# PropellerHat-Controller

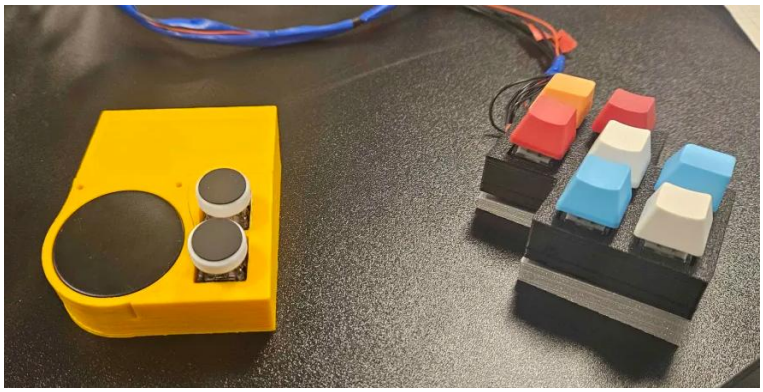
Development of assistive controller devices designed to support people with motor impairments. The project focuses on accessible input solutions using customizable hardware and firmware, including a trackpad-based controller and a button-based controller.

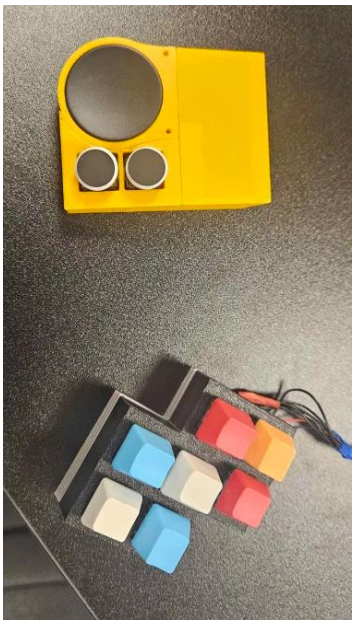
---

## Team PropellerHat


---

## Final Prototyp





*Date 21.01.2026*

# Summary

# What was the Problem?

The ASSIST HEIDI project was initiated to address the needs of a user named **Oliver**, who experiences **physical limitations** that affect his ability to use standard gaming input devices comfortably and for extended periods of time. Due to **reduced fine control and rapid fatigue**, prolonged use of a conventional mouse and keyboard is physically demanding and often not feasible, especially during longer gaming sessions.

Oliver expressed a strong interest in playing **fast-paced shooter games**, which typically require precise cursor control, quick reaction times, and frequent button presses. Standard input devices are not well suited to his situation, as they often require sustained hand positioning, repetitive movements, and continuous input actions. These requirements can quickly lead to exhaustion and significantly limit gameplay duration and comfort.

During initial discussions with the user, it became clear that an **alternative assistive controller solution** was needed—one that would reduce physical strain while still allowing accurate, responsive, and reliable input suitable for shooter games. The solution needed to be easy to operate, adaptable, and customizable in order to match individual capabilities and preferences.

To build upon existing experience and proven concepts, the project team decided to base the new solution on an earlier controller project. From this starting point, the design was further refined in close collaboration with the user. As a result, two complementary controller concepts were developed:

- a **button-based mini keyboard controller**, focusing on flexible and customizable key input, and

- a **trackpad-based controller**, enabling precise cursor movement combined with two easily accessible buttons.

Together, these controllers aim to provide accessible and flexible input options that better match the user's physical capabilities, enabling more comfortable and sustainable gameplay while maintaining the responsiveness required for fast-paced shooter games.

# Detailed Description

## What was solved? What can be done with the solution?

Within the ASSIST HEIDI project, an assistive controller solution was developed to address challenges faced by users who experience **rapid fatigue and reduced precision** when using conventional gaming input devices. Standard mouse and keyboard setups require continuous hand positioning, precise movements, and frequent button presses, which can quickly lead to physical strain and exhaustion, particularly in demanding gaming scenarios.

To overcome these limitations, the project team designed **two complementary controller solutions** that provide alternative input methods while significantly reducing physical effort. Both controllers were developed in close coordination with the user and build upon experience gained from an earlier controller project.

The first solution is a **trackpad-based controller**, which combines a capacitive trackpad for precise cursor movement with two easily accessible hardware buttons. This design allows accurate aiming and clicking actions with minimal hand movement and supports adjustable behavior to match individual needs.

The second solution is a **compact mini keyboard controller** featuring seven fully customizable buttons. Each button can be mapped to mouse or keyboard functions, enabling flexible input configurations tailored to user preferences and the specific requirements of different games.

Together, these controllers enable users to:

- control the mouse cursor with reduced physical strain,
- perform mouse clicks and trigger keyboard actions reliably, and
- engage in fast-paced shooter games as well as general computer use for extended periods with improved comfort.

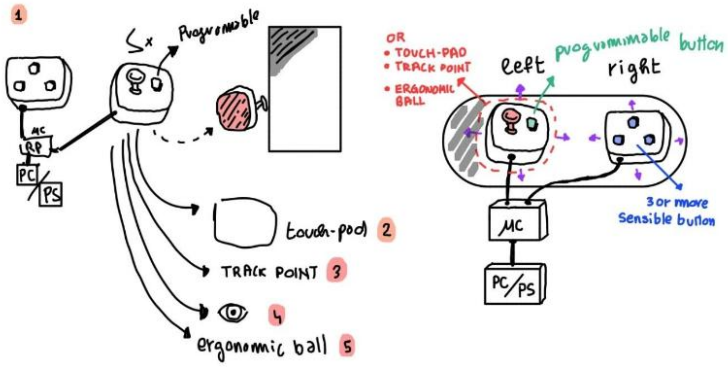
---

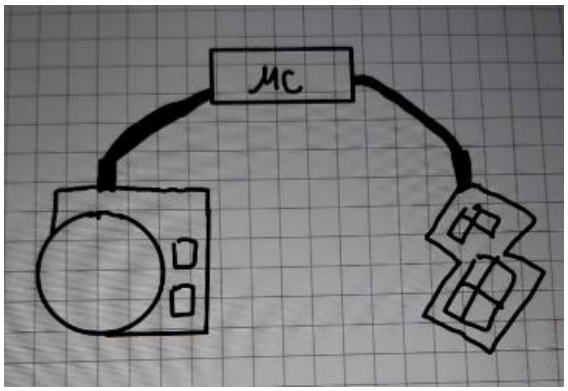
## Platforms and Compatibility

Both controllers are implemented as standard **USB Human Interface Devices (HID)**. When connected to a computer, they are automatically recognized by the operating system as a mouse and keyboard without requiring additional drivers or software installation.

During testing, the controllers functioned reliably across all tested platforms. Because the solution relies exclusively on the USB HID standard, it is **platform-independent** and compatible with any system that supports USB mouse and keyboard input.

## Sketches and Photos




Trackpad - Prototyp

Keyboard-Switch - Prototyp

## How can the solution be put into operation?

The solution is designed for simple and immediate operation. The finished controller is connected to the computer using a micro-USB cable. As soon as the device is connected, it is ready for use. All required input mappings are already configured in the firmware, so no additional setup or calibration steps are necessary.

### Controllers and Connection

Two different controller variants are available. The trackpad controller combines a capacitive trackpad for cursor movement with two configurable buttons that can be assigned to mouse or keyboard actions. The second variant is a seven-button controller, where each physical button can be mapped to a specific function. Both controllers are connected to the host system via USB and operate independently of each other.

## Table of Controls and Associated Actions

### Trackpad Controller

| Control Element | Function | Associated Action |
|---|---|---|
| Trackpad | Cursor movement | Mouse movement (X/Y axis) |
| Button 1 | Primary action | Left mouse click (configurable) |
| Button 2 | Secondary action | Right mouse click (configurable) |

### Seven-Button Controller

| Button | Default Function | Customizable Action |
|---|---|---|
| Button 1 | 0 | Combination With all other Buttons (1+1 action) possible |
| Button 2 | 1 | |
| Button 3 | 2 | |
| Button 4 | 3 | |
| Button 5 | 4 | |
| Button 6 | 5 | |
| Button 7 | 6 | |

All button assignments can be adapted in the firmware to match individual user needs or specific application requirements.

# Technical Documentation – How can it be reproduced by somebody else?

## CAD Design Files and 3D Printing

The controller housings and mechanical components are based on custom CAD designs developed specifically for this prototype. Their purpose is to define the physical placement of the trackpad, switches, and electronic modules while ensuring an ergonomic and accessible layout suitable for assistive interaction. The overall design prioritizes ease of access, clear separation of input elements, and stable mounting of all internal components.

In order to accelerate development, the mechanical structure did not originate entirely from scratch. Instead, we started from existing open-source designs and adapted them to meet the new functional and ergonomic requirements of this project. In particular, the switches controller was inspired by Special Gaming Controller -JBMK project (https://github.com/jackjburnett/JBMK-Controller), while the touchpad enclosure took reference from the Flippad project (https://github.com/asterics/FLipPad). These initial models were extensively modified to accommodate new dimensions, mounting points, and cable-routing needs.

The final mechanical system is composed of three separate elements:

### Touchpad Controller:

that consists of four distinct printed components:

1.1. Inclined Base and Lower Housing (Base_TouchPad.stl)
This part provides the ergonomic tilt and forms the lower shell of the enclosure.

1.2. Microcontroller Compartment Lid (MicroController_Lid_TouchPad.stl)
A removable cover that protects the microcontroller and allows access for wiring and maintenance.

1.3. Main Body with Touchpad Mount and Switch Compartments (Main_TouchPad.stl)
This central block integrates the mounting surface for the touchpad and the housing for the two auxiliary switches.

1.4. Touchpad Outer Frame (Top_TouchPad.stl)
A protective enclosure that secures the touchpad in place while leaving the active surface fully accessible.

### 7Switches Keyboard Controller:

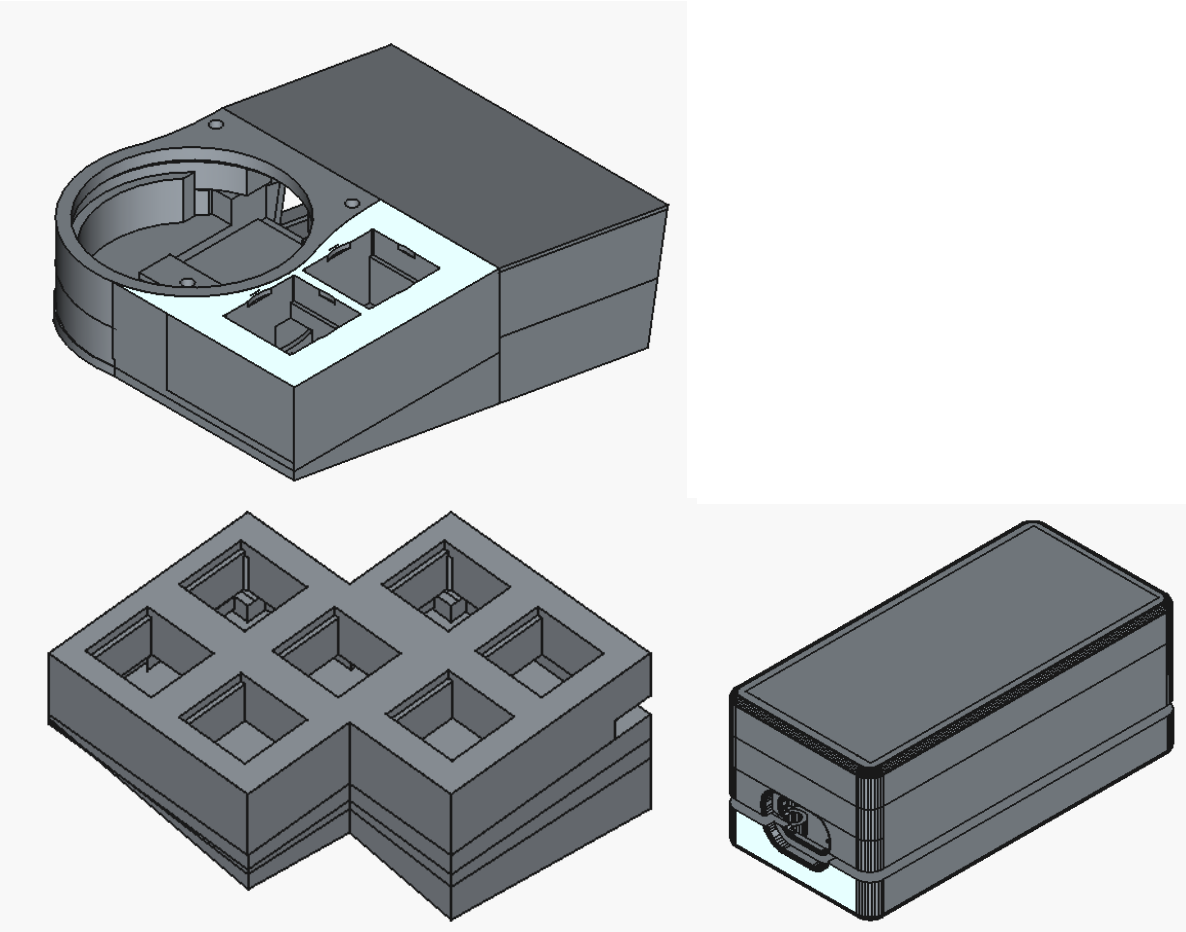
that consist of two parts:

1.5. Switch Mounting Base (Main_7Switches_.stl)
Designed to hold seven mechanical switches in a fixed and stable configuration.

1.6. Inclined Lower Base (Base_slope5_7Switches_.stl,Base_slope10_7Switches_.stl,Base_slope23_7Switches_.stl)
Available in two different tilt angles (height 5, 10 and 23), allowing users to select the most comfortable ergonomic orientation.

## Secondary Microcontroller Case:

1.7. Base Plate (Bottom_case.stl)
1.8. Top Cover (Top_case.stl)



All CAD source files (`.stl`) and the corresponding 3D-printing files (Total_Project_Printing.3mf) are available in our GitHub repository, allowing full reproducibility of the prototype.

## Printing setting

- Printing settings: 0.3mm QUALITY @0.g nozzle MK3
- Filament: Prusa PLA
- Printer: Original Prusa i3 MK3S & MK3S+ 0.6 nozzle
- Support: Everywhere
- Infill: 15%
- Brim: on

Estimated printing time: 5h28m

# Materials Needed and Availability

## Microcontroller

- ***2 × Raspberry Pi Pico W***
  *The Raspberry Pi Pico W is used as the main microcontroller for both controller variants. It provides native USB support for Human Interface Device (HID) functionality, sufficient processing performance, and a large number of GPIO pins suitable for handling multiple buttons and input devices.*
  *The Pico W is widely available from electronics distributors and online retailers.*

## Input Components

- ***Trackpad TM035035 with FlipPad accessory***
  *The capacitive trackpad is used for cursor movement and precise pointing control. The FlipPad accessory improves ergonomics and accessibility for assistive use.*
  *Supplier: Asterics Foundation*
  *Website: https://www.asterics-foundation.com/*
- ***9 × Cherry MX mechanical switches***
  *These switches are used in the button-based controller to provide reliable and tactile button input. Cherry MX switches are well known for their durability and consistent actuation behavior.*
  *Availability: Commonly available from electronics and mechanical keyboard suppliers.*
- ***BoaJAJUHU OEM Profile Blank PBT Keycaps (Rainbow Mixed Colors)***
  *Standard OEM-profile keycaps made from durable PBT material are used for the mechanical buttons. The blank design allows flexible labeling or user-specific customization.*
  *Availability: Online marketplaces and keyboard accessory stores.*
- ***2 × 3D-printed custom keycaps***
  *Two custom keycaps were reused from a previous controller design. The original design files are available online and can be reproduced if needed.*
  *Source:*
  *https://github.com/jackjburnett/JBMK-Controller/blob/main/CAD%20Files/ButtonCreation.ipynb*
- ***SensoryBoost DIY Grip Tape***
  *Grip tape is applied to selected surfaces to improve tactile feedback and handling, especially for users with reduced fine motor control.*
  *Availability: Assistive technology suppliers and online stores.*

### Mechanical Parts and Housing

- ***3D-printed enclosure and components***
  *The controller housing and internal mounting components were manufactured using FDM 3D printing.*
  ***Printing details (per enclosure):***
  - *Used filament: ~119.76 g*
  - *Estimated material cost: ~3.33 €*
  - *Estimated printing time:*
    - *Normal mode: ~5 h 18 min*
    - *Stealth mode: ~5 h 22 min*
- *Standard PLA filament was used. Other common filaments (e.g. PETG) can also be used depending on durability requirements.*
- ***Assorted screws***
  *A small number of standard screws were used to securely mount the electronic components and close the housing. No special or proprietary screws are required.*

## Estimated Total Cost of the Solution

*The total cost of the assistive controller solution is kept intentionally low in order to ensure accessibility and reproducibility for a broad user base.*

**Cost Breakdown (approximate)**

- ***2 × Raspberry Pi Pico W:*** *~20–24 €*
- ***Trackpad TM035035 with FlipPad accessory:*** *~25–30 €*
- ***9 × Cherry MX switches:*** *~5–7 €*
- ***Keycaps (OEM PBT + reused custom keycaps):*** *~5–8 €*
- ***SensoryBoost DIY Grip Tape:*** *~5 €*
- ***3D-printed enclosure (PLA filament):*** *~3.33 €*
- ***Screws and small mounting hardware:*** *~2 €*

**Estimated Total Cost:**

***💶 ~65–75 € per complete controller setup***

*This estimation assumes small-quantity purchases at common retail prices. Costs may be reduced further when sourcing components in bulk or reusing existing parts. No specialized or proprietary components are required, keeping the overall solution affordable and suitable for community-driven assistive technology projects.*

# Microcontroller and Electronic Components

## Trackpad Controller

*The trackpad controller is based on a **Raspberry Pi Pico W**, featuring the RP2040 microcontroller. This platform was selected due to its native USB support, sufficient processing performance, and a large number of available GPIO pins, making it well suited for Human Interface Device (HID) applications. The microcontroller handles all input processing and communicates mouse and keyboard events to the host system via the standard USB HID protocol.*

*Two momentary push buttons are directly connected to the microcontroller and serve as primary input actions. **Button 1 is connected to GPIO 0**, and **Button 2 is connected to GPIO 1**. Both buttons are configured using the internal pull-up resistors of the Raspberry Pi Pico W. Button inputs are processed in software and include debouncing as well as explicit detection of press and release events to ensure reliable operation.*

*The capacitive **TM035035 trackpad** is connected to the microcontroller using the interface defined by the trackpad driver implemented in the firmware. The exact pin assignment for the trackpad communication lines depends on the specific trackpad module and is defined within the source code. The trackpad is used for cursor movement and supports configurable behavior such as sensitivity adjustment and rotation handling.*

### Firmware Used

*The firmware for the trackpad controller is implemented using the **Arduino framework** and is available in the project's GitHub repository. It contains the complete logic required for capacitive trackpad input processing, button handling, rotation handling, and USB Human Interface Device (HID) communication.*

*The firmware enables the controller to operate as a standard USB mouse and keyboard without requiring any additional drivers or host-side software. All input events generated by the trackpad and the two buttons are translated into standard HID mouse movement, mouse clicks, and optional keyboard events, ensuring full compatibility with modern operating systems.*

## Configuration Files

*No external configuration files are required to operate the trackpad controller. All configuration parameters are defined directly within the firmware source code.*

*The firmware architecture separates **configuration** and **behavior**:*

- ***Header files (.h)** are used to define configurable parameters such as*
    - *button mappings,*
    - *rotation settings (e.g. 0°, 90°, 180°, 270°),*
    - *feature enables or disables.*
- ***Source files (.cpp)** contain the implementation logic and allow adjustment of*
    - *cursor speed,*
    - *sensitivity,*
    - *smoothing behavior,*
    - *trackpad response characteristics.*

*This structure allows the controller behavior to be easily adapted by modifying predefined constants without changing the overall program structure or firmware logic. The modular design also supports future extensions and user-specific customization.*

## Assembly and Operation

*To assemble the trackpad controller, the mechanical housing is first manufactured using the provided 3D printing files. The capacitive trackpad and the two push buttons are then mounted into the housing, followed by wiring the electronic components according to the wiring diagram.*

*After mechanical and electrical assembly, the firmware is flashed onto the **Raspberry Pi Pico W** via a USB connection. No additional programming hardware is required.*

*Once connected to a computer via USB, the controller is automatically recognized as a standard mouse and keyboard. The device is immediately ready for use and does not require any further configuration, calibration, or software installation on the host system.*

# Keyboard Controller and Chording Logic

## Microcontroller and Electronic Components

*The keyboard controller is powered by a Raspberry Pi Pico W, utilizing the RP2040 microcontroller. This hardware was chosen for its integrated USB support, high-speed processing, and extensive GPIO capabilities, which are essential for low-latency Human Interface Device (HID) tasks. The system manages all hardware interrupts and translates physical button states into standardized keyboard reports sent to the host via the USB HID protocol.*

*Ten momentary switches are connected to the microcontroller via GPIO pins 0 through 9. To minimize component count and simplify the circuit, all pins are configured using the RP2040's internal pull-up resistors. Input reliability is maintained through a software-based stabilization routine that introduces a 50ms delay; this ensures that multi-button chords have settled into a final state before the microcontroller broadcasts the corresponding keycode.*

### Firmware and Chording Functionality

*The firmware is implemented using CircuitPython and leverages the* `adafruit_hid` *library to provide "driverless" compatibility across all modern operating systems. The core of the firmware is a custom chording engine that allows the ten physical buttons to output a significantly larger set of commands.*

*The logic is divided into two primary tiers:*

- ***Direct Mapping:*** *Individual presses on GP0 through GP9 are mapped to the numeric keys 0 through 9.*
- ***Logical Chording:*** *Simultaneous presses (combos) are detected as a set. The firmware sorts these inputs and references a lookup table to trigger unique keycodes, including the full ISO alphabet (A–Z), function keys (F1–F12), and essential system keys like* ***SPACE****,* ***ENTER****, and* ***ESCAPE****.*

*This approach allows the device to function as a full-featured keyboard despite its compact physical interface.*

### Configuration and Modular Design

*The system follows a "configuration-as-code" philosophy. Keycode assignments, pin ordering, and combination mappings are defined within the main script for easy modification.*

- ***`PIN_ORDER`:*** *Defines the physical-to-logical pin relationship.*
- ***`COMBINATION_PINS_NUMERIC`:*** *A dictionary mapping button pairs to their respective HID Keycodes.*

*This modular structure allows users to redefine the entire keyboard layout by simply modifying the dictionary keys without altering the underlying input-processing logic.*

## Assembly and Operation

*Assembly involves mounting the ten tactile switches into a custom housing and wiring them to the corresponding GPIO pins on the Pico W. Because the system uses internal pull-ups, each switch is wired directly between its GPIO pin and a common ground.*

*Flashing the firmware is done by dragging and dropping the CircuitPython script onto the Pico W's mass storage drive. Once connected to a host, the device is instantly recognized as a standard USB keyboard. No special software is required on the computer, as all chording logic and debouncing are handled locally on the RP2040.*