://doi.org/10.1177/0018720812465005`.

[4] Laroche Charles Rempel David Pereira Anna Hsieh Chih-Ming. *The Effect of Keyboard Key Spacing on Typing Speed, Error, Usability, and Biomechanics, Part 2: Vertical Spacing. Human Factors.*

[5] *The early years of custom keyboards.* 2020. URL: `https://kbd.news/The-early-years-of-custom-keyboards-1440.html`.

[6] *Wireless Charging 10mm Coil.* URL: `https://www.amazon.com/Wireless-Charger-Transmitter-Module-Output/dp/B08CVGYDJP/ref=sr_1_4crid=1UKLC2GBZRNXW&keywords=smallest%2Bwireless%2Bcharging%2Bcoil&qid=1680533303&s=electronics&sprefix=smallest%2Bwireless%2Bcharging%2Bcoil%2Celectronics%2C88&sr=1-4&th=1`.

[7] *ZMK power Profiler.* URL: `https://zmk.dev/power-profiler`.

Table 1: Bill of materials

Description #	Manufacturer	Quantity	Qty used in Product	Cost @	Total
Seeed Studio XIAO nRF52840	Digikey	20	16	$9.59	$153.44
Flat Coil Inductors	Digikey	40	4	$2	$80
LiPo Batteries (x4)	Amazon	4	4	$18.99	$75.96
Hot Swap PCB	Digikey	20	16	$1.30	$26
Microcontrollers	Ben Sun and Zhejia Yang	3	3	$0	$0
Wireless Charging Kit 10mm Coil	Amazon	2	1	$12.71	$25.42
Inductive Charging Kit (Part 1407)	Adafruit	5	0	$9.95	$49.75
3D Housing	RoboClub	16	16	$0	$0
Low Profile Switches	Amazon	20	20	$1.10	$22.00
3M Dual Lock Velcro Rolls	Ben Sun	2	1	$0	$0
Circuit Components	220 Lab, Tech Spark	many	none	$0	$0
					$432.57

Table 2: Mass Production Price of Materials for 1000 Keys Per 16 Keys

Description #	Manufacturer	Quantity	Cost @	Total
Seeed Studio XIAO nRF52840	Digikey	16	$9.59	$153.44
Arduino Nano 33 BLE	Arduino	2	$12	$24
Flat Coil Inductors	Digikey	32	$1.62	$25.60
LiPo Batteries	Amazon	16	$1.00	$16.00
Hot Swap PCB	JLCPCB	16	$0.0125	$0.02
Wireless Charging (includes component costs and chip fab)	JLCPCB	16	$1.96	$31.39
3D Housing	Xometry	43g	$0.025	$1.07
Low Profile Switches	Amazon	16	$0.094	$1.50
3M Dual Lock Velcro Rolls	Ben Sun	1	$0	$0
Assembly	Employee	1	$15	$15
				$288.19

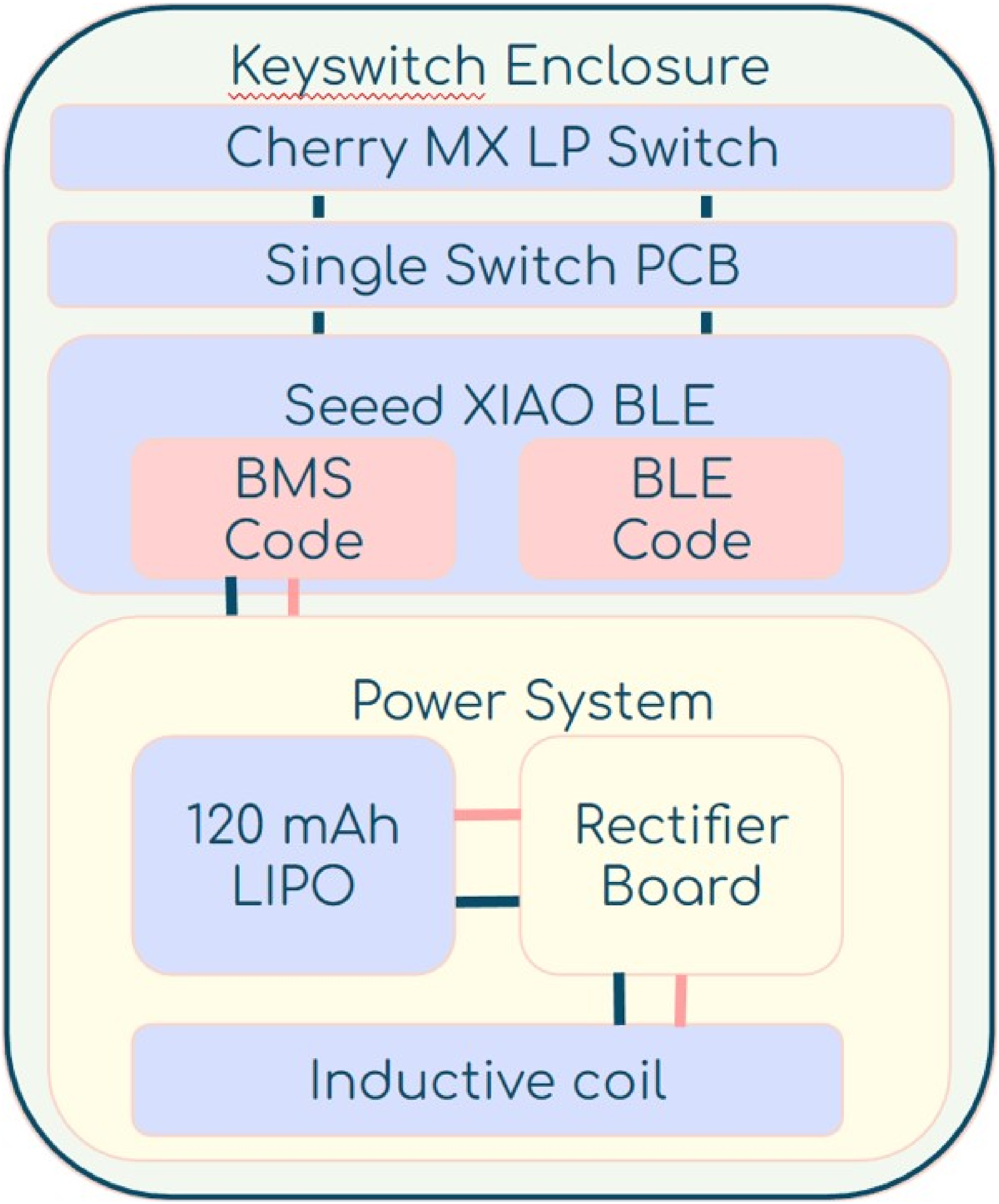

Figure 10: A full-page version of the same system block diagram as depicted earlier.

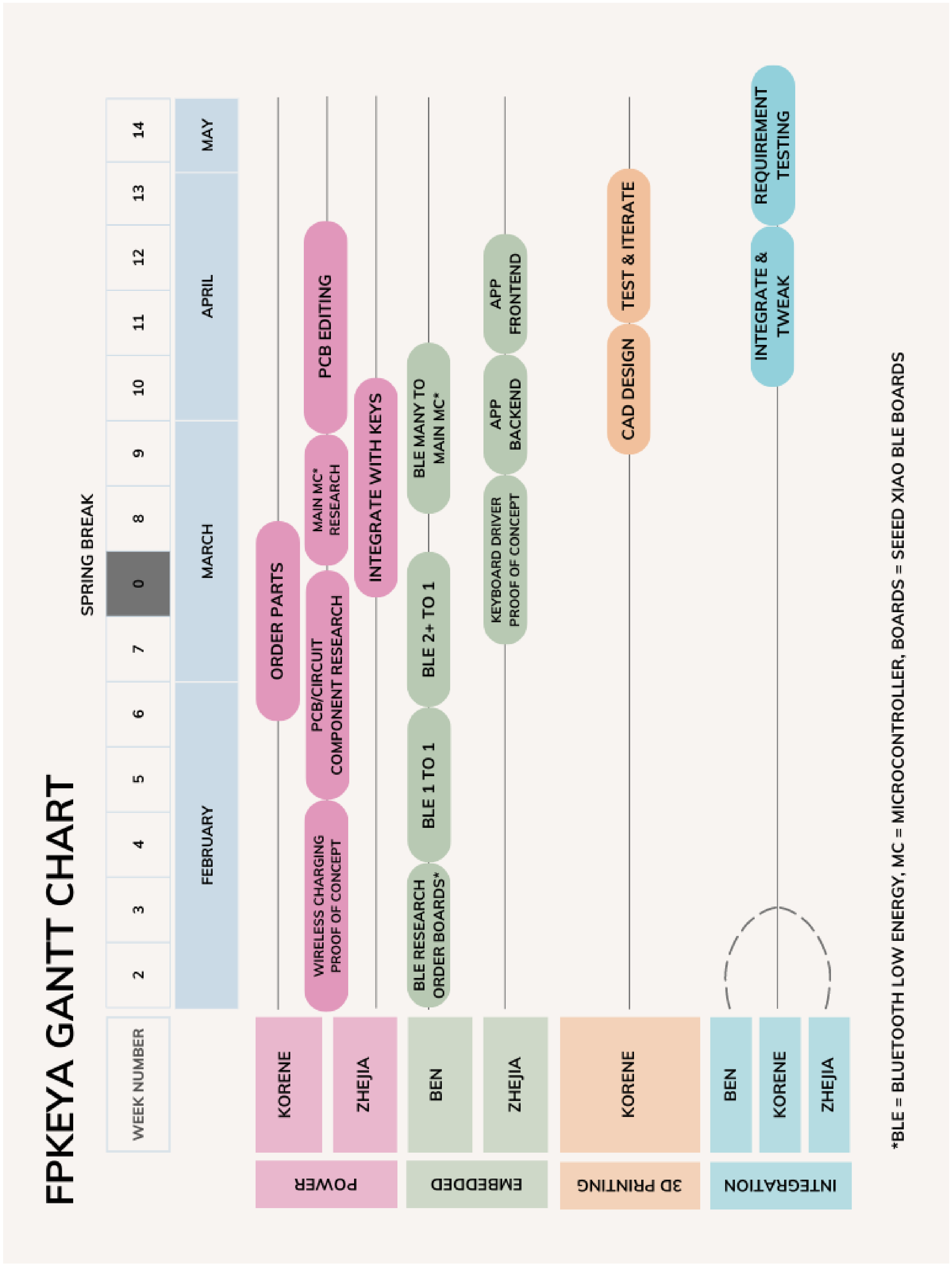

Figure 11: Gantt Chart