CIVO

USB-MIDI TO CV DEVICE
FOR MONOME NORNS

SOME REMARKS

CIVO (short for CV input/output) is an USB modul for the monome Norns/Nornsshield that provides the ability to send and receive CV signals via USB-Midi. It is based on an Arduino Micro.
Initial goal for myself was to build a device that lets me send CV signals from a Norns script and manipulate my modular devices with it. Giving me more possibilities for coding my little Norns and interacting with modular devices. Additionally the device I wanted to make should not be very expensive. CIVO is the result of that endeavour. In this document I will provide the knowledge and an outline of what you'll need to build your own.

Here I am descsribing my version of this device a six CV IN, six CV OUT 0-10V version. But you can use less inputs/outputs or just do outputs or whatever you imagine. This document is more like a general guideline.

!

PLEASE BE AWARE I AM NOT AN ELECTRONICS TECHNICIAN NOR DO I HAVE PROPER EDUCATION IN ELECTRONICS. I AM JUST AN ARTIST WHO IS TINKERING AROUND. I DOUBLE CHECKED EVERYTHING AND EVERYTHING PRESENTED HERE IS WORKING FINE FOR ME. BUT EVERYTHING HERE IS PUBLISHED UNDER THE MIT LICENSE WITH NO WARRANTY AND NO LIABILITY.
PLEASE DOUBLE AND TRIPLE CHECK AND MEASURE EVERYTHING BEFORE HOOKING IT UP TO YOUR GEAR TO AVOID DAMAGING IT.

DEVICE STRUCTURE

INPUT: An Arduino Micro is operating on 5V but eurorack devices operate with up to 12V. Therefore voltage needs to be scaled down on the input to match the Arduinos operating voltaage otherwise it will fry the device. This is achieved with a voltage divider circuit between signal input and analog input into the Arduino. Nothing more is needed on the input side.

OUTPUT: The Arduino Micro (as most microcontrollers) does not have analog output. Therefore we must use a DAC IC (digital to analog converter) to convert a digital signal into an analog one. The output range is limited to the 0-5V operating range of the Arduino and the DAC. Since I wanted a bigger range the output of the DAC is then doubled by an OP AMP to a 0-10V range. (You can just leave them out if you do not need more than 5V). The output of the OP AMPs then goes straight to the output jacks.

Here is a basic schematic of the device's structure:

$$\begin{tikzcd}
\text{CV IN} \arrow[r] & \text{SCALING} \arrow[r] & \underset{\text{ANALOG IN — DIGITAL OUT}}{\text{ARDUINO}} \arrow[r] & \text{DAC} \arrow[r] & \text{AMP} \arrow[r] & \text{CV OUT}
\end{tikzcd}$$

On the next page you'll see the circuit layout for this layout.

WHAT YOU‘LL NEED

PREPARATION

In orderd to built this device you'll need:

● 1x Arduino Micro	~17€
● 12x mono jacks (3.5mm)	12€ (there are cheaper ones available)
● wire	3-4€
● soldering iron	
● soldering stuff and utils	
● micro USB to USB A cable	3-4€ (if you don't have one lying around)
● 6x 22k resistors	~3€
● 6x 33k resistors	
● 12x 100k resistors	
● 3x MCP 4922-E/P ICs (+ sockets / optional)	~12€
● 3x LM358P ICs (+ sockets / optional)	~1€
● 3x 0,1 µF capacitators	less than 1€
● 10 pin eurorack power pinout	0,1€
● circuit board	1-2€
	about ~60€ total (whis is about the same in US$)

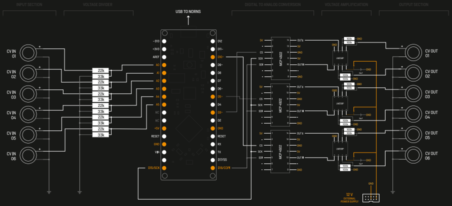
costs may vary depending on your region and availability of parts

Note – Using other Arduino boards:

You can use other Arduino boards for this if you like. Some even have more in/outputs so technically you could add more mono jacks to it for more input/output options. But be sure to use a board that is compatible with the MIDIUSB.h library meaning it has to have built in USB communication since we want it to appear on our Norns as a USB device. Not all Arduinos have that. So be sure to check beforehand. And you may have to tweak the arduino code I've written to fit it your board.

For safety reasons the Arduino micro is powered by the Norns via USB. Only the op-amps are supplied with external 12V.

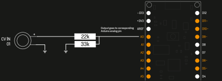
CIRCUIT
LAYOUT



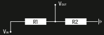
FOR A BETTER OVERVIEW 5V AND GROUND CONNECTIONS ON THE ICs ARE NOT DRAWN. THOSE ARE PAINTED YELLOW AND HAVE TO BE CONNECTED TO THEIR RESPECTIVE 5V OR GROUND SOURCE.

VOLTAGE DIVIDER

SCALING DOWN



To be able to read voltage higher than 5V on the analog inputs, the incoming voltge must be scaled down to the voltage range of 0–5V. This is achieved with a »voltage divider circuit«. It is a simple circuit where two resistors are in row. The voltage that will be scaled down goes through those resistors to ground. The scaled voltage is beeing picked up between those two resistors. It looks like this:

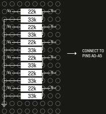


The formula to calculate the resulting voltage (Vout) is:

$$V_{OUT} = V_{IN} * \frac{R2}{R1 + R2}$$

So basically using two resistors of the same value will scale down the incoming voltage by 0.5. I went for R1 = 22k and R2 = 33k resulting in 0.4 as the scale factor. If a voltage of 12V is coming in it would result in 4.8V at Vout. Leaving some headroom if incoming voltage overshoots.

You just need to add this circuit to every CV-in jack. Top illustrations shows the basic layout. Bottom one shows how I aligned everything to keep it tight. Connect to Arduino pins A0–A5.



DIGITAL TO ANALOG
CONVERSION

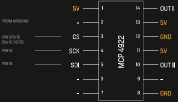
As mentioned above the Arduino Micro does not have analog output. An analog signal is produced via the MCP 4922 IC. To communicate with the chip three connections are necessary:
a clock signal (SCK/ serial clock on pin3 connects to D15 from Arduino), a data input signal (SDI/ serial data input on pin5 from D16) and a channel select signal (CS on pin3 – for each MCP a different from the Aarduino D3, D5, D10) to let the chip know which channel (output circuit) is receiving data. Pin 10 and 14 are outputting the final analog signal. All other pins are for power supply and ground. Pin 2,6-7 and 9 have no connection)

Serial communication with the chip is too complex to explain here. But luckyly there is a library for this chip written by Alexander Hinkel which makes it easy to use. You'll need it to compile the Arduino script and you can download it here (https://www.arduino-projekte.de/download.php?id=12)

The MCP4922 is a 12bit DAC giving us 4096 steps to from 0 to 5V (~0,00122V per step).

Note:
In theory you can add as many of these chips as long as you have enough digital output pins on the Arduino to control the channel select pin of each chip because SCK and SDI signals are shared on all chips. But at least my Arduino Micro was picky and not every digital out worked as channel select.

The MCP4922 works on a 5V level like the Arduino. Higher voltages will fry it. Therefore its output must be scaled up with an OP AMP afterwards.

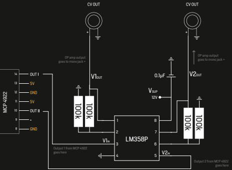


VOLTAGE AMPLIFICATION

AMPING UP

Since the Arduino (and the DAC) operates on 5V it can only output a maximum of 5V. This is enough to modify signals in the modular world, you can skip this section if 0–5V is all you need. But if you want the full range you'll need to amplify the output. This is done with an »operational amplifier«. I used the LM358P because it is cheap and can amplify two signals individually (saving some space). But to amplify a voltage you'll need a higher voltage as supply. This voltage must be higher than the range you want to scale your original range up to. We have 0–5V from the Arduino and we are going to raise this to a 0–12V range. Since I was making a eurorack module out of this I went for the 12V power supply my case already provided. This is sufficient. If you want to use an external power supply instead it should deliver 12V. (Don't forget to get a suitable input jack for your power supply).
For wiring see illustration on the right. V-IN is connected to the output of the MCP 4922. V-OUT goes to the positive pin of the corresponding mono jack.

Note:
Ratio of scaling is regulated via the two resistors at each output. Having two ressistors of same value will result in doubling the voltage range. Output of the Arduino is 5V but it is actually a little more resulting in a range that is more like 0–11.9V (at least that is what I observed).


CV OUT
CV OUT
+
+
OP amp output goes to mono jack +
OP amp output goes to mono jack +
MCP 4922
14 OUT I
13 5V
12 GND
11 5V
10 OUT II
9 -
8 GND
V1OUT
V2OUT
0.1µF
VSUP
12V
100k
100k
100k
100k
LM358P
1
2
3
4
5
6
7
8
V1IN
V2IN
Output 1 from MCP 4922 goes here
Output 2 from MCP 4922 goes here

ARDUINO SETUP

SOFTWARE PREP

Download the Arduino IDE from here:
https://www.arduino.cc/en/software

Download the script here:

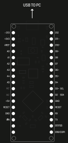
Download the library for the MCP4922 here:
https://www.arduino-projekte.de/download.php?id=12

Connect your Arduino to your computer and open the Arduino script via the IDE. Open the menu at the top called „sketch" hover over „include library" and then click on „add ZIP library". Navigate to where you've downloaded the MC4922 lib ZIP and select it.

Next to the three icons on the top left is a box where you must select your Arduino Micro as target device to connect to it. Do that. Then hit the upload button (again top left, the little circular button with an arrow to the right in it).

And you are done!

CONGRATULATIONS!!!



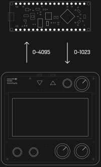
CODE/SCRIPT
COMMUNICATION

If the Arduino detects an input on the CV INs it will send a value. If we want to set a voltage on a CV OUT we need to send a value to the Arduino. Since the resolution on the Arduinos analog inputs is 1024 we will receive a value between 0–1023 representing incoming voltage. And since the resolution of the MCP4922 is 4096 we have to send a value between 0–4095. In both cases 0 = 0V and 1023/4095 = max voltage.

Communication between the Arduino and an external device (in this case a Norns) is done via MIDI USB. We can't send values greater than 128 via midi but since both 1024 and 4096 are multiples of 128 we can split up the value we want to send. The first value send is the multiplicator indicating how often 128 is included in the value we want to send. The second value is simply the rest. For example: we want to set the CV OUT to 2000. 2000 is divided by 128 which is 15,625. 15 willl be the first value send. The rest 0,625 will be multiplied by 128 which equals 80, which is the second value send. And on the receiving end the first value will be multiplied by 128 and the second value simply added. In this example: 15*128+80 which is 2000.
We are using MIDI note and velocity to send those values.

In both directions the Midi-channel on which these informations are sent determines the output (1–6) (or refers to where the input values are coming from). *(Please note: on the Arduino the Midi-channel count starts at »0«.)*

But you do not have to worry about conversion. In the Norns script you just send a value between 0–4095 to the output.



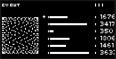
CODE/SCRIPT

SCRIPT OVERVIEW

In the Norns script you have three screens. CV OUT, CV IN and beat settings. ENCODER 3 switches between those screens.
ENCODER 1 selects which value you want to edit.
ENCODER 2 changes that value.
KEY 2 switches between 4 functions indicated by the bars in the top right corner.
KEY 3 executes that function. 1 = set all values to zero. 2=set all values to a random number. 3 = activate beat, 4 = no function yet.

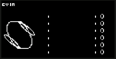
In the CV IN menu you can't change anything. It is just to monitor incoming values.

If BEAT is activated the CV OUT values are generated based on an euclidian/ modulo based system. In the beat screen you have for every CV OUT slot the option to set on which beat it will trigger and what value it should send in that case. These values are additive. You can still change values in the CV OUT screen.



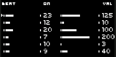
```
CV OUT              III
         + ▬▬▬▬     1676
           ▬▬▬▬▬▬▬▬ 3417
           ▬        350
           ▬▬▬▬     1806
           ▬▬▬      1461
           ▬▬▬▬▬▬▬▬ 3633
```

#1 - CV OUT SCREEN



```
CV IN
              0
              0
              0
              0
              0
              0
```

#2 - CV IN SCREEN

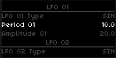


```
BEAT        ON            VAL
▬▬▬         23  ▬▬▬▬      125
▬           12  ▬         10
▬▬          20  ▬▬▬       100
▬            7  ▬▬▬▬▬▬    200
▬           10            3
▬            9  ▬         40
```

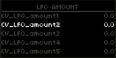
#3 - BEAT SETUP SCREEN

LFO SECTION

In the EDIT menu you'll find that every CV OUT has its own LFO. Every LFO has three options: lfo type (sinus, triangle, square, sample&hold or tangential), period and amplitude settings.
Further down you'll find a section where you can specify how strong the lfo will affect the CV OUT value. The LFO value is added/substracted from the value you've set in the CV OUT screen.



```
        LFO 01
LFO 01 Type          SIN
Period 01           10.0
Amplitude 01        20.0
        LFO 02
LFO 02 Type          SIN
```



```
      LFO AMOUNT
CV_LFO_amount1       0.0
CV_LFO_amount2       0.0
CV_LFO_amount3       0.0
CV_LFO_amount4       0.0
CV_LFO_amount5       0.0
```

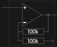
CODE/SCRIPT
REMARKS

The Norns script is a work in progress at the moment. It is a standalone script to simply control the Arduino. I'm planning to add more ideas and functions to it over the time. If anyone is interested in contributing please feel free to do so!

Nothing is done with incoming CV yet.

I am planning to release a standalone library for the Norns in the future. To make it possible to add it to any script you want and send and receive CV with your favourite Norns script.

THANK YOU
FOR YOUR INTEREST!
HAVE FUN!



THIS CODE AND INFORMATIONS ARE PUBLISHED
UNDER THE MIT LICENSE