# Slate: A Multi-Input Wireless Macro Keypad

Diego Agudelo, Andhres Bolano-Melendez, Samuel Chodur, Jacob Goodman

### Dept. of Electrical and Computer Engineering, University of Central Florida, Orlando, Florida, 32816-2450, USA

### *Abstract* — We present the design methodology employed for the creation of an electronic device for use as a multi-input wireless-capable macro keypad that combines a touch-display and multiple physical inputs, all in a compact and robust form factor. The result is expected to provide users with a product that increases productivity using key-bindings and shortcuts that are offered to access the most frequently used features of the software being utilized. This solution is meant to reduce the complexities and learning curve associated with the process of implementing these shortcuts by offering 16 discrete physical inputs and customizable shortcut access screen which can also contain status information.

### *Index Terms* — macro keypad, touchscreen devices

I. Introduction

Multi-tasking during device use is being more and more prevalent. A single user may not be expected to accomplish the same amount of work that was accomplished by a multiple member team. As an example, production of a live television broadcast could be made up of the on-screen personality, a camera operator, and an audio engineer. With the increase in popularity of live streaming of events, being able to accomplish all these job descriptions as a single person is sometimes necessary or desired. There are many other such examples when it comes to creating quality artistic productions with minimal time and team members and the market for products that aim to solve these problems is growing at a high rate.

A solution to the time and personnel problem is providing the essential functions of multiple disciplines at the user’s fingertips. There is currently a niche market made up of a few companies that aim to provide live streamers, content creators and other artistic disciplines a solution to their problems. We decided to target this rapidly expanding market with a new device solution that provides customizability and portability while still being able to interface with a wide variety of software. We will now provide a more detailed description of the project to include our market analysis, our product’s requirement specifications, and a high-level overview of both our hardware and software design.

II. project goals

The goal of this project is to have a fully functioning customizable multi-input wireless macro keypad that combines a touch-display with multiple physical inputs to provide macro functionality to the user in a compact and robust form-factor. Slate would allow for the user to configure key data to be displayed back on the built-in display. These features allow for the user to navigate their device in a more efficient way and monitor important aspects of their work without information being lost on their main output device.

The design of Slate will be a mixture between portability and functionality, while still maintaining an aesthetic that easily blends into the desktop of most users. We want the users of Slate to be proud of their device. It should be a device that catches the eye of all types of computer users when perusing the shelves or scanning digital advertisements and reviews. Slate aims to give users additional customization, choice, and flexibility they did not know they could benefit from.

The intuitiveness of the device must be clear from the first use. We hope to avoid a steep learning curve by making the most basic and useful functions immediately apparent whilst still providing the customizability sought after by the power user. Ideally, Slate becomes as important of a peripheral to the power user as their mouse and keyboard.

III. System overview

A. Hardware

A diagram representing the hardware system is shown in Figure 1. The hardware for the design of Slate consists of six major components which include the battery management IC, the voltage regulator, the microcontroller, the screen, the analog inputs, and NOR flash memory, all of which were selected based on compatibility with one another and the engineering specifications.

A brief overview of the hardware system is as follows: the power for the system comes from either USB or the battery internal to the system. A battery management IC will select the power source for the rest of the system. The possible selections include USB (VBUS) or the battery. Whatever the selection, it will be regulated to 3.3V by a voltage regular. This 3.3V is then fed to the rest of the system’s components which include the MCU, screen, analog inputs, and the NOR flash unit.

B. Software

There are two major software components required for the system: the configuration software and the system software. The configuration software runs on a user’s host device, while the system software is executed by the MCU onboard the system itself. Both pieces of software are written using Python 3. This choice was made amongst other programming language options because Python provided the quickest path to prototyping and testing an actual code base. This quickest to a usable system was necessary to meet the timeline requirements of Senior Design.

At this time, the configuration of Slate must be accomplished through the configuration software which runs on a user’s host machine and not on Slate itself. Once the configuration is loaded onto the system’s memory, it will remain there until the configuration is modified again by the user. Figure 2 shows what the user will be presented with when first opening the configuration software.

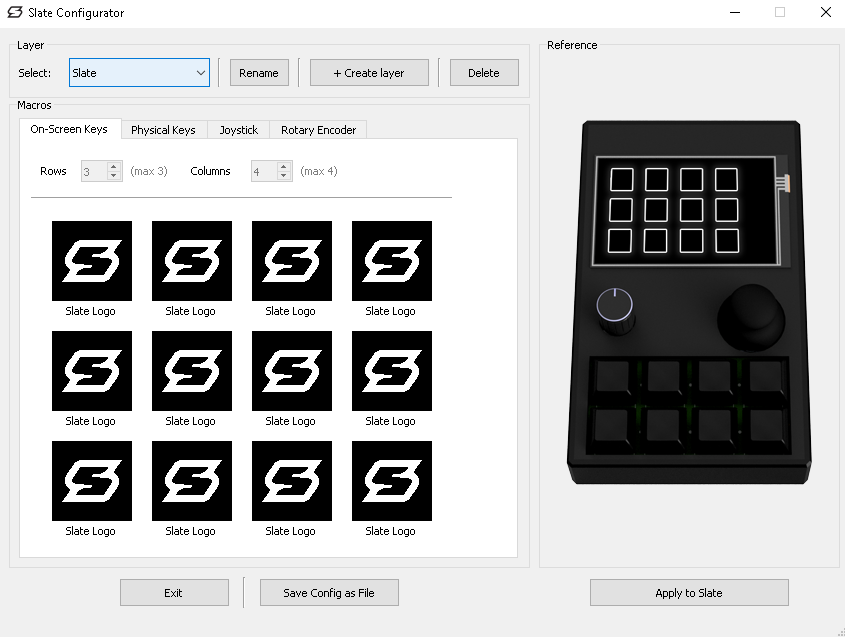


Fig. 2. A screenshot of the configuration software for the system.

The software that runs on the system provides the interface between the user’s inputs, and host system Slate is connected to. At this time, the system is only meant to be used with the Microsoft Windows operating system. When a user of Slate engages an analog or screen input, the desired result will be executed on the host system by Slate. A block diagram depicting how the system software prepares to enter its main loop is shown in Figure 3.

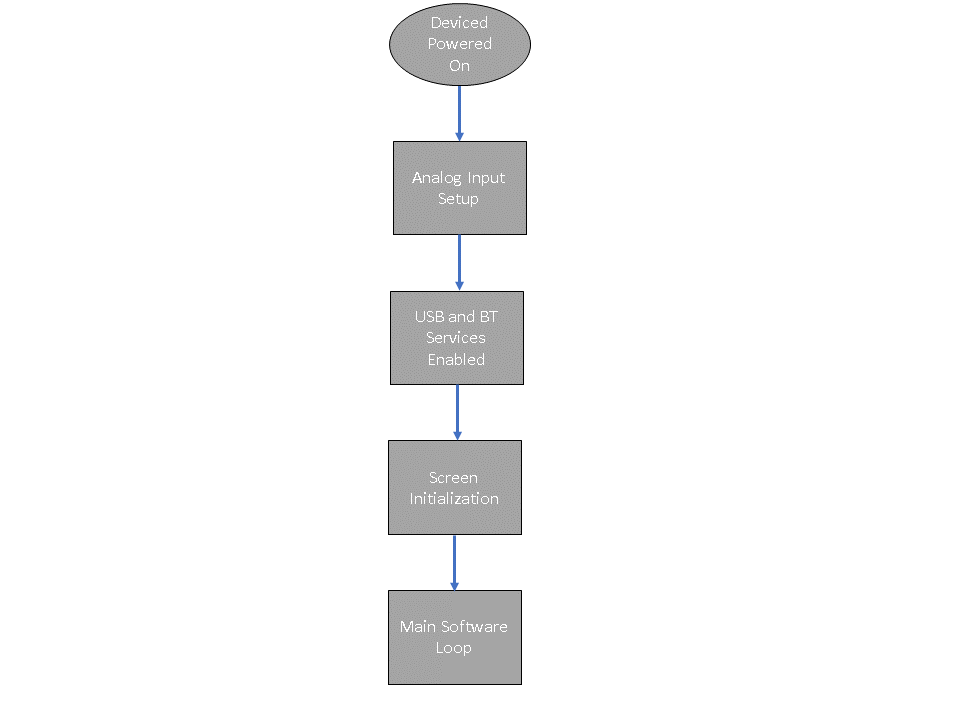


Fig. 3. Diagram depicting the flow of events after Slate is powered on.

IV. system components

Technical insight into the main components of the system will now be discussed.

*A.* Microcontroll*er*

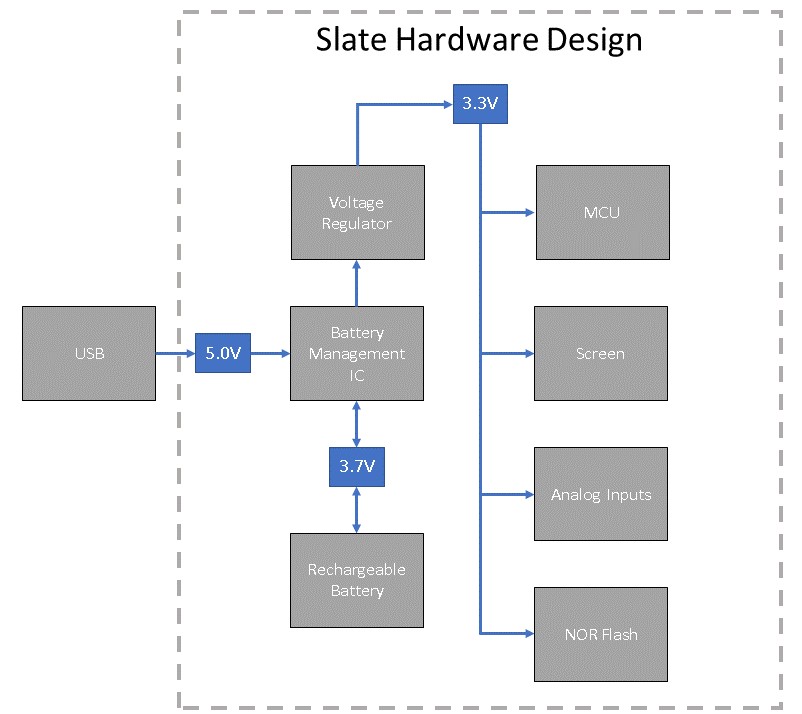


Fig. . A block diagram of Slate’s hardware design

The microcontroller utilized by the system is the Nordic Semiconductor nRF52840. At the time of this writing, it is a relatively new microcontroller which first appeared on the market in 2018. Many of the newest and currently ongoing keyboard-related hardware projects are using this chip due to its onboard wireless capability, powerful processor, and sizeable memory space.

The nRF52840 features a 64Mhz 32-bit ARM Cortex M4F processor, 1MB of flash memory, 256KB RAM, built-in Bluetooth 5.0 and 2.4Ghz Wi-Fi, native full-speed (12Mbps) USB support, NFC Tag-A capability, and on-chip DC-DC buck converter. Due to its ample 1MB of flash storage, this microcontroller is compatible with higher-level programming languages like CircuitPython [2].

B. Touchscreen

The touchscreen within the system was decided upon mostly for budgetary reasons and known compatibility with the nRF52840 microcontroller, although six different displays were investigated. After investigation, the decided upon screen was the Adafruit 3.5” FeatherWing. The FeatherWing utilizes the HX8357 driver chip and includes a dedicated touchscreen controller, STMPE811. It has a resolution of 480x320 and operates at 3.3V for logic and 5.0V for the backlight. Communication between the microcontroller and the touchscreen is accomplished via SPI.

C. Charge Management Controller

The charge management controller selected for use within the system is Microchip’s MCP73831. This linear charge management controller is meant for space-limited, applications and typically those that are cost-sensitive and meant to operate with Lithium-Ion and Lithium-Polymer batteries.

The primary considerations that led to this design decision were the price, footprint, package, and availability. We found the greatest weakness of our selection to be the small charge current which will lead to longer battery charge time but found this tradeoff acceptable as the system’s engineering requirements regarding charge time would still be met.

D. Voltage Regulator

The selected voltage regulator for the system is the AP7365 which is manufactured by Diodes Incorporated. This low dropout linear regulator was selected because of its low dropout voltage characteristics, which make it suitable for low and medium power applications, especially battery powered devices and consumer electronics.

E. NOR Flash

The system’s NOR Flash memory chip is the GD25Q16C from GigaDevice Semiconductor. The most important characteristics of this component are the capacity of 16Mbit, its support of Serial Peripheral Interface (SPI) and its single power supply full voltage range of 2.7~3.6V.

F. Encoder

The selected rotary encoder for the system is Bournes’ PEC11R Series 12m. This model was decided upon after comparing the notable specifications and features that were targeted as having the most weight on meeting the engineering requirements for Slate.

The aspects considered for selection were encoder size, shaft type, shaft length, shaft diameter, total dimensions and whether the encoder included a switch.

G. Thumb Joystick

The selected joystick for the system is SparkFun’s thumb joystick accessory, COM-09032. This selection was made after comparing specifications and features of several possible options. Those features that were identified as being critical to meeting the engineering specifications were the number of pins, the profile, whether a connector was required or not, and whether the joystick could act as a switch.

H. Keyboard Switch

Our system is equipped with 8 switches that can be mapped to several different types of macro actions. When choosing the type of switch for our system we considered our environment, tactile preference, target hand size, strength, typing style, and the general purpose of the keyboard. Tactile preference depends ultimately on the user. The general rule of thumb is that for gaming linear switches are what should be bought since they are consider speed switches with a smooth and consistent keypress which helps with rapid movements. For general typing tactile and “clicky” switches are preferred because it provides great feedback which results in a pleasant user experience. A tactile actuation type was desired to help assist the end-user in feedback for when a macro is pressed while also providing the tactile bump that helps reduce clicking errors. The selected tactile switches were the Durock T1 switches.

V. Hardware design details

The hardware design details of the most important aspects of the system will now be discussed. The discussion will be organized by the primary functions of the design and can be visualized by the provided images of the schematic.

A. MCU and BT Module

The design for the MCU/BT module is shown in Figure 4. The naming convention used for the nets that are connected to GPIO pins was discussed and agreed upon by members of the project team. The datasheet and similar designs that incorporate the MDBT50 from Raytac were referenced heavily to aid in the development of this portion of the schematic.

Examples of the naming conventions used for the nets included in the overall schematic are SW\_ENC and SW\_JOY. These nets are associated with the switches for both the encoder circuit and the joystick circuit respectively. Similarly, the nets that begin with QSPI are those that need to be connected to the memory module that utilizes QSPI for serial communication.

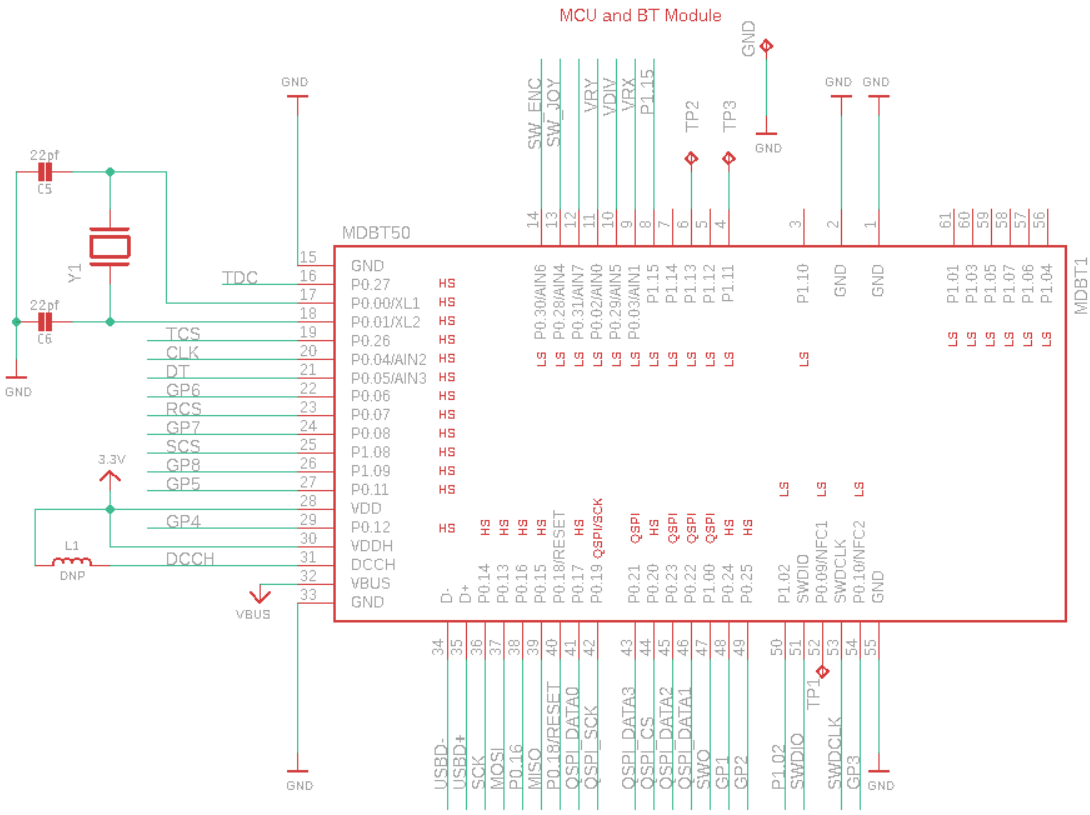


Fig. 4. MCU/BT module schematic design

B. Power and Filtering

The design for the power and filtering circuit is centered around three major components: a P-channel MOSFET, a Schottky barrier diode, and an LDO voltage regulator. The diode labeled ‘D2’ will allow for the circuit to choose whether to use USB (VBUS) or the battery (VBAT) as its voltage source. A pull-up resistor is used on the ENABLE pin of the IC to activate voltage regulation.

The result of the circuit is a regulated 3.3V output that is used to provide voltage to the rest of the system. The result of the circuit is a regulated 3.3V output that is used to provide voltage to the rest of the system. A switch is implemented to turn the product on/off the product, the enable pin on the low drop linear regulator will be utilized to accomplish this, bypassing the pullup resistor to ground.

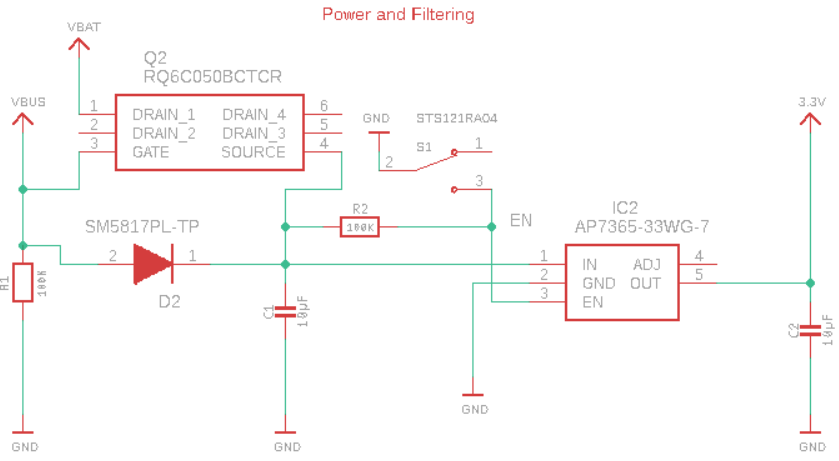


Fig. 5. The system’s power and filtering circuit

C. LIPO Charging

The circuits responsible for the LIPO battery charging and monitoring are shown in Figure 6. The STAT pin of the LIPO charger is left disconnected as there will be no LED indicator on the final PCB design of Slate.

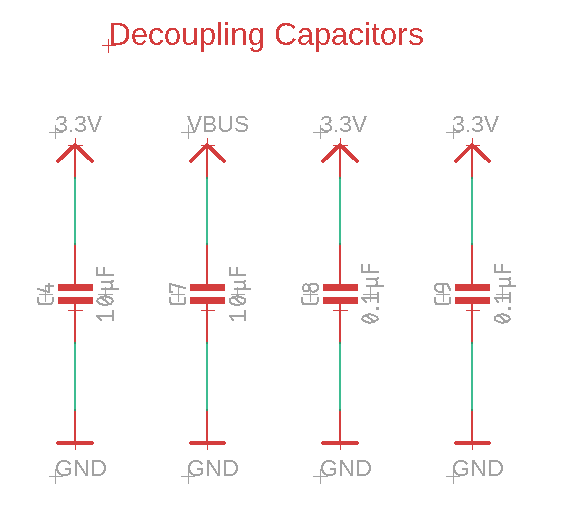


Fig. 7. Decoupling capacitors used in system’s design

The LIPO monitoring circuit’s voltage divider will be connected to a GPIO pin of the MCU to monitor the battery percentage. The possibility of displaying the current battery charge percentage on Slate’s user interface will be explored as software is further developed as the design of Slate continues.

From the datasheet of the MCP73831 LIPO charging IC, we know that VDD needs to be between 3.75V and 6.0V, which is satisfied by the provided voltage from USB (VBUS). The temperature range of the LIPO charging IC easily fall within the range of operating temperatures specified by the engineering requirements for Slate. Testing will be necessary to ensure the temperature within Slate’s enclosure falls within these temperatures even during times of heavy load.

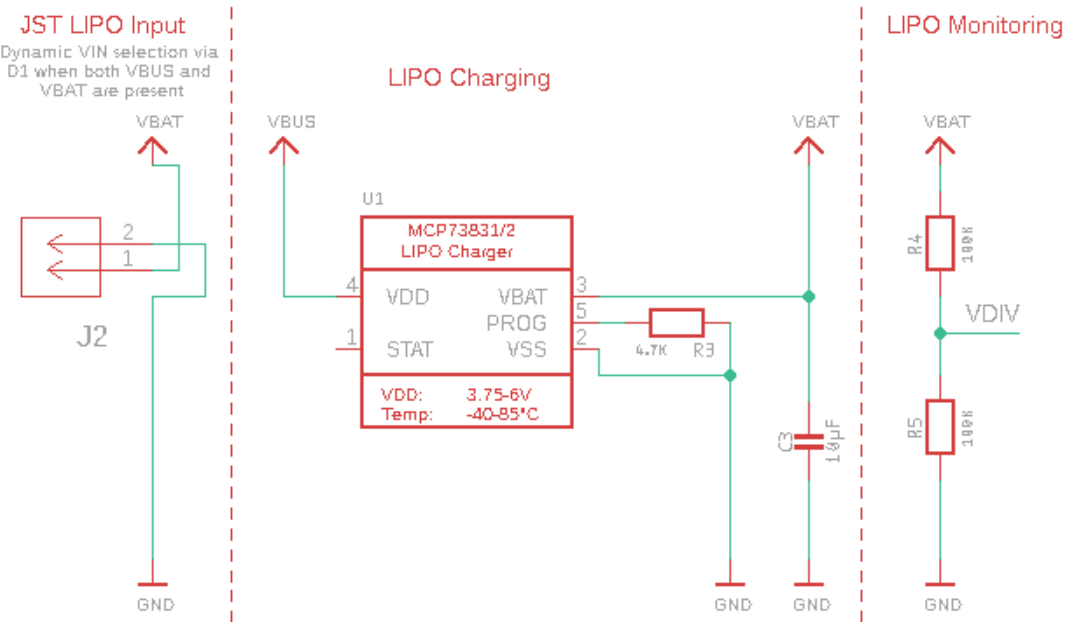


Fig. 6. LIPO charging and monitoring schematic design

D. Decoupling Capacitors

Several capacitors are used and placed near the ICs that are included in our design and are shown in Figure 7. These capacitors act as shunts against noise caused by the other components in the design. The capacitors are normally located between power and power return conductors. This location helps stabilize the voltage provided to any active digital devices [1].

Attempting to review literature on the best location for decoupling capacitors, we found conflicting information. There are several options normally given: 1) near the device meant for decoupling, 2) location doesn’t matter, 3) connected directly to device via short traces, 4) traces shouldn’t be used with decoupling capacitors.

Because of all this conflicting information, the strategy used for the system was based on the prototype used for software development. We inspected the board design of the known working prototype and followed their design methodology for location of the decoupling capacitors.

VI. Software design details

We will now provide an overview of the system’s software design. As previously discussed, there are two main software components required for the realization of the system: configuration software and system software.

*A. Slate System Software Functionality*

The Block diagram shown in Fig. 8 shows the basic flow for the software that runs on Slate. When the user turns Slate on, it begins to import libraries and initialize peripheral components, HID services for USB and Bluetooth, define the layout of the UI, and loads the first layer before entering the main loop. Once in the main loop, Slate will check battery status and host connection. If no host is detected, a connect screen will overlay the interface and wait until USB or Bluetooth is connected. If a host is connected, the Slate proceeds to check for user input and runs the macro or function assigned to any active input. This loop continues until the device is powered off, or a new config is written.

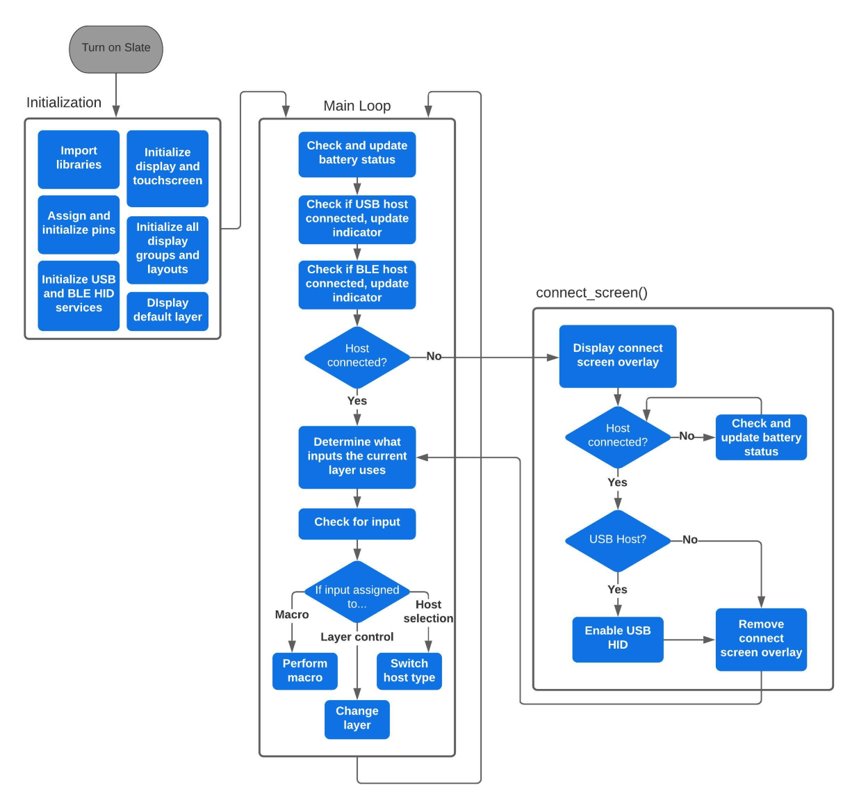


Fig. 8. Slate Software block diagram

B. Configurator Functionality

The Block diagram shown in Fig. 9 shows the flow the configurator application has when a user adjusts layers (layers are profiles, for the purposes of this project we are calling them layers) and eventually apply these layer customizations to the Slate.

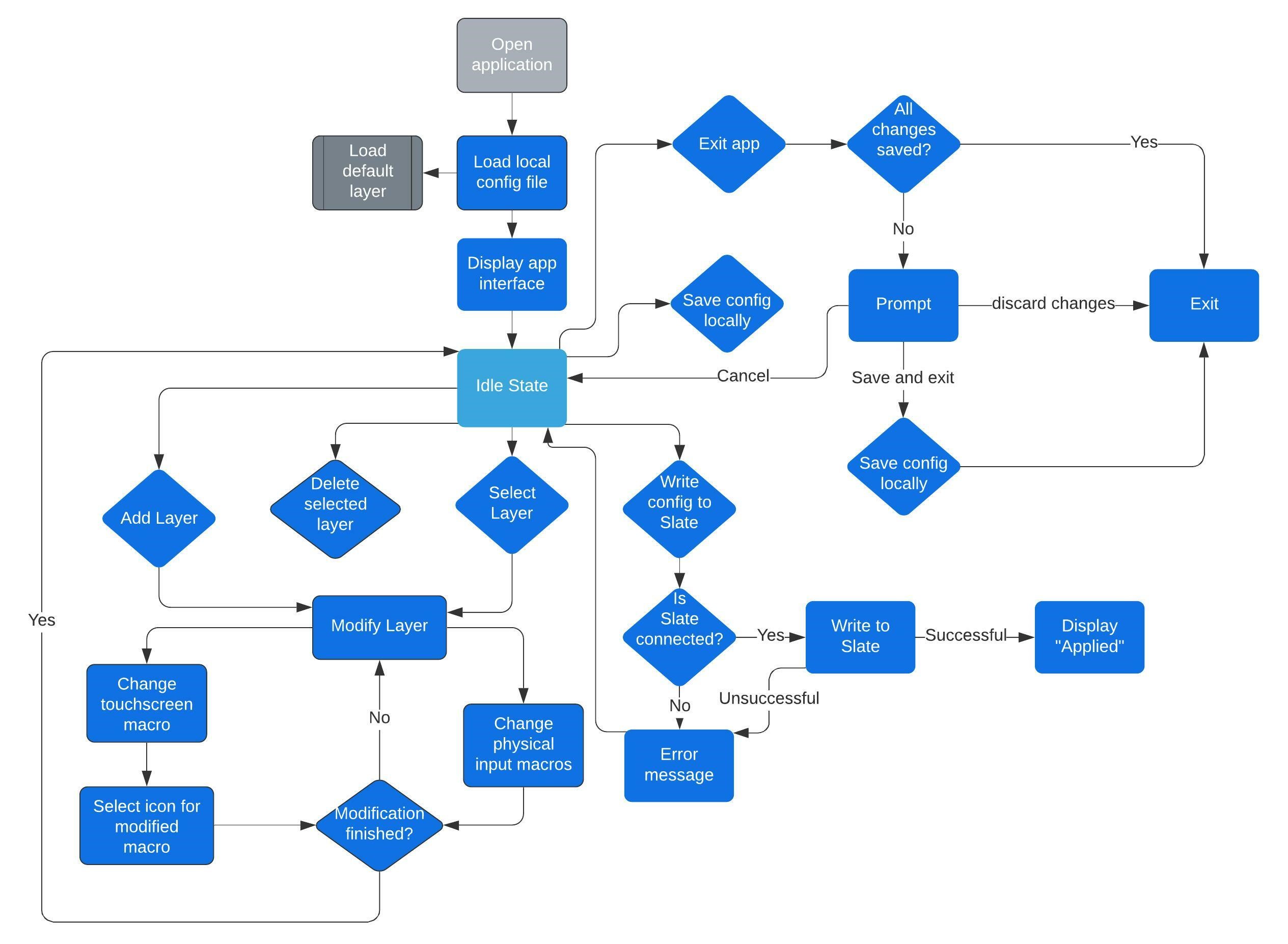


Fig. 9. Configurator block diagram

Initially the configurator loads a default config file called layers.py that contains three different layers that act as a template for the user to modify. All the set macros for each layer are loaded into the app interface. Here the user can select one of the layers and either modify it to meet their use case or delete the layer.

When modifying a profile, you can either configure a physical macro which consists of key switches, an encoder, and the joystick or a touchscreen macro to your desired function. It lists ten physical and twelve digital macros for the user to modify as shown in Fig. 2. If the user doesn’t finish editing a profile they can simply save the state of their changes by clicking save to config and continue from that point when they open the application again. When the user clicks done, it saves the profile you selected or created with specified macro settings on the list and goes back to the main interface.

The configurator app also allows the user to either load the entire list of layers back to the Slate or for the user to select one specific layer and load it onto the Slate. The user can also set one of the listed profiles as the default profile, which then will be used as the initial boot profile when the Slate is turned on. The program then overwrites the Slate’s defaulted layer but not the generic profile used when the Slate is first initialized.

Since the on-board Slate software and the config file is written in Python, the configurator app is also written in Python to maintain consistency and minimize extra time needed to familiarize with a different language.t. This loop continues until the device is powered off, or a new config is written.

VII. Prototyping

The prototype for the system is based around Adafruit’s nRF52840 Feather Express and is shown in Figure 10. Our prototype served as an initial model so we could test our hardware selection and software methodology prior to the manufacturing and assembly of the system’s PCB. This prototype is considered fully functional, as the parts used for the user interface (screen and analog inputs) are full-scale and match what would be used in the final version of the design [3].

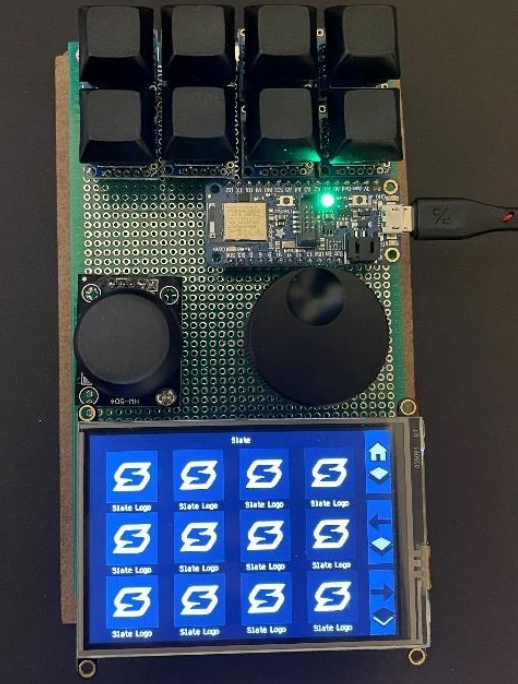


Fig. 10. Prototype for testing the system and software development

The full-scale prototype was also used for testing any possible design problems and allowed for any modifications to the schematic and PCB design to be accomplished before manufacturing and assembly. It was also used so different positions of components could be tested and analyzed. The assembly that was required for this initial prototype was wiring components by hand to the protoboard.

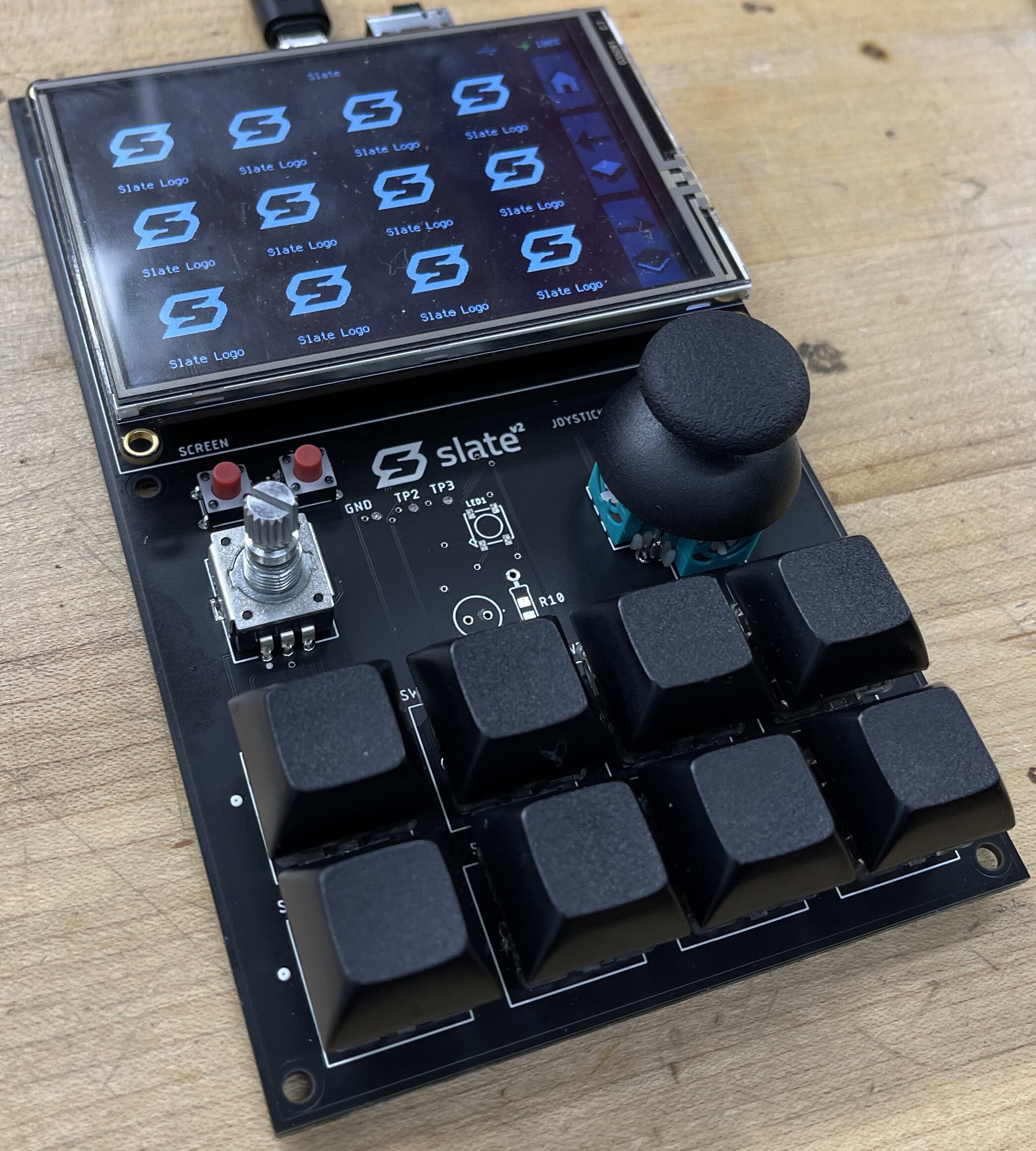


Fig. 12. Final revision of the custom PCB assembled and powered on

VIII. PCB Design and assembly

Following prototyping and the design of the PCB, the first version of the custom PCB was manufactured and assembled. After troubleshooting, initial issues (mostly related to USB data transfer) with the first version of the board design were identified and corrected in the second version of the PCB shown in Figure 11 [4].

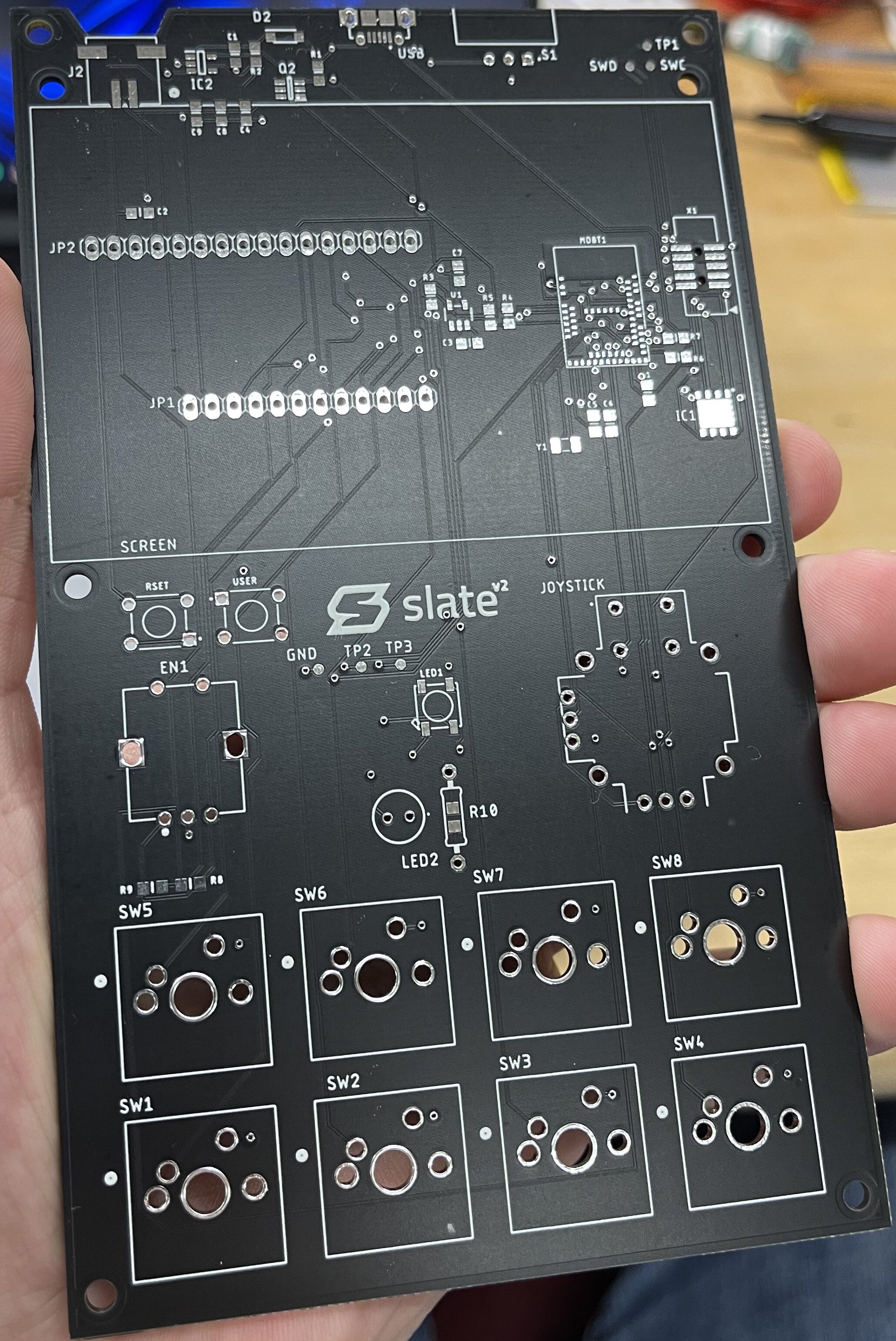


Fig. 11. Final revision of the custom PCB designed and manufactured for the system

Surface mount components of the PCB were assembled in the Senior Design lab at UCF utilizing a NeoDen IN6 reflow oven.

IX. Conclusion

Senior Design at UCF offered the opportunity for the Slate project team to practice the engineering skills which we have amassed over the course of our time as students at UCF. We learned several skills and concerns related to the field of engineering to include the history of engineering education, engineering as a vocation, engineering management, practical design constraints, standards-based design, engineering research, engineering ethical responsibility, and engineering economics.

Our project, Slate, allowed us to demonstrate the ability to identify, compose and resolve engineering problems. Studying the problem, we were able to develop several design alternatives and selected an approach that could be implemented. We performed the necessary parts acquisition, hardware and software realization and debugging necessary to complete both the project and the requirements for the Senior Design course. The completion of our project afforded us the opportunity to understand the impact of engineering solutions on a global, economic environmental and societal context.

Acknowledgement

The authors wish to acknowledge the assistance and support of Dr. Samuel Richie, Dr. Lei Wei, and Dr. Aravinda Kar throughout our time as students at UCF, especially during our last two semesters of Senior Design. We would also like to thank Quality Manufacturing Services, Inc. for their aid with the assembly and troubleshooting of the first version of our PCB. We would like to express our most sincere gratitude to all the instructors and staff we had the opportunity to learn from and work with during our time at UCF. Finally, we are especially grateful for our family and friends for their patience and understanding throughout our journeys to become college graduates.

biographies

Diego Agudelo is a senior design student expecting his BSCE form the University of Central Florida in December 2021. After graduation he will be pursuing a career in computer hardware in an undecided company. As with that he will be further expanding and growing his custom automotive lighting business. He plans to obtain his MSCpE after working and finishing his ongoing projects.

A person taking a selfie

Description automatically generatedAndhres Bolano-Melendez is a senior design student expecting his BSCE from the University of Central Florida in December 2021. After graduation he is pursuing a career as a Java/Python developer at IBM and possibly coming back to UCF to pursue a MSCpE.

Samuel J. Chodur, Jr. is a senior student expecting his BSEE from the University of Central Florida in December 2021. He presently works for Leidos, Inc. as a Data Analyst supporting the United States Prompt Diagnostic System and the United States National Data Center at the Air Force Technical Applications Center. He plans to obtain his MSCpE after the birth of his child in December 2021 and continue pursuing his interests in the fields of software development, data analysis and machine learning.

Jacob Goodman is a senior design student expecting his BSCE from the University of Central Florida in December 2021. After graduation he will be joining Intel Corporation in Folsom, CA as a technical marketing engineer, specializing in enterprise SSD products. As he continues his career, he is considering pursuing higher management in technology and returning to a university to complete an MBA.

References

[1] T. Hubing, "Effective strategies for choosing and locating printed circuit board decoupling capacitors," *2005 International Symposium on Electromagnetic Compatibility,* 2005. EMC 2005., 2005, pp. 632-637, doi: 10.1109/ISEMC.2005.1513591.

[2] Fisher, D. , Fletcher, R. and Anapalli, S. (2021) Python Software Integrates with Microcontrollers and Electronic Hardware to Ease Development for Open-Source Research and Scientific Applications*. Advances in Internet of Things*, 11, 42-58. doi: [10.4236/ait.2021.111004](https://doi.org/10.4236/ait.2021.111004).

[3] Farrier, B. (1998). UXL encyclopedia of science. *School Library Journal, 44(5)*, 173. Retrieved from <https://www.proquest.com/trade-journals/uxl-encyclopedia-science/docview/211699329/se-2?accountid=10003>

[4] USB Implementers Forum. (2000) USB 2.0 Specification. Retrieved from https://www.usb.org/document-library/usb-20-specification