

Design and Build of a Custom H-Pattern Gearbox for Sim Racing

Date: 04/26 | **Time spent:** approx. 10 hours | **Estimated cost:** £20

1. Project Summary

I designed and built a working five-speed-plus-reverse H-pattern gearbox for racing games on my PC. It has a hand-built plywood housing, a wooden gimbal-style pivot for the lever, six microswitches that sense which gear is selected, and a Raspberry Pi Pico that reads the switches and appears to the computer as a USB game controller. I use it with my steering wheel in Assetto Corsa and Euro Truck Simulator 2, and it responds to every gear change. I built it for about £20 and used the project to teach myself soldering, wiring a microcontroller and programming in CircuitPython.

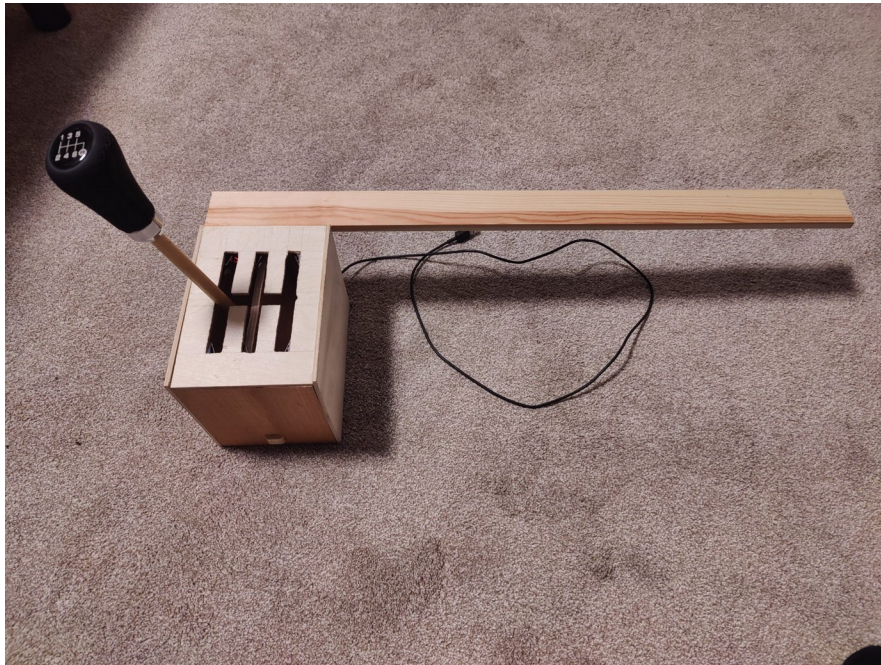


Figure 1: The finished shifter. The box is glued to a long timber batten that slides under my desk, so my wheel clamp holds both.

2. Why I Did This Project

Commercial H-pattern shifters were too expensive for me, so I decided to build my own and use the project to learn the skills I wanted to develop.

- **Motivation:** I play sim racing games such as Euro Truck Simulator 2 and Assetto Corsa and wanted a manual shifter to go with my wheel. I also wanted a reason to learn how to solder and how to use microcontrollers.
- **Problem to solve:** Existing H-pattern shifters were too expensive. I also had a mounting problem: my desk is thin, so my wheel's clamp could not close fully, and any shifter needed a way to be fixed in place that dealt with this.
- **What I wanted to learn:** Soldering, using a Raspberry Pi Pico, and programming it to work as a USB game controller. I also picked up CAD, woodworking and mechanism design along the way.

3. Design Brief and Requirements

Requirement	Target / measure	Result
Detect 5 forward gears + reverse + neutral	One dedicated switch per gear. Neutral when no switch is pressed. Two gears never active at once.	Met. Each gear registers as its own input in the Steam controller detector and in Assetto Corsa. The layout means two switches cannot be pressed together.
Free, realistic H-pattern movement	Lever moves forward/back and side to side freely so it can be placed into each gear.	Mostly met. Moves freely and sits in every gate after the redesign. It does not self-centre, so moving between gates takes care (feedback from my brother).
Self-centring lever (original aim)	Springs return the lever to the neutral position.	Not met. Springs proved impractical, so I removed them from the design (section 6).
Connect to PC as a game controller	Recognised as a USB HID game controller, one button per gear.	Met. Detected in the Steam controller detector and works in Assetto Corsa.
Sturdy and mountable	Fixed in place by my wheel's clamp, even with my thin desk.	Met. The L-shaped batten sits under the desk and the wheel clamp holds both the wheel and the shifter.
Budget	Far cheaper than a commercial H-pattern shifter.	Met. About £20 in total.

4. Research

I started by looking at other people's DIY builds to see how they solved the mechanism, the sensing and the USB connection.

- **Other DIY builds:** Looking at DIY shifters showed me the core idea: a lever that can move on two axes through a gate, with one sensor per gear position.
- **Sensing options:** I chose microswitches because they are cheap and give a simple on/off signal, the lever can press them directly, and I could solder wires to the normally open (NO) and ground terminals. Analogue options such as Hall-effect sensors or potentiometers would have needed magnets, calibration or analogue readings, which added complexity I did not need.
- **Making a Pico act as a USB game controller:** I first tried MicroPython but found it impractical for making the Pico appear as a USB game controller. I then discovered that CircuitPython's `usb_hid` module lets the board define its own USB HID device, so I switched to CircuitPython.

5. Design Development

5.1 Concepts considered

Concept	Pros	Cons	Decision
A: 6 forward gears + reverse gate, cut from thicker wood (modelled in CAD)	Full 6+R layout.	I had not allowed for the thickness of the wood, so the lever could not move forward and back in the outer gates.	Rejected after test-fitting.
B: Springs to self-centre the lever	More realistic feel, returns to neutral like a real gearbox.	Impractical to fit into this mechanism.	Dropped.

Concept	Pros	Cons	Decision
C (chosen): 5 forward gears + reverse gate in thinner balsa, gimbal pivot, one microswitch per gear	Lever moves freely in every gate. Simple, reliable sensing with no calibration.	No self-centring, and one fewer forward gear than the original plan.	Selected because it fixed the clearance problem and kept the design simple.

5.2 Mechanical design

- Lever pivot:** The lever is a wooden dowel held in a two-axis, gimbal-style pivot. The dowel passes through a small inner box. A shorter piece of dowel inside the inner box lets the lever tilt forward and back. The whole inner box can also rotate freely in a hole on either side of the main box, which gives the left-to-right movement. The two movements are independent, so the lever can reach every gate position.

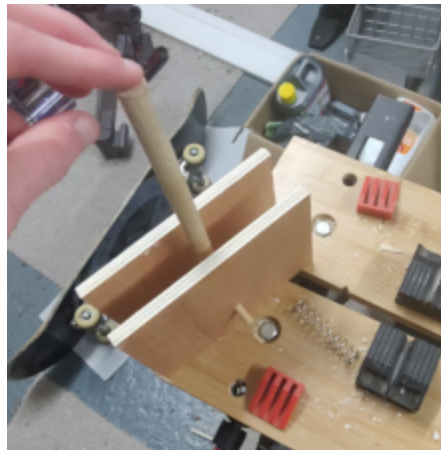


Figure 2: The inner box during early development. It carries the lever and controls the forward/back movement. The whole box rotates left and right inside the larger box.

- H-gate:** I modelled a 6+R pattern in CAD software and cut it out with a coping saw. The lever tilts as it moves, so a thick panel reduces the clearance it has. Because I had not accounted for the wood's thickness, the lever would not slot forward and back in the outer gates. I redesigned the gate as a 5+R pattern and remade the top panel from thinner balsa, which fixed it (see section 6).

- **Dowel for the shaft:** Dowel is cheap, easy to cut to length, strong enough for the small forces involved, and round, so it turns smoothly in drilled holes. I used it for the lever and for the pivots.
- **Shift knob:** The shifter knob was bought off amazon and it screws into the end of the dowel. The dowel was 12mm which is the standard diameter for shifter knobs allowing ti to fit without any issues.
- **Stability and mounting:** I considered several options and chose to glue the whole box to a long timber batten, supported by a 45° brace. From above this forms an L-shape. The batten goes under my desk so my wheel's clamp holds both the wheel and the shifter. This also solved the problem of the clamp not closing fully on my thin desk.
- **Materials:** Plywood for the housing, balsa for the revised gate panel, wooden dowel, a timber batten, and glue.
- **Tools and techniques:** CAD for the gate design, a coping saw for cutting, a soldering iron, a crimping tool for the connectors, and hot glue for fixing the switches.

5.3 Electronics

Microswitch wiring (one switch per gear)

*Pressed: switch connects the GPIO pin to GND.
Released: the Pico's internal pull-up holds the pin high (3.3 V).*

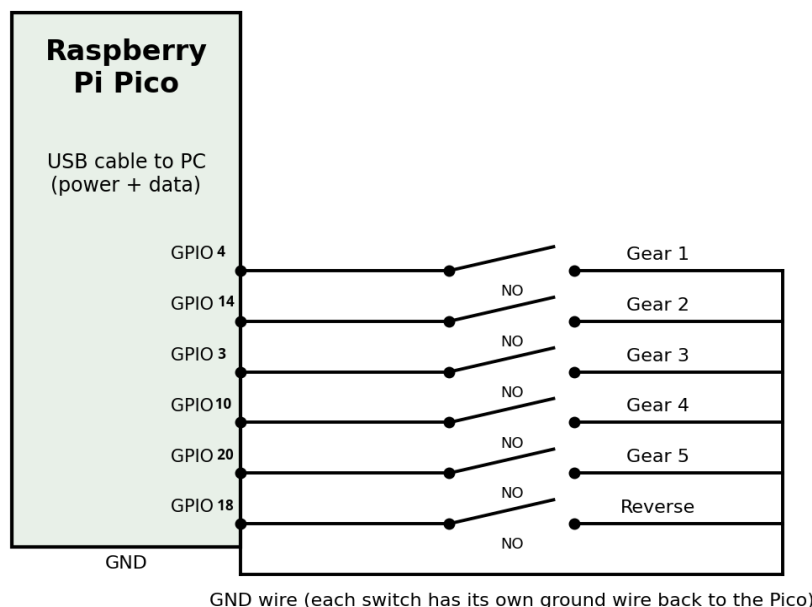


Figure 3: Wiring schematic.

- **Components:** Raspberry Pi Pico, six microswitches (one per gear), hook-up wire, self-crimped jumper cables and a USB cable.
- **How gear position is detected:** Each microswitch is hot-glued at the end of its gate, sticking out slightly. When the lever is in a gear it rests against that switch and holds it pressed. Moving out of the gear releases it. The design means only one switch can be pressed at a time.
- **How components connect to the Pico:** Each switch has its own GPIO pin and its own ground wire, soldered to the switch's NO and ground terminals. I chose the GPIO pins based on where they sit on the board, to keep the routing simple. Because each switch connects its pin to ground when pressed, the input reads as active-low, and the Pico's internal pull-up resistor keeps the pin high when the switch is released. This means no external resistors were needed.
- **Removable connections:** I crimped my own jumper cables between the switches and the Pico. This means I can remove the Pico without taking off the top panel or unsticking the glued switches.
- **Power and connection to PC:** A single USB cable powers the Pico and carries the game controller data, so there is no separate power supply. The circuit only uses the Pico's low-voltage 3.3 V logic.

5.4 Software

- **Language and libraries:** CircuitPython, using the `usb_hid` module to make the Pico appear as a game controller. I tried MicroPython first but it was impractical for this.
- **Logic:** Each of the six switches is mapped to a game controller button (gears 1 to 5 and reverse). While a switch is held down, the matching button is held. When none are held, no button is pressed, which is neutral.
- **How it appears to the computer:** As a USB game controller with a button for each gear. I bind these buttons to the gear controls in the game.
- **Handling edge cases:** A switch has to be held down for a set time (in milliseconds) before it counts as pressed, and has to be released for a set time before it counts as released. This debouncing stops brief blips from flicking the controller in or out of gear. Neutral is

simply the state where no gear is selected, and a gear only stays active while its switch is actively held down. The mechanical design prevents two switches being pressed together.

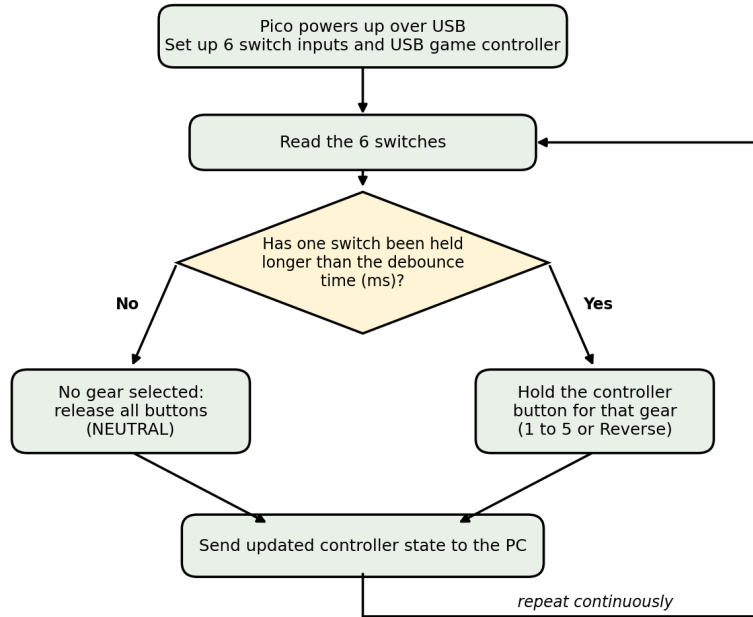


Figure 4: Flowchart of the shifter logic.

Code excerpt:

```

while True:
    now = time.monotonic() * 1000 # ms
    for pin, button_num in PIN_TO_BUTTON.items():
        if is_pressed(pin):
            if pressed_since[pin] is None:
                pressed_since[pin] = now
            elapsed = now - pressed_since[pin]

            # debounce: only act once press has been stable a bit
            if elapsed >= DEBOUNCE_MS and not button_down[pin]:
                gp.press_buttons(button_num)
                button_down[pin] = True
        else:
            if button_down[pin]:
                gp.release_buttons(button_num)
                pressed_since[pin] = None
                button_down[pin] = False

    time.sleep(POLL_INTERVAL_S)
  
```

6. Iterations and Problem Solving

Version	What I built	What went wrong / what I learned	What I changed
Prototype 1	Gate panel for a 6+R pattern, designed in CAD and cut by hand with a coping saw.	I had not accounted for the thickness of the wood, so the lever could not move forward and back in the outer gates.	Decided to redesign the gate rather than force it.
Prototype 2	A 5+R gate cut from thinner balsa. I reworked the top gate several times to get it right.	A thinner panel gives the tilting lever more clearance. I learned that material thickness must be part of the design.	Moved to the 5+R layout in thinner balsa, which fixed the problem.
Springs (dropped)	Planned springs to self-centre the lever like a real gearbox.	Springs were impractical in this mechanism.	Dropped the idea. The lever now stays where it is placed.
Electronics build	Switches and wires soldered to the Pico, six GPIO pins and six grounds.	I kept getting cold joints, which gave unreliable connections.	Switched to a better soldering iron. Added crimped jumper cables so the Pico can be removed without undoing the joints.
Final version	5+R balsa gate on a gimbal pivot, six microswitches, Pico on removable leads, L-shaped batten mount.	Moving between gates is a little difficult without self-centring or angled corners (my brother's feedback).	Not changed yet. Ideas are in section 9.

Key problem I solved: The first gate would not let the lever move forward and back in the outer gates. The cause was that my CAD design treated the gate as a flat shape and I had not accounted for the thickness of the wood. A tilting lever needs more clearance through a thick panel than a thin one, so it jammed. I redesigned the gate as a 5+R pattern and cut it from thinner balsa. I checked the fix by moving the lever through every gate, and it now moves forward and back freely in all of them. The lesson I took from this is to include material thickness in my designs from the start.

A second problem: I kept getting cold solder joints, which made connections unreliable. I switched to a better soldering iron, which fixed it.

7. Testing and Evaluation

- **How I tested it:** I used the Steam controller detector to confirm that each gear shows up as its own input, and I played Assetto Corsa to test it in use.
- **Results against the requirements:** All the core requirements in section 3 are met except self-centring, which I dropped because it was impractical.
- **What works well:** Each gear is detected as its own input. Debouncing stops false gear changes. The Pico can be removed without damaging the wiring. The mount is stable and also solved my wheel clamp problem. The cost was about £20.
- **What I would still improve:** Self-centring, and a gate with angled or rounded corners so it is easier to move between gates.
- **Feedback from others:** My brother tried it and found moving through the gate a little difficult because of the lack of self-centring and the right-angled corners.

8. Skills and Knowledge Demonstrated

Area	Evidence from this project
Mechanical design and manufacture	Modelled a gate pattern in CAD, cut it by hand with a coping saw, designed a two-axis gimbal pivot and an L-shaped mount with a 45° brace.
Electronics	Soldered six switches to NO and ground terminals, with one GPIO pin each. Understood active-low switch inputs with pull-up resistors. Crimped my own jumper cables so the Pico is removable.
Programming	Wrote CircuitPython to read six inputs and present them as a USB game controller using usb_hid, with debouncing and neutral handling.
Problem solving	Found the cause of the gate jamming (wood thickness) and fixed it. Solved unreliable joints by improving my equipment. Switched from MicroPython to CircuitPython when it could not do the job. Solved the clamp problem with the mounting design.
Iterative design	Revised the gate several times, moved from 6+R to 5+R, and dropped the springs after finding they were impractical.

Area	Evidence from this project
Planning and budgeting	Built the shifter for about £20 instead of buying a commercial one, and worked on it in short sessions over nearly a year (about 10 hours in total).
Safety and working practice	Soldered in a large, well-ventilated space, used only low-voltage USB electronics, and used hand tools carefully.

9. Reflection and Next Steps

- **What I learned:** How to solder reliably, how to use a microcontroller, and how a mechanical design depends on real material dimensions. I also learned that stopping to redesign is better than forcing a flawed design to work.
- **What I would do differently:** Include material thickness in my CAD model, make a quick cardboard mock-up of the gate before cutting wood, and log photos of each stage.
- **Future improvements:** Add self-centring with a different mechanism, make the gate corners angled or rounded, restore the sixth forward gear, 3D print the gate and pivot, build a full CAD model, and replace the loose wiring with a PCB.
- **How this connects to engineering:** This project brought together mechanical design, circuit building and embedded programming in one product, which is the mix of skills I want to keep developing in electrical engineering. It also took me through the full cycle of specifying, building, testing and improving a system.