Badger 2040 Tastatur

Info

Ein vielseitiges, 3D-druckbares Makro-Keypad zur Erweiterung Ihres Badger 2040. Verwandeln Sie Ihr E-Ink-Badge mit dieser 12-Tasten-Erweiterung in ein leistungsstarkes Makro-Keyboard. Es verfügt über austauschbare mechanische Switches, eine nahtlose magnetische Pogo-Pin-Verbindung und eine einfach zu konfigurierende CircuitPython-Firmware für maximale Produktivität.

Inhaltsverzeichnis

Projektübersicht

Das Badger 2040-Tastatur ist eine kostengünstige, programmierbare USB-Makro-Tastatur mit Tastenbelegungsanzeige. Die Firmware basiert auf CircuitPython und lässt sich ganz einfach selbst erweitern.

Wenn Sie lediglich die Tastenbelegungen anpassen möchten, brauchen Sie nicht einmal Programmierkenntnisse, da es hierfür eine Konfigurationsdatei gibt, die Sie ganz einfach in einem Texteditor bearbeiten können. Da die Firmware eine Erweiterung meiner PicoSplit-Firmware ist, bietet sie die Möglichkeit, den Tasten mehrere Funktionen zuzuweisen. Du kannst Tastenbelegungen für mehrere Programme speichern, und jedes Programm kann zudem mehrere Tastenbelegungen haben. Du kannst ganz einfach zwischen den Programmen und deren Tastenbelegungen wechseln.

Du kannst Tastenanschläge je nach Dauer des Tastendrucks ausgeben. So kommst du mit nur wenigen Tasten aus. Um beispielsweise in der Liste der Tastaturlayouts vor- und zurückzuspringen, benötigst du nur eine Taste. Ein kurzer Tastendruck springt zum nächsten Layout. Ein längerer Tastendruck springt zum vorherigen Layout.

Nur für Badger
Wenn du es einfach nur auf deinem Badge ausprobieren möchtest, findest du die Firmware auf GitHub: Badger 2040 Keypad Firmware

Wenn du mehr Tasten benötigst
Wenn du mehr Tasten als die fünf auf dem Badge benötigst und einen 3D-Drucker hast, dann habe ich etwas für dich: eine Erweiterung mit 12 mechanischen Tasten. Du kannst sowohl MX-Tasten in Standardgröße (rechtes Bild) als auch die Low-Profile-Tasten von Kailh (linkes Bild) verwenden. Das Ganze wird von Hand verdrahtet, ist aber wirklich einfach herzustellen.

Das Badge wird magnetisch an der Tastatur gehalten und lässt sich leicht anbringen oder abnehmen, da für die elektrische Verbindung Pogo-Pins verwendet werden. Das geht sogar, während das Badge an den Computer angeschlossen ist.

Das Badge wechselt dann automatisch in den entsprechenden Modus. Im angeschlossenen Zustand zeigt es die Tastaturbelegung der zusätzlichen Tasten an, und wenn es ohne die externe Tastatur betrieben wird, zeigt es die Belegung seiner integrierten Tasten an.

Insgesamt benötigte Teile & Materialien

Menge	Teile-Name	Links & Schritte
2 x	**10 mm x 2 mm Neodym-Magnete**	Amazon 1

Menge	Teile-Name	Links & Schritte
2 x	**Badger2040Keypad_Bottom.stl**	Printables 1 26
2 x	**Badger2040Keypad_Insert.stl**	Printables 2
1 x	**Badger2040Keypad_Connector.stl**	Printables 3
4 x	**Pogo-Pins**	Amazon 4
4 x	**Anschlusskabel mit einem Querschnitt von 0,2 mm²**	4
1 x	**Dünnflüssiger Sekundenkleber**	9
1 x	**Schrumpfschlauch**	10

Menge	Teile-Name	Links & Schritte
1 x	**MCP23017 / 16-Bit-E/A-Erweiterung**	 eBay 14
12 x	**Mechanische Tasten (MX)**	22
12 x	**MX-Steckbuchsen**	23
12 x	**MX-Tastenkappen**	22
6 x	**2 mm x 10 mm Spitzschrauben**	 Amazon 26

Anleitung

Einige Anmerkungen zu den Bauteilen

Pogo-Pins

Ich weiß, dass sie etwas teuer sind. Man kann die Tastatur auch ohne die Pogo-Pins bauen und stattdessen einfach ein Kabel mit einem Qwiic-Stecker. In diesem Fall wird das Kabel durch die Anschlussöffnung geführt und in die Buchse auf der Rückseite des Badges gesteckt. Wenn du es auf diese Weise machen möchtest, musst du „Connector.stl“ nicht drucken. Ich empfehle jedoch die Variante mit den Pogo-Pins, da die Qwiic-Buchse am Badge nicht dafür gedacht ist, ständig ein- und ausgesteckt zu werden. Die Buchse kann leicht beschädigt werden, und es ist nicht so einfach, das Badge von der Tastatur zu trennen.

Tasten

Du benötigst zwölf mechanische Tasten. Sowohl Tasten im MX-Stil als auch Low-Profile-Tasten von Kailh passen in die Tastatur. Außerdem benötigst du zwölf passende Tastenkappen in einer Farbe deiner Wahl. Man braucht sie nicht unbedingt, aber ich empfehle Tastenfassungen, damit man die Tasten später austauschen kann. Außerdem erleichtern sie das Anlöten der Kabel an die Tasten, da sie eine größere Lötfläche haben. Ich bestelle meine Tasten, Tastenkappen und Fassungen hier:

- *keycapsss.com*
- *keygem.store*

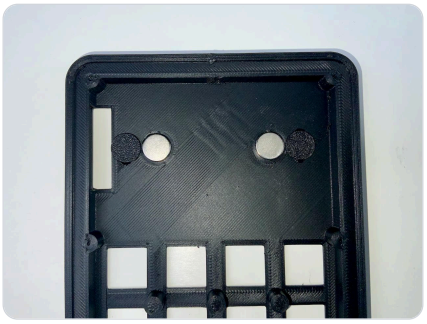
Schrumpfschlauch
Kurze Stücke Schrumpfschlauch werden über die Pogo-Pin-Anschlüsse gezogen. Dies dient hauptsächlich der optischen Aufwertung, da die Pogo-Pins von der Seite sichtbar sind. Verwende eine Farbe, die zu deinem Gehäuse passt.

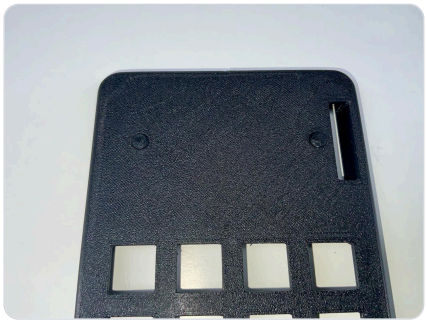
1 Magnete

Kleben Sie zunächst zwei der Magnete mit Sekundenkleber an die dafür vorgesehenen Stellen. Halten Sie das Abzeichen von hinten daran, damit die Magnete richtig ausgerichtet sind. Die Magnete im Abzeichen müssen die Magnete im Tastenfeld anziehen.

Benötigte Teile:
- 2 x 10 mm x 2 mm Neodym-Magnete
- 1 x Badger2040Keypad_Bottom.stl

2 Beilagen





Die beiden Einsätze halten das Abzeichen an seinem Platz. Kleben Sie sie mit Sekundenkleber fest.

Benötigte Teile:
- 2 x Badger2040Keypad_Insert.stl

3 Vorbereitung der Verbindungsstücke

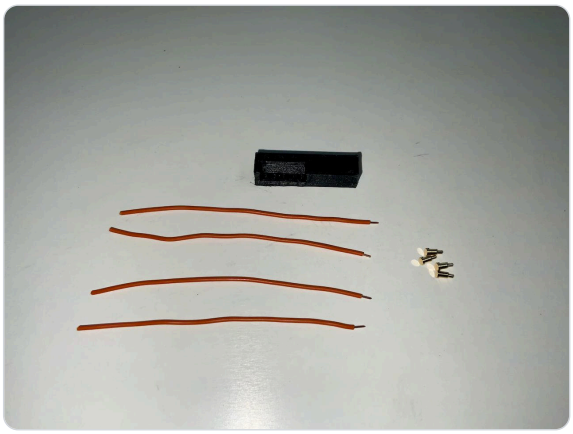
Bevor Sie mit der Arbeit an den Pogo-Pins beginnen, vergewissern Sie sich, dass sich der Stecker mit etwas Kraftaufwand in die dafür vorgesehene Öffnung einstecken lässt. Er sollte nicht zu locker sitzen, damit Sie seine Position später noch anpassen können. Wenn das Teil nicht fest in die Öffnung passt, versuchen Sie, es mit anderen Einstellungen erneut zu drucken.
Der Stecker sollte leicht in die Rückseite des Badges gleiten.

Benötigte Teile:
- 1 x Badger2040Keypad_Connector.stl

4 Pogo-Pins – Ersatzteile

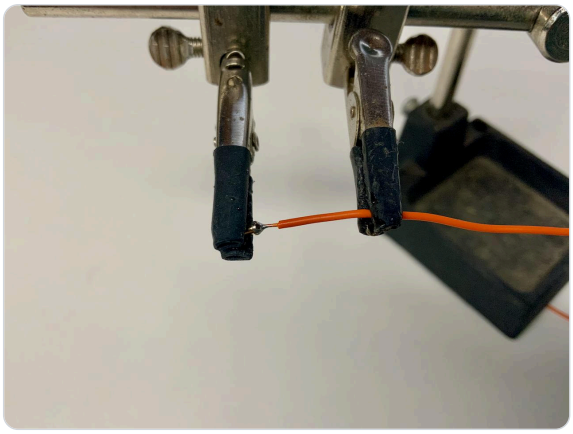
Das ist das erste Projekt, bei dem ich Pogo-Pins verwendet habe, und ich dachte eigentlich, dass die Handhabung schwierig sein würde, aber letztendlich ist es ganz einfach.

Benötigte Teile:
- 4 x Pogo-Pins
- 4 x Anschlusskabel mit einem Querschnitt von 0,2 mm²



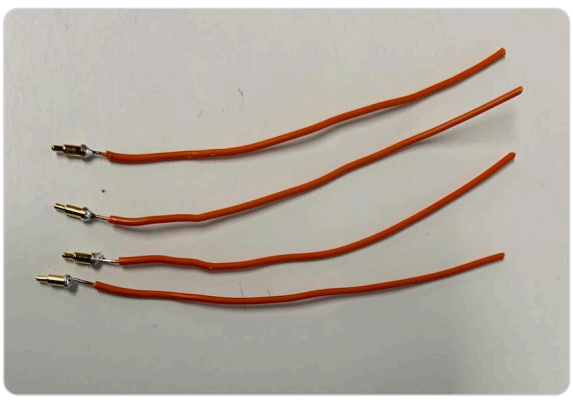
5 Pogo-Pins – Löten

Mit Hilfe einer dritten Hand können Sie die Kabel an der Rückseite der Stifte anlöten.



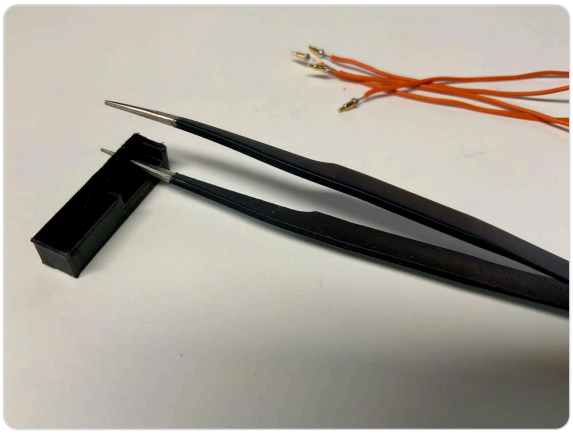
6 Pogo-Pins – Drähte und Stifte

Das Ergebnis.



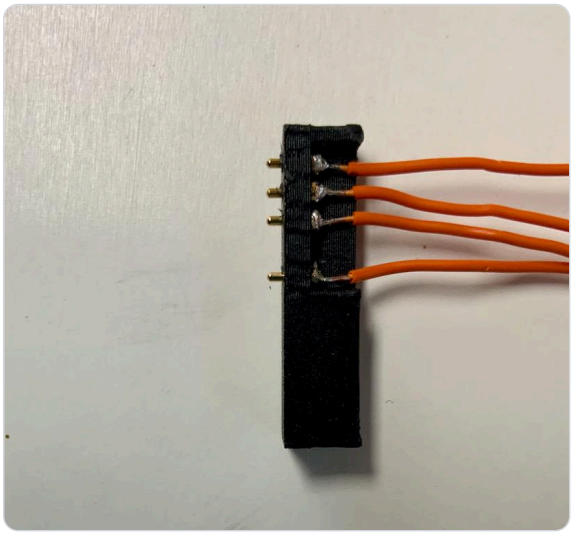
7 Pogo-Pins – Vorbereitung des Steckverbinders

Die Löcher im Stecker sind oft zu klein für die Pogo-Pins. Du kannst sie mit einer spitzen Pinzette etwas erweitern. Aber nicht zu sehr. Du solltest sie mit etwas Druck einstecken können.



8 Pogo-Pins – Einsetzen in den Steckverbinder

Achten Sie darauf, dass alle Stifte etwa gleich weit herausragen.



9 Pogo-Pins – Verkleben

Befestigen Sie die Stifte auf der Rückseite mit Sekundenkleber, aber verwenden Sie nicht zu viel Kleber, damit der Federmechanismus nicht beschädigt wird.

Benötigte Teile:

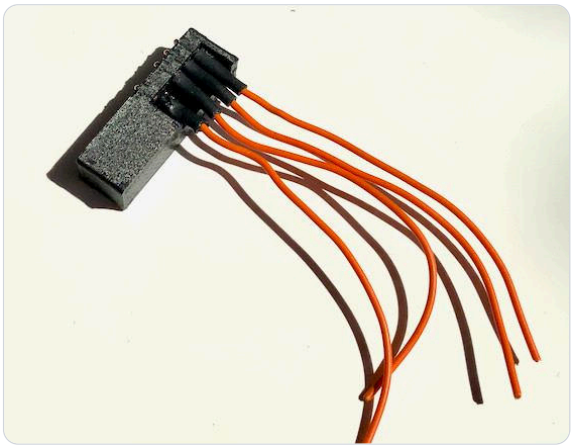
- 1 x Dünnflüssiger Sekundenkleber

10 Pogo-Pins – Isolierung

Isolieren Sie die Kabelverbindungen mit dünnem Schrumpfschlauch.

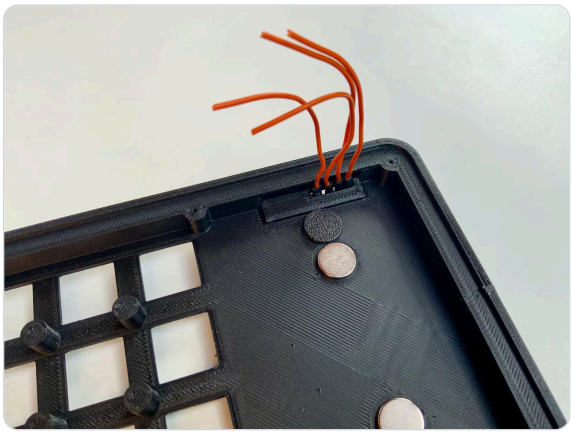
Benötigte Teile:

- 1 x Schrumpfschlauch

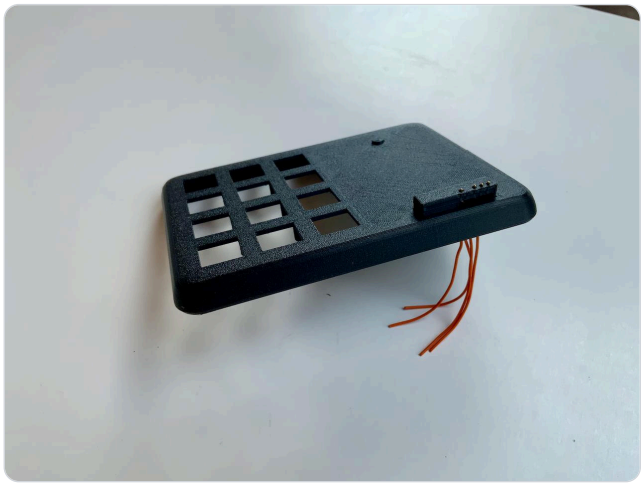


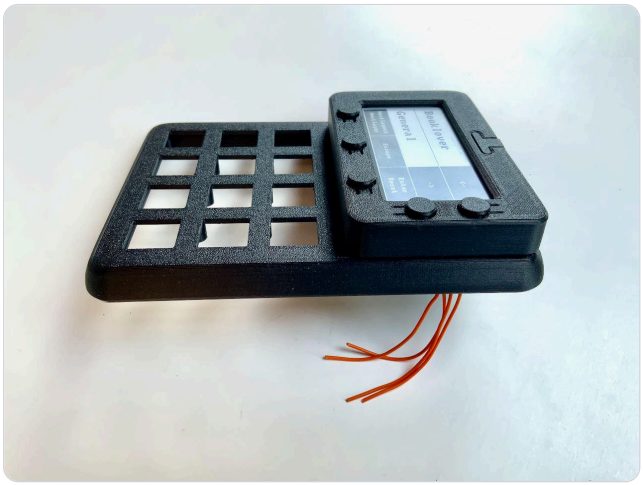
11 Stecker

Stecken Sie den Verbinder von oben in das Oberteil ein und setzen Sie dann das Abzeichen auf das Oberteil.



12 Stecker probeweise einstecken

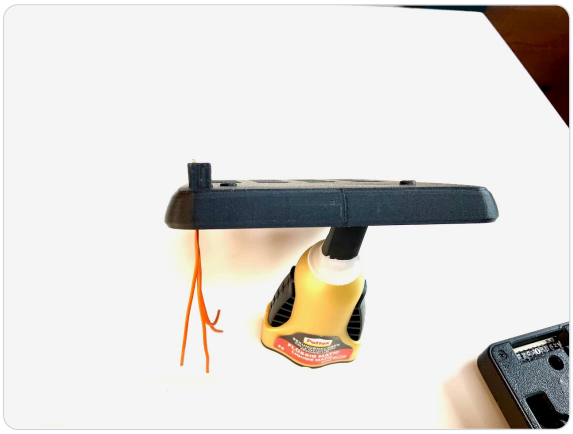




Drücken Sie den Stecker von hinten gegen die Leiterplatte des Ausweises, sodass die Pogo-Pins angedrückt werden.

13 Kleben Sie den Stecker fest

Entfernen Sie das Abzeichen und achten Sie darauf, dass der Stecker oben gerade sitzt. Wenn Sie mit der Position zufrieden sind, kleben Sie ihn mit etwas Sekundenkleber fest.



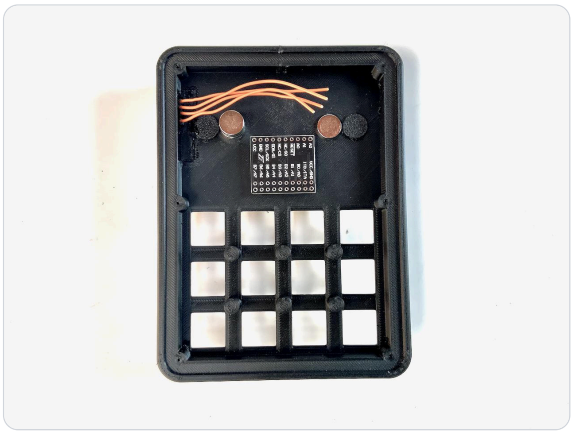
14 IO-Expander

Der IO-Expander ist eine kleine Leiterplatte, auf der ein MCP23017 montiert ist. Da es so viele verschiedene Module mit diesem IC gibt, habe ich keine Halterung dafür vorgesehen. Ich habe mein Modul einfach mit Heißkleber befestigt und dabei darauf geachtet, dass die Beschriftung sichtbar bleibt.

Benötigte Teile:

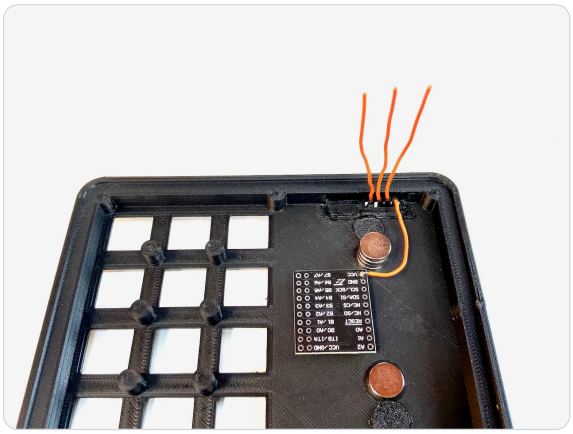
- 1 x MCP23017 / 16-Bit-E/A-Erweiterung

15 IO-Expander vorbereiten

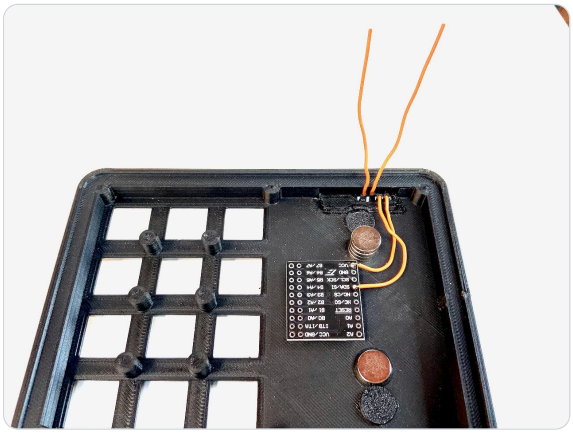


16 Kabelverbinder für IO-Erweiterung

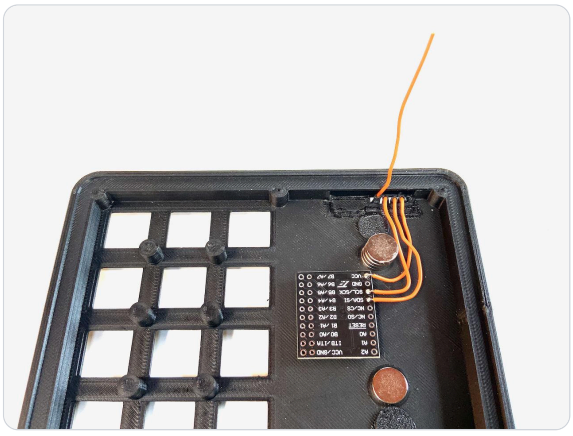
Die folgenden Abbildungen zeigen Ihnen, wie Sie die Kabel vom Stecker an das MCP23017-Modul anschließen.



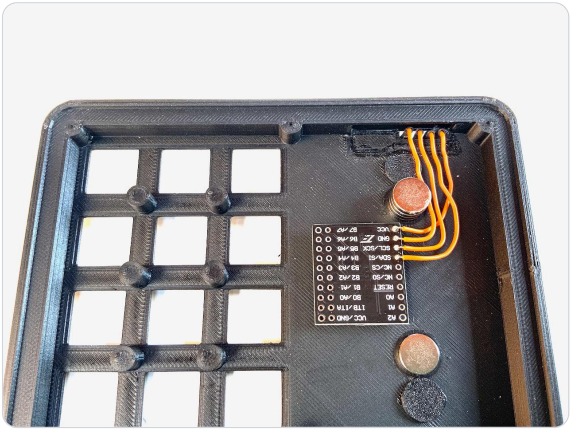
17 Lötverbindungen (Teil 1)



18 Lötverbindungen (Teil 2)

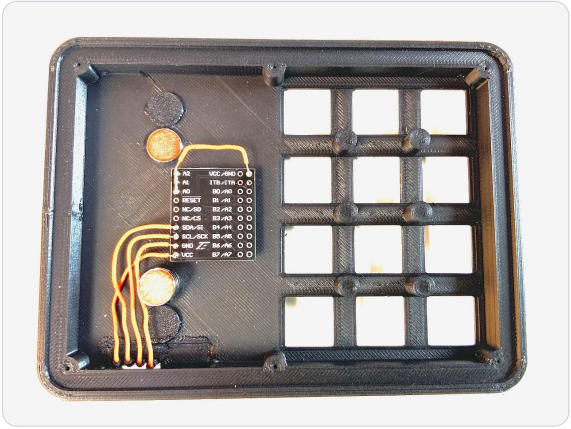


19 Lötverbindungen (Teil 3)



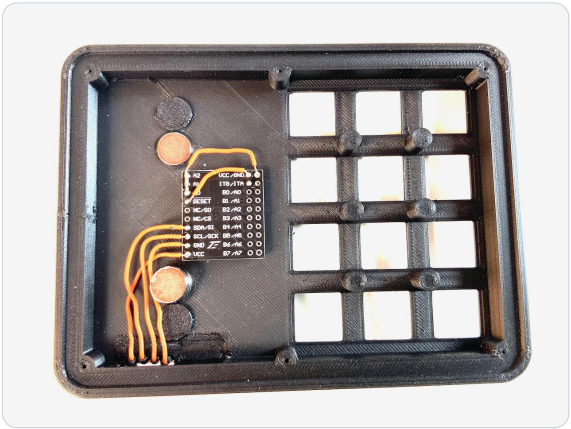
20 Verbinden Sie die Adresspins mit GND

Alle Adresspins sind mit GND verbunden



21 „Reset“ an VCC anschließen

Reset ist mit VCC verbunden

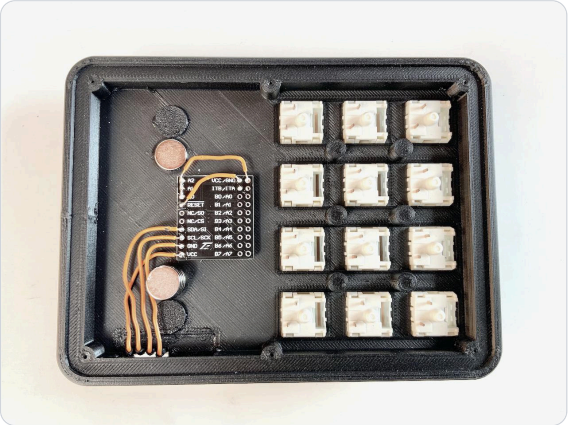


22 Tastenschalter

Setzen Sie zwölf Tastenschalter ein. Sie können Tastenschalter im MX-Stil oder Low-Profile-Tastenschalter von Kailh verwenden (ich habe nur die ursprünglichen Low-Profile-Tastenschalter getestet, nicht Version 2).

Benötigte Teile:

- 12 x Mechanische Tasten (MX)
- 12 x MX-Tastenkappen

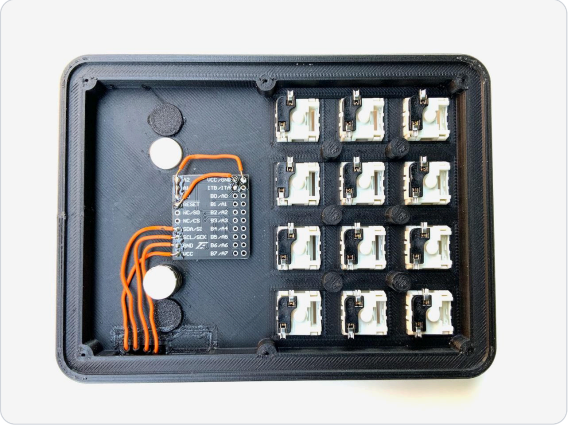


23 Schalterdosen einbauen

Bringen Sie Buchsen an den Anschlüssen der Schalter an, falls Sie diese später einmal austauschen möchten. Aber auch wenn Sie sie nicht austauschen möchten, empfehle ich die Buchsen, da sich die Kabel leichter daran anlöten lassen.

Benötigte Teile:

- 12 x MX-Steckbuchsen



24 Schalter an GND anschließen

Verbinden Sie jeweils einen Pin jedes Schalters mit GND

25 Schalter an den IO-Expander anschließen

Verbinden Sie den zweiten Pin jedes Schalters mit einem Eingang des MCP23017-Moduls. Sie können alle Eingänge außer den letzten vier (B4, B5, B6 und B7) verwenden. Es spielt keine Rolle, welche Taste Sie an welchen Pin anlöten.

26 Bodenplatte

Nun sind Sie mit der Montage der Hardware fast fertig. Sie müssen nur noch die Bodenplatte mit sechs 2-mm-Schrauben am Oberteil befestigen.

Benötigte Teile:

- 1 x Badger2040Keypad_Bottom.stl
- 6 x 2 mm x 10 mm Spitzschrauben

27 Grundplatte montieren

Firmware

Die Firmware für das Badger 2040-Tastenfeld ist eine Weiterentwicklung meiner PicoSplit-Firmware. Diese Firmware verfügt zwar über weniger Funktionen als manche andere Firmware, bietet aber alles, was für die Konfiguration erforderlich ist. Sie müssen keine zusätzliche Software auf Ihrem Computer installieren. Ein einfacher Texteditor und ein Webbrowser reichen völlig aus. Ich stelle die Firmware kostenlos unter der MIT-Lizenz zur Verfügung. Sie wird auf GitHub gehostet.

Funktionen

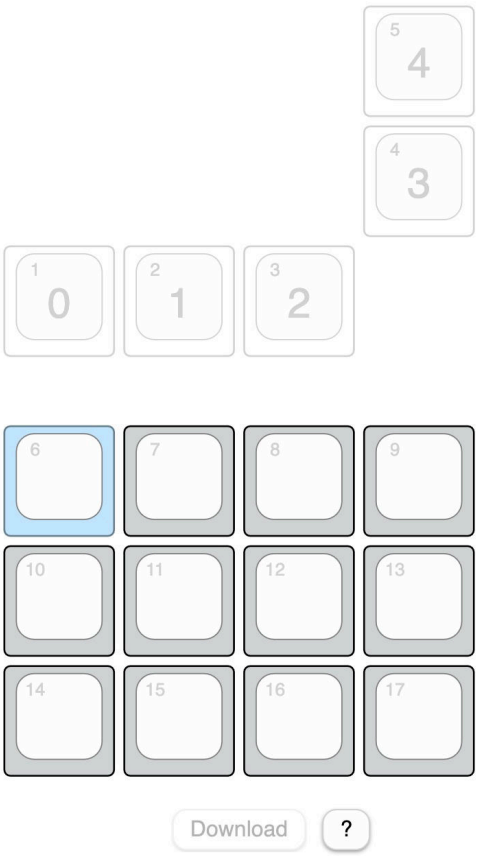
- *Zeitsparendes System für die Verdrahtung der Taster.*
- *Es muss keine weitere Software auf Ihrem Computer installiert werden.*
- *Lesbares Format der Konfigurationsdatei.*
- *Mit etwas Python-Kenntnissen leicht anpassbar.*
- *Erweiterte Tastaturlayouts sind möglich.*
- *Unterstützt mehrere Layouts, wobei jedes Layout mehrere Ebenen haben kann.*

Installation der Firmware

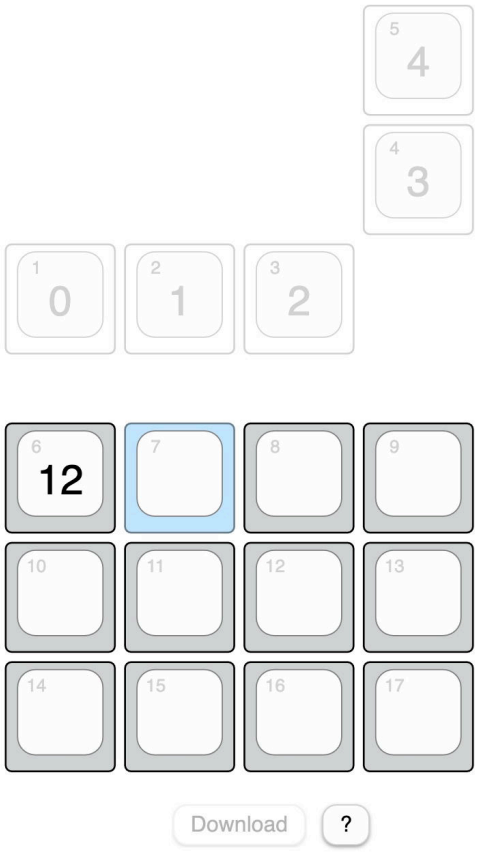
Die Firmware für das Badger 2040 Keypad basiert auf CircuitPython Version 7.3. Ältere Versionen funktionieren nicht. Gehen Sie zu https://circuitpython.org/board/pimoroni_badger2040/ und installieren Sie CircuitPython auf Ihrem Badge. Fahren Sie anschließend mit den nächsten Schritten fort.

1. *Schließen Sie das Badge mit einem USB-Kabel an Ihren Computer an.*
2. *Im Finder (macOS) oder im Explorer (Windows) erscheint ein USB-Speicherlaufwerk mit dem Namen „CIRCUITPY“.*
3. *Kopieren Sie die Dateien und Ordner aus dem Ordner „CIRCUITPY“ im Repository auf diesen Speicher. Wenn Sie kein externes Tastaturmodul verwenden möchten, überspringen Sie die Abschnitte „Einrichtung des Tastaturmoduls“ und „Zuordnung der Hardwaretasten“.*
4. *Öffnen Sie einen Texteditor.*
5. *Drücken Sie nacheinander alle Tasten auf Ihrem Tastenfeld. Jedes Mal, wenn Sie eine Taste drücken, sollte im Texteditor eine Zahl erscheinen und der Cursor anschließend in die nächste Zeile springen. Wenn beim Drücken einer Taste keine Zahl erscheint, überprüfen Sie die Verkabelung der entsprechenden Taste.*

Wenn wir von einem Tastaturlayout sprechen, meinen wir oft zwei Dinge: die Anordnung der Tasten auf einer Tastatur und die Tastenbelegung. In der Firmware des Badger 2040-Tastaturmoduls wird der Begriff „Layout“ für die Tastenbelegung verwendet. Diese Firmware ist darauf ausgelegt, die Verdrahtung so einfach wie möglich zu gestalten. Beispielsweise müssen die Tasten nicht an bestimmte Eingangs-Pins des MCP23017 angeschlossen werden, um einer bestimmten Tastenanordnung zu entsprechen (Sie können alle Eingänge außer den letzten vier verwenden: B4, B5, B6 und B7). Dies erleichtert den Bau handverdrahteter Tastaturen, da Sie jede Taste an jeden beliebigen Pin anschließen können, solange Sie die in der Firmware festgelegten Pins verwenden. Die Zahlen, die Sie beim Schnellfunktionstest sehen, sind Hardware-Tastennummern. Diese Hardware-Tastennummern werden standardisierten Tastennummern zugeordnet, die in der Datei zur Definition des Tastaturlayouts (layout.js) verwendet werden. Auf diese Weise können Sie dasselbe Layout mit unterschiedlich verdrahteten Tastaturen verwenden. Die Zuordnung von Hardware-Tastennummern zu Layout-Tastennummern kann nicht automatisch erfolgen, da das System nicht weiß, wie die Tasten verdrahtet sind. Es gibt jedoch ein Tool für diese Aufgabe. Öffne die Datei „PinMapper.html“ in einem Webbrowser, und du siehst die folgende Seite:



Dies sind alle Tasten des Badger 2040, wenn er an das Tastaturmodul angeschlossen ist. Die kleine Zahl in der oberen linken Ecke ist die Tastennummer, die in der Layout-Definitionsdatei (layout.js) verwendet wird. Die ausgegrauten, in den Ausweis integrierten Tasten haben eine feste Zuordnung, die nicht geändert werden kann. Sie werden nur zu Referenzzwecken angezeigt. Um die Hardware-Tastennummern den in layout.js verwendeten Nummern zuzuordnen, drücken Sie die Taste auf Ihrem Tastenfeld, die blau hervorgehoben ist.



Die erfasste Hardware-Tastennummer ist nun in der Mitte der Taste (12) sichtbar. Anschließend wird die nächste Taste markiert. Fahren Sie fort, bis Sie alle Tasten gedrückt haben. Sie können jederzeit eine Taste mit der Maus auswählen, um fehlerhafte Eingaben zu korrigieren. Wenn Sie alle Tasten gedrückt haben, klicken Sie auf die Schaltfläche „Download“. Dadurch wird eine Datei „mapping.js“ in Ihrem Download-Ordner gespeichert. Kopieren Sie diese Datei auf Ihre Tastatur. Befindet sich eine „mapping.js“-Datei auf der Tastatur, gibt die Tastatur keine Hardware-Tastennummern mehr aus, sondern die Tastencodes, die durch die Regeln in „layout.js“ definiert sind.

Das Badge kann in zwei Modi betrieben werden:

- *Ohne das Tastaturmodul. In diesem Fall wird die Beschreibung aus „standalone_layout.js“ gelesen.*
- *Mit dem Tastaturmodul wird die Beschreibung aus „layout.js“ gelesen. Falls du dich fragst, warum diese und andere Dateien die Endung .js haben: Sie können als JavaScript in eine lokal geöffnete HTML-Seite geladen werden. Beispiel: PinMapper.html nutzt diesen Mechanismus, um die Tastenpositionen aus keypositions.js auszulesen. Derzeit gibt es noch kein HTML-basiertes Tool zur Verwaltung von layout.js und standalone_layout.js, aber vielleicht wird es in Zukunft eines geben.*

Abschnitte

layout.js und standalone_layout.js bestehen aus Abschnitten. Jeder Abschnitt beginnt mit einem Abschnittsschlüsselwort in einer Zeile und endet mit der darauf folgenden Leerzeile. Es gibt drei verfügbare Abschnittstypen:

- *keyboard*
- *layout*

- *layer Die Reihenfolge und die Anzahl sind wichtig. Es beginnt mit einem einzelnen „keyboard“-Abschnitt, gefolgt von einem oder mehreren „layout“-Abschnitten, die jeweils einen oder mehrere „layer“-Abschnitte enthalten.*

„keyboard“-Abschnitt
Der „keyboard“-Abschnitt definiert einige Timeouts, die für alle Tasten gelten.

```
keyboard
tap_timeout=0.15
long_tap_timeout=0.4
```

Um diese Werte zu erläutern, möchte ich kurz auf verschiedene Aktionsauslöser eingehen. Die Firmware unterstützt für jede Taste drei Auslöser:

- *tap*
- *long_tap*
- *hold Eine Tap-Aktion wird ausgelöst, wenn Sie eine Taste drücken und wieder loslassen, bevor tap_timeout (in Sekunden) abgelaufen ist. Eine Long-Tap-Aktion wird ausgelöst, wenn Sie eine Taste länger als `tap_timeout`, aber noch vor Ablauf von `long_tap_timeout` (in Sekunden) loslassen. In Kombination mit der Shift-Aktion, die später beschrieben wird, können Sie beispielsweise einen Großbuchstaben eingeben, indem Sie die Taste einfach etwas länger gedrückt halten. Eine Halte-Aktion wird sofort ausgelöst, wenn Sie eine Taste drücken. Wenn Sie die Taste vor Ablauf von long_tap_timeout loslassen, wird die Halte-Aktion beendet, bevor ein Antippen oder langes Antippen ausgelöst wird. Wenn Sie die Taste länger als long_tap_timeout gedrückt halten, wird die Halte-Aktion beendet, sobald Sie die Taste loslassen. Ihre Aktion hat keinen Einfluss auf einen Tap oder Long-Tap, wenn sie Modifikatortasten ausgibt oder zu einer anderen Ebene wechselt. Sie fragen sich wahrscheinlich, wozu das gut ist? Mit der Hold-Aktion können Sie Tasten, die normalerweise zum Eingeben von Zeichen verwendet werden, Modifikatoren zuweisen. Hold-Aktionen werden auch verwendet, um bestimmte Ebenen zu aktivieren.*

Layout-Abschnitt
Ein Layout-Abschnitt hat nur eine Eigenschaft: den Titel des Layouts. Dieser Titel kann in einer ChangLayer()-Aktion verwendet werden, um auf eine Ebene zu verweisen.

```
layout
title=Booklover
```

Abschnitt „Layer“
Ein Layer hat einen Titel. Er enthält Regeln für Aktionsauslöser (Tippen, Langes_Tippen oder Halten). Es muss mindestens eine Ebene geben – die Basisebene – und du kannst mehrere zusätzliche Ebenen definieren. Es kann jeweils nur eine Ebene aktiv sein. Es müssen jedoch nicht für alle Tasten auf jeder Ebene Regeln definiert werden. Für Tasten ohne Regel wird die Regel aus der Basisebene verwendet. Hier ist ein Beispiel mit allen drei Aktionsauslösern und einigen der Aktionen:

```
layer
title=Base
1 : tap=NextLayout : long_tap=NextLayer : title=Next Layout\nNext
```

```
Layer
2 : tap=Codes[ ESCAPE ] : title=Escape
3 : tap=Codes[ ENTER ] : long_tap=ResetKeyboard :
title=Enter\nZurücksetzen
4 : tap=Codes[ RIGHT_ARROW ] : title=->
5 : tap=Codes[ LEFT_ARROW ] : title=<-
```

Eine Zeile, die Regeln für eine Taste beschreibt, beginnt mit der Tastennummer, gefolgt von mindestens einer Regel. Eine Regel besteht aus dem Namen des Aktionsauslösers (tap, long_tap oder hold), gefolgt vom Gleichheitszeichen und der Aktion. Regeln werden durch Doppelpunkte getrennt. Jede Zeile kann einen optionalen Titelteil enthalten. Wenn ein Titel angegeben wird, wird er auf dem Display für die Taste angezeigt. Sie können bis zu drei Zeilen verwenden, indem Sie Zeilenumbrüche mit \n angeben.

Verfügbare Aktionen

Codes[< Tastencode >, < Tastencode >, …] gibt die angegebenen Tastencodes gleichzeitig aus. Alle verfügbaren Tastencodes findest du unter Keycodes.html. Sequence[< Aktion > ; < Aktion >] gibt die angegebenen Aktionen nacheinander aus. Sequenzen sind derzeit nicht verschachtelbar und wurden bisher nur mit Code-Aktionen getestet. Hinweis: Das Trennzeichen zwischen Aktionen innerhalb von Sequenzen ist ein Semikolon. „Shift“ kann durch ein langes Tippen ausgelöst werden und ist nur sinnvoll, wenn eine Tipp-Aktion vorhanden ist. Es löst die Tipp-Aktion aus und gibt gleichzeitig den Tastencode für die Umschalttaste aus. Dies wird verwendet, um bei einem langen Tippen Großbuchstaben auszugeben. „NextLayout“ aktiviert die erste Ebene des nächsten Layouts. „PreviousLayout“ aktiviert die erste Ebene des vorherigen Layouts. „SwitchLayout(< Layoutname >)“ aktiviert dauerhaft das Layout mit dem angegebenen Namen. „NextLayer“ aktiviert die nächste Ebene des aktiven Layouts. „PreviousLayer“ aktiviert die vorherige Ebene des aktiven Layouts.

Layout-Definitionsdateien

ChangeLayer(< Ebenenname >) aktiviert vorübergehend die Ebene mit dem angegebenen Namen, solange der Auslöser (Tippen oder Gedrückthalten) aktiv ist. Hinweis: Das Display zeigt vorübergehende Ebenenwechsel nicht an, da es dafür zu langsam ist. SwitchLayer(< Ebenenname >) aktiviert dauerhaft die Ebene mit dem angegebenen Namen. ResetKeyboard startet das Badge neu bzw. setzt es zurück.

Startverhalten

Die Firmware verfügt über einen Wartungsmodus, in dem ein USB-Speicherstick mit allen Konfigurationsdateien angezeigt wird. Die Tastatur befindet sich automatisch im Wartungsmodus, wenn sich keine „mapping.js“-Datei auf dem Speicherstick befindet. Sie können den Wartungsmodus manuell aufrufen, indem Sie die Tastatur zurücksetzen oder erneut an den Computer anschließen, während Sie eine der integrierten Tasten des Badges gedrückt halten. Wenn Sie die Tastatur immer im Wartungsmodus starten möchten, setzen Sie „maintenance_mode“ am Anfang der Datei „boot.py“ auf „True“.

```
maintenance_mode = True
```