

Intent Controller: detailed user guide

For the Android app.

Intent Controller lets you control a game using the inputs that suit you. Those inputs can be a normal controller, a keyboard, a mouse, a joystick, your face, switches connected to a board, or buttons on the phone screen.

You choose where the finished controller actions go. You then tell the app what each input should do.

For example, one foot switch could send both the aim and fire buttons. A different switch could send a short sequence of game buttons. A joystick could handle movement while face tracking handles another action.

This guide explains the app's current screens. It does not teach you how to build or wire a controller. The screenshots are real app screens captured on Android with example profiles. Board readings are simulated for the pictures. The pictures do not show a successful connection to a real console. Your phone's permission windows and keyboard may look different.

Guide for app 1.0.0 · Updated 2 October 2026

This PDF includes all screenshots. Use the clickable contents or PDF bookmarks to find a section. Zoom in to read small details in a picture.

Contents

1. [Understand inputs, outputs, and Intents](#)
2. [Your first setup](#)
3. [Find your way around the home screen](#)
4. [Connect an ESP32 or Pico W](#)
5. [Use several boards and other inputs together](#)
6. [Configure every input type](#)
7. [Configure every output type](#)
8. [Use the Intent Engine](#)
9. [Worked Intent examples](#)
10. [Check your Control Map](#)
11. [Save, copy, and share profiles](#)
12. [Start again next time](#)
13. [Fix common problems](#)
14. [Button names and useful words](#)

Understand inputs, outputs, and Intents

Inputs are the things you use

An **input** is something you press, move, or do. A switch, key, stick, and smile are all possible inputs.

A profile can contain **up to eight input devices**. These can be a mixture of types. Each device can have several mappings.

Input in the app	What it reads	A reason to use it
Controller	A game controller connected to Android	Change button positions or combine actions
Keyboard	Keys from a connected keyboard	Use large keys or a familiar layout
Mouse	Mouse buttons, movement, and wheel input	Use clicks for actions or movement for a stick
Joystick	Buttons and axes from an Android joystick device	Use a flight stick, adapted stick, or similar device
Face Tracking	Expressions and head movement through the camera	Play without pressing a physical button
ESP32 Link	Switches and sensors reported by an ESP32	Use a custom set of physical controls
Pico W Link	Switches and sensors reported by a Pico W	Use a custom set of physical controls
Virtual Input	The app's on-screen controller	Use touch controls, with positions and sizes you can change

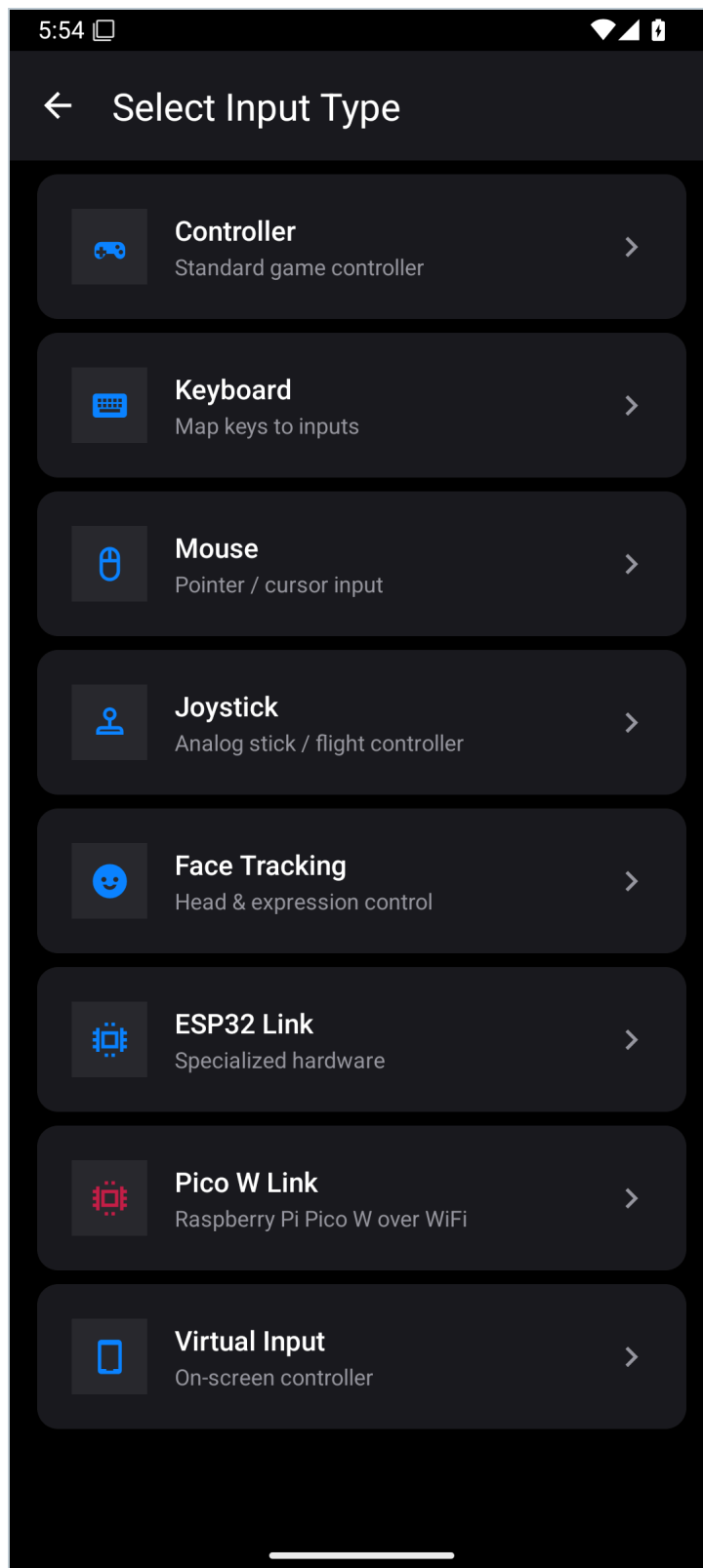


Figure 1. Tap **+ Input** on the home screen to reach these choices.

Outputs are where the actions go

An **output** is the destination for the controller actions. Each profile uses **one output at a time**.

Output in the app	Where the actions go	What you need to set up
This Device	The phone broadcasts a Bluetooth gamepad	Bluetooth permissions and pairing with the receiving device
ESP32	An ESP32 acts as a controller bridge	Phone-to-board connection, controller identity, and board-to-host connection
Pico W	A Pico W acts as a controller bridge	Phone-to-board connection, controller identity, and board-to-host connection
PS Remote Play	The app's Remote Play connection to a PlayStation	Console details, account ID, and console pairing
Xbox Remote Play	The app's Remote Play connection to an Xbox	Microsoft sign-in and selection of your console

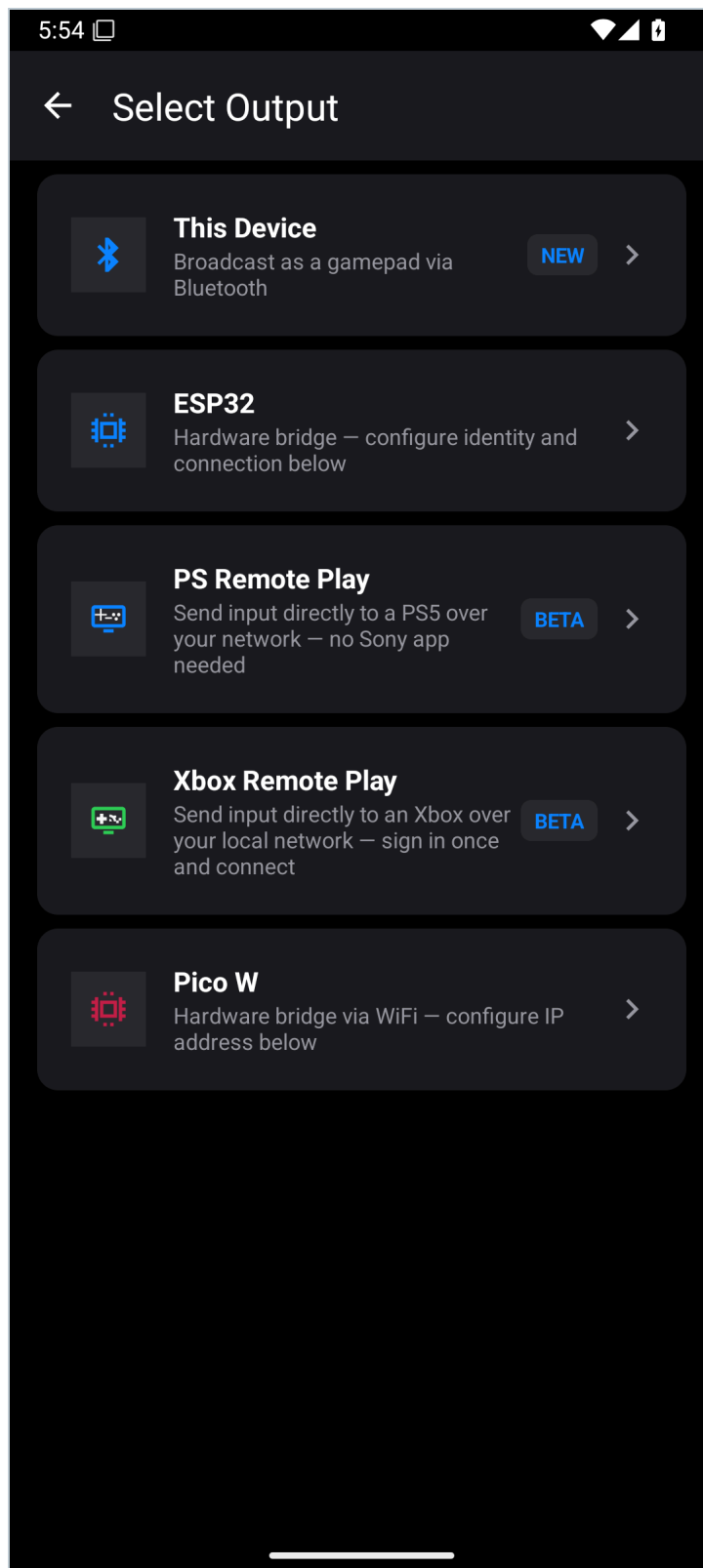


Figure 2. Choose the destination that matches your equipment. The Remote Play outputs are marked **BETA**.

You can use your configured inputs with any of these supported output choices. You do not need a different aim-and-shoot mapping just because you change from a board output to Remote Play.

However, each destination still needs its own connection setup. It must support the controller actions you send. For example, a Touchpad action needs a destination that understands it. Motion data currently goes to PS Remote Play. Choosing an output does not make every computer, console, or

game compatible with it.

An Intent is your instruction

A **mapping** joins an input to an action. An **Intent** can make that instruction more useful.

Here are three levels of setup:

Level	Example
Connect the input	Connect the board that holds your foot switch
Choose its source and target	Read the switch port and send R2
Edit its Intent	Send L2 and R2 together while the switch is held

The app sends controller controls such as **South**, **L2**, and **Right Stick**. It does not know what your game calls those controls. You must check the game's own control settings. **R2** might mean fire in one game and accelerate in another.

The journey is:

Connect inputs → configure their mappings and Intents → choose and connect an output → Start Mapping.

You may choose the output first. The important point is that both sides must be ready before you play.

Your first setup

What to have ready

Have your phone, a supported output destination, and at least one input ready.

For a USB input or board, use a data cable and an Android USB host adapter, often called an **OTG adapter**. A charging-only cable cannot carry input data. A USB hub may be needed for several devices.

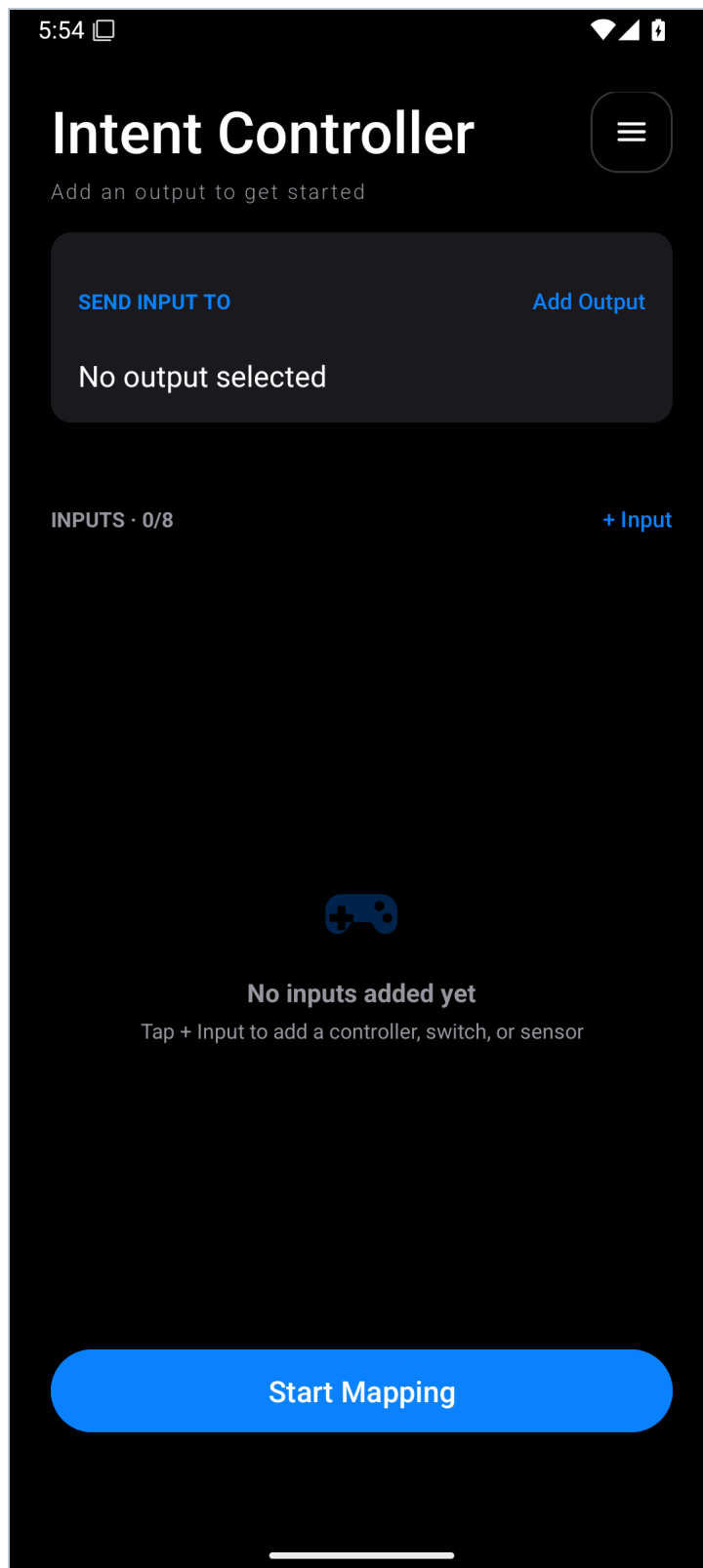
For a Wi-Fi board, have the network name and password ready. The board needs a supported **2.4 GHz** network. The phone and board must be able to reach each other on that network.

For face tracking, place the phone where the camera can see your face clearly. For Remote Play, have the console switched on for the first setup.

A simple first run

1. Open **Intent Controller**.
2. Tap **Add Output** in the **SEND INPUT TO** area. If an output is already selected, the button says **Change Output**.
3. Choose one output and complete its setup. Use the matching [output instructions](#) below.
4. Return to the home screen.
5. Tap **+ Input** beside **INPUTS**.
6. Choose your input type.
7. Connect or select that input. Then check its mappings. Some input types create starter mappings for you.
8. Set one easy test action. For example, map a switch or key to **South** with **Hold** behaviour.
9. Tap **Done** in the input editor.
10. Return to the home screen. Read the status below the app title.
11. Let go of buttons and return sticks to the centre.
12. Tap **Start Mapping**. Complete any Bluetooth pairing or permission request that appears.
13. Test the action in a game menu or practice area.
14. Tap **Stop** when you finish.

Do this first with one input and one action. Add more controls after that test works. This makes it easier to find a mistake.



First-use view. Choose **Add Output** and **+ Input** before starting.

If **Start Mapping** is unavailable, complete the missing step shown by the app. A plugged-in input still needs a mapping. A board output also needs a verified connection and an applied controller identity.

Find your way around the home screen

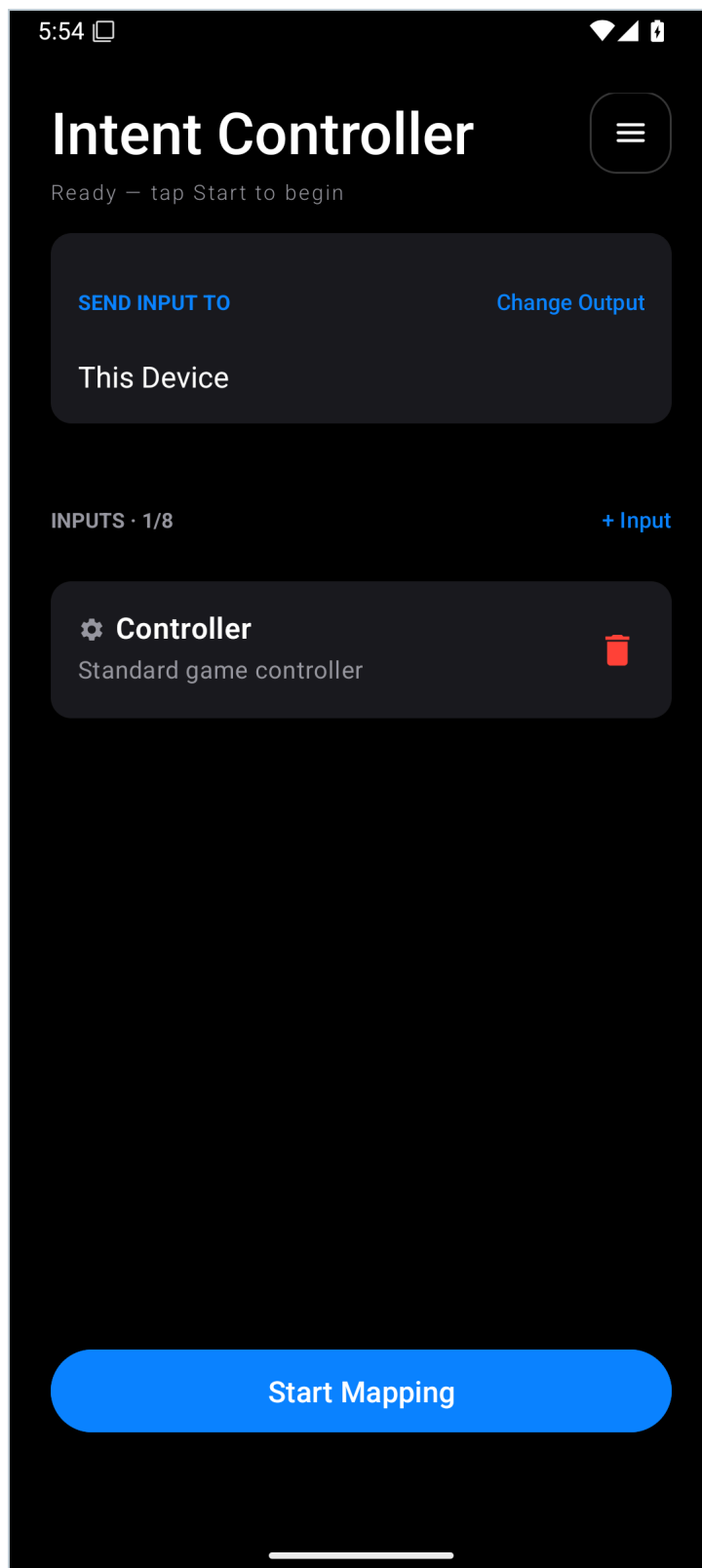


Figure 3. This example already has an output and one input. A new empty profile has **Add Output** and no input cards.

Button or area	What it does
Status below Intent Controller	Tells you whether setup is ready or a connection needs attention
Add Output / Change Output	Chooses or changes the destination
+ Input	Adds another input device, up to the eight-device limit
Input card or gear	Opens that input's mappings
Connection on a board card	Opens connection setup for that particular board
Continue setup	Resumes an unfinished board setup
Red bin	Removes the selected input from the profile
Reorder	Changes the order of input devices
Start Mapping	Begins sending your configured actions
Stop	Ends mapping and releases the output controls
Three-line menu at the top right	Opens profiles, Control Map, screen dimming, safety settings, and licences

Remove only the card you mean to remove. With several boards, check the name first.

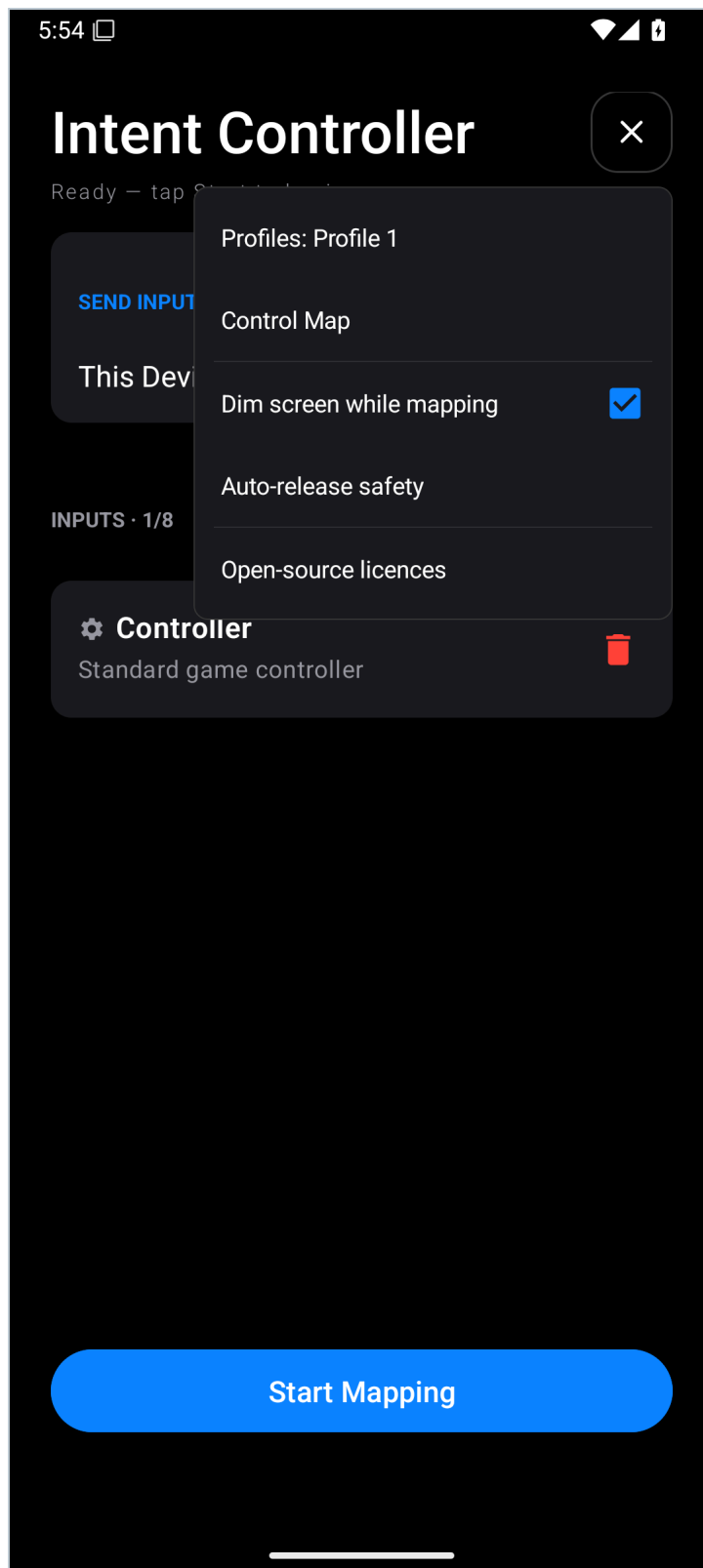


Figure 4. The home menu. The name after **Profiles:** is the current profile.

Dim screen while mapping lowers screen brightness during mapping. This can be useful when you are watching a television instead of the phone. Change it from this menu.

The input editor's live feedback helps you check a source. It does not replace **Start Mapping**. Saving a mapping also does not start a game session.

Connect an ESP32 or Pico W

Both board types use the same three-part connection journey:

1. **Connect** to the board.
2. Set up its **network if needed**.
3. **Verify and name** the board.

You use this journey for board inputs and board outputs. The final save button changes to match the job.

An input board reads your controls. An output board sends controller actions to the host. These are separate roles, even if you choose to use one physical board for both.

Open the board setup

For an input, tap **+ Input** → **ESP32 Link** or **Pico W Link**.

For an output, tap **Add Output / Change Output** → **ESP32** or **Pico W**.

For a board already saved as an input, tap **Connection** on its own home-screen card. From its **Input Hub**, you can also use the three-dot menu and **Connection**.

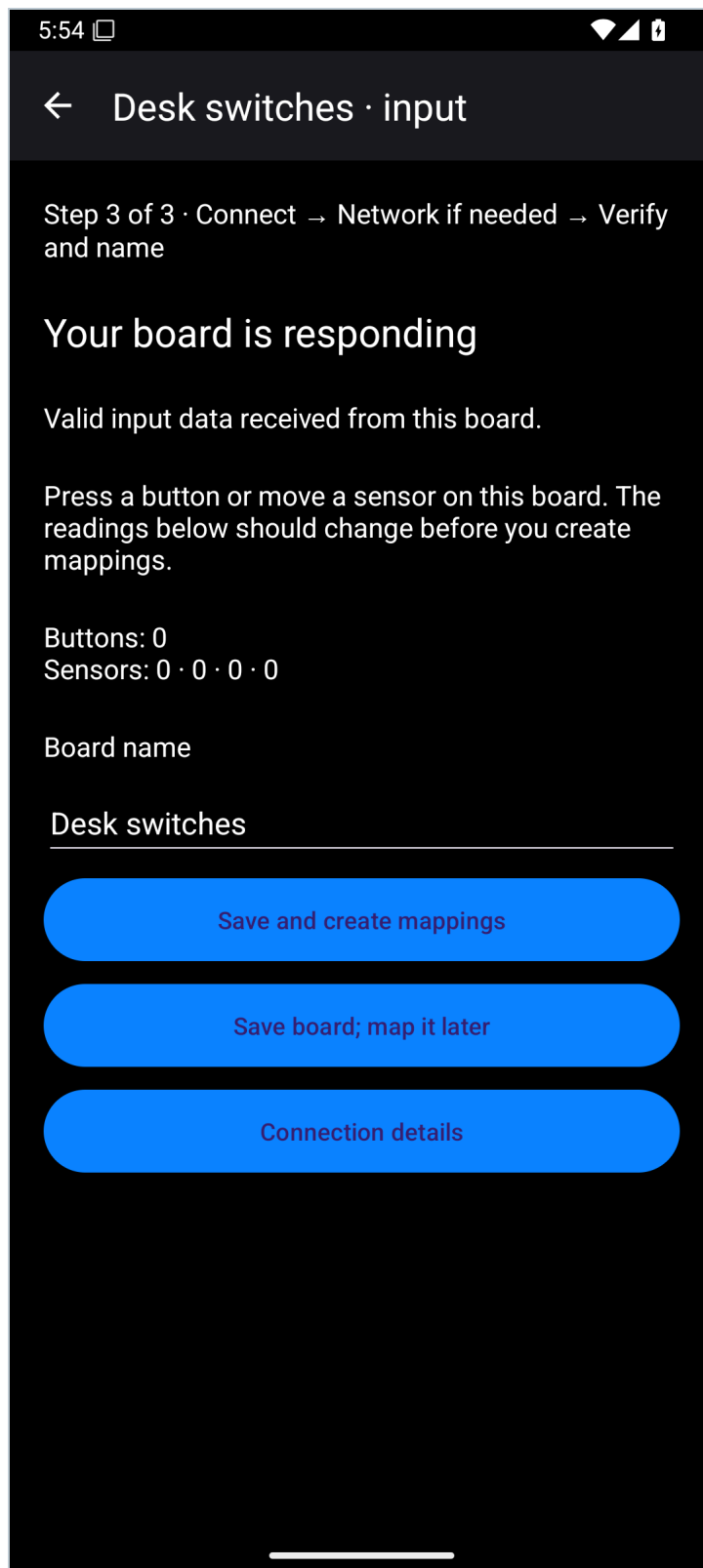


Figure 5. Choose how this phone will talk to this board.

Connect a board that is already on Wi-Fi

1. Power the board.
2. Connect the phone to a network that can reach the board.
3. Tap **Wi-Fi** in board setup.

4. Tap **Find boards on this network**.
5. Choose the correct board from the results. Check its address and identity, especially if several boards appear.
6. Wait for the app to receive valid board data.
7. If the board is already on the right network, continue to the verification and naming screen.

If a saved address is available, **Reconnect saved board** lets you try it again.

If discovery does not find a known board, tap **Enter board address**. Enter that board's IP address and port. Use the values for your board, not the example address shown in this guide. An IP address is the board's address on the network; it can change when the router restarts.

Connection details shows information you can use to check which board you selected. A network connection alone is not the final check. The app also waits for valid input data.

Put a new board on your home Wi-Fi

Use this when the board has the app's supported firmware but has not joined your home network.

1. Tap **Wi-Fi** → **Set up a new board**.
2. Read the setup message.
3. If several new boards are nearby, power only the one you are setting up. Their initial Wi-Fi network may have the same name.
4. Tap **Open Wi-Fi settings**.
5. Join the board's **Intent Controller** Wi-Fi network.
6. If Android says this network has no internet, stay connected to it.
7. Return to Intent Controller.
8. Tap **Connect to setup board**.
9. When the network form opens, enter your home Wi-Fi details.

5:54

← Desk switches · input

Step 2 of 3 · Connect → Network if needed → Verify and name

Join your Wi-Fi

Valid input data received from this board.

Choose a 2.4 GHz network. Its name and password are sent to this board only. The app does not save the password.

Network name

Example Home Wi-Fi

Password

Password

☐ This network has no password

☐ Show password

Choose nearby network

Send and reconnect

Back to connection

Connection details

Figure 6. Enter the network the board should join. The example name is not a real network you need to use.

10. In **Network name**, enter the name exactly, including upper- and lower-case letters.
11. You can tap **Choose nearby network** to select a name. Android may ask for location access to scan networks. You can type the name instead.
12. Enter the **Password**. Tap **Show password** if you need to check it.

13. Only tick **This network has no password** if the network is actually open.
14. Tap **Send and reconnect**.
15. Let the board restart. A brief disconnection is expected.

The current form supports network names from 1 to 32 printable ASCII characters, without commas. Passwords must be 8 to 63 printable ASCII characters, or blank for an open network. Letters, numbers, spaces, and common punctuation are ASCII. Names containing characters such as emoji are not supported by this setup form. Passwords can contain commas.

The app does not keep the Wi-Fi password as a saved profile setting. You normally enter it once to set up the board. If you close setup and later need to change the network, you may need to enter it again.

Reconnect after the board restarts

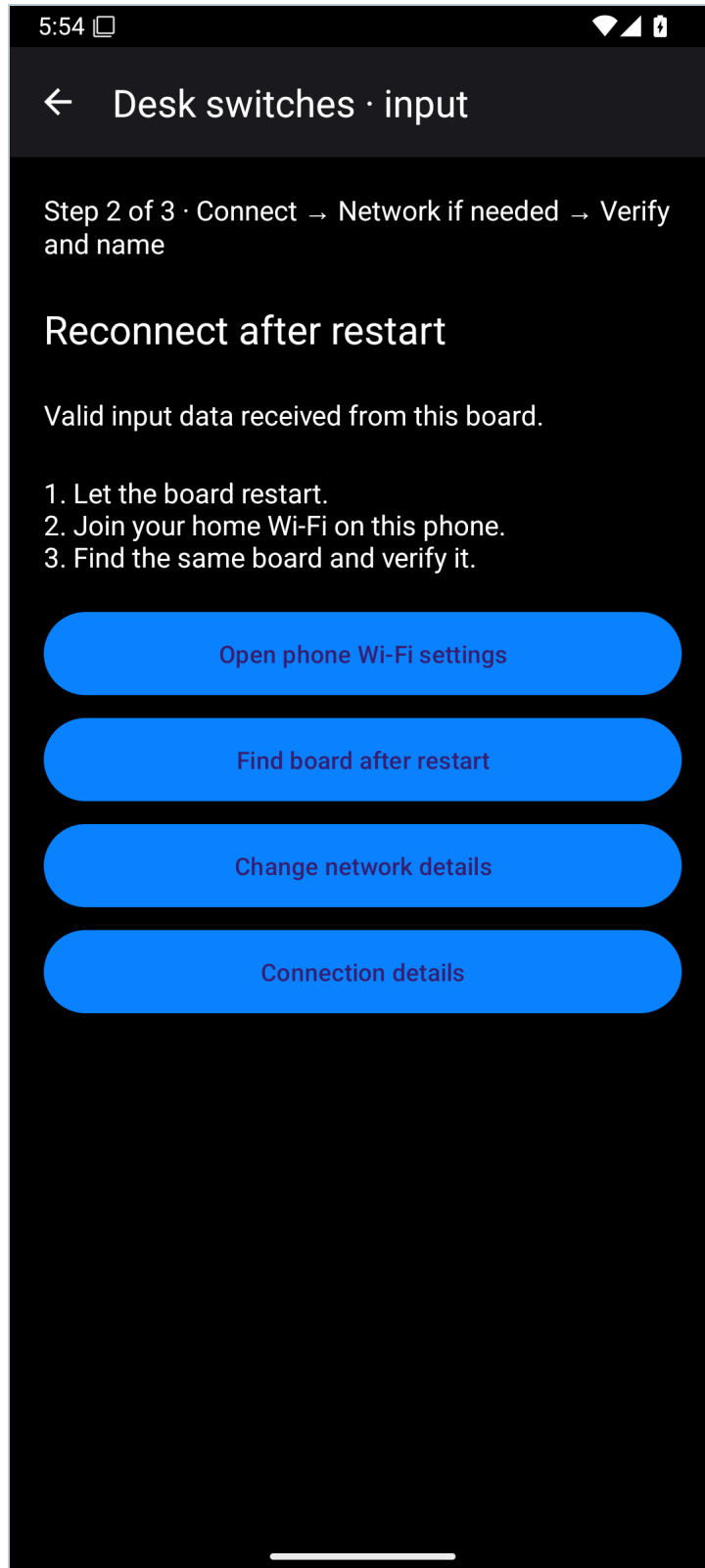


Figure 7. This screen helps you move from the board's setup network to your home network.

1. Tap **Open phone Wi-Fi settings**.
2. Join your home Wi-Fi on the phone.
3. Return to the app.

4. Tap **Find board after restart**.
5. Choose or reconnect to the same board.
6. Wait for valid board data.

If it does not reappear, check the network name and password. Use **Change network details** to correct them. Do not add another input card just because the board is restarting.

Connect over USB

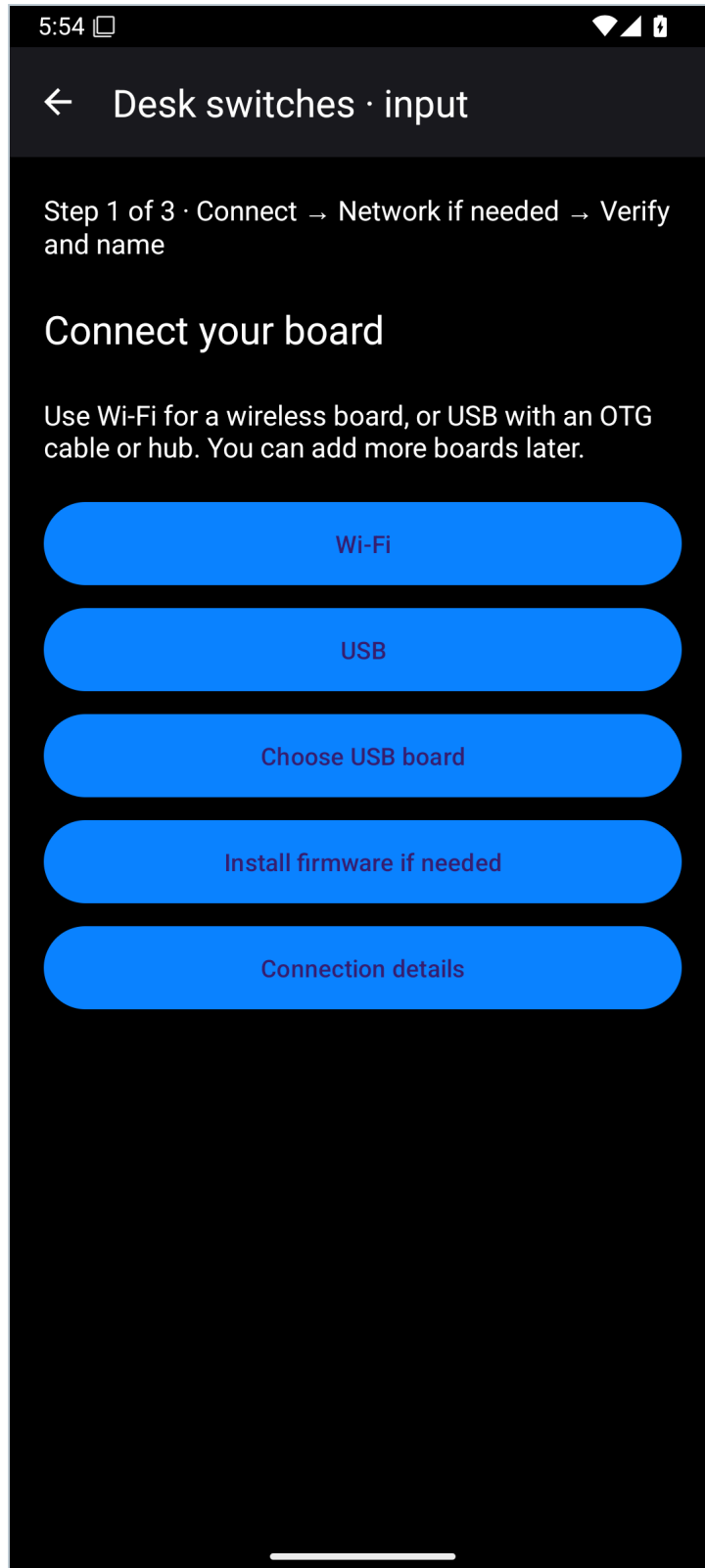


Figure 8. USB setup chooses a particular device attached to the phone.

1. Connect the board to the phone with a data cable and OTG adapter or hub.
2. Tap **USB**.
3. Tap **Choose USB board**.

4. Select the correct device. If several are attached, check the device details.
5. Choose its serial port if asked. Many boards have one port, numbered **0**.
6. Accept Android's USB access request for the selected board.
7. Wait for valid board data.
8. Continue to verification and naming.

A board appearing in the USB list only means Android can see it. It must also speak the supported board protocol before setup can be verified.

If the board needs firmware

The app offers **Install firmware if needed**. Use this when the board has no suitable firmware or the app cannot read valid data from it. Choose the board and follow the on-screen instructions. Installing firmware changes the software on that board.

For a Pico W installation, the app may ask you to connect it in **BOOTSEL** mode. Hold the board's BOOTSEL button while connecting USB, then follow the app's storage-device selection and installation steps.

For ESP32, follow the app's USB and boot-button instructions for the supported board. A product labelled ESP32 is not automatically compatible with every bundled firmware image.

After installation, return to the connection journey and verify the board. You do not need to install firmware every time you connect. Detailed wiring and firmware building are outside this app guide.

Verify and name the board

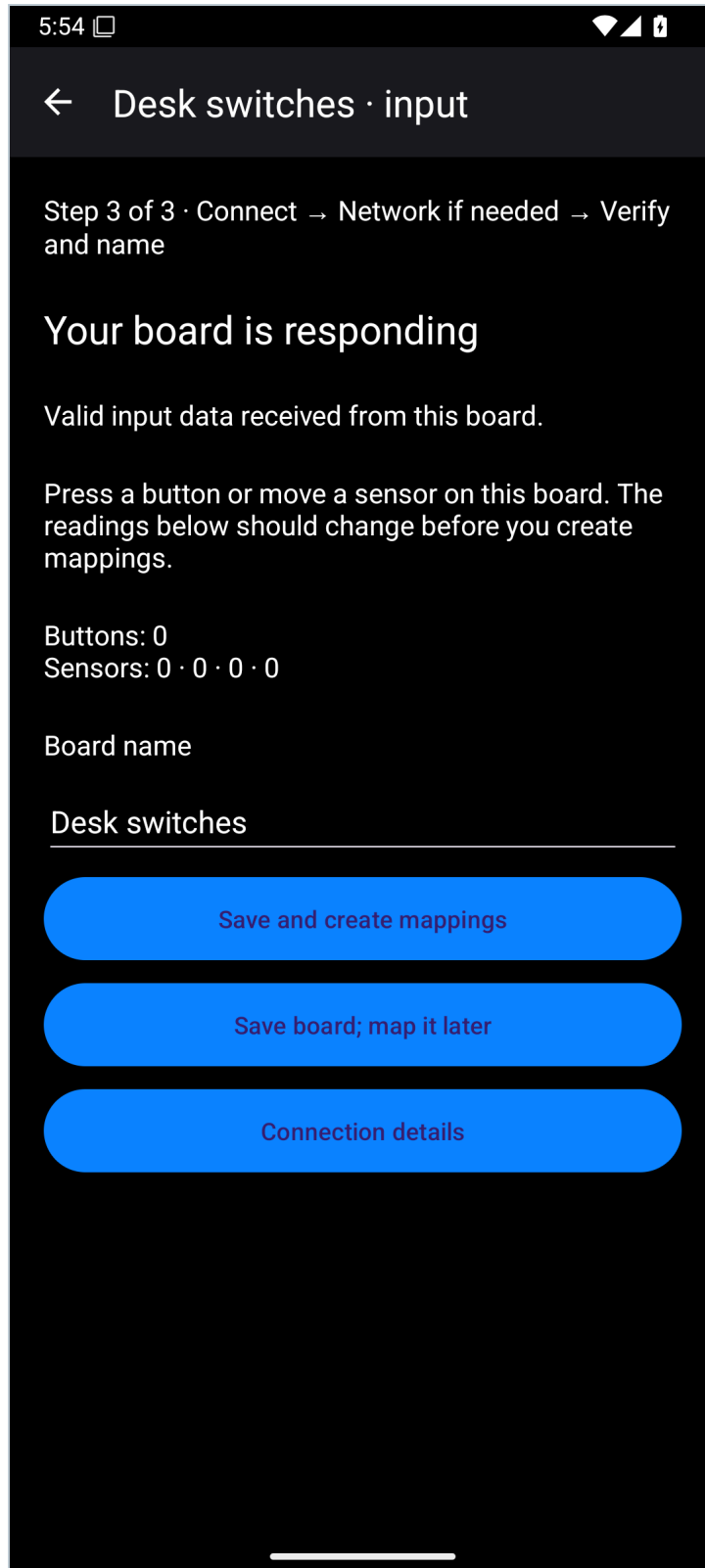


Figure 9. Verification, a useful name, and the next step.

1. Wait for **Your board is responding**.
2. Press a physical switch or move a sensor attached to that board.
3. Check that its readings change. This also helps confirm you selected the right physical board.

4. Enter a useful **Board name**. Try **Desk switches**, **Chair controls**, or **Left-hand stick**.
5. For an input, tap **Save and create mappings** to configure controls now.
6. Alternatively, tap **Save board; map it later**. The connection is saved, but that input still needs mappings before it can help you play.
7. For an output, tap **Save and choose controller**. Continue with the output instructions below.

When editing an existing connection, the button may say **Save connection**. This returns you to the screen you came from.

If you verified through USB and want to switch to Wi-Fi, use **Set up Wi-Fi instead** when offered.

Resume an unfinished setup

If you leave input-board setup before finishing, the home screen can keep a draft card. Tap **Continue setup** to resume it. This avoids creating duplicates.

Leaving a new output setup does not turn an unverified draft into your working destination. Finish connection setup and controller selection before expecting it to work.

Use several boards and other inputs together

You can add several ESP32 boards, several Pico W boards, or a mixture. They share the **eight-input limit** with the other input types.

For example, this is a three-input profile:

Input	Job
Desk switches, ESP32	Aim, fire, and menu buttons
Chair controls, Pico W	Movement sensors
Controller	Extra buttons or a helper's controls

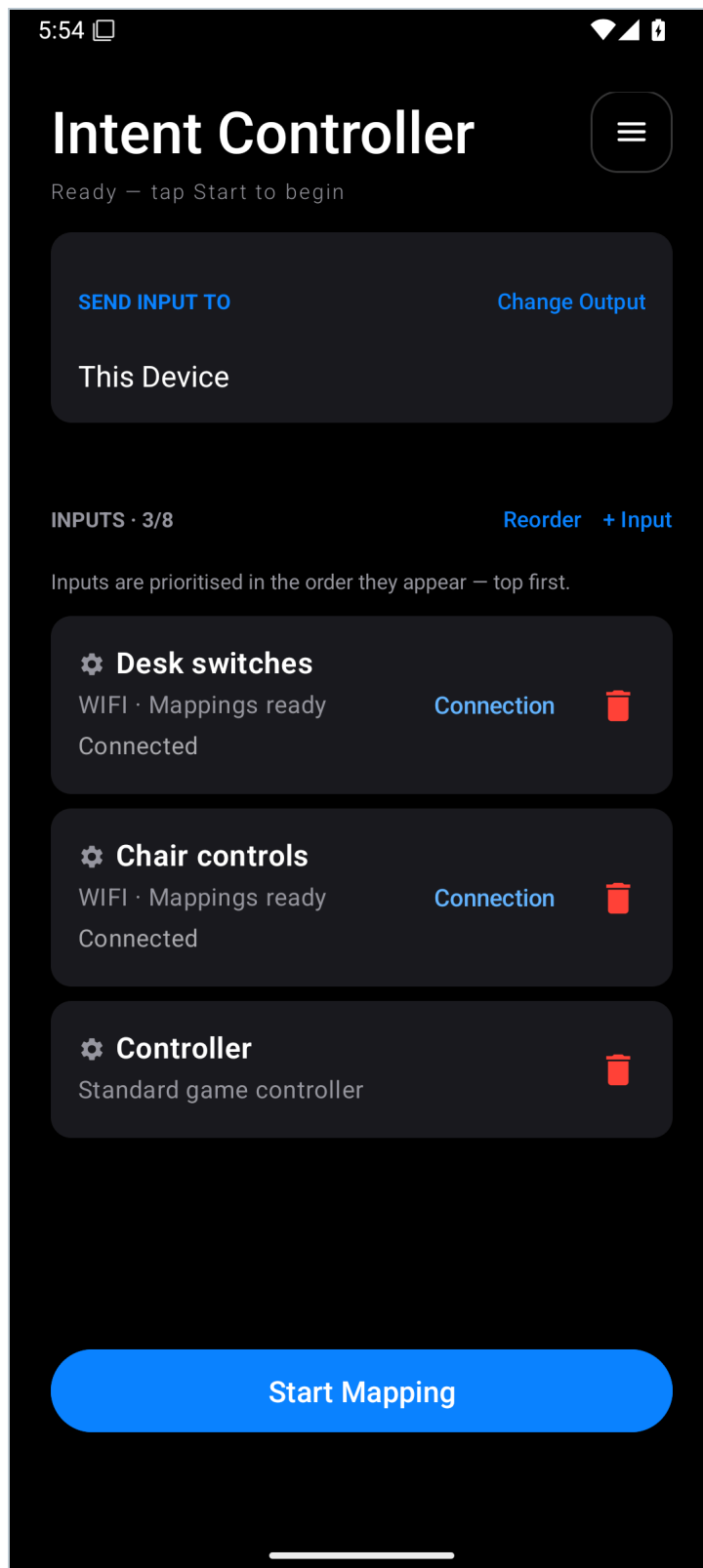


Figure 10. Each board has a separate name, connection, and set of mappings.

Add the next board

1. Finish the first board's setup and give it a clear name.
2. Tap **+ Input** again.
3. Choose **ESP32 Link** or **Pico W Link** for the next board.

4. Select that physical board explicitly in discovery or the USB list.
5. Verify it by moving one of its controls.
6. Give it a different name.
7. Create its mappings.
8. Repeat for any other inputs.

Do not overwrite the first board's connection to add a second one. **Connection** changes the board on an existing card. **+ Input** creates another input.

If one board goes offline, its own card shows the problem. Use its **Connection** button to repair that link. You do not need to recreate all the other inputs.

When you press **Start Mapping** with some mapped inputs offline, the app can show **Some inputs are offline**. Use **Retry connections** to try them again. If other mapped inputs are available, **Start with available inputs** lets you make that choice explicitly. Controls from the missing inputs will be unavailable.

Change which input has priority

1. Tap **Reorder** above the input cards.
2. Use ↑ and ↓ to move the cards.
3. Put the input you want to have first choice at the top.
4. Tap **Done**.

Order matters when two mappings try to control the same output. It does not affect mappings that control separate buttons or axes. [Priority and handover](#) explains this in detail.

An output board does not use an input slot unless you also add it as an input. If one board has both roles, keep the selected physical board consistent. Do not assume the app can recognise the same board automatically across different USB and Wi-Fi identities.

Configure every input type

Controls shared by the mapping editors

The editors use cards for mappings. A card can be folded to keep the screen short.

1. Tap a card or its down arrow to open it.
2. Choose the source: the key, button, port, gesture, or axes you will use.
3. Choose the action or output target.
4. Use the live feedback to check the source.
5. Tap **Edit Intent** to change timing, add more outputs, or create a sequence.
6. Tap **Save & Close** in the Intent editor.
7. Tap **Done** in the input editor when finished.

Use **+ Add Intent** to add another mapping. The red bin deletes that mapping. It does not remove the whole input device.

If **Reorder** is offered inside an editor, it changes the order of that input's mappings. Use the arrows and finish with **Done**. Earlier mappings get priority when there is a conflict.

Some setup routes offer **Use defaults**, **Create your own**, or a saved-profile choice. Defaults are a starting point. Review the source and target before relying on them.

Controller input

Use **Controller** for a normal gamepad that Android recognises.

1. Connect the controller by USB, or pair it in Android's Bluetooth settings.
2. On the app home screen, tap **+ Input** → **Controller**.
3. Choose the controller in **Select Controller**.
4. The app creates starter gamepad mappings. Return to the home screen and tap the controller card to edit them.
5. On **Controller Mapping**, press a button and watch the detection or feedback area.
6. Open the mapping card for the control you want to change.
7. Change its input button and **Map to Action**, or its axis and stick target.
8. Use **Edit Intent** for combinations and timing.
9. Tap **Done**.

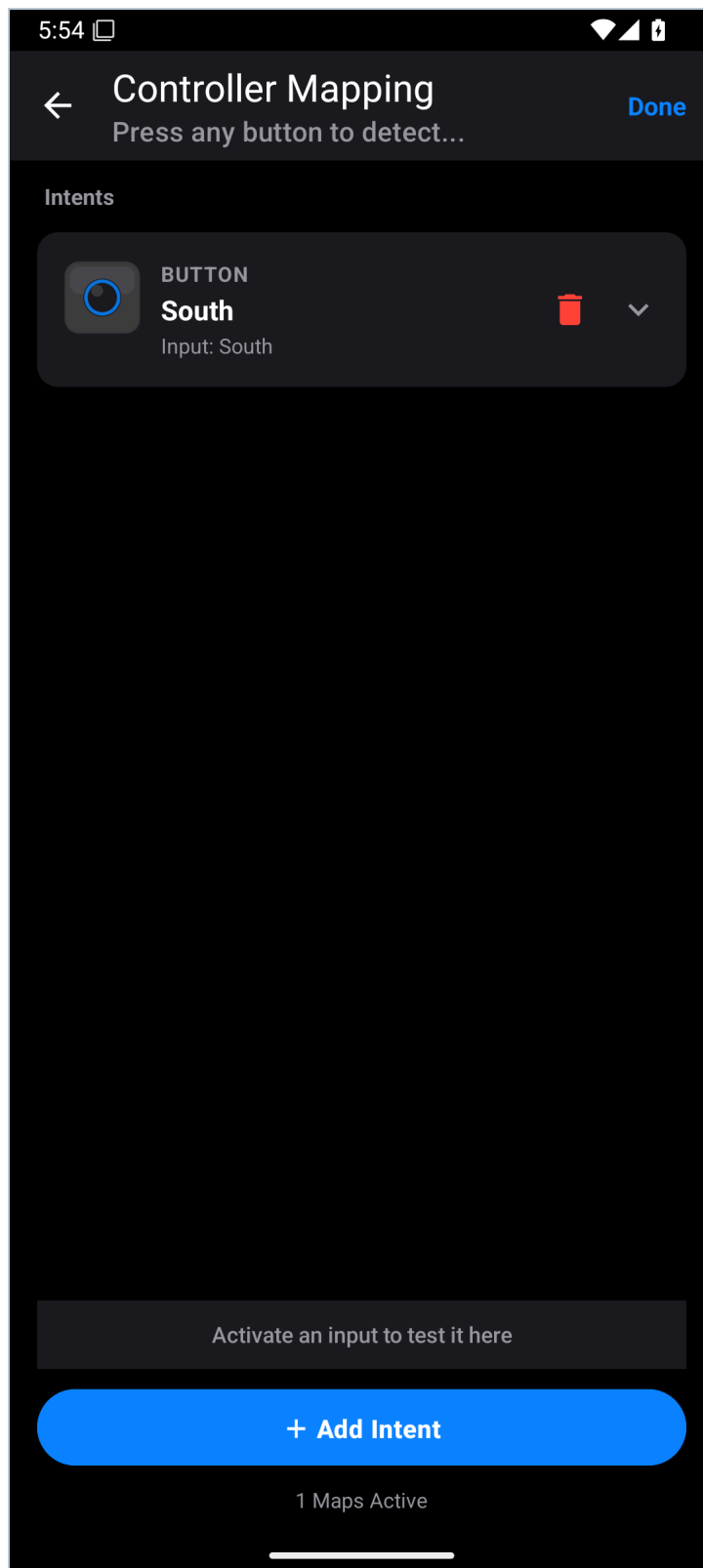


Figure 11. A gamepad button mapping. Tap its card to open the settings.

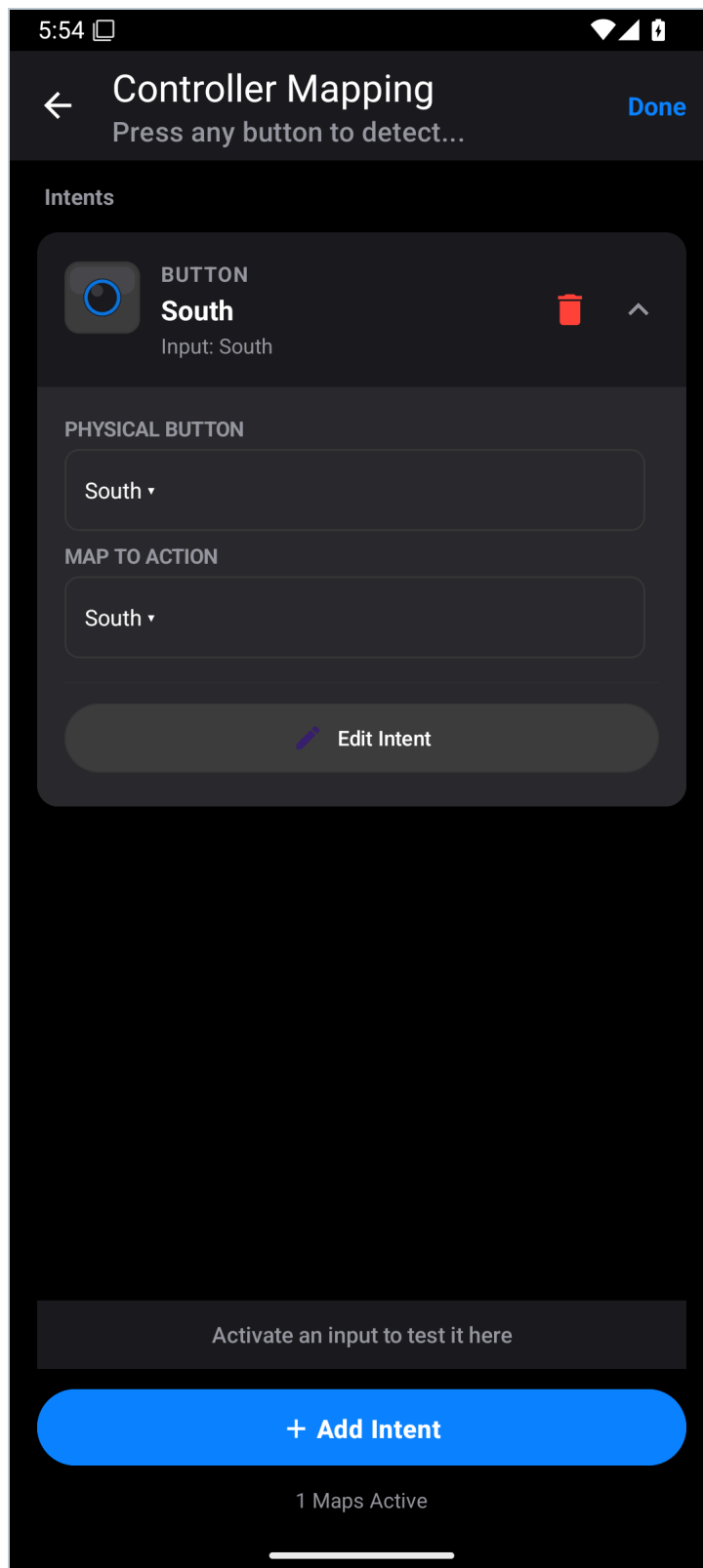


Figure 12. Edit the input and action here; use **Edit Intent** for more than a simple mapping.

Switch is useful for buttons. **Axis** is useful for a trigger. **Dual Axis** joins X and Y movement to one target stick.

If the selector says no controller was found, connect it to Android first. Saving a controller identity for a board output does not add that controller as an input.

Keyboard input

Use **Keyboard** for an external USB or Bluetooth keyboard. The phone's typing keyboard is not the same input device.

1. Connect or pair the keyboard with Android.
2. Tap **+ Input** → **Keyboard**.
3. Choose it in **Select Keyboard**.
4. On **Keyboard Mapping**, press a key and check the feedback.
5. Open a mapping card.
6. Choose the source key. If **Detect** is shown, tap it and press the key you want.
7. Set **Map to Action**. For example, choose **SPACE** → **South**.
8. Add other keys with **+ Add Intent**.
9. Tap **Done**.

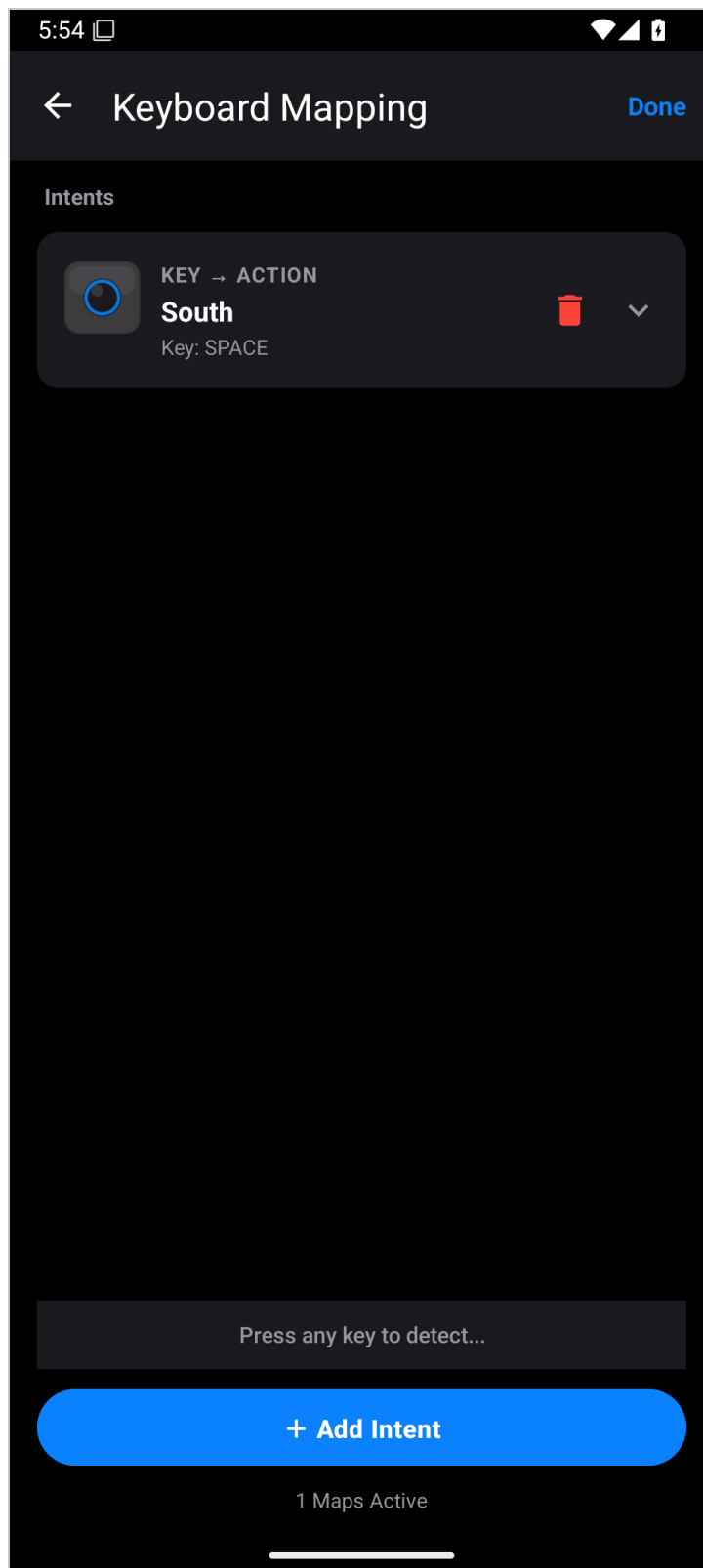


Figure 13. This example uses SPACE for the South button.

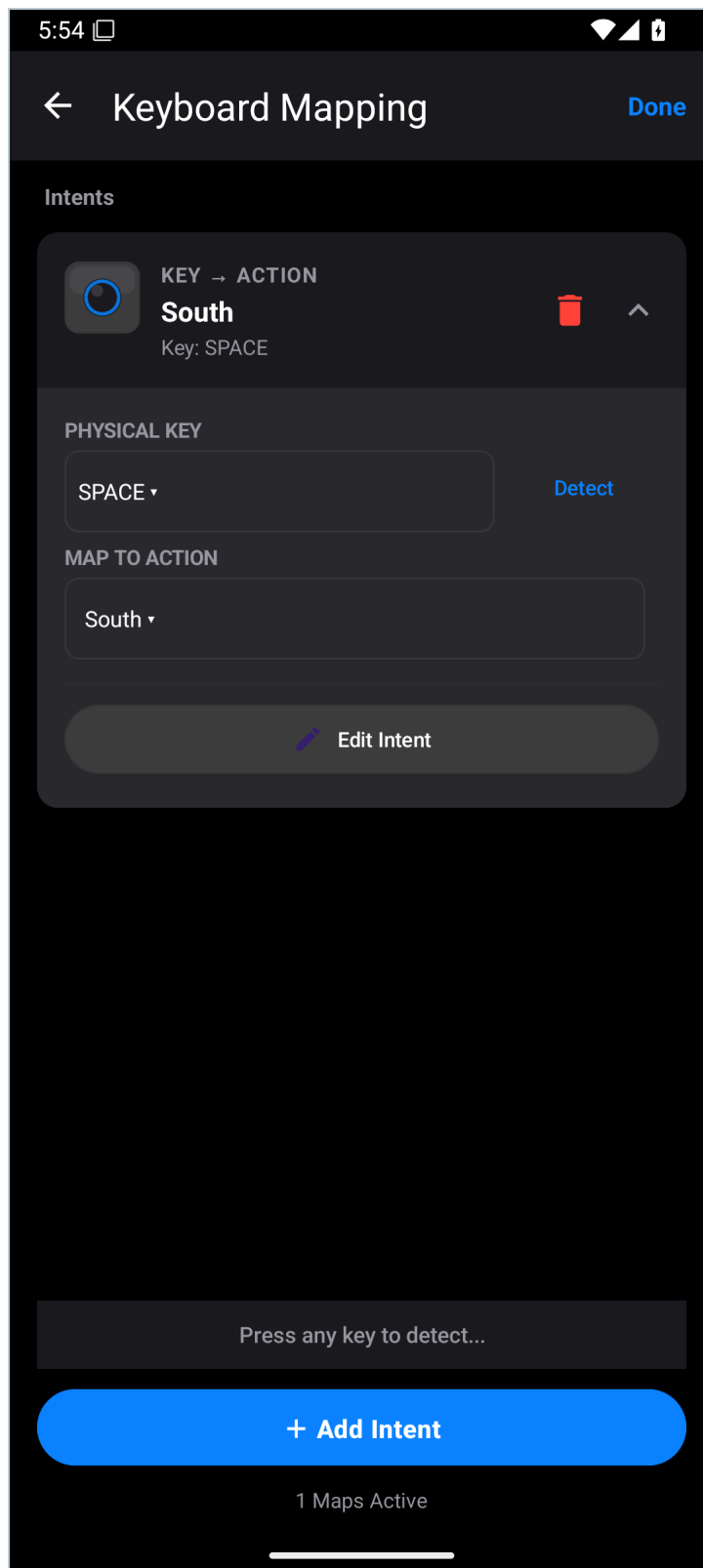


Figure 14. Choose the key and the controller action separately.

Some starter layouts use **W, A, S, D** for the D-pad. D-pad movement and stick movement are different controls. If your game needs the left stick, create a **Dual Axis** mapping and assign its directions to the keys you want.

Keyboard input becomes controller output. Mapping SPACE to South sends a controller button, not typed text to the console.

Mouse input

1. Connect the mouse to Android by USB or Bluetooth.
2. Tap **+ Input** → **Mouse**.
3. Choose it in **Select Mouse**.
4. Click a button or move the mouse to check detection.
5. Open a card, or tap **+ Add Intent**.
6. For a button mapping, choose **Left Button**, **Right Button**, **Middle Button**, **Back Button**, or **Forward Button**, as supported by your mouse.
7. Choose the gamepad action. For example, **Left Button** → **R2**.
8. For movement, choose the available X/Y movement axes and the output axis or target stick. Wheel inputs include **Scroll Up** and **Scroll Down**.
9. Use **Edit Intent** if a click should send several actions.
10. Tap **Done**.

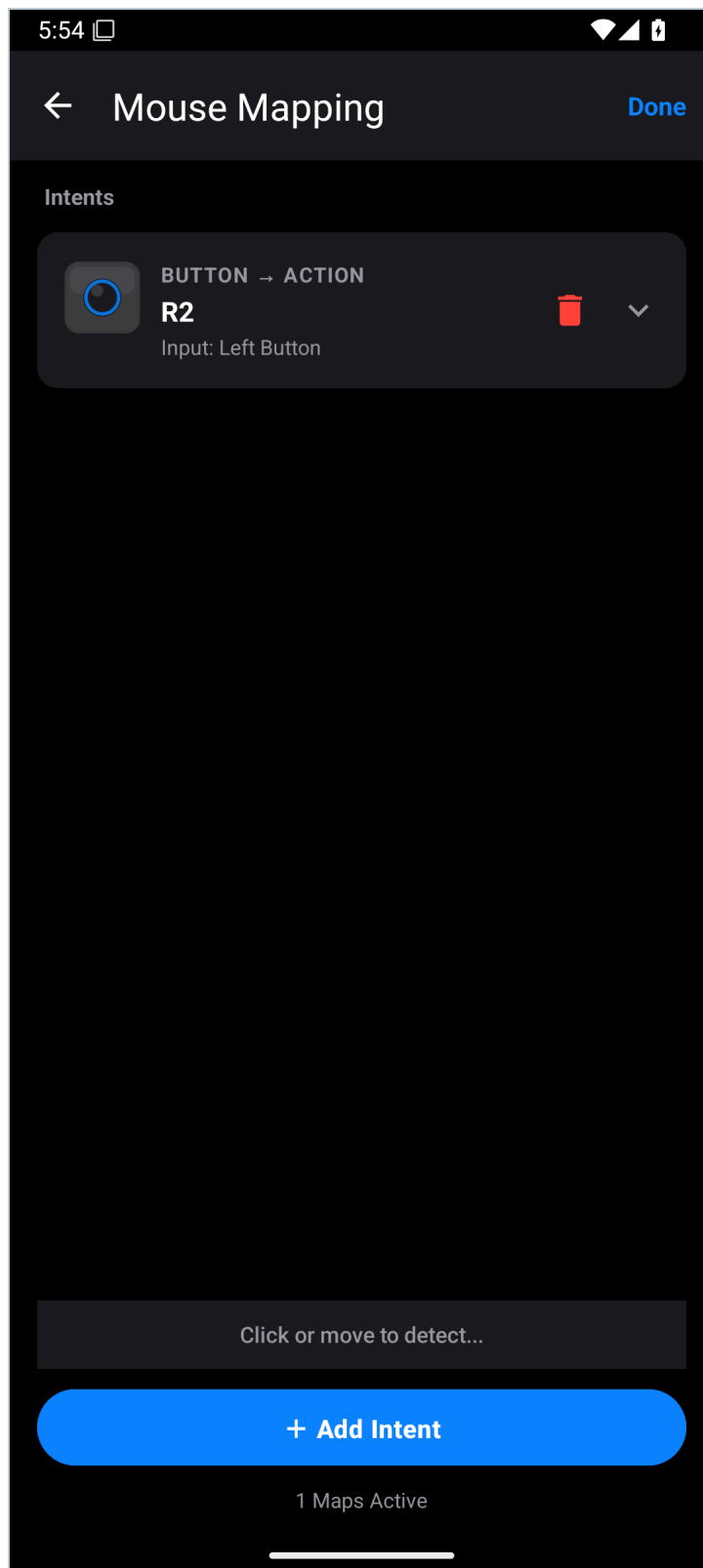


Figure 15. A mouse click can become a controller trigger.

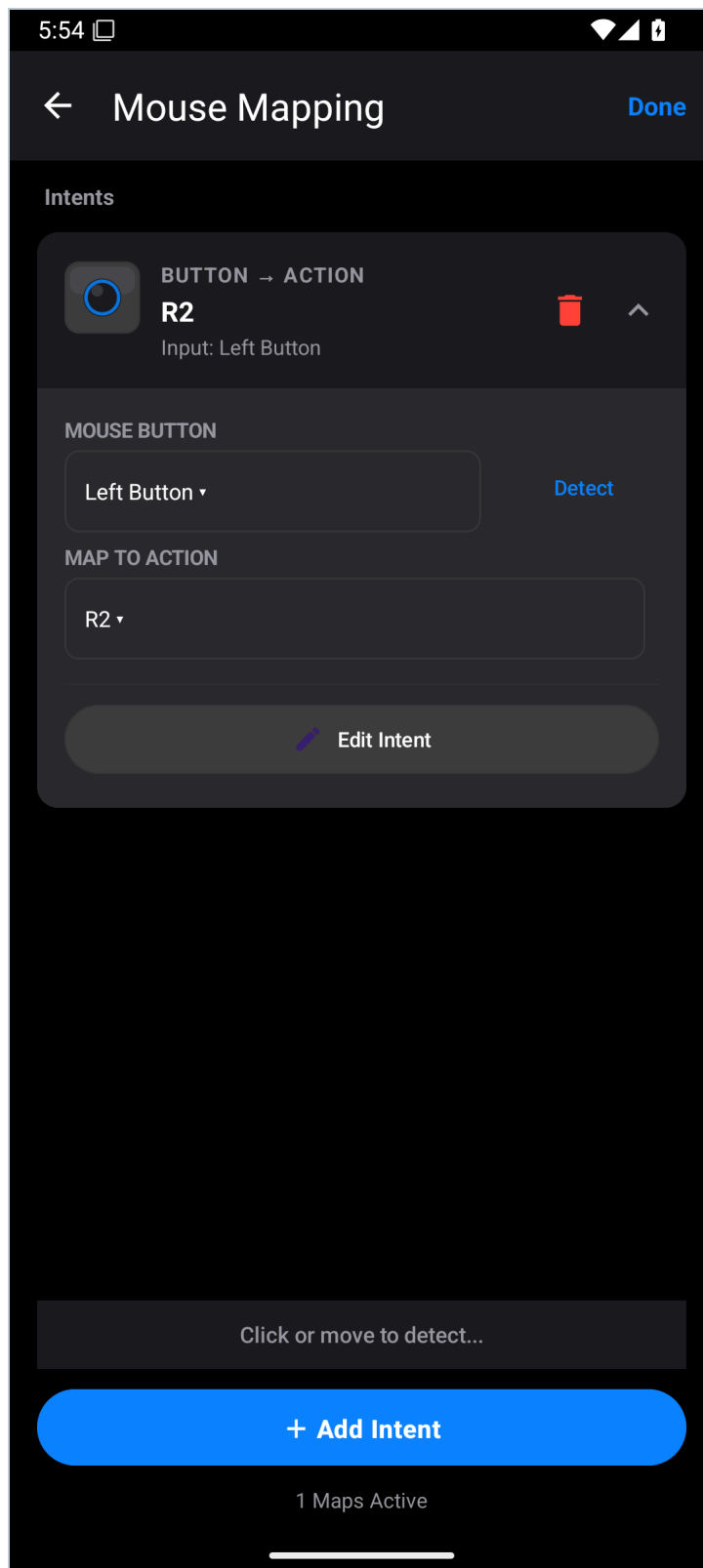


Figure 16. Choose the mouse source before choosing its controller output.

Mouse movement mapped to a stick behaves as controller movement. It is not a promise of the same pointing behaviour as a desktop mouse. Check the game's stick sensitivity as well as your mapping. Start with small movements in a practice area.

Joystick input

Use **Joystick** for a device that Android reports as a joystick, such as a flight stick or some adapted controls. Use **Controller** for a normal gamepad unless you have a reason to use the joystick editor.

1. Connect or pair the joystick with Android.
2. Tap **+ Input** → **Joystick**.
3. Select the device.
4. Move a control or press a button to check detection.
5. Tap **+ Add Intent** and choose the mapping shape.
6. Use **Switch** for a button, **Axis** for a single control such as a throttle, or **Dual Axis** for X and Y stick movement.
7. Choose the input button or axes. Use **Detect** where offered.
8. Choose the output action or target stick.
9. Check that pushing up produces the direction you expect. Correct the source choice or inversion if that control offers it.
10. Tap **Done**.

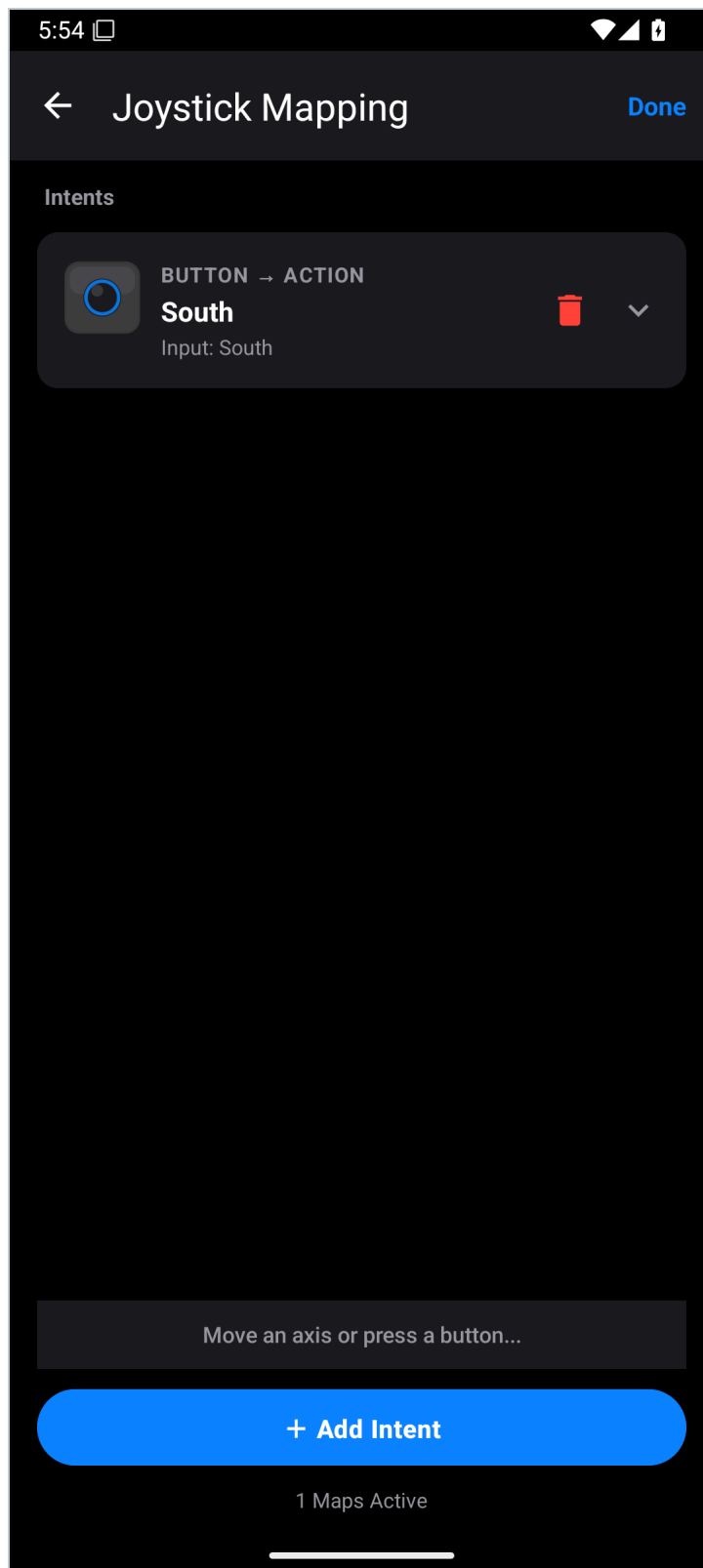


Figure 17. Joystick mappings use the same card pattern as other physical inputs.

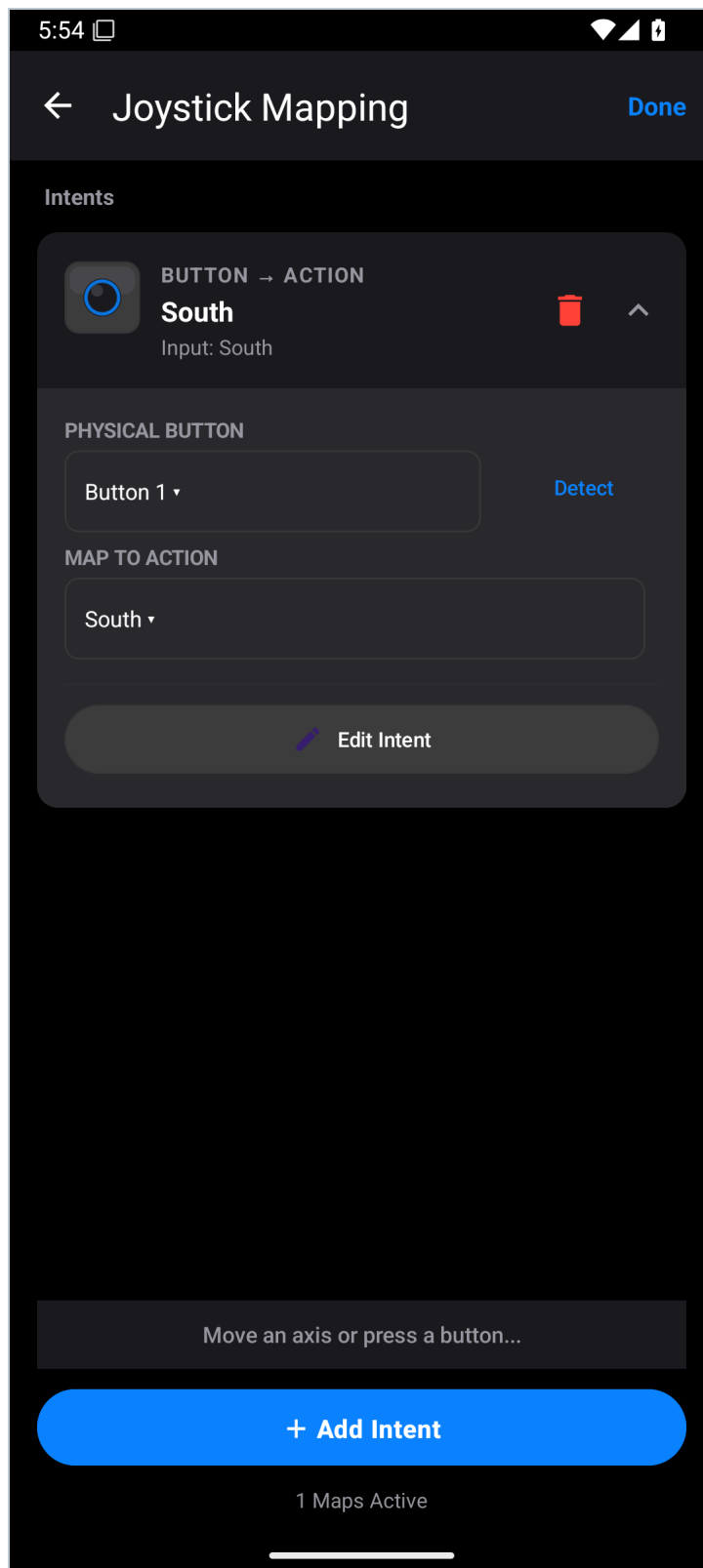


Figure 18. Check the source control and its destination before starting.

The controls offered depend on what Android reports for your device. The app cannot read an axis that the device does not provide.

ESP32 Link input

1. Tap **+ Input** → **ESP32 Link**.
2. Complete [board connection setup](#).
3. On the verification screen, tap **Save and create mappings**.
4. The **Input Hub** opens for that board.
5. Tap **+ Add Intent**.
6. Choose **Switch**, **Axis**, **Centered Axis**, or **Dual Axis**.
7. Open the new card and select the board input port or ports.
8. Choose the action or stick target.
9. Calibrate an analog sensor if needed.
10. Tap **Edit Intent** for button timing, combinations, and sequences.
11. Tap **Done** in the Input Hub.

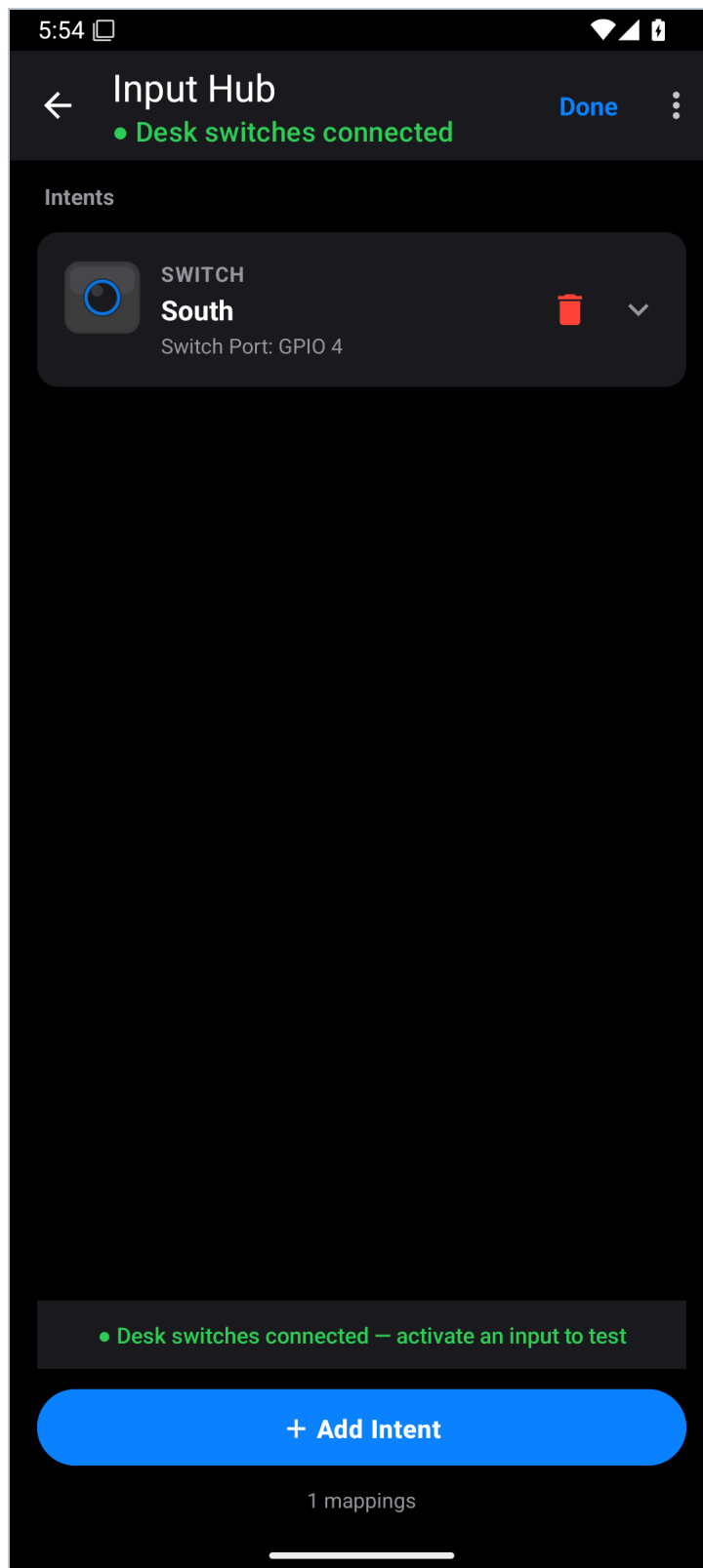


Figure 19. Check the board name and connection status at the top.

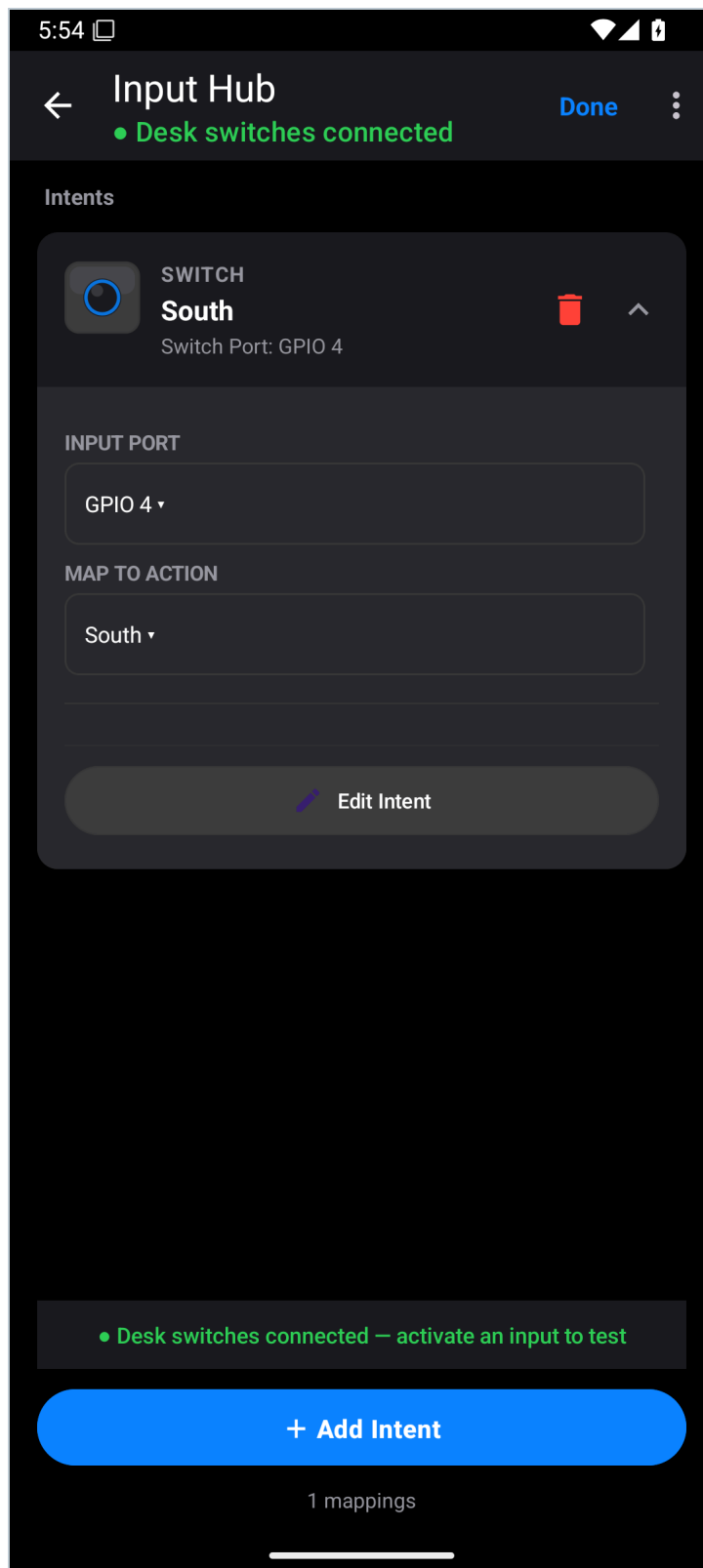


Figure 20. The port is the source; the action is what the game receives.

Pico W Link input

1. Tap **+ Input** → **Pico W Link**.
2. Select and verify the Pico W through the same connection journey.
3. Give it its own name.

4. Tap **Save and create mappings**.
5. Use **+ Add Intent** in its Input Hub.
6. Choose the mapping shape, source channel, and output target.
7. Check live readings and calibrate sensors as needed.
8. Tap **Done**.

The board editor currently shares port labels between board families. Some labels come from ESP32 naming, such as **GPIO 34**. They are not a Pico W wiring diagram. Do not assume a label means the same physical pin on both boards. Use the channel-to-pin information for the firmware installed on your board, and verify the changing reading in the app.

The protocol shows four analog channels. The current Pico W implementation has three distinct external ADC inputs, so four displayed channels do not mean four independent Pico analog pins.

Choose the right board mapping shape

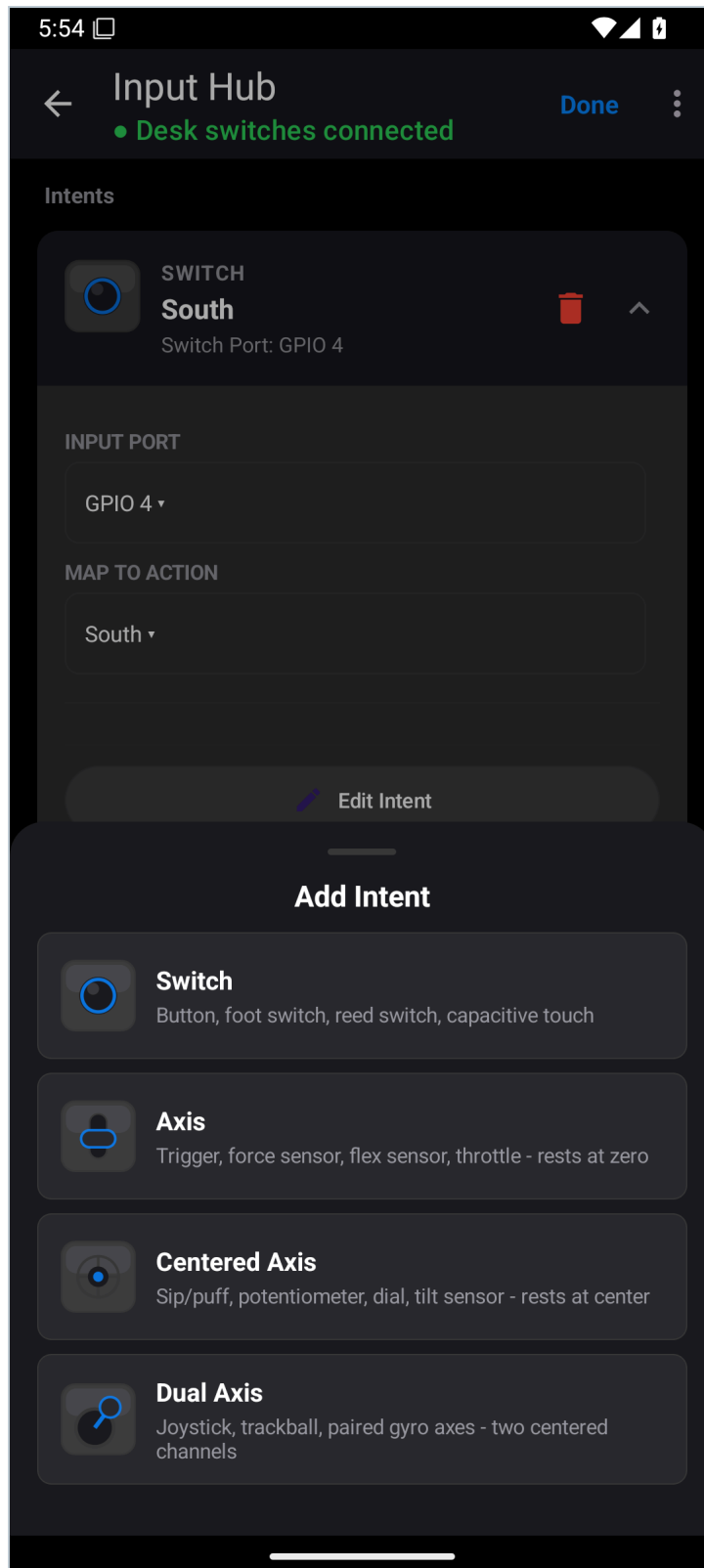


Figure 21. Choose based on how the physical control moves.

Shape	What it means	Example
Switch	Off or on	Button, foot switch, or contact switch

Shape	What it means	Example
Axis	One amount, usually resting near zero	Trigger, force sensor, flex sensor, or throttle
Centered Axis	Resting point between two directions	Sip/puff sensor, tilt control, or centred dial
Dual Axis	Two centred channels used together	X/Y joystick or two-axis movement sensor

For **Switch**, choose **Input Port**, then **Map to Action**. Press the switch and confirm its live feedback.

For **Axis**, choose the port and action. Release the sensor to its normal resting position. Tap **Record** to store that resting value. Check its full range with the calibration bar.

For **Centered Axis**, leave the control at rest and tap **Record**. Set the two actions, shown as **Sip Action** and **Puff Action** for this style of mapping. Check both directions. Neutral should sit between the active zones.

For **Dual Axis**, choose different X and Y ports and select **Target Stick**. Leave the stick centred and record its centre. Check each axis's minimum, neutral, and maximum values. Move left, right, up, and down to confirm the directions.

Calibration is how you tell the app what the sensor considers rest and full movement. If it moves the character while you are not touching it, check the resting value first. If a direction can never reach its full amount, check the range.

The board Input Hub allows up to eight mapping modules. Analog mappings also need free analog channels; a Dual Axis mapping needs two. This is separate from the eight input-device limit on the home screen.

Face Tracking input

Face Tracking uses the phone camera. Start with one gesture so you can learn how it feels.

1. Place the phone where your face is easy to see.
2. Tap **+ Input** → **Face Tracking**.
3. If asked, tap **Grant Permission** and allow camera access.
4. Choose the camera using **Switch Camera** if needed.
5. Tap **Calibrate** →.
6. Follow the capture prompts before adding complex mappings.

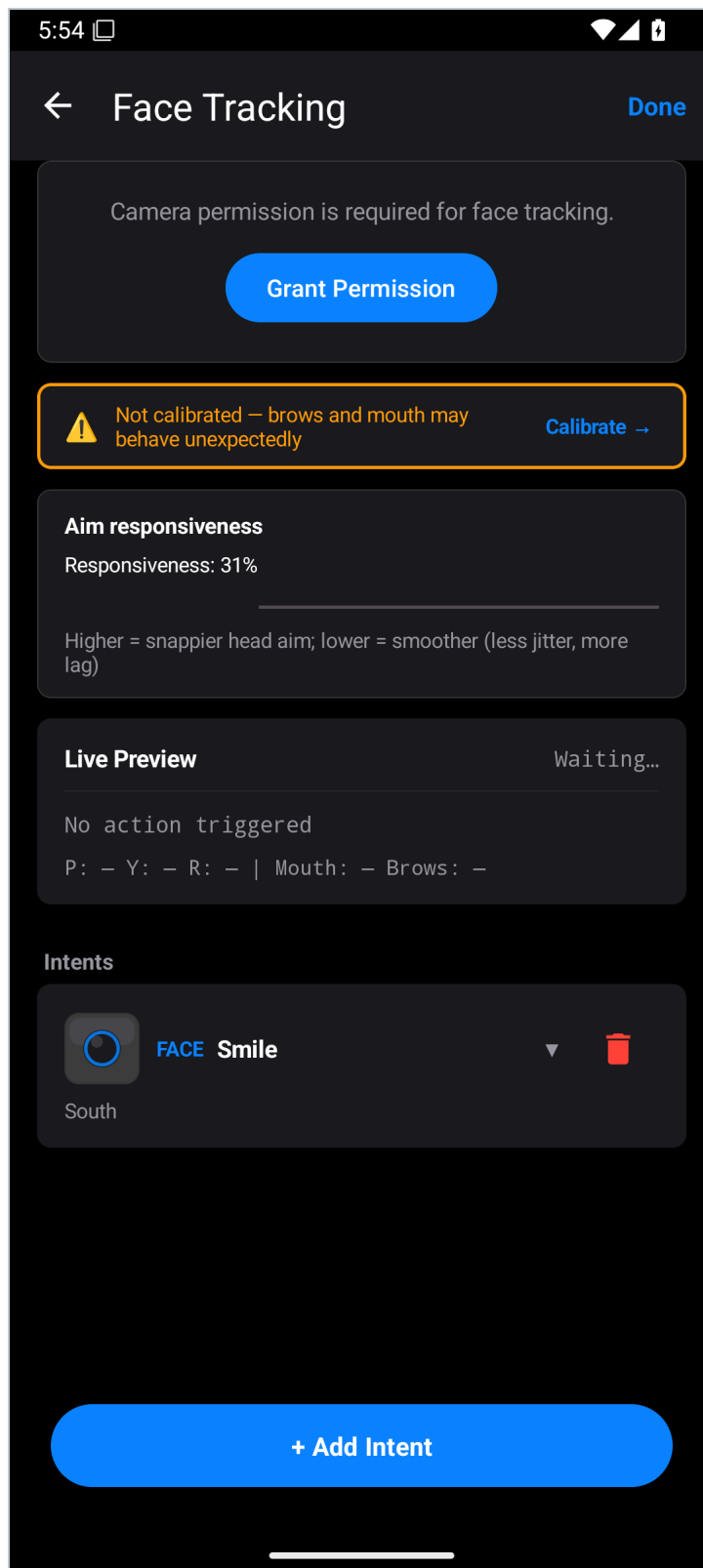


Figure 22. Camera permission is needed before live face readings can appear.

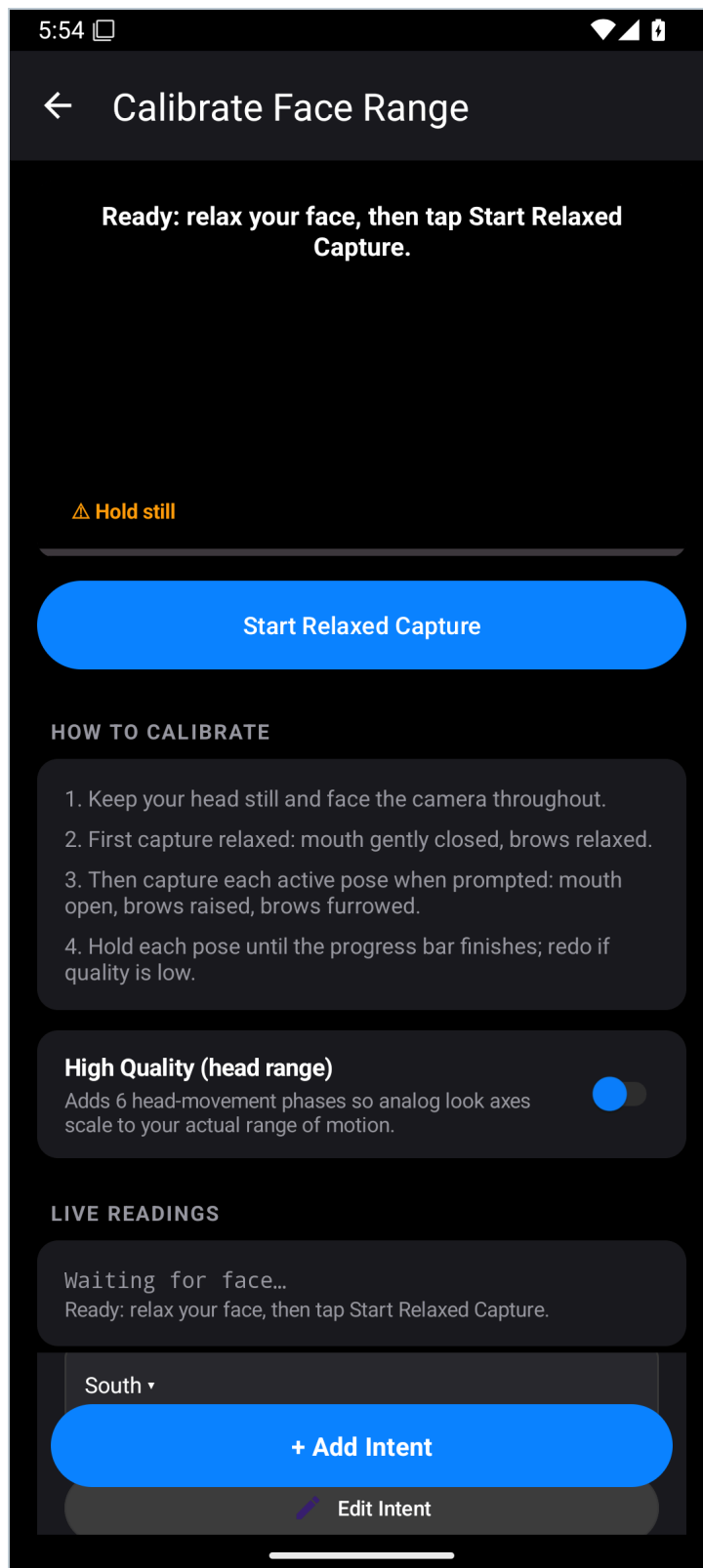
Calibrate your face

Sit in the position you will use while playing. Keep your face visible and use steady lighting.

For extra head movement calibration, turn on **High Quality (head range)** before you start. It adds comfortable turn, nod, and tilt captures after the expression captures.

1. Relax your face and tap **Start Relaxed Capture**.
2. When prompted, open your mouth and tap **Capture Mouth Open**.
3. Raise your brows and tap **Capture Brows Raised**.
4. Furrow your brows and tap **Capture Brows Furrowed**.
5. Wait for each capture to finish. Keep the requested expression steady.
6. If you use the longer calibration option, follow the extra prompts for right and left turns, looking down and up, and tilting left and right.
7. Let the app finish and return to the face editor. Retry a capture if its quality is too low.

Recalibrate if another person uses the profile, the camera position changes a lot, or neutral expressions keep causing actions.



Face calibration. Follow the pose shown above the camera before pressing each capture button.

Create face mappings

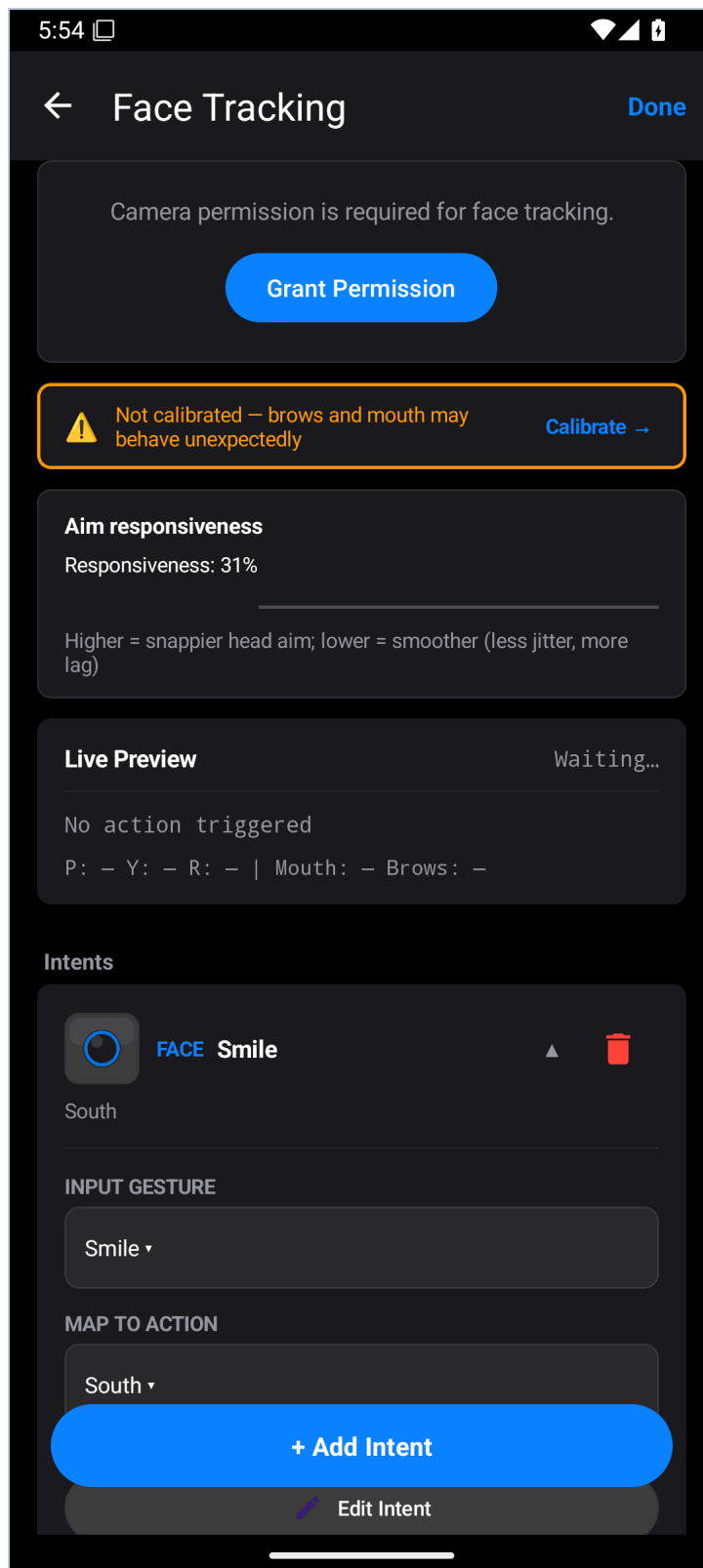
1. Tap **+ Add Intent**.
2. Choose the mapping shape. **Switch** suits a simple expression; **Axis** suits an amount; **Centered Axis** suits two sides of a resting point; **Dual Axis** suits head movement to a stick.

3. Open the card.
4. Choose its **Gesture** or the directional gestures for a stick mapping.
5. Choose **Map to Action**, the two centred actions, or **Target Stick**.
6. Watch **Live Preview** while you make that movement.
7. Open **ADVANCED SETTINGS** if you need to tune detection.
8. Tap **Edit Intent** for combinations, repetition, or timing.
9. Tap **Done**.

Available gesture choices depend on the mapping shape. The app includes winks, blink, smile, head nod and shake, look directions, tilt directions, mouth, and brows. A head-stick mapping uses directional gestures for up, down, left, and right.

Face setting	What it changes	When to adjust it
Aim responsiveness	How quickly head-based aiming reacts versus how much it is smoothed	Increase for quicker motion; reduce if movement feels shaky
Sensitivity	How strongly a movement affects the mapping	Change if a comfortable movement gives too little or too much output
Deadzone	The small area near rest that is ignored	Increase a little if tiny movements cause drift
Invert	The direction of an analog control	Change if movement goes the wrong way
Threshold	How far an expression must go before it counts	Raise to reduce accidental triggers; lower if it is hard to trigger
Hold Frames	How many camera frames must agree before a gesture counts	Increase for steadier detection; expect more delay
Separate open/closed or raised/furrowed thresholds	The two active zones of a centred expression mapping	Adjust each side without moving the other side's target

Not every setting appears for every gesture. Change one setting at a time and test it. Avoid using a movement you often make by accident for a critical game action.



Face mapping. The gesture is your source; the action is the controller output.

Virtual Input

Virtual Input is a controller drawn on the phone. Only one Virtual Input is allowed in a profile.

1. Tap **+ Input** → **Virtual Input**.
2. The app adds starter virtual-controller mappings.

3. Tap its card or edit control to open the layout editor.
4. Turn the phone sideways if needed. The editor is in landscape.

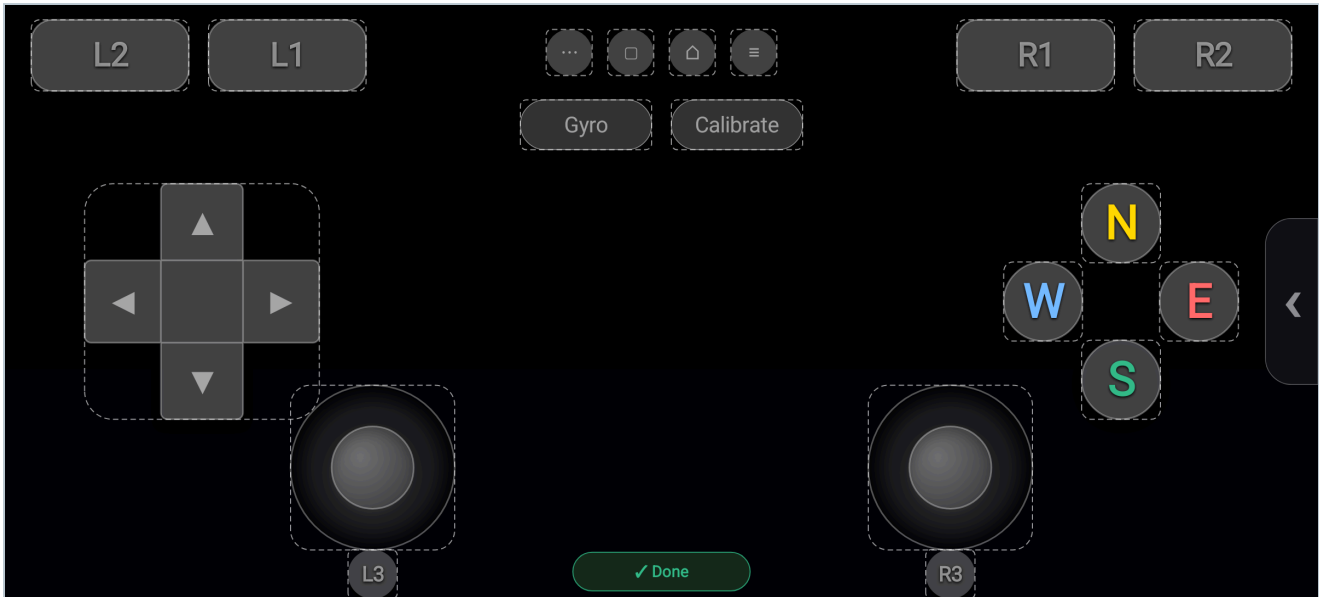
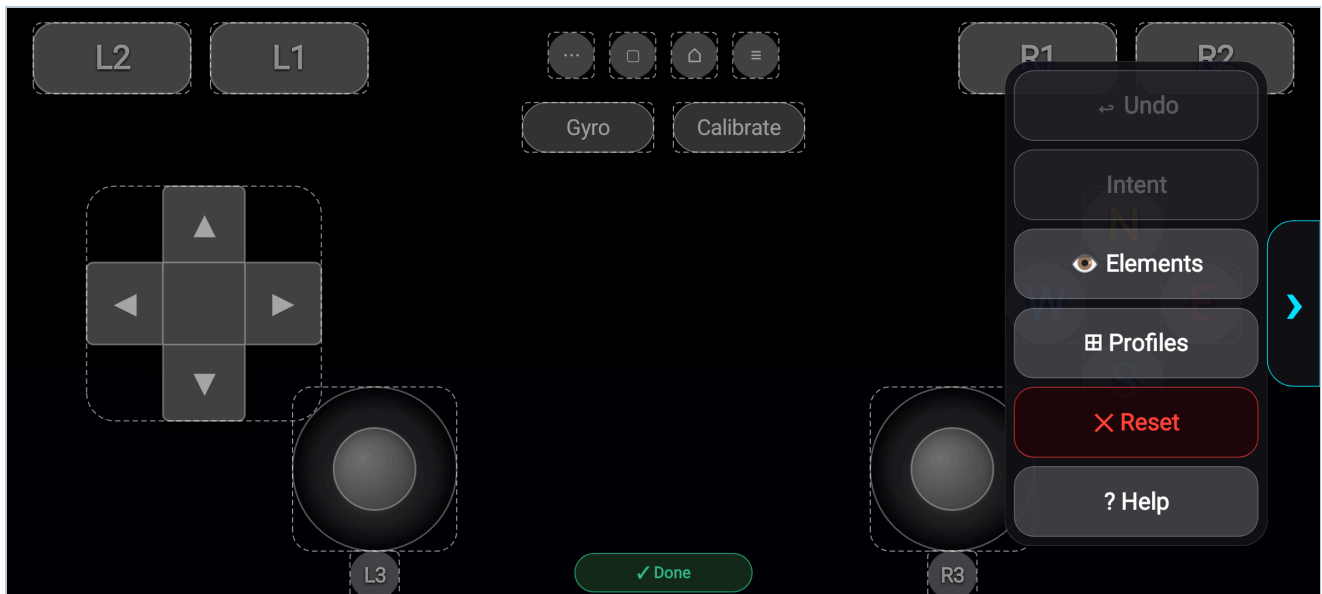


Figure 23. Dashed outlines show that the layout is being edited.

Change positions and sizes

1. Drag a control to move it.
2. Tap a control to select it.
3. Pinch with two fingers to resize the selected control.
4. Use the ✕ badge to hide a selected element.
5. Tap the chevron on the right edge to open **Edit tools**.
6. Tap 👁 **Elements** to show or hide controls using the list.
7. Use ↶ **Undo** to undo recent layout edits.
8. Tap ✓ **Done** to leave edit mode when finished.

Make the buttons you use often large enough to press comfortably. Move less-used buttons away from your thumb's usual path.



Virtual edit tools. Open the right-edge chevron to reach **Intent**, **Elements**, **Profiles**, **Reset**, and **Help**.

Change what a virtual button does

1. In edit mode, select a button.
2. Open the right-edge tools panel.
3. Tap **Intent**.
4. Edit its outputs and behaviours.
5. Tap **Save & Close**.

For a D-pad, choose the direction to edit when asked. The button Intent option is for buttons and directions. A stick's continuous movement is not edited as a single button Intent.

Save a layout

1. Open **Edit tools** → **Profiles**.
2. Tap **+ Save current as new profile...**
3. Enter a layout name.
4. Tap **Save**.
5. To load it later, open **Profiles** and tap the layout name.

Default Layout restores the default positions. **Manage...** offers actions for saved layouts. **X Reset** resets the current layout after confirmation. **? Help** shows the layout editing hint.

Layout profiles save button positions, sizes, and visibility. The main home-menu profiles save your input mappings and output choice. These are different things.

Use the virtual controller

Return to the home screen and tap **Start Mapping**. The virtual controller play screen opens. Use its **Stop** control to end the mapping session.

If your phone has a gyroscope, **Gyro** and **Calibrate** can be used for motion controls. Hold the phone still during the calibration countdown, check the result, and use **Looks good** or **Redo** as offered. Motion output is currently supported by PS Remote Play. Selecting Gyro does not add motion support to every other output.

Configure every output type

This Device: the phone as a Bluetooth gamepad

Choose this when you want a compatible receiving device to see the phone as a game controller.

1. Turn on Bluetooth on the phone and the receiving device.
2. Tap **Add Output / Change Output** → **This Device**.
3. Add and configure your inputs.
4. Tap **Start Mapping**.
5. Accept the Bluetooth or nearby-device permission request if it appears.
6. Allow the phone to be discoverable when Android asks.
7. On the receiving device, open Bluetooth settings and add a new device.
8. Select the phone or its **Intent Controller** gamepad entry. The name shown can depend on the receiving device.
9. Accept pairing on both devices if asked.
10. Test a simple action.

On later runs, the app can try a remembered Bluetooth host. You still need to power the host, enable Bluetooth, and tap **Start Mapping**.

This output sends a Bluetooth gamepad. It does not send typed keyboard text or desktop mouse input. A receiving device must support the Bluetooth gamepad connection; some consoles do not accept a generic Bluetooth controller.

ESP32 output

An ESP32 output is a bridge. There are two separate connections:

Phone → ESP32 board → receiving computer or console.

The **Wi-Fi / USB** choice in board setup controls the first connection. It does not choose how the board presents a gamepad to the receiving host. That second connection depends on the board hardware and its firmware.

1. Tap **Add Output / Change Output** → **ESP32**.
2. Select and verify the output board using [board setup](#).
3. Tap **Save and choose controller**.
4. On the ESP32 screen, open **CONTROLLERS** if it is folded.
5. Select a saved controller identity, or tap **+ Add**.
6. Enter an identity or scan a USB controller as described below.
7. Select the identity in the controller list. A tick shows the selection.

8. Tap **Apply to ESP32**.
9. Wait while the app applies the details and checks the board. A restart may briefly disconnect it.
10. Finish with the offered completion or **Continue with...** control, or return home once the applied setup is confirmed.
11. Connect or pair the board with the receiving host using the method supported by that board.
12. Configure your inputs and tap **Start Mapping**.

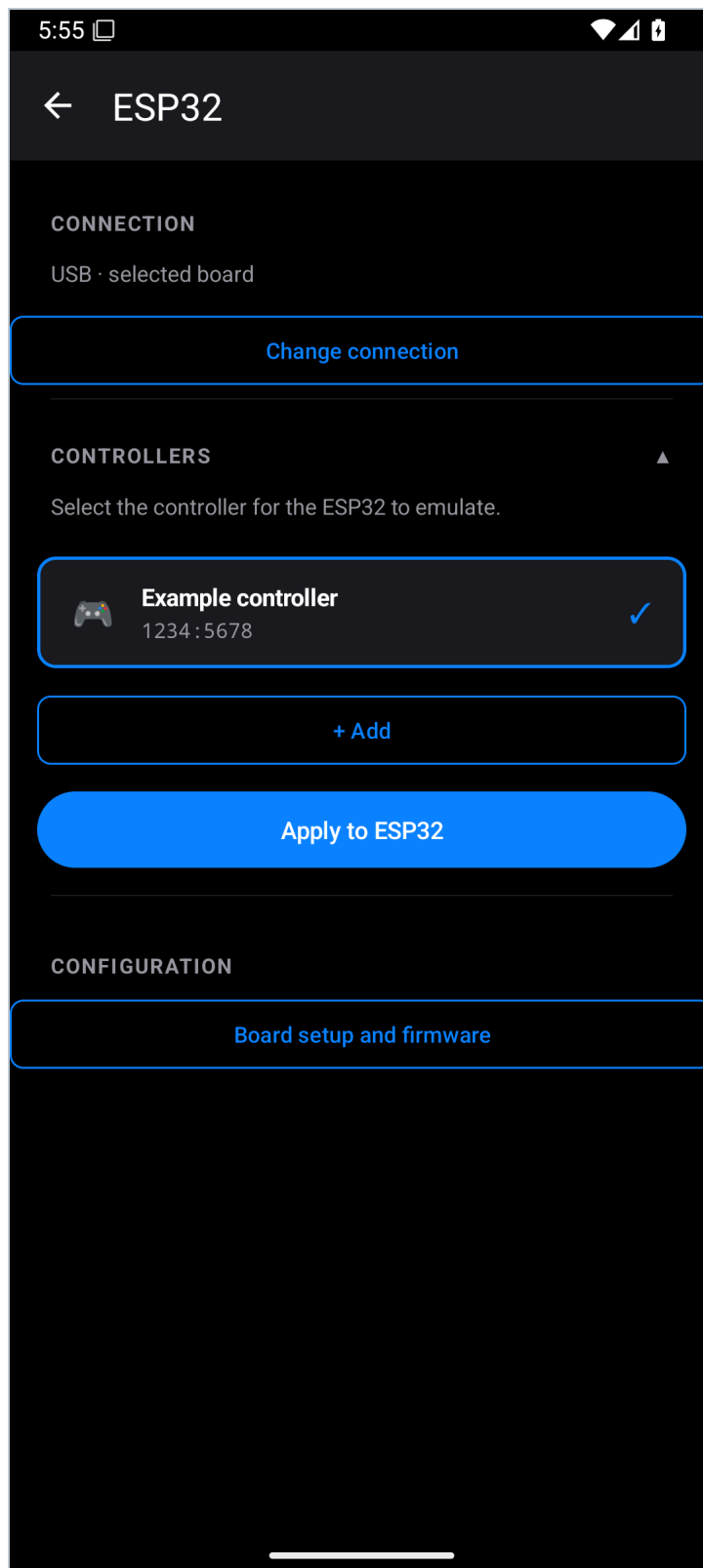


Figure 24. This example shows a USB phone-to-board connection. Selecting an identity is separate from applying it.

Use **Change connection** if the phone-to-board link changes. **Board setup and firmware** opens the board setup tools. Your input mappings do not need to be rewritten just because the bridge's address changes.

If you change the selected identity or replace the output board, apply and verify the identity again. The app can block starting while that configuration still needs to be applied.

Add or scan a controller identity

A controller **identity** is the name and identifying numbers the board should use. It is separate from your input mappings.

The identity form can include:

Field	Meaning
Display name	The label you recognise in the app
Canonical or HID name	The controller name presented by the hardware
VID	Vendor ID, entered in the format requested by the form
PID	Product ID, entered in the format requested by the form
Controller MAC, where offered	A controller address for hardware that uses it

Use the details for your actual supported controller setup. Copying a name or VID/PID does not guarantee that a console will accept the board. Some hosts need controller features or authentication beyond those fields.

To read an attached controller's details:

1. Choose **Scan USB Controller** from the add-controller flow.
2. Connect the physical controller to the phone through USB OTG.
3. Allow USB access if asked.
4. Check the detected device details.
5. Tap **Mirror Device**, or the button naming the detected device.
6. Return to the output's controller list, select the saved identity, and tap **Apply to ESP32**.

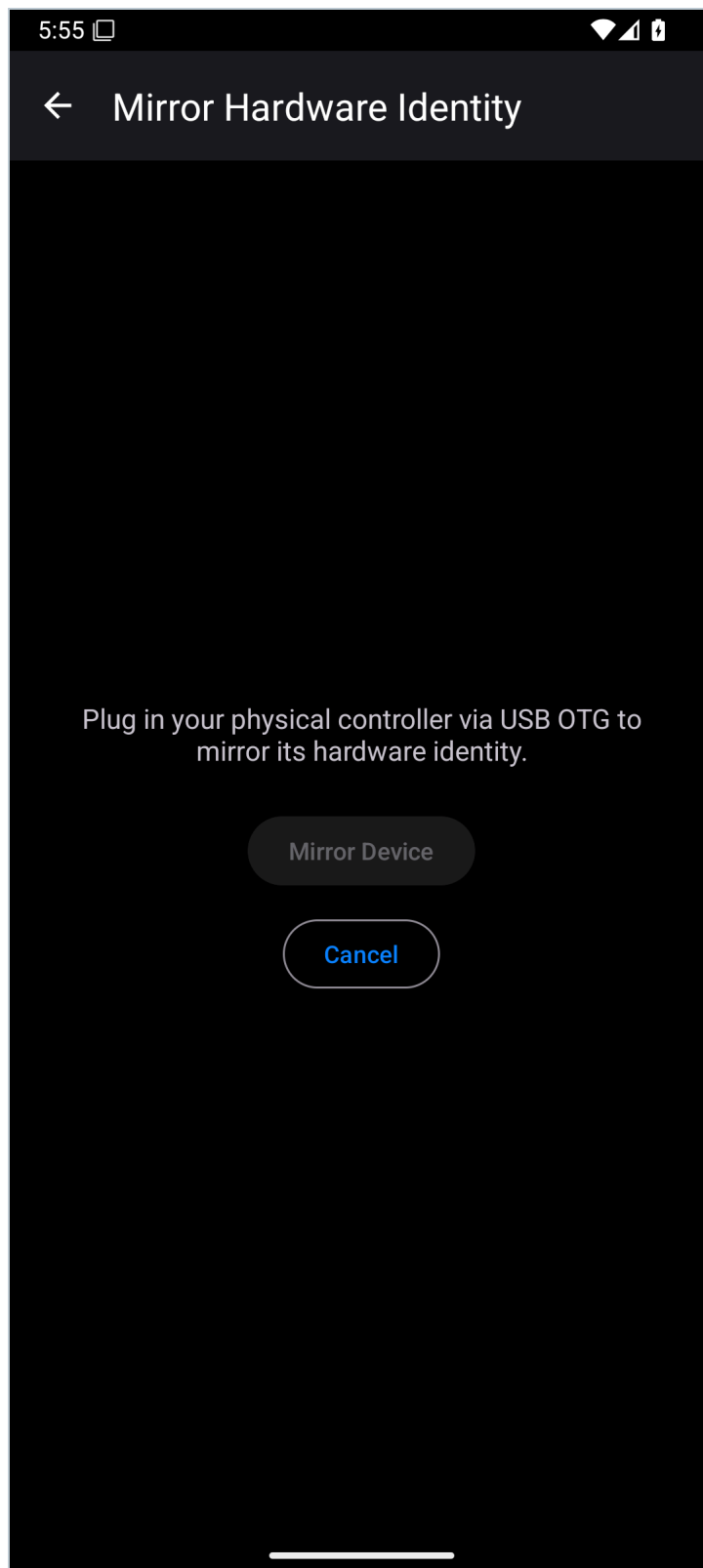


Figure 25. The scan button becomes usable when a suitable controller is detected.

Scanning an identity does not add that physical controller as an input. Use **+ Input → Controller** separately if you also want to read its buttons.

Pico W output

1. Tap **Add Output / Change Output** → **Pico W**.
2. Complete the board connection journey for the selected Pico W.
3. Tap **Save and choose controller**.
4. Open **CONTROLLERS**.
5. Select a saved identity or tap **Add Controller** to create one.
6. Tap **Apply to Pico W**.
7. Wait for the board to respond and for the applied setup to be checked.
8. Tap **Confirm** when the setup is ready.
9. Connect the board to its receiving host using the supported board output method.
10. Return home, configure inputs, and tap **Start Mapping**.

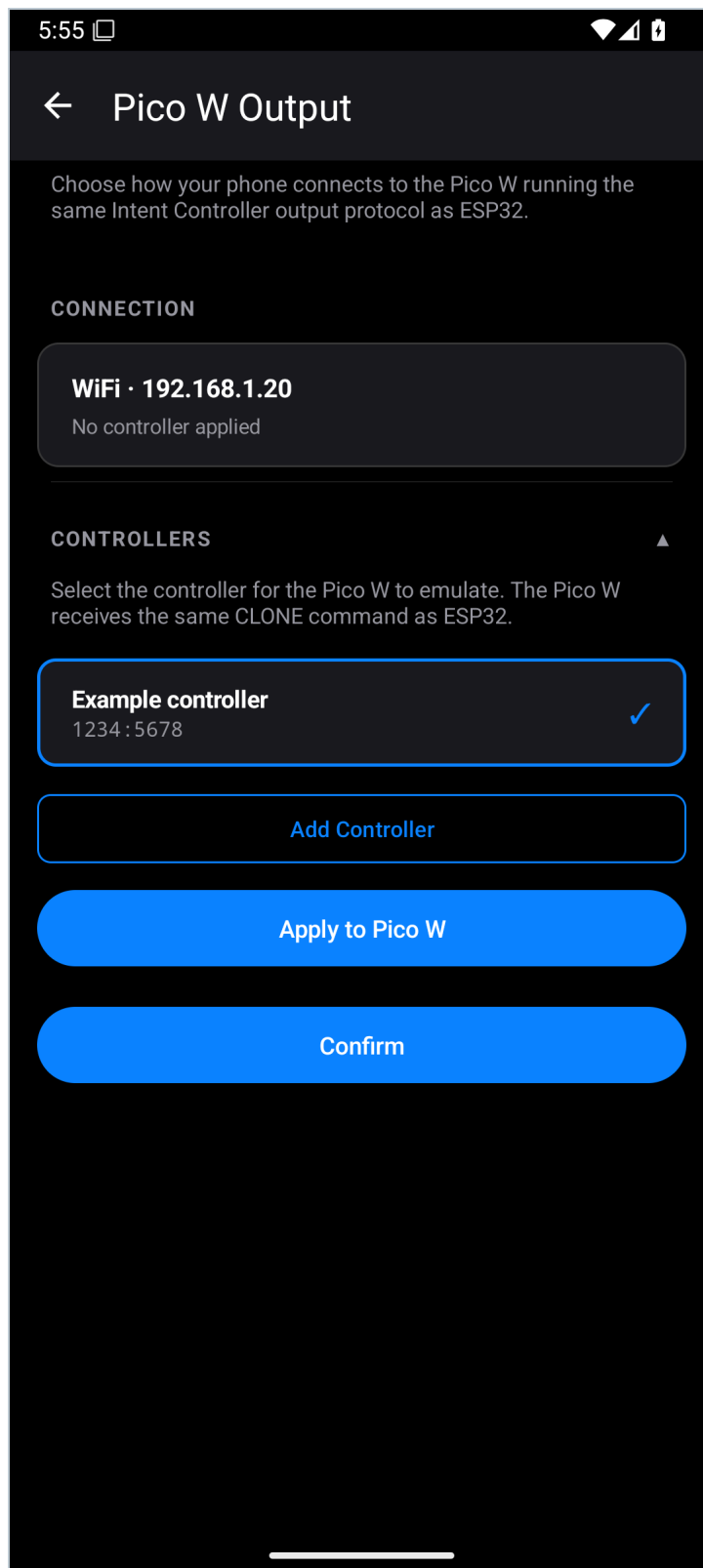


Figure 26. **No controller applied** means that selecting the row alone has not finished setup.

Use **Change connection** or the connection settings control offered on this screen to update the phone-to-board connection. The identity fields and scanning idea are the same as for ESP32. Confirming the phone's connection does not prove the receiving console accepts the controller; test that host connection too.

PS Remote Play output

This is Intent Controller's own Remote Play connection. It is marked **BETA** and is not Sony's app. The app currently offers PS5 and PS4 console choices in setup.

Prepare the console

For first pairing, put the phone and console on the same network and turn the console on.

- **PS5:** open **Settings** → **System** → **Remote Play** and turn on **Enable Remote Play**. To display a pairing code, use **Pair Device**. Some console or app text calls this **Link Device**.
- **PS4:** open **Settings** → **Remote Play Connection Settings**, turn on **Enable Remote Play**, and use **Add Device** to display the code.

These console menu paths are described in [PlayStation's Remote Play settings guide](#). Console wording can change with system updates. The app's pairing screen currently shows PS5-style instructions, so use the PS4 path above if that is your console.

Step 1: find the console

1. Tap **Add Output / Change Output** → **PS Remote Play**.
2. Read the app's screen-and-audio notice. Tap **I Understand** to continue when you have read it.
3. Tap **Find My Console**.
4. Select your console from the results.
5. Tap **Next**.

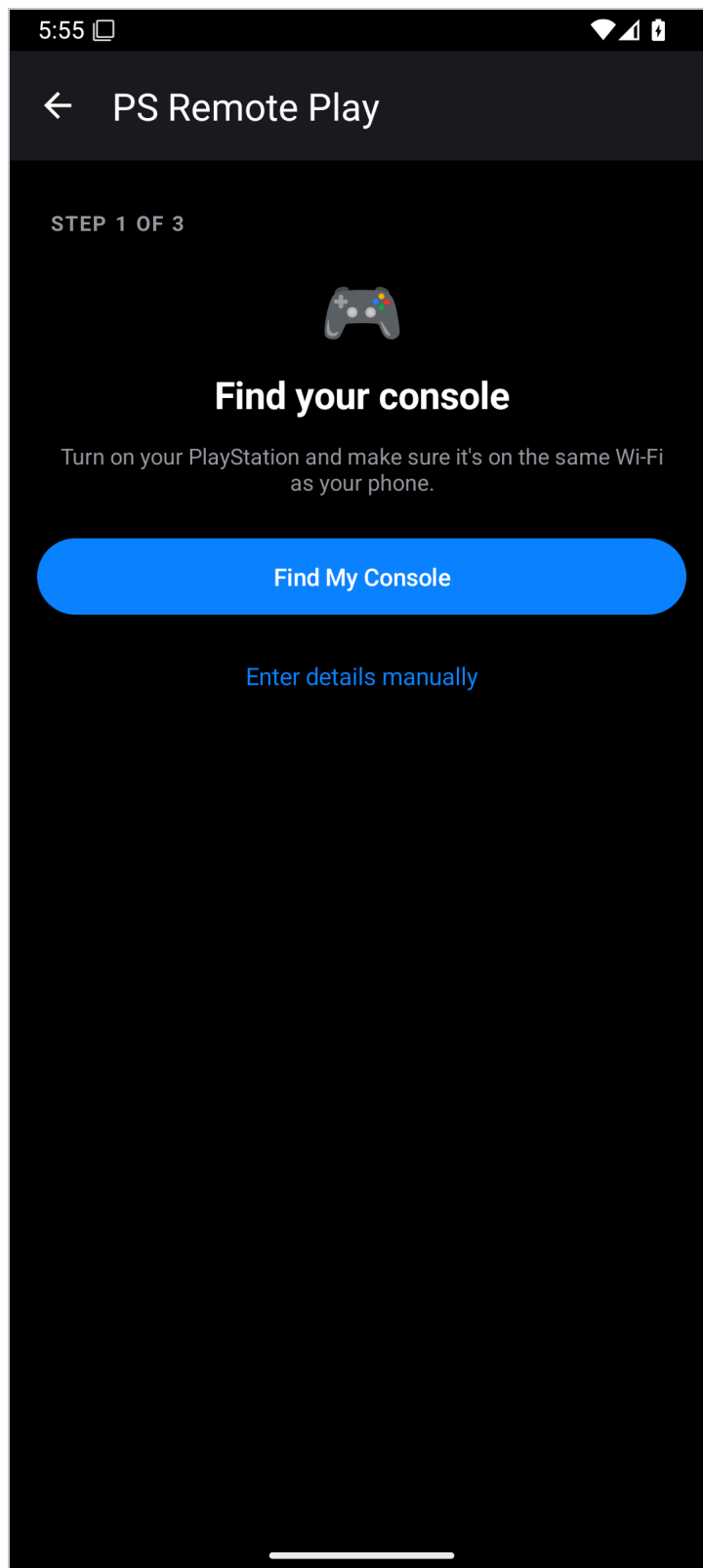


Figure 27. Start with discovery on the same network.

If the console is not found, use **Search again** or **Enter details manually**. In manual setup, choose **PS5** or **PS4** and enter the console's IP address. Find the address in the console's network information. Then tap **Next**.

Step 2: enter the account ID

1. In **Your Account ID**, enter the PSN account ID for the profile you are pairing on the console.
2. This is an account identifier, not your email, password, or display name.
3. If you do not know it, **Find a lookup site** opens a search for account-ID lookup tools.
4. Use a service you are comfortable using. The app describes username-based lookup; it does not need your password for this field.
5. Copy the returned ID and paste it into **Account ID**. The app accepts the supported numeric or encoded forms described on screen.
6. Scroll down if needed and tap **Next**.

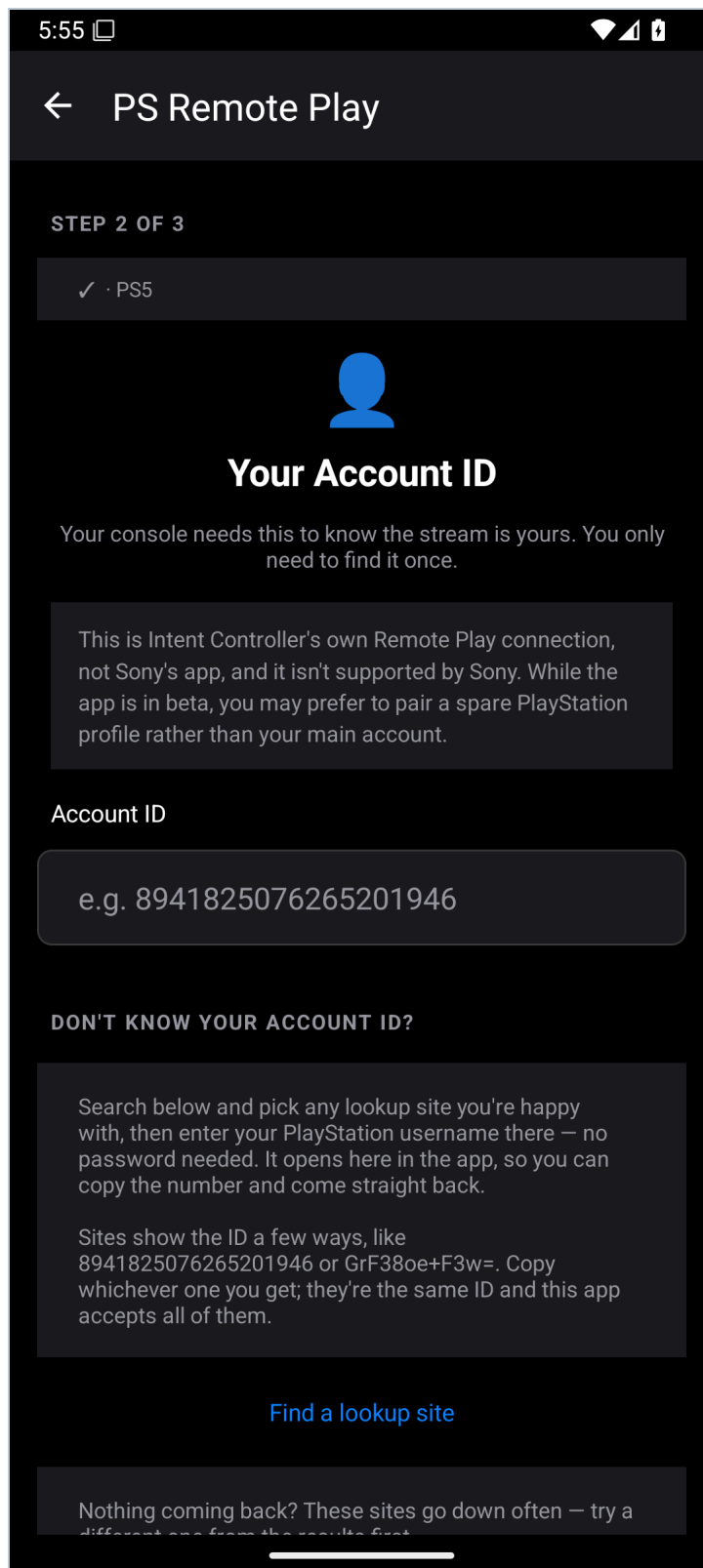


Figure 28. Read the account-ID help before entering a username in the wrong field.

The screen suggests using a spare PlayStation profile while this connection is in beta. Follow that option if it suits your setup.

Step 3: pair

1. Display the pairing code on the console using the console menu above.

2. In the app, tap **Pair with Console**.
3. Enter the current eight-digit PIN when asked.
4. Tap **Pair**.
5. Wait for the success message.
6. Tap **Finish**.
7. Return home, configure inputs, and tap **Start Mapping**.

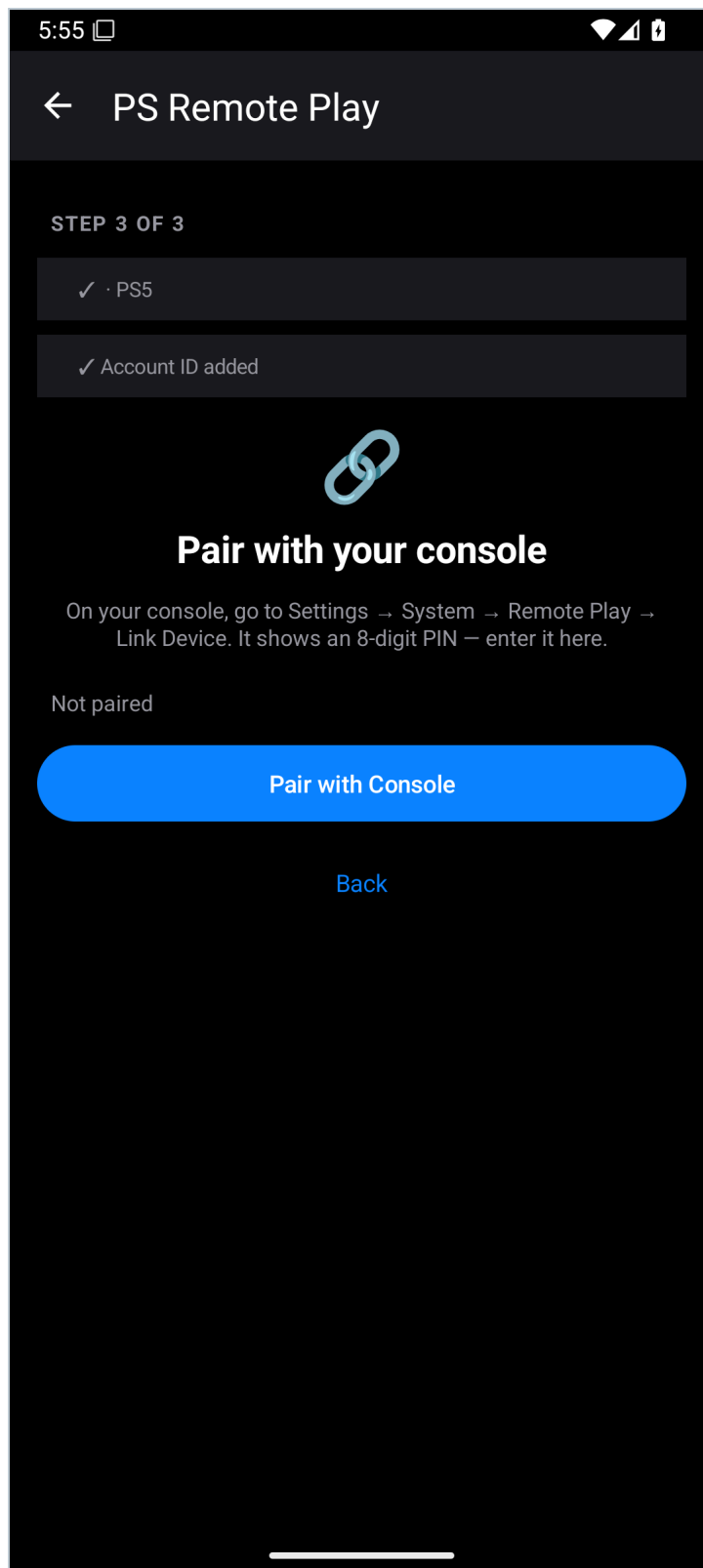


Figure 29. This example is still **Not paired**. Your own setup must complete pairing before you play.

For later sessions, use the saved console and account registration. You normally do not enter a new PIN each time. If the console's address changes or registration no longer works, reopen output setup and repair the connection or pair again.

The app connects directly to the console; installing Sony's Remote Play app is not a step in this setup. The destination uses the app's own session. Motion controls are supported on this output when the input provides them.

Xbox Remote Play output

This output is also **BETA**. It uses Microsoft sign-in and the consoles linked to your account. The connection uses Microsoft's services, so internet access is needed even though some selection text refers to a local network.

Prepare your Xbox with remote features enabled. On Xbox One or Series consoles, look under **Settings** → **Devices & connections** → **Remote features** for **Enable remote features**. Use [Xbox's Remote Play setup support](#) if your console's menu differs. Microsoft's [Remote Play page](#) confirms the need for a supported console, remote features, a Microsoft account with an Xbox profile, and an internet connection.

1. Tap **Add Output / Change Output** → **Xbox Remote Play**.
2. Read the screen-and-audio notice and tap **I Understand** when ready.
3. Tap **Sign in with Microsoft**.
4. Complete the Microsoft sign-in page with the account linked to your console.
5. Return to the app and check the signed-in account.
6. Tap **Next**.
7. On the console-selection step, choose the Xbox you want to use. Check the name if there is more than one.
8. Tap **Finish**.
9. Return home, configure your inputs, and tap **Start Mapping**.

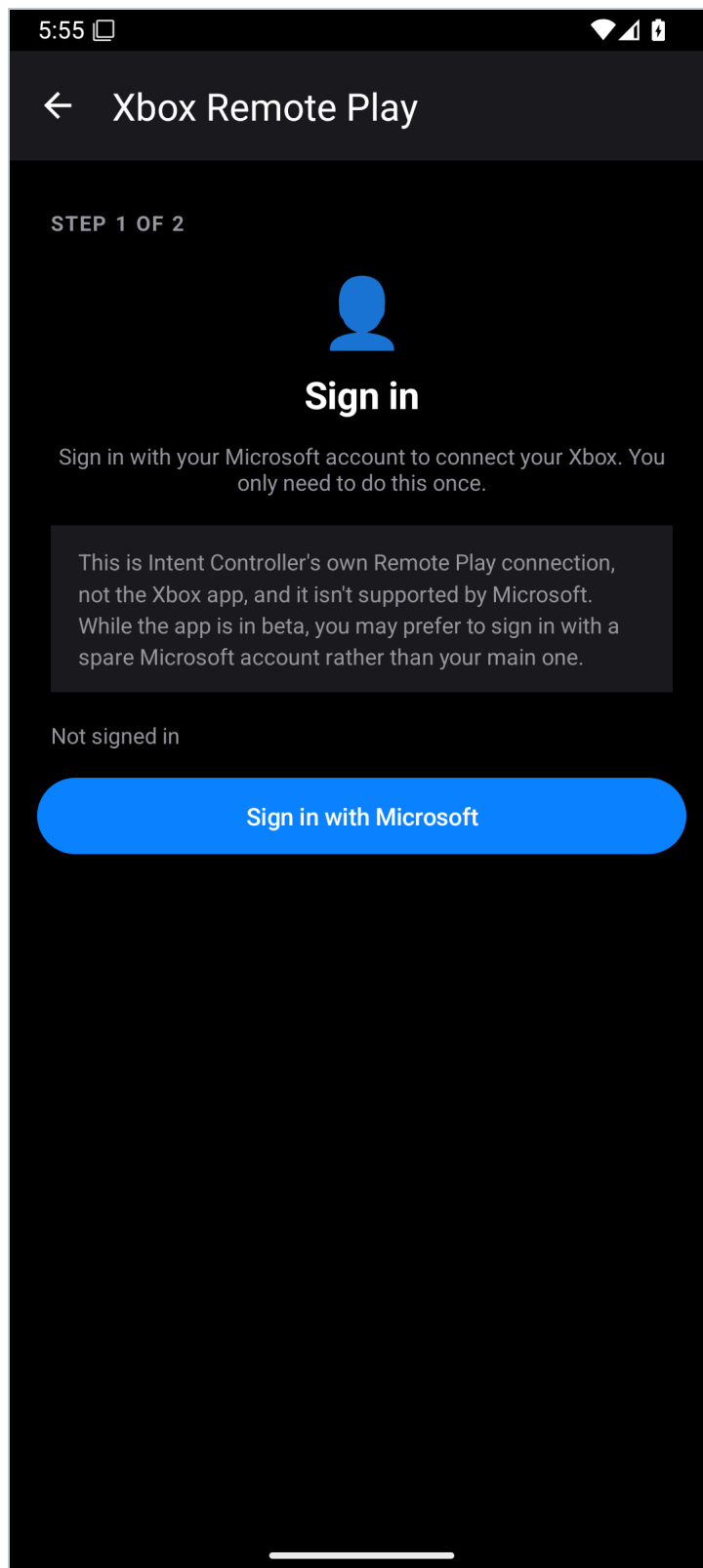


Figure 30. Sign in first; console selection follows.

Use **Remove account** if you need to sign out or switch Microsoft accounts. If sign-in expires, sign in again.

If the app says sign-in is unavailable in this build, that build cannot complete this output setup. Use a build with Xbox sign-in configured. Re-entering a Wi-Fi password will not fix that message.

This connects to your own console. It is not a cloud-gaming subscription setup, and selecting it does not grant access to games you do not own or have access to.

Use the Intent Engine

The Intent Engine turns a simple source into one or more controller actions. It also decides how timing, requirements, and conflicts work.

This section covers the features available in the current app editor. You do not need to edit configuration files or know how the software is written.

Open and save an Intent

1. Open the input's mapping editor.
2. Expand the mapping card.
3. Tap **Edit Intent**.
4. Give it a useful name, such as **Aim and shoot**.
5. Make your changes.
6. Tap **Save & Close**.
7. Finish the input editor with **Done**.

For Virtual Input, select a button in the layout editor and use **Edit tools** → **Intent**.

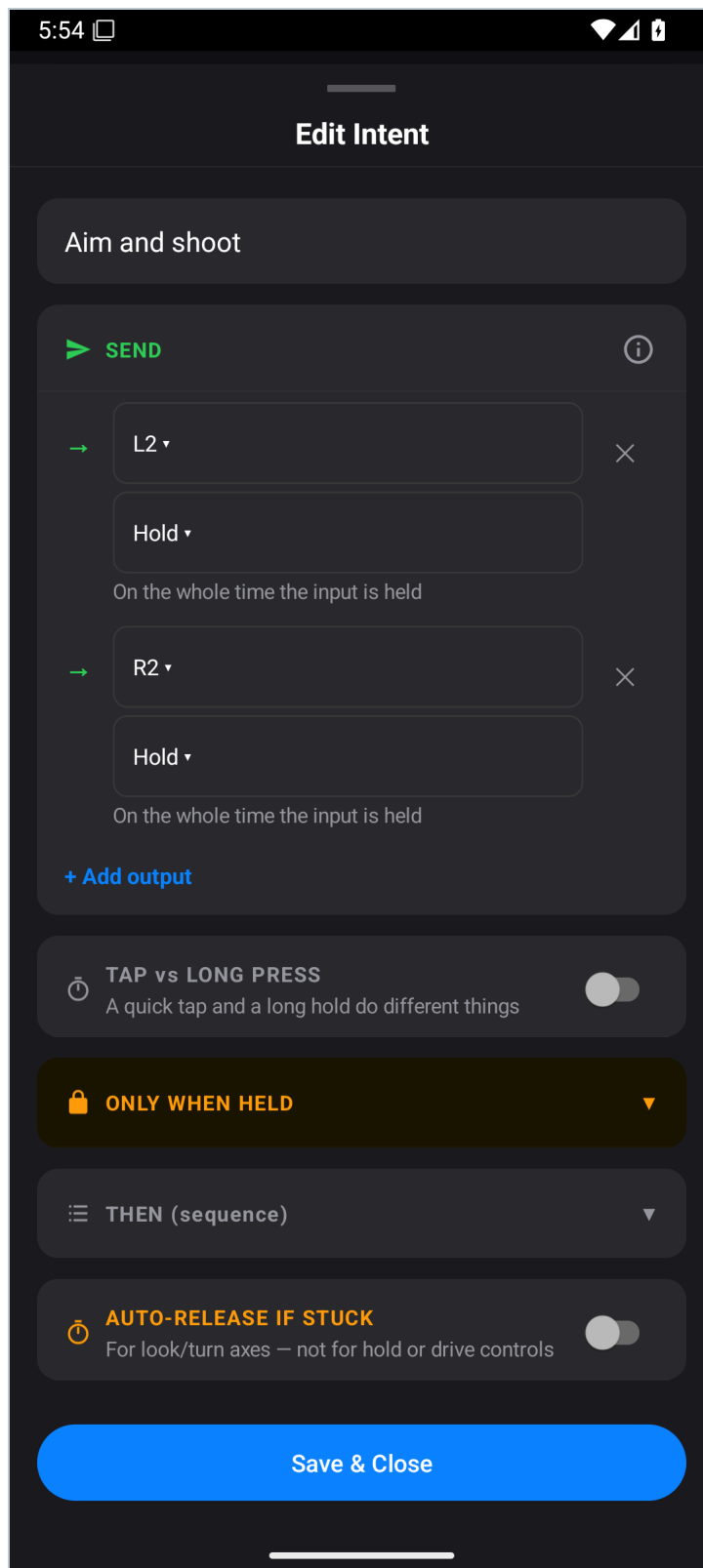


Figure 31. One input can send more than one output. This example holds both triggers.

The name

The name is a label for you. It helps you recognise the mapping in cards and the Control Map. A name such as **Aim and shoot** does not set up those actions by itself. You still need to add the right outputs.

SEND: choose the actions

Each row under **SEND** has a target and a behaviour.

1. Tap the target drop-down to choose the controller control.
2. Tap the behaviour drop-down to choose what it should do.
3. Tap **+ Add output** to add another row.
4. Remove an unwanted row with its **×** control.
5. Use the row's timing controls when shown. The clock icon opens timing fields where available.
6. Tap the ⓘ icon to read the section help.

Typical targets include the four face buttons, D-pad directions, shoulder buttons, triggers, stick clicks, menu buttons, and sticks. [The button-name table](#) translates the neutral names into familiar controller labels.

For **Left Stick** or **Right Stick**, choose the direction chips shown by the editor. These represent stick movement, which is different from a D-pad direction.

Multiple outputs are handled separately. They can start together, but each one still goes through the app's priority rules. Adding two rows does not give them special priority over other inputs.

All six SEND behaviours

ms means milliseconds. There are 1,000 ms in one second. **500 ms** is half a second.

Behaviour	What happens	When it is useful
Hold	The output stays active while the input is active, then releases	Aim, accelerate, block, or hold-to-sprint
Tap	A press starts one short output pulse	Jump, confirm, reload, or a single menu press
Rapid tap	A press starts repeated taps for a set time	A short burst of repeated game-button presses
Tap while held	Taps repeat while you keep the input active	Repeated fire or repeated menu movement that stops when you let go
On release	A short output pulse starts when the active input is released	Confirming a choice when you let go of a switch
Long press	Nothing happens on a short press; after the hold threshold, the output becomes active until release	Avoiding accidental activation of a secondary action

Hold

Set the target and choose **Hold**. Press or activate the source to hold the output. Let go to release it.

For example, **L2 + Hold** is useful when the game uses L2 for aiming. If you use an analog source, its value may also affect an analog target. A simple switch gives an off/on action.

Tap

Set the target and choose **Tap**. The normal short pulse is 80 ms. Change its timing where the row offers it.

The tap starts on the press. It does not wait for you to let go. It can finish its short pulse even if you release the physical input sooner.

Choose this for a single press. Holding the source does not automatically turn it into repeated taps.

Rapid tap

Choose **Rapid tap**, set the **For** duration, and set **at** taps per second. For example, **1,000 ms at 5 taps/sec** creates a short repeated-tap burst.

This burst can continue for its configured duration after you release the source. Use **Tap while held** if you need repetition to stop as soon as you let go.

The rate is limited to 1–20 taps per second. A game may ignore taps that arrive too quickly. Test a slower rate first.

Tap while held

Choose **Tap while held** and set its taps-per-second rate. It repeats only while the source remains active.

This can help when repeatedly pressing a button is tiring. It cannot make a game accept a faster fire rate than the game supports.

On release

Choose **On release**. First activate the source, then let go. The output sends a short pulse when the source becomes inactive.

This is useful when release is easier to control than the first press. It also means there is no action until you release. Do not use it for a stop control that you need to work immediately on pressing.

Long press

Choose **Long press** and set the hold threshold, normally 500 ms to start with.

A short press does nothing. Once the source has stayed active long enough, the output stays active until you release it.

Use this when an action should need a deliberate hold. It differs from the two-function **TAP vs LONG PRESS** mode below.

TAP vs LONG PRESS: two jobs on one input

Turn on **TAP vs LONG PRESS** to give one input a quick action and a held action.

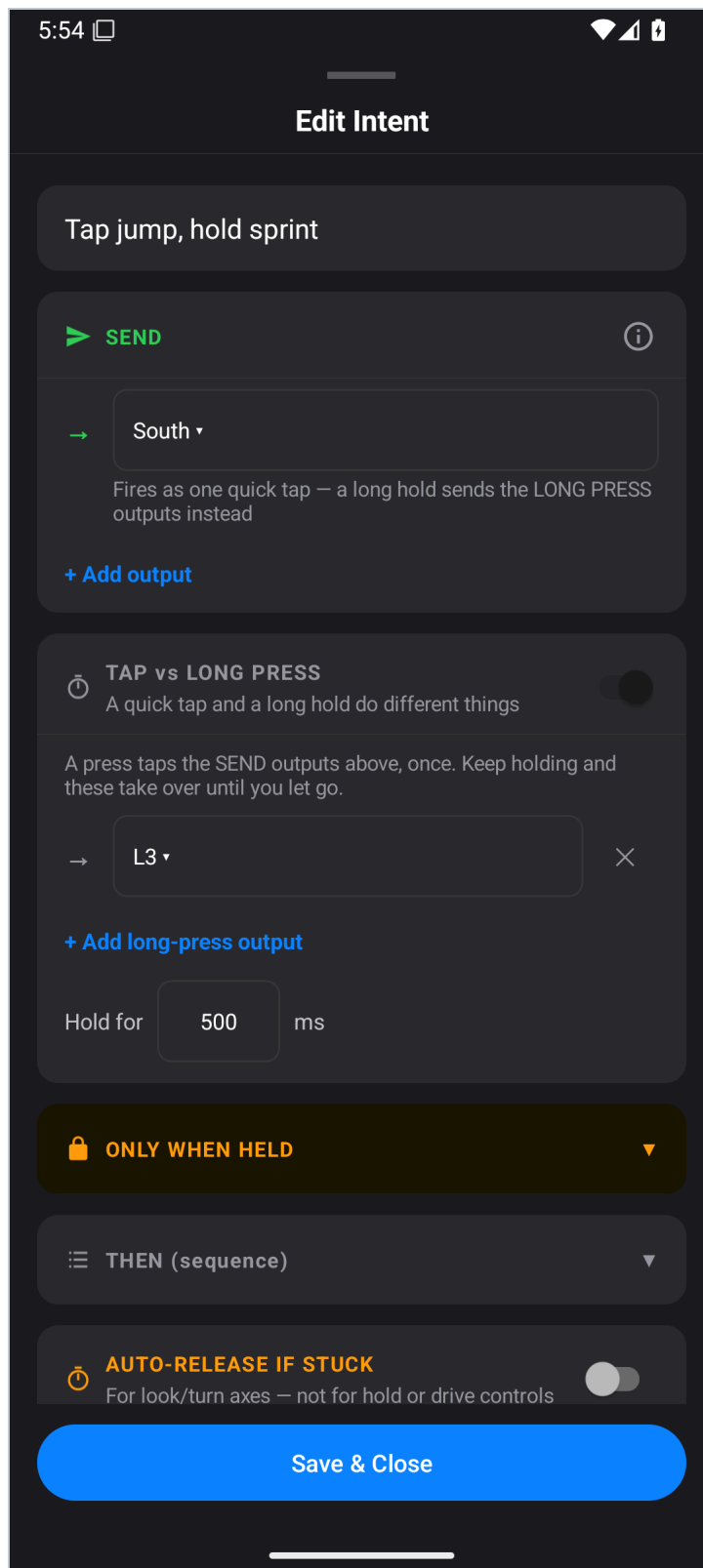


Figure 32. The initial press taps South. Holding for 500 ms then holds L3.

1. Put the quick action in **SEND**.
2. Turn on **TAP vs LONG PRESS**.
3. Choose a long-press target.
4. Use **+ Add long-press output** if you need more than one held output.

5. Set **Hold for** in ms.
6. Tap **Save & Close**.

The quick action happens when you first press, even if you keep holding. After the threshold, the long-press outputs take over until you let go. The app does not wait to discover whether you meant a tap or a hold before sending the first tap.

For example, the screenshot sends a jump tap, then adds sprint if you keep holding. This is useful only if that first jump is also acceptable when you want to sprint.

If you want a held action with no initial tap, leave this mode off and use the normal **Long press** behaviour in SEND.

When this mode is on, SEND acts as the quick-tap set. It is not the normal set of independent Hold/Rapid tap rows.

ONLY WHEN HELD: require another control

This is a condition. It checks that other listed controls are already active before this Intent can send its actions.

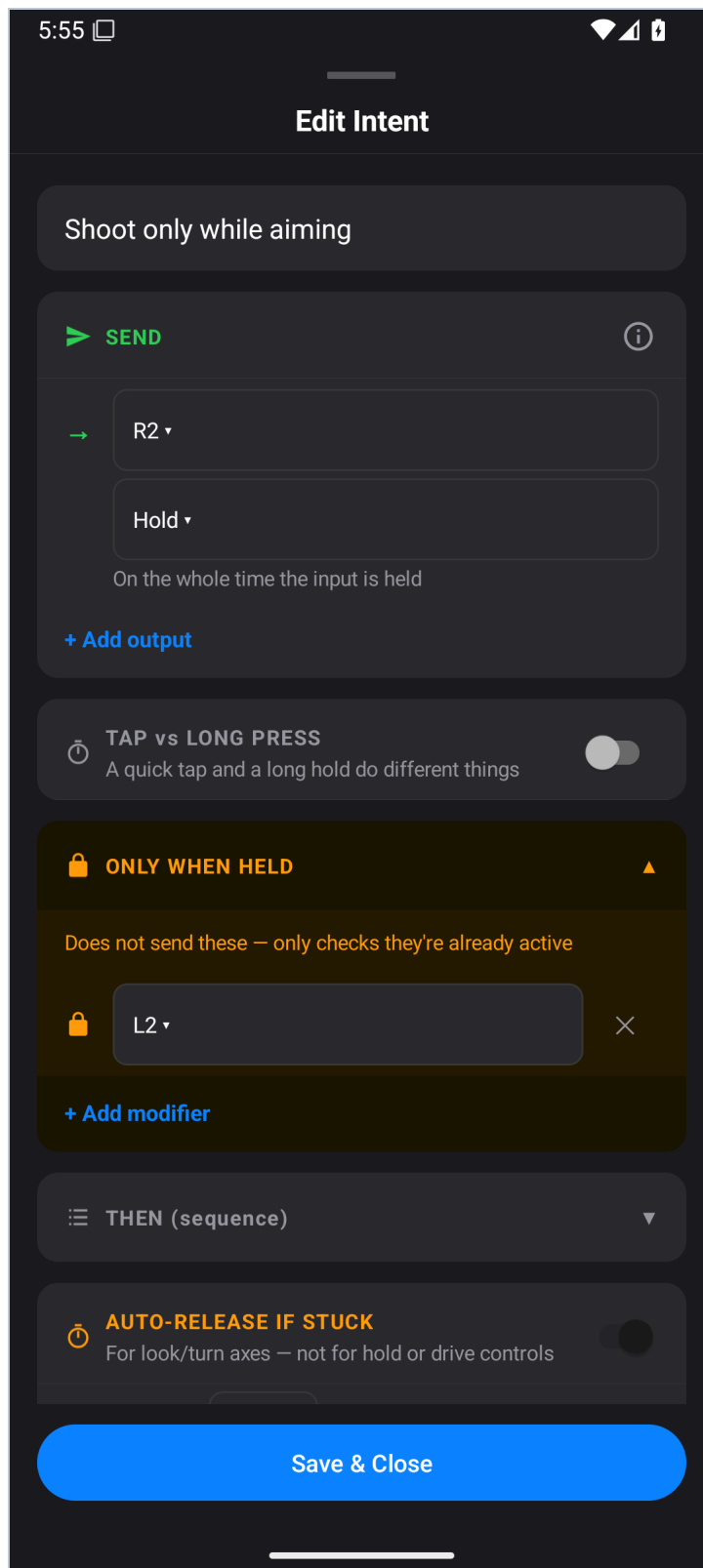


Figure 33. The L2 requirement checks aiming; it does not press L2 for you.

1. Open **ONLY WHEN HELD**.
2. Tap **+ Add modifier**.
3. Choose the required control.
4. Add more modifiers if needed.

5. Remove one with ✕ if it is no longer wanted.
6. Tap **Save & Close**.

All listed requirements must be active together. One requirement is not enough if you have listed two.

The check can use an active input with the matching control name or an active controller output from another mapping. This lets a board switch mapped to L2 act as the aiming requirement for another input's R2 Intent. An analog requirement must be far enough active to count; a tiny trigger movement may not pass it.

Use this for a second layer of controls or to reduce accidental actions. For example, permit fire only while aim is held.

Avoid a circular condition. If an Intent is the only thing that can produce L2, requiring L2 before that same Intent may prevent it from starting. Create a separate way to hold the required control.

If a required control stops being active, the gated Intent stops and its sequence is cancelled.

THEN (sequence): send an ordered set of steps

A sequence is useful when a game needs several controls in a particular order. It can reduce the number of physical movements you need to make.

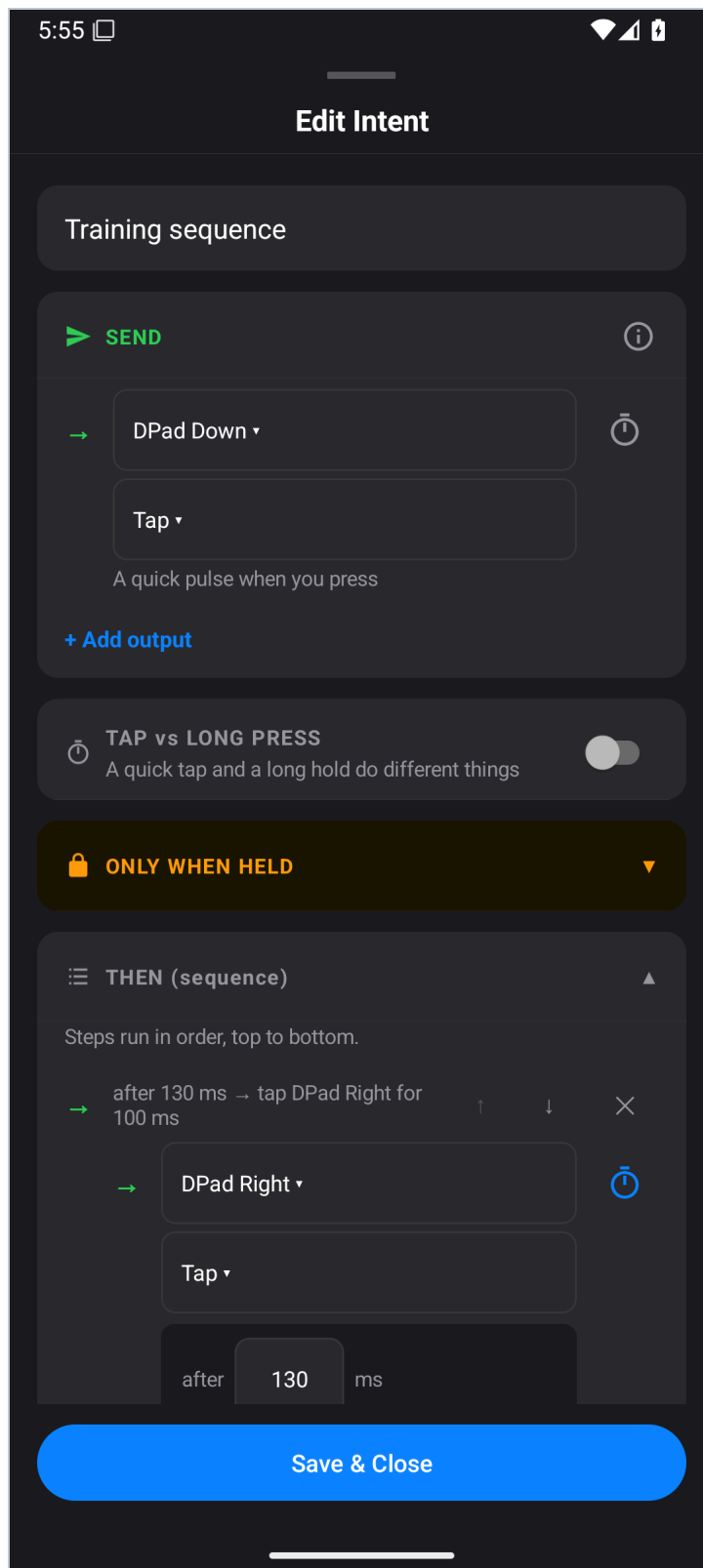


Figure 34. SEND starts the action. THEN contains the timed follow-up steps.

1. Put the starting action in **SEND**.
2. Open **THEN (sequence)**.
3. Tap **+ Add step**.
4. Choose the output target and behaviour for that step.

5. Tap the clock icon to show **after** and **for** timings.
6. **after** is the delay before that output starts within the step.
7. **for** is how long that output's step window lasts.
8. Use **+ Add output** inside the step if several outputs belong in the same step.
9. Add the next step with **+ Add step**.
10. Use **↑ / ↓** to change the step order. Use **×** to remove the selected step or row.
11. Tap **Save & Close**.

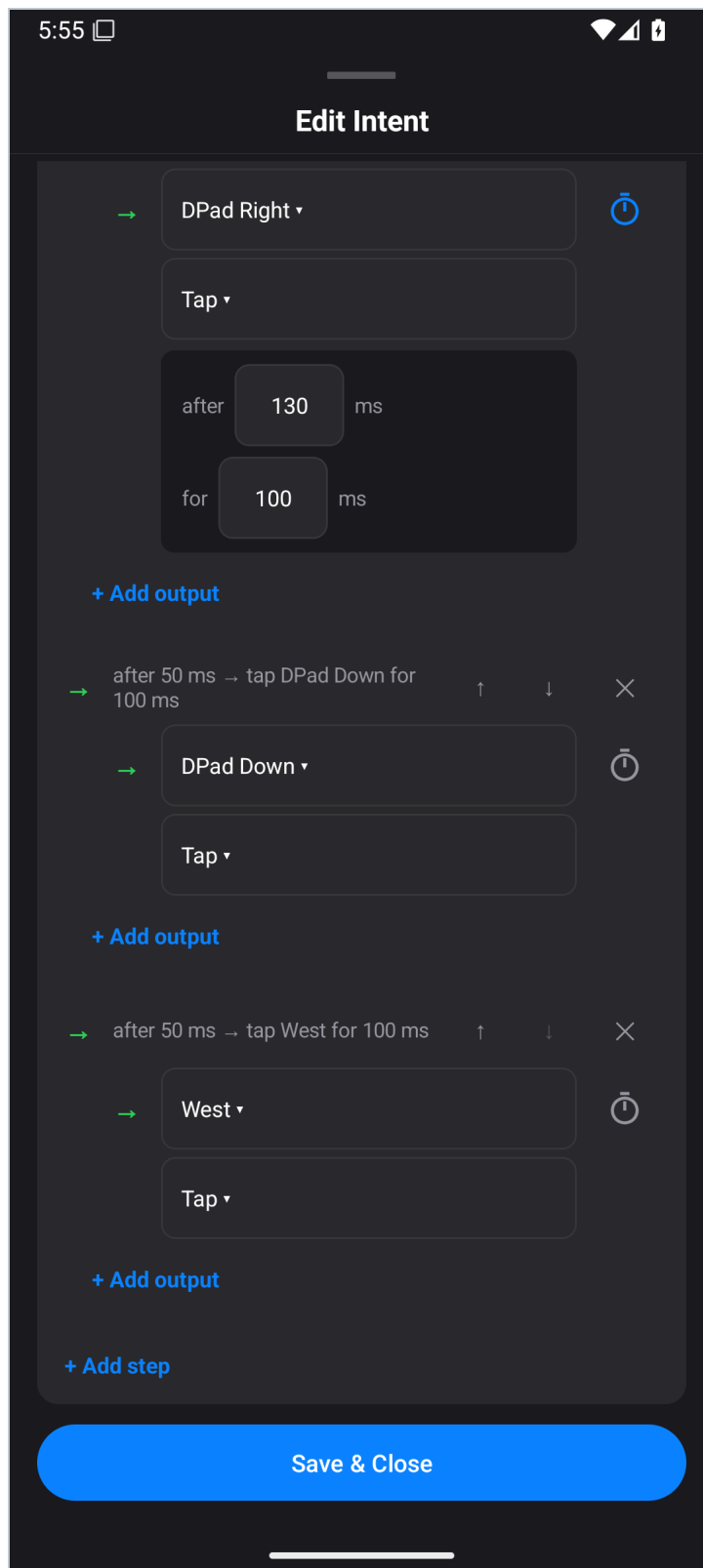


Figure 35. Scroll down to reach later steps and their timing controls.

How the sequence clock works

The first sequence step is timed from activation of the input. It can run alongside SEND. The word **THEN** does not make it wait for a held SEND action to finish.

After a step finishes, the next step begins. If a step contains several outputs, it finishes when the last of those outputs reaches the end of its timing window. Each following step's delay is measured from that step's own start.

Use a short **Tap** for a starting direction if the next direction should happen separately. A starting **Hold** could remain active underneath the rest of the sequence.

Each step's behaviour runs inside its timed window. A step is not a new physical press by the user. The current editor sets step timing like this:

Behaviour inside a step	Result within its for window
Hold	Active throughout the window
Tap	A single press for the length set by for
Rapid tap	Repeated presses within the window
Tap while held	Also repeats within the window; the sequence's held source controls whether the chain continues
On release	A pulse at the end of the window, normally the final 80 ms or less for a shorter window
Long press	Becomes active halfway through the window and stays active to its end

The step's **after** / **for** fields are its timing controls. The ordinary SEND hold-threshold and tap-rate fields are not separate step fields. For a straightforward button sequence, use **Tap** steps and check their gaps.

Keep the source held for the whole sequence

Releasing the source early cancels the sequence. Keep a switch pressed until the last step has finished. A brief tap does not start a sequence that always runs to completion.

A lost input connection, an unmet modifier, or a stop action can also interrupt it. Repeated taps and simple tap pulses have their own behaviour; do not assume the sequence uses those same release rules.

A sequence still follows priority rules for each output. Another higher-priority mapping can block one of its actions. The whole sequence is not treated as one indivisible action.

AUTO-RELEASE IF STUCK

This is an optional guard for an output that appears stuck. It is especially useful for a look or turn axis that could keep spinning after a control problem.

1. In **Edit Intent**, turn on **AUTO-RELEASE IF STUCK**.
2. Set **Release after** in seconds.
3. Tap **Save & Close**.

4. On the home screen, open the menu and **Auto-release safety**.
5. Check **Enable auto-release**. This master setting must also be on.
6. Review the listed guarded mappings.

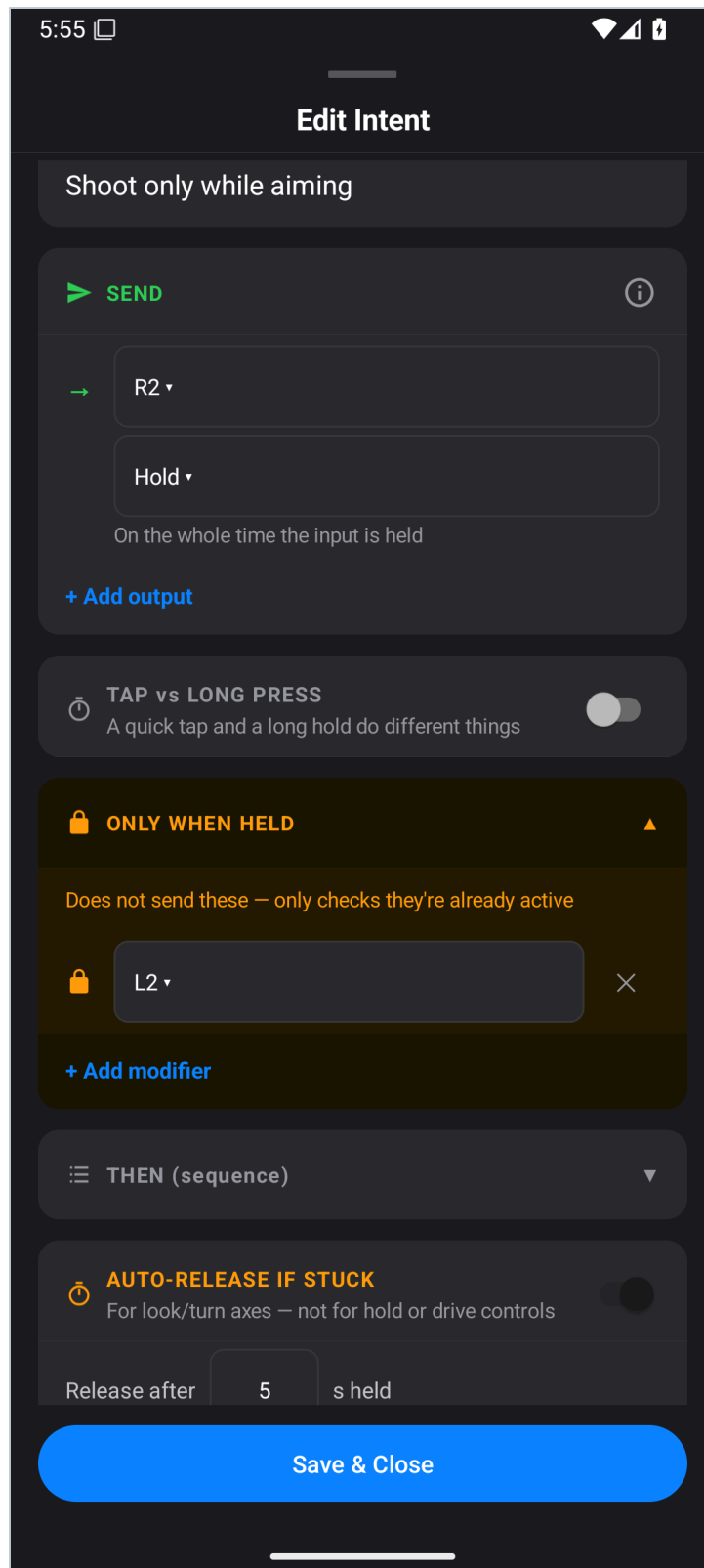


Figure 36. The threshold belongs to this Intent. The home-menu setting also controls whether guards are enabled.

The guard releases a control when its sustained output meets the stuck condition for the set time. Analog guarding considers sustained, unchanging movement. It is not a promise to stop every kind of unwanted action.

Release the source or return it to neutral before trying again after a guard trips.

Leave this off for controls that must stay held for a long time, such as accelerating, moving forward, or hold-to-aim. Otherwise an intended long hold can be interrupted. The screenshot shows where the option is; the aiming example would normally leave it off.

Safety actions in the target list

The output target list also includes **Safety Enable**, **Release All**, and **Stop Mapping**. These are app actions rather than normal game buttons.

Target	What it does	A useful setup
Safety Enable	Once activated in a session, allows ordinary output while the enable control remains active	A held enable switch with Hold behaviour
Release All	Releases current outputs while keeping the session available	A separate reset or release button
Stop Mapping	Stops the active output flow and releases controls	A large, easy-to-reach stop switch

For a stop switch:

1. Create a simple **Switch** mapping on the source you can reach easily.
2. Tap **Edit Intent**.
3. Set the SEND target to **Stop Mapping**.
4. Choose **Hold**, so pressing it acts immediately.
5. Leave modifiers and sequences off for that mapping.
6. Save it, start a test session, and confirm the switch stops output.
7. Return home and use **Stop**, then **Start Mapping** when you want a fresh session.

Release All is useful for clearing outputs without choosing a different profile. Release the source controls before continuing. It is not the same as ending the session.

Safety Enable is activated by the first enable signal seen during a session. Merely adding the mapping does not guarantee that all output is blocked from the moment the session starts. Test by holding it, releasing it, and checking that ordinary actions are blocked afterward. It resets with the session and also reacts if the enable signal stops arriving. Use this as an app convenience, not as a substitute for hardware safety equipment.

Priority and handover

When two inputs ask for the same output at the same time, the app needs to choose one. This is why input order matters.

The rules you need to know are:

1. Inputs higher on the home screen have earlier priority.
2. Within an input, mappings earlier in its list have earlier priority.
3. An idle source does not keep pressing a control just because it is first.
4. Different output controls are handled separately.
5. A lower-priority source may need to return to neutral before taking over from a higher-priority source.

For example, a helper's controller and your joystick can both control the left stick. If the helper takes control and then lets go while your joystick is already tilted, the app may keep the affected output neutral until you centre and move your joystick again. This helps avoid a sudden jump to an old held position.

If you see **Return control to centre to continue**, release the affected button or centre the stick, then deliberately activate it again.

A trigger's digital and analog forms can control the same target. Two mappings to R2 are not independent just because one comes from a switch and one comes from a sensor. Similarly, left and right movement share one stick axis.

The engine checks whether input signals are valid and still arriving. When one board disconnects, that board's controls are cleared. Other working inputs can remain useful, subject to handover checks. If the failed board is also your output bridge, the destination connection is affected too.

Use the **Control Map** to inspect conflicts rather than adding duplicate mappings until something works.

What the current editor does not decide for you

The editor sends controller actions. It does not read the game screen, identify an opponent, choose a finishing move, or know when a menu is open.

There is no game-aware condition such as “only fire when an enemy is visible”. **ONLY WHEN HELD** checks controls, not game state.

The current behaviour list does not offer a general tap-to-latch toggle or a general app-launch command. For aim that lasts while a switch is pressed, use **Hold**. Do not look for an unlisted “toggle aim” behaviour in this editor.

Some tuning controls vary by input editor. Face controls have their own advanced settings. Board cards offer their current source, target, and calibration controls. A setting described for one input is not necessarily present for every other input.

Worked Intent examples

Example 1: aim and shoot with one button

Use this if pressing aim and fire together is difficult. First check the game's controls. This example assumes **L2 = aim** and **R2 = fire**.

1. Open the input you want to use, such as an ESP32 foot switch.
2. Add or open its **Switch** mapping.
3. Select the physical source port, key, or button.
4. Tap **Edit Intent**.
5. Name it **Aim and shoot**.
6. In **SEND**, set the first row to **L2 → Hold**.
7. Tap **+ Add output**.
8. Set the new row to **R2 → Hold**.
9. Leave **TAP vs LONG PRESS** off.
10. Leave **ONLY WHEN HELD** empty and **THEN (sequence)** empty.
11. Leave auto-release off if long aiming holds are intentional.
12. Tap **Save & Close**, then **Done**.
13. Tap **Start Mapping** and test in a practice area.

What you do	What the app sends
Press the switch	L2 and R2 become active together
Keep holding	Both remain active
Let go	Both release

See [Figure 31](#) for the completed editor.

This presses the game's aim and fire controls. You still aim with the game's aiming control, such as the right stick. It does not automatically line up a shot.

Variation: repeated fire while aiming. Keep L2 as **Hold** and change R2 to **Tap while held**. Start at a slow rate such as 5 taps/sec. Use this only if the game benefits from repeated presses. For a weapon that already fires continuously while held, ordinary R2 Hold may be better.

Variation: aim first, then fire. Some games need time to aim before shooting. Keep **L2 → Hold** in **SEND**. Put **R2** in a sequence step with a delay and duration that you have tested, such as after 200 ms for 100 ms. Keep the source held through the step. Increase the delay if firing starts too soon.

Example 2: only shoot while another input holds aim

Use this when you want separate aim and fire controls, but want to prevent fire when aim is not held.

1. Make one input send **L2** → **Hold**.
2. Make a second mapping send **R2** → **Hold**.
3. Open the second mapping's **Edit Intent**.
4. Name it **Shoot only while aiming**.
5. Open **ONLY WHEN HELD**.
6. Tap **+ Add modifier** and choose **L2**.
7. Save the Intent and input editor.
8. Start mapping.
9. Hold aim, then press fire. Fire should work.
10. Let go of aim and check that the fire Intent no longer passes its requirement.

The fire Intent does not create L2. The first mapping supplies it. This is different from the one-button combination in Example 1.

Example 3: tap for one action, hold for another

Suppose South means jump and L3 means sprint in your game.

1. Open the source mapping's **Edit Intent**.
2. Name it **Tap jump, hold sprint**.
3. Set SEND to **South**.
4. Turn on **TAP vs LONG PRESS**.
5. Choose **L3** as the long-press output.
6. Set **Hold for** to **500 ms**.
7. Save and test.

Source movement	Result
Short press	South sends a short tap
Hold past 500 ms	South tapped at the start; L3 then stays active
Release after the hold	L3 releases

If you want sprint without that first jump, use a separate normal **Long press** action instead. See [Figure 32](#).

Example 4: a Mortal Kombat finishing-move sequence

Use a sequence when a finishing move needs several precise button presses and making them separately is difficult.

The pattern below is a **training example**, not a claimed Fatality for a particular character. Mortal Kombat moves depend on the game, character, move, distance, and controller settings. Get the real command from that game's move list, then replace the example controls.

Prepare

1. Open a training or practice mode that lets you test the required move.
2. Find the finishing move in the game's move list.
3. Note every direction, button, and required distance.
4. Translate its buttons using the [button-name table](#).
5. Decide how long you can comfortably hold your source switch.

Back and **Forward** depend on which way your character faces. Intent Controller sends actual left and right controls. It does not automatically swap them when the opponent changes sides. You may need two named Intents or profiles for opposite directions.

Build the example

Our practice pattern is:

Down → Right → Down → West.

1. Open a source switch mapping and tap **Edit Intent**.
2. Name it **Training sequence**.
3. Set SEND to **DPad Down → Tap**. Use the normal **80 ms** tap.
4. Leave **TAP vs LONG PRESS** off and **ONLY WHEN HELD** empty.
5. Open **THEN (sequence)**.
6. Tap **+ Add step**.
7. Choose **Dpad Right → Tap**.
8. Open that row's clock control. Set **after = 130 ms**, **for = 100 ms**.
9. Tap **+ Add step** again.
10. Choose **Dpad Down → Tap**, **after = 50 ms**, **for = 100 ms**.
11. Add the final step: **West → Tap**, **after = 50 ms**, **for = 100 ms**.
12. Check the order and save.
13. Start mapping and hold the source for about one second while testing.

The intended windows for this example are:

Action	Starts after the source press	Ends after the source press
SEND: DPad Down	0 ms	80 ms
Step 1: DPad Right	130 ms	230 ms
Step 2: DPad Down	280 ms	380 ms
Step 3: West	430 ms	530 ms

The first 130 ms delay leaves a gap after the 80 ms SEND tap. Each later 50 ms delay starts after the previous step finishes. Actual delivery also depends on the output connection and the game; this is not a frame-perfect timing guarantee.

See [Figures 34 and 35](#) for the editor and timings.

Replace it with the real move

1. Change SEND to the move's first control.
2. Replace, add, or remove steps to match the move list.
3. Use ↑ / ↓ if the order is wrong.
4. Set the required gap and press length for each step.
5. Position the character at the required distance.
6. Hold the source until the whole sequence finishes.
7. If it fails, change one timing at a time and test again.

If releasing the switch is part of how you normally press it, remember that release cancels unfinished steps. Choose an input you can keep active long enough. Also check the Control Map for another mapping that may be taking priority over a direction or button in the sequence.

Example 5: a pressure sensor for movement

Use this for a sensor with two directions around a resting value, such as a sip/puff-style control.

1. Add a board **Centered Axis** mapping.
2. Choose the correct analog source port.
3. Rest the sensor and tap **Record**.
4. Set one side's action, such as a left direction, and the other side's action, such as right.
5. Check the neutral and active zones on the calibration bar.
6. Activate each side separately and watch the preview.
7. Save, start mapping, and test gently.

If rest triggers one side, correct the neutral value or active zone. A dual-axis stick needs two source channels; one centred sensor does not create two independent axes.

Check your Control Map

The Control Map helps answer two questions: “What drives this controller button?” and “What do my inputs do?”

1. On the home screen, open the three-line menu.
2. Tap **Control Map**.
3. Use **Controller** to inspect output controls.
4. Tap a button or stick in the diagram.
5. Read the contributing mappings and their order.
6. Use **What my moves do** to inspect the input side.
7. Tap a mapping when offered to open its editor.

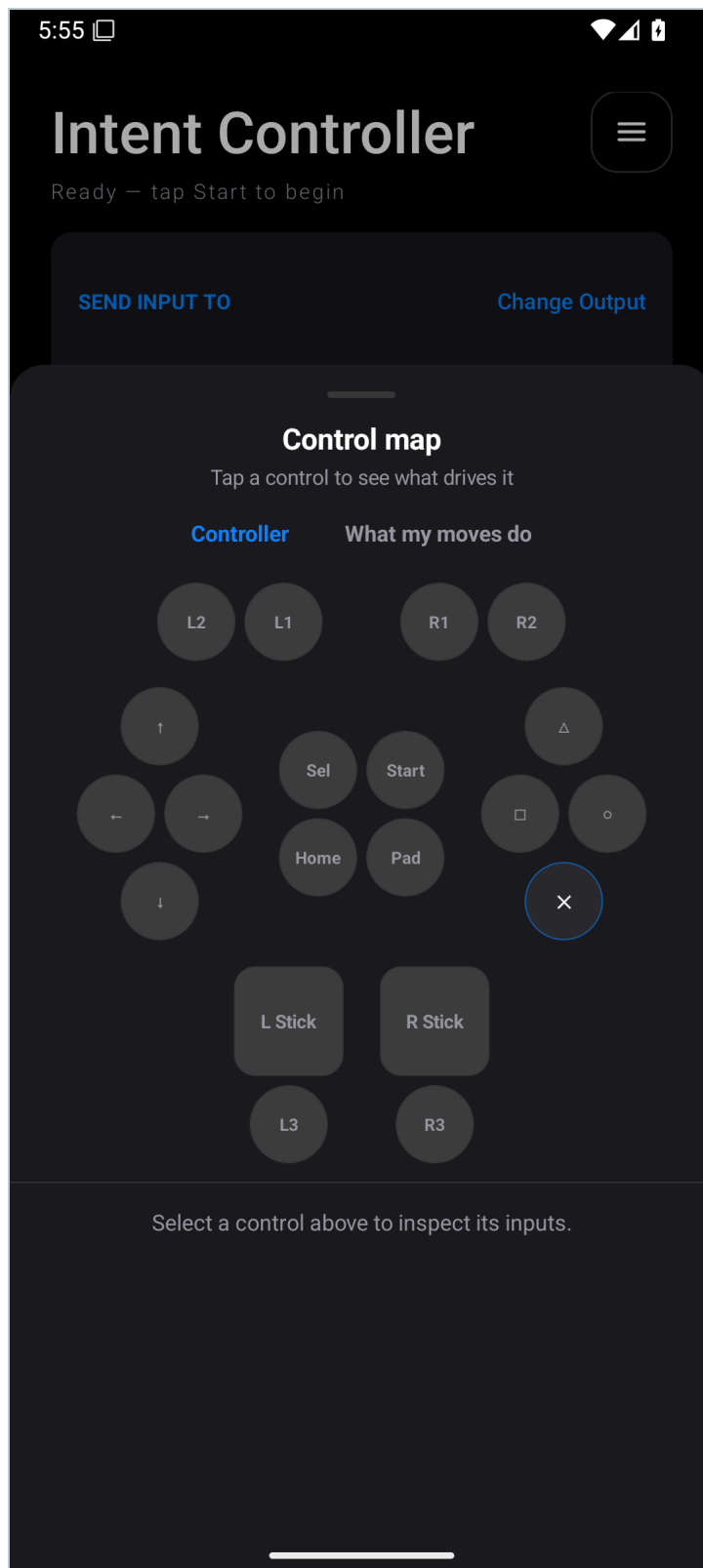


Figure 37. Tap an output control to see what is assigned to it.

Use this before a long play session and after importing a profile. Look for missing controls and unexpected overlaps. If your new fire mapping appears to do nothing, another mapping to R2 may have earlier priority.

Save, copy, and share profiles

A main profile stores a set of inputs, their mappings and calibration, and the output choice. Separate profiles are useful for different games, different people, or different positions for your equipment.

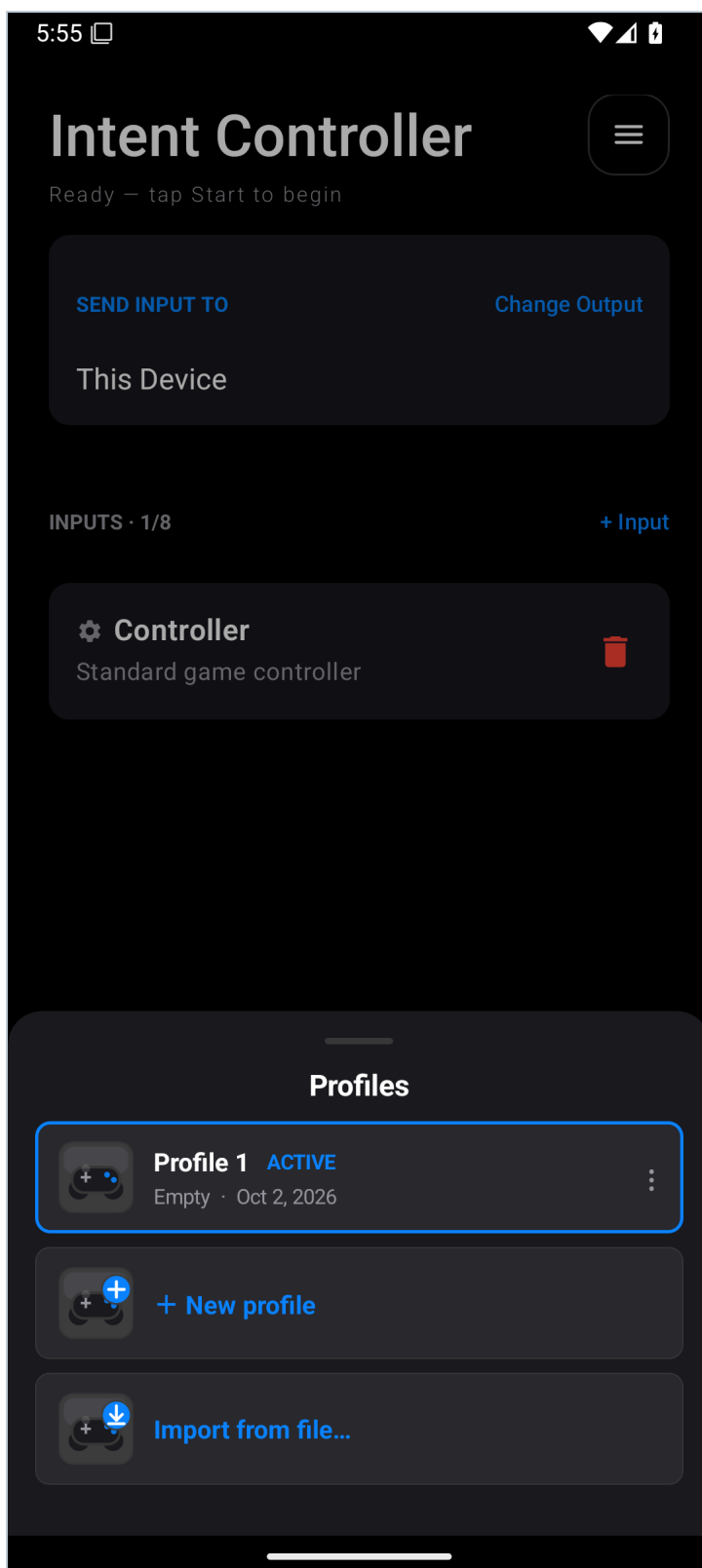


Figure 38. The active profile is marked **ACTIVE**. Use a profile row's three-dot menu for its file actions.

Create and switch profiles

1. Open the home menu.
2. Tap **Profiles**:
3. Tap **+ New profile**.
4. Enter a clear name, such as **Racing — chair controls**, and confirm it.
5. Tap the new profile's row to make it active. Creating a profile does not switch to it automatically.
6. Set up that profile's output and inputs.
7. To switch later, reopen **Profiles** and tap the profile you want.
8. Check the active label and connections before starting.

Switching profiles releases current output first. Treat it as a setup change, then start the required session again.

Rename, duplicate, or delete

1. Open **Profiles**.
2. Tap the **:** on the profile row.
3. Choose **Rename**, **Duplicate**, or **Delete**.
4. Complete the name or confirmation prompt.

Duplicate is useful before experimenting with an existing game layout. Check the copy's board connections before using it; you may need to select and verify physical boards again.

Delete removes the profile. Export a copy first if you might want it later.

Export and import

To export:

1. Open the profile row's **:** menu.
2. Tap **Export / Share....**
3. Save or share the `.icprofile` file with the Android chooser.

To import:

1. Open **Profiles**.
2. Tap **Import from file...** below **+ New profile**. Scroll within the sheet if needed.
3. Select the `.icprofile` file with Android's file picker.
4. Read any import result or error.
5. Select the imported profile.
6. Review its inputs, mappings, and calibration.

7. Select and verify your own physical boards.
8. Set up its output connection on your phone.

An exported profile carries portable mapping information. It does not carry a working Wi-Fi connection or console login to another phone. Board names and mappings can be preserved, but board addresses and physical identifiers are removed on import, and the board must be verified again. Output connection settings are not exported as a ready-to-use connection.

An import over the eight-input limit is rejected. If a profile came from another person, recalibrate face tracking and check sensor ranges for your own controls.

The virtual controller's **Layout Profiles** are separate. Saving a touch layout does not export the main input profile.

Start again next time

Everyday start checklist

1. Turn on the receiving computer or console.
2. Power your boards and other inputs.
3. Connect USB devices or enable Bluetooth as needed.
4. For Wi-Fi boards, connect the phone to the network that reaches them.
5. Open Intent Controller.
6. Check the active profile in the home menu.
7. Check the output and every board card's status.
8. Use **Connection** or **Continue setup** for any card that needs attention.
9. Release switches and centre sticks.
10. Tap **Start Mapping**.
11. Test one action before entering a busy game.

You normally do not re-enter the board's Wi-Fi password, install firmware, or rebuild mappings every time. Those are setup or repair tasks.

If a board's network address changes

1. Tap **Connection** on that board's card.
2. Try **Reconnect saved board** or **Find boards on this network**.
3. Select the same physical board and verify it.
4. Tap **Save connection**.
5. Keep its existing mappings and test them.

For an output board, use **Change connection** in its output settings. Verify the applied identity too if you replaced the board.

If you change Wi-Fi networks

Use that board's network setup to send the new network name and password. After it restarts, join the same reachable network on the phone and find the board again. Repeat for each board that needs the new network.

If you unplug and reattach USB

Reconnect the cable and accept USB access if asked. If the saved device cannot be identified reliably, use **Choose USB board** and select it again. Check the board name so you do not bind the wrong card to another attached device.

Finish a session

1. Use **Stop** in the app's active mapping screen.
2. Check that the game stops receiving held controls.
3. Then disconnect or power down your equipment.

The app uses its own active mapping and virtual/Remote Play screens. Do not assume input continues unchanged after moving to an unrelated Android app or leaving the mapping screen. Return to Intent Controller and check the session state.

Fix common problems

Problem	What to check and press
Start Mapping is unavailable	Read the home status. Choose an output, add a mapped input, and complete required board verification or identity application.
The board is listed but never verified	Discovery or USB detection is not enough. Check that the board runs compatible firmware and sends valid readings. Try Connection details .
A new board's Wi-Fi says "no internet"	Stay connected during setup. Return to the app and tap Connect to setup board . Join home Wi-Fi after sending the network details.
No boards found on home Wi-Fi	Check board power, phone network, and network access between devices. Guest networks can isolate devices. Use Enter board address for a known address.
Board disappeared after Send and reconnect	Let it restart, use Open phone Wi-Fi settings , join home Wi-Fi, then Find board after restart .
Wrong board's controls react	Check that card's name and Connection details . Select the right device and verify by moving its physical control.
One of several boards is offline	Repair that card with Connection . Use Retry connections ; choose Start with available inputs only if you can play without the missing controls.
USB device is missing	Check OTG support, a data-capable cable, device power, and Android's USB permission. Reopen the device chooser.
Controller, keyboard, mouse, or joystick is not in the input list	Connect or pair it with Android before using the app's selector. Check that Android recognises the correct device type.
Live input changes but the game does nothing	Check the mapping's target, selected output, active session, host connection, and Control Map. The editor preview alone does not start output.
Selecting a controller identity does not finish board output setup	Tap Apply to ESP32 or Apply to Pico W and wait for verification. On Pico W, finish with Confirm when ready.
Board works with the phone but the console rejects it	Check the board-to-host connection and supported controller protocol. Identity fields alone do not supply all console compatibility features.
A sequence stops halfway	Keep the source active until it finishes. Check connection loss, modifiers, stop controls, and priority conflicts.
A sequence's first direction overlaps the next	Use Tap for the starting SEND action and leave a large enough delay before the first step.
A long hold unexpectedly sends the quick action	TAP vs LONG PRESS sends its first tap on pressing. Use ordinary Long press if you need no starting tap.

Problem	What to check and press
Repeated taps continue after release	You may be using Rapid tap . Choose Tap while held if release should stop repetition.
An action with a modifier never starts	Hold every required control first. Check for a circular requirement or an analog control that has not crossed its active level.
The app asks you to return a control to centre	Release or centre the affected source, then activate it deliberately again. This is a handover check.
Aim or acceleration stops during a long hold	Check that AUTO-RELEASE IF STUCK is off for an intentionally held action.
Face tracking does not see you	Allow camera access, check the chosen camera, improve lighting and placement, then calibrate.
Face actions trigger accidentally	Recalibrate neutral, raise the relevant threshold, or increase Hold Frames. Change one setting at a time.
A sensor moves at rest	Record its neutral position and check its calibration range and active zones.
Phone Bluetooth gamepad is not visible	Start mapping, allow discoverability and Bluetooth permissions, then search from the receiving device. Check Bluetooth gamepad support.
PS console pairing fails	Check console type, same network, Remote Play enabled, matching account ID, and a current pairing PIN.
Xbox console list is empty	Check the Microsoft account, console remote features, and internet connection. Sign out and back in if you selected the wrong account.
Xbox sign-in is unavailable in this build	Use a build configured for Xbox sign-in. This is not a board network setting.
Imported profile does not connect	Imported mappings still need local board selection, output setup, and suitable calibration.

When asking someone for help, note the app screen, exact status message, selected output, input type, and which connection is failing. For boards, use **Connection details**. Describe whether live readings change. Do not send your Wi-Fi password or account password.

Button names and useful words

Translate controller button names

The app uses neutral names so a mapping can move between supported outputs. Check the game and host for the exact interpretation.

App target	PlayStation label	Xbox label	Common Nintendo layout label
South	Cross / ×	A	B
East	Circle / ○	B	A
West	Square / □	X	Y
North	Triangle / △	Y	X
L1 / R1	L1 / R1	LB / RB	L / R
L2 / R2	L2 / R2	LT / RT	ZL / ZR
L3 / R3	Left/right stick click	Left/right stick click	Left/right stick click
Select	Share / Create, depending on controller	View	Minus
Start	Options	Menu	Plus
Home	PS button	Xbox button	Home
Touchpad	Touchpad button	No direct equivalent	No direct equivalent
DPad Up / Down / Left / Right	Direction buttons	D-pad	Direction buttons
Left Stick / Right Stick	Analog sticks	Analog sticks	Analog sticks

Nintendo labels are shown to explain button positions. They do not add a separate Nintendo output mode to the app. The host's accepted controller protocol still decides compatibility.

Where offered, **Gyro** represents rotation and **Accel** represents acceleration. These motion targets currently have a supported route through PS Remote Play. They are not ordinary direction buttons.

An axis selector may show **Left Stick X**, **Left Stick Y**, **Right Stick X**, or **Right Stick Y**. X is left/right; Y is up/down. These select one component of a stick. **L2** and **R2** cover trigger output in the current selectors; older saved profiles may use names such as **L2-Analog** for the same trigger channel.

Glossary

Word	Meaning in this guide
Input	A device or signal you use to control something
Output	The destination receiving the finished controller actions
Mapping	The link from a source control to its action
Intent	A named mapping instruction with actions, behaviours, and optional conditions or sequence
Source	The key, switch port, gesture, or axis being read
Target	The controller control or app action being sent
Axis	One line of movement or amount, such as left/right or trigger pressure
Dual Axis	Two axes, usually X and Y, used together for a stick
Neutral	Rest: a released button or a centred control
Calibration	Teaching the app the rest position and useful range of a sensor or movement
Threshold	The amount needed before an input counts as active
Deadzone	A small range near rest that is ignored
Modifier	Another control that must already be active
Sequence	An ordered set of timed action steps
Priority	Which active mapping gets first choice when several want the same output
Profile	A saved set of input mappings and an output choice
Controller identity	Hardware name and ID details used by a board output
IP address	A device's address on a network
MAC address	An identifier used to help recognise a particular network device
OTG	Android USB host connection used to attach a USB device
Firmware	The software installed on an ESP32 or Pico W board
ms	Milliseconds; 1,000 ms equals one second
Beta	A feature still being developed and tested

For a first successful session, connect one input, map one action, connect one output, and test it. Build the rest of your controls from that working setup.