Beta Build – Prototroller

Britton M., Caleb O., Evan Z., Merrick R., Yu-yang H.

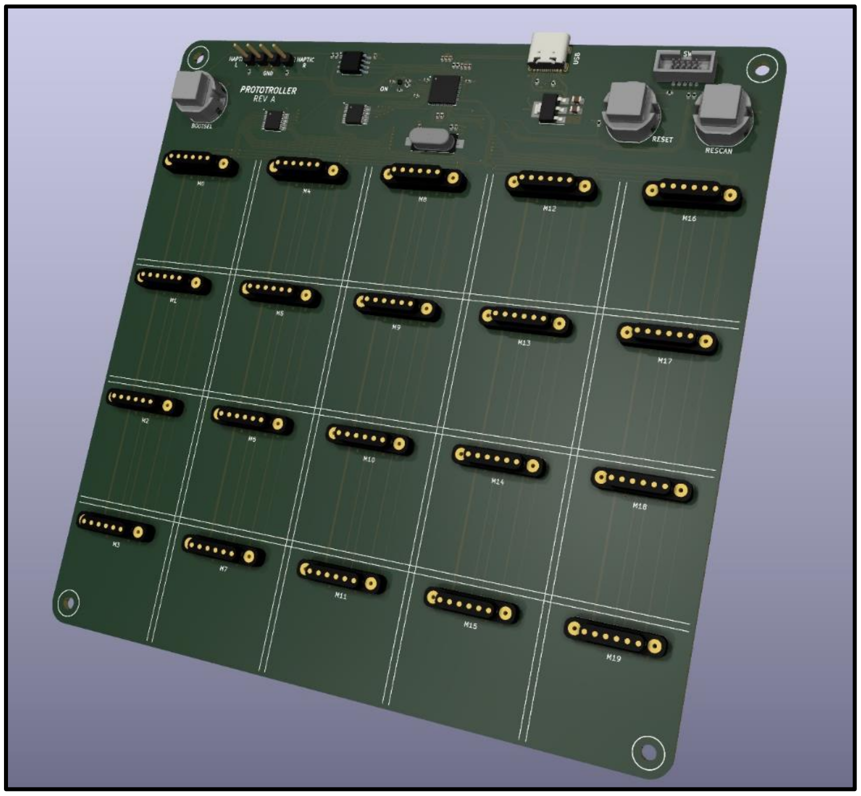

Figure 1: Completed PCB Prototroller Master Board.

Usability

Interface

Users interact with modules, placeable anywhere on a 4x5 "Protogrid" through magnetic pogo-pin connections. Data is forwarded from modules to the master board, which drives gamepad HID reports to the host through USB (Universal Serial Bus) 2.0 connection. We have created 3D printable enclosure models for all aspects so the user may grip the Prototroller with ease. Serial logging is available and may be inspected with a program like PuTTY. We have put much effort into schematic design, PCB design, and overall aesthetic design. The modules we currently support is joystick, push-button, tactile switch, slider, XYAB button array, accelerometer/gyroscope, with plans for more. On the master board side, we have several user interfaces: RGB LEDs for power and status indication; large buttons for programming, reset, and module rescanning; USB-C port; haptic feedback headers; and finally, SWD headers for debugging.

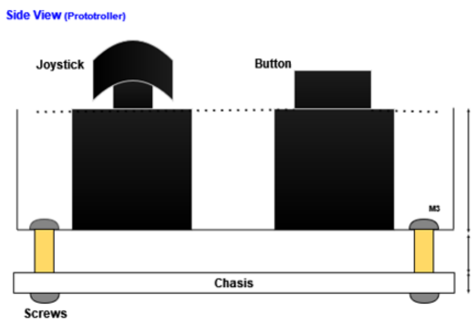

Figure 2: Module interfaces showcasing joystick and push-button.

Navigation

User mechanics are functional, and controls are in a good place to be tested in the Beta Testing phase. All controls, including master board interfaces and module components themselves, were designed to be predictable and easy to use. Features are discoverable through testing and especially through placement of swappable modules. Navigation comes naturally to both those who have never used a gamepad before and experienced gamers.

Perception

The Prototroller is highly intuitive. Simply plug in the desired module in the desired slot, and interact with it, indicating changes in application by reflecting a user input on the connected host. If a module is not connected correctly, or a module is inserted and no rescan occurred, the module will be in "disconnected" state (interaction proves futile). The state of connected modules can be observed by serial output, but if a module is unresponsive the user may initiate a rescan or reset. Some sensory experiences are haptic tactile feedback or audible responses from buzzers integrated into the chassis. All outputs would be defined by the connected computer or the firmware running on the controller.

Responsiveness

We use high frequency 12MHz crystals on both master and module boards. This clock speed enables sufficient controller responsiveness, from the time a user interacts with a module to the time an effect is seen on the host. However, we will continue to optimize the system for minimal latency by utilizing PLLs. Due to the unique notion of modularity, optimizing latency for all modules proves to be difficult. Our goal is to match modern controller responsiveness of around 4ms.

Build Quality

Robustness

Modules will conjoin with the chassis using magnetic pogo-pin connectors, making the event of modules falling out unlikely. The pogo pins - in addition to a 3D printed support, will be used to firmly secure the module to the board whilst simultaneously making it just as difficult to remove under different circumstances without the use of hands.

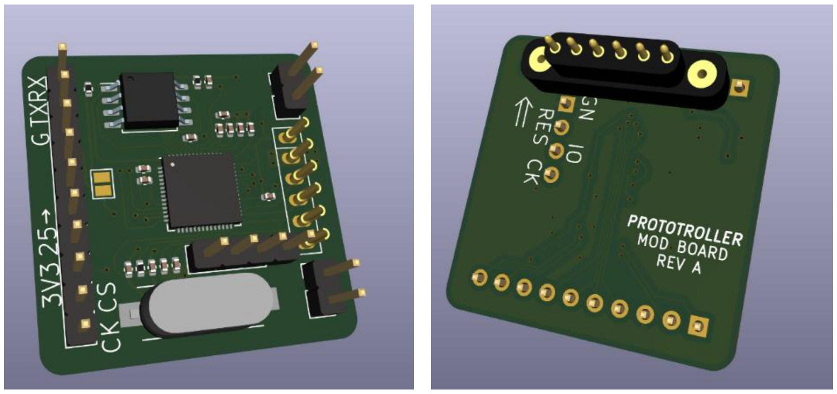

Figure 3: Module PCB Render – Note the magnetic POGO pins for connection to Master board.

The casings (on both chassis and modules) will be engineered to protect against drops from heights from 5 feet and below. The quality of inputs to the master, and hence, the host, is entirely dependent on the quality of connections in the PCB and peripherals used.

In terms of firmware and drivers, the target SPI speed is sufficient to routinely poll all modules with minimal issue. The transactions will send packets of data between the module and master and employ error-checking to determine that the data is formatted correctly. All firmware is designed to be fault-tolerant, that is, able to recover from expected or unexpected faults. Additionally, the user is given the option to reset the master board with a dedicated button. Our experience with the prototype build was that bugs are rare, but more work must be done once the hardware is in.

Our board designs use decoupling capacitors for local energy storage in case of power draw spikes, as well as generic RLC filters to create a barrier against undesirable frequencies in signals. This ensures that the Prototroller is sturdy enough to handle incidental discharge of electricity or potential outages.

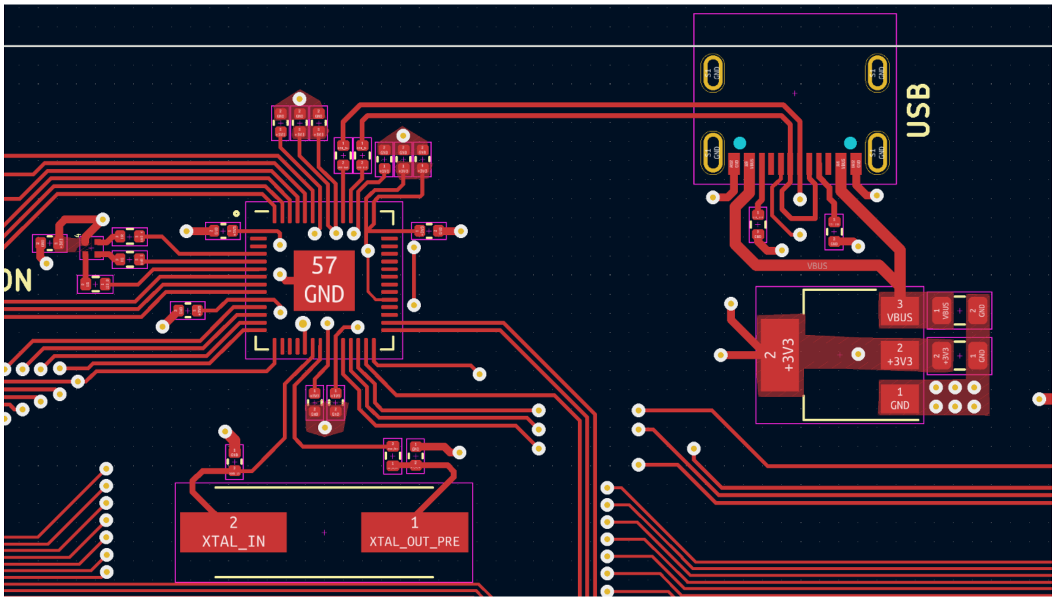

Figure 4: PCB Design for the Prototroller Master Board contains protection circuitry and decoupling capacitors both located at USB power rails and primary power pins of the RP2040 IC.

Consistency

The Prototroller is a real-time predictable system that should not have any issues with inconsistent inputs. Our SPI data transmissions and HID reports will contain predictable information with a set buffer size. The system will be connected through USB and shall not draw more than 500mA through the port. A rescan button will be visible and pushing it will trigger a module rescan on all slots. Doing so will have all modules synced up with the master board. A reset button will restart the currently flashed program for the respective module. All modules will have a respective output in respect to their input due to their simple functionality and should consistently yield the same results – barring hardware failures that cannot be controlled.

Aesthetic Rigor

The design allows for inputs to be handled simultaneously (i.e., with both hands) to give the same feeling of using a controller. The chassis will be assembled in an ergonomic manner to feel akin to using modern controllers. Modules mount securely to their own physically secure enclosures. For future builds we plan to incorporate grips to complete the gamepad aesthetic. Overall, there is sufficient integration of all assets to be functional, easy to use, and stable.

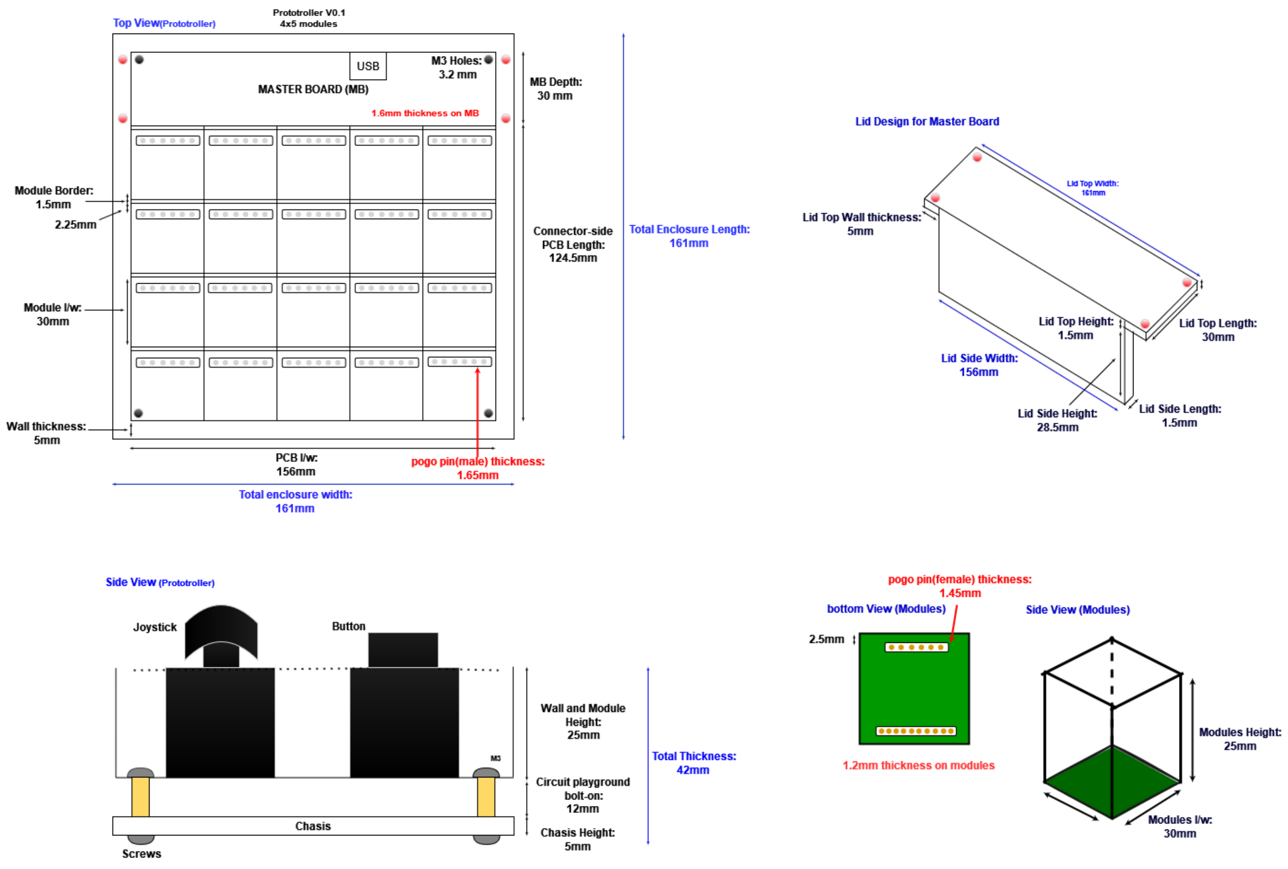

Figure 5: Mockup of the overall physical design aesthetics. The master board will be contained within an enclosure ("chassis") intended to replicate the traditional form factor of a game controller, with the module segments all consistent in measurement to allow a dynamic layout. Grips, not pictured, can be attached to each side of the enclosure for ergonomic support and aesthetics.

Vertical Features

External Interface

A major use case is the reconfigurability of modules on the Prototroller chassis. To disconnect, the user may grab a module and pull it away (severing the physical and electrical connections). To connect, the user may insert a module into a slot and press the "rescan" button to update the persistent state. The other major use case is the I/O interaction, from which the user interacts with the mounted component while the module is connected to the chassis. In addition, we order four 12mm (about 0.47 in) circuit playground bolt-on to connect with two screws on top of PCB board and bottom of chassis.

Persistent State

The Prototroller is a real-time device that operates with little persistent state. As we integrate more swappable modules into the project, the host device will keep track of which modules have been inserted in an internal data store. The data store is persistent until the user presses a rescan button, which triggers a routine to check if the physical module configuration has been changed. This routine could also trigger during faults, such as an unresponsive module, to update the data store accordingly. The data store of connected modules is routinely iterated to transmit and receive fresh data to/from the modules over SPI.

Internal Systems

When a module is initially connected, neither the host device nor master device will be aware of the change until the rescan button is pressed. This allows a user to fully set up their controller layout before the host attempts to use it. Additionally, it eliminates the need to continually run the detection routine during normal operation. Once the master detects new modules and retrieves their associated IDs, it will store these objects in an internal data store. The master may then iterate over the connected modules and read/write data at a constant rate. The master will sequentially poll each module using two 4-16 decoders to pull each chip-select low, read/write serial data, and pull the chip-select high. Only one chip-select will be pulled low at a time. Regardless of whether input is registered, data will be transferred between MISO and MOSI lines between the master and slave.

Hardware Design

Schematic Creation

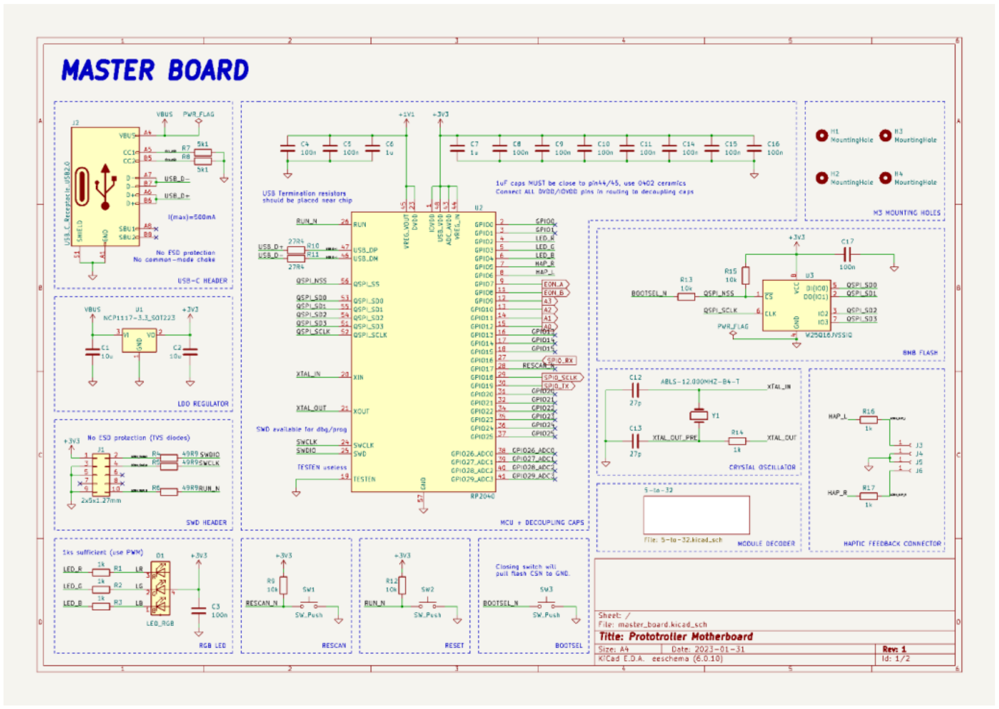

Figure 6: Master board completed schematic.

When creating the first drafts of our board schematics, we utilized the publicly available schematics of the RP2040 Pico and Tiny2040 microcontrollers. In tandem we also referenced the Hardware Design Manual of the RP2040 chip, cross-referencing design decisions and ensuring we adhered to the base requirements, as well as adapted any improvements utilized in boards produced most like ours. Our stakeholder, Carsten, provided lots of great feedback along this design process that helped us push the beta build to the finish line!

Primary components of both board schematics are the crystal oscillator, SWD header, flash chip, and the RP2040. After reviewing and confirming the proper configurations for these components, we appended additional components to each board schematic. For the master board, it was pertinent to include an LDO regulator, USB-C header, as well as decoders, haptics, and various button inputs.

The module board required a different subset of components, most being header pins for use with the Veroboard components that would be mounted on a backpack atop the primary board. As our module boards are generic across all components, we could not include any specific component pieces in the design, and thus opted for a concise number of available pins for interaction with components of our choosing. These boards also required interfacing pins from the base board so they could receive SPI communication and power and ground terminal points.

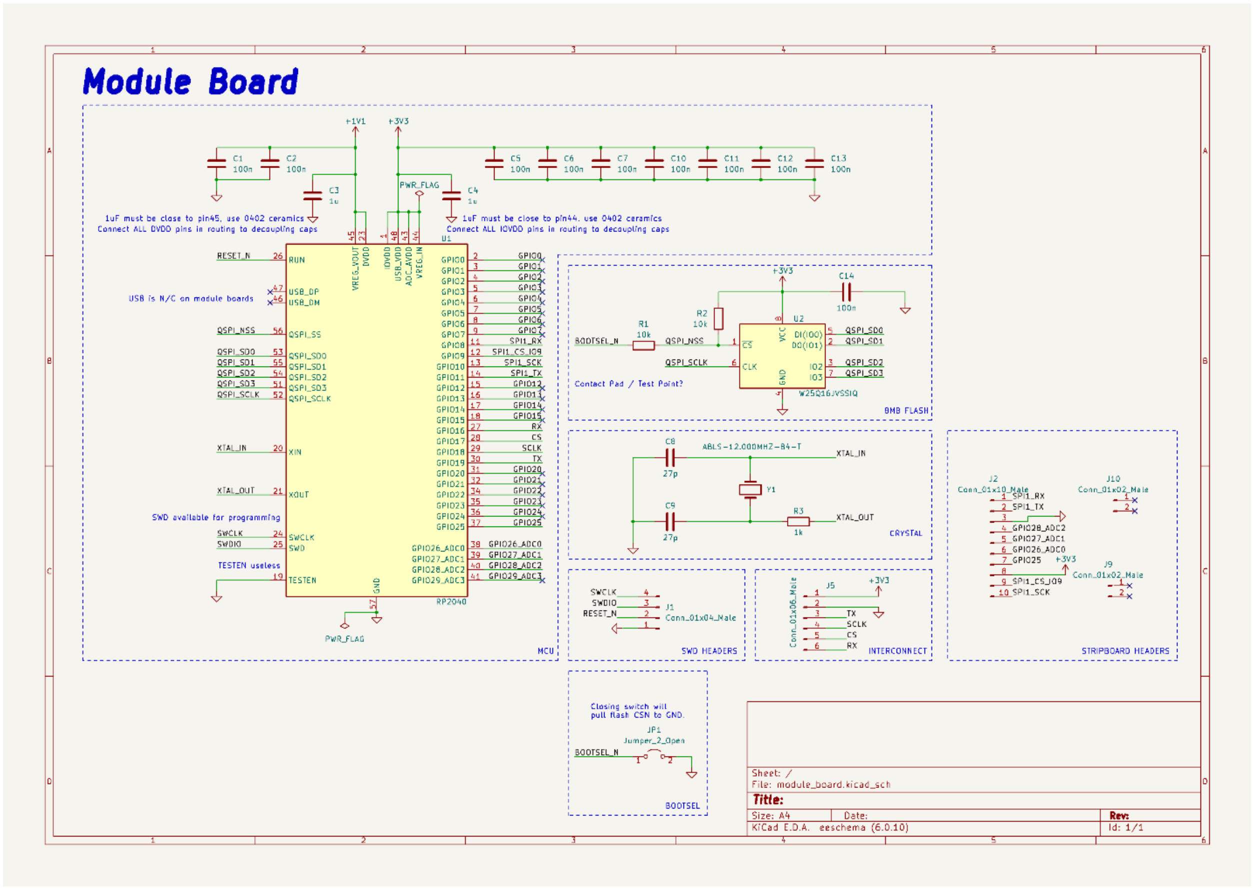
Module Board
+1V1
+3V3
1uF must be close to pin45, use 0402 ceramics
Connect ALL DVDD pins in routing to decoupling caps
1uF must be close to pin44, use 0402 ceramics
Connect ALL IOVDD pins in routing to decoupling caps
USB is N/C on module boards
SWD available for programming
TESTEN useless
RP2040
MCU
8MB FLASH
W25Q16JVSSIQ
Contact Pad / Test Point?
ABLS-12.000MHZ-B4-T
CRYSTAL
SWD HEADERS
INTERCONNECT
STRIPBOARD HEADERS
Closing switch will pull flash CSN to GND.
Jumper_2_Open
BOOTSEL
Sheet: /
File: module_board.kicad_sch
Title:
Size: A4
Date:
Rev:
KiCad E.D.A. eeschema (6.0.10)
Id: 1/1

Figure 7: Completed Prototroller Module Board Schematic

Part Selection & Limitations

To assign the proper footprints to each component in our schematics for use in PCB design, we had to select specific parts and form factors that were affordable, readily available, and constrained to the tighter proportions of our module boards. We created a Bill of Materials (BOM) containing our chosen components for order and confirmed that the clearances of each footprint were within specification for our PCB production facility. We encountered an interesting issue at this stage where the form factors of SMD components are labelled 01005 and 0402, which are interchangeable unit label measurements between metric and imperial, which have vastly differing PCB footprints. After correcting our errors in this, we placed our components onto our PCB board within specification.

PCB Placement & Routing

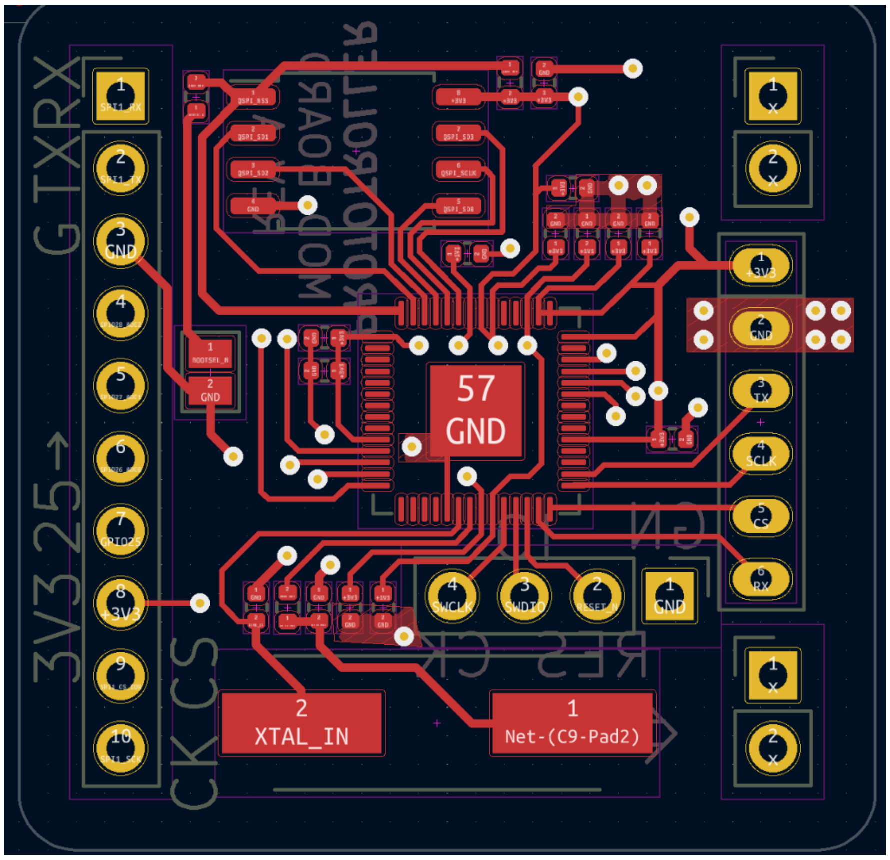

Figure 8: PCB Design, fitted with routing and component footprints, 30x30mm. Hard to do!

We used a 2 layer PCB design for both boards, keeping the bottom layer as primarily a ground plane, limiting the amount of traces trenched through the pour as much as possible. We ensured that placements of high-speed timing devices were as close to their destination as possible, as was the also the case for decoupling capacitors. We fit our connecting pins alongside the top and bottom edge of our board, with the SWD header intended to be unpopulated through holes, used once for programming the RP2040. The module side of the pogo-pin connector is to be soldered onto the underside of the board, and the disconnected through-holes are intended for structural support with the Veroboard.

Our master board has more trenches in the bottom layer due to multiple crossovers, but the inter-board pinouts were matched and designed to minimize these as much as possible. The board has more area for routing, thus, when possible, traces were given additional space between one another, and routed across the top layer in straight lines where possible. Placements from the module board were mostly duplicated, with mild changes due to the increased space available. Rule checks were run on each board with the manufacturing specification parameters provided by our PCB manufacturer. The resulting designs successfully passed with no warnings or errors and were quickly approved for manufacturing.

Final Design Considerations

There were multiple considerations involved in the 'final' beta build, one of which was the inclusion of mounting holes. These holes would have been included to mount the component to the module board. Upon further consideration, these mounting holes would have taken too much space that would have otherwise been taken by other essential components of the module board. These mounting holes would also have not made as much sense with our idea with a Veroboard, a grid of holes with parallel copper strips running from one edge to the other.

As a result, we decided to use 2.54mm pitch male header pins and place them around the board in such a way that one row is tied to one signal. These pins will also allow us to cut an exact square of Veroboard and precisely fit on top of our module board.

All routing, components, and vias were placed with intent and with the idea of aesthetic in mind.

A issue that may arise from this design in practice, is that the pogo pins may overlap with the bottoms of the soldered male header pins and the GND on the RESET and CLK, which is not a big deal since we will not be using it. Cutting these header pins from the bottom of the module board so that they are flushed will potentially resolve this issue.

The project GitHub repository contains all sources and schematics. Instructions are there and will be updated over the project.