

Hardware article

Development of an Arduino-based, open-control interface for hardware in the loop applications

Anniken Semb Kvalsund, Dietmar Winkler*

University of South-Eastern Norway, Porsgrunn, Norway



ARTICLE INFO

Keywords:

Arduino
Python
Electric drive
Voltage converter interface
IO-module
Electronics
Low-cost

ABSTRACT

This article presents a flexible control interface based on low-cost hardware solutions for electric drives which classically come either with a proprietary hardware solution or a high-cost interface solution. The interface presented can be used to connect a standard PC with an electric drive to enable testing simulation and control applications.

The control interface is developed based on the open-source Python scripting language and Arduino's open-source and accessible hardware. The new interface communicates with the test stand through its I/O terminals via developed electronic amplifiers and creates a solid base for further development towards more extensive hardware in the loop simulations.

Specifications table

Hardware name	Python/Arduino based DAQ I/O module
Subject area	<ul style="list-style-type: none"> • Engineering and material science • Educational tools and open source alternatives to existing infrastructure • General
Hardware type	<ul style="list-style-type: none"> • Field measurements and sensors • Electrical engineering and computer science • Other (Field device control)
Closest commercial analogue	USB I/O DAQ, e.g., National Instruments cDAQ 9174
Open source license	Creative Commons Attribution, Share-Alike (BY-SA)
Cost of hardware	ca. \$70
Source file repository	Zenodo[1]/GitHub[2]

1. Hardware in context

As the sensor and automation technology rapidly evolves to embrace applications ranging from automatic door openers and vehicle controls to weather reports and medical supervision, the need for low-cost, reliable controllers also increases. Unfortunately, the modern market for reliable digitally configured analogue controllers primarily consists of advanced, expensive tools such as PLCs or RTUs. The other alternatives are small, often USB-based DAQ or microcontroller units, usually with limited and fixed in-/output ranges. An issue arises whenever there is a need for a controller that exceeds the capabilities or range of essential USB-DAQ devices, but there is no room or budget for an advanced PLC.

This article presents a solid base for a flexible control interface based on low-cost hardware solutions, initially targeted towards hardware in the loop simulations. The interface was tailored to fit the analogue control requirements of two electric drives that

* Corresponding author.

✉ @dietmarw (D. Winkler).

E-mail addresses: askvalsund@gmail.com (A.S. Kvalsund), dietmar.winkler@usn.no (D. Winkler).

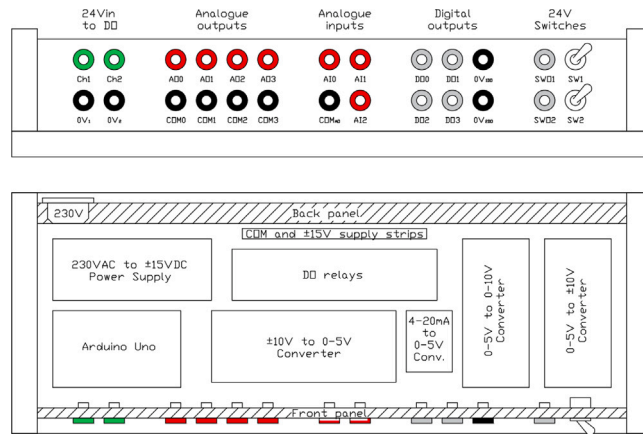


Fig. 1. Converter box layout.

classically come either with a proprietary hardware solution or a high-cost interface solution. The interface can, however, with minimal effort, also be adapted to a range of other applications. The control interface is developed based on the open-source Python scripting language and Arduino's open-source and accessible programming language and hardware.

2. Hardware description

The Arduino will not be used to store any data in this build but will act as an I/O-module, translating and transporting values back and forth between the test stand setup and a Python script running on a computer. A challenge is the Arduino's output and input voltage range 0–5 V, which is way too narrow compared to the 4–20 mA, 0–10 V, 10 V and 24 V popularised by many industrial devices. To communicate with instruments outside the 0–5 V range, the Arduino is equipped with external amplifiers, reducers, and relays based on low-cost electronic components.

Some of the advantages of this controller build are:

- Low-cost, highly customisable DAQ and controller solution.
- Open-source Arduino and Python-based interface controlled by any standard PC through a USB-interface.
- Built for hardware in the loop experiments, including analogue signal ranges of 0–5 V, 0–10 V, ± 10 V and 4–20 mA.
- Ideal for educational settings due to its low cost and high accessibility.

The IO module consists of the Arduino Uno as the main DAC/ADC, connected via various converters to banana plug inputs and outputs. Fig. 1 shows the full IO module's internal layout and converter placement. The amount and type of converters can be adapted according to needs, and the one described here is customised to fit the need for a low-cost IO controller for a frequency drive setup.

The Arduino board is based on the typical 5 V USB power and will need external amplifiers, reducers, and relays to correctly communicate with the hardware demanding other voltage- or current ranges. This section presents the converters made to equip the Arduino with the required voltage and current ranges. All the converter circuits were first designed as standard schematics and tested using an electronics simulation tool CircuitLab [3]. After confirming the circuit works, the stripboard layout is designed manually and illustrated using a CAD (Computer-Aided Design), where the circuit design and the components' physical size determine their spacing and board placement. Detailed construction descriptions are provided in a later section.

A generic 17 W AC-DC converter [4] producing a DC output voltage of ± 15 V powers the circuits' ICs and various reference voltages. In addition, all strip-board layouts for circuits containing op-amps are based around the IC op-amps LM1458 with two channels, or LN324-N containing four independent channels with a pinout configuration as displayed in Fig. 2.

2.1. Output voltage and filtering

As the equipment uses 0–10 V and ± 10 V as analogue input values, the Arduino's 0–5 V output voltage needs to be amplified before being sent to the devices. This section describes the process from the Arduino PWM output pins to the Variable Frequency Drive and servo drive's analogue input terminals.

To begin with, the PWM output voltage should go through a low-pass filter to reduce the voltage ripples. The low-pass filter is a low-cost and straightforward way to create a more stable voltage, allowing for more accurate control. It cannot produce a perfectly flat voltage output but is sufficient for this control circuit. The filter consists of a 47 k Ω resistor in series and a 1 μ F capacitor

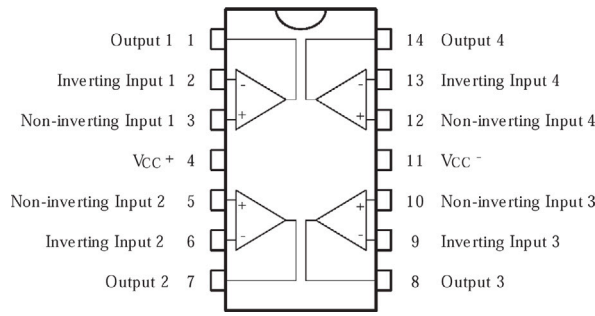


Fig. 2. Operational Amplifier IC Lm324-N pinout diagram [5].

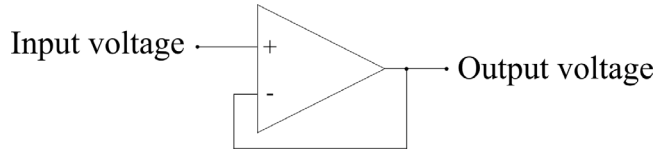


Fig. 3. Voltage follower.

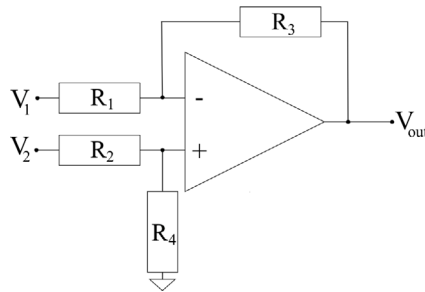


Fig. 4. Op-Amp differential amplifier.

connected in parallel to ground. These values are chosen based on Eq. (1), creating a low ripple, albeit reasonably slow response time due to its high resistance.

$$V_{out} = V_{in} \left(\frac{X_c}{\sqrt{R^2 + X_c^2}} \right) \tag{1}$$

The slow response should however not cause any significant disadvantage compared to the current equipment’s inertia.

After the low-pass filter, the filtered signal is sent through an operational amplifier (op-amp) configured as a voltage follower. The voltage follower is added because the low pass filter used is passive, causing any component added to the circuit to affect the filter’s characteristics without a buffer. For example, the configuration shown in Fig. 3 offers an op-amp where the output is connected to the input, which forces the op-amp to adjust its output voltage to equal the input voltage. Hence, the output voltage “follows” the input voltage and avoids any retroactive influence on the filter connected to its input [6].

From the voltage follower, the signal is sent through another op-amp, configured as a differential amplifier. In short, the op-amp multiplies the difference with a factor determined by the resistors. If $R_1 = R_2$ and $R_3 = R_4$, the output voltage in Fig. 4 can be calculated as shown in Eq. (2).

$$V_{out} = \frac{R_3}{R_1} \cdot (V_2 - V_1) \tag{2}$$

2.2. 0–5 V to 0–10 V converter

The 0–5 V to 0–10 V converters allow the Arduino’s 0–5 V outputs to communicate with the Variable Frequency Drive’s 0–10 V analogue inputs. The circuit illustrated in Fig. 5 is created by combining the low pass filter, voltage follower, and differential amplifier. Here, the resistors R1 and R2 from Eq. (2) consist of a connected resistor and potentiometer (R_3+R_6 and R_1+R_7 ,

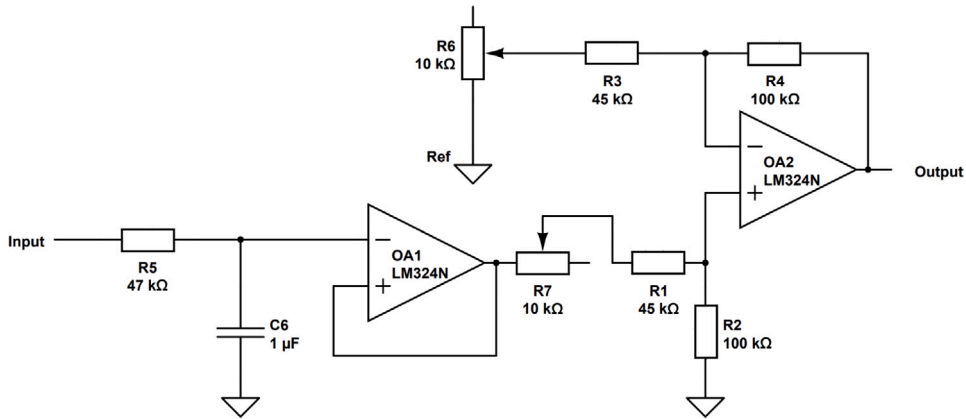


Fig. 5. 0-5 v to 0-10 V converter circuit.

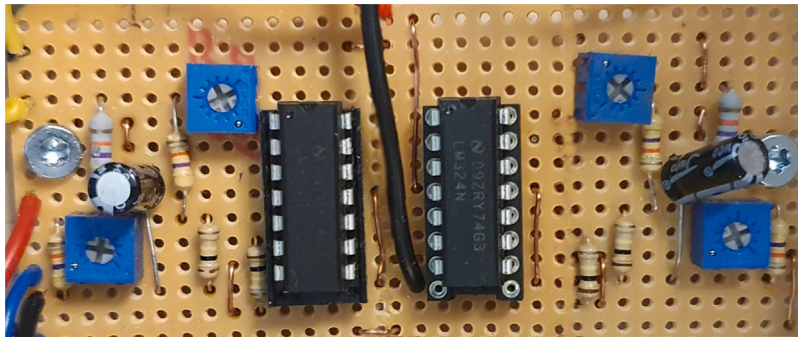


Fig. 6. Two-channel 0-5 v to 0-10 V converter.

respectively). The potentiometers allow resistance adjustments for calibration even after the circuit is soldered, and their ideal value is calculated as the mid position.

This differential amplifier uses ground as its reference voltage and a resistance ratio that based on Eq. (2) results in the voltage amplification:

$$V_{out} = \frac{R_4}{R_3 + \frac{R_6}{2}} \cdot (V_{input} - 0) = \frac{100 \text{ k}\Omega}{45 \text{ k}\Omega + \frac{10 \text{ k}\Omega}{2}} \cdot V_{input} = 2 \cdot V_{input} \tag{3}$$

Thus resulting in a linear amplification circuit with a low pass filtered input, where $V_{output} = 2V_{input}$, i.e., a voltage doubler. The commercial hardware connected to these module outputs will receive double the PWM voltage the Arduino pins outputs.

As the setup needs two voltage doublers, and each voltage doubler circuit needs two op-amps, one four-channel lm324-N IC should ideally cover the needs for both voltage doubler circuits. However, due to a weakness discovered during testing, where top and bottom half of the IC’s amplifiers seem to affect each other, this setup only utilises two of the four available in each lm324-N. The components are all pin-mounted and 1/4w rated, and the 15 V power supply is connected directly to the IC’s positive power input.

All wires connecting to the power supply, the microcontroller and the output sockets are placed the left hand side to improve wire management.

2.3. 0-5 V to ±10 V converter

The 0-5 V to ±10V converters allow the Arduino’s 0-5V outputs to communicate with commercial hardware using ±10V analogue inputs. This converter is based on the same principles as the 0-5V to 0-10V converter, the most noticeable difference being its other negative voltage range. The circuit shown in Fig. 7 contains the same low pass filter and differential amplifier as the voltage doubler, albeit with different resistor values and the reference voltage. For its reference voltage, the circuit uses a potentiometer as a voltage divider powered with 5 V from a voltage regulator to create an input of 2.5 V, which is further sent through a voltage follower. With a reference voltage of 2.5 V and an input voltage ranging from 0-5V, the differential amplifier senses a difference between -2.5V and 2.5 V.

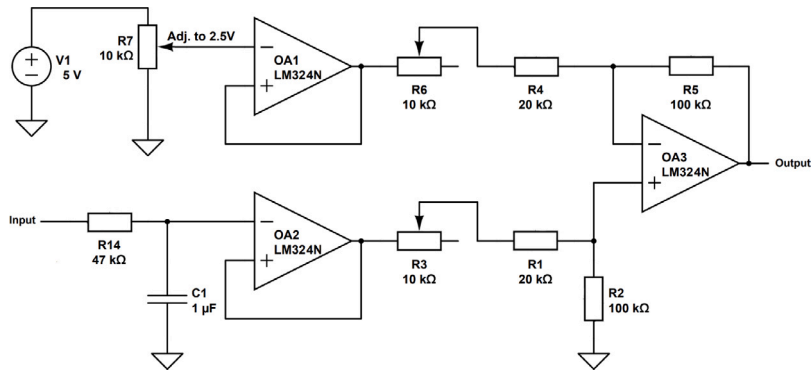


Fig. 7. 0-5V to ±10V converter circuit.

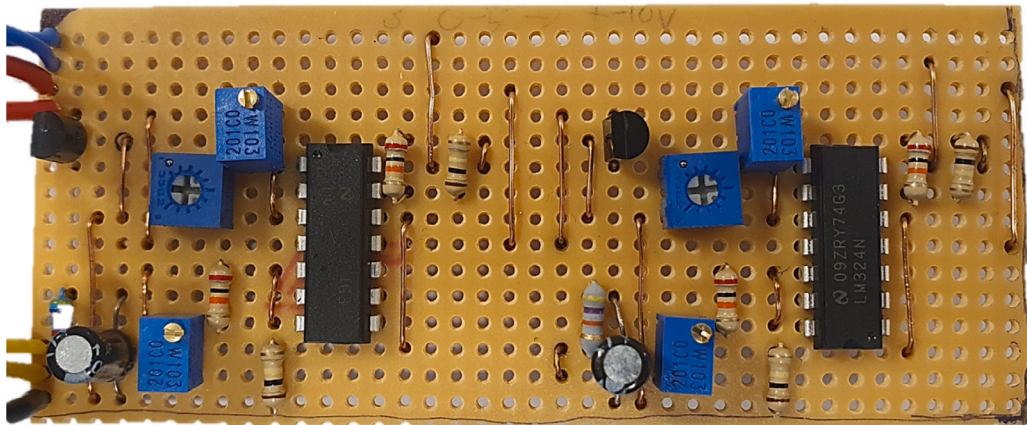


Fig. 8. 0-5V to ±10V converter stripboard.

By choosing resistive values to create an amplification of $4 \cdot V_{input}$, the circuit’s output voltage will reach a range from $-10V$ to $+10V$. The circuit’s amplification is calculated using Eq. (2) accordingly:

$$V_{out} = \frac{R_5}{R_4 + \frac{R_6}{2}} \cdot (V_{input} - V_{ref}) = \frac{100\text{ k}\Omega}{20\text{ k}\Omega + \frac{10\text{ k}\Omega}{2}} \cdot (V_{input} - 2.5\text{ V}) = 4 \cdot (V_{input} - 2.5\text{ V}) \tag{4}$$

The stripboard displayed in Fig. 8 shows two voltage quadruplers based on their IC op-amp lm324-N as shown in Fig. 2, built using the same techniques as the voltage doubler board in Fig. 6. The most noticeable features separating them are the need for an additional IC and the two voltage regulators. The voltage regulators reduce the 15 V input voltage down to 5 V before using a potentiometer as a voltage divider to create the 2.5 V reference voltage.

2.4. ±10 V to 0-5 V converter

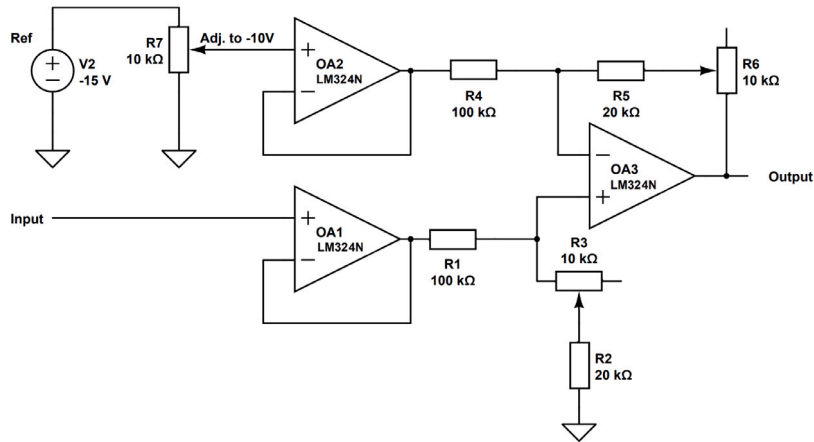
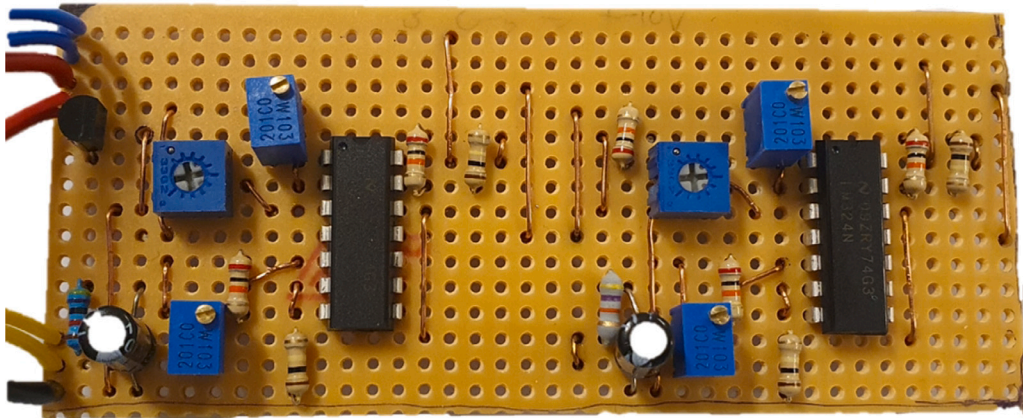
The ±10V to 0-5V converters allow feedback from the servo drive’s analogue outputs to be read by the Arduino’s analogue input pins. The converter bases its voltage transformation on the same principles as the amplifiers but with the resistor values reversed to reduce the voltage instead of amplifying it (see Fig. 9).

In this case, $-10V$, the reference voltage is achieved by a potentiometer voltage divider receiving the $-15V$ from the generic power supply. In addition, the input low pass filter used in the two previously described amplifiers is omitted, as the servo drive outputs levelled DC voltage. The circuit’s voltage reduction can be calculated using the same Eq. (2) as the amplifiers:

$$V_{out} = \frac{R_5 + \frac{R_6}{2}}{R_4} \cdot (V_{input} - V_{ref}) = \frac{20\text{ k}\Omega + \frac{10\text{ k}\Omega}{2}}{100\text{ k}\Omega} \cdot (V_{input} - (-10\text{ V})) = \frac{1}{4} \cdot (V_{input} + 10\text{ V})$$

which results in the voltage read by the Arduino essentially being a quarter of the output voltage, if the reference voltage $-10V$ is perceived as 0 V.

As illustrated in Fig. 10, the voltage reducers’ stripboard layout resembles the voltage quadrupler (Fig. 8) in many ways. The stripboard contains two circuits, each based around three out of the four op-amps in each IC. The layout neither contains any low pass filter nor voltage regulator in contrast to the quadrupler, and their reference voltage is adjusted using R7 to $-10V$.

Fig. 9. $\pm 10\text{V}$ to 0–5V converter circuit.Fig. 10. $\pm 10\text{V}$ to 0–5V converter stripboard.

2.5. 4–20 mA to 1–5 V converter

The 4–20 mA to 1–5 V voltage converter allows feedback from the drive's analogue output to be read by the Arduino's analogue input pins. This converter is the simplest of the ones created for this setup and consists in all its simplicity of one resistor and an IC used as a voltage follower, as shown in Fig. 11.

Using ohm's law results in a voltage between 1.0 V and 5.0 V converted linearly from the 4–20 mA signal, measured between the ICs output and ground: [7]

$$V_{\max} = 0.02 \text{ A} \cdot 250 \Omega = 5 \text{ V} \quad (5)$$

Using the same procedure for the lowest 4 mA Variable Frequency Drive output gives a voltage of approximately 1 V, which results in a voltage input range of 1–5 V. This method consists of few elements and is simple to implement, albeit not the most accurate. It is, however, considered satisfactory for the current application.

The converter is built on a small stripboard as shown in Fig. 12.

2.6. Digital terminals

The digital terminals allow for two channels, each containing one voltage input and two relay controlled outputs on each channel. The maximum voltage run through the channels depends on the relays, the hardware this system is intended for uses 24 V, which means 24 V power supplies has to be connected to the voltage inputs. The control relays are Fujitsu Takamisawa A5W-K miniature relays with nominal voltage of 5 V and nominal current draw of 28 mA, ideal for the Arduino outputs. As seen in Fig. 13, the

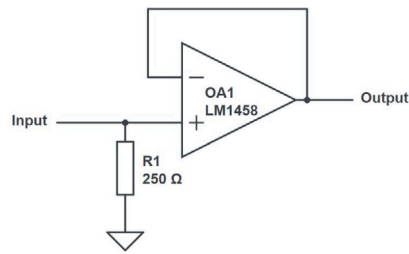


Fig. 11. 4–20 mA to 1–5 V converter circuit.

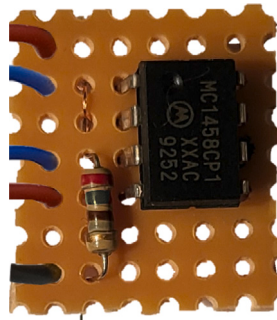


Fig. 12. 4–20 mA to 1–5 V converter stripboard.

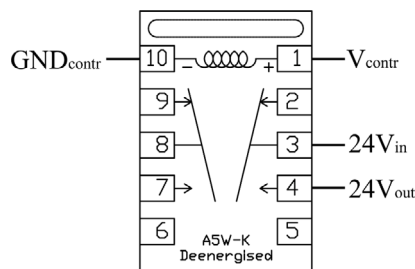


Fig. 13. A5W-K relay [8] (edited).

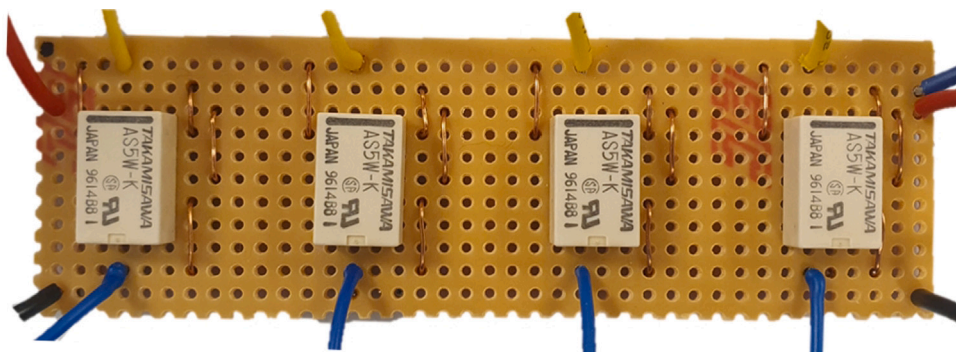


Fig. 14. Relay stripboard.

relays are to be controlled by digital Arduino output pins, V_{contr} with GND_{contr} as ground, while the 24 V input is supplied by the 24 V outputs on the Variable Frequency Drive and servo drive. The relays are double, but only one side is used to feed the digital terminals.

As seen in Fig. 14, the relays are wired onto stripboards to save space and keep the wiring organised. In addition, the four relays are divided into two groups, keeping the 24 V two drive voltage sources separated.

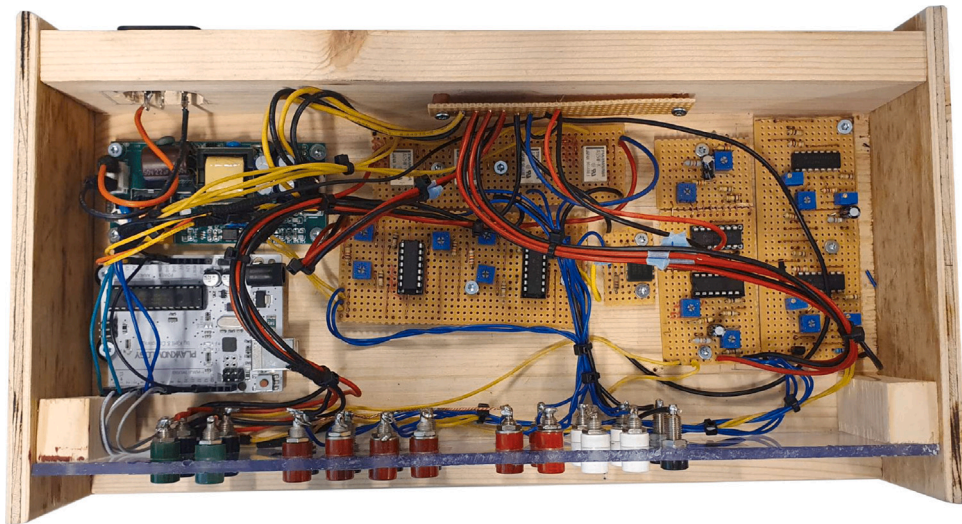


Fig. 15. All Converters, Power supply and Arduino combined.

2.7. Internal wiring

All the converters are wired together with the Arduino and powersupply and put in a box with banana sockets as access points, for convenience. The sockets have no fixed location, but the Arduino should be connected to the suitable converters, based on the capabilities of each pin. Fig. 15 shows the combined result.

3. Design files summary

Design filename	File type	Open source license	Location of the file
05t010vero.dwg	CAD	CC BY-SA	Zenodo[1]/GitHub[2]
05tpm10vero.dwg	CAD	CC BY-SA	Zenodo[1]/GitHub[2]
420t15vero.dwg	CAD	CC BY-SA	Zenodo[1]/GitHub[2]
pm10t05vero.dwg	CAD	CC BY-SA	Zenodo[1]/GitHub[2]
relaysvero.dwg	CAD	CC BY-SA	Zenodo[1]/GitHub[2]
fullWir.dwg	CAD	CC BY-SA	Zenodo[1]/GitHub[2]
cBoxLayout.dwg	CAD	CC BY-SA	Zenodo[1]/GitHub[2]
05t010sch.pdf	pdf	CC BY-SA	Zenodo[1]/GitHub[2]
05tpm10sch.pdf	pdf	CC BY-SA	Zenodo[1]/GitHub[2]
420t15sch.pdf	pdf	CC BY-SA	Zenodo[1]/GitHub[2]
pm10t05sch.pdf	pdf	CC BY-SA	Zenodo[1]/GitHub[2]
relaysvero.pdf	pdf	CC BY-SA	Zenodo[1]/GitHub[2]
arduinoMain.ino	ino	CC BY-SA	Zenodo[1]/GitHub[2]
pythonMain.ipynb	ipynb	CC BY-SA	Zenodo[1]/GitHub[2]
pythonMain.py	py	CC BY-SA	Zenodo[1]/GitHub[2]

05t010vero.dwg — Stripboard layout for 0–5 V to 0–10 V converter circuit(editable file)

05tpm10vero.dwg — Stripboard layout for 0–5 V to ± 10 V converter circuit (editable file).

420t05vero.dwg — Stripboard layout for 4–20 mA to 1–5 V converter circuit (editable file).

pm10t05vero.dwg — Stripboard layout for ± 0 –10 V to 0–5 V converter circuit (editable file).

relaysvero.dwg — Stripboard layout for 4-channel relay board (editable file).

fullWir.dwg – Full internal wiring diagram (editable file).

cBoxLayout.dwg — Converter box component layout/placement (editable file).

05t010sch.pdf — Stripboard layout for 0–5 V to 0–10 V converter circuit (PDF file).

05tpm10sch.pdf — Stripboard layout for 0–5 V to ± 10 V converter circuit (PDF file).

420t15sch.pdf — Stripboard layout for 4–20 mA to 1–5 V converter circuit (PDF file).

pm10t05sch.pdf — Stripboard layout for ± 0 –10 V to 0–5 V converter circuit (PDF file).

relaysvero.pdf — Stripboard layout for 4-channel relay board (PDF file).

fullWir.pdf — Full internal wiring diagram (PDF file).

cBoxLayout.pdf — Converter box component layout/placement (PDF file).

arduinoMain.ino — Main sketch for running the Arduino.

pythonMain.ipynb — Main Python script, presented in Jupyter Notebook format.

pythonMain.py — Main Python script, presented in the ordinary Python file format.

4. Bill of materials summary

The module is highly customisable, and costs will vary accordingly. The base components needed for every module are the Arduino and the power supply, costing \$37.72. The material cost of the converters between the inputs and outputs ranges from \$0.45 to \$4.30, while the relays cost about \$3.80 a piece, excluding the mounting boards.

- Two-channel 0–5 V to 0–10 V converter: \$2.12
- Two-channel 0–5 V to ± 10 V converter: \$4.28
- Two-channel ± 10 V to 0–5 V converter: \$2.79
- Single-channel 4–20 mA to 0–5 V converter: \$0.45
- Four-channel, dual supply relay board: \$15.34

The module created for this specific project includes one of every converter described above. It adds up to a total of about \$77, including \$13.85 in miscellaneous materials such as wiring, switches and sockets.

Designator	Component	Number	Cost per unit-currency	Total cost-currency	Source of materials	Material type
Microcontroller	Arduino Uno	1	\$29.95	\$29.95	Amazon	Semi-conductor
Power supply	± 15 V Voltage Supply	1	\$7.75	\$7.75	ebay	Other
Circuit board Sockets	Stripboard	102 cm ²	\$0.013/cm ²	\$1.33	Amazon	Metal/Polymer
	Banana sockets	24	\$0.39	\$9.35	Amazon	Metal/Polymer
Power input	C14 socket	1	\$0.59	\$0.59	Amazon	Metal/Polymer
Wire	24awg solid	1.05 m	\$0.25	\$1,05	Amazon	Metal/Polymer
Wire	24awg stranded	7.60 m	\$0.26	\$2,01	Amazon	Metal/Polymer
Switch	SPST, mini	2	\$0.89	\$1.84	Amazon	Metal/Polymer
IC	LM324N	6	\$0.60	\$3.60	Amazon	Semiconductor
IC	LM1458	2	\$1.6	\$3.20	Amazon	Semiconductor
Electronic component	5 V Voltage Regulator LM78L05	2	\$0.08	\$1.60	Amazon	Semiconductor
Relays	AS5W-K	4	\$3.80	\$15.20	ebay	Metal/Polymer
Resistor	250 Ω	1	\$0.06	\$0.06	Amazon	Metal/Polymer
Resistor	10 k Ω	6	\$0.06	\$3.39	Amazon	Metal/Polymer
Resistor	20 k Ω	4	\$0.05	\$0.20	Amazon	Metal/Polymer
Resistor	45 k Ω	4	\$0.014	\$0.06	ebay	Metal/Polymer
Resistor	47 k Ω	4	\$0.03	\$1.20	Amazon	Metal/Polymer

Designator	Component	Number	Cost per unit-currency	Total cost-currency	Source of materials	Material type
Resistor	100 k Ω	6	\$0.05	\$0.30	Amazon	Metal/Polymer
Potentiometer	10 k Ω	16	\$0.17	\$2.72	ebay	Metal/Polymer
Capacitor	1 μ F	4	\$0.08	\$0.32	Amazon	Metal/Polymer

All electronic components are $\frac{1}{4}$ W. Components with a higher power rating can be used, but are not required.

5. Build instructions

This design offers an adaptable result determined by the available hardware and desired applications, first step is therefore deciding which output and input ranges the I/O module should have. The module used in here was tailored to control a servo machine test stand consisting of two asynchronous motors controlled by a Variable frequency drive and a servo drive. One motor is used for driving, and the other simulates an adjustable load. This setup required control voltage ranges of 1 M Ω 0–10 V, \pm 10 V and provided feedback through \pm 10 V and 4–20 mA signals.

As the Arduino is able to read voltages between 0–5 V, this module will contain:

- 1 \times Output Two-channel 0–5 V to 0–10 V converter.
- 1 \times Output Two-channel 0–5 V to \pm 10 V converter.
- 2 \times Output Two-channel relay boards.
- 1 \times Input Two-channel \pm 10 V to 0–5 V converter.
- 1 \times Input Single-channel 4–20 mA to 1–5 V converter.

The converters and relay modules are built using stripboards due to their accessibility, low price, and lack of special tools required to build circuit boards. As displayed in Fig. 16, the plastic boards consist of pre-holed rows with a thin copper layer on one side used for soldering the components' legs onto.

Cut appropriate stripboard units using a stripboard cutter, fine saw, or knife. The boards needed for this module are shown in Fig. 17, picturing:

- 1 \times 25 \times 14 for the two-channel 0–5 V to 0–10 V converter.
- 1 \times 39 \times 11 for both of the two-channel relay outputs.
- 1 \times 6 \times 7 for the single-channel 4–20 mA to 0–5 V converter.
- 1 \times 39 \times 15 for the two-channel 0–5 V to \pm 10 V converter.
- 1 \times 38 \times 17 for the two-channel \pm 10 V to 0–5 V converter.

The following text describes how to cut and assemble the various stripboard converters. Table 1 contains the description of all components' symbols.

The stripboards' copper lanes must be divided into sections by adding carefully placed breakage points shown in Fig. 18. Notice the boards in Fig. 18 are mirrored, displaying the copper side correctly when looking directly at it.

Break the thin copper lanes in their appropriate places as demonstrated in Fig. 19 by twisting a 3 mm HSS drill bit by hand while applying light pressure.

After preparing the stripboards with breakage points according to Fig. 18, prepare wires for the stripboards' jumpers as displayed in Fig. 20. Red and black illustrate power jumpers, while green carries signals. Stranded wires can be used, but single-core wires are easier to thread through and solder to the stripboards.

Place the wires according to the drawings in Fig. 20 and solder them in place. Fig. 21 shows an example of a short wire soldered onto a stripboard.

Once the wiring and breakage points are done, solder the rest of the resistors, potentiometers and ICs to the stripboards following Fig. 22 to Fig. 23 one by one.

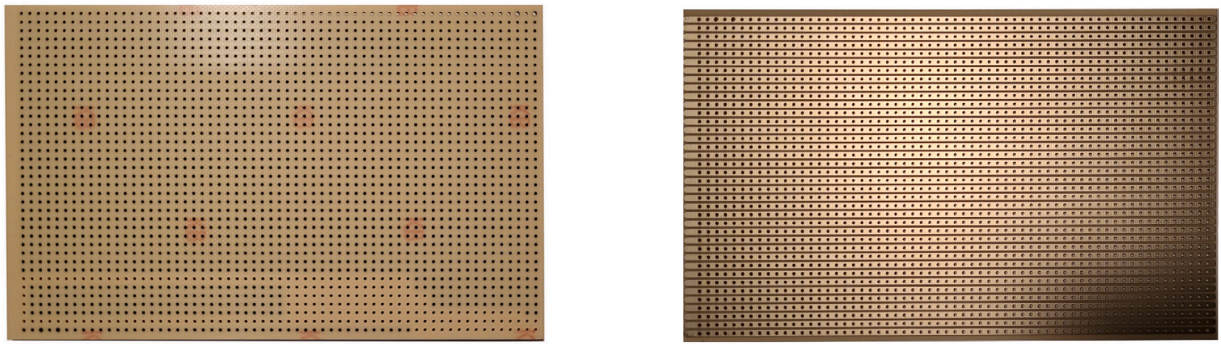
Fig. 22 illustrates the 0–5 V to 0–10 V converter stripboard layout, with one amplifier mirrored on each side of the IC. The full source files for this layout are found in 05t010vero.dwg and 05t010vero.pdf which are available at Zenodo[1]/GitHub[2]. The end result is shown in Fig. 6.

Fig. 23 shows that the four relays are divided into two groups, keeping the 24 V two voltage inputs separated. Full drawings are found as relaysvero.dwg and relaysvero.pdf at Zenodo[1]/GitHub[2]. The end result is shown in Fig. 14.

The 4–20 mA to 0–5 V converter is built on a small stripboard as shown in Fig. 24, also found as 420t15vero.dwg and 420t15vero.pdf at Zenodo[1]/GitHub[2]. The end result is shown in Fig. 12.

Fig. 25 illustrates the layout of the 0–5 V to \pm 10 V converter. Full drawings are found in 05tpm10vero.dwg and 05tpm10vero.pdf which are available at Zenodo[1]/GitHub[2]. The end result is shown in Fig. 8.

Fig. 26 contains the layout of the \pm 10 V to 0–5 V converter. Full drawings are found in pm10t05vero.dwg and pm10t05vero.pdf which are available at Zenodo[1]/GitHub[2]. The end result is shown in Fig. 10.



(a) Stripboard front.

(b) Stripboard rear.

Fig. 16. Common copper stripboard.

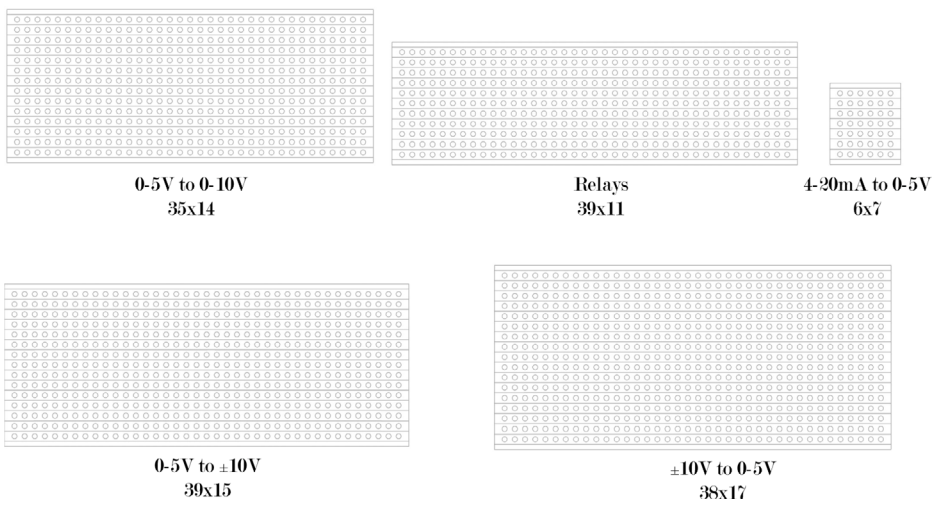


Fig. 17. Stripboard cutout dimensions.

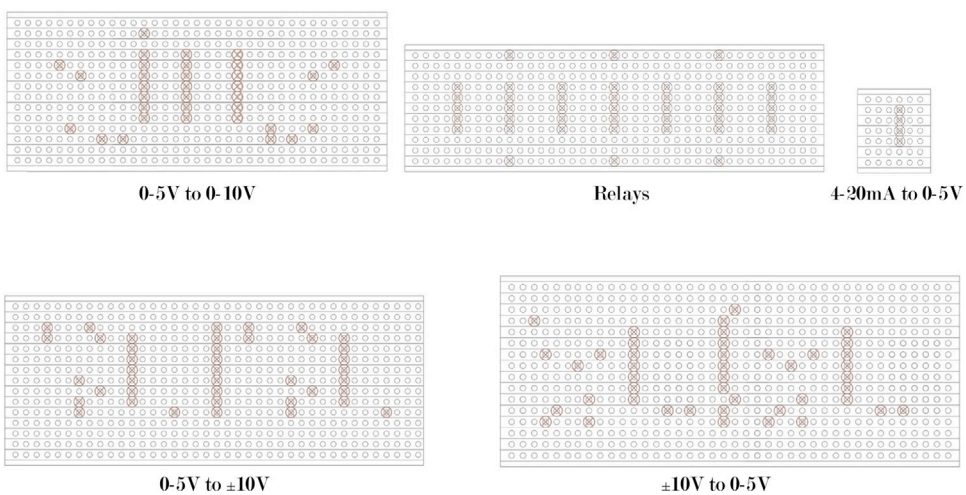
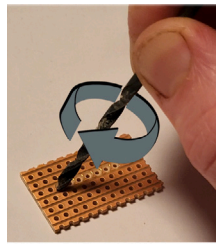
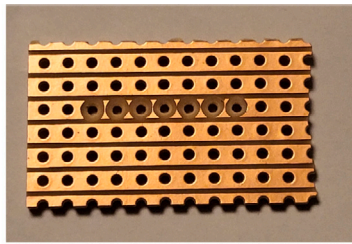


Fig. 18. Stripboard copper breakage points.



(a) Twist the drill bit clockwise.



(b) Broken copper lanes.

Fig. 19. Illustration of cutting the stripboard copper lanes.

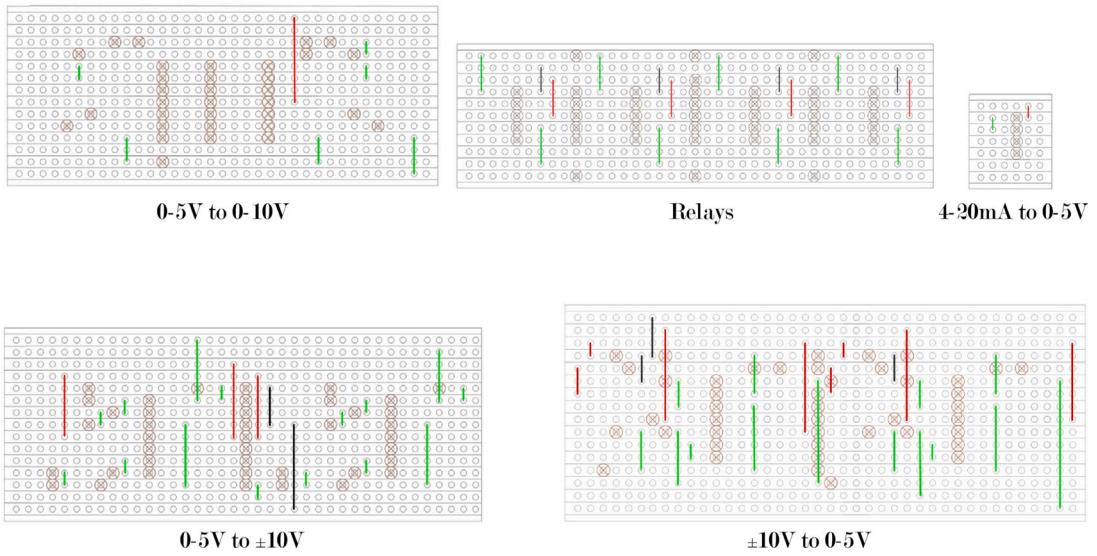
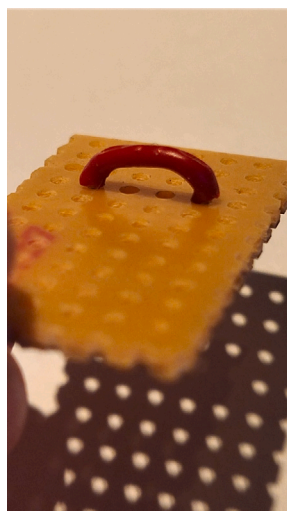
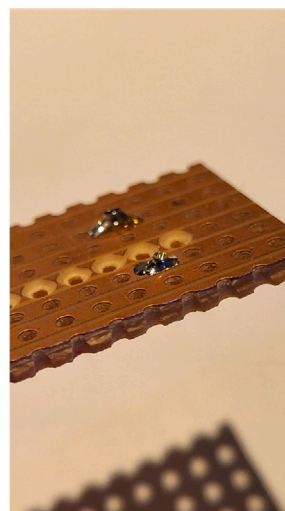


Fig. 20. Stripboard wire placements.










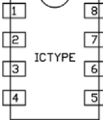

(a) Wire in stripboard, front view.



(b) Backside view with solder and cut excess wire.

Fig. 21. Illustration of soldered wire on stripboard.

Table 1
Strip-board circuits symbol description.

Symbol	Description	Symbol	Description
	Control signal wire		Resistor
	Power supply wire		Potentiometer
	Ground wire		Voltage Regulator
	Wire break		IC (size varies)
	Capacitor		

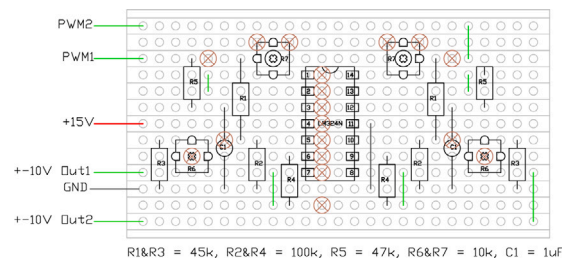


Fig. 22. 0-5V to 0-10V converter stripboard layout.

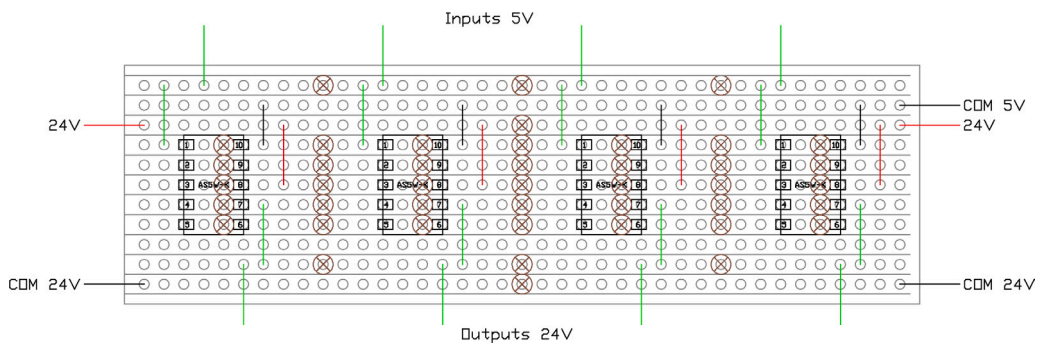


Fig. 23. Relay stripboard layout.

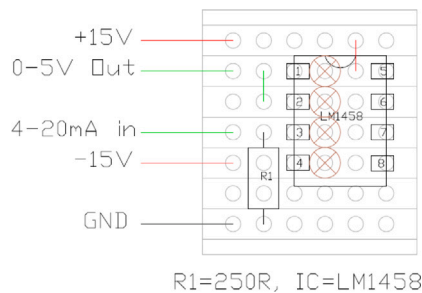


Fig. 24. 4-20mA to 1-5V converter stripboard layout.

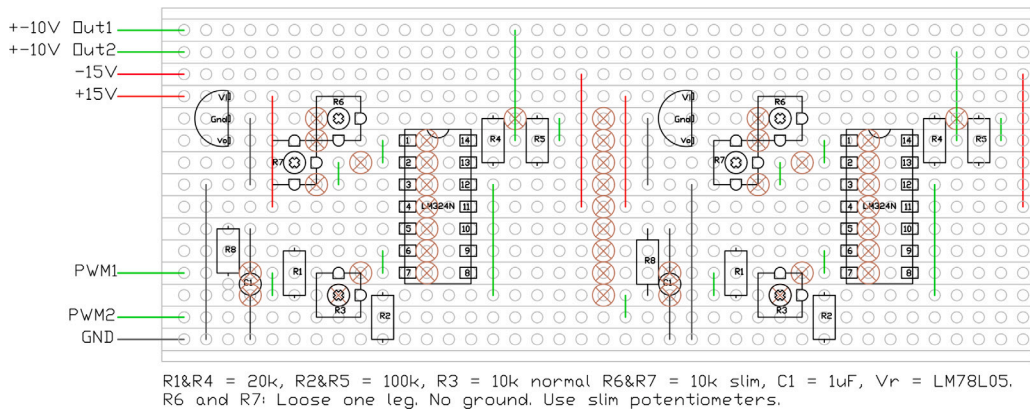


Fig. 25. 0-5V to ±10V converter stripboard layout.

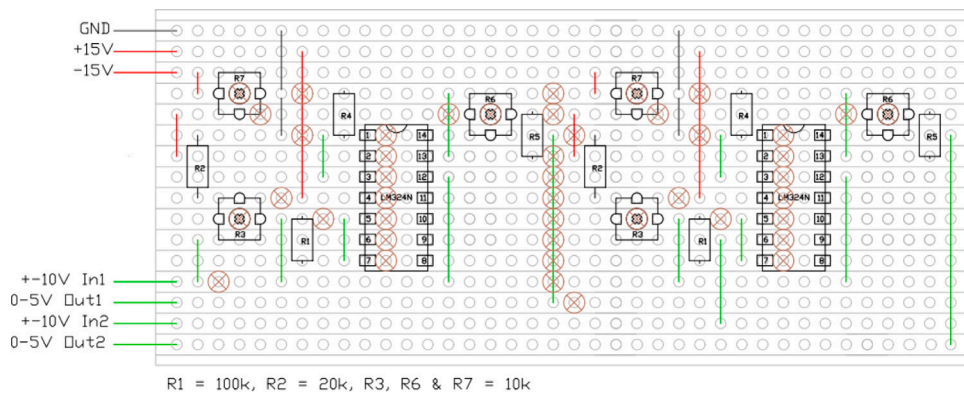


Fig. 26. ±10V to 1-5V converter stripboard layout.

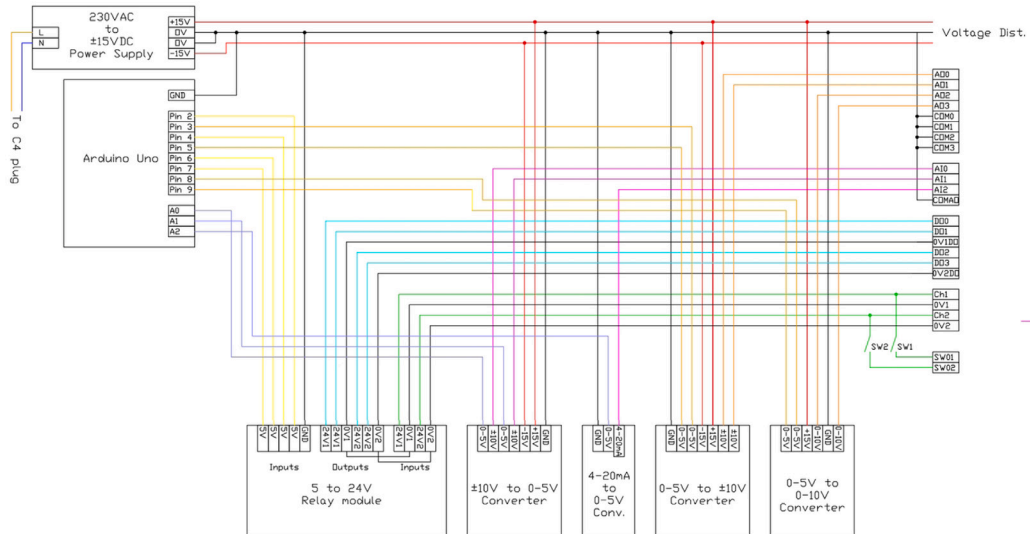


Fig. 27. Full I/O module wiring layout.

After constructing all the converters, connect them to the standard $\pm 15V$ power supply, the Arduino and the box using the wiring diagram in Fig. 27, preferably using a wire gauge of 0.5+, mm². For the final result, see Fig. 15. The wiring diagram is available as fullWir.dwg and fullWir.pdf from Zenodo[1]/GitHub[2].

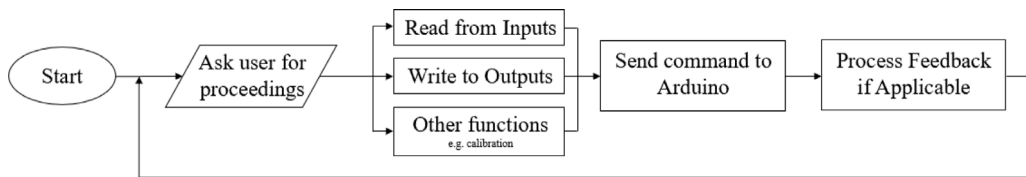


Fig. 28. Simplified Python script flowchart.

```

Get Data?
- read = analogue input values
- write = Write to analogue or digital out
- cal = Calibrate arduino analogue outputs
- on = LED on
- off = LED off
- q = Close Port

Please choose mode:
  
```

Fig. 29. Text-based user interface.

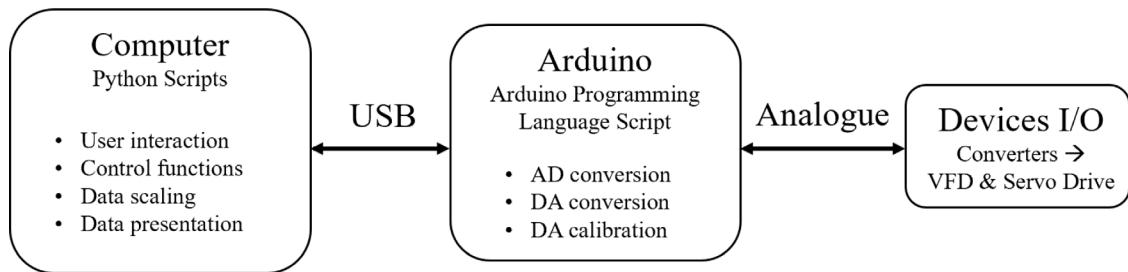


Fig. 30. Communication between hardware and software.

When the I/O module is finished, power it up, connect it to a computer and upload the Arduino sketch `arduinoMain.ino` onto the Arduino controller. As soon as the sketch is uploaded, initiate the `pythonMain.py` or `pythonMain.ino`.

6. Operation instructions

The module was designed with the control of a servo drive setup in mind but can be altered to fit other application areas. The programming employed is twofold and consists of an Arduino microcontroller sketch and a Python control script containing all the logic and commands used to control the devices. In all its simplicity, the Python script forms a text-based user interface asking for input to decide what action to take: Write read from inputs, write to outputs, or perform other miscellaneous functions, as described in Fig. 28. After a received command, the script sends a command to the Arduino and processes any resulting feedback.

The script will ask for user input to decide upcoming actions, and display the alternative options, as illustrated in Fig. 29.

The Arduino sketch can be viewed as the created I/O module's firmware and is not to be altered once completed and uploaded, as it merely acts as an intermediary device and a translator between the Python script and the analogue values. Fig. 30 illustrates a simplified version of the control signal path and its communication media.

Before going into details, the key to understanding the process shown in Fig. 30 is to know how the USB transmits data between the Arduino and the Python script. The USB is a serial communication protocol, meaning it can only send one message at a time, albeit with a speed that enables the illusion of it transmitting multiple messages simultaneously. The same goes for the USB communication between the computer and the Arduino. The transferring process is described in the later sections and utilises the essential serial port read and write functions found in Python's `serial`-library and Arduino's integrated `Serial()`-functions.

6.1. USB transmission structure

This section merely describes how the communication between the Python scripts found as `pythonMain.ipynb` or `pythonMain.py` communicates with the Arduino running the sketch `arduinoMain.ino` (available at Zenodo[1]/GitHub[2]). All communication between the Python script and the Arduino happens through the USB, using functions created for serial communication. However, one weakness in Arduino's serial read functions is the excessive use of timeout-based solutions to

axbxc~

Fig. 31. Python to Arduino message example.

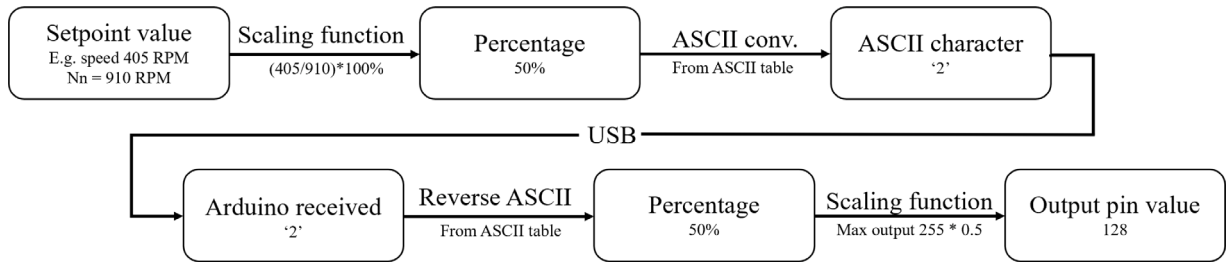


Fig. 32. Example of a 50% speed set-point value conversion from Python to Arduino.

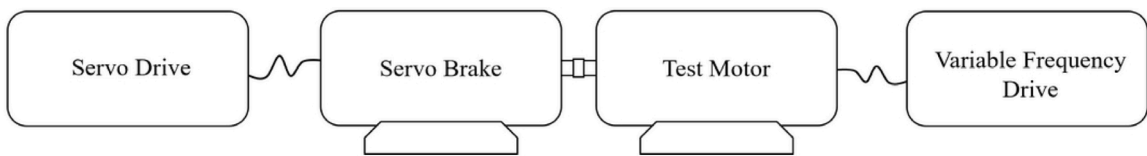


Fig. 33. Servo drive test stand.

determine the end of a serial message consisting of multiple elements, for example, a string. This weakness can lead to issues when attempting to read, e.g., something as simple as a number with multiple digits, as Arduino perceives it as not one multi-digit number but multiple one-digit numbers. When reading from the serial buffer into a String, the Arduino waits until the read function times out to determine if the string has ended. The timeout is often set to 1000 ms, which is too slow for most control-related purposes and leaves the Arduino occupied even after receiving the entire message. One can reduce the timeout duration to minimise the dead time, but that leaves the risk of ending the read function too early and hence end up with only partial messages received. Therefore, to avoid the timeout issue, the serial communication from the Python script to the Arduino in this setup is configured to send and accept only one character at a time, whereas Arduino manually stores them in an array until a terminating character is received.

Most messages sent from the Python script to Arduino are based around three individual characters sent one by one, separated by the character 'x' and ended with a termination character, chosen to be '~'. A typical example of a message structure from Python to Arduino is therefore:

In these messages, the first character, in Fig. 31 'a', tells Arduino the primary intent behind the command, whether to write values to the output pins, calibrate its maximum values or any of the other implemented functionalities.

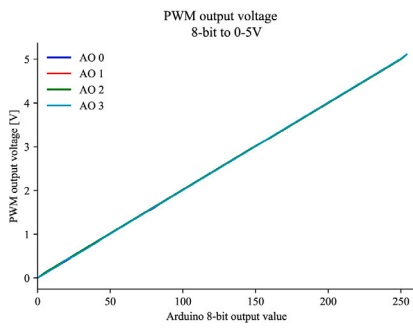
The second character, 'b' in Fig. 31, decides which channel number the message applies. For example, 'ax2' as the initial characters tell the Arduino to write *something* to analogue channel number two. That *something* is decided by the final character 'c', and requires a more thorough explanation to comprehend fully.

As the Arduino cannot read strings without a function including timeout, as stated previously, multi-digit numbers can be challenging when wanting to write to analogue outputs. As a solution, numbers can be converted into single characters using their corresponding ASCII characters and sent as a single byte. ASCII code is the numerical representation of a character and was originally created to represent characters as numbers due to computers only understanding numbers [9]. However, one big drawback is the limited number of available ASCII characters, causing the numerical range and resolution to be limited. All numbers to be sent as an ASCII character are therefore scaled to integer percentage values as a workaround, meaning all values sent from Python to Arduino is within the range 0–100. When Arduino receives the percentage value encoded as an ASCII character, it decodes it back to its integer value and scales it to fit the analogue output values, which in the 8 bit PWM outputs are in the range between 0 and 255. Fig. 32 shows an example of the conversion process where the set-point of a nominal 910 rpm motor is set to 405 rpm.

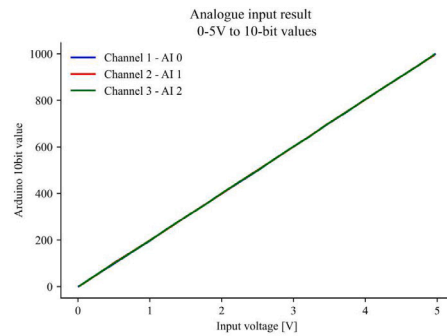
7. Validation and characterisation

The interface was built to control a servo machine test stand consisting of two asynchronous motors controlled by a Variable frequency drive and a servo drive. The test motor runs at a fixed speed controlled by the variable frequency drive, while the asynchronous servo-brake applies a programmable, variable torque using regenerative braking. With caution not to overload the test motor, the brake can also run to create the effect of a generator. A simplified illustration of the system setup is shown in Fig. 33.

The servo drive and the Variable Frequency Drive both include analogue I/O modules, which are non-vendor-specific and rely purely on analogue voltage- and current levels. As the drives are equipped with different, proprietary control interfaces, the analogue

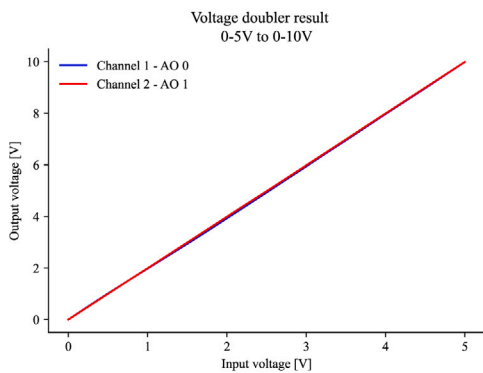


(a) Output voltage vs input 8-bit value.

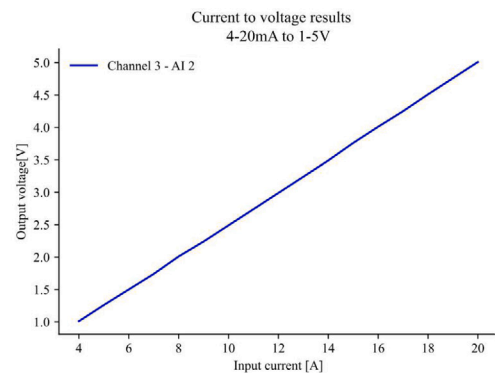


(b) 10bit result vs 0 – 5 V input voltage.

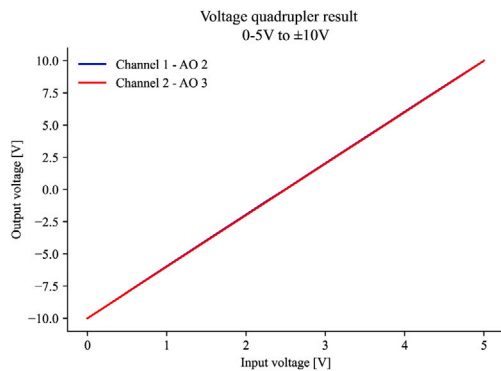
Fig. 34. Arduino voltage output and input response.



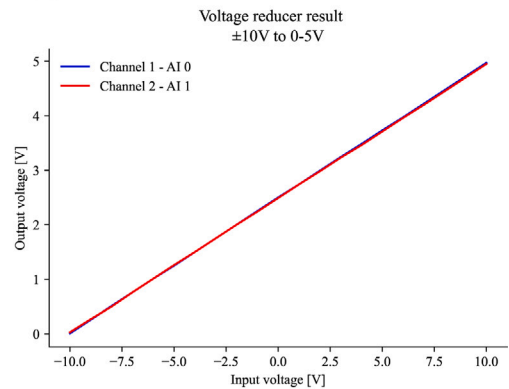
(a) 0 – 5 V to 0 – 10 V converter.



(b) 4 – 20 mA to 0 – 5 V converter.



(c) 0 – 5 V to ±10 V converter.



(d) ±10 V to 0 – 5 V converter.

Fig. 35. Voltage and current converter values.

I/O terminals remain the only ideal communication path to merge the two cross-brand devices into one system. The servo drive allows for speed- and torque control and feedback using a $\pm 10V$ range, while the VFD uses 0–10V for speed- and torque set-points and a 4–20mA signal for speed feedback. Both drives use 24 V digital inputs to control start/stop, direction and operating modes. Using the developed Arduino and converter interface allows for a complete system integration of both drives. The end goal is to implement the stand into various simulation tools using Python as the gateway between the Arduino and the simulation software.

To determine the accuracy of the I/O module, the output voltages were measured and compared with their input values, both before and after the converters. Fig. 34(a) shows the Arduino PWM output voltage compared with the 8-bit value determining the output value. The linear graph proves the Arduino is able to provide a linear voltage increase all the way up until almost 5 V.

The 10-bit Arduino analogue inputs, which are capable of reading voltages between 0–5 V provided similar linear results as seen in Fig. 34(b)

The converters were tested for their accuracy, and as can be seen in the results are very linear. Combining the numbers from Fig. 35 with the graphs in Fig. 34 proves a very linear decent I/O module, especially taking the very low cost of material into consideration.

Relevant applications:

- Hardware in the loop simulations.
- Sensor and monitoring systems.
- Educational settings, due to low cost and easy accessibility.

Capabilities:

- Affordable DAC/ADC and Data Acquisition.
- Adaptable and expandable layout up to ± 15 V signals, depending on the chosen power supply unit and converter circuits.
- Up to 13 digital in/outputs (if no analogue PWM outputs), 6 analogue inputs and 6 analogue outputs, depending on the Arduino of choice.

Limitations:

- Resolution on analogue output is limited to Arduino's 8 bit PWM resolution, and even further limited by the 0-100 ASCII conversion from Python.
- Analogue input resolution limited to 10 bit for Arduino Uno.
- (For now) only controlled by the text-based user interface.

CRedit authorship contribution statement

Anniken Semb Kvalsund: Conceptualization, Methodology, Software, Writing – original draft. **Dietmar Winkler:** Supervision, Review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

References

- [1] Anniken Semb Kvalsund, Dietmar Winkler, Electric drive arduino interface (Zenodo repository), 2022, <http://dx.doi.org/10.5281/zenodo.6762572>.
- [2] Anniken Semb Kvalsund, Dietmar Winkler, Electric drive arduino interface (GitHub repository), 2022, [Online]. Available: <https://github.com/OpenSimHub/ElectricDriveArduinoInterface>.
- [3] CircuitLab Inc., Online circuit simulator & schematic editor - CircuitLab, 2020-03, [Online]. Available: <https://www.circuitlab.com/>. (visited on 05/11/2022).
- [4] eBay Inc., ACDC switching transformer board, 2014-03, [Online]. Available: <https://www.ebay.com/itm/322743968466>. (visited on 05/13/2022).
- [5] STMicroelectronics, LM324N datasheet, 1999, [Online]. Available: <https://pdf1.alldatasheet.com/datasheet-pdf/view/22756/STMICROELECTRONICS/LM324N.html>. (visited on 05/13/2022).
- [6] Monolithic Power Systems, Operational amplifier basics, types and uses | article | MPS, 2019-09, [Online]. Available: <https://www.monolithicpower.com/en/operational-amplifiers>. (visited on 05/13/2022).
- [7] B. Carter, Chapter 2 - Review of op amp basics, in: B. Carter (Ed.), Op Amps for Everyone, fourth ed., Newnes, ISBN: 978-0-12-391495-8, 2013, pp. 7–17, <http://dx.doi.org/10.1016/B978-0-12-391495-8.00002-7>.
- [8] Fujitsu, Fujitsu A Series Miniature Relay, Fujitsu, 2008, [Online]. Available: <https://datasheet-pdf.com/PDF/A5W-K-Datasheet-FujitsuMicroelectronics-625329>. (visited on 05/09/2022).
- [9] ASCII table, ASCII table - ASCII character codes, HTML, Octal, Hex, Decimal, 2022, [Online]. Available: <https://www.asciitable.com/>. (visited on 05/14/2022).



Anniken Semb Kvalsund graduated in the spring of 2022 at the University of South-Eastern Norway with an MSc. degree in Electrical Power Engineering. Her background includes a BSc. in Automation, and she has, for as long as she can recall, been fascinated by the idea of tinkering with electronics and bringing components to life.



Dietmar Winkler works as Assistant Professor II at the University of South-Eastern Norway where he is teaching Electrical Engineering students (under- and postgraduates) in Electrical Power Engineering. His field of research is in modelling of cyper-physical systems and the connection with real systems via hardware in the loop simulations.