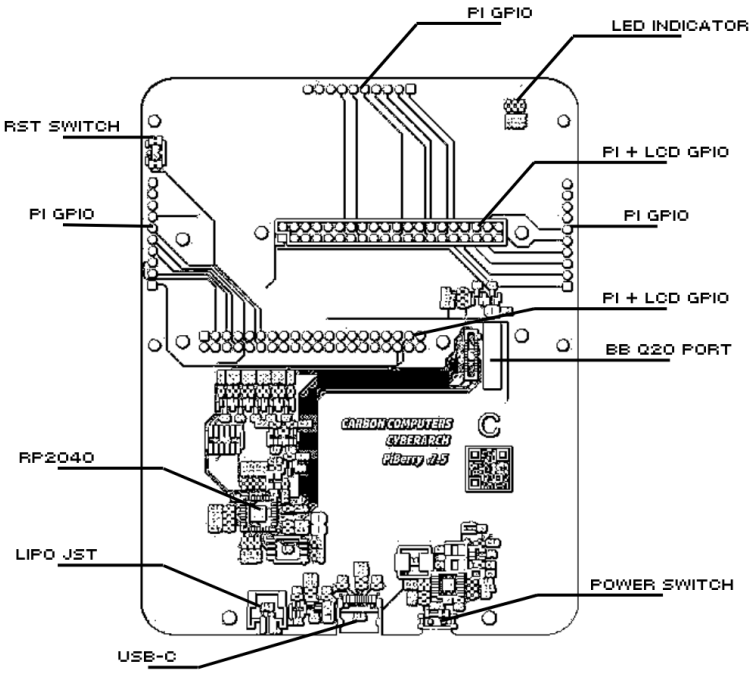
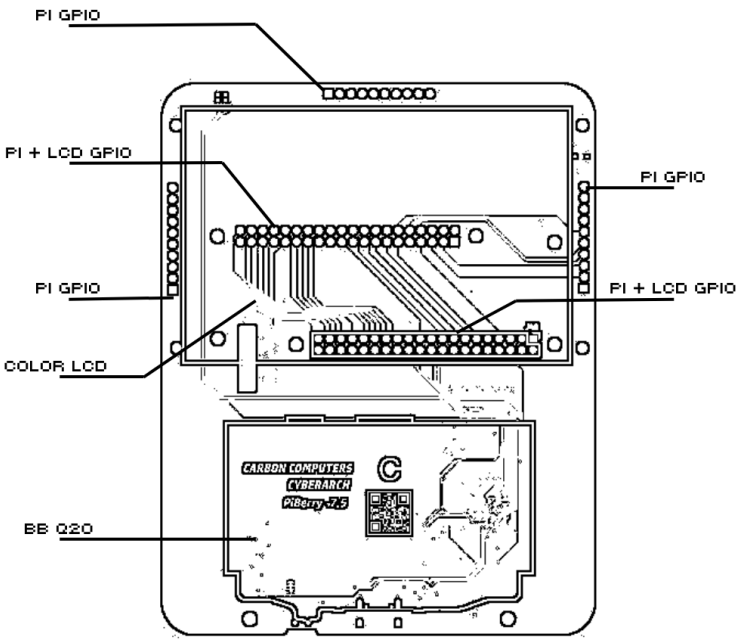
PiBerry v7.5

Build guide





CYBERARCH GIESCH

Getting Started

Follow these steps to build and set up your PiBerry. This guide is applicable whether you purchased the unit as a DIY kit or if you want to make upgrades or build a different Pi image. If you encounter any issues, please contact us for assistance.

Building the Hardware

If you are building your own PiBerry, the first step is to replace the GPIO female header on the display with a set of male GPIO headers. While it is possible to trim the preinstalled female header on the LCD to mount and solder onto the PCB, this process is delicate and could damage the screen.

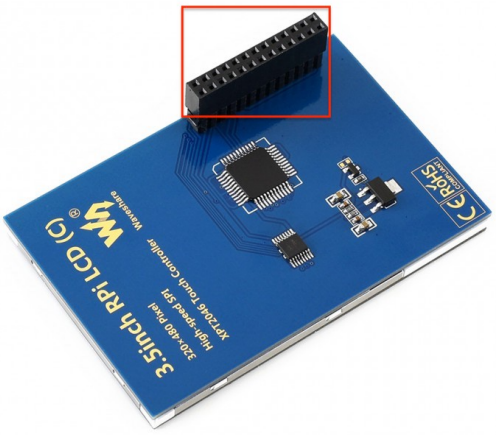
Recommended Tools for DIY PiBerry Build:

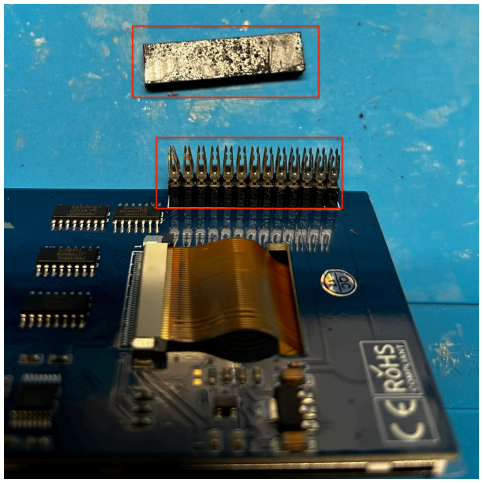
- Soldering iron
- Solder
- Flux
- Heat gun
- Tweezers
- Wire cutter
- Male GPIO headers
- Soldering pump and wick (for correcting mistakes and cleaning up)

PiBerry Parts:

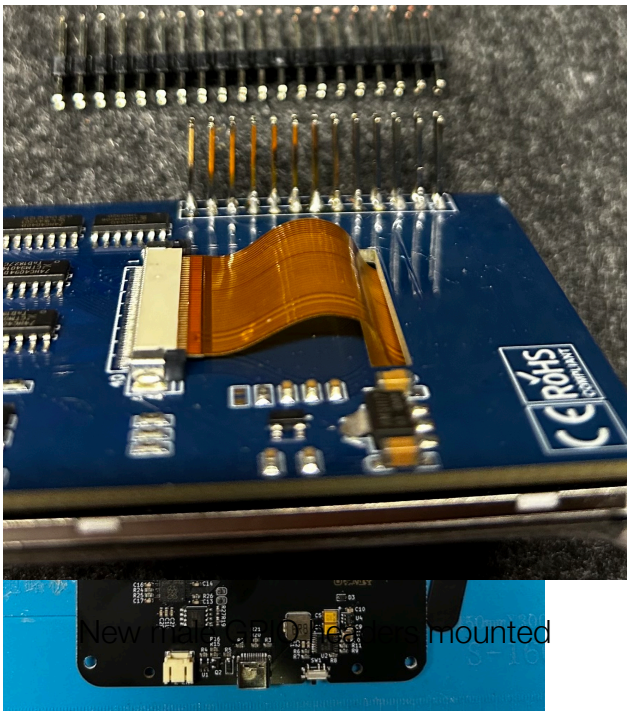
- PCB (Printed Circuit Board)
- LCD (Liquid Crystal Display)
- BB K20 keyboard
- Pi Zero 2W
- MicroSD card (16GB or higher, high speed recommended)
- Optional Battery & 3D printed case
- USB-C cable (for powering or flashing the PCB)

LCD Female GPIO Header Removal: To remove the female GPIO header from the LCD, apply heat to the top plastic cover as shown in the instructions. This will help you safely remove the header without damaging the screen.





Exposed female headers



New male GPIO headers mounted

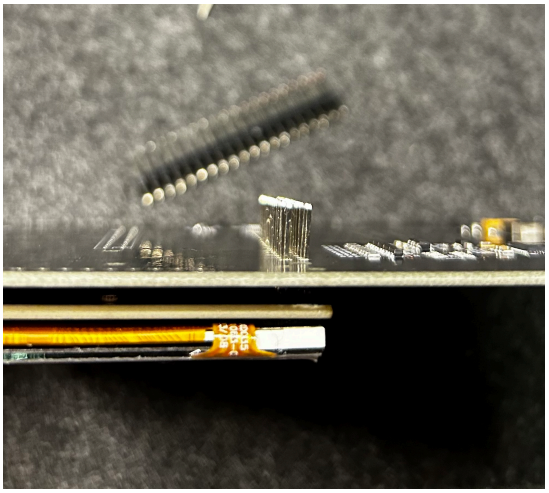
Mounted LCD with Kapton tape

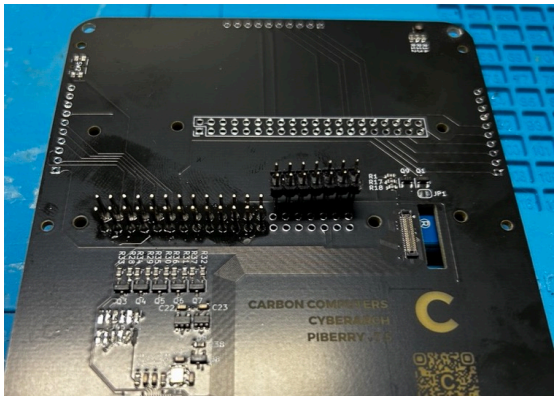
After removing the plastic top GPIO cover, you can continue by applying slight heat at an angle to soften and remove the remaining bottom plastic. Use tweezers to carefully extract the softened plastic.

Alternative Method: You may also trim the metal female header pins to remove the remaining plastic. The remaining metal pins can then serve as the male GPIO for the display. Note that some pins might be bent and may require adjustments to properly fit and mount onto the PiBerry PCB.

Important Note:

- If using a heat gun, be cautious with the amount of heat applied. Use only slight heat from above and the sides of the header to avoid overheating.
- Be careful not to damage any components by applying excessive heat.





Testing

Once the LCD has been modified with male headers, it's time to mount the Pi to the LCD for a preliminary test. It is good practice to verify that the display works before you start soldering. The Pi can be mounted loosely to the LCD, and some adjustments may be needed to ensure a proper connection. **Note** that the Pi will not boot without an SD card that has been flashed with an OS.

For initial testing, we recommend using a lightweight OS or the pre-built images provided by the LCD vendor for an easy setup as it includes display drivers. This helps verify that LCD has not been damaged, check colors and video quality.

Important Notes:

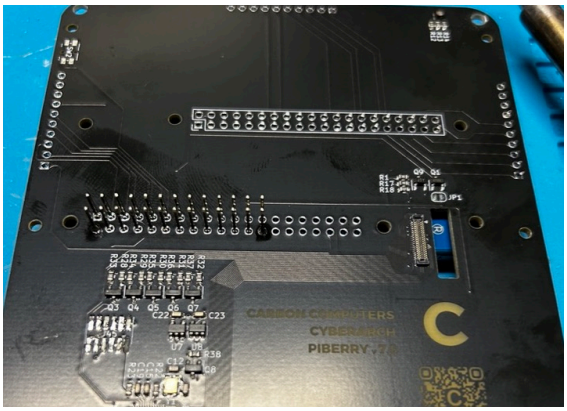
- Check the manufacturer's manual for proper LCD mounting to avoid short circuits or damage.
- Ensure the SD card is properly flashed with a compatible OS for testing.

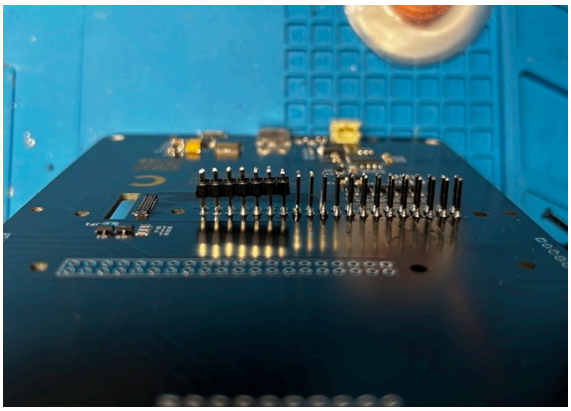
Once the display has been tested and confirmed to work, it can be securely mounted and soldered to the PiBerry PCB. Note that the rightmost 12 empty GPIO ports on the PCB will require GPIO pins to be soldered for the keyboard to communicate with the Pi (see images below).

By following these steps, you can ensure a smooth and successful setup of your PiBerry display and keyboard.

Make sure all the mounting pins are properly aligned, and that the display is leveled before soldering.

Images of the LCD male GPIO pins soldered to the PiBerry PCB





Left side plastic being removed from the additional GPIO header

After soldering the LCD to the PiBerry and adding and soldering the additional GPIO pins, it's time to mount the Pi for another test.

Important Notes:
The PCB will not power the system until it is flashed with the appropriate firmware. If you ordered the PiBerry from us, it should already be flashed. For instructions on how to flash the firmware, see the **Firmware Update** section.

Final Steps and Testing

If the last test passes, the Pi Zero can be soldered to the PCB using the male GPIO headers from the LCD. As seen in the photos, it is advisable to add some tape or an insulator to prevent a possible short circuit between the Pi and the extra GPIO ports on the PiBerry PCB. We used yellow Kapton tape, but any insulating tape will work.

Important Notes:

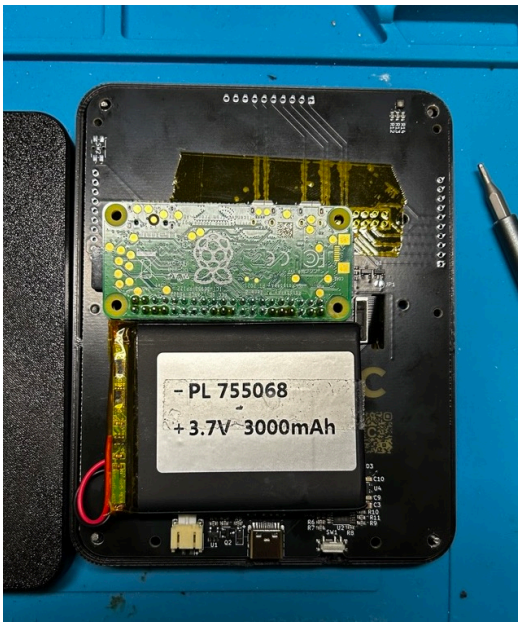
- The LCD should now be soldered to the PCB.
- The LCD GPIO header should be used to mount and solder the Pi Zero as shown below.

Lastly, the battery can be connected for use or charging. Before connecting, ensure the polarity is correct. Many LiPo batteries have incorrect polarity. The red battery wire should align with the + and the **black** wire with the -.

If the polarity is reversed, use tweezers to lift the plastic tab lock on the battery JST header, then pull the wire to switch positions.

Warning:

- Be careful not to short or contact the two battery wires while modifying the connection JST header, as this can damage the battery.



Firmware Update

Update PiBerry PCB firmware to ensure driver compatibility (boards come pre-flashed), without the firmware the keyboard, charging, and other functions will not work (we use the same firmware as Beepy - SQFM):

- Download the latest firmware image
- Slide the power switch off (to the left)
- Connect the PiBerry to your computer via USB-C
- While holding the "End Call" key (top right on the keypad), slide the power switch on
- The PiBerry will present itself as a USB mass storage device, drag'n'drop the new firmware (i2c_puppet.uf2) into the drive and it will reboot with the new firmware
- Give it about 10 seconds to install, you may reboot or reset the power

Choose an Operating System

We have tested three operating systems to run on your PiBerry. If you order the SD card from us, it will come preloaded with the display drivers, Kali Linux, and a desktop environment that includes our compatible documentation and tools from our CyberDeck builds. Ready to use images will be added soon for DIY builders, contact us for assistance.

Kali Linux: A powerful and user-friendly hacking OS loaded with a comprehensive suite of security tools. Ideal for penetration testing and cybersecurity enthusiasts.

Raspbian (Raspberry Pi OS): A versatile and customizable distribution with a familiar interface. Packed with features and applications, it's perfect for general use and development on the Raspberry Pi.

Buildroot: A streamlined, Beepy-centric image designed for fast boot times and minimal resource usage. Excellent for projects that require a lightweight and efficient operating system.

Setting up a Raspbian or Kali System

1. Use the Raspberry Pi Imager tool to flash an SD card with the Raspberry Pi OS *Lite* image or Kali for Pi Zero 2w

- Choose OS - Raspberry Pi OS (other) - *Raspberry Pi OS Lite (32-bit) image or from Specific Purpose OS - Kali*
- Click the gear icon ⚙ (or press CTRL + SHIFT + X) to set the username, password, WiFi, and enable SSH
- Make sure your computer and the Pi are on the same WiFi network in order to SSH in later

2. SSH into your PiBerry and install the driver packages

First Steps: Display Driver Installation

Before getting started with your chosen operating system, you need to install the display driver. The process can vary depending on the display model. We've tested it on models A, B, and C, with Model C offering a high refresh rate of 125MHz, making it ideal for gaming or desktop OS use.

After installing the display driver, you may notice that Kali OS will no longer boot to the desktop environment and will only start in terminal mode. This is expected, and we are working with the vendor on a solution.
If you use the pre-made Pi images, they will include a desktop environment. However, you may need to edit the touchpad mapping to function as a cursor.

The display driver installation process is generally similar for all models. For detailed information on calibration and rotation, please check the provided link.

Installing the LCD Driver

```
#clone the repo
git clone https://github.com/waveshare/LCD-show.git
cd LCD-show/
```

#add permissions & install

```
chmod +x LCD35C-show
./LCD35C-show
```

A. Clone & Install Keyboard Driver for RaspBerry OS lite

```
sudo apt-get update && sudo apt-get install raspberrypi-kernel
sudo shutdown -r now
curl -s https://raw.githubusercontent.com/carboncomputers/piberry/main/pi/setup.sh | bash
sudo shutdown -r now
```

B. Clone & Install Keyboard Driver for Kali

```
sudo raspi-config
```

05 - **Interfacing Options** -> P4 **- I2C** (**Enable**)

```
sudo reboot
```

B Make the following edits for Kali

#after reboot, clone and edit the following config

```
git clone https://github.com/sqfmi/bbqX0kbd_driver.git
```

```
sudo nano bbqX0kbd_driver/src/main.c
        **Comment line 92 with  //
       // .remove = beepykbdremove,
save and exit (ctrl + s and ctrl + x)
```

```
 GNU nano 8.0                                    bbqX0kbd_driver/src/main.c *
        bbqX0kbd_shutdown(i2c_client);
}

// Driver definitions

// Device IDs
static const struct i2c_device_id bbqX0kbd_i2c_device_id[] = {
        { "beepy-kbd", 0, },
        { }
};
MODULE_DEVICE_TABLE(i2c, bbqX0kbd_i2c_device_id);
static const struct of_device_id bbqX0kbd_of_device_id[] = {
        { .compatible = "beepy-kbd", },
        { }
};
MODULE_DEVICE_TABLE(of, bbqX0kbd_of_device_id);

// Callbacks
static struct i2c_driver bbqX0kbd_driver = {
        .driver = {
                .name = "beepy-kbd",
                .of_match_table = bbqX0kbd_of_device_id,
        },
        .probe    = bbqX0kbd_probe,
        .shutdown = bbqX0kbd_shutdown,
        //.remove   = bbqX0kbd_remove,
        .id_table = bbqX0kbd_i2c_device_id,
};
```

sudo nano bbqX0kbd_driver/beepy-kbd.dts

```
 GNU nano 8.0                                    bbqX0kbd_driver/beepy-kbd.dts
// SPDX-License-Identifier: GPL-2.0
/dts-v1/;
/plugin/;

/{
        compatible = "brcm,bcm2835";

        fragment@0 {
                target = <&i2c1>;

                __overlay__ {
                        #address-cells = <1>;
                        #size-cells = <0>;
                        beepy_kbd: beepy_kbd@1f {
                                compatible = "beepy-kbd";
                                reg = <0x1F>;
                                irq-gpio = <&gpio 11 0x2>;
                                interrupts = <11 2>;
                                interrupt-parent = <&gpio>;
                                interrupt-controller;
                                #interrupt-cells = <2>;
                        };
                };
        };
        __overrides__ {
                irq_pin = <&beepy_kbd>,"interrupts:0";
        };
};
```

Edit the beepy-kbd.dts file, add #interrupt-cells in **blue**

#make & build to install

cd bbqX0kbd_driver

make

```
┌──(kali㉿kali-raspberry-pi-zero-2-w)-[~/bbqX0kbd_driver]
└─$ sudo make install

make -C '/lib/modules/5.15.44-Re4son-v7+/build' M='/home/kali/bbqX0kbd_driver' modules_install
make[1]: Entering directory '/usr/src/linux-headers-5.15.44-Re4son-v7+'
  INSTALL /lib/modules/5.15.44-Re4son-v7+/extra/beepy-kbd.ko
  DEPMOD  /lib/modules/5.15.44-Re4son-v7+
Warning: modules_install: missing 'System.map' file. Skipping depmod.
make[1]: Leaving directory '/usr/src/linux-headers-5.15.44-Re4son-v7+'
# Rebuild dependencies
depmod -A
# Install keymap
mkdir -p /usr/share/kbd/keymaps/
install -D -m 0644 ./beepy-kbd.map /usr/share/kbd/keymaps/
# Install device tree overlay
install -D -m 0644 ./beepy-kbd.dtbo /boot/overlays/
# Add configuration line if it wasn't already there
grep -qxF 'dtoverlay=beepy-kbd,irq_pin=4' /boot/config.txt \
       || printf '[all]\ndtparam=i2c_arm=on\ndtoverlay=beepy-kbd,irq_pin=4\n' >> /boot/config.txt
# Add auto-load module line if it wasn't already there
grep -qxF 'beepy-kbd' /etc/modules \
       || echo 'beepy-kbd' >> /etc/modules
# Configure keymap as default
grep -qxF 'KMAP=/usr/share/kbd/keymaps/beepy-kbd.map' /etc/default/keyboard \
       || echo 'KMAP=/usr/share/kbd/keymaps/beepy-kbd.map' >> /etc/default/keyboard
rm -f /etc/console-setup/cached_setup_keyboard.sh
dpkg-reconfigure keyboard-configuration \
       || echo "dpkg-reconfigure failed, keymap may not be applied"
```

Successful make install

dtc -@ -I dts -O dtb -W no-unit_address_vs_reg -o beepy-kbd.dtbo beepy-kbd.dts

sudo cp beepy-kbd.dtbo /boot/overlays/
sudo make install

—(setup screen may start, press tab, Ok for each screen)

```
┌──(kali㉿kali-raspberry-pi-zero-2-w)-[~/bbqX0kbd_driver]
└─$ make
make -C '/lib/modules/5.15.44-Re4son-v7+/build' M='/home/kali/bbqX0kbd_driver'
make[1]: Entering directory '/usr/src/linux-headers-5.15.44-Re4son-v7+'
warning: the compiler differs from the one used to build the kernel
  The kernel was built by: gcc (Debian 11.2.0-19) 11.2.0
  You are using:           gcc (Debian 13.2.0-24) 13.2.0
  CC [M]  /home/kali/bbqX0kbd_driver/src/main.o
/home/kali/bbqX0kbd_driver/src/main.c:62:13: warning: ‘bbqX0kbd_remove’ defined but not used [-Wunused-function]
   62 | static void bbqX0kbd_remove(struct i2c_client* i2c_client)
      |             ^~~~~~~~~~~~~~~
  CC [M]  /home/kali/bbqX0kbd_driver/src/input_iface.o
  CC [M]  /home/kali/bbqX0kbd_driver/src/params_iface.o
  CC [M]  /home/kali/bbqX0kbd_driver/src/sysfs_iface.o
  CC [M]  /home/kali/bbqX0kbd_driver/src/ioctl_iface.o
  LD [M]  /home/kali/bbqX0kbd_driver/beepy-kbd.o
  DTC     /home/kali/bbqX0kbd_driver/beepy-kbd.dtbo
  MODPOST /home/kali/bbqX0kbd_driver/Module.symvers
  CC [M]  /home/kali/bbqX0kbd_driver/beepy-kbd.mod.o
  LD [M]  /home/kali/bbqX0kbd_driver/beepy-kbd.ko
make[1]: Leaving directory '/usr/src/linux-headers-5.15.44-Re4son-v7+'
```

sudo reboot

Successful make file

Keyboard and Display should now function

(default login: 'kali' & default pwd: ‘kali’)

Features

Powering on/off your PiBerry

Long holding the "**End Call**" key (5 seconds) will trigger the key KEY_POWER and safely shutdown the Pi. The LED will turn red until OS shutdown has completed.
Please wait another few seconds until the disk activity light has turned off to ensure disks are synced.
After shutting down using the "End Call" key, holding the key for 1 second will turn the Pi back on.

Changing Font Size

In /boot/cmdline.txt, edit fbcon=font:VGA8x8 to change the font/size. The next largest size is VGA8x16. See fbcon for more details.

Keyboard - Firmware

The keyboard firmware is a fork of **i2c_puppet** with a few additional features to support the PiBerry.
You can download the latest version of the firmware here: https://github.com/ardangelo/beepberry-rp2040/releases/latest/download/i2c_puppet.uf2

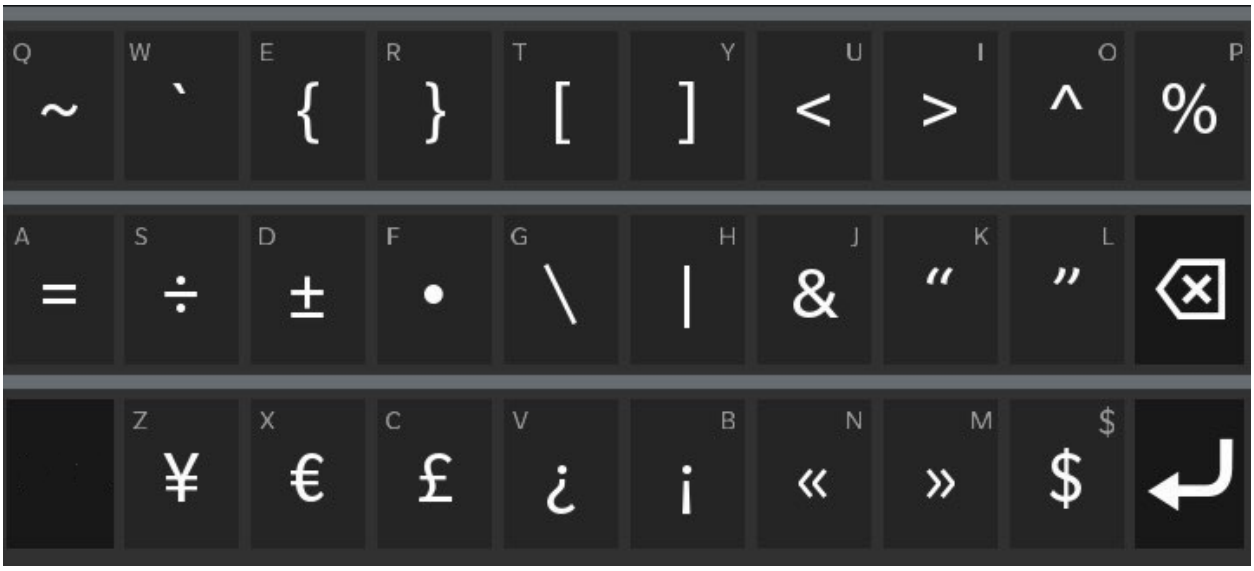
Basic key mappings

- **Call** is mapped to **Control**
- "**Berry**" key is mapped to Tmux prefix (customize the prefix in the keymap file)
- **Touchpad** click enters **Meta** mode (see the section on Meta mode).
- **Double** click enters touchpad scroll mode
- **Back** is mapped to Escape
- Holding "**End Call**" safely shuts down the Pi
- Physical **Alt** is mapped to symbols printed on the keycap
- **Symbol** is mapped to AltGr (Right Alt), mapped to more symbols via the keymap file
- Physical **Alt** + **Enter** is mapped to Tab

Alt and Sym modifiers

The alternate symbols printed directly on the Beepy keys are triggered by pressing the physical Alt key, then the key on which the symbol is printed. For additional symbols not printed directly on the keys, the Sym key is used.

Symbol key map



Sticky modifier keys

The keyboard driver supports sticky modifier keys. Holding a modifier key (Shift, Alt, Sym) while typing an alpha keys will apply the modifier to all alpha keys until the modifier is released.
One press and release of the modifier will enter sticky mode, applying the modifier to the next alpha key only. If the same modifier key is pressed and released again in sticky mode, it will be canceled.
Visual mode indicators are drawn in the top right corner of the display, with indicators for Shift, Physical Alt, Control, Alt, Symbol, and Meta mode.

Meta mode

Meta mode is a modal layer that assists in rapidly moving the cursor and scrolling with single keypresses. To enter Meta mode, click the touchpad button once. Then, the following keymap is applied, staying in Meta mode until dismissed:

- E: up, S: down, W: left, D: right

    - Why not WASD? This way, you can place your thumb in the middle of all four of these keys, and more fluidly move the cursor without mistyping
- R: Home, F: End, O: PageUp, P: PageDown
- Q: Alt+Left (back one word), A: Alt+Right (forward one word)
- T: Tab (dismisses Meta mode)
- X: Apply Control to next key (dismisses Meta mode)
- C: Apply Alt to next key (dismisses Meta mode)
- 0: Toggle display black/white inversion
- N: Decrease keyboard backlight brightness
- M: Increase keyboard backlight brightness
- $: Toggle keyboard backlight
- Touchpad click (while in Meta mode): Enable touchpad scroll mode (up and down arrow keys)
    - Other Meta mode keys will continue to work as normal
    - Exiting meta mode will also exit touchpad scroll mode
    - Subsequent clicks of the touchpad will type Enter.
- Esc: ("Back" button): exit meta mode

Typing any other key while in Meta mode will exit Meta mode and send the key as if it was typed normally.

sysfs Interface

The following sysfs entries are available under /sys/firmware/beepy:

- led: 0 to disable LED, 1 to enable. Write-only
- led_red, led_green, led_blue: set LED color intensity from 0 to 255. Write- only
- keyboard_backlight: set keyboard brightness from 0 to 255. Write-only
- battery_raw: raw numerical battery level as reported by firmware. Read-only
- battery_volts: battery voltage estimation. Read-only
- battery_percent: battery percentage estimation. Read-only

Module parameters

Write to /sys/module/beepy_kbd/parameters/<param> to set, or unload and reload the module with beepy-kbd param=val.

- touchpad: one of meta or keys
- meta: default, will use the touchpad button to enable or disable Meta mode
- keys: touchpad always on, swiping sends arrow keys, clicking sends Enter

Custom Keymap

The Alt and Sym keymaps and the Tmux prefix sent by the "Berry" key can be edited in the file /usr/share/kbd/keymaps/beepy-kbd.map. To reapply the keymap without rebooting, run loadkeys /usr/share/kbd/keymaps/beepy-kbd.map.

Join our Discord & Beepy's

CyberArch Invite

Beepy Invite

SQFMI