TinkerPod: An open-source hardware platform for makers

Juan Martín Sulca Coral

A thesis exhibition presented to OCAD University in partial fulfillment of the requirements for the degree of Master of Design in Digital Futures

Toronto, Ontario, Canada, 2025

Creative Commons Copyright Notice

Abstract

TinkerPod is an open-source platform designed for makers, creative coders, and designers, providing hackable and customizable hardware and software. Inspired by the evolution of multitools and influenced by contemporary open-source hardware and generative art, this project explores the application of critical making and personal fabrication to open-source hardware. TinkerPod aims to bridge the gap between specialized engineering tools and accessible, user-friendly hardware for creative and technical exploration.

This research follows a Research Through Design methodology combined with Kanban to create an iterative approach that refines prototypes based on accessibility, modularity, and robustness. The first phase synthesizes inspiration from open-source hardware and Do-It-Yourself electronics to situate the project. The second phase refines these insights into specific design criteria, focusing on hardware durability and accessibility for users with different skill levels. The development phase of the project applies iterative prototyping, leveraging rapid fabrication techniques such as 3D printing and Printed Circuit Board (PCB) manufacturing to refine the device's physical design attributes and functionality. Finally, the development of the prototypes is documented, including the thought process and design decisions made during each iteration.

The result is a device that can be manufactured using DIY methods and tools, incorporating a flexible hardware and software ecosystem that intends to encourage users of different skill levels to experiment, extend, and hack.

Keywords: personal fabrication, DIY, electronics design, open-source, hardware, making, iterative prototyping

Acknowledgments

My advisors, Nicholas Puckett and Kate Hartman thanks for all the guidance and helping me grow and all your teaching through these two years.

My friends and collogues Abha Patil and Aranya Khurana, for your support and insightful conversations about art and design.

Creative Code Toronto, this amazing community was a great influence for this research and my growth as a coder.

Matthew Nazarian thanks for your camaraderie and support.

To my wife, thank you for your love, patience, and unwavering support.

I am deeply grateful to my mother and sister for their unconditional support, for always cheering me on, and for being a constant source of inspiration.

Finally, I would like to thank everyone who played a part in this process. Your contributions, big or small, have made a difference, and I am truly grateful.

Table of Contents

List of Figures

List of Tables

1 Introduction

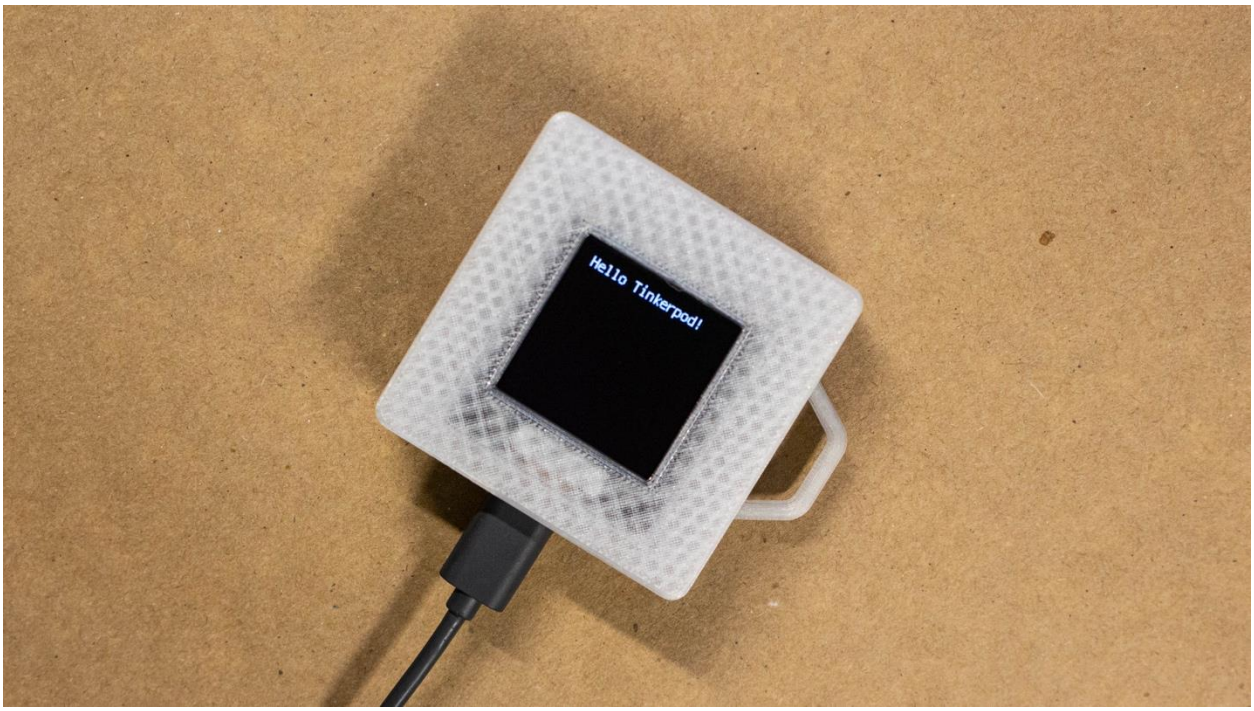


Figure 1 TinkerPod alternative handle case.

1.1 Background

The relationship between people and their tools has undergone continuous transformation throughout history. Nowadays with digital fabrication and open-source hardware democratizing technology creation, makers have access to knowledge, processes, and resources that enable them to design and fabricate devices for their daily lives (Chris. Anderson 2014). As these barriers continue to lower, there is a growing opportunity to bridge the gap between specialized engineering tools and user-friendly creative platforms.

1.2 Idea

TinkerPod is an open-source hardware platform for makers that emphasizes DIY and personal fabrication. Its purpose is to empower makers to explore, learn, and make electronic devices. Drawing inspiration from critical making and critical engineering, the project embeds artistic and design capabilities into its functionality, aesthetics, and making capabilities. TinkerPod is designed to be more than an engineered device, but a well-thought-out and built piece of open-source technology that prioritizes accessibility and optimized implementation over raw performance.

TinkerPod (Figure 1) aims to offer a versatile and approachable hardware platform that makes engaging with hardware enjoyable, with functionality like timers, games, and drawing capabilities. By exploiting transparency, accessibility, and personal fabrication, it intends to empower its users to learn, create, make, and engage with technology.

For clarity, the following terms are defined as they are used in the context of this project.

Easy

The term easy refers to interactions that feel natural and effortless, where underlying complexity is present but does not interfere with the experience, exploration, or making.

Enjoyable

Enjoyable describes the satisfaction that comes from engaging with device, whether using, making, studying, or creating with it.

Appealing

In this context, appealing goes beyond aesthetics to create an inviting presence that encourages deeper engagement and curiosity.

1.3 Motivation

As a formally trained software engineer and a beginner at 3D printing a PCB design, I have mostly developed my problem-solving approach around the following model: $f(inputs) = outputs$, outputs equal a function of inputs. Now, I have come to understand that *making* is more complex than just transforming inputs into outputs, rather it is an interconnected network of multiple equally important concepts interacting together as a single unit. This new view of the world sparked my interest in our relationship with technology and how it helps define our lives and interact with the environment around us.

The inspiration for this project came from the Flipper Zero, a portable multitool for pentesters (information security testers) and geeks in a toy-like body. But I am not a cybersecurity expert, so using or customizing the currently available tools was not something that seemed approachable despite its Open-Source nature. This led to a vision: creating a hackable platform that other people could extend, remix, study and make. The device needed to appeal to: creative coders, designers, artists, and makers, with the goal of empowering this community with a device they could integrate into their daily lives.

Open-Source hardware and software became fundamental to TinkerPod's design philosophy, giving users the freedom to hack, extend, repair, understand, and truly own their devices. TinkerPod aims to go beyond a gadget by engaging users in critical making-allowing them to participate in technology creation through building and utilizing the device.

As Figure 2 shows, digital fabrication techniques enable DIY devices and personal fabrication. This relationship allows TinkerPod to be designed as a device that people can manufacture and assemble themselves, transforming it into more than a weekend project.

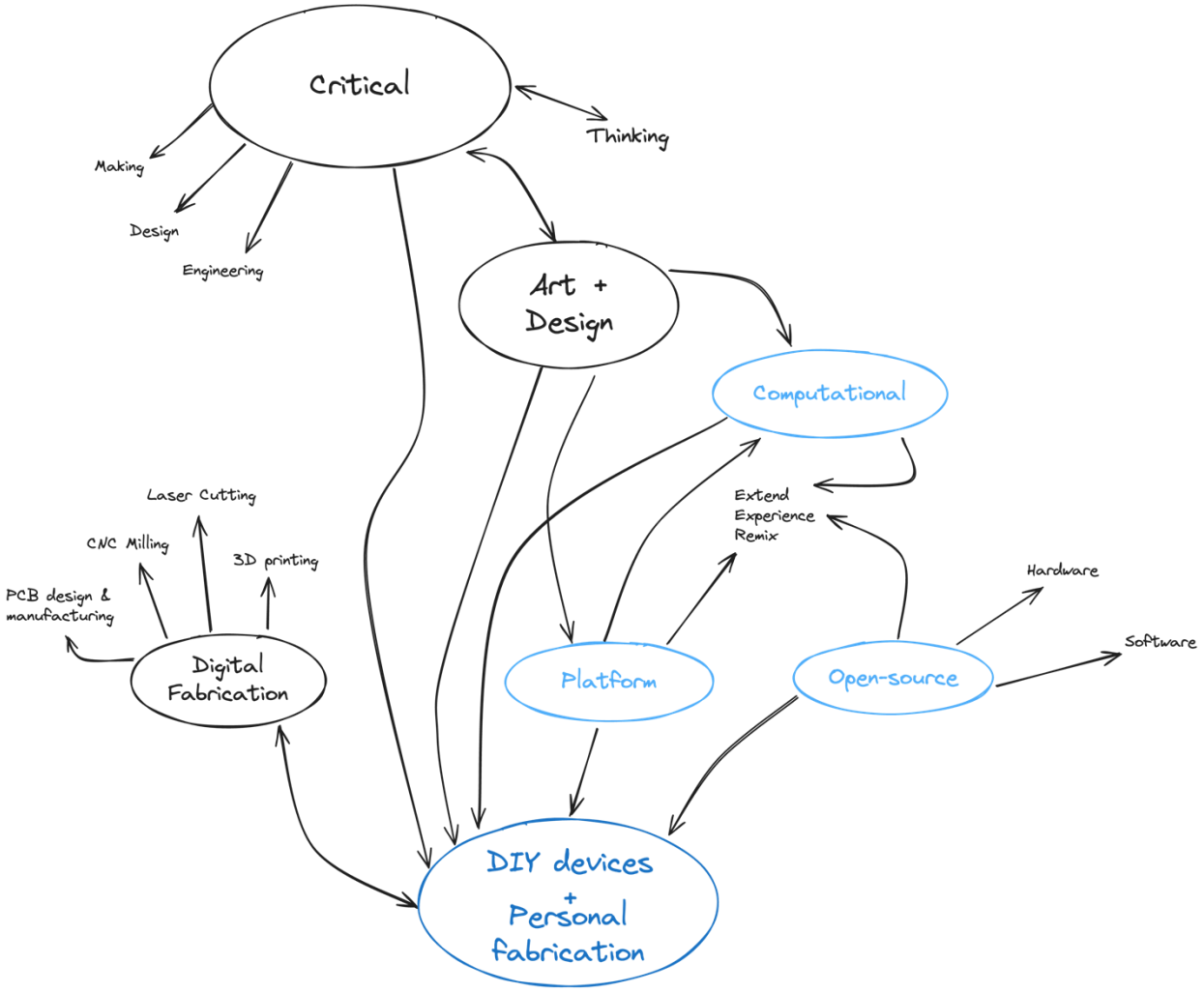


Figure 2 motivation mind map

By combining open-source hardware and software, personal fabrication, and critical making, this DIY device aims to bridge gaps between specialized engineering tools and accessible, user-friendly hardware for creative and technical exploration. This journey resulted in lessons about creating technology and designing making and building experiences for makers and DIY practitioners.

1.4 Research Questions

- What design criteria and fabrication techniques can make DIY devices accessible, robust, and appealing to makers, creative coders, and designers?
- What makes a device hackable and extendable by people with different technical skills and knowledge?

1.5 Scope and Limitations

This research is limited to the design and fabrication of the device, focusing on the research questions outlined in Section 1.4. Formal user testing of the prototypes was not conducted, and this was a deliberate decision. The intended experience with the device involves both its construction and its use as a complete, standalone object, rather than a weekend project. Testing an incomplete device could have drawn attention to temporary issues, rather than providing insights about the usability, ergonomics, documentation, and software functionality. Although the device is open-source and intended for sharing, part of the experience with the device should not be frustrating which could happen during development and before the device was completed.

2 Context review

TinkerPod is meant to be used, build, extended, exploited, hacked, remixed or any other word that represents this concept. This open-source hardware platform is meant to be designed and used as a product (not the commercial kind) in your daily life. The design of TinkerPod directly comes from multitools, DIY electronics, and creative coding as a means of expression. It takes heavy inspiration in contemporary devices but introduces some significant differences in regard to the mental model and design approach of the device.

2.1 The multitool

The multitool is seen as a tool; a functional object with various capabilities, an object people carry and use and finally as flexible device that can meet people needs or modified with their ideas (Chris. Anderson 2014). Through this, it aims to empower individuals to use tools, and to modify them to their needs.

In 1891 in Switzerland, Karl Elsener delivered the first batch of knives made to the Swiss Army specifications (Fiell and Fiell 2011); namely the model 1890. The original functions of the Swiss Army knife were picked to allow soldiers to open canned food and maintain the Swiss army's service rifle of the time which required a screwdriver to be disassembled. At the time, Swiss Army knives were manufactured in Germany, but in 1890, Elsener established the Association of Swiss Master Cutlers; with the aim of manufacturing the Army Knives in Switzerland. Despite failing to compete with the imported tools, Elsener continued to produce knives and iterate on his designs. By 1897 he refined the design using one spring to operate all the tools and blades; a feature that remains a key part of the Swiss Army Knife design even today.

As (Lees-Maffei 2014) mentions; by 1908 thanks to Elsener efforts, the Swiss army commissioned all their knives to two local companies, Victorinox (Elsener's company) and Wenger (bought by Victorinox in the 21st century). During the 20th century the knife became an icon; American soldiers took these tools back home as souvenirs. This gave birth to the "Swiss Army Knife" name since Offiziersmesser was not as easy to pronounce. During the last 140 years, Victorinox has been iterating and refreshing the design of their pocketknives to keep up with current times. The Swiss army knife is a prevalent milestone due to its functionality, compactness and unadorned form. The Swiss army knife is part of the Museum of Modern Art (Karl Elsener 1968) and Munich's State Museum of applied art and Design collections.

During a budget European trip in 1975, Tim Leatherman's encounters with leaky pipes and roadside repairs inspired him to envision what would become the first Leatherman Multitool (Leatherman Tool Group, n.d.). Built around pliers instead of the blade—in contrast to the Swiss army knife—Leatherman chose its functions with the focus of fixing things on the go. During the following years, Tim Leatherman focused on designing and prototyping to perfect his idea; it took

3 years to file his first patent on the multitool design. He spent the next 3+ years trying to pitch his idea without much success. That changed in 1983, when the Pocket Survival Tool (PST) was released with an order of 500 units from Cabela's.

For the last 41 years, Leatherman has continued to iterate and innovate in the multitool market with their iconic design and form factor. Despite having different approach and mindsets on multitool design; both Victorinox and Leatherman designs are iconic and appeal to professionals, makers and hobbyists alike. These two multitools demonstrate what (Lees-Maffei 2014) describes as "how objects are conceived with intended users and usage in mind". Despite designers' attempts to guide user behaviuor, tools end up being extended beyond their original purpose. Sometimes it is not about the best tool for the job, but the one available in the moment.

According to (Bdeir 2009), the designer of littleBits, tools have evolved from simply extending human capabilities to becoming indispensable devices that we carry with us everywhere. Bdeir highlights a significant shift in electronic products: "from being integral parts of our lives, to helping define them". For example, consider the question: *Are you an iPhone or Android person? Or a Swiss Army Knife or Leatherman person?* This shift reflects how our relationship with tools has changed, they are no longer just instruments for interacting with the environment but have become integral to how we identify ourselves and navigate the world.

2.2 Making and critical what?

According to (Savage 2020) "Putting something in the world that didn't exist before is the broadest definition of making, which means all of us can be makers. Creators". From furniture to coding and sewing, these are all acts of creation. And making is "simply a new name for one of the oldest human endeavors: creation". But making is more than just putting stuff together, it involves a deeper process of exploration, discovery, sharing and growing as a community.

Making is challenging self-perceived limit of what is possible and creating things that are important to you as an act of passion. As pointed out by (Charny 2012; Savage 2020) the spark of creation was slowly fading away during the early 2000s. However, as (Chris. Anderson 2014) highlights in Makers: The New Industrial Revolution, the rise of rapid prototyping technology, open-source software, and the internet played a crucial role in revitalizing this creative spirit. These advancements helped fuel the DIY maker movement—a community built around sharing, learning, teaching, and empowering individuals to make things again.

In the chapter named *We Are All Designers Now – So we might as well get good at it* from (Chris. Anderson 2014) narrates how the maker movement evolved trough the 70s to the 2010s and how it was turning into a mainstream phenomenon. Now around 10 years after that book was written, the techniques and tools for makers are more mature and broadly available.

Making greatly benefits from a *critical* approach; but what is the meaning of critical in the context of making. As described by (Ratto 2017), *critical making* is the combination of critical thinking with

hands-on making. It's important to state that critical making share values with critical design and other critical practices; critical making explore these values in society and their implementation and concretization as artifact, trying to connect humanistic practices to design methodologies.

In the exploration of the combination between critical thinking and other areas of knowledge, the concept of critical engineering appears as complement to critical making. As indicated by (Sipos, Chinoy, and Ruiz 2018; Claris and Riley 2012) critical thinking applied to engineering reflects on engineering itself; and opens questions about the production of technology and out relationships to it. This introduces the idea of not limiting critical thinking to logical/academic ability but turns into reflective and reflexive practice across disciplines including ethical and social considerations.

This relation between critical thinking and engineering can be seen in The Critical Engineering Manifesto by (Oliver, Savičić, and Vasiliev 2017). In their manifesto, the first statement reads:

> *The Critical Engineer considers Engineering to be the most transformative language of our time, shaping the way we move, communicate and think. It is the work of the Critical Engineer to study and exploit this language, exposing its influence.*

This manifesto serves as a guide for the creation of technology with an emphasis on the human, cultural and social impact of the creation and consumption of technology.

2.3 DIY and making devices

Do-it-yourself (DIY) is not a new concept; but it was consolidated as we know it after World War II. DIY has evolved and changed over time, experiencing significative changes thanks to the internet, digital fabrication techniques and the growing maker movement. In modern times finding a DIY solution for almost anything is just a matter of a Google search.

DIY is more than just *making* but it's a mind-set, and an approach to problem solving. While it might be easier to just buy ready-to-use products from Amazon or Temu. In contrast with this, DIY offers endless possibilities of customization, possibilities of learning, the right to own and repair your devices and even just be taken as a hobby. With increased availability of tools, materials and most important knowledge, about taking on new challenges has never been easier.

This accessibility has opened the door to take on challenges, and as (Chris. Anderson 2014) points out, we have the factories, and we are all designers and problem solvers capable of learning, imagining and building artifacts ranging from furniture to electronics. At this point it was clear, the tools and materials were present for designing TinkerPod, the only missing piece was how to materialize it and how to ensure empower others to learn, explore and make.

The design approach from this project was heavily inspired by (Mellis 2015) thesis on DIY devices. His work introduces the idea of personal fabrication; a term used to describe:

> *an individual combining low-volume PCB fabrication, off-the-shelf electronic components, digitally fabricated enclosures, and embedded software to form electronic devices for use in daily life*

These kind of DIY devices are meant to be fabricated by individuals and like critical making, is meant to allow someone to experience the making process hands-on. DIY devices rely on PCB manufacturing combined with off-the-self components like microcontrollers, switches, screens to provide the building blocks.

Personal fabrication relies on digital fabrication for production. This includes but is not limited to 3D printing, laser cutting, CNC milling, using a sizable range of materials available. It depends on embedded software to provide the last missing piece of any device. Finally, these devices are meant to be used in daily life, the mindset behind the design of this electronic devices is to be more than a weekend project; but to be used extensively.

To situate personal fabrication in the context of manufacturing methods, Table 1 compares it with approaches to making electronics. The key difference between approaches is mainly in the possibilities of customization, reproducibility and robustness of the final assembly. For example, something built using electronic prototyping is not meant for extended use.

	Personal fabrication	Electronics prototyping	Electronics kits	Digital design for mass production	Hardware startups
Degree of individual control	High	High	Low	Low	Medium
Reproducibility	High	Low	Medium	High	Medium
Robustness	High	Low	High	High	High

Table 1 Comparison of electronic making approaches from (Mellis 2015)

Other factors need to be considered for personal fabrication of electronic devices. Since a DIY device does not need to optimize for production costs or production efficiency; it allows for a greater variety of fabrication techniques and materials. Specific design considerations like the geometry or taper needed for injection molding does not necessarily apply to DIY devices using this approach.

(Mellis 2015) introduced some design considerations that are similar to Design for Manufacturing (DFM) rules required by some manufactures in successfully produce a part or component. The considerations listed below are best practices for DIY devices and apply for the fabrication process and the electronic components.

Repeatability: can you consistently fabricate it? can other people fabricate using similar tools and techniques?

Scalability: how many can you produce with the available resources?

Accessibility: can you access the tools, machines, or services needed to fabricate the parts? what is the cost of the parts and processes? Is there software available for the components?

Post-processing: what is needed to make the parts usable?

Reproducibility: is it possible to source the same parts, materials, and complete the assembly process? Do you need a manual process to complete it?

These considerations were fundamental for the design criteria and decision making for TinkerPod. Serving as guidelines for every point of the process, from picking components to small decisions in the case design and assembly process. This shifted the focus from the design of object to the design of the experience of building and assembling the device.

This later expanded to the documentation and firmware, making sure the documentation includes everything someone would need to study, reproduce or hack the device. From making a new case

from scratch to producing different versions of the PCB with different manufactures. And on the firmware side, trying to inform the user of small decisions made in the code and providing useful examples that can help beginners getting started writing their own applications.

The final point for this, was designing not only for the expert user, but also for the beginner user that want to get started with something new and exciting. How to get them to engage with technology in way that is approachable, flexible and friend. Something that is designed for them.

2.4 Open source

Open-Source Software (OSS) branched from Free Software around 1998, while the two terms might seem interchangeable; free software is older and more rigid in its definition of freedoms and philosophical principles, as explained by (Bretthauer 2001). Without Free Software, the Open-Source movement would have not existed.

According to (Open Source Initiative 2007) in the Open Source Definition, Open Source (OS) is more than just sharing the source code, which is from where the source in open source comes from. According to this definition, the distribution terms must comply with a set of criteria related to the use, modification, distribution and licensing.

The official Open-source definition does not mention anything about hardware as seen in the previous section. But there is a big difference between open-source hardware and software. While in software it is common practice to use licenses like the MIT license, BSD, among others; hardware is more complicated. Starting by what is the source for an Open-source hardware project? Is it just fabrication files, editable CAD files, is any other documentation required?

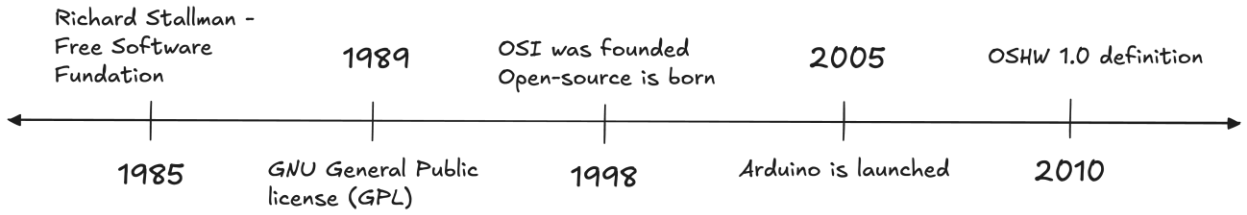


Figure 3 Open-Source Hardware timeline, constructed from: (Bretthauer 2001; Kushner 2011; OSHWA 2013)

For this reason, the Open-Source Hardware Association (OSHWA) made an effort to create a formal definition of Open-Source Hardware (OSH) derived from the open-source software definition created by the OSI but adjusted to be specifically for Hardware products as indicated on their website (OSHWA 2020).

The OSH definition details what is considered open-source hardware, the source files in terms of it and the criteria of distribution of OSH. This criterion specifies the documentation needed, what needs to be shared in terms of software, what is accepted or not as documentation, and since is derived from the Open-Source definition, it also includes the no discrimination clauses and how to distribute the license.

As shown by, (Bonvoisin et al. 2017) the definition is not always applied equally, and it can be inconsistencies on the application of OSH principles. This study analyzed different OSH projects in different factors like transparency (sharing CAD files and schematics), accessibility (editability and participation guidelines), replicability (assembly instructions and materials), and commercial usability (licensing). While most of the projects complied with transparency and commercial usability; a big percentage of projects lack emphasis on replicability and accessibility. This showed the challenges and opportunities in OSH.

Another key point presented by (Bonvoisin et al. 2017) is the distinction between two lifecycles of Open-Source Hardware. The first, *Open-Source Product Development*, follows a community-driven approach where collaboration occurs from the early stages of creation. The second, *Open-Source Product*, refers to a product in the late stages of development when it is defined and ready be built, and its documentation can be shared to support its diffusion, reparability, modifiability and upgradeability.

3 Methodology

This project applies Research through Design (RtD) in combination with Kanban. The choice of using Research through design was inspired by work on the field of Human-Computer Interaction (HCI) and the approach to making the "right" thing (Zimmerman, Forlizzi, and Evenson 2007). The methodology described in this section was also influenced in by (Bowers 2012) introducing the use of annotated portfolios in combination with Research through Design to better articulate the process and outcomes of the research. In addition, The Design of Every Day Things is influential in the moment of producing the artifacts with an emphasis at making design decisions explicit (Norman, D.A. 2013).

Kanban was used to complement RtD methodology, offering a simple framework to measure and organize work in progress (D. J. Anderson 2010). I chose Kanban due to familiarity with the framework and the need for structure and organization during prototyping. This results in a process influenced by an incremental and lean approach that explores the application of a software development mindset to creating hardware.

The process is divided into four stages which took inspiration from (Beşevli, Göksun, and Özcan 2022) on the way the authors applied RtD to generate insight about their research. The first stage of the process consists of gathering inspiration and situating the context for the project. The second stage takes from the context phase and generates design criteria that will guide and influence the next stage. The third stage applies the design criteria to generate prototypes that are analyzed in the fourth and final stage of the process to generate reflection about the process and the artifacts themselves. From the fourth stage (reflection) it is possible to go back to the prototype stage applying the generated insight to new prototypes. Figure 4 illustrates this process.

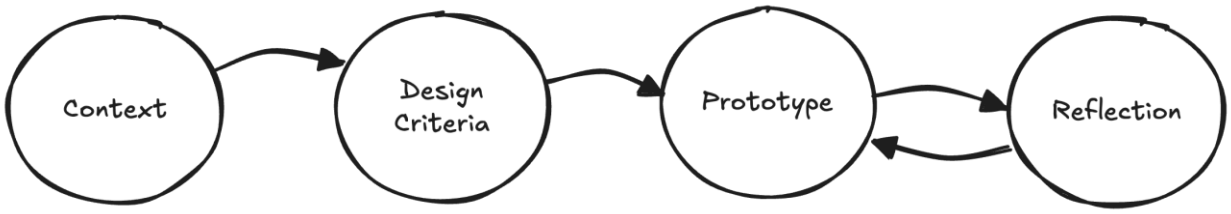


Figure 4 Applied Research through Design process

The design criteria derived from the context refers to set of guidelines and principles that inform decision-making, direction and mindset during prototyping and reflection on the generated artifacts.

The prototyping phase of the process starts with setting a goal for the prototype and follows these steps: first creating a plan for the prototype, then going through ideation to materialize a design

based on the design criteria and the plan, after the prototype is built during the making step and finally reviewed in contrast with the goal. This process can be repeated adjusting the plan and iterating until a functional increment that meets the goal is created.

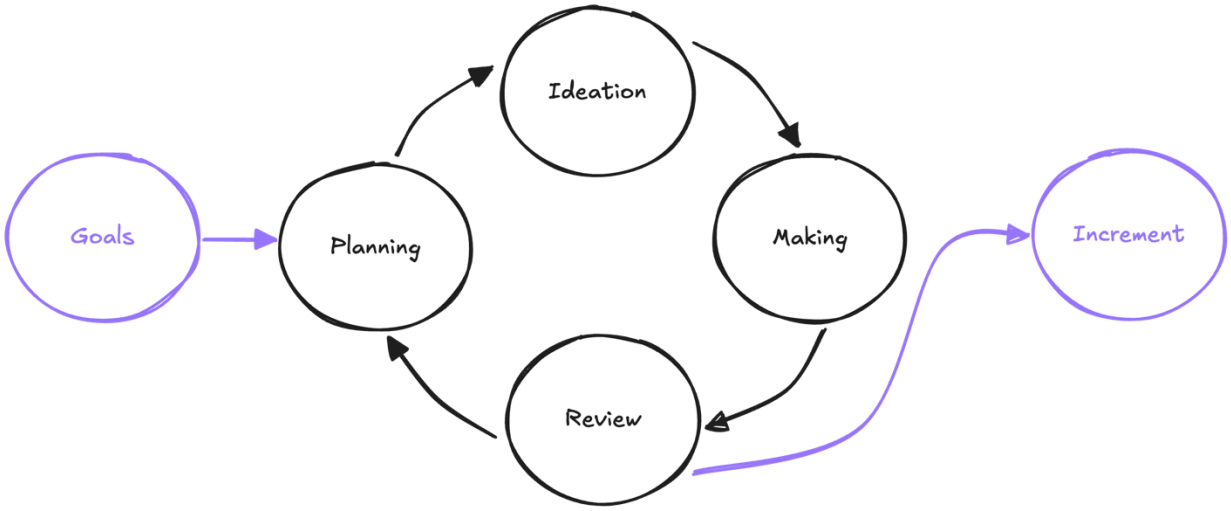


Figure 5 Prototyping phase steps

The *increment* at the end of prototyping is derived from software development; specifically, from the Scrum guide (Schwaber and Sutherland 2020). For this purpose, an increment is meant to represent a functional and usable bundle of software, hardware or both that is additive and works with previous increments to achieve the prototype's *goal*.

The final stage of the process (Reflection) goes over why and how the design decisions were made during prototyping and explains in more details the features and outcomes from each iteration. The purpose of this phase is to guide the next prototypes and illustrating the process followed to get to the prototype goal and how this goal was reached through iterations and adjustments during prototyping.

4 Design and fabrication

4.1 Context

The importance and classification of physical computation devices in computer science education is described by (Hodges et al. 2020). From their classification, two relevant categories were identified: packaged electronics and development boards. A third category was added to encapsulate devices that are self-contained standalone computing platforms with integrated interfaces and functionality.

Category	Type of product	Example
Packaged electronics; no programming	Kits of components and modules	littleBits
Development boards	Microcontroller Boards: With integrated input/output With low level input/output with support for modular input/output	Micro:bit, Inkplate 2, QT PY RP2040
Standalone devices	Self-contained and enclosed standalone device	Chumby, Flipper Zero, SQRT, Playdate

Table 2 Classification of physical computation devices

The devices in Table 2 were selected because of their relevance in the physical computation space, their interfaces, features, functionality and specially their relation to hacking and open source. The following section covers a short description of these devices and their relevance for the development of TinkerPod.

4.1.1 Flipper Zero

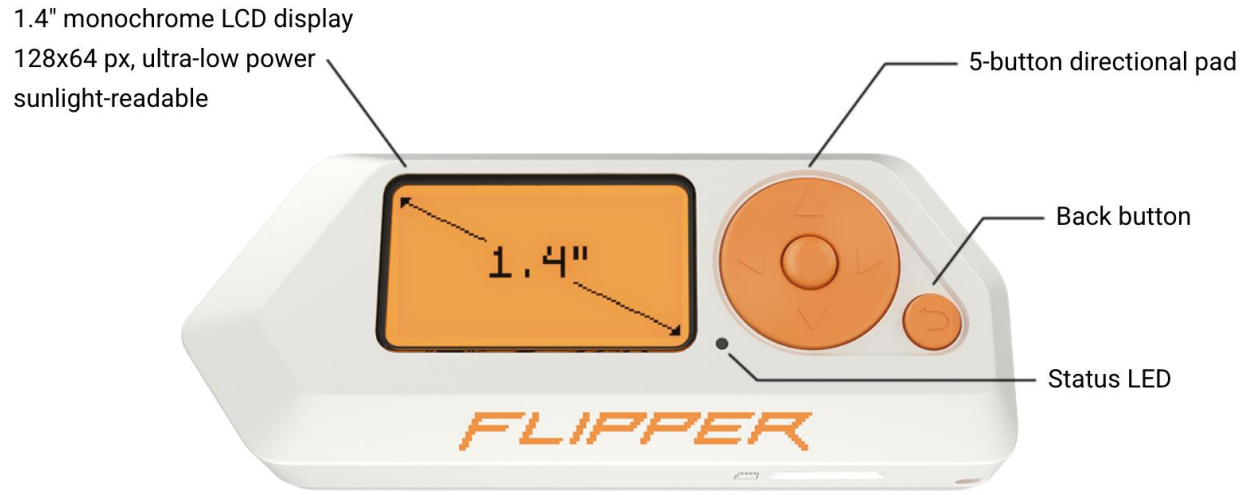


Figure 6 Flipper Zero front view. Source: (Flipper Devices Inc 2024)

As described on the Flipper Zero manufacturer website (Flipper Devices Inc 2024), "Flipper Zero is a portable multi-tool for penetration tester (cybersecurity experts) and geeks in a toy-like body". According to its creators, the Flipper Zero was inspired by the Pwnagotchi, a DIY device for "pwning" (a term usually referring to breaching) Wi-Fi networks. One of the Flipper Zero's main selling points is its design focused on everyday use convenience and robustness. It is a hardware tool for research and penetration testing designed to work on the go, including tools for radio protocols, access control systems, hardware and other functionality.

The Flipper Zero is relevant in this context because of its open-source software approach. While some of the CAD files for the case are shared through the official GitHub account and technical specifications are clearly listed, this is not open-source hardware product; only the device's firmware is open-source software. The Flipper Zero ecosystem is built around community collaboration on its firmware and applications, providing rich documentation, APIs and tooling for building software tools for the platform.

The Flipper Zero is a reference in terms of its simplicity of design and implementation, featuring a 128x64 pixel monochrome screen with orange backlight, a small footprint with 6 total buttons (five-button directional pad and a single back button) and exposed GPIO pins for accessories. The retail price for this device is $169 USD.

4.1.2 SQRT

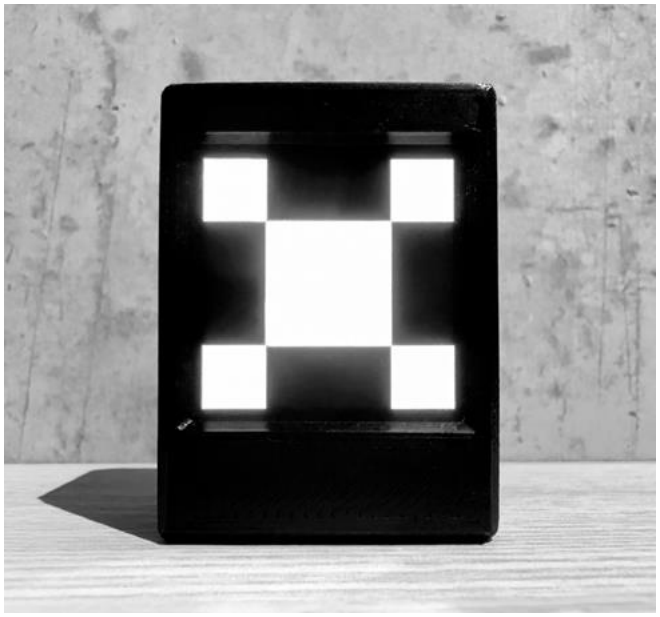


Figure 7 SQRT front view. Source: (Kirklewski 2024)

The SQRT is a miniature generative art collectible piece. It runs C++ code on a microcontroller housed in a 3D printed case with a battery, a 128x128 pixel LED screen and a USB-C port for charging. The main mode of interaction with this handheld device is through shaking it and according to (Kirklewski 2024), it "operates on a square root function, iterating through a doubling grid to generate new algorithmic compositions".

Although this device is not open-source, and its sole function is to display generative art, it opens possibilities for similar devices focused on generative art in a comparable form factor. Its functionality and aesthetic qualities heavily influenced the form factor and functionality of TinkerPod.

4.1.3 Soldered Inkplate 2

Figure 8 Inkplate 2 front view. Source:(Soldered Electronics 2023)

The Inkplate 2 is a development board by (Soldered Electronics 2023) that consists of an ESP32 and a 2.13" three-color e-paper display integrated on a custom PCB. The board includes a USB-C port, charging circuit and access to GPIO pins. The Inkplate 2 is available in three versions: base board without the e-paper display, board with the e-paper display and enclosure and a complete version with display, battery and enclosure. The included case is 3D printed and features a snap-fit design.

The Inkplate 2 exemplifies an open-source hardware device. All design files, including PCB designs and case files, are available in their GitHub. These include both editable CAD files and production-ready files (Gerber files for PCB manufacturing and STL for 3D printing). The project repository also contains complete circuit schematics and Bill of Materials (BOM). The device certified by OSHWA and is licensed under TAPR Hardware License for hardware components and GPL for software.

4.1.4 Chumby

Figure 9 Chumby One. Source: https://www.cnet.com/pictures/chumby-one-photos/

The Chumby was an open hardware content delivery device designed by Andrew "bunnie" Huang in 2007 (Huang 2019). Huang's vision for the Chumby was to create a platform as open as possible, allowing hackers to explore and modify the device in whatever way they could imagine. This vision was realized by making the source code and all design files freely available, inlcuding Bill of Matterials (BOM), schematics, board layouts and 3D CAD models.

Despite the company responsible for the Chumby closing in 2012, several key and powerful ideas emerged from this family of devices. The concept of open-source hardware device intentionally designed for hacking and tinkering opened a world of possibilities for its adopters. As (Huang 2019) noted, the best of open-source hardware is still yet to come.

4.1.5 Playdate

Figure 10 Playdate. Source: (Panic Inc., n.d.)

The Playdate is "a curious handheld gaming console" designed by (Panic Inc., n.d.) in collaboration with Teenage Engineering. It features a 400 × 240 1-bit (monochrome) screen, a Cortex M7 processor with WI-FI and Bluetooth capabilities, a D-pad, A+B buttons and a crank. This device is remarkable for its SDK that enables the Playdate community to build and share video games programmed in C or Lua. The platform emphasizes community-driven development of applications and games. While the Playdate is not open-source hardware, some games are released as open-source software.

Key elements to take away from the Playdate are its community aspect and ready-out-of-the-box functionality, including a store where games and apps can be purchased and installed on the device. The Playdate offers a rich developer experience through its desktop emulator and documentation for game development.

4.1.6 littleBits

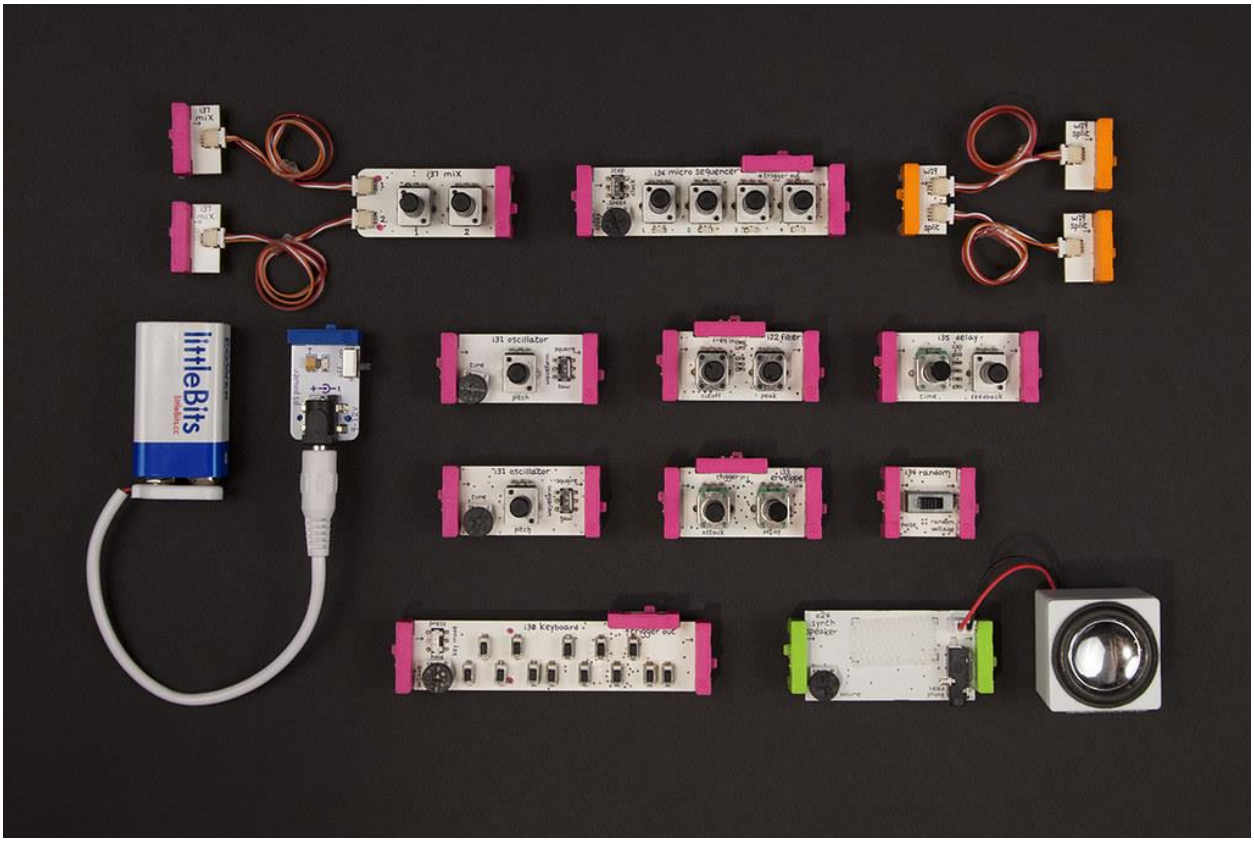


Figure 11 littleBits synth kit. Source: https://www.flickr.com/photos/130557019@N06/15898282384/in/photostream/

littleBits is "an open-source library of discrete electronic components pre-assembled in tiny circuit boards" (Bdeir 2009). It consists of a collection of LEGO-like pieces with magnets, designed in a playful way to reduce the complexity of electronics and make them accessible to artists, makers, designers and educators. These pieces can be combined to build a sizable number of circuits, with the limitation they are not programable and cannot be easily combined with components outside littleBits.

The littleBits project exemplifies accessibility, with its simple concept and elegant implementation serving as an inspiration for modularity and accessibility in this project. By limiting the number of available modules, it was possible to manage the complexity of assembly while maintaining flexibility. littleBits makes effective compromises to create modules that can be arranged in multiple configurations, with two downsides: they only work within the littleBits ecosystem and are not designed for extended use.

4.1.7 Reflection

To situate where TinkerPod exists in the landscape of the devices listed below, Table 2 provides a classification framework. Figure 12 presents these categories along a spectrum, from packaged electronics to standalone devices, illustrating two relationships. While devices on the left offer

high modularity and flexibility but require more assembly, devices toward the right have less modification possibilities but provide more ready-to-use functionality.

Figure 12 Landscape of physical computation devices

The Packaged electronics category consists mainly of the littleBits, which lacks programmable capabilities and offers discrete electronics. The development board category offers near-infinite possibilities at the cost of more knowledge needed to build something. The standalone devices category consists of devices that are not necessarily programable but offer a self-contained and enclosed standalone device. This category is ordered according to the level of intentionality to allow users to extend the hardware or software.

Devices like the Flipper Zero and Chumby are Open-Source and are intended to be hacked or extended with additional functionality in both hardware and software. The Playdate, while not meant to be extended using hardware, offers developer tooling to write games and applications using the Lua or C programming languages. Finally, the SQRT can't be extended at all but consist of a microcontroller, a PCB, and a screen inside a 3D printed enclosure.

In terms of the vision for TinkerPod, it aims to position itself between the development board and the standalone device categories while offering a level of modularity reminiscent of the littleBits. This will be achieved by using standardized and broadly available modules, controllers, and protocols. While this approach helps with the modularity aspect and the availability of modules, it is not a bulletproof solution, as it potentially requires software or hardware updates to support different modules.

TinkerPod also takes inspiration from the Flipper Zero and Chumby in its intentionality to be hacked, extended and modified by its users. While TinkerPod is not a standalone device in the sense that it requires assembly, this is also intentional and serves as an invitation to learn and explore during the assembly process.

Finally, TinkerPod was inspired by a combination of the SQRT's form factor and art collectible functionality, combined with the community and hackability of platforms like the Flipper Zero and Playdate, which offer means of extending and sharing user-created applications. The case and partially the internal layout of TinkerPod were inspired by the Inkplate 2 Open-Source hardware and 3D printed case design.

4.2 Design Criteria

In the context of this project, the multitool is seen in three layers. The first is by its definition as a collection of tools in a single object; directly from its roots on the Leatherman or the SAK. Then as object people carry, multitools are gear people carry and use on their daily lives, things like a bottle opener or the blade can be useful in many contexts. And as general but specific object, multitools come in different shapes, forms and with different models offering tools for outdoor adventures or urban explorers.

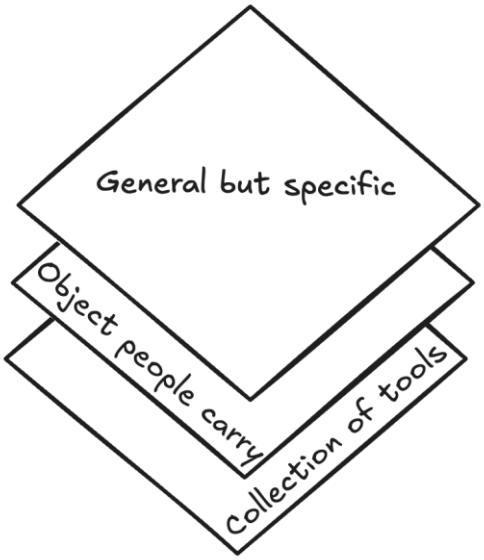


Figure 13 Three layers of the multitool

For example, Victorinox offers an electrician model, that comes with a blade intended for wire stripping; while some Leatherman models offer a lightweight design oriented to hikers with a blade designed to cut rope. In this case, both tools have blades with designs that appeal to different groups of people while not being tailored for a specific individual. This is where there is an opportunity for rethinking a hardware platform across these three layers.

Most of the decision during the design of TinkerPod come back to these three characteristics of the device being a hardware platform for people to use and experience. An everyday object meant to be carried around and used, and a general device built from of-the-shelf components with the capability of being tailored and customized to the user's needs using software and hardware.

The core criteria that guide the design of TinkerPod is outlined in this section.

4.2.1 Privacy

TinkerPod makes a clear distinction from devices like cellphones that are meant to be connected and carry all your personal information. By default, TinkerPod does not include any connectivity; these options are available for advanced users to implement by changing the microcontroller for a WIFI or Bluetooth enabled one.

This approach is informed by the principles of Critical Engineering, as discussed in section 2. In an environment increasingly dominated by connected devices, the introduction of another mobile application or data-collecting device is not necessary nor desirable. Therefore, this platform is designed to operate without an internet connection.

4.2.2 Simplicity and transparency

Technology does not need to be concealed behind complex layers of abstraction. Devices can be designed to be accessible, easy to repair, and straightforward to operate and maintain. A simple yet robust platform has the potential to foster creativity, much like the early video game platforms did.

This was also influenced by the Critical Engineering and the maker movement (section 2.2). Simple technology can be easier to understand and engaging at the same time, it is not matter of making things simpler but finding the right compromise in complexity that can be understood by makers around the world.

4.2.3 Modular

TinkerPod is meant to be a modular device allowing to use different components that integrate in the same system. For this, the choice of microcontroller was essential. Despite the pinout and form factor of development boards not being standardized; the Xiao form-factor boards are offered by multiple manufacturers like Adafruit and Seeed Studio. Offering a range of boards with multiple microcontrollers like the RP2040, ESP32 and SAMD21 with different connectivity options (WIFI, Bluetooth, and others), USB C ports and in some cases extra ports (JST, battery pads).

For the display, the choice of SPI or I2C was given by the STEMMA QT/Qwiic connector present in the Adafruit development boards like the QT PY RP2040 chosen for TinkerPod. The importance of this JST (Japan Solderless Terminal) allows for simple and solderless assembly and integration of development boards with screens and other accessories.

This criterion was influenced by the considerations of *accessibility* and *reproducibility* (section 2.3), taking extra effort to reduce complexity of assembly and looking ahead to ensure it is possible to build this device with components available in the future and by multiple vendors. While some adjustments might be needed to support completely different modules, this might not represent a complete rewrite of the firmware.

4.2.4 Accessibility

TinkerPod is intended to appeal to people with different understanding and experience with electronics. For this, the project has multiple versions at different assembly difficulties for users with multiple levels of experience, from simple completely solderless assembly to fully integrated Printed Circuit board design. Different choices of batteries, screens, microcontrollers and cases. In this accessibility refers to hardware, software, modification, extension and hacking of the device.

4.2.5 Customizable

Different cases, multiple screen choices, different 3D printed materials or colors, button choices, sensors. All aspects of TinkerPod are intended to be customized or redesigned to suit any of the user's needs including software and applications.

Embracing remix culture, a common practice in the maker and open-source movements, this project encourages remixed and derived work, it also tries to make this easier making as many resources available.

4.2.6 Ready to enjoy

By default, this project is designed to include a base level of functionality that can appeal to makers. This functionality includes everyday use cases like games, timers and generative art. Additional functionality can be added to the device by extending its firmware, writing new applications, adding new hardware or a combination of both.

This was decided based on the objective of the project to engage as many people as possible with different levels of experience with electronics. Reducing the skill floor needed to enjoy the project and offering usable and rich functionality for a new user building any of the versions.

4.2.7 Open

This device is intended to be open for modification and extension. This means that TinkerPod is meant to be tinkered with, can be studied, modified or completely redesigned to suit user needs. All schematics, drawings, measurements and documentation are available, allowing for options like changing case material and design, completely overhauling the PCB from scratch.

Relying on Open-Source tools and reusing different bits and pieces of other projects; this project also aims to be as open as possible on its source code, design files and decision making. With influences from devices like the Chumby and adhering to its Open-Source nature and Critical Engineering; hacking and modding TinkerPod is encouraged.

4.2.8 Robust

Taking inspiration from repeatability, accessibility, post-processing and reproducibility from section 2.3; the design and construction should be simple yet robust. The designs should be easy to manufacture with broadly available tools and methods, remove reliance from the process specific tricks or quirks and offer quality to the end user.

4.2.9 Experience

Inspired by critical making, this project can be experienced in multiple ways. By building it, by customizing it and by creating your own applications for it. The idea is to have these options to experiment, learn, designing, printing new cases, mod the hardware, update the firmware with new functionality or simply enjoy the functionality.

4.3 Tools and materials

This project relies on personal manufacturing for its distribution. As this project is an Open-Source Hardware and software platform, manufacturing files like STL for 3D printing, or Gerber for PCB manufacturing need to be available and CAD files like KiCAD or fusion files can come in handy to study or extend the platform. For this reason, a set of tools and materials have been picked to suit the design criteria from the previous section.

4.3.1 CAD

For CAD software two different programs are used. First KiCAD as the Electronic Design Automation (EDA) Open-Source Software (KiCad Project 2025). KiCAD was chosen instead of Autodesk Fusion, because it is completely free and facilitates accessibility to editable files for Schematics, PCB board files and manufacturing files.

For 3D modeling, Fusion was selected as the primary design tool due to its parametric modeling capabilities (Autodesk Inc. 2024), which are essential for creating precise and adjustable enclosure designs. While other options like Blender (Blender Foundation 2025) and Plasticity (Nicholas Kallen 2025) were considered, Fusion's ability to modify dimensions and features through its history-based workflow made it ideal for this application.

4.3.2 3D printing

For 3D printing, Prusa machines and Prusa Slicer (Prusa Research 2024) were used during prototyping. However, all the 3D models were designed to be slicer and printer agnostic, not relying on any specific slicer settings or features. Two materials were used: PLA and PETG filaments. PLA was used for rapid prototyping and concept testing due to its high printing speed and versatility. PETG was selected for final prototypes because of its material properties, including greater flexibility and durability, while remaining broadly accessible for anyone with a Fused Deposition Modeling (FDM) 3D printer.

While this project targets FDM 3D printing, other 3D printing and manufacturing methods like Stereolithography (SLA), laser cutting, CNC milling, and casting are viable alternatives, though they fall outside the scope of this research.

4.4 Prototyping approach

Since this project takes an iterative incremental approach, the first step was to create and ideate the form factor and initial concept for the project. This concept was a stack of layers containing different functionality for the project. From the beginning, the device was mean to be modular and offering options related to microcontrollers as detailed in the previous section, also the design needed to be simple yet elegant to accommodate the components in a simple to assemble and durable 3D printed enclosure. Inspired by software development, the physical construction of the device was meant to occur in incremental steps that added or refined some features of the device.

For this the project rely heavily on rapid prototyping techniques to iterate and adjust case and internal layout for the device.

The mindset for prototyping, was derived directly from the Agile approach trying to deliver a functional increment each time. This allows for experimentation and freedom. If one of the experiments did not go well, it could be discarded or reimplemented for the next iteration. This also allowed to account for unexpected roadblocks; like the top case for the first version of TinkerPod that was designed based on an available drawing of the screen module. The first iteration of the top part of the case had minor adjustment issues and a misalignment of the screen bezel; this was corrected for the second minor iteration of the case design. Most of this adjustment during development occurred during a single day and were possible thanks to the parametric capabilities of Fusion, that allowed to modify dimensions and readjust tolerances of the 3D model with ease.

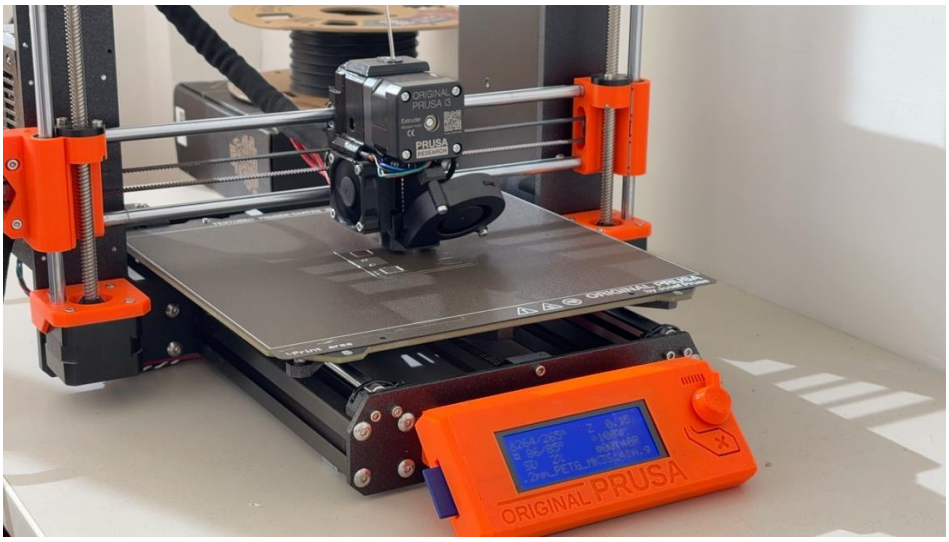


Figure 14 Prusa i3 MK3 printing a TinkerPod case

This allowed for a fast feedback loop, that enabled fast testing of multiple concepts and adjustments of the physical aspect of the device. One exception to this was the PCB which in the end was sent to be manufactured by a third-party service overseas. The first PCB attempt for v1 was CNC milled from a FR4 etching plates, with subpar results. The process was tedious, requiring specific design considerations to accommodate for the milling bit and took a time of around 2 hours for milling a single PCB; during this time the operator needed to be overseeing the mill to ensure success. After the successful milling of a PCB, it required post processing and testing to ensure no pads or traces were shorted and the application of a coat to the copper surface to prevent rusting. Because of this complicated process and requirement of manual post processing and testing of the resulting PCB, the best option for this was outsourcing the manufacturing overseas, this resulted in cheaper cost with the trade-off of a longer wait time and more commitment for each PCB design.

In order to test for proper fitment of the components and the layout of the PCB inside the case, the outline of the PCB was 3D printed before sending to manufacture to ensure proper fitment and alignment with the case. Lacking previous experience in product design, the approach for design was unusual starting by the case and then the PCB instead of the other way around. The case design started from the screen model or drawing and considered the future dimensions of the PCB to accommodate all the components and in the case of v2, this help correctly align the button assembly to the push button. In general case design was a balancing act of trying to minimize case bulk and screen bezel while having enough space for the PCB, screen and battery.

For the firmware, some concerns were common for all the prototypes, like screen setup and initialization. From v1 to v2 and v3 most of the setup and initialization code was carried over. The programming language of choice for this project was Python, thanks to the Circuit Python (Adafruit Industries 2025). This was chosen instead of using the Arduino IDE or C++ because Python is an easier to learn and use programming language with a minimum loss in performance compared to other alternatives. By default, most people start using Arduino IDE, which provide a variety of libraries and a complete development environment that can suit most beginner projects. The current state of Circuit Python, provide built in libraries and compatibility with the main modules most project might use. The only downside is the smaller community and user base compared to Arduino that has been in the market for longer, thus having a bigger knowledge base and more people using it.

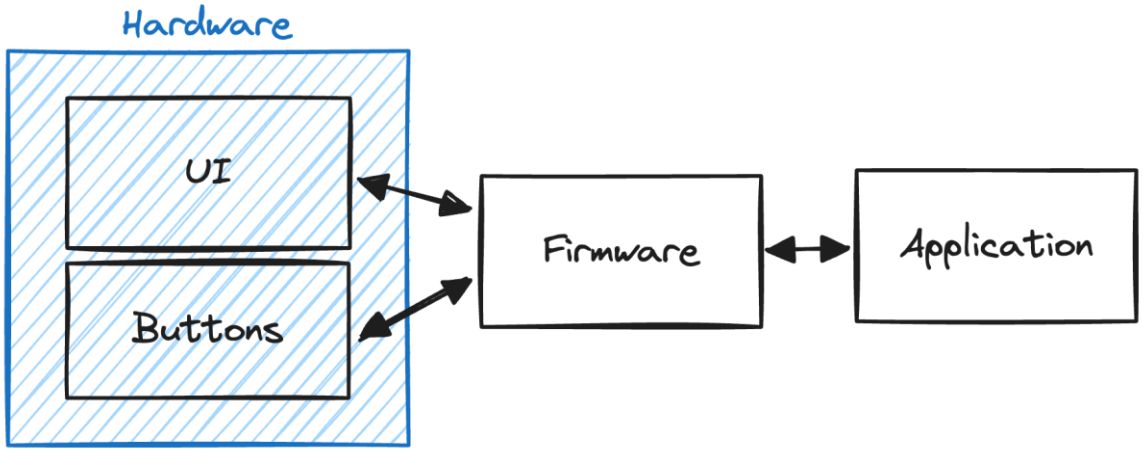


Figure 15 Software model for TinkerPod firmware

The code structure emphasizes modularity, separation of concerns, standardization and reusability to enable multiple devices to use the same firmware with only configuration changes. The firmware uses the Circuit Python abstraction for screen coupled with a class to manage the firmware core and a dedicated class for each application (Figure 15). The core piece of code focuses on Input/Output (IO) and application management.

4.5 The prototypes

4.5.1 v1 TinkerArt

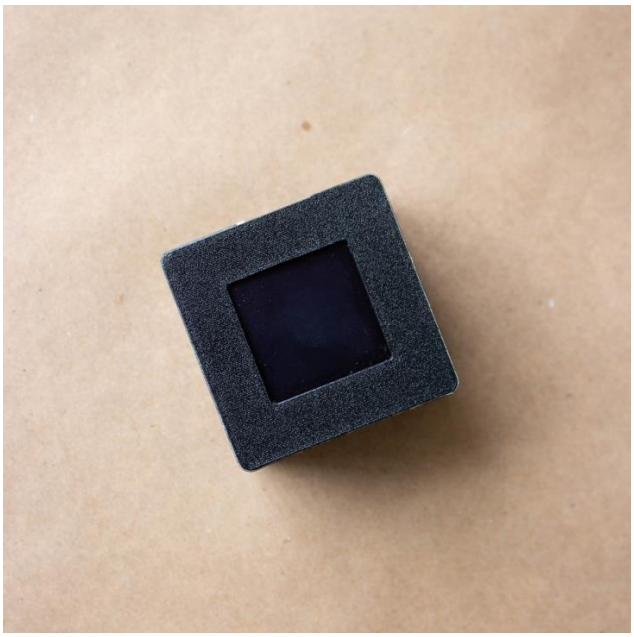


Figure 16 TinkerPod version 1 inactive

The first version goal was to be a simple and easy to assemble version of TinkerPod; this version focuses on solderless assembly that can be done in under 1 hour with the trade-off of reduced functionality.

For this version, the only components needed are a breadboard, microcontroller, screen and 3D printed case. This version mainly focuses on generative art coming with a simple mode of interaction by shaking and no buttons. The user can choose between default algorithms included with the firmware or the possibility of implementing their own by modifying the firmware.

Qty	Part	Notes
1	Adafruit QT PY RP2040	Can be replaced with XIAO pinout boards
1	Waveshare 1.5" Gray scale OLED	SSD1327 driver chip, SPI/I2C interface
1	Tiny Breadboard	46mm x 36mm x 10mm
5	Jumper wires	
1	TinkerPod v1 3D printed case	

Table 3 TinkerPod v1 parts list

See Appendix A: TinkerPod v1 assembly guide

This version can work with a Waveshare 1.5” grayscale OLED or the Adafruit 1.5” grayscale OLED. Both options require minimum assembly and just a slight adjustment on the firmware to support the different screens. Figure 17 shows the internals of TinkerPod including, the microcontroller, tiny breadboard, screen and two-part case, with the device on partially assembled state. The parts for this prototype were sourced from Adafruit and Amazon, complete parts list can be found in Table 3.

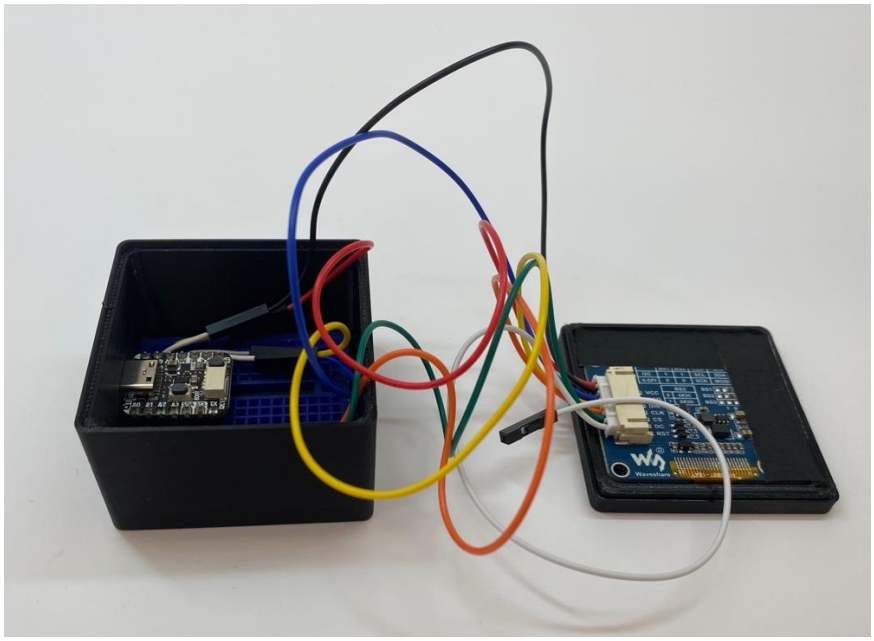


Figure 17 TinkerPod version 1 view of the internals

The case for v1 was designed around the screen module, trying to minimize the screen to body ratio while accommodating the components and keeping a handheld aesthetics and form factor. This case design went through multiple iterations and refinement to achieve its final form. To simplify design and manufacturing, this case used friction to keep the top component (the side with the screen) in place.

Multiple versions of the case were printed in two materials, PLA and PETG. The first prototype was done using PLA to check for tolerances and test fit the components. Minor adjustments were done to the first version to correct for dimensions of the screen. The next versions of the case were printed in PETG for added strength and to test the material for the next case iterations using a snap fit design.

The first case design was purposely constrained to be printable on any FDM 3D printer, without much fiddling around, meant to be printed without supports and requiring no post processing. These considerations were taken to maximize repeatability of the case manufacturing process and remove reliance on the slicers for a successful outcome.

In the end, the version 1 case was left with the friction fit design printed in PETG using a textured plate; with an infill of 20% using a 0.4mm Nozzle and 0.2mm layer height. Other case style options were considered like screw in design, but in the end these options were discarded for an easier to assemble and print option.

The functionality of v1 was inspired by the SQRT as detailed in section 4.1.2; this is an art collectible piece, the key differences are in the endless customization possibilities offered by allowing to modify the firmware and add different input and output using the available GPIO pins.

The first version of the firmware was first prototyped in p5.js by constraining the available API from p5.js to just draw lines and squares in grayscale (Figure 18). The first idea for the generative art function was inspired by Vera Molnár using a tilling pattern with lines that can go diagonally to the right or left and are randomized creating a unique maze-like pattern that changes after a couple of seconds.

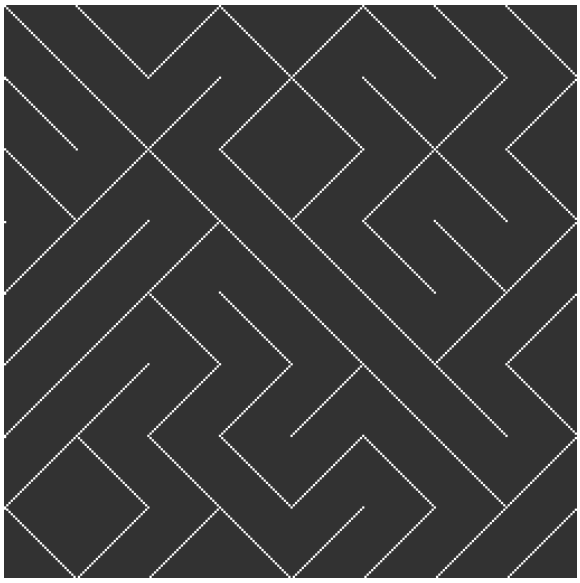


Figure 18 First generative art prototype in P5.js

The next iteration of this functionality kept the tiling concept, and it changes over time, but instead of using just diagonal lines introduced 9 different types of tiles that are randomized for 4 by 4 grid (Figure 19). After generating the initial grid, every 2 seconds, one random tile is updated with a different tile pattern from the pool. The updated tile and tile generator function are pseudo randomly each time and do not depend on each other.

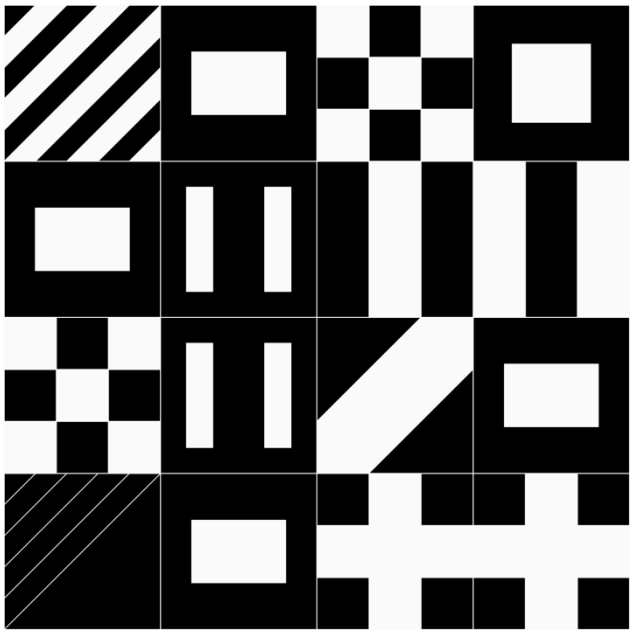


Figure 19 Second generative art prototype

This version set the foundations for the next versions of the firmware, stablishing its base structure and primary drivers like the screen driver and setup. This version compromised some of the modularity in favor of simplicity since its oriented as an entry point to the project focusing more in allowing users to create art.

Another difference with other versions, is the lack of inputs from the user. This version does not have buttons but can be extended to have a shake sensor or it was even tested with capacitive buttons which results were usable but did not provide much tactility to the user and were finicky at best.

The takeaway from this version, was the form factor of the device, that proved satisfactory and the workflow for the next iteration of the design. This version consolidated the aesthetics for the device and created the foundations for the firmware and the basis for the UI.

4.5.2 v2 TinkerPlay

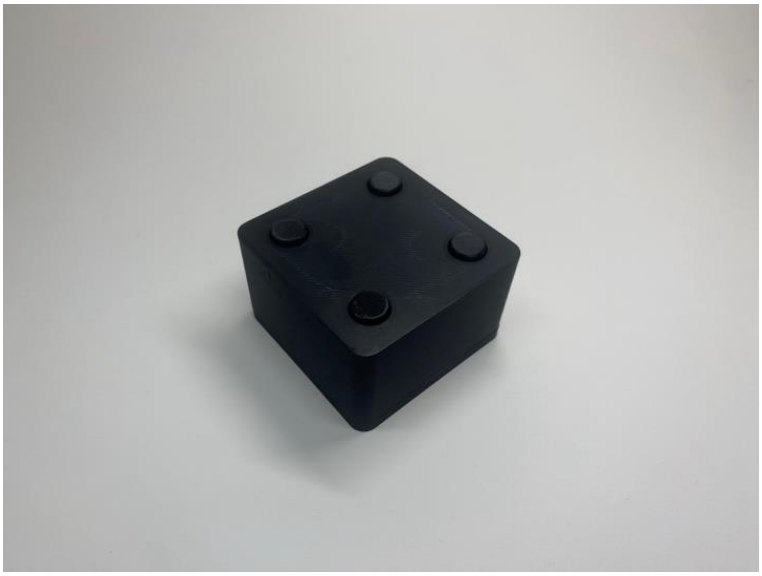


Figure 20 TinkerPod v2 with the buttons facing upwards

The goal, of this version was to increase the robustness of the platform and add interaction capabilities to the device to introduce new functionality.

The second version builds on the foundation of v1, maintaining a similar form factor, but changes the screen module and introduces a custom PCB with button to add a new layer of interaction to the device (Figure 20). This version trades the simplicity of assembly for a more reliable platform. The firmware for version 2 also introduced significant changes to manage the logic for Input/Output (IO), introduced a splash screen and modular structure based on pseudo applications. This introduced for the first time in the project the ability to extend the functionality of the device; not only modify the algorithm for the art generation.

Compared to v1, the parts list for is not significantly larger but the assembly complexity increased; a necessary upgrade to support a more robust platform that now requires soldering. The 3D printed case was also adjusted to accommodate the buttons and uses a bigger footprint to fit the Adafruit OLED screen module.

Qty	Part	Notes
1	Adafruit QT PY RP2040	Can be replaced with XIAO pinout boards
1	Adafruit 1.5" Gray scale OLED	SSD1327 driver chip, SPI/I2C interface
1	TinkerPod v2 PCB	
1	STEMMA QT/Qwiic cable	50mm long
4	Push buttons	8mm rubber dome push buttons
1	Adafruit LiIon or LiPoly Charger BFF Add-On for QT Py	(Optional)
1	Lithium Ion Polymer Battery	(Optional) 3.7v – 420mAh
4	Threaded Inserts	M2 size bolts
4	M2 screws	
1	TinkerPod v2 3D printed case	

Table 4 TinkerPod v2 parts list

See Appendix B: TinkerPod v2 assembly guide.

There are two screen modules that can work with TinkerPod, both are 1.5in OLED grayscale screen modules that use the same screen controller; the SSD1327. The key difference between both screen modules is the presence of the STEMMA QT/Qwiic connector present on the Adafruit module. This connector allows for seamless and easy connection of other modules using a 4-pin JST connector. Using this connector only the I2C protocol can be used, compared to the version 1 that used SPI. The key difference being the speed of each protocol, which can be noticeable in some applications.

The choice of using the on-board STEMMA QT connector was to reduce PCB complexity and balancing the intricacy of assembly, it is straightforward to connect JST cable into a socket, compared to soldering multiple cables between two modules or soldering a ribbon cable connector to the PCB which can be trickier and require more specialized equipment.

This version also includes a LiPo battery and charging circuit, both sourced from Adafruit and compatible with the QT PY RP2040. Both modules are meant to be soldered together back-to-back, in this case to reduce the footprint and maintain accessibility of the integrated on/off switch both modules are soldered on the same side of the PCB side by side.

Figure 21 TinkerPod v2 inside view

Following the same logic, the top part of the case was friction fitted to the bottom part. This friction fit case allows for easy accessibility to the inner power switch, which will be relocated in future iterations. This was a compromise for v2 of TinkerPod to be an incremental increase in complexity compared to v1.

The 3D printed buttons included with the case were inspired by the Super Nintendo injection molded controller. It uses 3 legs and 3 channels to maximize stability of the plastic piece while keeping low friction with the case. This produced a satisfying and responsive button experience. The button designed required some iterations to get tolerance right for a frictionless yet firm button movement.

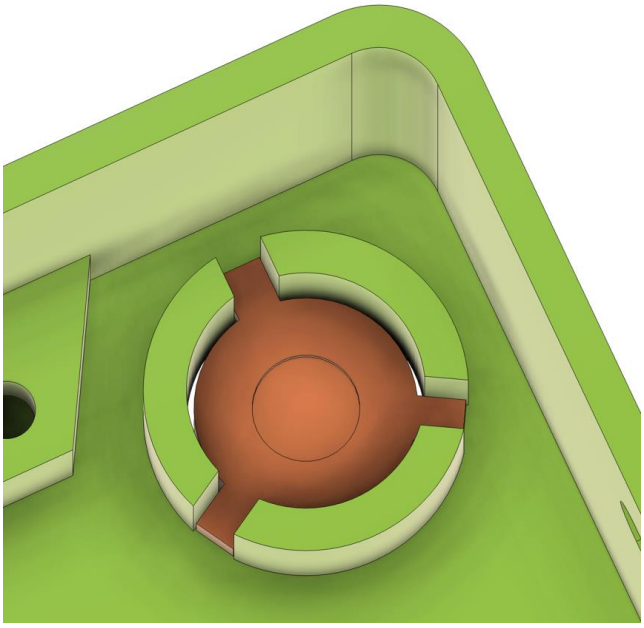


Figure 22 3D model for TinkerPod v2 buttons

The final modification to the case was the use of threaded inserts to securely hold the PCB in place. The PCB was designed with M2 screws mounting holes and the case build with this in mind. This was done to hold the PCB firmly to the case and ensure the buttons feel sturdy.

On the firmware side, the generative art functionality was carried over to the new version but modified to make use of the buttons. Now the user can interact with the device using the buttons to increase or decrease the number of lines on the screen and generate a new pattern on command by the press of a button.

Since this version supports more than just one function, a menu was introduced. Since the buttons are placed on the back of the device, the UI design was built to match the layout. The letters A, B, X and Y were chosen as a reference to game controllers and to help with familiarity for people. The UI elements display a matching on press effect by changing the color of box to white and the color of the text to back highlighting the performed actions.

Figure 23 Inspiration for the TinkerPod menu

The placement of the button deviates from what is usually the form factor of a gaming console with the buttons on the front. This was done intentionally to make a clear distinction between this device (a hardware platform) and the common design of gaming consoles.

This layout was inspired by the 4-button old school digital watches, which UI relied on labeled corners to help the user navigate different functionality and configuration setting (Figure 23). While there is room to improve the implementation of the menu with better icons or an improved layout, this was an incremental approach from version 1.

To test the gaming potential of the device, a version of pong was implemented using the screen and buttons. Pong was chosen for its simplicity, multiplayer capability and a tribute to one of the first video games ever created. In addition, its simplicity and monochromatic aesthetic were a perfect match for the first game implemented on TinkerPod.

The firmware was overhauled to support the menu and the two functions, and use an object-oriented approach, where the entire firmware is a class orchestrating the I/O of the device while smaller objects execute the individual functions like Pong. The firmware interfaces directly with the hardware to get user input and update the screen content, the firmware constantly runs in a loop and is only concerned with this and managing the applications as shown in Figure 24.

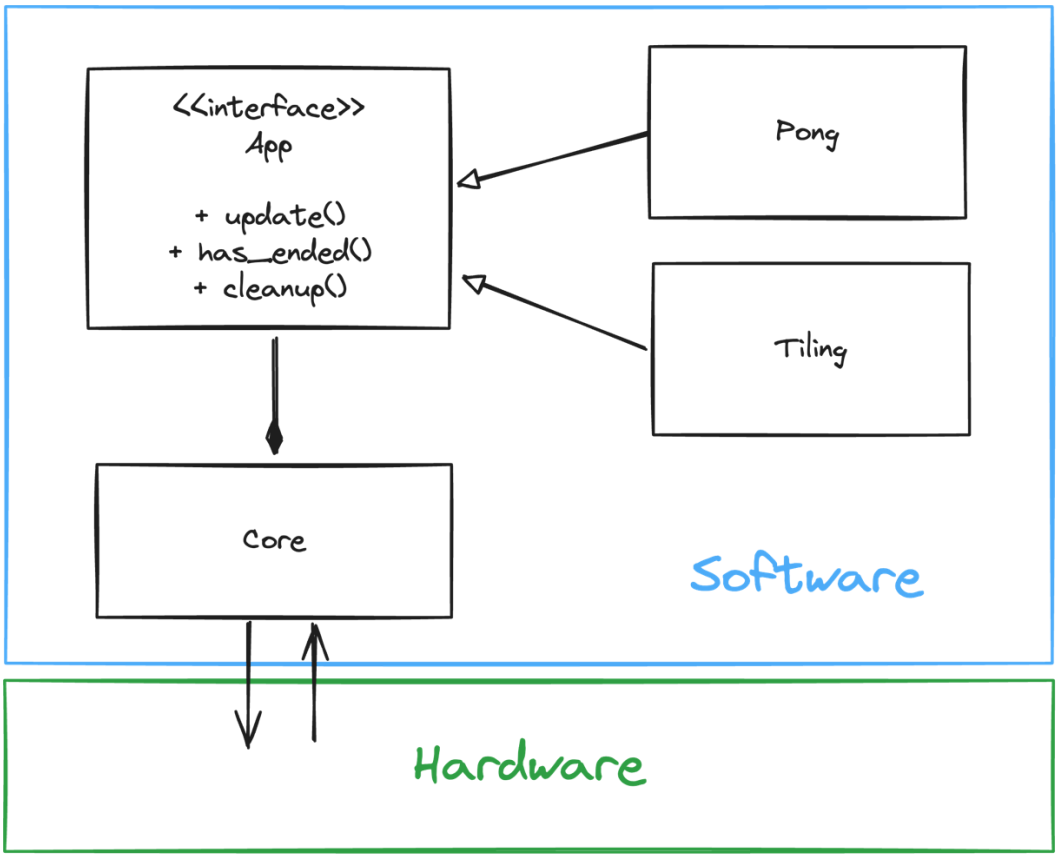


Figure 24 TinkerPod firmware v2 software model

The application is only concerned with its state and needs to expose two functions to the firmware to work. These two functions are update and draw. Update should be called to mutate the

application state (if necessary) every tick, while draw tells the firmware how to render the application UI to the screen controller.

With this tweak to the firmware structure starting from TinkerPod v2 the firmware can support user written applications. To support these applications, the user needs to add the code file to the device and configure the application in the main function of the firmware. Even though having an application store or loader might seem like a better implementation, this would go against the spirit of the project, trying to encourage the audience to dive deeper into the intricacies of the device.

While the approach to this is unconventional, offering full control of the hardware from the software side and allowing to easily tap into any of the software controls available to the device. This means that the user owns the device, having access to the high-level and low-level APIs and allowing extension and hacking of the device in all possible ways.

The PCB for TinkerPod was created to integrate 3 main components, the microcontroller, the buttons and the charging plus battery circuit. Despite the QT PY microcontroller and the BFF add-on boards being meant to be soldered together, this was not space efficient and did not allow for access to the on/off switch. For this reason, the board was integrated to the PCB alongside the buttons which were strategically placed in the corners of the device.

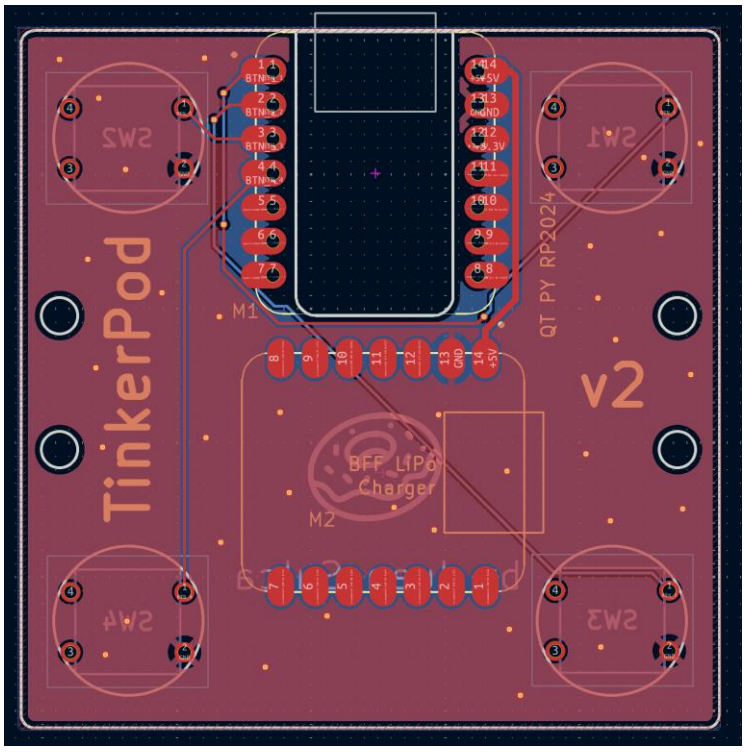


Figure 25 Kicad PCB design for TinkerPod v2

This first version of the PCB focused on simplicity and integrating the platform together. The design of the PCB was kept to a two-layer PCB; the cheapest and most accessible PCB

commercially available from multiple manufactures, while allowing for fewer trade-offs compared to a single layer PCB.

Version 2 of TinkerPod consolidated the form factor and aesthetic qualities of the device while making slight improvements. The black box design stayed as a reference to the mysterious nature of everyday tech while keeping a clean and minimalistic look. With a more robust platform, and a clear direction related to the hardware and software approach.

4.5.3 v3 TinkerPod

Figure 26 TinkerPod handle case attached to a backpack

The objective of version 3 was refining everything that was implemented on v2. Moving responsibilities to the hardware and improving the firmware capabilities.

Like the change from v1 to v2, the change to v3 kept a similar form factor but brought changes to the 3D printed case. These changes included removing the 3D printed button to improve the manufacturing of the case. The issue with the 3D printed buttons was the friction caused by the layers lines, making the buttons get stuck. An tempt to fix for this was to print the buttons with a different layer height, for example if the case was printed with a 0.2mm layer height, the button could be printed with 0.1mm layers. This mitigated the problem, but the friction was still there and the buttons tool longer to print so the buttons were removed for v3.

This version also included a hardware update, integrating the charging circuit to the PCB and adding a debounce circuit for the buttons. This change was done to make assembly of TinkerPod easier, requiring soldering the microcontroller, buttons and a power switch to the PCB assuming the v3 PCB is manufactured by with all the SMD components in place. The PCB can still be fully assembled by the user, but this is harder due to the tiny size of components yet still possible. See Appendix C: TinkerPod v3 PCB assembly guide.

The Bill of Materials (BOM) for the PCB is detailed on Table 5, it includes a description of all the parts needed and the making on the PCB for the component and the manufacturer and part number sourced for v3. This can be used to manufacture an assembled PCB.

Qty	Value	Package	Designator	Part number
4	.1uF, ceramic, 25V, 10%, X5R	1210	C1, C2, C3, C4	KEMET - C1210X104K1RACTU
2	10uF, ceramic, 16V, 10%, X5R	1210	C5, C6	KEMET - C1210C106K3RACAUTO
1	LED	1206	D1	Broadcom - HSMG-C150
1	Schottky Diodes, 1A, 20V	SOD-123	D2	onsemi - MBR120ESFT1G
1	B2B-PH-SM4-TB	B2BPHSM4TBLFSN	J1	JST - B2BPHSM4TBLFSN
4	1K, 1/4W, 1%	1206	R1, R2, R3, R4	Vishay / Beyschlag - MCA12060C1001FP500
2	5.1K, 1/4W, 0.1%	1206	R5, R6	Panasonic - ERA-8ARB512V
1	MCP73831T-2ACI/OT	SOT-23-5	U2	Microchip Technology - MCP73831T-2ACI/OT
1	DPDT	JS202011AQN	S5	C&K - JS202011AQN

Table 5 Bill of Materials TinkerPod v3

See Appendix C: TinkerPod v3 PCB assembly guide

With the PCB assembled, the parts needed for this v3 are less compared to v2.

Qty	Item	Description

1	Adafruit QT PY RP2040	Can be replaced with XIAO pinout boards
1	Adafruit 1.5" Gray scale OLED	SSD1327 driver chip, SPI/I2C interface
1	TinkerPod v3 PCB	
1	STEMMA QT/Qwiic cable	50mm long
4	Push buttons	8mm rubber dome push buttons
8	M2 machine screws	M2 size bolts (4mm x2, 6mm x6)
1	TinkerPod v3 3D printed case	

Table 6 TinkerPod v3 parts list

See Appendix D: TinkerPod v3 assembly guide

The same decisions made for v2 apply for v3, this was done to make possible to move from v1 to v3 with by getting the corresponding PCB and matching case. The last hardware addition to version 3 was a vibration sensor to detect when the user shakes the device, adding another layer interactivity to the device that was complemented with the software.

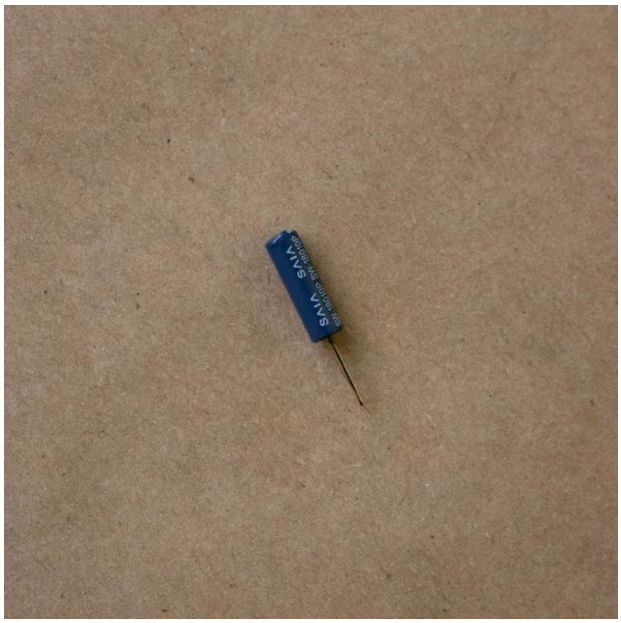


Figure 27 SW-18010P vibration sensor

The vibration sensor (Figure 27) added is a mechanical switch that is wired like a push button. This decision was taken to add the shake device functionality without adding the hardware and code complexity of using gyroscope, that is technically more elegant solution but adds complexity in

wiring and code for a simple functionality like shake detection. Yet the vibration sensor is a simple, durable and cost-effective solution for shake detection.

With the firmware from v2, more complex applications that required update the application state faster or slower were hard to implement. Originally, the interface of the applications was meant to use the class constructor as the *setup* function, the *update* function runs every frame, the *has_ended* functions returns true to signal termination of an application and *cleanup* is the teardown function.

This precented limitations for functionality like the timer, that relied on displaying a an 8x8 matrix to filled tile by tile along the duration of the timer. For this, a tile needed to appear at a random position of the matrix on a time delta different to the frame time (1/60s) and calculated based on the duration of the timer.

The solution was simple, instead of going back and forth between the menu and each application, the application takes control of the device and finishes when the *run* function returns. Now each application can decide when it need to refresh the screen, read the inputs and update the screen. And the menu is responsible of instantiating the applications, calling *run* and finally executing the *cleanup* function. Figure 28 presents the new structure for the firmware.

This change allowed for flexibility in how applications can behave and removed the dependency on flags to manage the application state making the firmware easier to maintain and debug.

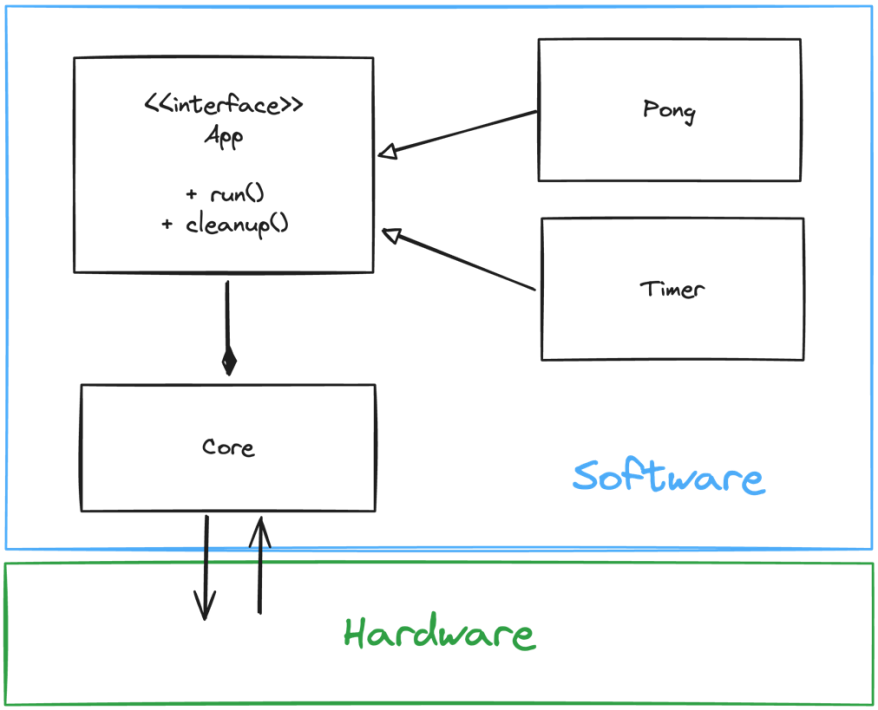


Figure 28 TinkerPod v3 software model

Another addition to the firmware was the introduction of sprites for the menu and applications. Circuit python allows the use of sprites through indexed image files. These sprites can be easily loaded and displayed on the screen and improved the visual quality making the device feel more polished. The additions of sprites allowed to fully use the 16-color grayscale display of the device and simplified rendering of elements compared to drawing every element programmatically.

The sprites for TinkerPod were designed using Aseprite, an application specialized on pixel art drawing and animation. This application was chosen for its workflows and ease of use for sprite sheets; it is possible to make different sprits and then export everything as a sprite sheet eliminating empty spaces and merging identical tiles to save space.

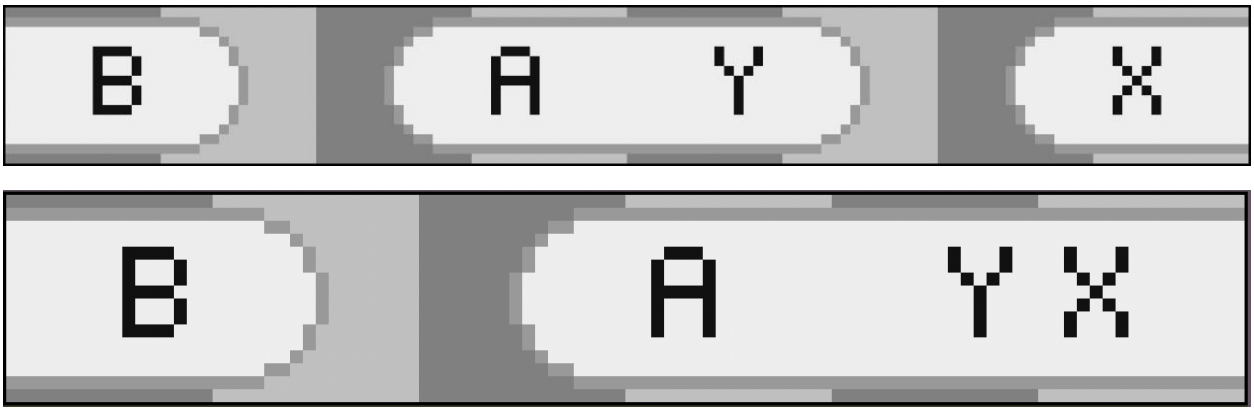


Figure 29 TinkerPod v3 menu sprites

The tiles for TinkerPod are 16x16 pixels using the 16-color grayscale. On the the firmware side, this was a matter of updating the code for the menu to load the sprite sheet, and instead of drawing a rectangle and text for the button, a tile grid with the corresponding tiles for the button are displayer. A map with the corresponding tiles for each button is stored as part of the firmware. Figure 30 shows a mock-up of the menu screen.

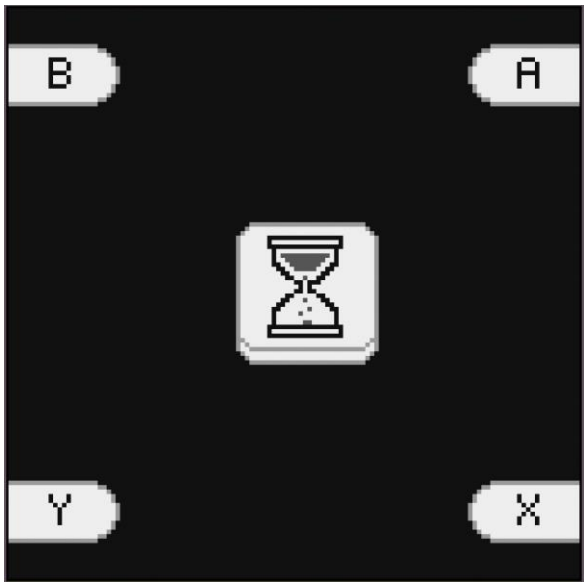


Figure 30 Menu screen mock-up

With the purpose of testing the capabilities of the platform, one more application was implemented using the newly added features, this application was inspired by the classic Etch A sketch using the 4 buttons on the TinkerPod for moving and the shake detection for opening a menu to access a pen up/pen down function, clearing the screen, getting back to drawing or closing the application. Another addition to this application was a marker to tell the user their current location in the canvas. Making it easier to move around and draw figures.

The final steps for v3 firmware were retrofitting the new applications and redesign to v2 as a firmware update. The goal was to have similar functionality due to the similar hardware between both versions. The only exception to this is v1 that aims to be a more approachable and easier to assemble version and due to the limited interaction capabilities v1 has with the optional vibration sensor.

During this phase, the case was redesigned to accommodate the new PCB and battery. The starting point was the v2 case, keeping a similar form factor and improving elements like adding a chamfer to the screen edges to stylize its shape and add the opening for the on/off switch. Another addition was the battery bracket to separate it from the rest of the component and prevent damage when the device is shaken. Figure 31 shows an exploded view of the v3 case with all the internal components.

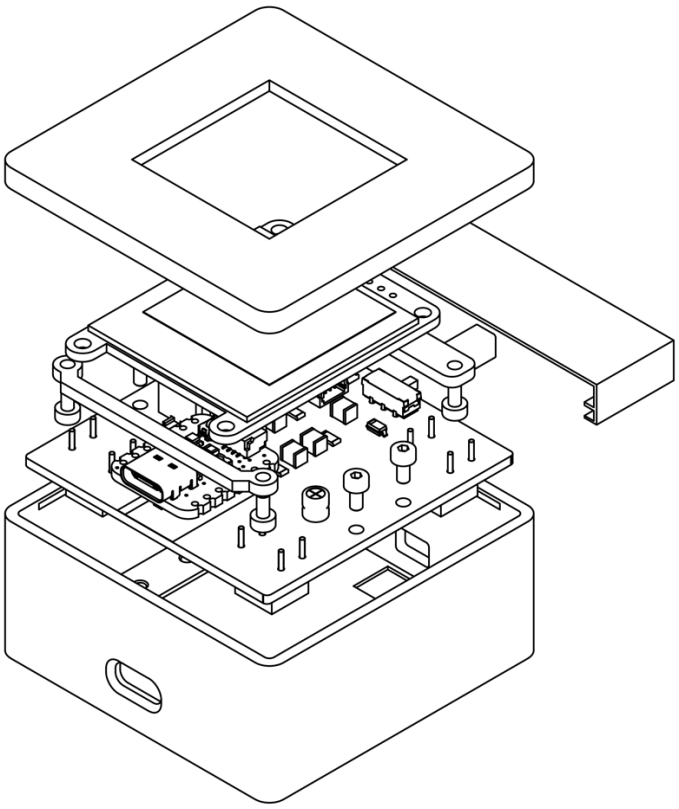


Figure 31 TinkerPod v3 exploded view

The final exploration for this phase was to design different case alternatives for the device. This was done to verify that it is possible to remix the case design from the original CAD files. Two alternative cases were made for this, one with a simple mounting point to hang the device with a keychain and another one with a handle that can be used to hang the device or manipulate the device from the handle (Figure 26).

This experiment also showed the unexpected possibilities of the devices, like powering the device with a phone or power bank and completely removing the battery and making the device slim and lighter. Figure 32 illustrates this possibility.

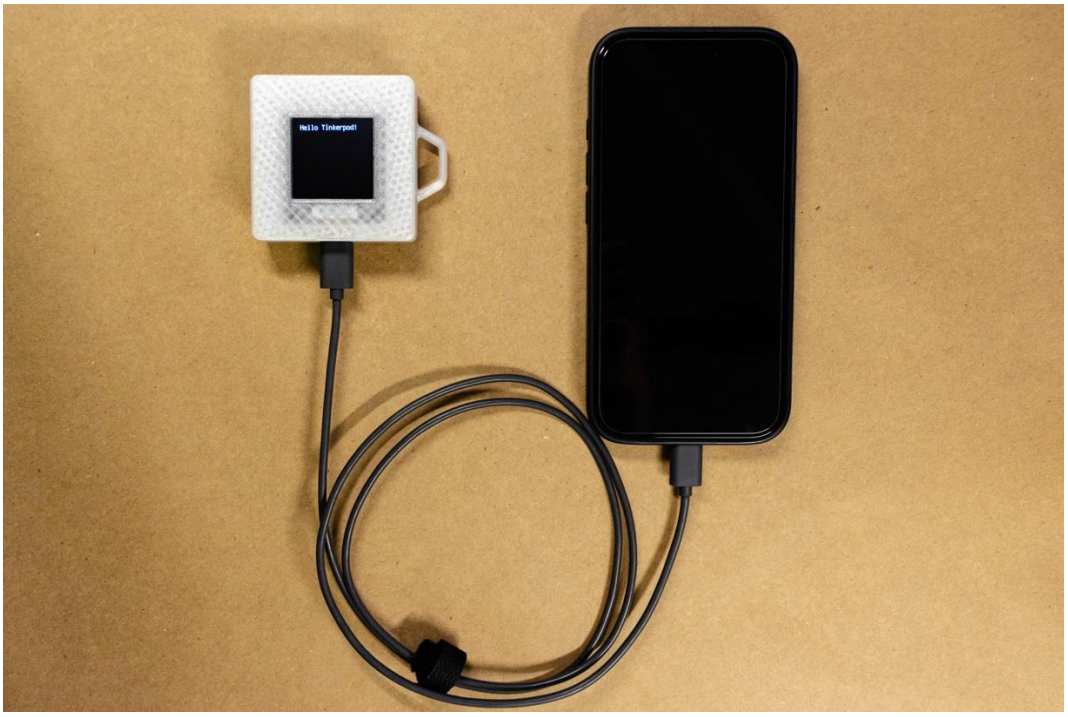


Figure 32 TinkerPod with slim case powered by an iPhone

This experiment and final version of TinkerPod made the device feel more finished, robust and ready to face the world. The changes made to the device turn a flexible DIY platform into an engaging device that feels more polished and complete. The improvement specially on the firmware side, documentation and code examples aim to attract adopters and keep growing and improving the documentation.

4.5.4 Documentation

For the documentation, GitHub was used to host a set of Markdown files in combination with the assets for the project like 3D models, CAD files and the source code. Despite GitHub being the go-to place for open-source software project, hosting and sharing code; the platform offers a powerful tool for hosting documentation and the assets publicly and completely free. Figure 33 shows the current state of the GitHub repository for the project.

Another added benefit for using GitHub is Git itself. The use of version control helps organize and keep track of different changes specially the ones related to text files, like source code. Only on binary files, Git falls short, where most of the binary files (i.e..: .dwg) are just replaced with the newer version. Fortunately, most of the files used in the project can be version controlled using git effectively.

The final benefit of using GitHub is the possibility to connect with the community and keep track of issues with the project and get external contributions from people around the word. The only downside of using GitHub is that it is not the most user-friendly platform for people outside of

coding. While this project expects its adopters to have some experience, it does not expect them to have high technical expertise in using Git.

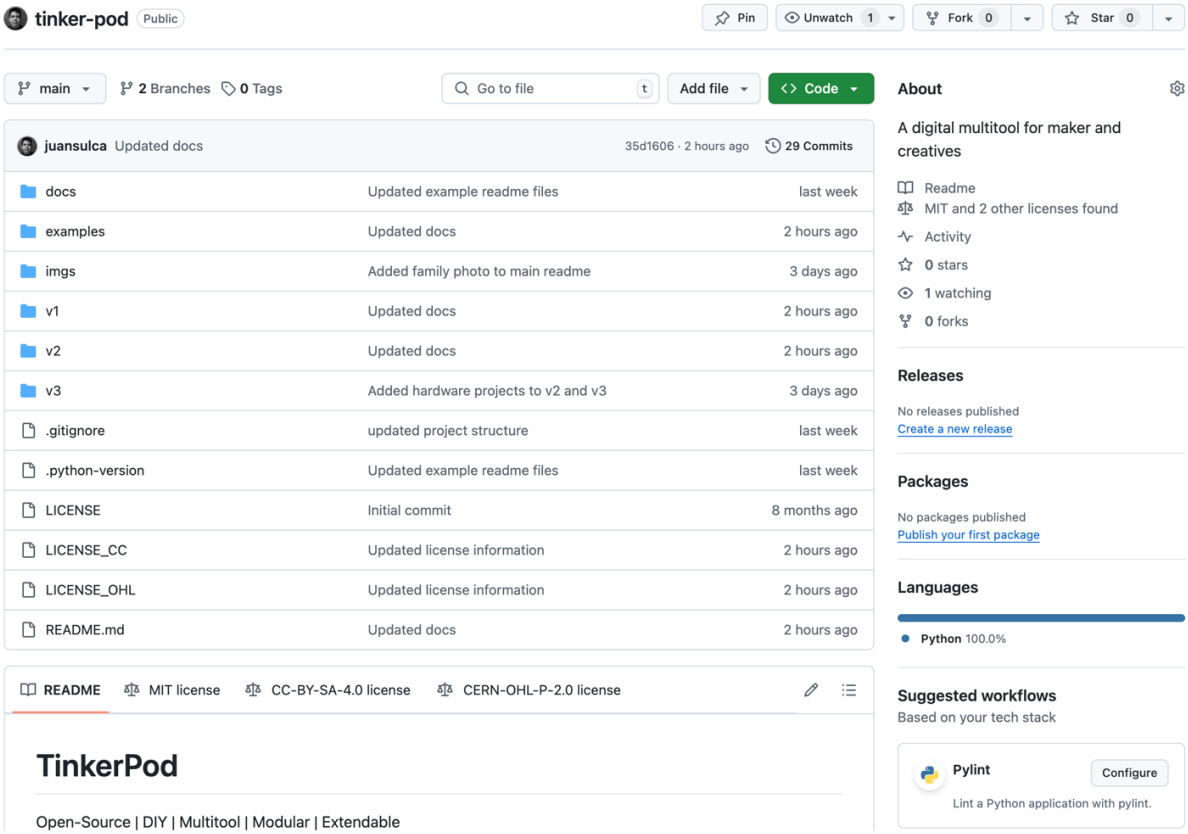


Figure 33 GitHub repository for the project

The documentation for the project is organized by versions of TinkerPod. Each version folder contains a firmware directory with the code for that version, a case directory with the CAD files and production files for the 3D printed case. And a hardware folder with the CAD files and production files for the PCB, the only exception to this is version 1 that does not have a PCB thus this folder is not present.

4.5.5 OSHWA certification compliance

The TinkerPod project aims to comply with the Open-Source Hardware Association (OSHWA) certification program (OSHWA 2018). This certification requires clear specification of open-source components, attach of software, hardware and documentation licenses and public availability of all design and source files.

For software compliance, the TinkerPod firmware and code examples are licensed under MIT, the same license used by Circuit Python. This license grants freedom to use, modify, copy and distribute the software, including both commercial and in proprietary applications.

The hardware components are licensed under the CERN-OHL-P-2.0, which provides users the freedom to make, use, sell and import hardware designs, including the use for commercial use. While this license requires attribution of the original work, it does not require sharing modifications or derivatives.

The project documentation is protected under the Creative Commons CC-BY-SA license, granting the user the freedom to use, modify, and distribute the documentation. This license allows commercial use and requires appropriate attribution and the sharing of any derivatives under the same CC-BY-SA terms.

All the licenses chosen for this project have been approved by the Open Source Initiative (OSI) and are recommended by OSHWA for certification (OSHWA 2018; "The MIT License – Open Source Initiative," n.d.). These permissive licenses were selected maximize accessibility and engagement with the project.

To minimize dependence on proprietary components, this implementation is based on open-source designs from Adafruit and the OSHWA-certified Soldered Inkplate 2 (HR000071) for the charging circuit. The switch debounce circuit implementation was inspired by a Hackaday blog post (Williams. Elliot 2015). Except for the battery management integrated circuits, the design use widely available standard components such as capacitors, resistors, and diodes.

TinkerPod was certified by OSHWA on April 11th, 2025, with OSHWA UID CA000064 (Figure 34).

Figure 34 OSHWA certification mark for TinkerPod

5 Lessons and future work

The journey of designing and fabricating DIY devices presented significant challenges. While creating a device using off-the-shelf components, a 3D-printed case, and a PCB may seem straightforward, extending it to be reproducible, robust, and appealing to makers, creative coders, and designers outlined unique considerations. The process involved learning new tools and skills, as well as creative problem-solving. Each iteration of TinkerPod helped shape and refine the platform's aesthetics, functionality, and design philosophy.

TinkerPod's design prioritized accessibility across different skill levels and durability while balancing appeal to creative coders, makers, and designers. This was achieved by approaching TinkerPod as a platform for people to use and not just as a project. Durability was built into the into the system at every step, avoiding minor compromises that could add up to a major compromise (compromise drilling). Robustness was the driver for decision-making and was not frequently compromised.

The emphasis on robustness influenced design choices that enhanced modularity, such as using connectors instead of soldered wires and screws instead of hot glue. All these decisions mattered, not at an individual level, but at a system level. While modularity supported repeatability, it could increase complexity rapidly. Finding the optimal balance between modularity and complexity while maintaining repeatability was a decisive factor for the project's outcome.

To make TinkerPod hackable and extendable by people with different technical skills and knowledge, the design emphasized transparency, modularity, and accessibility. Transparency was achieved by providing documentation, including assembly guides, schematics, and code examples, which showcased the device's inner workings and enabled experimentation. Modularity played a key role, offering multiple options and entry points, such as version 1, which is built to be simple, low-commitment and solder-less in its assembly. The combination of hardware, software, and the 3D-printed case offers users options for customization and extension. They can choose to extend all aspects or just focus on a single one. The platform is flexible and can be customized by using a different 3D-printed case or sprites for the software.

5.1 Designing for DIY

The development of TinkerPod was challenging. Things did not always turn out as expected, and there are many lessons, especially for a software engineer learning to build hardware. Simple things, like placing a capacitor and a battery connector, may seem trivial, but you will never forget the moment when a tall capacitor blocks the way of a JST connector.

During this exploration, there were no instructions or assembly guides to produce a TinkerPod. There were tutorials, code examples, and open-source hardware to study, but no defined way to find out how to design your own DIY kit. The following points outline the ideas and approaches

that worked to solve some of these challenges and turn off-the-shelf components, ideas, and a vision into a DIY device.

Start simple

Designing TinkerPod v3 from the beginning would be almost impossible; the complexity of the system is too high to solve all at once. Starting with a code example to set up the screen and connect the screen to the microcontroller is significantly simpler but still a step toward the goal. The approach for most feature of TinkerPod involved making small experiments or prototypes and build from them. Moments like starting with a button that changes the size of letters on the screen become the foundation for supporting buttons. Creating a PCB to connect two development boards together becomes the starting point for a fully integrated PCB.

Documentation is more than building guides

While build guides are useful, and schematics and source files are essential for an Open-Source Hardware device, other resources—like the design philosophy and the criteria that informed decision-making during the project—can also be valuable for people seeking inspiration or ideas on how to solve similar problems. Code examples and comments provide valuable insight into how a codebase works and may inspire others to build something from those examples.

Design for DIY methods and tools

Understand and design for the fabrication process used. This means acknowledging that you are not designing for methods like injection molding, so you don't need to follow its constraints. For example, TinkerPod use 3D printing, so it is not necessary to add holes in the structure to reduce the amount of material, as FDM 3D printing typically produce semi-hollow objects, it is possible to control material properties by adjusting the infill percentage and pattern.

When possible, remove reliance on slicers for successful completion of a print. Try to make the process as simple and easy as possible for adopters. Avoiding slicer-specific setting increases accessibility for makers. While all slicers allow adjustment to layer height and infill percentage, not all support features like scarf seams. Designing supports as part of the 3D model can improve the building experience and prevent issues where someone forgets to enable supports in the slicer.

Simple, not simpler

Earlier in my career, I learned that the trick to writing great code is making things simple, not simpler. This doesn't mean reducing the complexity of a problem but solving it in a way that does not introduce unnecessary complexity. This principle applies to both hardware and software, whether it involves choosing the right level of abstraction for the code or designing around form factor and ease of assembly.

During the development of TinkerPod, the idea of using a daughterboard to hold the USB-C port emerged during ideation for v2. This idea was quickly discarded due to the unnecessary complexity it introduced. Instead, the solution of increasing the size of the device was favored,

despite adding a larger bezel to the device. This compromise was acceptable, the difficulty of making a case bigger was significantly lower compared to design, manufacturing, and assembly of two PCBs.

Caring makes the difference

Especially for code, caring makes a difference in the outcome. Constantly think about the people who will change the code, and be consistent with the small details, like variable names and code style conventions. Writing code is an iterative process and a conversation with the next person who will modify it; that person could be your future self. Finally, consider the experience you want people to have when they change your code and the overall developer experience (DX) around it.

Iterate, Iterate, Iterate

During the development of TinkerPod, multiple ideas and designs were tested for things like PCBs, enclosure design, and firmware. Most of these ideas were tested with a mindset focused on improvement and iteration of prototypes, avoiding attachment to a single idea or implementation. At the same time, it is important to not only think about the next iteration but also reflect on the current implementation and the details of the prototype.

5.2 Future work

During the development of TinkerPod, the user experience of building and extending the software was a significant consideration, which partly influenced the decision to use CircuitPython instead of a compiled language. This choice provided a fast feedback loop from code updates to execution. However, despite this advantage, the development experience lacked modern conveniences common in web development. This limitation inspired the creation of an emulator for TinkerPod using Python to improve the development experience. A preliminary experiment during v3 development used Pyxel to render a 128x128 pixel canvas. This showed potential for improving the developer experience and creating a new entry point to the project, similar to how the Playdate provides an emulator for game developers and the general public to access its ecosystem.

The bulk of time and effort during this research went to the design and fabrication of the TinkerPod platform. It took the device multiple iterations and revisions to refine the usability, functionality, robustness, and building experience of the device. This focus on design and development allowed for the creation of a foundation for the platform that satisfies the design criteria outlined in section 4.2. While TinkerPod was demonstrated multiple times to peers, at a creative coding meetup, and during a class exhibition, exploration of user engagement with the platform and feedback about documentation, build guides, and experience extending the platform is pending. The platform is now ready for the world, in a state where it can be replicated, used, and extended.

To understand who is using TinkerPod, how people engage with the device, how it is being used, and understand what is needed around TinkerPod. This can be achieved by scrapping the Internet for things people used the device for, inspired by (Qi et al. 2018; Buechley and Hill 2010). Through this approach, it is possible to determine if the goal of appealing to people in design, creative coding and making was achieved, identifying community needs, improvements for the platform in future iterations, and understanding creative ways people can extend the platform.

Figure 35 TinkerPod family photo

5.3 Reflection

In the end, all versions of TinkerPod fulfilled different parts of the project's vision. While not perfect and with room for improvement, TinkerPod offers a base layer of functionality that is usable and may appeal to people with various backgrounds and skill sets in technology.

This document encapsulates the experiences and outcomes of exploring and learning how to make electronic devices, from figuring out how to design for 3D printing, to PCB design and manufacturing. Bringing TinkerPod from an idea to a real device was a hard but rewarding journey. Mistakes, like putting a tall capacitor in the way of a JST connector or getting multiple PCBs wrong, taught valuable lessons in how to make designs more reliable and helped consolidate a pipeline to design, test, and manufacture TinkerPod.

TinkerPod is a device designed and built by a single person, without formal user testing or feedback. Its characteristics and affordances reflect the journey outlined in this document, showcasing the exploration and learning involved in building DIY electronic devices from scratch.

User testing, feedback, and experiences from running workshops and getting the device into other people's hands will help guide improvements in documentation and source files, enabling people to study, remix, and make new creations with TinkerPod.

Bibliography

Adafruit Industries. 2025. "CircuitPython Version: 9.2.4." https://circuitpython.org/.

Anderson, Chris. 2014. "Makers : The New Industrial Revolution," 257.

Anderson, David J. 2010. "Kanban: Successful Evolutionary Change For Your Technology Business," 278. http://www.openisbn.org/download/0984521402.pdf.

Autodesk Inc. 2024. "Fusion 2.0.21550." https://www.autodesk.com/ca-en/products/fusion-360/rapid-prototyping.

Bdeir, Ayah. 2009. "Electronics as Material: Littlebits." *Proceedings of the 3rd International Conference on Tangible and Embedded Interaction, TEI'09*, 397–400. https://doi.org/10.1145/1517664.1517743.

Beşevli, Ceylan, Tilbe Göksun, and Oğuzhan Özcan. 2022. "Designing Physical Objects for Young Children's Magnitude Understanding: A TUI Research Through Design Journey." https://doi.org/10.1145/3501712.3534091.

Blender Foundation. 2025. "Blender Version: 4.4." https://www.blender.org/download/.

Bonvoisin, Jérémy, Robert Mies, Jean-François Boujut, and Rainer Stark. 2017. "What Is the 'Source' of Open Source Hardware?" *Journal of Open Hardware* 1 (1). https://doi.org/10.5334/JOH.7.

Bowers, John. 2012. "The Logic of Annotated Portfolios: Communicating the Value of 'Research Through Design.'"

Bretthauer, David. 2001. "Open Source Software: A History." *Published Works*, December. https://digitalcommons.lib.uconn.edu/libr_pubs/7.

Buechley, Leah, and Benjamin Mako Hill. 2010. "LilyPad in the Wild: How Hardware's Long Tail Is Supporting New Engineering and Design Communities." *DIS 2010 - Proceedings of the 8th ACM Conference on Designing Interactive Systems*, 199–207. https://doi.org/10.1145/1858171.1858206.

Charny, Daniel. 2012. "Power of Making." In *Critical Making: Manifestos*, edited by Garnet Hertz. https://www.conceptlab.com/criticalmaking/.

Claris, Lionel, and Donna Riley. 2012. "Situation Critical: Critical Theory and Critical Thinking in Engineering Education." *Engineering Studies* 4 (2): 101–20. https://doi.org/10.1080/19378629.2011.649920.

Fiell, Charlotte., and Peter. Fiell. 2011. "Industrial Design A-Z."

Flipper Devices Inc. 2024. "Flipper Zero — Portable Multi-Tool Device for Geeks." 2024. https://flipperzero.one/.

Hodges, Steve, Sue Sentance, Joe Finney, and Thomas Ball. 2020. "Physical Computing: A Key Element of Modern Computer Science Education." *Computer* 53 (4): 20–30. https://doi.org/10.1109/MC.2019.2935058.

Huang, Andrew. 2019. *The Hardware Hacker : Adventures in Making and Breaking Hardware*. No Starch Press.

Karl Elsener. 1968. "Swiss Officers' Knife Champion (No. 5012)." https://www.moma.org/collection/works/4332.

KiCad Project. 2025. "KiCad Version: 9.0.0." https://www.kicad.org/.

Kirklewski, Ksawery. 2024. "SQRT at RRGGBB." Vancouver. 2024. https://ksawerykomputery.com/works/sqrt.

Kushner, David. 2011. "The Making of Arduino." IEEE Spectrum. October 26, 2011. https://spectrum.ieee.org/the-making-of-arduino.

Leatherman Tool Group. n.d. "The Leatherman Story." Accessed November 7, 2024. https://www.leatherman.com/en_CA/leatherman-story.html?geo=y.

Lees-Maffei, Grace. 2014. *Iconic Designs: 50 Stories about 50 Things*. Bloomsbury Visual Arts.

Mellis, David Adley. 2015. "Do-It-Yourself Devices : Personal Fabrication of Custom Electronic Products." https://dspace.mit.edu/handle/1721.1/101847.

Nicholas Kallen. 2025. "Plasticity Version: 25.1.7." https://www.plasticity.xyz/.

Norman, D.A. 2013. *The Design of Everyday Things*. *Physics*. Revised.

Oliver, Julian, Gordan Savičić, and Danja Vasiliev. 2017. "The Critical Engineering Manifesto." 2017. https://criticalengineering.org/.

Open Source Initiative. 2007. "The Open Source Definition – Open Source Initiative." 2007. https://opensource.org/osd.

OSHWA. 2013. "Brief History of Open Source Hardware Organizations and Definitions." 2013.

———. 2018. "OSHWA Certification Process." 2018. https://certification.oshwa.org/process.html.

———. 2020. "Open Source Hardware Definition." 2020. https://www.oshwa.org/definition/.

Panic Inc. n.d. "Inside Playdate." Accessed February 7, 2025. https://sdk.play.date/2.6.2/Inside%20Playdate.html.

Prusa Research. 2024. "PrusaSlicer Version 2.9.0." https://www.prusa3d.com/page/prusaslicer_424/.

Qi, Jie, Patricia Ng, Leah Buechley, Sean Cross, Andrew Huang, and Joseph A. Paradiso. 2018. "Chibitronics in the Wild: Engaging New Communities in Creating Technology with Paper Electronics." *Conference on Human Factors in Computing Systems - Proceedings* 2018-April (April). https://doi.org/10.1145/3173574.3173826.

Ratto, Matt. 2017. "Defining Critical Making." In *Conversations in Critical Making*, edited by Garnet Hertz. https://www.researchgate.net/publication/320344201_Conversations_in_Critical_Making.

Savage, Adam. 2020. "Every Tool's a Hammer: Life Is What You Make It," 309.

Schwaber, Ken, and Jeff Sutherland. 2020. "The Scrum Guide The Definitive Guide to Scrum: The Rules of the Game."

Sipos, Regina, Saad Chinoy, and Ricardo Ruiz. 2018. "The Critical Making Movement."

Soldered Electronics. 2023. "Inkplate 2." 2023. https://soldered.com/product/inkplate-2/.

"The MIT License – Open Source Initiative." n.d. Accessed March 15, 2025. https://opensource.org/license/mit.

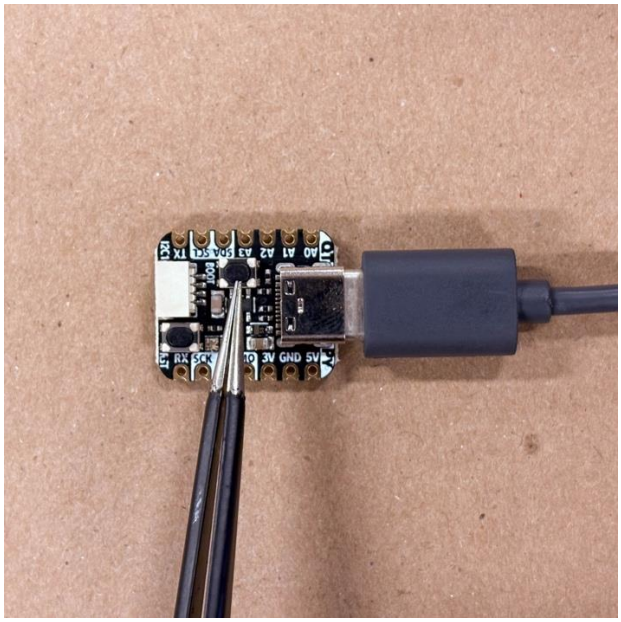
Williams. Elliot. 2015. "Embed With Elliot: Debounce Your Noisy Buttons, Part I | Hackaday." 2015. https://hackaday.com/2015/12/09/embed-with-elliot-debounce-your-noisy-buttons-part-i/.

Zimmerman, John, Jodi Forlizzi, and Shelley Evenson. 2007. "Research Through Design as a Method for Interaction Design Research in HCI."

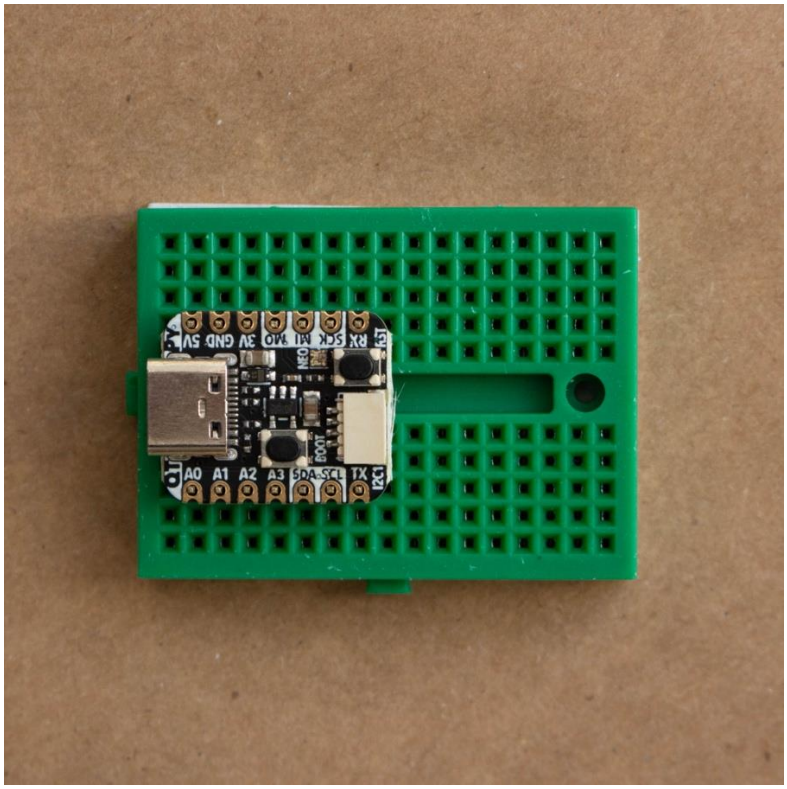
Appendix A: TinkerPod v1 assembly guide

For the latest version of this guide visit: https://github.com/juansulca/tinker-pod/blob/main/v1/README.md

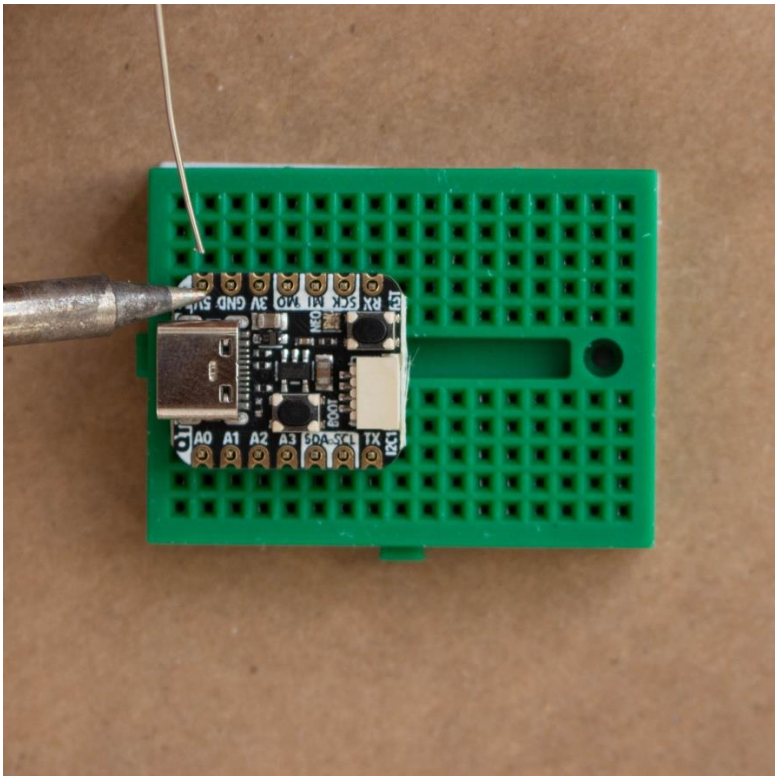
1. Flash CircuitPython on the QT PY RP2040.
    a. Open the official site(https://circuitpython.org/board/adafruit_qtpy_rp2040/) and download the latest version of circuit python.
    b. While holding the *boot* button in the QT PY, plug it into a computer using the USB-C cable. Different version of the QT PY might need a different button to access bootloader mode, please check the official docs(https://learn.adafruit.com/welcome-to-circuitpython/installing-circuitpython) for the specific board.



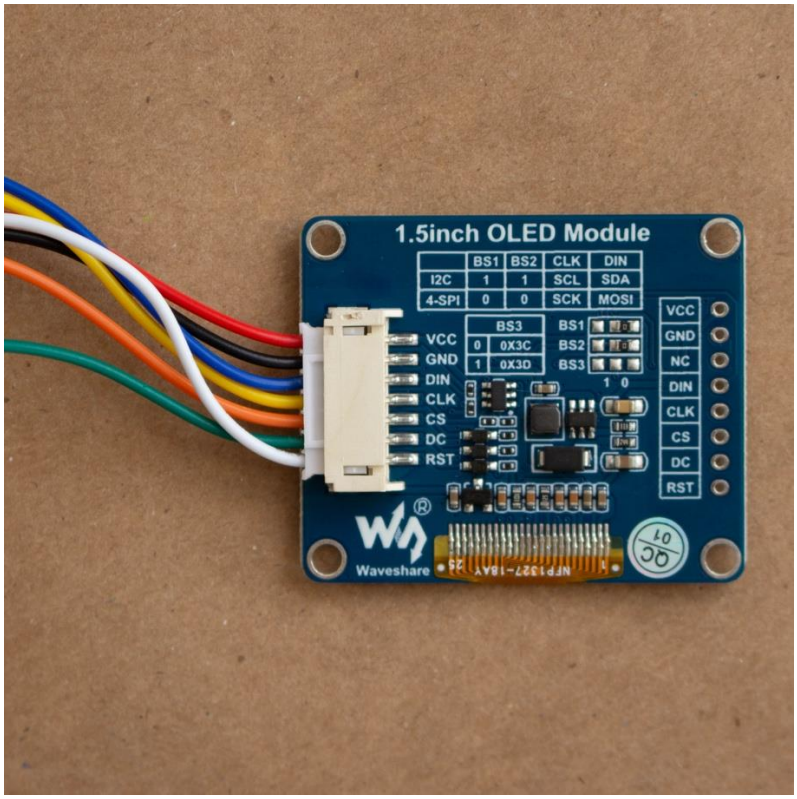
    c. The device should appear mounted in the OS as **RPI-RP2**. This might be different for other boards other than the RP2040 board.
    d. Drag and drop the *.UF2* file (downloaded in step 1) to the *RPI-RP2* drive.
    e. After a couple of seconds, the onboard neopixel (LED) will flash and a new drive will appear in the computer, this time it should be called **CIRCUITPY**.
    f. ⚠ Always eject the device before unplugging the cable.
2. Solder the headers to the QT PY.
    a. Position the headers in the QT PY and carefully place both the breadboard.



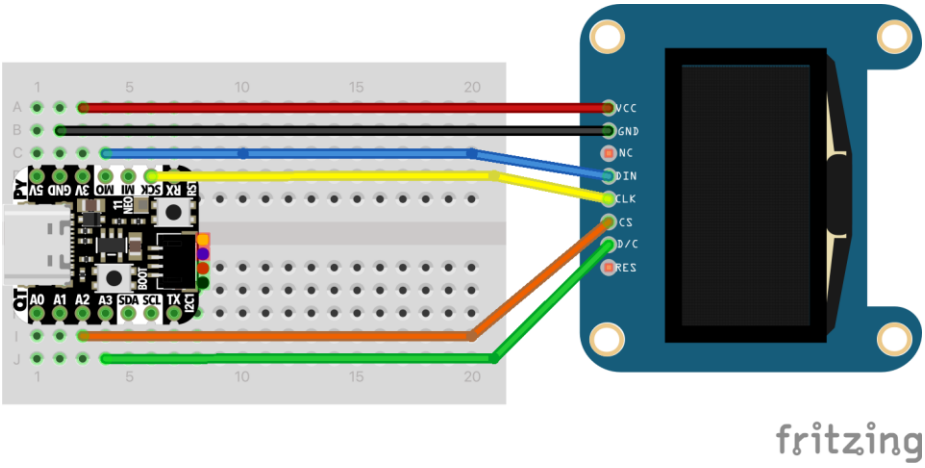
b. Solder the headers to the microcontroller.



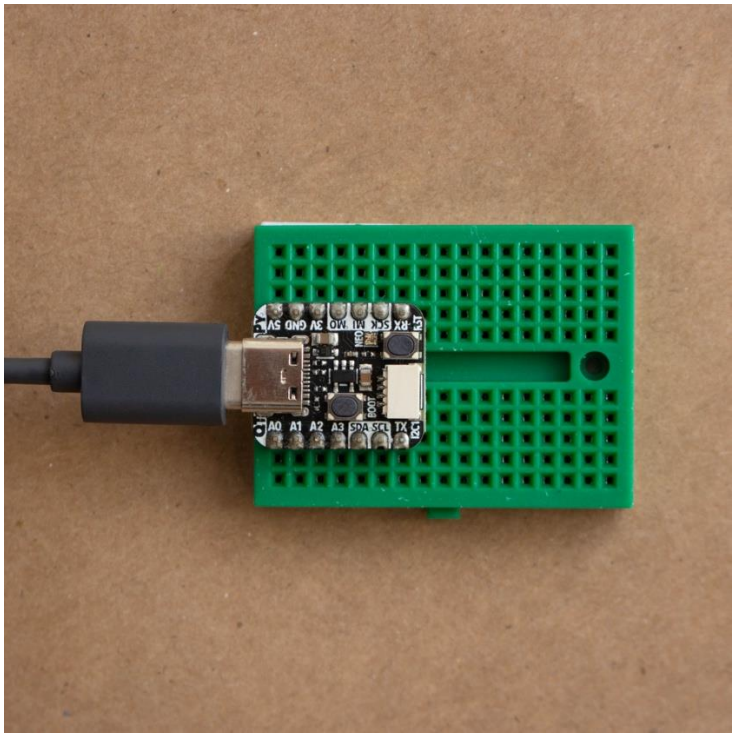
3. Place the microcontroller on the breadboard and connect the screen cable to the screen socket.



4. Connect the screen to the QT PY following this diagram:

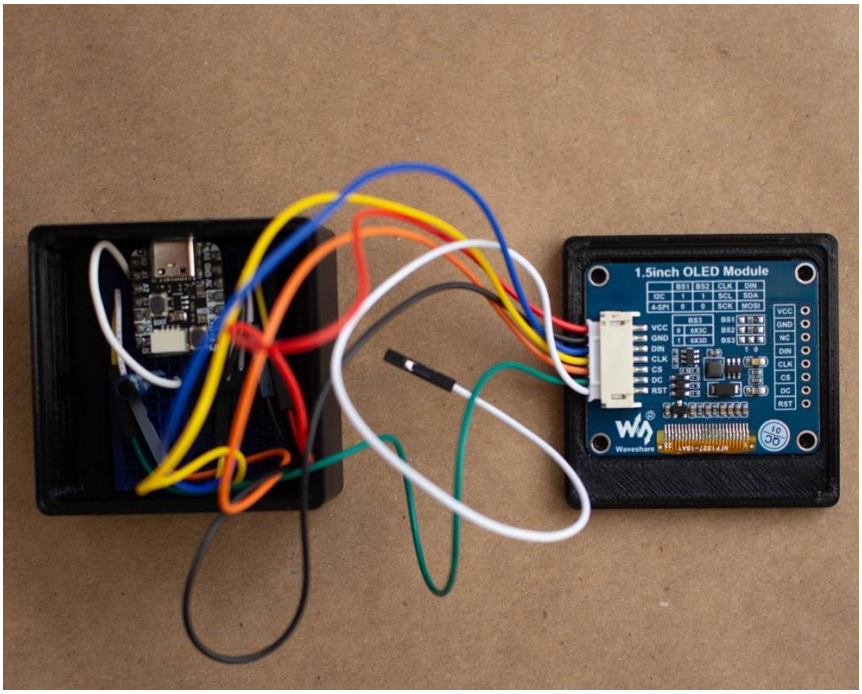


5. Plug the USB-C cable to the micro controller.



6. Check your file explorer for new drive called **CIRCUITPY**
7. Download the *firmware/* directory for this version.
8. Copy all the files from the *firmware/* directory (*code.py* and *lib/*) to the **CIRCUITPY** drive.
9. After the device restarts, look at the OLED screen which will display a generative art function.
10. Place the screen in the top part of the enclosure.

11. Place the breadboard in the bottom part of the enclosure.

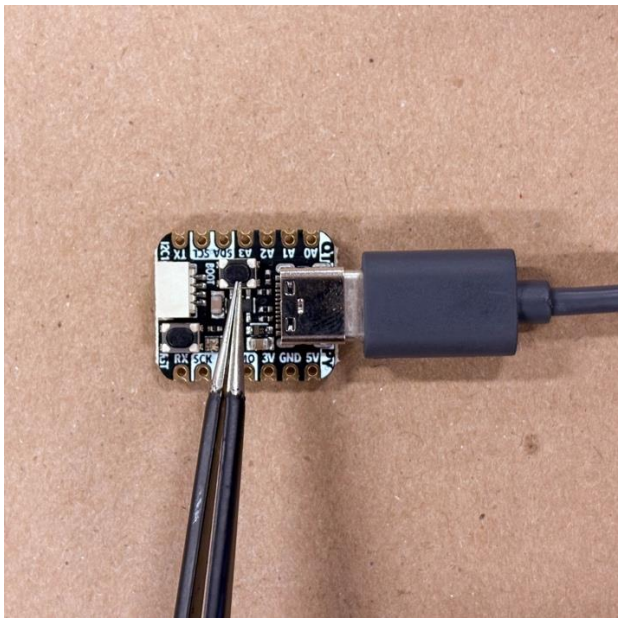


12. Close the enclosure with some tape.

Appendix B: TinkerPod v2 assembly guide

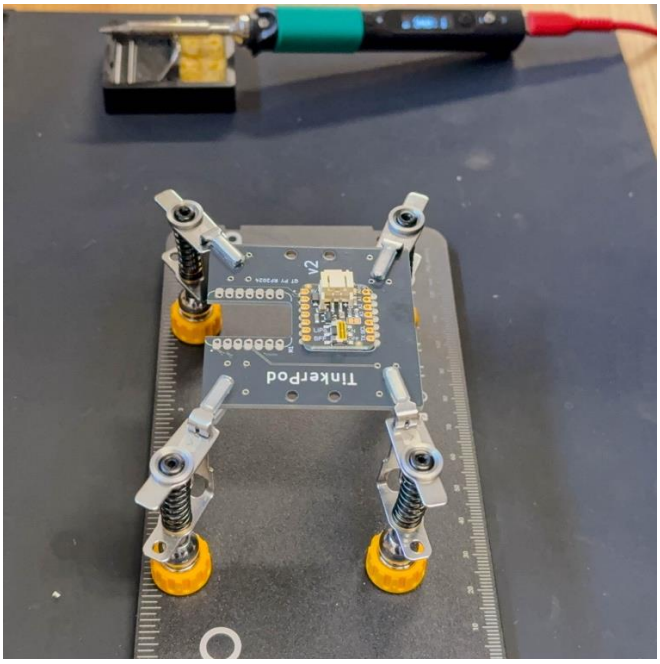
For the latest version of this guide visit: https://github.com/juansulca/tinker-pod/blob/main/v2/README.md

1. Flash circuit python on the QT PY RP2040.
    a. Open the official site and download the latest version of circuit python.
    b. While holding the *boot* button in the QT PY, plug it into a computer using the USB-C cable. Different version of the QT PY might need a different button to access bootloader mode, please check the official docs for the specific board.

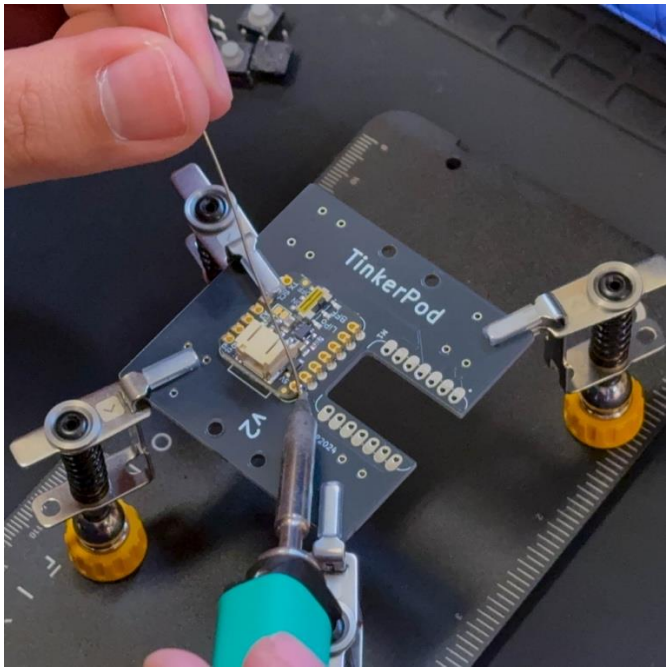


    c. The device should appear mounted in the OS as **RPI-RP2**, this might be different for other boards other than the RP2040 board.
    d. Drag and drop the .UF2 file (downloaded in step 1) to the RPI-RP2 boot drive.
    e. After a couple of seconds, the onboard neopixel (LED) will flash and a new drive will appear in the computer, this time it should be called **CIRCUITPY**.
    f. ⚠ Always eject the device before unplugging the cable.
    g. The device is ready to be installed.
2. (Optional) Solder the Li-Ion Charger BFF to the PCB.
    a. Place the PCB donut side down. The **TinkerPod** and **v2** silkscreen labels should be visible.

b. Carefully line up the BFF with the PCB, in the M2 space, also marked as BFF LiPo charger. The JST connector (white box) should be facing the **v2** text on the PCB.



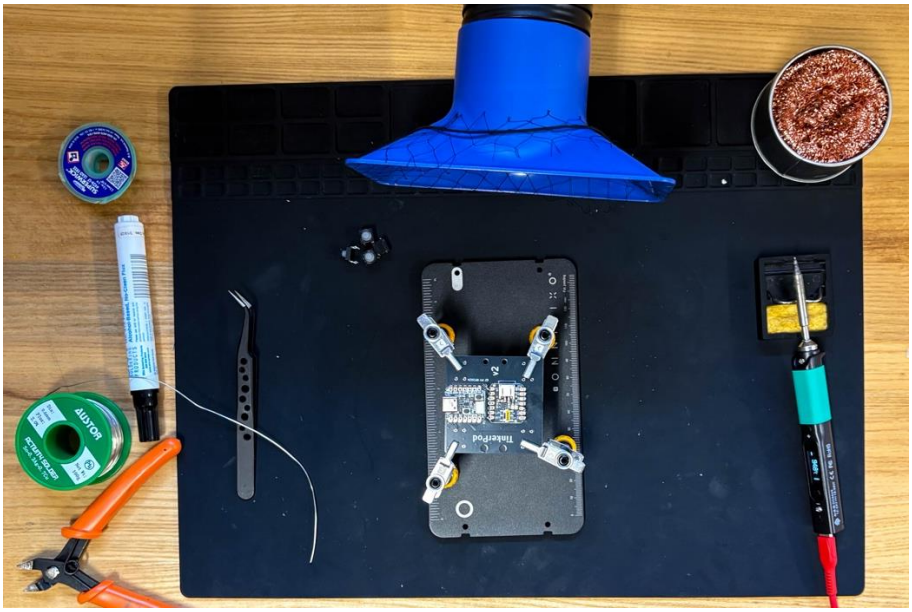
c. Solder all the pads. take extra care on the 5V and GND pads.



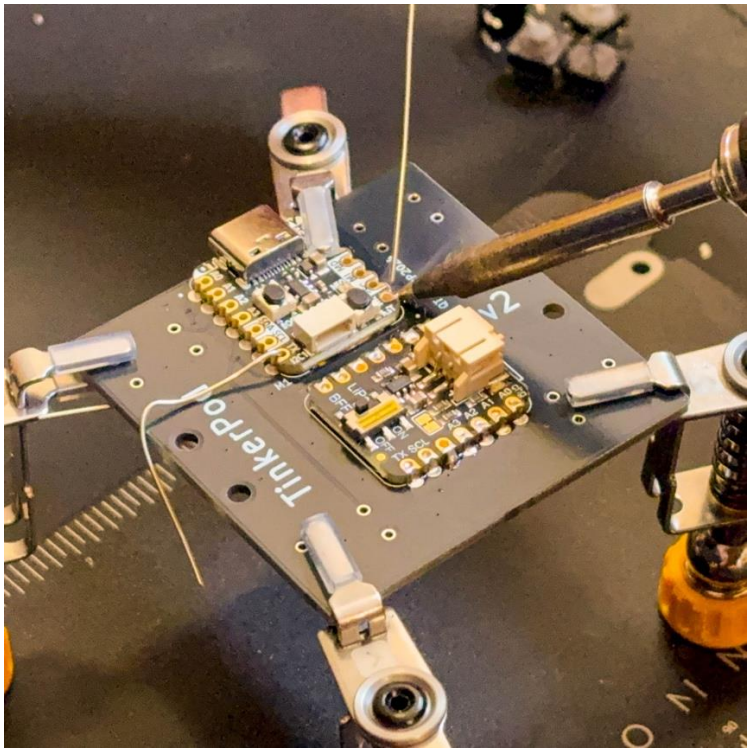
3. Solder the QT PY to the PCB.

a. Place the PCB donut side down. The **TinkerPod** and **v2** silkscreen labels should be visible.

b. Carefully line up the QT PY with the PCB, in the M1 space also marked on the side as QT PY RP2040. The pads can be identified for the cutout in the middle. Make sure the USB port is pointing towards the outside of the board and the reset and boot buttons, and the STEMMA connector are accessible.

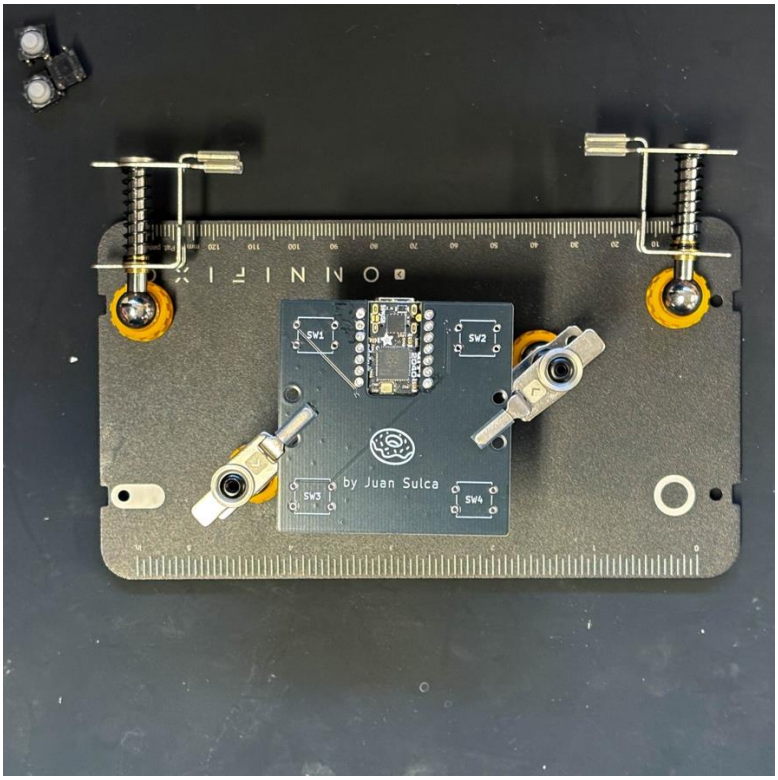


c. Solder the **14** pins, 7 on each side. (No headers are needed)

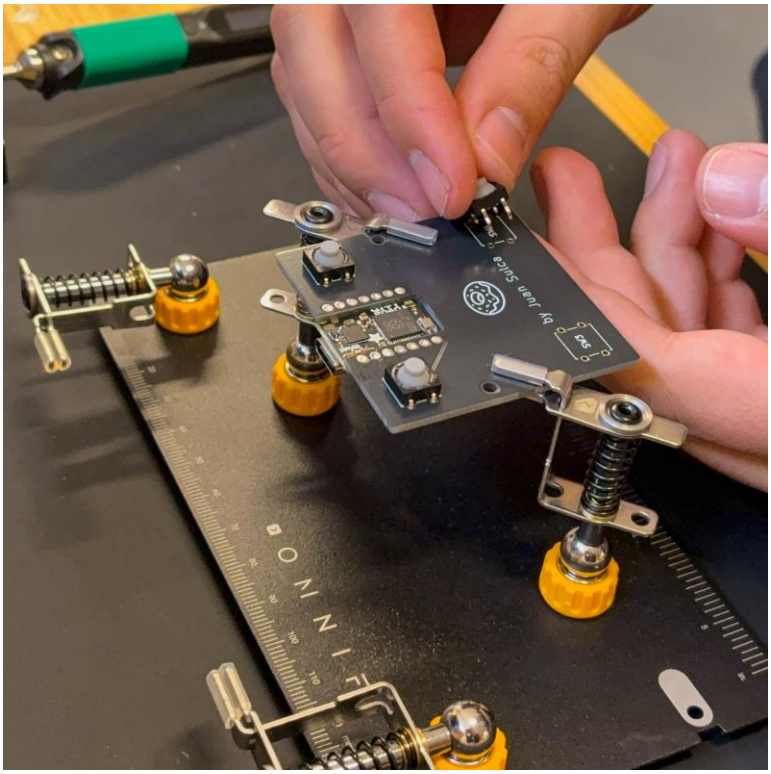


4. Solder the 4 buttons to the PCB.

a. After completing steps 2 and 3, flip the PCB (Donut side up).

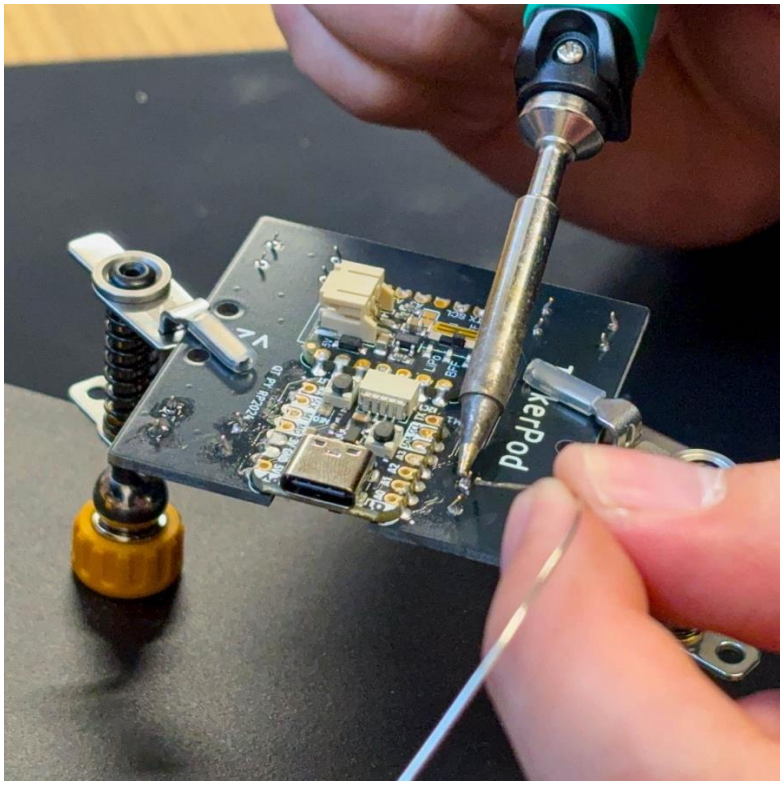


b. Place the buttons in the markers SW1, SW2, SW3, SW4.



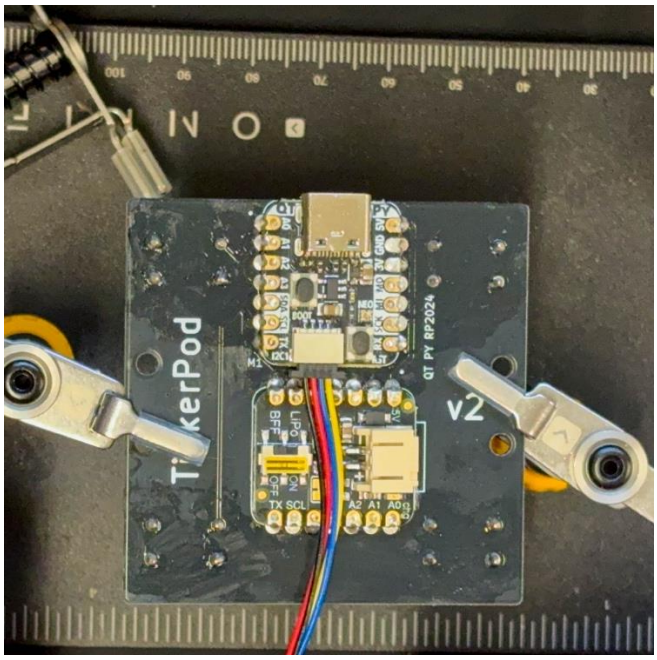
c. Flip the PCB again (Donut side down).

d. Solder all 4 pins of every push button. (16 solder joins in total).

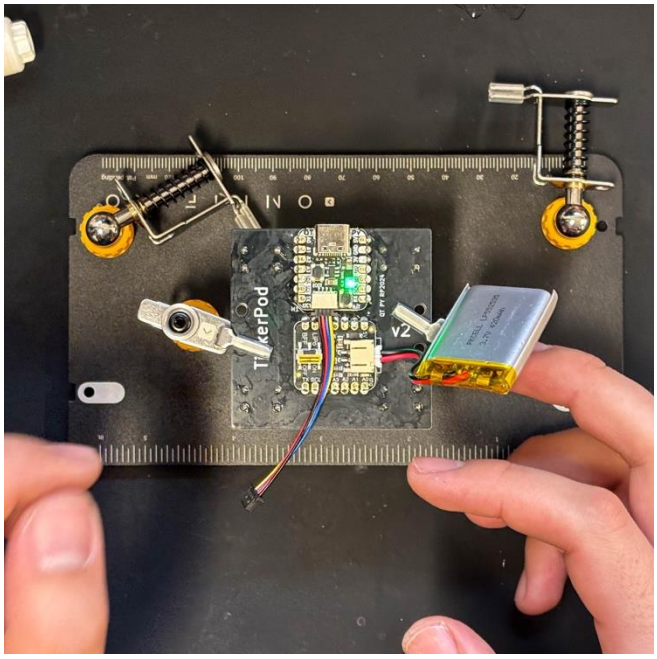


5. Now is a great moment to flash the firmware and test the TinkerPod before installing it into the enclosure.

    a. Connect the STEMMA connector to the QT PY and Screen. It should slide in place without force.



b. (Optional) Connect the battery using the JST connector in the BFF board.



c. Download the TinkerPod firmware from GitHub. (firmware folder)

d. Using a USB C data cable. Plug the device to the computer.

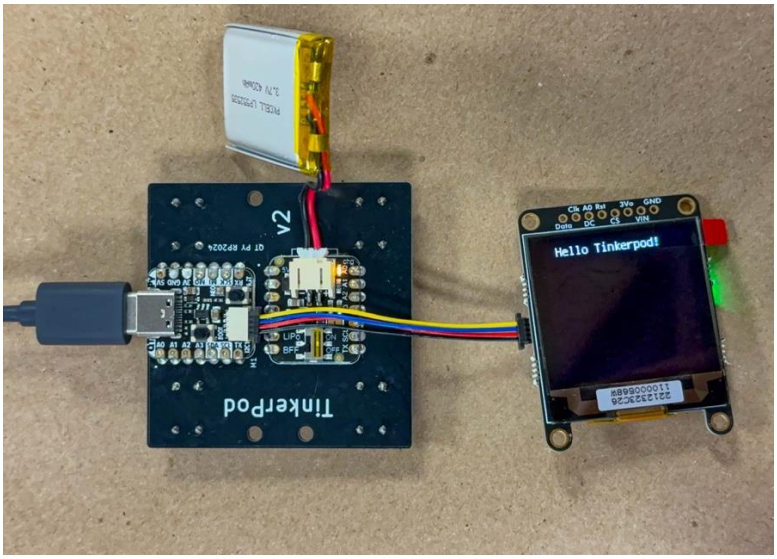
e. Copy the content of the firmware folder (where the code.py file is) to the device.

f. Remember to copy the lib directory.

g. Alternatively check this project documentation for alternative installation methods.

h. The splash screen followed by the menu should appear. Navigate the menu and use the applications by pressing the 4 buttons.

i. The charging indicator (orange LED) in the BFF board should light up if there is a battery connected, and the BFF installed.



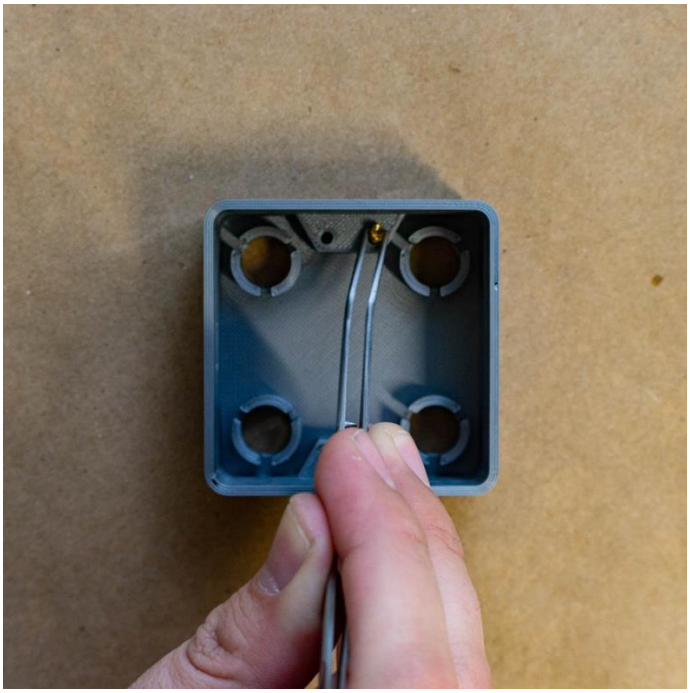
j. If everything works as expected, continue with the next step.

    1. If something is not working:
    2. Check for cold solder joins
    3. Verify that that the firmware was flashed correctly.
    4. If there is red LED flashing, there is an issue with the firmware.

k. Eject the device and unplug the USB cable.

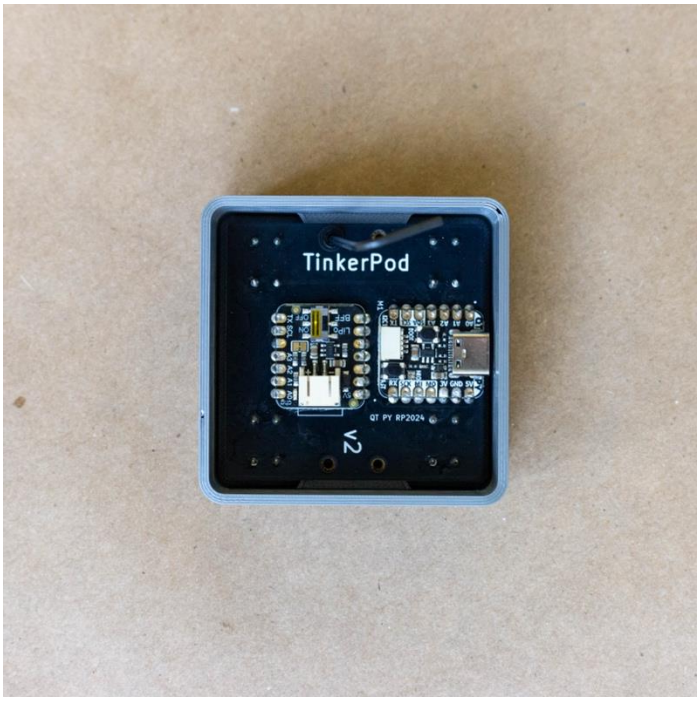
6. Place the 3D printed buttons in the enclosure and make sure they slide smooth.

7. Prepare the bottom enclosure.
    a. Place the threaded inserts in the holes located in the top part of the enclosure.



    b. Using the soldering iron push the inserts as straight as possible until they are flush with the surface of the enclosure.

c. Place the PCB in place (with the USB-C port towards the hole in the enclosure) and screw it in place using the M2 machine screws.



d. Make sure the power switch on the BFF board is in the *off* position.

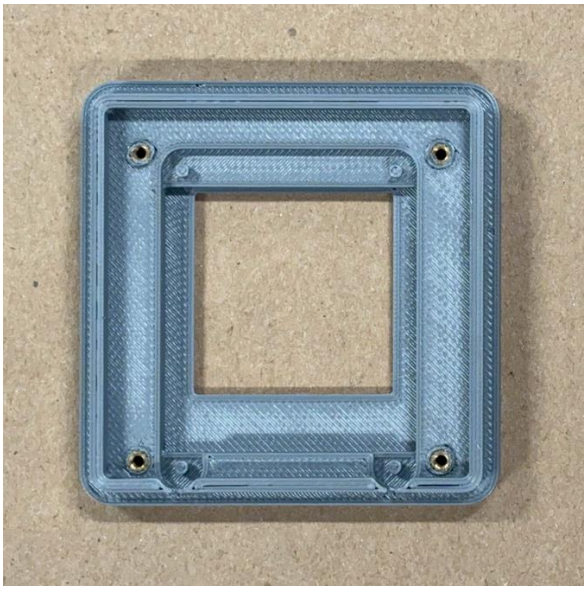
e. Plug in the battery.

f. Secure the battery using the battery bracket and some double-sided tape.

8. Prepare the screen assembly

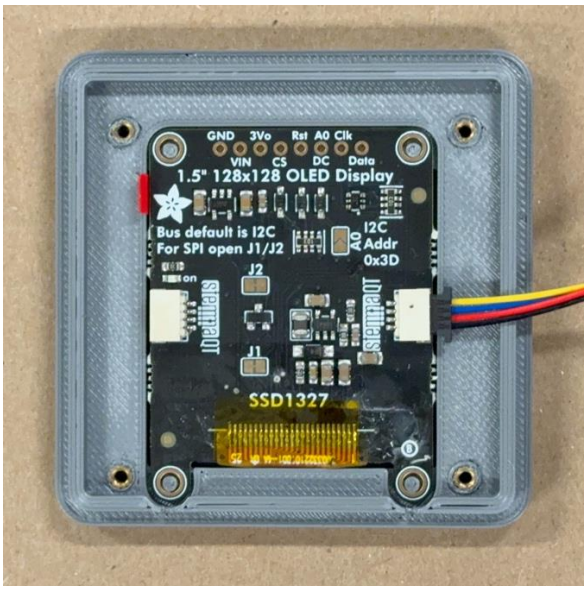
   a. Place the threaded inserts in the holes located in the top part of the enclosure.

   b. Using the soldering iron push the inserts as straight as possible until they are flush with the surface of the enclosure.



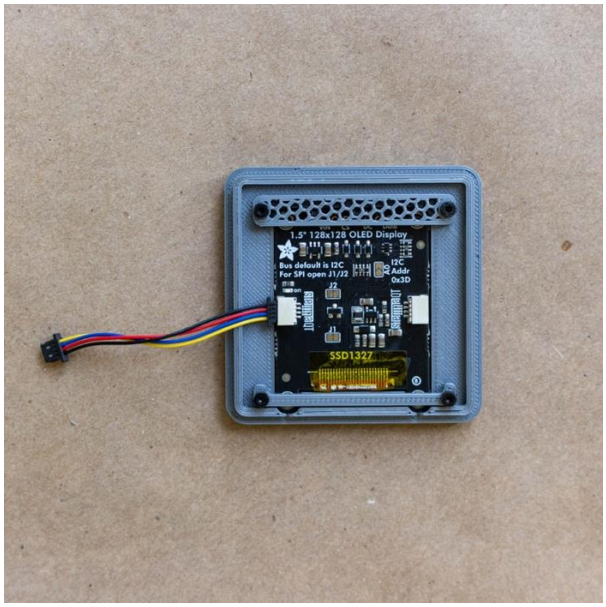
   c. Connect the STEMMA cable into the screen, it should slide in place without force.

   d. Gently place the 1.5" screen into the slot, do not apply pressure on the screen.



e. Place the screen supports and line up the holes. The rectangular piece should go towards the top (can be placed in any orientation) and the remaining piece should go to the bottom. The bottom piece will only line up one way, with the legs pointing to the top.

f. Screw the supports to the top enclosure, tighten the screws until there is some resistance. Do not over tighten the screws.



g. The screen might have some play, or some gaps might be visible from the front side. This is expected.

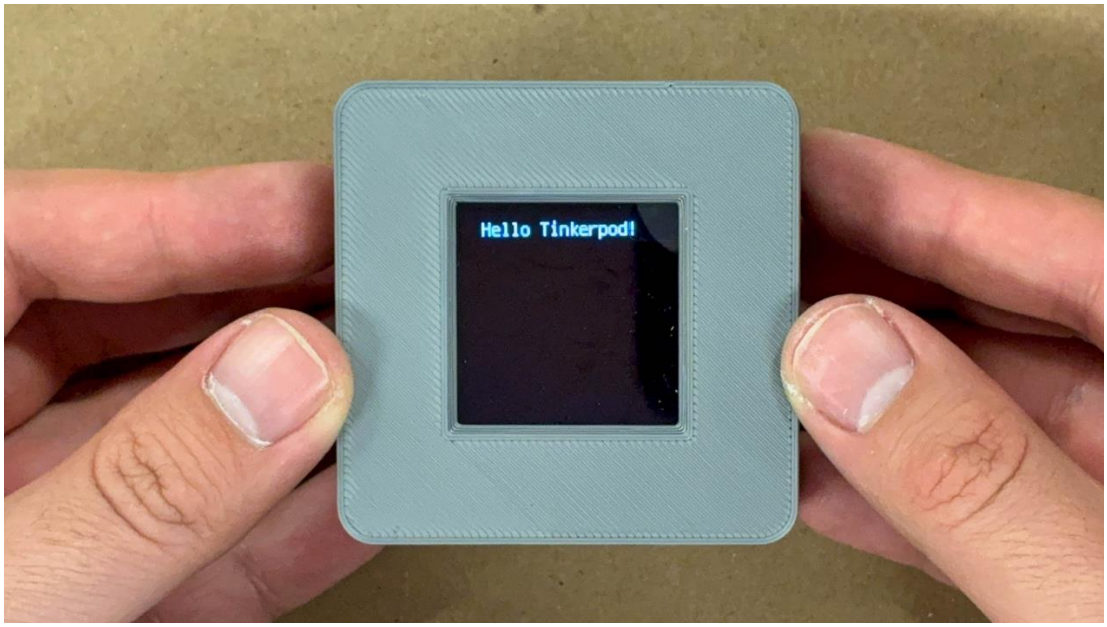
9. Closing the enclosure.

a. Plug the other side of the STEMMA connector to the QT PY.

b. Using the power switch in the BFF board, turn the device *on*.

c. Gently push the two parts of the enclosure together.

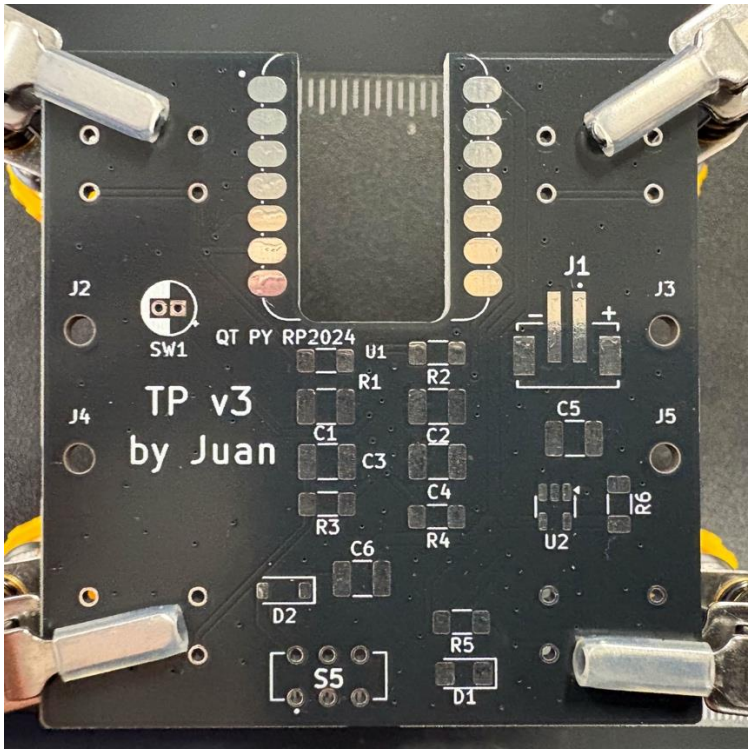
10. Enjoy!



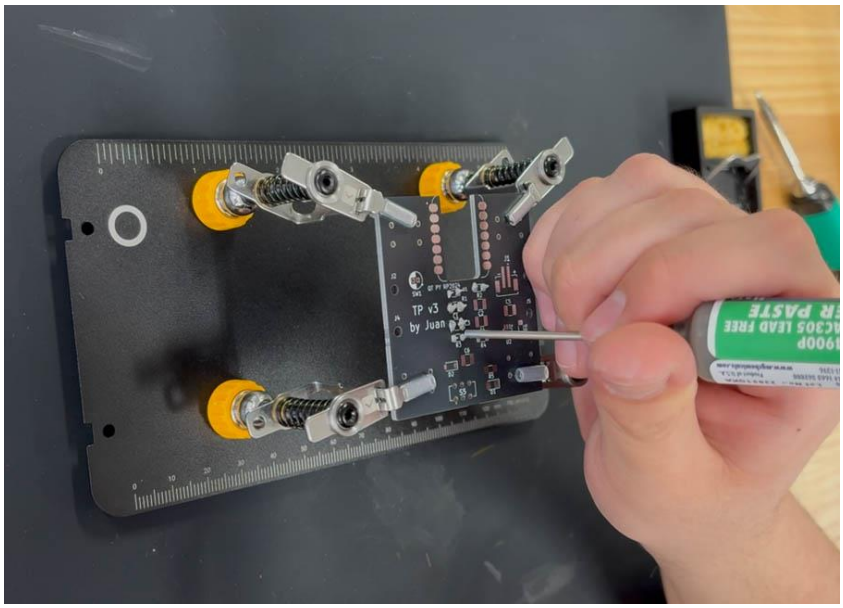
Appendix C: TinkerPod v3 PCB assembly guide

For the latest version of this guide visit: https://github.com/juansulca/tinker-pod/blob/main/v3/README.md

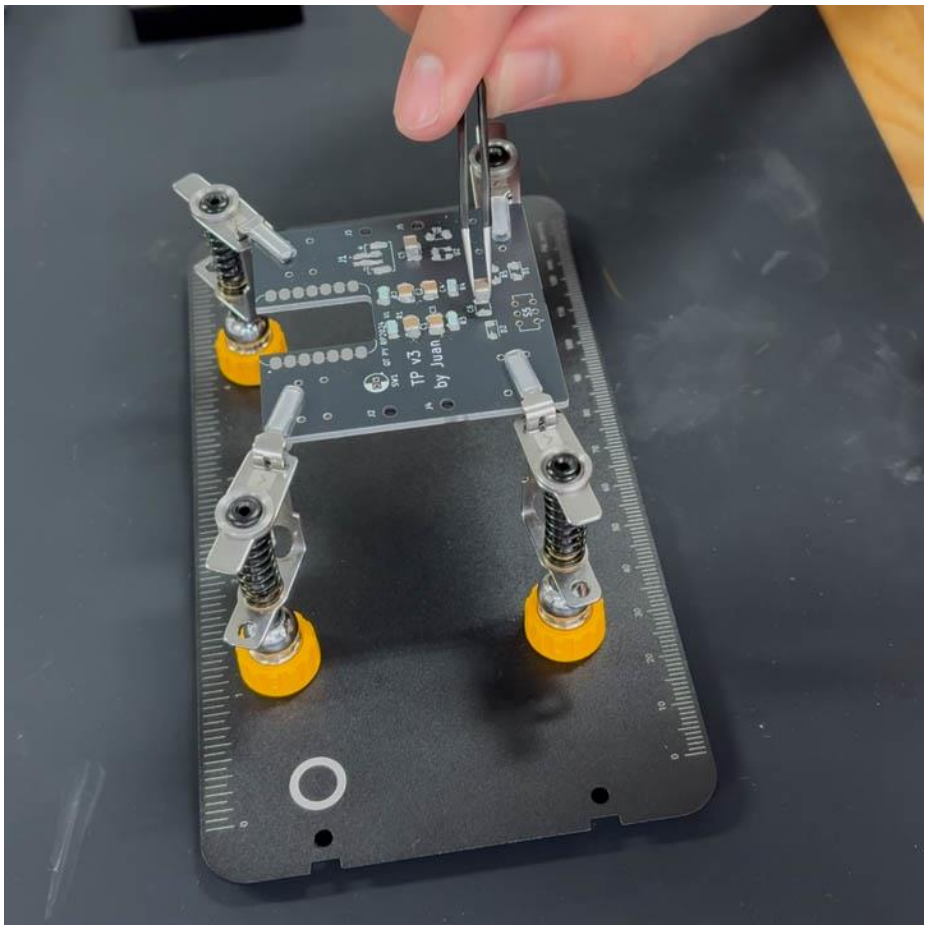
1. Secure the PCB with the TP v3 label facing up.



2. Apply paste in all the SMD pads, except for the QT PY pads, D1 and J1.
    1. Best results were achieved soldering the LED (D1) and JST (J1) connector with a soldering iron.
    2. Apply solder paste to the pads, but enough to cover the pad but not a blob bigger than the pad.

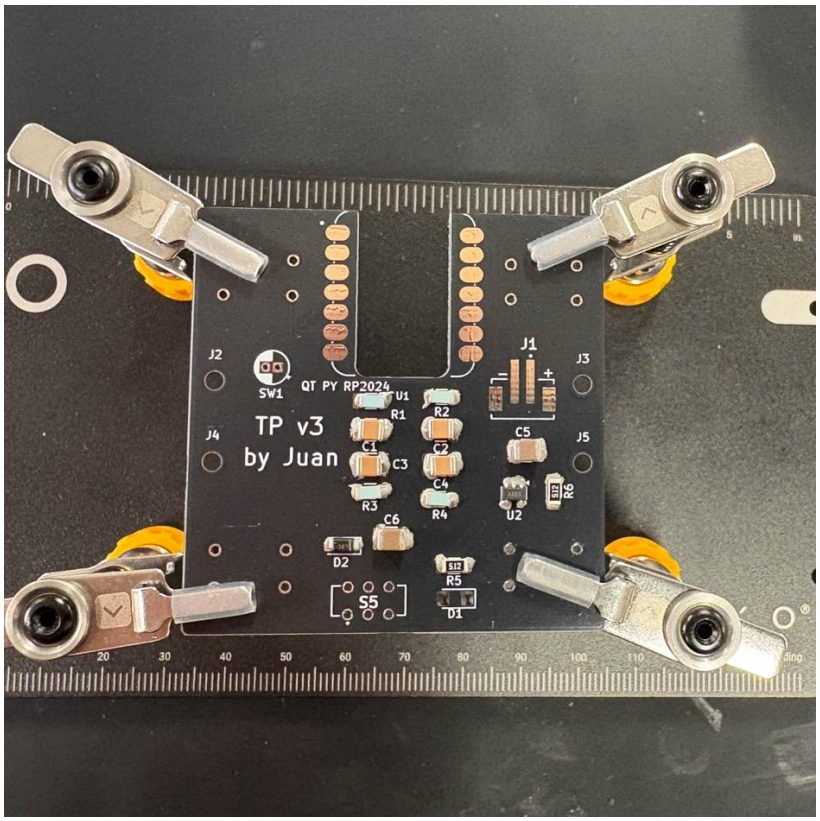


3. Use tweezers or a tiny plastic point to remove excess solder paster.
4. Place the components that do not have polarity.
    1. Place C1, C2, C3, C4 - the .1uF capacitors.
    2. Place R1, R2, R3, R4 - 1k resistors.
    3. Place C5, C6 - 10uF capacitors.

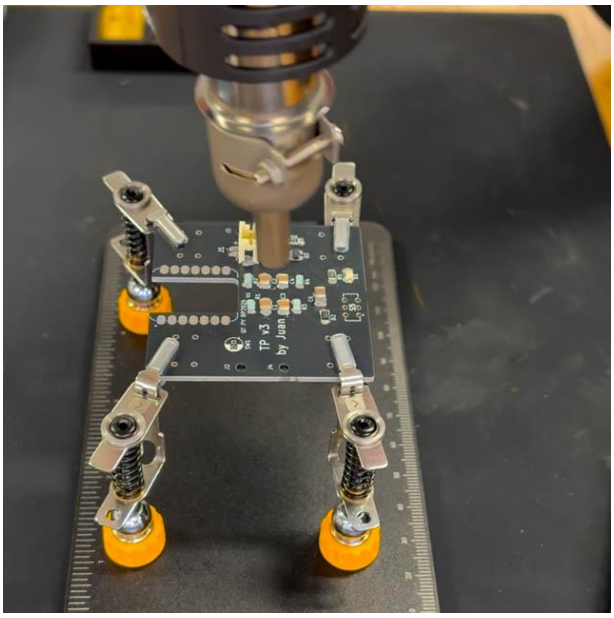


4. Place R5, R6 - 5.1K resistors.

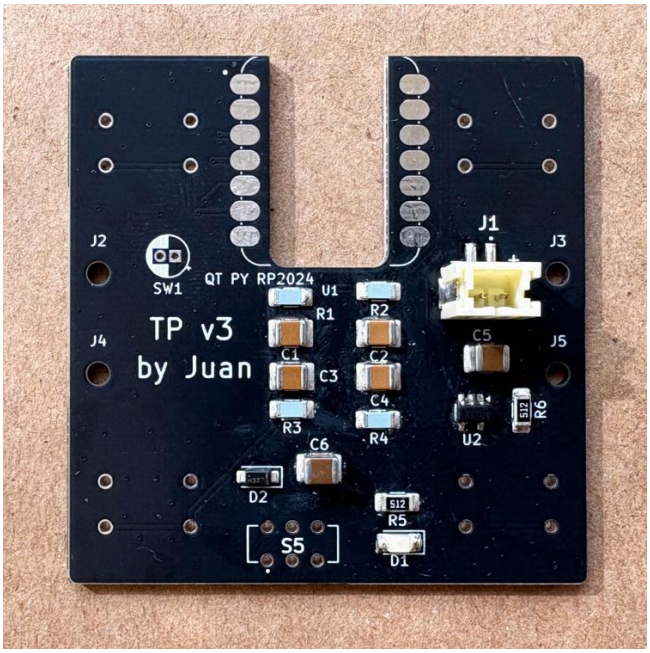
5. Place the components with polarity
    1. Place D2: diode with the line facing the closed side of the rectangle
    2. Place U2: the battery charging IC
        a. The side with 3 legs should match the side with the tiny arrow



6. Solder the components using a hot air reflow station.
    1. According to the specification of the solder paste, set the temperature, starts from the lowest recommended temperature.
    2. Start from far away with a medium to low airflow.
    3. Approach the individual components and make sure they fall correctly into place.



4. After all components are soldered, visually inspect the joints, make sure there are no pads shorting and that all the paste melted. Be careful, the PCB might be hot

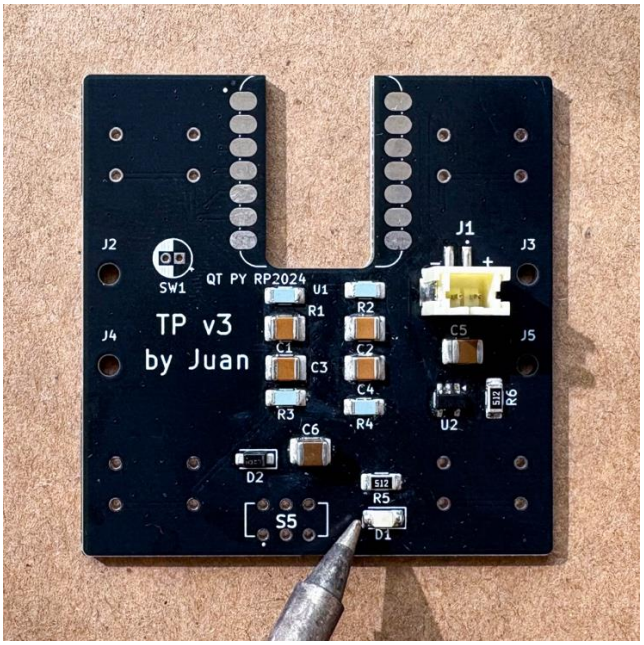


5. If there are is a blog of solder between two pads or components. Retouch them with a soldering iron and some flux.

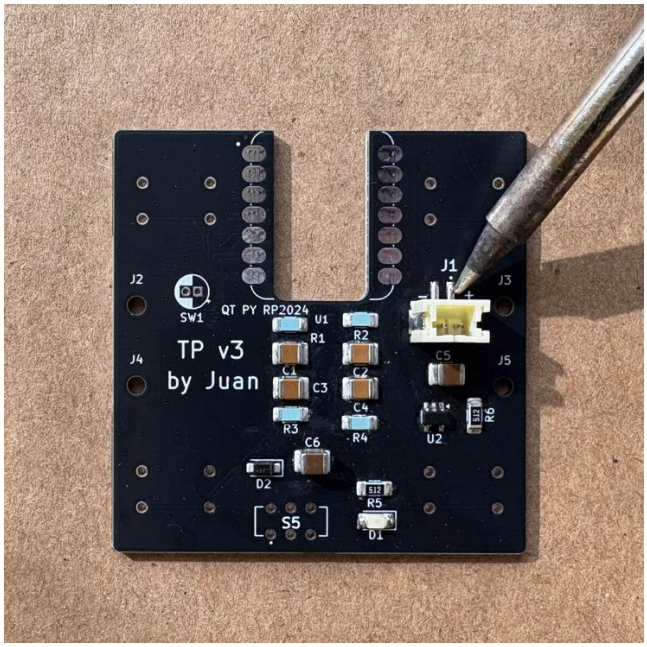
7. Solder D1 LED

    1. Check the polarity of the LED
    2. Place the LED with the Cathode (-) towards the closed side of the rectangle

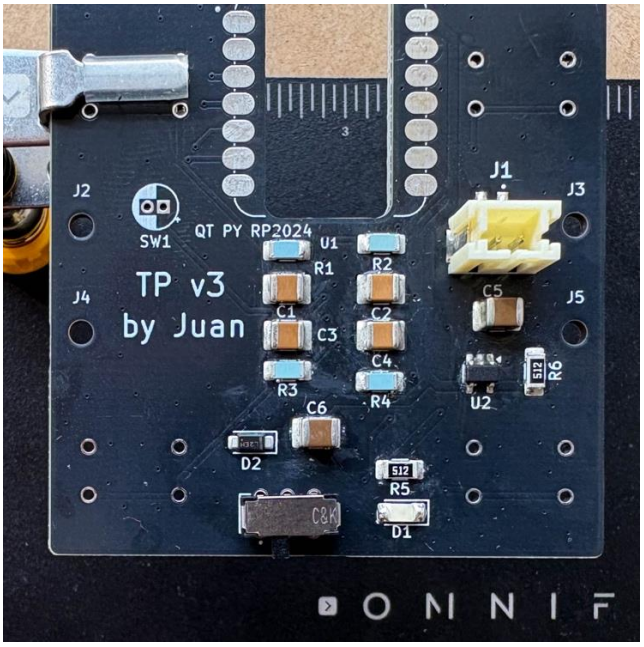
3. With the help of tweezers, solder the LED in place with a soldering iron.



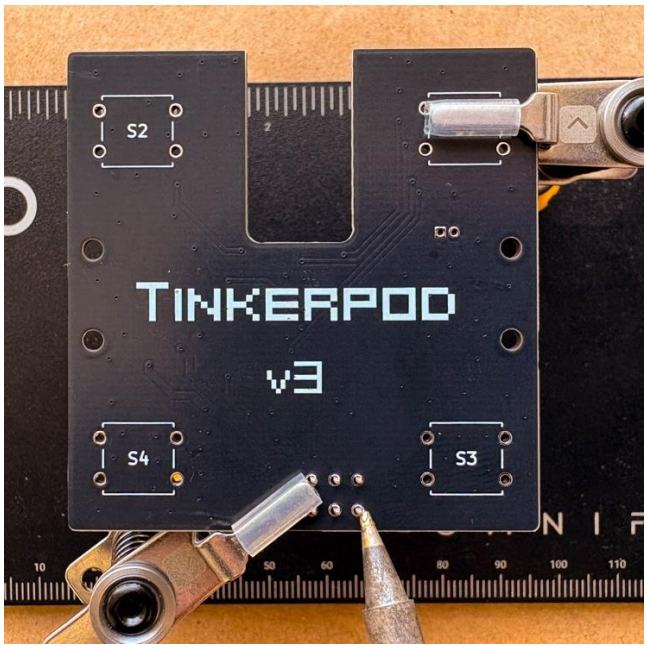
8. Solder J1 JST connector.
    1. Apply some flux on the pads.
    2. Place the JST connector in place, aligning the box with the body of the connector.
    3. With the soldering iron carefully solder one of the legs of the connector.
    4. Solder the remaining legs.



9. Solder the S5 slide switch.
    1. Place the switch from the side. (TP v3 side)



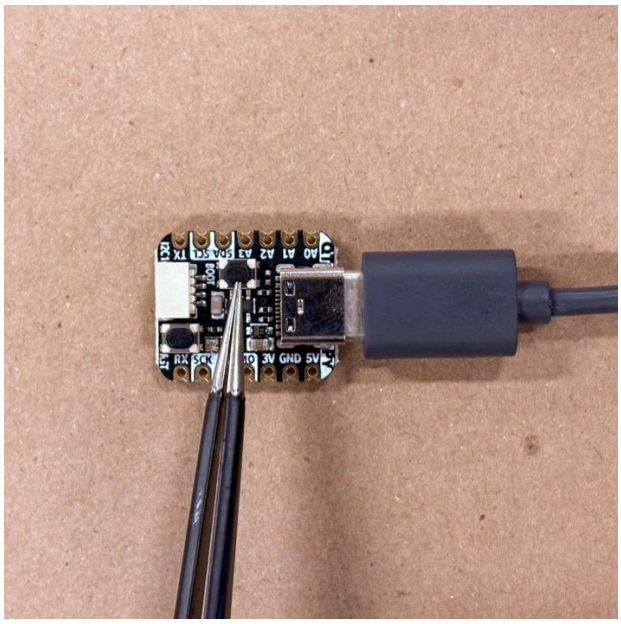
    2. Flip The PCB (TinkerPod pixelated logo up)
    3. Solder the six pins.



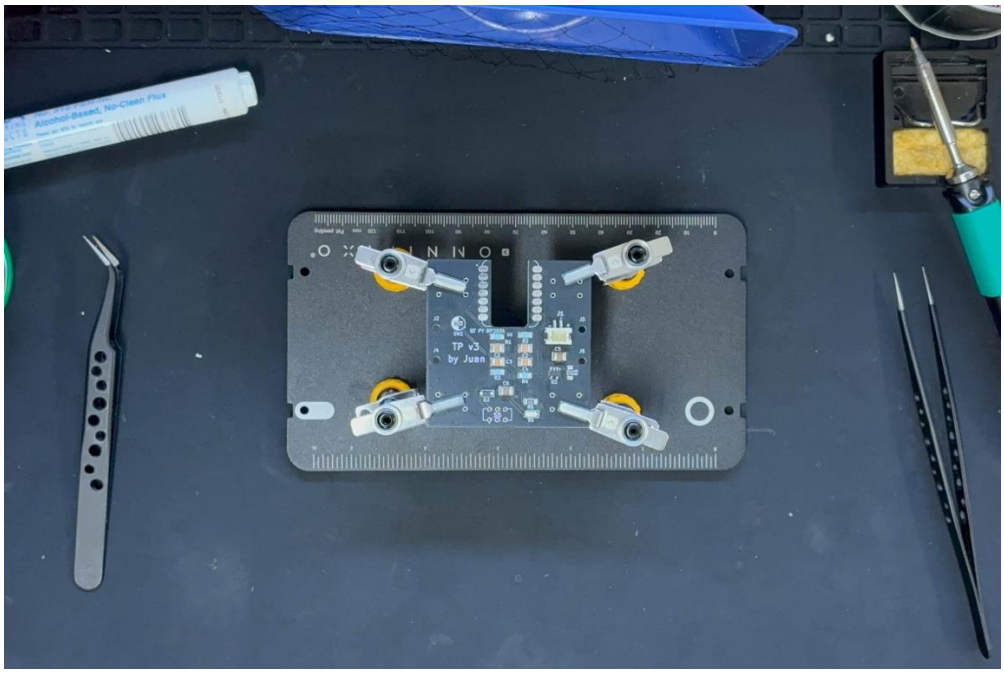
10. The PCB assembly will continue in the next section.

Appendix D: TinkerPod v3 assembly guide

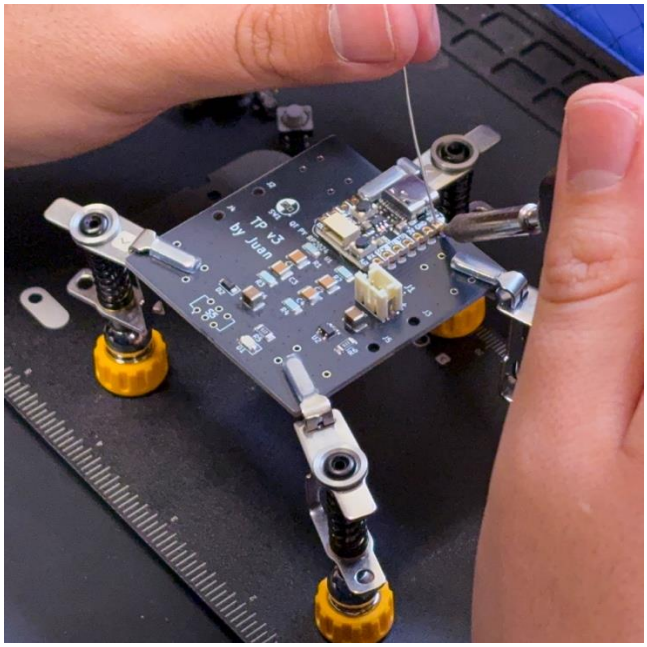
1. Flash circuit python on the QT PY RP2040.
    1. Open the official site and download the latest version of circuit python.
    2. While holding the *boot* button in the QT PY, plug it into a computer using the USB-C cable. Different version of the QT PY might need a different button to access bootloader mode, please check the official docs for the specific board.



    3. The device should appear mounted in the OS as **RPI-RP2**, this might be different for other boards other than the RP2040 board.
    4. Drag and drop the .UF2 file (downloaded in step 1) to the RPI-RP2 boot drive.
    5. After a couple of seconds, the onboard neopixel (LED) will flash and a new drive will appear in the computer, this time it should be called **CIRCUITPY**.
    6. ⚠ Always eject the device before unplugging the cable.
    7. The device is ready to be installed.
2. Solder the QT PY to the PCB.
    1. Place the PCB with the pixelated TinkerPod logo facing down. The **TP v3** should be visible.

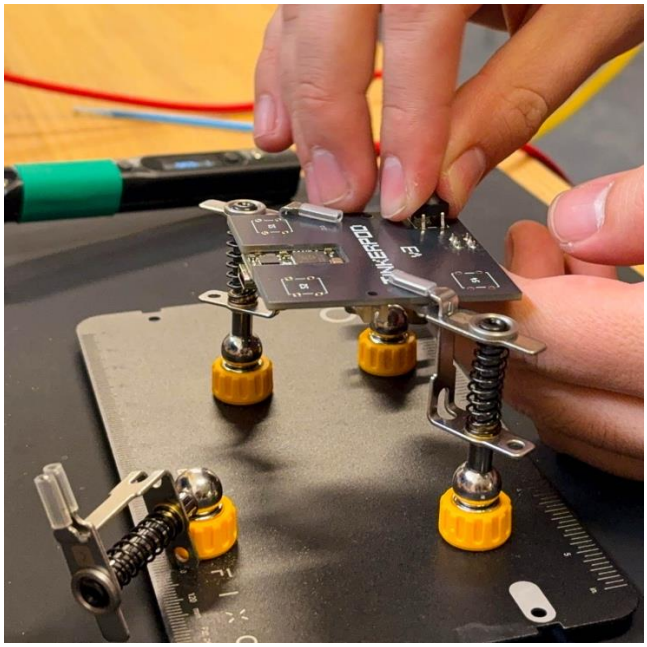


2. Carefully line up the QT PY with the PCB, in the QT PY RP2040 space. The pads can be identified for the cutout in the middle. Make sure the USB port is pointing towards the outside of the board and the reset and boot buttons, and the STEMMA connector are accessible.
3. Solder the **14** pins, 7 on each side.

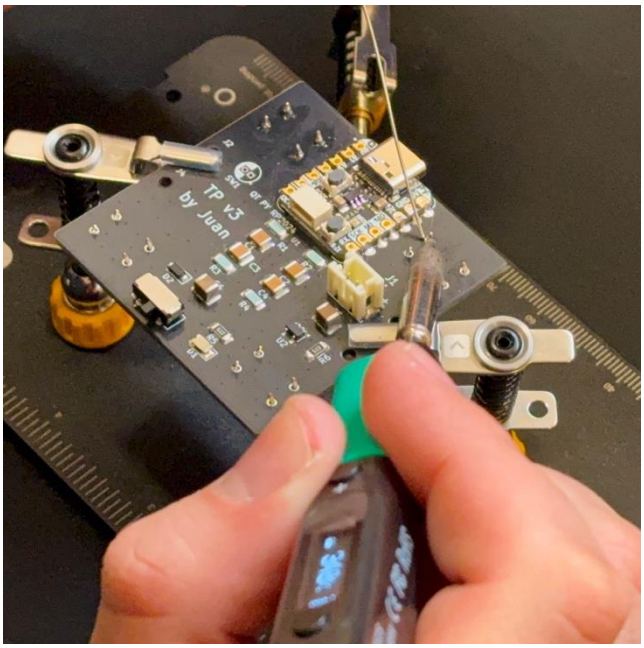


3. Solder the 4 buttons to the PCB.
    1. Flip the PCB (pixelated TinkerPod logo facing up).

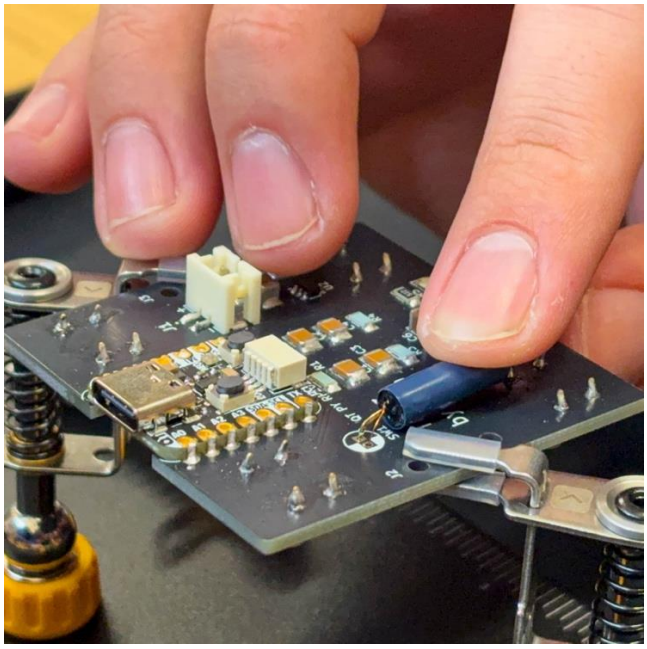
2. Place the buttons in the markers SW1, SW2, SW3, SW4.



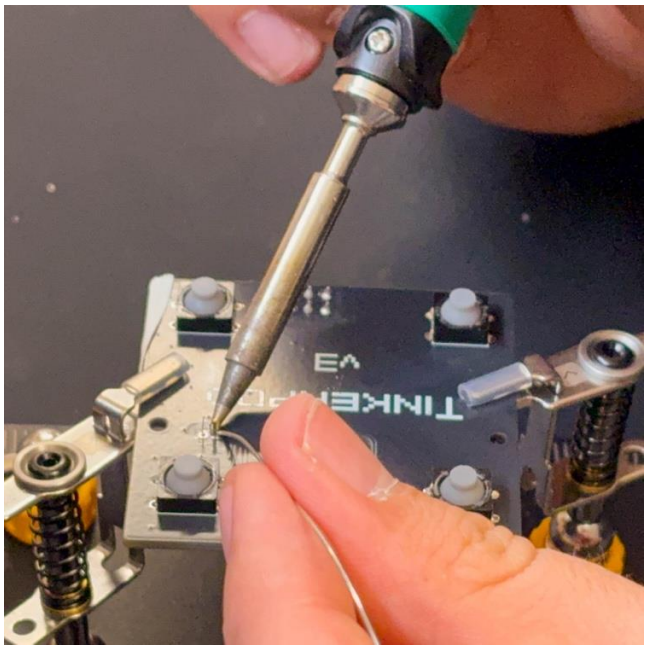
3. Flip the PCB again (pixelated TinkerPod logo facing down).
4. Solder all 4 pins of every push button. (16 solder joins in total).



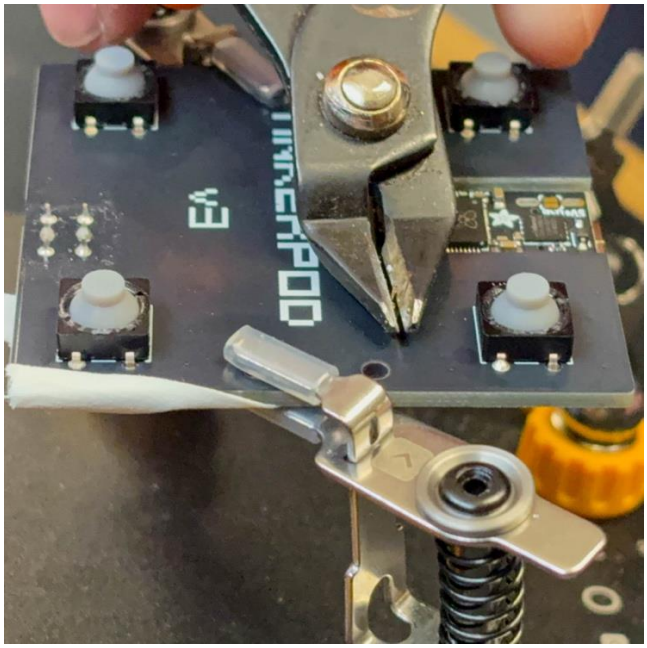
4. Solder the vibration sensor
    1. Bend the vibration sensor legs to 90 degrees.
    2. Put the vibration sensor into the SW1 slot. The vibration sensor does not have polarity.



3. If needed use some tape to secure the vibration sensor in place while soldering.
4. Flip the PCB (TinkerPod logo up)
5. Solder the two pins.

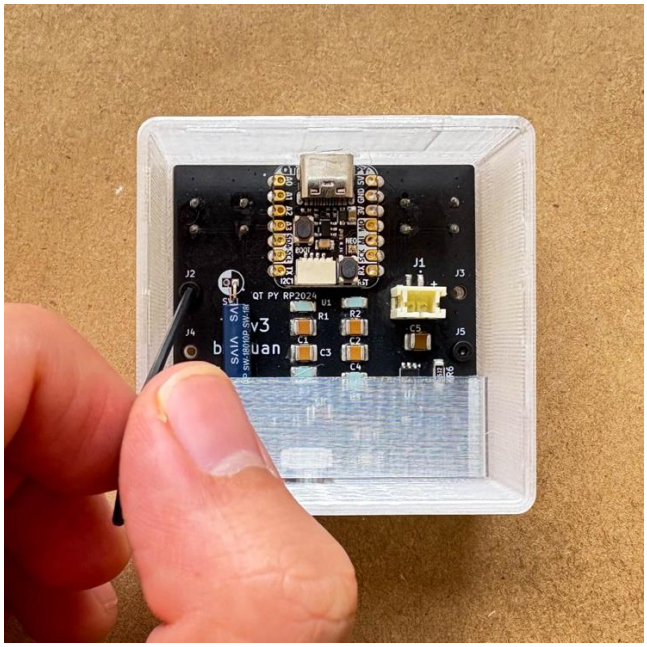


6. If the pins seem a bit long, cut them with a pair of flush cutters.



5. Now is a great moment to flash firmware and test the TinkerPod before installing it into the enclosure.
    1. Connect the STEMMA connector to the QT PY and Screen. It should slide in place without force.
    2. (Optional) Connect the battery using the JST connector in the BFF board.
    3. Download any of the TinkerPod examples from GitHub. (examples folder)
    4. Using a USB C data cable. Plug the device to the computer.
    5. Copy the content of the firmware folder (where the code.py file is) to the device.
    6. Remember copy the lib directory.
    7. Alternatively install the dependencies for the firmware using *circup*.
    8. A green LED should be *on* if there is battery connected to the JST terminal.
    9. If everything works as expected, continue with the next step. If something is not working, check for cold solder joins and check that that the firmware was flashed correctly. If there is red LED flashing, there is an issue with the firmware.
    10. Eject the device and unplug the USB cable.
6. Prepare the bottom enclosure.
    1. Slide the battery bracket on the PCB.

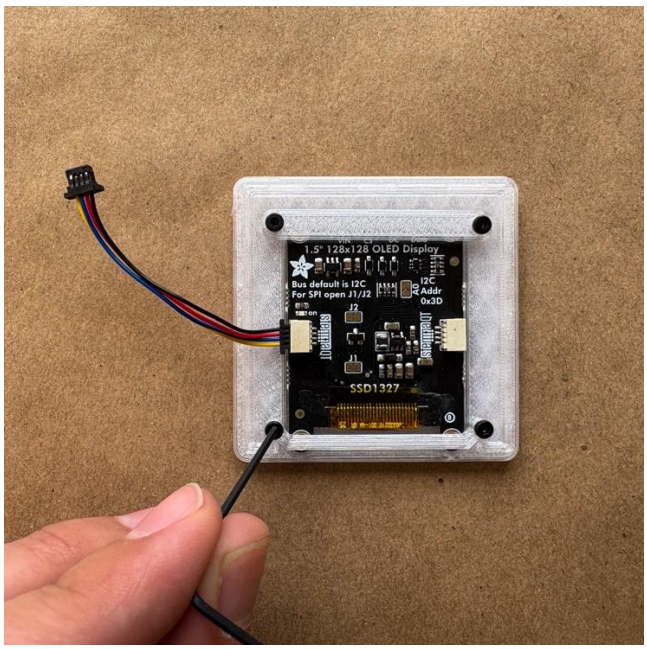
2. Place the PCB in place (with the USB-C port towards the hole in the enclosure) and screw it in place using the M2 machine screws.



3. Push the 3D printed on/off switch and make sure the power switch is in the *off* position (right side).

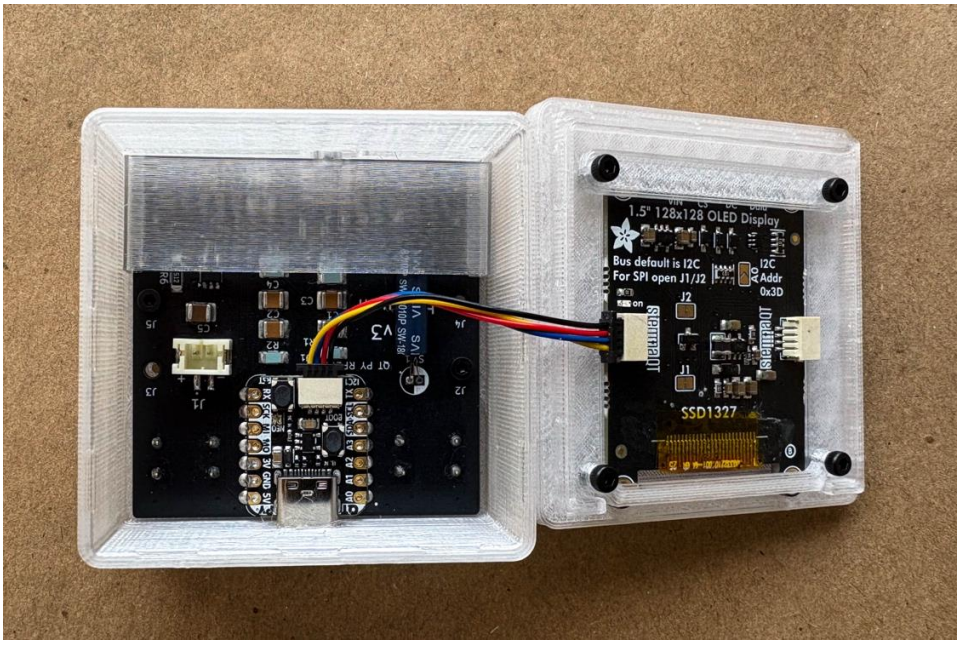
7. Prepare the screen assembly
    1. Connect the STEMMA cable into the screen, it should slide in place without force.
    2. Gently place the 1.5" screen into the slot, do not apply pressure on the screen.

3. Place the screen supports and line up the holes. The rectangular piece should go towards the top (can be placed in any orientation) and the remaining piece should go to the bottom. The bottom piece will only line up one way, with the legs pointing to the top.

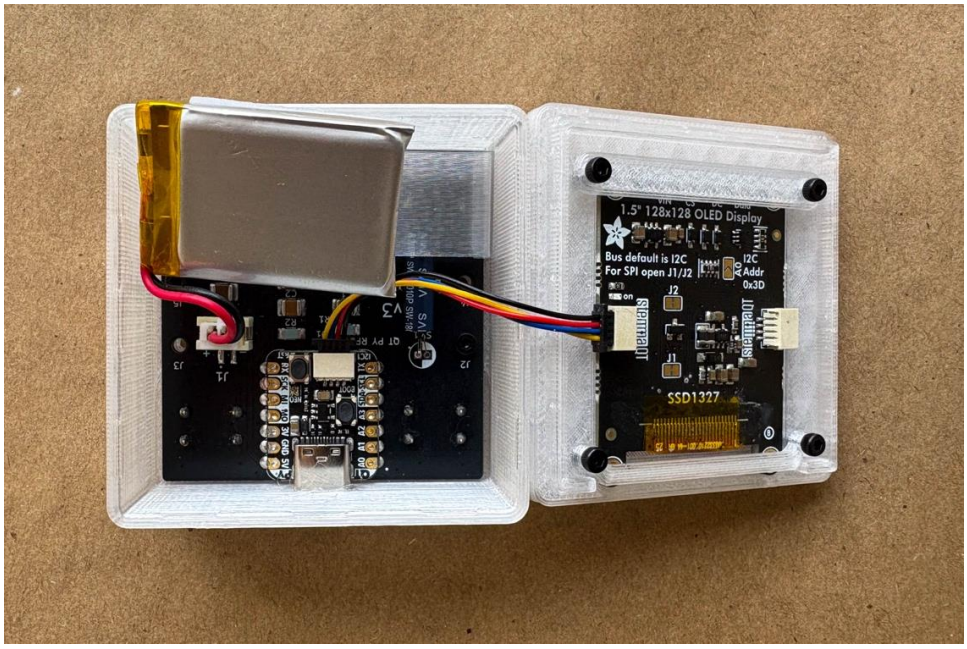


4. Screw the supports to the top enclosure, tighten the screws until there is some resistance. Do not over tighten the screws.
5. The screen might have some play, this is expected. And some gaps might be visible from the front side, which is also expected.

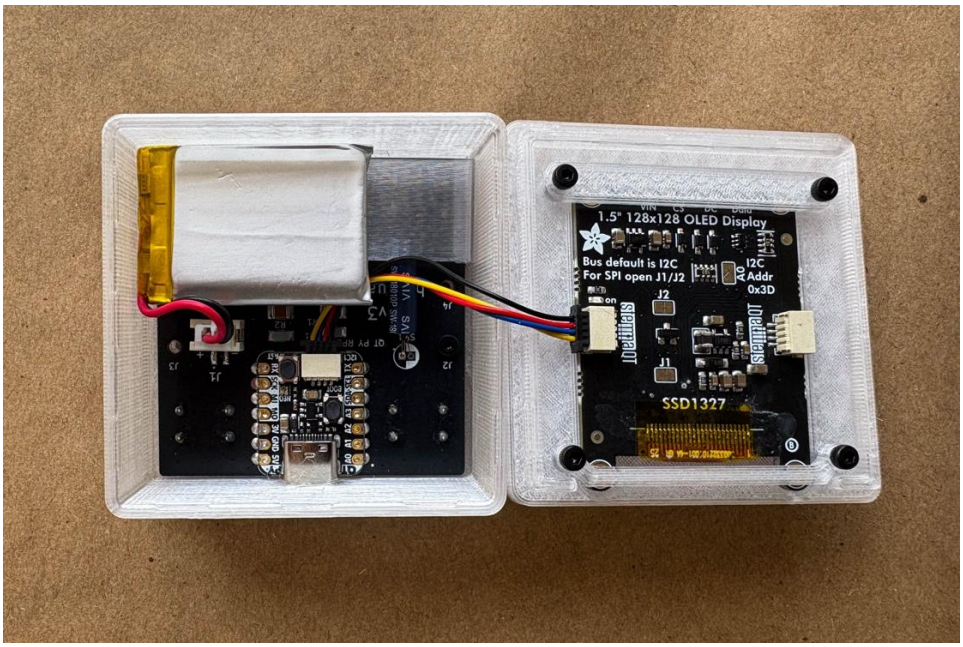
8. Closing the enclosure.
    1. Plug the other side of the STEMMA connector to the QT PY.



2. Plug in the battery.



3. Secure the battery using the battery bracket and some double-sided tape.



4. Close the enclosure

9. Turn on the device

10. Happy tinkering!