Retro

FUTURISTIC

ENGINEER

USB UNIVERSAL SERIAL BUS

RetroARDU Input

The three-headed Monkey

Inhalt

1 Introduction

The USB-RetroARDUInput is a project based on the Arduino Mega2560 to translate modern USB input devices to old standards.

The Swiss Army knife among retro tools makes it possible to always have the right input devices at hand.

The inspiration for this comes from Adrian Black (Youtube: Adrian's Digital Basement), who developed a similar converter for PS/2 to Tandy 1000 based on an Arduino. His code (link in chapter Sources) is also the one on which the Tandy output of RetroARDUInput is based and the one on which I developed the other keyboard protocols by changing timings and control behavior. The well-known USB Host Shield Library from felis is probably used by everyone who works with the Arduino as USB host. A few more code snippets, especially in the ADB area, were taken from other open source projects, and a lot of own work was invested.

Considering the basic problem that old input devices are getting rarer and rarer, some USB input devices did not want to run properly with older PCs with USB and PS/2 (in advance: Since gaming keyboards and mice have some special features that make them incompatible with the Arduino USB Host Shield, this dilemma was not solvable that way), there was also the dream of the "one for all" KVM with an HDMI/USB KVM switch, an HDMI monitor and USB input devices, which with the help of adapters like the RGB2HDMI, VGA to HDMI, MCE2VGA, RetroTink and just for the input devices the RetroARDUInput make every computer from the early 80s to today controllable via the same keyboard, the same mouse, the same joystick and the same monitor.

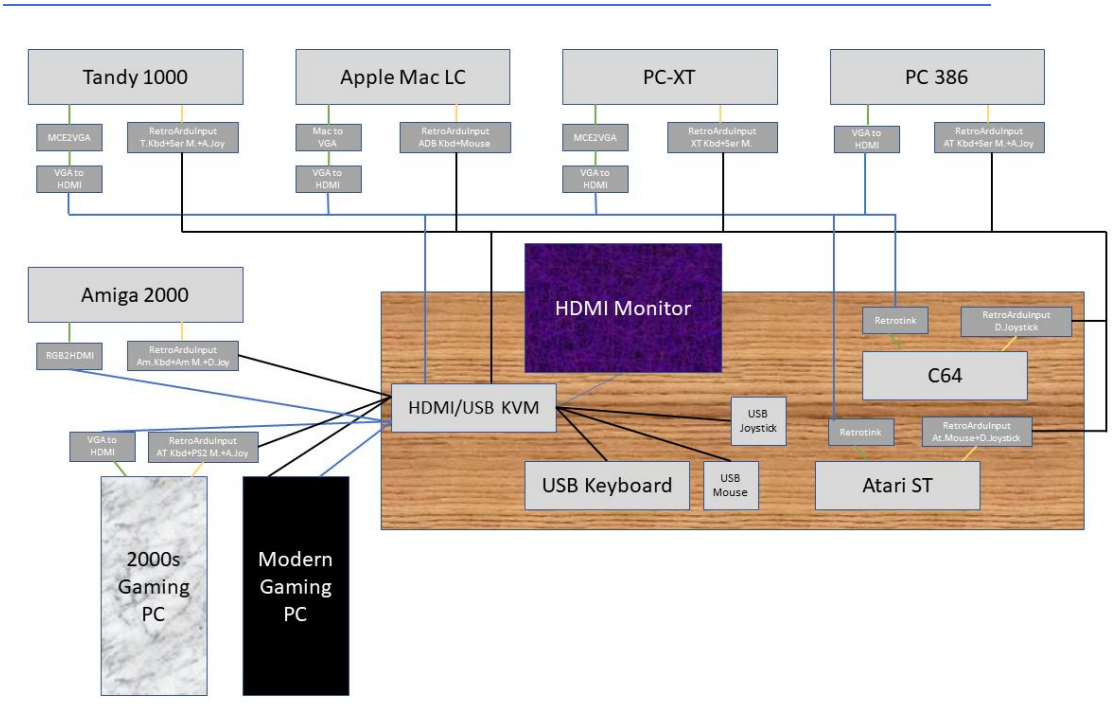

Figure 1: The Vision – One KVM for Everything!

Whether this will ever work, or whether the limitations of the USB host shield will prevent this, is an open question.

But even so, the device should be helpful in any situation. Thus, the following retro devices are emulated and output:

- Keyboards:
  - PC/XT
  - PC/AT und PS/2 incl. Scancode-Sets 1 (XT-Style), 2 (AT Default) und 3 (some PS/2) as well as Typematic Rate and control of the keyboard LEDs, PS/2 via adapter cable DIN->Mini-DIN
  - Amiga 500/2000/3000, where it is unlikely that somebody will adapt the internal keyboard port of an Amiga 500 to the DIN-plug, but for the Desktop-Amigas 2000 und 3000, this is a good option
  - Tandy 1000 (thanks to mechanical Pinout-switching without adapter)
  - Apple (via ADB)
- Mice:
  - Serial mouse with 2-Button Microsoft-Mode and 3-Button Logitech-Mode
  - PS/2 with 3-Button-Mode (Scaling and Sample Rate currently ignored)
  - Amiga / Atari (Electric Pinout switching)
  - C64 1350 Joymouse (just for fun), currently no 1351 Proportional Mouse
  - Apple (via ADB)
- Joysticks:
  - Analog-Joystick for PCs (Poti-output open)
  - Analog-Joystick for Tandy TRS-80 CoCo, Tano/Dragon and BBC Micro (Poti-Output grounded, mechanical switching), requires adapter cable from PC-Joystick-Port, also for the BBC Micro, which has the same connector but with a different Pinout
  - Commodore/Atari Digital-Joystick, incl. consideration of a „Dead Zone“ until the axis trigger when operated with an Analog-Joysticks

This ought be enough for everyone, just like the famous 640kB.

Joystick broken? No problem, a USB joystick now works on any PC, Amiga or C64. Whether analog joystick or the good old Competition Pro, everything is now available as USB device and can thus be connected to any device.

Bought an IBM XT, Tandy 1000 or desktop Amiga at the flea market and don't have a suitable keyboard or mouse at hand? No problem, just translate a USB keyboard to fit.

The USB-RetroARDUInput consists of three components. The Arduino Mega2560 itself, the USB Input Shield and the so-called Three-Headed Monkey Shield. Unmistakably the allusion to the three-headed monkey from "The Secret of Monkey Island", fitting that a keyboard, a mouse and a joystick symbolize each of the three heads. Even though each head speaks several dialects and thus also has several ports.

The case design in my personal color scheme is meant to be a small homage to IBM's first PCs, the 5150 PC, 5160 PC-XT and 5170 PC-AT. These have a light gray front, a medium gray lid (top/side) and a black back panel with white lettering. Therefore, I decided to use the following colors when 3D printing the case:

- Medium gray (IBM case lid): Geeetech PLA “Gray” (Medium Gray)
- Light gray/white (IBM Frontpanel): Amazon Basics PLA Nature Gray (Light Gray)
- Black (IBM backside): Geeetech PLA Black
- Rot (IBM Switches): Geeetech PLA Red, alternative 3DFils TPU Red for Soft-Touch Switches
- Transparent (Accentuation, Optional): Geeetech PLA Transparent

This is not perfect but already very close to the IBM colors and fits very well with various computers from the "off-white" 1980s. Since I have the front with transparent base with engraved lettering (solid light gray front panel with recessed lettering that shows through transparently), the lettering will glow, fueled by 6 LEDs, which are of course green, as befits power LEDs from the 80s and early 90s.

Parts are printed:

- Base plate: Medium gray
- Top-Cover und right Cover: Base Medium Gray, switch to black for the labelling
- Rear Cover: Black, change to Light Gray for labelling
- Left cover (the unavoidable recess for the USB host port due to its design, based on the switch recess on IBM's right side): Start with Red for the cover in the USB port's area, change to black for the inscription of the USB-Port and the frame of the recess, change to Medium Gray for the outer cover, change to Black for the labelling
- Front Cover „engraved/debossed": Start with Transparent, if the LED-lighting shall be usedor Black, if there is no lighting intended, then change to light gray
- Front Cover „embossed ": Start with light gray, change to Black for the labelling (transparent not foreseen). I do not recommend to use this type of front cover with a transparent base, as the light distribution might be quite uneven.
- Slider-Switch-Cover, Reset-Switch Pin und Reset-Switch Sleeve: Red

The housing was created in FreeCAD and is sliced with Cura. Printing is done on a Creality Ender 3 V2.

The well-known China PCB manufacturers, who mostly also offer 3D printing, print in SLA technology, which, unlike FDM, does not allow color changes. In addition, they usually only offer a light gray. Those at home with an FDM printer can in principle use any color scheme, favorably but with a larger time investment (depending on the part, 1 to 8 hours of printing time).

In case someone is wondering about the order of the chapters, and why the operation relevant chapters still come before the assembly, although you actually have to assemble the device first: If there are enough people in the community who build finished devices for colleagues with less soldering knowledge, the assembly chapter is certainly less interesting for most people and was therefore positioned after the chapters "Configuration", "Known Limitations", "Compatibility List" and "Special Key/Axis Mappings". These are more interesting for everyday use. For the nerds among us, who want to assemble the device themselves, as a retro nerd should, the chapter "Material list", "Assembly" and "Programming" follows, and those who also want to learn something about the interfaces, for example to actively work on the community project and further improve the product, get the "Information about the interfaces" as a conclusion. Finally, since a lot is based on my own brainpower, but not everything, a reference to other projects, from which I have taken some code.

Note: The project started in the middle of 2020 and besides a vision and necessity also a bit as a distraction from a certain boredom in the "Corona-Lockdown". Moreover, I saw it as a great chance to refresh my somewhat rusty knowledge in hardware and software development after almost 15 years in a job without these skills and to explore new fields like PCB design and 3D printing. Since my job fortunately took up a lot of time during the lockdown, there wasn't always much free time and so the development time stretched over several years. In the meantime the idea was caught up by another project called "USB4VC" based on the Raspberry Pi, currently with PC/XT and PC/AT/PS/2 keyboard and serial and PS/2 mouse as well as analog joystick on a PC module and ADB on an Apple module. But since open source community projects do not compete but complement each other, and I still see it as a learning project for me as well as a different approach (Atmega microcontroller on Arduino instead of ARM-SOC on RasPi) and good addition (more and different protocols), I continue -

and here is now the preliminary result, as a community banana (actually finished, but needs to mature in the community).
The parallel project „USB for Vintage Computers“ or short „USB4VC“ by dekuNukem:
On Github: GitHub - dekuNukem/USB4VC: USB Keyboard/Mouse/Gamepads on Retro Computers!
On Kickstarter: USB4VC: USB Inputs on Retro Computers! by dekuNukem — Kickstarter

2 Connectors

The device has the following connectors:

On the Arduino:

- USB Type B for programming, power supply and debugging
- Power: Power Supply – To be preferred over USB in operation, especially when USB-Hubs are used. Polarity Positive Pin/Negative Sleeve. The Arduino has a voltage regulator and usually demands 7-12V via this port. It runs with 5V, but the detection only starts at 7V. Below that it is supplied via USB, if a USB programming cable is connected (usually sufficient, but possibly not always) or parasitic voltages, as they creep in here e.g. via the reference voltages in the joystick port (definitely not sufficient!).

On the USB Host Shield:

- USB Type A – One or more USB devices can be connected here (via USB hub). However, only one device of one type should be connected at a time, multiple devices of the same type such as two joysticks are not supported. Exception: A USB composite HID, usually wireless keyboard/mouse sets with only one receiver dongle, can be connected together with a mouse and keyboard. The devices then compete with each other and provide both inputs.

On the „Three-Headed Monkey“ Shield:

- -Analog joystick (PC): Here the standard X/Y axis is transferred as the X/Y axis of joystick 1 and any thrust controls or 3D function (rotate stick) as the X/Y axis of joystick 2. For the buttons (button 1 and 2 as button 1 and 2 of joystick 1 and 3 and 4 as 1 and 2 of joystick 2) the same applies. This was also implemented in early joysticks with more than two axes or more than two buttons, e.g. CH Products Flightstick / Flightstick Pro. The digital potentiometers can be switched to ground (e.g. BBC Micro, Tandy TRS-80 or Dragon with suitable adapter) or operated open (standard with PC).
- Serial mouse (PC): Switchable between 2-button Microsoft and 3-button Logitech protocol
- Digital Joystick (C64, Amiga, Atari): The port can be switched to listen to the USB mouse instead of the USB joystick. This is to emulate the Commodore 1350 "Joystick Mouse" for the C64. An implementation for the 1351 "Proportional Mouse" is currently not planned and would probably require a major redesign.
- Amiga/Atari Maus: Maus basierend auf den Radimpulsen der optomechanischen Impulsgeber alter Kugelmäuse. Umschaltbar zwischen Commodore (Amiga, PC1, PC10-III/PC20-III/Colt) und Atari ST (hat einige Pins getauscht)
- Keyboard: 8-pin DIN connector to allow full compatibility with all systems. With an 8-pin cable for the Tandy 1000 and a 5-pin cable that mechanically fits into this connector as well, for IBM PC-XT, IBM PC-AT and PS/2 (mechanical adapter from DIN to Mini-DIN required) and Commodore Amiga from the Desktop/Tower series.
- PS/2 Mouse: Standard PS/2 Mouse
- ADB: For older Apple computers (Macintosh Classic / Color Classic, LC, Performa, early Power Macs and similar)

3 Configuration

3.1 Explanation

The device has the following switches:

- SW01-SW04 and SW05: SW01 and SW04 mechanically switch the pin layout of the keyboard connector. In position 1-2 the pins are aligned for Tandy 1000. In position 2-3 for PC (XT/AT/PS/2), Amiga and Apple (output via ADB). The pin layout of the Tandy 1000 differs so much from the PC standard that switching in the Arduino was not possible (5V and ground lines). SW05 switches between Tandy and PC/Amiga protocol. A large 3D pressure lever encompassing all five switches should help to ensure that all switches are always flipped simultaneously. If SW05 is in position 1-2 (Tandy), SW06 and SW07 are ignored.
- SW06: PC/Amiga+Apple switch: If SW01-SW05 are in position 2-3 (PC/Amiga) this switch selects between Amiga/Apple (1-2) and PC (2-3). SW07 then selects between PC-XT/AT and Amiga/Apple
- SW07: XT/AT switch: Switches in PC mode (SW01-SW05 and SW06 in position 2-3) between PC-XT (position 1-2) and PC-AT and PS/2 (position 2-3) or in Amiga/Apple mode between Amiga (position 1-2) and Apple ADB (position 2-3), depending on whether PC or Amiga/Apple mode is preselected at SW06. In ADB mode the DIN keyboard port is disabled and ADB outputs keyboard and mouse. In any other mode the DIN port outputs keyboard signals of the selected system and ADB remains disabled.
- SW08: Serial Mouse Mode: 1-2 (3 Button Logitech) and 2-3 (2-Button Microsoft)
- SW09: Digital Joystick Mode: 1-2 (C64 Commodore 1350 Joystick Mouse) and 2-3 (Digital Joystick for C64, Amiga and Atari). The port does the same, but in 1350 mode it listens to the mouse and not the joystick.
- SW10: Amiga/Atari Mouse: 1-2 (Atari, some pins swapped, happens in the Arduino) and 2-3 (Commodore Amiga, Commodore PC1/PC10-III/20-III/COLT)
- SW11: Debug: Position 1-2 Debug On, waits after power on for the serial UART of the debugging console of the Arduino development environment and then outputs messages, 2-3 Debug Off, device initialized (takes a few seconds) and is then ready for use even without serial console. No output via the serial console
- SW12-SW15: Joystick potentiometer pulled to ground (position 1-2) for compatibility to devices without resistor at the input (Tandy TRS-80, Dragon) or open at the end (position 2-3) for devices with resistor at the input (PC Joystick Interface).
- Reset: Here a hole was left for the reset button on the USB host shield, through which a 3D printed button can be inserted, in order to be able to operate the reset in an emergency also from the outside.

3.2 Tabular switch overview

3.2.1 Keybord

Concerns: CN5 – Keyboard – DIN-5/180 or DIN 8/270 and CN7 – ADB – Mini-DIN-4
1 = On/Right, 0 = Off/Left, / = switch ignored

Big switch Tandy					PC/A*	AT/XT		
SW01	SW02	SW03	SW04	SW05	SW06	SW07	Mode	Connector
0	0	0	0	0	0	0	AT, PS/2 (with Adapter	DIN-**5**/8
0	0	0	0	0	0	1	PC/XT	DIN-**5**/8
0	0	0	0	0	1	0	Apple	ADB
0	0	0	0	0	1	1	Amiga	DIN-**5**/8
1	1	1	1	1	/	/	Tandy 1000	DIN-5/**8**

Note: Only Tandy 1000 uses pins 6 to 8 of the 8-pin DIN connector, the pins located there are not relevant, use of a solder pin compatible 5-pin DIN connector is possible. The arrangement of pins 1-5 is different on the Tandy 1000 than on the pin-compatible IBM PC/XT, AT, PS/2 (with adapter DIN to Mini-DIN) and desktop Amigas.

Using the Tandy pinout on another system or the PC/Amiga pinout on the Tandy will not work and in the worst case can damage the hardware. Therefore, SW05 (protocol) and SW01-SW04 (pinout) must always be set equally to 1 when using a Tandy 1000.

In Apple mode, the memory area for the scan code table is occupied for the Apple scan codes and the keyboard commands are issued via ADB and the ADB mouse is also activated. The DIN port remains without function here. In any other mode, the keyboard commands are issued via the DIN port. The ADB mouse remains inactive in the PC/Amiga/Tandy mode.

3.2.2 Serial Mouse

Concerns: CN2 – PC Serial Mouse 9-Pin D-SUB Female
1 = On/Right, 0 = Off/Left

SW08	Mode
0	Serial mouse using Microsoft 2-Button protocol
1	Serial mouse using Logitech 3-Button protocoll

3.2.3 Digital Joystick Port

Concerns: CN3 – Commodore/Atari Joystick or C64 1350 Joystick Mouse – 9-Pin D-SUB Female
1 = On/Right, 0 = Off/Left

SW09	Mode
0	Joystick – Uses joystick commands (with Deadzone for analog joysticks) for output to digital joystick
1	1350 – Uses USB mouse commands for digital joystick (C64 „1350 JoyMouse“)

3.2.4 A-Mouse

Concerns: CN4 – Commodore/Atari Mouse – 9-Pin D-SUB Female
1 = On/Right, 0 = Off/Left

SW10	Mode
0	Commodore-Pinout. Compatible to Commodore Amiga, COLT, PC-1, PC10-III, PC-20-III, PC-30-III
1	Atari-Pinout. Compatible to Atari ST Series

3.2.5 Analogue Joystick

Concerns: CN1 – Analogue Joystick – 15-Pin D-SUB Male
1 = On/Right, 0 = Off/Left

Switch Joystick 1		Switch Joystick 2		
SW12	SW13	SW14	SW15	Mode
0				Potentiometer output open for: - PC (native pinout, no adapter)
1				Potentiometer output to GND for: - BBC Micro (with Adapter, different D-SUB 15 Pinout) - Tandy TRS-80 CoCo (with Adapter) - Tandy 1000 (with Adapter) - Tano/Dragon (with Adapter)

Since the switch only makes sense per joystick axis (or for the entire joystick port), it was considered to either relocate all 4 switches (discarded) or at least the 2 switches for the X and Y axis of a single joystick (i.e. SW12+SW13 and SW14+SW15), so switch caps with overlap for 2 switches each were installed in the case

3.2.6 Debug-Mode

1 = On/Right, 0 = Off/Left

SW11	Mode
0	Off: No initialization of USB serial port, no output of debug information over the serial port using serial.println
1	On: Initializes the USB serial port and waits for communication with the Arduino IDE. Provides debug information over serial.println, as long as the programmer has foreseen them

Since the switch is not relevant for everyday use, it was not routed to the outside via a switch cap. The right casing wall, where the keyboard connections are located, has to be removed for switching.

Waiting for the USB UART "Serial" can cause critical timings to diverge. The idea was therefore to link the global variable debugMessage to the switch position and always query it via if command before actions are performed on the UART (establish connection, Serial.println) and to be able to conveniently switch between stable everyday operation and debugging via messages by this switch.

4 Known limitations

The following are known to have limited or no functionality:

- USB hubs: Do not work in every combination. Power supply for the Arduino via the 7-12V barrel jack (polarity: positive pin/negative sleeve) instead of USB programming port and use of a USB hub with its own power supply recommended. Incompatibilities with certain hubs are limitations of the library, maybe a few hubs have to be tried.
- Mice: The library does not support reading the mouse wheel, so it is also not possible to pass it to PC mice, although both PS/2 and even serial in Logitech mode would allow it.
- Gaming input devices with controllable RGB lighting: These do not always behave like a normal USB HID. This can lead to the USB library not recognizing them. These devices are also not 100% compatible to older PCs, e.g. the Corsair K68 keyboard does not work with the Arduino USB Host Shield, but also boards like the Asus P5KR (Socket 775) do not react in the BIOS, even if the keyboard is set to BIOS mode. More about this in the compatibility list.
- Joystick: Since currently no automatic detection works (it is being worked on, but the limitations of the host shield library hinder more than help) and in the current revision no manual switching is provided, to change the joystick the firmware has to be changed (JoyType in globals.cpp based on the #define values in globals.h) and uploaded again.
- Joystick: Each joystick has a different mapping or even different resolutions. It is tried to implement as many joystick types as possible. If a joystick is not available and does not work with the default joystick template, it must be implemented accordingly.
- Due to the limited memory of the Mega2560, two scan code mappings cannot be used simultaneously. Therefore, the use of an Apple ADB keyboard must be explicitly selected. The DIN keyboard port is then deactivated. Consequently, the ADB mouse also runs only in Apple keyboard mode to save computing power and to leave the ADB libraries completely disabled in any other mode..
- The keyboard modes XT, Tandy 1000 and Amiga do not provide for bidirectional communication between host and input device. Therefore the Arduino can only guess the status of the Caps Lock, Num Lock and Scroll Lock LEDs. If there is no 5V supply at the keyboard port, the LEDs are switched off (computer off) to start with the correct status, in case of crashes with soft reset the computer may have a different lock status than the keyboard. But this is normal, "real" keyboards for these computers have the same problem.

5 Compatibility list

- Input devices:
  - USB Hubs:
    - In general: Some Hubs will only function reproducible if one of the connected devices is a USB wireless receiver of a Logitech keyboard/mouse set (here: K270), some don't work at all
  - Compsite HIDs (Keyboard/Mouse Sets, typically wireless with a receiver dongle):
    - Logitech K270 Wireless Keyboard/Mouse PASS
  - Keyboards:
    - Corsair K68 (Gaming keyboard with red lighting, also in BIOS-Mode) FAIL
    - Microsoft Wired Keyboard 600 PASS
  - Mice:
    - Logitech G502SE HERO (Gaming mouse with RGB) FAIL
    - Microsoft Basic Optical Mouse PASS
    - Logitech MX500 PASS
  - Joysticks:
    - Speedlink Phantom Hawk (Incl. calibration due to bad factory centering) PASS
    - Speedlink Competition Pro USB (sometimes unreproducible outages where the joystick stops working, but sometimes it runs flawlessly for hours, maybe a defective joystick) PASS
    - Trio Linker Plus (Playstation 1+2, Dreamcast, Gamecube -> USB)
      - Playstation 2 Dual Analogue Controller PASS
      - Dreamcast Controller PASS
      - Gamecube Controller (Cannot be tested for lack of a controller, but I assume the same mapping as for the PS2) Partial Pass
    - Logitech Extreme 3D Pro PASS
    - Thrustmaster TC Sidestick Airbus Edition (likely also FCS Hotas) PASS
    - Logitech F-Serie Gamepads (sometimes there are connection issues with the F710 Wireless, but this also works, wired gamepads like the F310 use the same mapping) – Attention, switch on the gamepad has to be on D (DirectInput), X (XInput) will not work PASS
- Retro-PCs to be controlled:
  - Keyboard Interface:
    - IBM PC/XT class:
      - Commodore PC20-II PASS
    - IBM PC/AT und PS/2 class:
      - Headland HT12 286: LEDs in CheckIt not properly set, otherwise okay Partial Pass
      - KMC-A419-3 386: LEDs in CheckIt not properly set, otherwise okay Partial Pass
      - Tyan Trinity S1590: LEDs in BIOS and CheckIt not properly set, otherwise okay Partial Pass
    - Commodore Amiga Desktop/Tower und Amiga 500 with adapter to PCB connector:
      - All keys working, Reset (Ctrl+Amiga+Amiga) and self test/handshake implemented, but A2000 specific things cannot be tested with the A500 PASS (Amiga 500)/Untested (Amiga 2000)

        - Tandy 1000:
            - Adaptation of PS/2 to Tandy by Adrian Black fully implemented, currently no possibility to test it Untested
    - Serial Interface (Mainboard/mouse driver):
        - In 2-Button Microsoft Mode:
            - Commodore PC20-II / Cutemouse PASS
            - Headland HT12 286 / Cutemouse PASS
            - KMC-A419-3 386 / Cutemouse PASS
        - In 3-Button Logitech Mode:
            - Commodore PC20-II / Cutemouse PASS
            - Headland HT12 286 / Cutemouse PASS
            - KMC-A419-3 386 / Cutemouse PASS
    - PS/2 Mouse Interface (Mainboard/mouse driver):
        - Tyan Trinity S1590: Works with Logitech driver but not with Cutemouse and Windows Partial Pass
    - Analogue Joystick:
        - ISA cards (5V reference)
            - Creative Labs Soundblaster 2.0, Aztech Sound Galaxy NXII (CPS Soundblaster 2.5, various Multi-I/O cards) PASS
        - PCI cards (3.3V reference)
            - Untested
        - Home Computers with interface requiring potentiometer on GND
            - Untested
    - Commodore/Atari Mouse:
        - Amiga 500 PASS
        - Atari ST (PASS? Untested due to lack of device, just pinout change)
    - Commodore/Atari Joystick:
        - Joystick-Mode (reacts to USB joystick):
            - Amiga 500 PASS
        - 1350 „Joymouse“ Mode (reacts to USB mouse):
            - Amiga 500 as Joystick, no C64 Software available PASS
    - Mac Classic etc. (ADB Keyboard and Mouse):
            - Implementation pending

6 Special Key-/Axis -Mappings

6.1 Keyboards

PC keyboards are PC keyboards, regardless of the interface, and the layout does not differ between PS/2 and USB. There are no special features here. With the XT, the right Alt and Ctrl keys were not intended, so there is a double use of the scan codes (left and right keys on the USB keyboard use the same XT scan code). Otherwise, there are no special features.

It gets more special with the Amiga and Apple.

6.1.1 PC to Amiga

Some keys of the Amiga do not exist on the PC keyboard and vice versa. Therefore, the most practical assignment possible was chosen to be able to map all keys.

- Modifier keys:
  - Alt-Left und Alt-Right mapped 1:1
  - Ctrl can be triggered over both Ctrl-keys of the USB keyboard
  - Closed A (left Amiga key): Left Windows key
  - Open A (right Amiga key): can be triggered via right Windows key or Context Menu Key, as some keyboards don't have the right Windows key
- Upper number row:
  - Backslash/Pipe left of the Backspace key remapped to „Insert"-key on the Inverse-T Pad
- Del/Help:
  - Del remains on the Delete key of the Inverse-T Pad
  - Help: Remapped to Pause/Break
- 10-Key Pad:
  - Open Bracket to PgUp key of the Inverse-T Pad
  - Closed Bracket to PgDn of the Inverse T-Pad

6.1.2 PC to Apple

Some keys of the ADB keyboard do not exist on the PC keyboard and vice versa. Therefore, an assignment as close as possible to practical use was chosen in order to be able to map all keys.

- Power-Key:
  - Power: Was mapped to Print Screen/System Request. The Apple Scancode for Print Screen/F13 is mentioned in the source code comments, in case somebody wants to map this key here and use the Power-Key of a multimedia keyboard instead.
- Further F-Keys:
  - Scroll/F14 = Scroll
  - Pause/F15 = Pause
- Modifier keys:
  - Option-Left = Alt-Left
  - Option-Right = Alt-Right /AltGr
  - Command Left/Right = Control Left/Right
  - Apple Left/Right (mirrored on the Apple keyboard) = Windows Left/Right
- 10-Key Pad
  - =-Key = Windows context menu (as there is no requirement to mirror the Right Apple to here)

6.2 Joysticks

All joysticks have two axes and two buttons. The digital joystick port for Amiga/Atari does not use more than X/Y and keys 1 and 2. On the PC at some point the possibility moved in to use the axes and buttons of joystick 2 for up to 4 joystick axes and buttons. In the last days of the analog gameport there were also codings of more than 4 buttons. However, I didn't implement these because they are rather typical for the Windows 95 era and rather difficult to use in DOS games. Therefore, the countless keys that today's joysticks offer are strategically redirected to be used ergonomically and logically as keys 1-4 of the analog joystick in any hand position.

All implemented joysticks are in my possession. I aim to expand the collection in the future and also add 8-Bit-Do gamepads or older USB joysticks like Logitech Wingman or Microsoft Sidewinder and of course accept source code extensions from other community members with new joysticks.

For example, whether it makes sense to implement a Microsoft Sidewinder that can be used with a 15-pin joystick port and USB is debatable. However, if we get to the point of "one KVM for everything" implementation and someone wants to use the Sidewinder on everything from the latest gaming PC to the oldest retro device via their USB KVM switch, then of course we need a Sidewinder implementation should it differ from the standard joystick. In any doubt, it would be an idea in regards oft he „One KVM for All“ Setup to even think about implementing the Joystick-to-USB Adapter and so the opposite way by Necroware (GitHub - necroware/gameport-adapter: GamePort adapter to connect old DB15 joysticks to USB port), if it requires special implementation beyond the standard profile and so e.g., be able to use an old Gravis-Joystick in the all-computers-from-all-eras Setup.

6.2.1 Speedlink Competition Pro USB

Mapping "Straight Forward": the keys are in the traditional position above "Up", the keys are from left to right large round keys 1 and 2 and small square keys 3 and 4.

Figure 2: Speedlink Competition Pro

6.2.2 Speedlink Phantom Hawk

The Phantom Hawk is poorly centered at the factory and must be calibrated. This is done automatically for the X, Y and Twist axis when the Phantom Hawk profile is selected.

The Phantom Hawk has its second button as the lower trigger button for the little finger below the index finger trigger button 1. Since I find this unintuitive in retro terms - button 2 on a flightstick has to be operated with the thumb - , I have reversed the buttons. The trigger for the little finger is now button 3 and the thumb trigger on the back of the stick button 2. The hat at the top left is ignored, the central 2-way switch and buttons in the joystick base were each placed on buttons 1-4, with a limitation that the stand button at the top left also reports as button 1 via USB and thus has a part of the mapping predefined.

X and Y are mapped to X1 und Y1, Twist is X2 and Throttle Y2.

Figure 3: Speedlink Phantom Hawk

6.2.3 Logitech Extreme 3D Pro

The Logitech Extreme 3D Pro has a higher resolution and therefore needs its own profile.

The Logitech Extreme 3D Pro is mapped quite directly. The hat is ignored. Buttons in the joystick base were again assigned to 1-4.

X and Y are mapped to X1 und Y1, Twist is X2 and Throttle Y2.

Figure 4: Logitech Extreme 3D Pro

6.2.4 Logitech F-Serie Gamepads (F310, F710) Direct Input

For the Logitech F310 and F710, the mode switch must be set to D (Direct Input). The X (X-Input) mode does not work.

The Mode button toggles the behavior of the D-pad and left analog stick. If the Mode LED is off, the left analog stick is used and the D-pad is ignored, if the Mode LED is on, the D-pad is used and the left analog stick is ignored. The right analog stick always works analog.

The keys are arranged in a circle 1, 2, 3 and 4, starting with X (blue)=1, A (green)=2, B (red)=3 and Y (yellow)=4. In addition, buttons 1, 2, 3, and 4 are replicated by the index finger/middle finger triggers on the front of the pad (not shown in the photo), with buttons 1 and 3 on the left (1=index finger button, 3=middle finger trigger) and 2 and 4 on the right (2=index finger button, 4=middle finger trigger).

Figure 5: Logitech F710

6.2.5 Thrustmaster TC Sidestick Airbus Edition (maybe also FCS Hotas)

The Thrustmaster TC Sidestick has a higher resolution and therefore needs its own profile.

The Thrustmaster TC sidestick is quite directly assigned to buttons 1-3, whereby the orientation of buttons 2 and 3 depends on the left/right switch of the joystick. The stick imitates the Airbus Fly-By-Wire system's sidestick, which is located on the left side of the Captain's seat (left) and on the right side of the First Officer's seat (right). The side buttons are adapted to the seat positions, a fact that this joystick takes into account with switchable logic as well as interchangeable side buttons. The hat is ignored. Buttons in the joystick base have been put back to 1-4. The presumably also compatible FCS Hotas has 4 buttons on the stick, these should be correctly connected through (not tested).

The Airbus sidestick also activates another button when the throttle is fully retracted (for flight simulators: Activate Thrust Reverser), but this button is not evaluated.

X and Y are mapped to X1 und Y1, Twist is X2 and Throttle Y2.

Another plus in terms of retro: Absolute oldschool gamers, who still come from the era before the wild Microsoft Sidewinder and Logitech Wingman joysticks, might not like the twist axis. This can be locked mechanically on the Airbus Sidestick, as ambitious hobby pilots would probably rather fly with a real thrust lever set and pedals than abuse the simple throttle and twist. Thus, the Thrustmaster is the only modern USB flight stick that can still be used classically without a twist axis.

Figure 6: Thrustmaster TC Sidestick

6.2.6 Trio Linker Plus with Playstation 2 Dual Analogue Controller

With the Playstation 2 controller, the assignment of the axes depends on whether the analog mode is activated.

When analog mode is turned off (LED off), only X1/Y1 are controlled, using the D-pad with values at full stop. In analog mode, the left stick does X1/Y1 and the right stick does X2/Y2.

Keys 1 to 4 are mapped to the fire keys on the right, clockwise from the top, with 1 on the green triangle, 2 on the red circle, 3 on the blue X and 4 on the magenta square. In addition, the keys are on the index finger triggers, on the lower larger keys 1 (left) and 2 (right) and on the upper smaller keys 3 (left) and 4 (right).

Figure 7: Playstation 2 Dual Analogue / Trio Linker Plus

6.2.7 Trio Linker Plus with Dreamcast Controller

In the Dreamcast controller, the axes X1/Y1 are assigned twice and can be controlled either analog via the analog stick or digital (full stop) via the D-pad. X2 is located at the left index finger trigger, Y2 at the right index finger trigger.

Keys 1, 2, 3 and 4 are found on the key pad on the right, starting at the top of the green key with 1 and then running clockwise 2 on the blue key on the right, 3 on the red key at the bottom and 4 on the yellow key on the left.

Figure 8: Dreamcast Controller / Trio Linker Plus

6.2.8 Trio Linker Plus with Gamecube Controller

Due to the lack of a Gamecube controller, I can't test this one, but assume that the mappings compare well with the Playstation 2 Dual Analogue.

6.2.9 Default Joystick (if such thing exists)

Since the Phantom Hawk worked relatively well with the default template from felis' github library, this serves as a template for a "standard joystick" for the axes (X1=X, Y1=Y, X2=Twist, Y2=Throttle, Button 1-4=Button 1-4), but no extra buttons are extended to Button 1-4 and no calibration takes place.

7 Bill of Materials

- Arduino Mega 2560, to match the case, the version with large USB Type B Jack – Attention, there are different Mega 2560 variants circulating. Most of them follow the official arduino.cc layout. On the "AZ Delivery" variant on Amazon Germany, both the USB socket and the power socket are slightly offset, so the left side panel does not fit. The power jack can also clash with larger USB connectors for the USB Host Shield even with a redesign of the left side panel. Make sure that the Arduino follows the standard layout, e.g. Arduino Official or Elegoo.

- USB Host Shield 2.0 – Currently I am not aware of different versions of the Uno/Mega USB Host Shield 2.0. However, depending on the supplier, a short or a long reset switch can be installed. The pin was designed for the short switch and can be shortened with a side cutter for long reset switches if necessary.

- Three-Headed Monkey Shield PCB

    - PCB itself:
        - Download Gerber file set from Github: RetroARDUInput/PCB/Gerber at main · RetroFuturisticEngineer/RetroARDUInput (github.com)
        - Order at https://www.pcbway.com/ or https://jlcpcb.com/ (minimum order 5 Pieces)
    - 3x Socket DIL-16 Raster 2,54, if you want to socket the ICs
        - DE: 3x MPE 001-1-016-3: Präzisionsfassung, 16 polig, RM 2,54 bei reichelt elektronik
        - International: 3x 115-47-316-41-003000 Mill-Max | Mouser Europe
    - 1x Maxim MAX232 RS232-Transceiver, DIL-16
        - DE: 1x MAX 232 DIP: RS232, 2 Treiber - 2 Empfänger, DIL-16 bei reichelt elektronik
        - International: 1x MAX232CPE+ Maxim Integrated | Mouser Europe
    - 2x Maxim/Dallas Semiconductor DS1803-100 (DS18030-100+) 100kOhm Dual Digital-Potentiometer, 8 Bit resolution per Poti, DIL-16
        - DE: 2x Digitales Potenziometer DS18030-100+, 10kΩ 256-Position Linear 2-Kanal DIP 16-Pin | RS Components (rs-online.com) (only for commercial customers, be aware that Reichelt only has the DS18030-010 with 10kOhm which delivers wrong values to the joystick port)
        - International: 2x DS18030-100+ Maxim Integrated | Mouser Europe
    - 5x Electrolytic Capacitor Radial 10µF/50V
        - DE: 5x FR-A 10U 50: Elko radial, 10 µF, 50 V, 105°, Low ESR, 5 x 11 mm, RM 2 bei reichelt elektronik
        - International: 5x EEU-FR1H100 Panasonic | Mouser Europe
    - 3x D-SUB 9 Female Jack Print-Mount, short case, depth 14,3mm
        - DE: 3x W+P 107-09-2-1-2: D-SUB Buchse, 9-polig, gewinkelt bei reichelt elektronik
        - International: L77SDE09S1ACH4F Amphenol Commercial Products | Mouser Europe
    - 1x D-SUB 15 Male Jack Print-Mount, short case, depth 12,3mm
        - DE: 1x D-SUB ST 15US: D-SUB-Stecker, 15-polig, gewinkelt, RM 7,2 bei reichelt elektronik

        - International: 1x ID15P33E4GV00LF Amphenol FCI | Mouser Europe
    - 1x DIN-5/8 Keyboard Jack
        - DIN-8 270° Jack Print Mount (Kycon KCDX-8S-N oder pin compatibel)
            - DE: 1x DIN-Buchse, 8-polig (270°), Printausführung - Restore-Store
            - International: 1x KCDX-8S-N Kycon | Mouser Europe, not in stock, long lead time, minimum order 10000 pieces!
        - or DIN-5 180° Jack with same base pinout if you can spare out the Tandy 1000 Keyboard Reset signal in future (currently unused)
            - DE: 1x MABPM 5S: DIN-Buchse, 5-polig, halbrund 180°, Print, Metallpl. bei reichelt elektronik
            - International: 1x 57PC5F Switchcraft | Mouser Europe
    - 1x Mini-DIN 6-Pin Print-Mount
        - DE: 1x EB-DIO M06: Mini-DIN-Printbuchse, 6-polig bei reichelt elektronik
        - International: 1x 5749266-1 TE Connectivity | Mouser Europe
    - 1x Mini-DIN 4-Pin Print-Mount
        - DE: 1x EB-DIO M04: Mini-DIN-Printbuchse, 4-polig bei reichelt elektronik
        - International: 1x 5749264-1 TE Connectivity / AMP | Mouser Europe
    - 15x Slider Switch 1-1 (Changeover Switch), Raster 4,7mm
        - DE: 15x SS 13ASP: Schiebeschalter 1x UM, stehend, Print, RM 4,7 bei reichelt elektronik
        - International: 15x CS12ANW03 NKK Switches | Mouser Europe
    - 2x socket strip plugs 1x6-Pin, Raster 2,54mm, short
        - DE: 2x MPE 087-1-006: Stiftleisten 2,54 mm, 1X06, gerade bei reichelt elektronik
        - International: 2x G800W302018EU Amphenol Commercial Products | Mouser Europe
    - 2x socket strip plugs 1x8-Pin, Raster 2,54mm, short
        - DE: 2x MPE 087-1-008: Stiftleisten 2,54 mm, 1X08, gerade bei reichelt elektronik
        - International: 2x 10129378-908003BLF Amphenol FCI | Mouser Europe
    - 2x socket strip plugs 1x8-Pin, raster 2,54mm, long or socket strip plugs plus socket strip plug connectors 8,5mm height plus socket strip plug connectors flat (see example)
        - DE: 2x MPE 087-1-008: Stiftleisten 2,54 mm, 1X08, gerade bei reichelt elektronik
          + 2x MPE 094-1-008: Buchsenleisten 2,54 mm, 1X08, gerade bei reichelt elektronik
          + 1x BKL 10120990: Präzisionsbuchsenleiste 2,54mm 1x36 bei reichelt elektronik (shorten to 2x8)
          (Alternative: Stacking strips like these, trim to length STAPELLEISTE 50G: Stapelleiste, 50-polig, einreihig, Höhe 38mm bei reichelt elektronik)
        - International: 2x 10129378-908003BLF Amphenol FCI | Mouser Europe
          +2x 8Fx1-254mm Gravitech | Mouser Europe
          +1x 310-87-136-41-001101 Preci-dip | Mouser Europe (kürzen auf 2x8)
          (Alternative: Stacking strips like these, trim to length 8-146455-6 TE Connectivity | Mouser Europe)
    - 1x socket strip plug 2x18-Pin, long or socket strip plug plus socket strip plug connector height 7mm plus socket strip plug flat (see example)

        - DE: 1x MPE 087-2-040: Stiftleisten 2,54 mm, 2X20, gerade bei reichelt elektronik (shorten to 2x18)
          +1x BL 2X18G7 2,54: Buchsenleiste - 18-pol, gerade, RM 2,54, H: 7,0 mm bei reichelt elektronik
          +1x BKL 10120990: Präzisionsbuchsenleiste 2,54mm 1x36 bei reichelt elektronik (cut to half for 2x 18 Pin)
          (Alternative: Stacking strips like these, trim to length STAPELLEISTE 50G: Stapelleiste, 50-polig, einreihig, Höhe 38mm bei reichelt elektronik)
        - International: 1x M20-9981845 Harwin | Mouser Europe
          + 1-534206-8 TE Connectivity | Mouser Europe (8mm)
          +1x 310-87-136-41-001101 Preci-dip | Mouser Europe (cut to half for 2x18)
          (Alternative: Stacking strips like these, trim to length 8-146455-6 TE Connectivity | Mouser Europe)
    - 3x Resistor 10kOhm min. 1/4W (for Pulldown R1 to R3)
        - DE: VI MBA02040C1002: Dünnschichtwiderstand, axial, 0,4 W, 10 kOhm, 1% bei reichelt elektronik
        - International: MBA02040C1002FC100 Vishay / Beyschlag | Mouser Europe
    - Optional: 6x LED 5mm, color as you like, and 1x Resistor (around 10 Ohm minimum), depending on your brightness preferences, for the case lighting if you use a transparent front panel
        - DE:
            - Resistor: VIS PR0100010102: Widerstand, Metallschicht, 10 Ohm, axial, 1 W, 5% bei reichelt elektronik
            - LED 8000mcd 30° 4V 30mA Green: LED 5-08000 GN: LED, 5 mm, bedrahtet, grün, 8000 mcd, 30° bei reichelt elektronik
        - International:
            - Resistor: CPF110R000GKRE6 Vishay / Dale | Mouser Europe
            - LED WP7113LGD Kingbright | Mouser Europe
- Case (buy screws, 3D-print case parts)
    - 21x screws 2,9mm x 6,5mm self-tapping PH1 lens head, where used for:
        - 4x fixation Arduino
        - 4x fixation 3-Headed Monkey
        - 4x fixation top cover
        - 6x (3x each 2x) fixation front/left/right cover
        - 3x fixation rear cover
            - DE: SBL 29065-100: Blechschrauben, PAN Head, PZD, 2,9 x 6,5 mm, 100 Stück bei reichelt elektronik oder im lokalen Baumarkt
            - International: Im lokalen Baumarkt
    - Base plate
    - Top cover
    - Left cover (Arduino Ports)
    - Rear cover (1x D-SUB 15, 3x D-SUB 9)
    - Right cover (1x DIN, 2x Mini-DIN, hidden debug switch)
    - Front cover (Device logo)
    - 1x Guide sleeve for Reset switch
    - 1x Pin for Reset switch
    - 1x switch cap over 5 switches (Tandy/other keyboards)
    - 2x switch cap over 2 switches (Joystick-Potentiometer Open/Ground)
    - 5x switch cap over 1 switch (various mode switches)

If you want to order printed cases, download STL models (RetroARDUInput/Case/STL at main · RetroFuturisticEngineer/RetroARDUInput (github.com)) and order at https://www.pcbway.com/, https://jlcpcb.com/ (both SLA print) or with a local vendor (just search – eventually, they can also do FDM print) – Limitations like „only one part per STL“ (don't use the STL with all switches in it, instead order 5x the 1-switch cap, 2x the two-switch cap and 1x the 5-switch cap), warnings for too small elements (please select „I accept the risk“ when receiving the check report) and separate shipping from PCBs (unfortunately separate orders necessary) can apply with PCBWay and JLCPCB.

8 Assembly

The chapter covers all topics from parts and files to the finished device. The USB Host Shield needs some preparation. The Three-Headed Monkey Shield is assembled on both sides. Some parts are placed on the top side, e.g. the switches, which have to be accessible from the top, but other parts are placed on the bottom side to reduce the height of the case, e.g. the connectors for the retro-computers, but also the capacitors for the MAX232, which could otherwise hit the case lid, or the connectors to the Arduino, which sits under the shield. All positioning should be marked on the Silk Screen, you can hardly do anything wrong. If you want to print the case yourself with your 3D printer (specifically FDM printers are covered here), you will get some more tips for settings, but also filament changes for multicolor printing, and then the case is assembled. Finally, the Arduino is programmed.

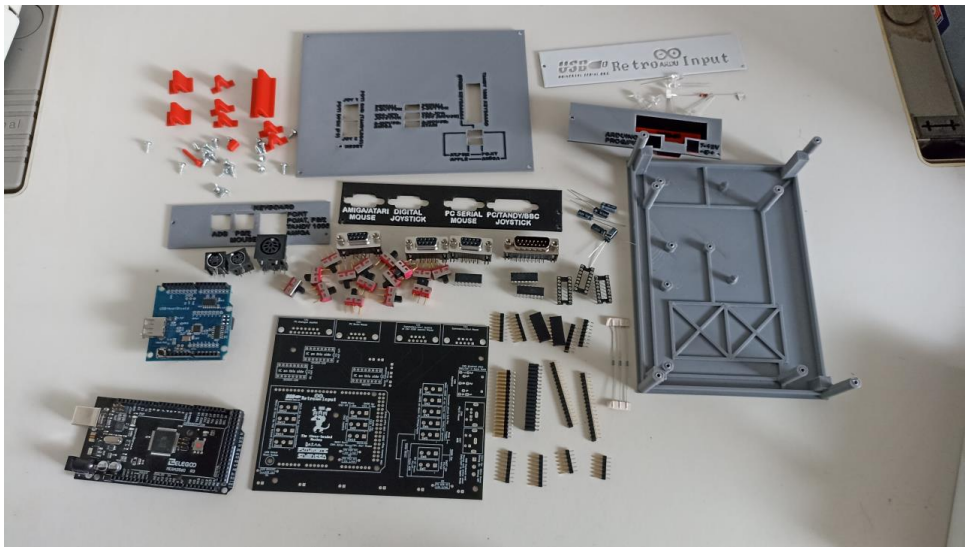

Figure 9: The project in single parts

8.1 Preparing the USB Host Shield (Set solder bridges)

The USB Host Shield is universally prepared for operation on 3.3V and 5V and must first be configured to the respective mode by setting solder bridges. If these solder bridges are not set, the Arduino crashes immediately and cannot even be programmed via the USB cable.

We are working in 5V mode and therefore we have to set the solder bridges as shown in the figure below:

- At the Header 3.3V und 5V
- At the USB-Port only 5V

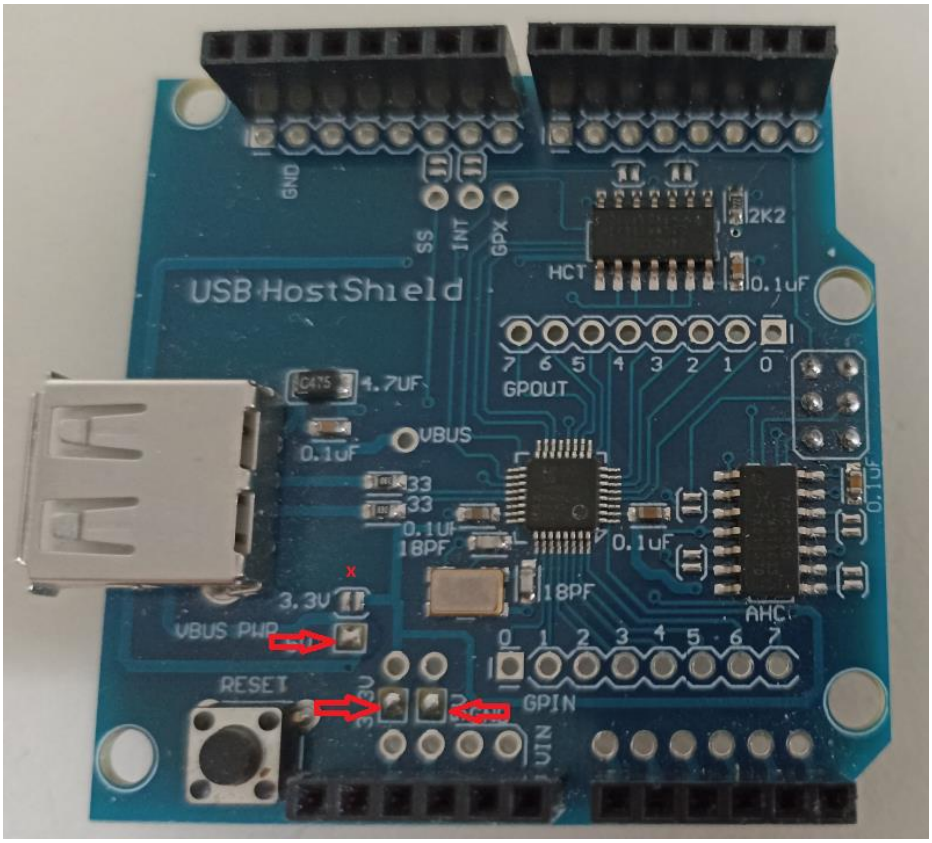

Figure 10: Solder bridges to jumper the 5V mode

8.2 Parts on top of the Three-Headed Monkey Shield

For the assembly we start on the top side. This contains only the IC sockets/ICs and the switches, apart from the debug message switch SW11, the pulldown resistors R1 to R3 and optionally the series resistor R4 for the LEDs. I recommend to socket the ICs.

Sequence (smallest parts first):

- Solder the resistors.
- Solder the IC sockets and check for solder bridges.
- Solder (except SW11) the switches (the pins are so far apart that there should be no bridges). The switches have a thin and a thick part of the pins. The thick part does not go through the solder pads and it is not intended to. The switches sit straight on the thickening, but can wobble a bit. Therefore, first solder with one pin, heat the solder again and press the switch from above so that it sits straight, so that afterwards the switch caps sit cleanly.)
- It is a matter of taste whether you insert the ICs into the sockets now or do this only at the end, after checking for solder bridges for the components on the bottom side.

8.3 Parts on the bottom of the Three-Headed Monkey Shield

On the bottom side are the switch SW11, the capacitors for the MAX232, the connectors for the emulated input devices and the pins to connect the Arduino, and optionally the LEDs.

Sequence:

- Solder LEDs, if you want lighting.

- Connectors, in order of "small to large": ADB and PS/2, then all D-SUB, then the DIN keyboard connector.
- Solder capacitors (note marking on PCB Silk Screen and capacitor, polarized electrolytic capacitors). To ensure that they hold, bend the pins slightly after insertion if necessary.
- Solder switch SW11.
- Contacting to the Arduino. Recommendation: Insert the pin rows into the stack of Arduino and USB Host Shield, the short rows into the USB Shield and the long rows (or stacks for extension) into the Arduino, put on the Host Shield and solder it. This way the pin rows will be nice and straight.
    - Tip for the pin header stack for the long pins: The pin headers (pins above and below) come with the long side into the Arduino. The high female headers go on top of these. These are then plugged into the low female headers, which come through the holes in the 3-Headed Monkey board. The high female connector for the 36-pin connection is a bit lower than the 8-pin versions, but it still fits without problems, because the sum of the pins is long enough.

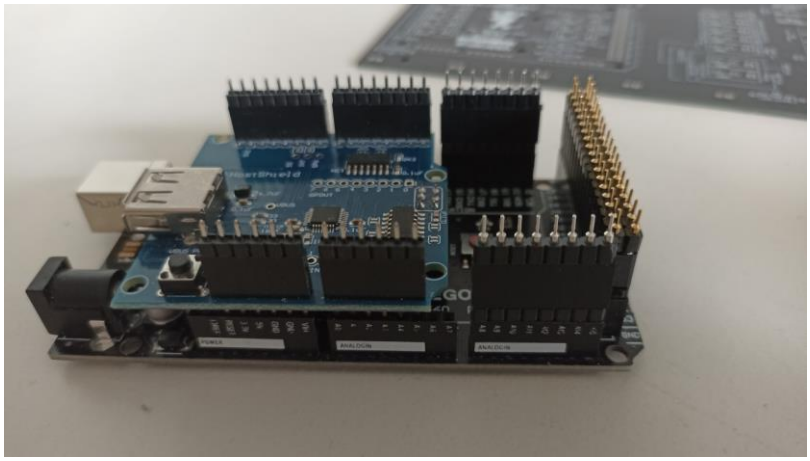

Figure 11: Inserting and soldering pin rows

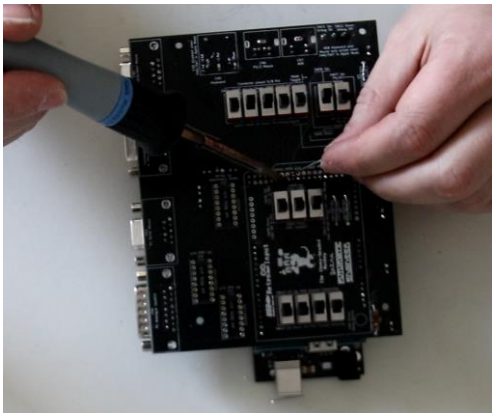

Figure 12: Completed PCB while soldering the connector pin rows

8.4 Slicing and printing the 3D-printed case

For those who want to print this case themselves, here are some slicing tips:

You'll have to find the best settings for temperature and speed yourself. In general, I can recommend the "Monotone Top Layer" from Cura (exception: left side, leads to very crude movement patterns with the USB host recess borders and is not necessary for the front surface here), because otherwise it can lead to increased repositioning, which leads to streaking. With the Monotone Top Layer, there are also some repositionings happening, but, at the cost of minimally more print time, much more intelligently with less to no streaking. Ironing led to a deterioration of the result for me, if you have had better experiences there, feel free to use it.

Generally, I print all parts with 100% infill, since a reduced infill is hardly ever used due to the design and only saves a minimum of time and also means more stability due to the screw connection with self-tapping screws using more material.

The printer must support the layer change command to print in multiple colors. Try this with a small part first, and check out any firmware updates if your printer can't handle this with the stock firmware. For example, the Ender 3 V2 with its Marlin-based stock firmware ignores the M600 filament change command and continues to print. In addition, I recommend that your printer (factory or retrofitted) has a double-sided Z-guide, a bed-level sensor, and a high-quality print bed (e.g. PEI), as filament changes require more precision. At this point I recommend the free alternative Marlin fork Jyers UI, which is available as source code and ready compiled package (manual leveling or BL-Touch/CR-Touch) and which also brings a few more useful additions, but otherwise feels like the stock firmware. When updating, check which board version you have (for the Ender 3 V2 a silent board version 4.2.2 or 4.2.7), a firmware for the wrong board version leads to malfunctions, but can be easily fixed by reapplying.

In the Post Processing item, the filament changes and other changes (ChangeAtZ) can be inserted. For multicolor prints (layer height 0.2, no rafts or other adhesion aids) I handle it as follows:

- Case baseplate and switches: No filament change, all one color, case in medium gray, switches in red
- Top-Cover (alignment stays, support structures required):
    - Start with medium gray
    - Filament change and eventually (you need to try)speed change ChangeAtZ, speed in %) for labelling at layer 23
- Right Cover (turn 90° so it faces labelling up):
    - Start medium gray
    - Filament change to black eventually, speed change (ChangeAtZ, speed in %) for the labelling at layer 11
- Rear Cover (turn 90° so it faces labelling up):
    - Start black
    - Filament change to light gray, eventually speed change (ChangeAtZ, speed in %) for the labelling at layer 11
- Left Cover (turn 90° so it faces labelling up / USB recess down, supports required):
    - Start red
    - Filament change to black, eventually speed change (ChangeAtZ, speed in %) for the labelling at layer 6
    - If speed was reduced at layer, it can be increased again at Layer 11 (ChangeAtZ)

    - Filament change for the outer cover area to medium gray at layer 88 – 5 Layers oft he outer shell were already printed, as it is not a good idea to change filament directly on top of the support
    - Filament change to black, eventually speed change (ChangeAtZ, speed in %) for the labelling at layer 94
- Front Cover „Engraved/Debossed“:
    - Start transparent or black depending on your preferences
    - Filament change to light gray fot he top layers at layer 6
- Front Cover „Embossed “:
    - Start light gray
    - Filament change to black for the labelling at layer 11

If you have problems with the font when changing filaments in multicolor printing, you may be able to print a Purge Tower. However, Cura only supports purge towers by default if a dual extruder is installed. My workaround for this is to build a narrow-walled hollow cylinder (tube), e.g. 5mm outer radius, 4.5mm inner radius, whose height is chosen so that the last layer printed is the one where you change colors. For example, a side panel (except left) is 2mm thick, so at 0.2mm layer thickness it has 10 layers. At the 11th layer the filament is changed and the font is printed. So the Purge Tower must be 2.2mm high and positioned to print before the main part (preview for the appropriate layer and then turn the print head path to start). Then the Purge Tower is printed first, with all the problems that can be expected after a filament change (too much or too little filament), the printhead is then filled correctly, a retract takes place and the printhead changes to the font area in a controlled manner and writes cleanly there.

8.5 Assembling the 3D-printed parts for the case

8.5.1 Tip in advance - Disassembling and re-tightening screws in plastic housings

Should you ever remove screws from the case, a tip applies that can also be applied to other plastic cases, e.g. injection molded ABS cases like our beloved retro computers have:

Self-tapping screws cut themselves a thread when first screwed in. If you turn the screw out and back in again and don't catch the original cut thread, a new thread will cut through the old thread and the screw mount will wear out and no longer provide any grip if repeated too often.

Therefore, with any plastic housing with self-tapping screws, but even more so with the somewhat more sensitive 3D printing, the following applies: When screwing into a virgin housing for the first time, simply screw in. When unscrewing and screwing in, however, use the following technique: Put the screw in place and first turn it "up" (counterclockwise) until you feel and hear a slight jerk. Then the screw has fallen into its already cut thread and can then be screwed in (clockwise).

8.5.2 And finally – Assembling the case

The case is assembled as follows:

Step 1: Insert the Arduino into the housing. Two of the spacers have pins. These help align the Arduino, but are also due to the fact that screw heads would have no place here. Tighten the Arduino with a total of 4 screws

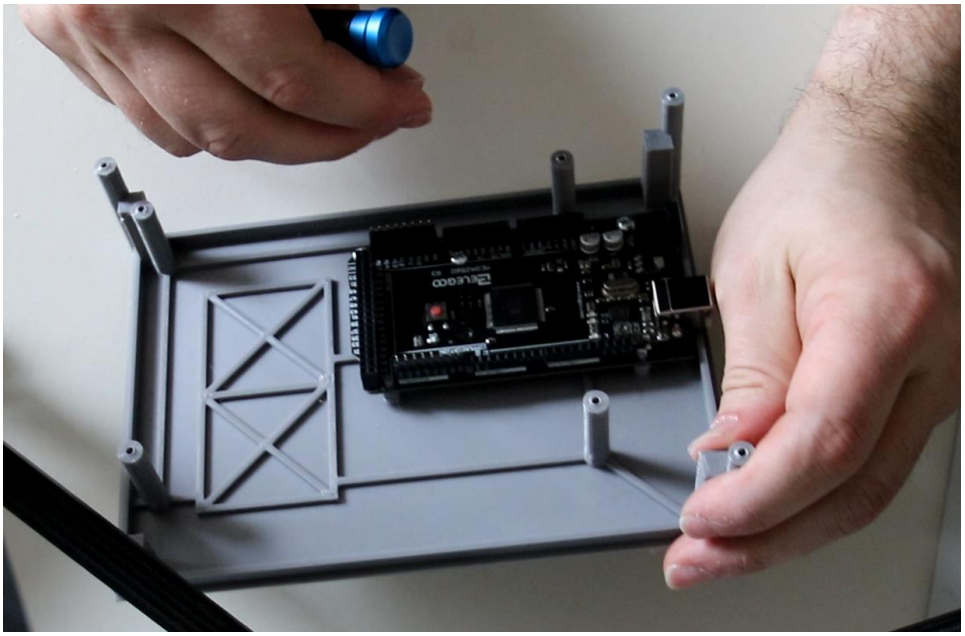

Figure 13: Mounting the Arduino

Step 2: Put on the USB Host Shield

Step 3: Place the 3-headed monkey shield and screw it in place with a total of 4 screws. Loosen the screws in the front left, front right and rear left and lift the 3-Headed Monkey slightly. Place the guide sleeve for the reset pin on the reset switch of the USB host shield (the hole fits exactly around the button) and clamp the sleeve between reset switch and 3-Headed Monkey. Press the 3-Headed Monkey again and screw it tight.

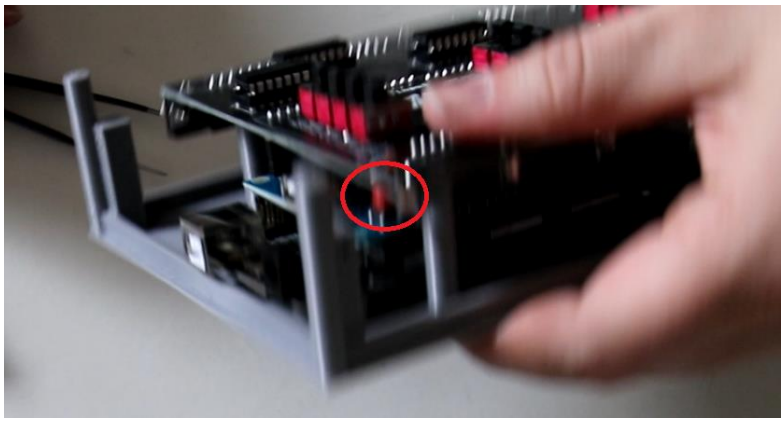

Figure 14: Fixing the 3-headed monkey and the reset guide sleeve

Step 4: Place switch caps on the switches. The 5-way cap covers the 5 switches (mode and pinout change) for the Tandy keyboard mode. The two 2-way caps cover the 2 switches each that hold the analog joystick pots open or pulled to ground (PCs open, BBC Micro, Tandy and Tano/Dragon to ground). The 1-way caps go on the remaining switches. Put the reset pin through the hole in the 3-Headed Monkey Shield so that it rests on the reset switch of the USB Shield. Then carefully put on the lid and guide the switches through their respective holes, carefully insert the reset pin into the

guide sleeve in the lid. As soon as the cover including all switches is cleanly placed on the housing, screw the cover with 4 screws.

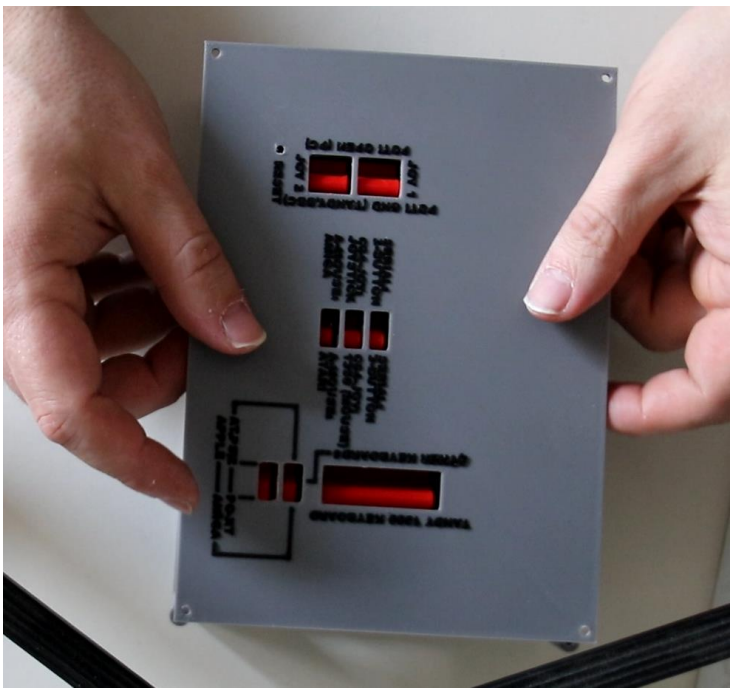

Figure 15: Put on the top cover

Step 5: Screw on the side covers with 2 (rear cover 3) screws

Figure 16: The case is finally assembled

9 Programming the Arduino

The Arduino now needs its firmware of course. Download the package with the .ino sketch and some .cpp and .h files for the RetroARDUInput from my Github page: [RetroARDUInput/RetroArduInput_FW at main · RetroFuturisticEngineer/RetroARDUInput (github.com)]

You also need the USB Host Shield Library 2.0 from Circuits@Home, which you can download from the library manager (Ctrl+Shift+I). The version is secondary, but I recommend the current 1.6.1. On the Github of felis (link in the sources) you can find the library (but I recommend the library manager, because it can install the latest library globally, but you can also download it there and include it manually) and some examples, which I used as a basis.

A reprogramming is currently necessary due to missing joystick recognition also for the change to another joystick (adjust variable JoyType in globals.cpp with one of the values defined by #define from globas.h), therefore everyone should deal once with the Arduino development environment. But if you can keep a DOS-PC alive, this should be the least hurdle.

The directory for the sketch must have the same name as the .ino file, namely RetroArduInput_FW. Then the sketch can be loaded and should be displayed, with several more automatically opened

tabs for the .h and .cpp includes in the IDE. If the Host Shield Library has been correctly included as a global library, all you really need to do is click the programming button and the sketch will compile and upload.

10 Information on the interfaces

10.1 CN1: Analog Joystick

The PC joystick interface was originally designed for 2 joysticks with 2 axes and 2 buttons each, which could be connected via a Y-cable. Since the USB libraries for the Arduino USB Host Shields do not support two simultaneous joysticks, the approach of more modern joysticks was chosen, which used the pins for joystick 2 for additional axes (e.g. throttle on the CH Products Flight Stick) and buttons. Since most USB joysticks also have 3 or 4 axes (mostly X, Y, rotation and throttle) and can handle 4 or more buttons, 4-axis/4-button joysticks can be provided.

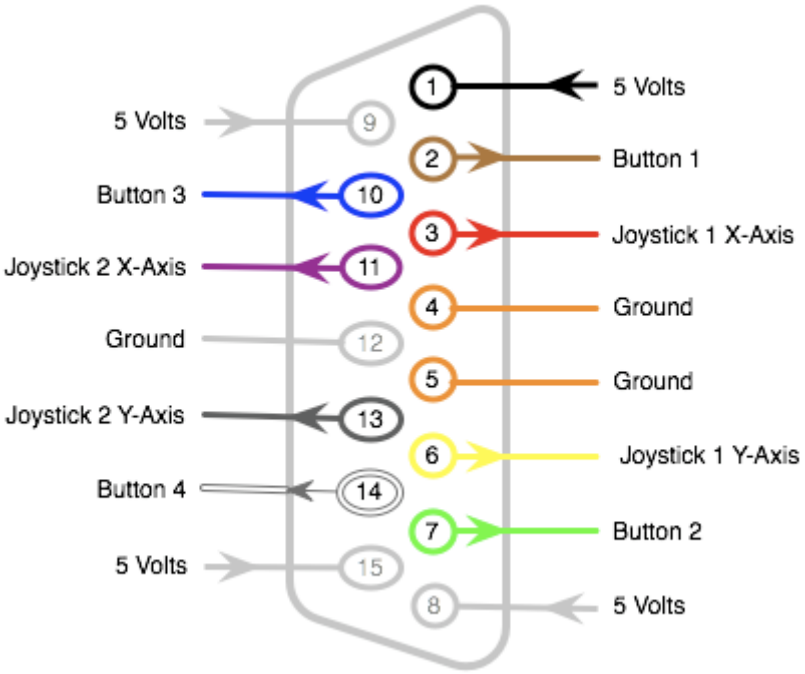

Figure 17: Pinout Analog Joystick Port

The 15-pin analog joystick port is used natively for PC joysticks. The 15-pin joystick interface provides a reference voltage for this purpose (5V for ISA cards, 3.3V for PCI cards). These are provided per axis and sent through one 100kOhm potentiometer each. PC sound cards have a 10kOhm resistor to ground at each input. This resistor is used to measure the voltage drop. The potentiometer, in a real joystick a linear potentiometer or a mechanically translated rotary potentiometer, in the case of the RetroARDUInput a digital potentiometer, provides a resistance of 0-100kOhm between input and slider, which together with the 10kOhm potentiometer on the sound card according to the voltage divider rule results in the respective measuring voltages in the range of 5V (or 3.3V) at R1=0Ohm and 0.45V (or 0.3V) at R1=100kOhm.

$$U = Uref * \frac{R2}{R1 + R2}$$

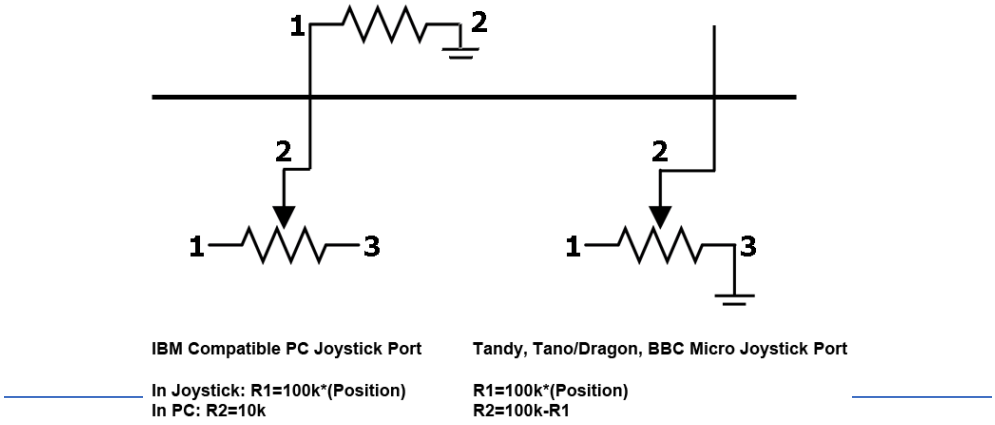

Figure 18: Joystick voltage divider rule for different computer types

Since some computers with analog joystick ports do not have this measuring resistor, the potentiometer must be pulled to ground for this. This is done by the switches SW12 to SW15. Affected computers are (no claim to correctness) the BBC Micro (also a 15-pin D-SUB, but with a different pinout), the Tandy TRS-80 (and thus probably also the Tandy 1000, which has ports for TRS-80 joysticks) and the Tano/Dragon. For these devices, R1 is the path between the Uref and the slider (0-100kOhm) and R2 is the path between the slider and ground (R2-R1 = 100-0kOhm). By switching the potentiometer outputs to ground, the joystick port of the RetroARDUInput can also be used for these computers with adapter cables.

The buttons are switched to ground by pulling the button pins (button pressed=connection to ground).

10.2 CN2: PC Serial Mouse

The serial mouse uses the RS232 interface, through which the data is transmitted serially as packaged data packets. Since this interface uses a voltage range of +12 to -12V, a MAX232 is necessary as a converter from and to the 5V TTL voltage of the Arduino.

We use the Serial2 UART of the Mega2560 here, which is set to the bit rate and handshake 1200-7N1 used by Microsoft (2-button) and Logitech (3-button).

In addition, the RTS pin is given to an I/O pin via a receive line of the MAX232. In this case we take one with interrupt. The PC normally supplies power to the mouse via CTS and RTS, but drops RTS for a short moment when the mouse driver wants to query the type of the connected mouse. Returned is "M" for mouse (2-button Microsoft) or "M3" for Logitech 3-button mode.

Transmitted packets are:

Bit Packet	6	5	4	3	2	1	0
1	1	L-Button	R-Button	Y7	Y6	X7	X6
2	0	X5	X4	X3	X2	X1	X0
3	0	Y5	Y4	Y3	Y2	Y1	Y0
(4)	0	M-Button	0	0	0	0	0

The fourth data packet is only used in the Logitech 3-button protocol, and even then not always.

It is a signed byte where the eighth bit represents the sign and accordingly values from +/- 0 to 127 can be reached.

The Logitech mouse only transmits packets 1 to 3 in the normal state and thus behaves like a Microsoft mouse. If the middle button is pressed, the fourth packet with the value 0x20 (bit 5 = 1) is transmitted. If the middle button is released, the fourth packet with the value 0x00 must be transmitted once if the mouse is stationary. In motion, a set with X/Y data consisting of only 3 data packets is also sufficient for the mouse driver to regard the middle button as released.

10.3 CN3: C64/Atari Joystick and 1350 Joystick Mouse

Note: The 1351 Proportional Mouse has not been implemented here. This would actually specify different positions via the paddle registers of the SID, which can change depending on the movement speed. However, an implementation did not seem desirable to me for the time being, measured by effort and benefit. Perhaps this will come in a later revision.

The 1350 Joystick Mouse uses the direction pins of the digital joystick and closes them against ground, just like the microswitch in the joystick, when the mouse is moved in a certain direction. Therefore it can only "move" and "not move". So in this case the connector CN3 reacts to the USB mouse and not to the USB joystick. This is more to be considered as a nice gimmick and was quickly implemented.

For the joystick implementation, the USB joystick is listened to. Digital joysticks like the D-pad of a USB gamepad or the switches of a Competition Pro USB produce full deflection. Analog joysticks, on the other hand, provide different values and, depending on the model, are trapped in a circular motion field, so cannot report full deflection on diagonals. Therefore the reaction, whether a direction is triggered or not, is given at a Dead Zone (default +/- 40 around the center of 128 at 8 bit resolution (0-255)), which can also be changed in the globals.h. Within the dead zone, i.e. a radius around the center, the pin is considered open. If the value leaves the dead zone, the pin is closed. Thus, an analog joystick does not trigger the digital joystick emulation already at the smallest movement, but also not only at the end stop.

The operation of the digital joystick is very simple. Each direction and each switch is represented by a pin which is either open (no action) or closed against ground (movement, fire button pressed). The pinout is the same for all systems (Atari, C64, Amiga) and as follows:

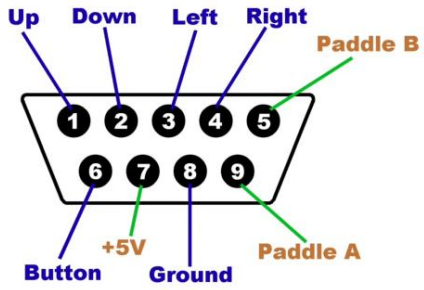

Figure 19: Commodore/Atari Digital Joystick Port

Anmerkung: Paddle bei C64, Pin 9 auch Button 2 möglich

10.4 CN4: Commodore/Atari Mouse

Commodore and Atari both use the raw data of the quadrature sensor, a rotary encoder wheel with two offset light barriers, which allow the mouse to detect direction and speed. All ball mice use this method, but interfaces like RS232, PS/2 and ADB evaluate the data and encapsulate the movement steps into a protocol.

Since the bar and slot in the rotary encoder wheel are wide enough to open or close both light barriers, this results in the following possible combinations:

0-0 – Both light barriers covered
0-1 – Light barrier 1 covered, light barrier 2 open
1-1 – Both light barriers open
1-0 – Light barrier 1 open, light barrier 2 covered

The transition speed indicates the mouse speed, the sequence of opening and covering the light barriers (00-01-11-10 or 00-10-11-01) the direction.

Commodore (Amiga, PC 1, PC 10-III, PC-20-III, COLT) and Atari (ST series) use the same pins, but Atari arranges the pins for the quadrature pulses slightly differently. This can be countered by the Arduino simply using different output pins for the values, configured by the mode switch SW10.

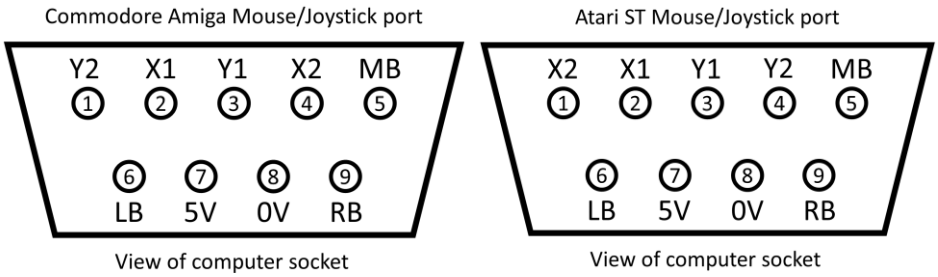

Figure 20: Commodore and Atari Mouse Port

The buttons are switched by pulling the button-pins to ground (Button pressed=connection to ground).

10.5 CN5: Keyboard

Common to all computers with DIN keyboard connector (PC-XT, PC-AT, Desktop Amigas, Tandy 1000) are the first 5 pins of the DIN connector.

Except for the Tandy 1000, all systems use the same 5-pin 180° DIN connector with the same pinout. The Tandy 1000 uses an 8-pin 270° DIN connector with a different pinout. Since the 5-pin 180° DIN cable also fits into the 8-pin 270° socket, an 8-pin 270° socket was used here to be able to cover all pins of the Tandy.

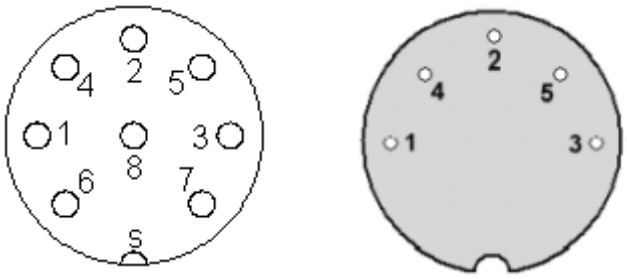

Figure 21: 5-Pin 180° (PC, Amiga) and 8-Pin 270° (Tandy 1000) Keyboard Pinout Connector Side

Tandy1000 8-Pin	PC-XT/PC-AT/Amiga 5-Pin
1: Data	1: Clock
2: Busy	2: Data
3: Ground	3: Reset
4: Clock	4: GND
5: VCC +5V	5: VCC +5V
6: Reset	
7: NC	
8: NC	

In fact, a 5-pin socket would also be sufficient, since only pin 6, the keyboard reset of the Tandy 1000, is on a pin not occupied in the 5-pin socket. The pins Keyboard Reset and Busy are actually not needed and were not considered in the development. But since they are there now and have been assigned on the Arduino as well as the keyboard connector, nothing stands in the way of future developments that use these pins.

The timings, one-way/two-way communication, scan codes and handling of the so-called modifier keys (Ctrl, Alt, Shift) are different for all systems. These were implemented individually for each system in the respective programming.

The timings usually look like this: first the data is set and then the clock edge follows with a slight time delay. The clock is always set by the keyboard, even with host-to-device communication, which is a special feature of AT or PS/2. Here the host initiates a handshake via the clock line (pulls to LOW), whereupon the keyboard sets the clock on which the host writes its communication.

The timings for writing to the keyboard are (source: www.kbdbabel.org Keyboard Signalling Collection):

10.5.1 Timings XT

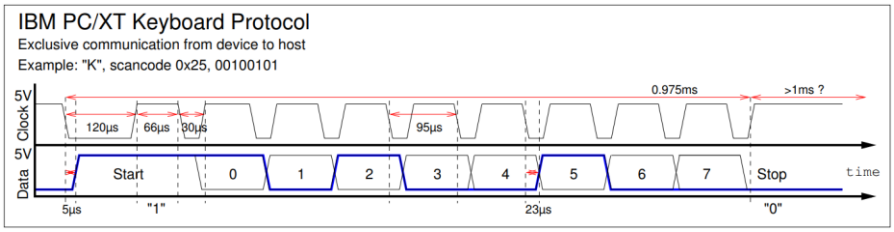

Figure 22: Timings XT

10.5.2 Timings AT and PS/2

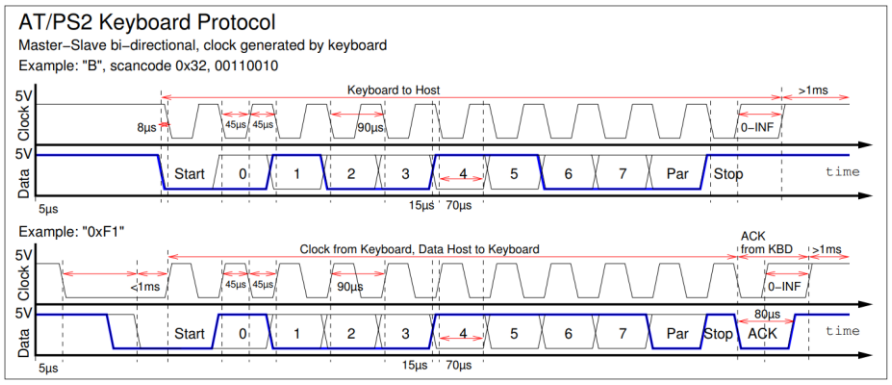

Figure 23: Timings AT und PS/2

10.5.3 Timings and specifics Amiga

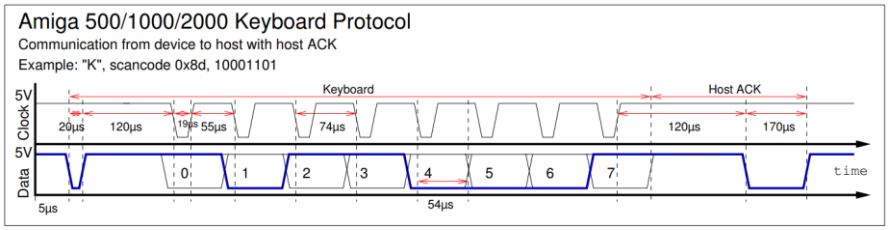

Figure 24: Timings Amiga

Note that here the timing diagram (original and taken unchanged from kbdbabel.org) shows the wrong bit order, the Amiga transmits the bits backwards, but only from bit 6 to 0 (7 bit key scan code highest bit first) and lastly bit 7 (0=Make, 1=Break), so in the order 6-5-4-3-2-1-0-7. Also note that the Amiga uses inverted logic, 1=LOW and 0=HIGH.

In addition, the Amiga keyboard for the Amiga 2000 triggers a reset warning when Ctrl+Amiga+Amiga are pressed (special scan code) and waits up to 10 seconds for the Amiga to confirm that all emergency shutdown commands have been completed and can be restarted. The Amiga 500 ignores this. After that the keyboard reset line is pulled to 0 on the Amiga 500 or the clock signal is pulled for min. 500ms on the Amiga 2000, whereby a counter chip triggers the reset. Both methods were implemented here, so the emulation should be compatible for Amiga 500 and 2000, if the keyboard reset line on the Amiga 2000 (unfortunately I don't have one for testing) should lead to problems, its digitalWrite must be commented out.

10.5.4 Timings Tandy 1000

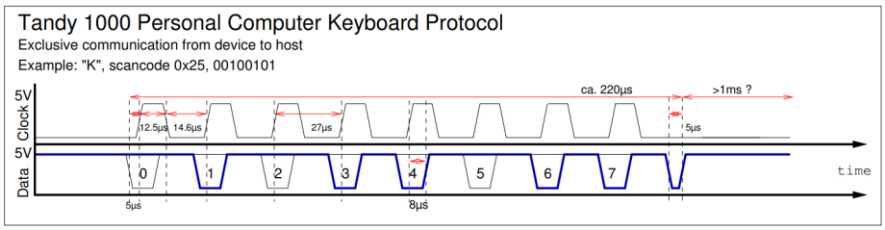

Figure 25: Timings Tandy 1000

10.6 CN6: PS/2 Mouse

The PS/2 mouse uses the same communication as the AT and PS/2 keyboard. Therefore certain routines can be recycled relatively easily.

The communication is, as with the keyboard, bidirectional. The host can thus reset/query the mouse, but also control it. Parameters like resolution and scaling factor are ignored for the time being, but stored, because the host also queries these if necessary. Here, of course, should be answered correctly with the previously set values. Due to a certain sequence of resolution changes, mice with scroll wheels are put into Microsoft Intellimouse mode and answer with type 3 instead of type 0. We ignore this, because the USB Host Shield Library cannot handle scroll wheels. We tell the PC with the repeated output of type 0: Here is a normal 3-button mouse. An Intellimouse or other scroll wheel mouse would answer with 3 at some point here and tell the PC that the scroll wheel is present.

With the help of the mouse replies, the BIOS (namely for the activation of the PS/2 port during the self-test and reservation of IRQ 12) and mouse driver recognize that a PS/2 mouse is connected.

The default mode that the driver should put the mouse into is stream mode. Here the following three data packets are transmitted with each movement:

Bit Packet	7	6	5	4	3	2	1	0
1	X-Overflow	Y-Overflow	X-Sign	Y-Sign	1	M-Button	R-Button	L-Button
2	X7	X6	X5	X4	X3	X2	X1	X0
3	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0

We don't use the overflows, the sign gives the direction, so the mouse can move in a speed from +/- 0 to 255.

10.7 CN7: ADB

ADB is a bus system that works via addresses and bidirectional communication. The default addresses are 2 for the keyboard and 3 for the mouse. The host repeatedly sends ADB commands which are Reset (line timing), Flush (command 0 - ignored in most implementations like this one), Listen (command 2 - host sends data or commands) or Talk (command 3 - the device now responds with data packets like pressed keys or mouse movements).

The whole thing takes place bidirectionally via only one data line.

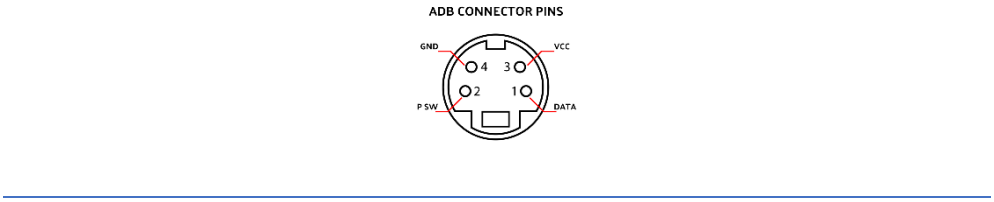

Figure 26: ADB Pinout

The Power Switch Pin is used to turn on the computer. Especially on older Macs, there is no power switch on the computer itself, but when you press the corresponding key on the keyboard, it is switched on by pulling the power switch pin to ground.

11 Code Sources

- USB Host Shield Library –Example Codes for the Circuits@Home USB Host Shield Library 2.0 and the library itself by felis, GitHub - felis/USB_Host_Shield_2.0: Revision 2.0 of USB Host Library for Arduino. / https://github.com/felis/USB_Host_Shield_2.0
- Communication to Tandy 1000 keyboard: PS/2 to Tandy Converter, Adrian Black (Adrian's Digital Basement), adapted to the USB keyboard input. Derived from this with different timings and other necessary adjustments all other keyboard output drivers for the keyboard port, GitHub - misterblack1/ps2-to-tandy1000-keyboard: PS2/AT Keyboard to Tandy 1000 Keyboard Adapter / https://github.com/misterblack1/ps2-to-tandy1000-keyboard
- Small extracts for AT/PS/2 Mouse/Keyboard from ps2dev/src at master · Harvie/ps2dev · GitHub / https://github.com/Harvie/ps2dev/tree/master/src
- Small extracts from Apple PS/2 to ADB by schenapan, GitHub - schenapan/PS2TOADB: simple ps2 mouse/keyboard arduino adapter to adb for old macintosh / https://github.com/schenapan/PS2TOADB

12 List of figures