

final version - v0.989

systole.fr

Welcome curious nerd, poor musician, or whatever you are !

Here is the full kit to master your own *Bidule* :

1. this **MANUAL**
2. an all-platforms editor for your presets and stage performances : **Bidule_EDITOR.html**
Open it with a web browser with Midi capabilities enabled (click on HELP! or see Manual).
Chrome family browsers are the most compatible.
3. the Arduino **firmware** and much more, with an unified code, several instruments definitions, options, tools, examples etc.
You shall #include your instrument "description.h" here : "INSTRUMENTS/BIDULE_SETTINGS.h"

Le BIDULE is an attempt to bring together, in a resilient open source project that is easy to build: a versatile Midi controller and processor that can mimic different behaviours of real instruments, and a synthesizer. The whole thing is compact and self-contained, but with so many possibilities and settings that a multi-platform editor was necessary.

It has no specific design, almost no electronic component apart from an Arduino board; only tacks and wires for the simplest version. Depending on the instrument, complex combination of effects can give unpredictable results; but once you have chosen the presets that suit you, *Le BIDULE* becomes a stable musical tool that will not disappoint you !

Since it began as a fake wind instrument, and because my goal was also to prove that you can do anything with the limited capabilities of the Arduino, this family of instruments is monophonic. However, in the case of keyboards for example, or other options, *Le BIDULE* is capable of Midi polyphony.

Since it's always possible to add code and invent behaviours, and because some combinations of options will never be tested and may be buggy, I consider this version 0.99 as a final version. However, once you build the instrument and make the presets you need, it becomes a stable music companion. I'm glad you're reading this manual after all these years of work. Good luck, and don't fuck it up !!!

Matthieu Metzger



GETTING STARTED

Although our firmware allows for a wide variety of behaviours, here are the basics :
plug it, change presets, start taming your Bidule.

POWER

Power your Bidule with an Usb cable, or a regular Arduino power supply.

Touch detection will react better if the device is grounded - wireless instruments may be less sensitive.

To achieve the "touch magic", calibration happens after Power On.

Please do not touch the instrument during the 2 seconds calibration.

If some keys do not work, just "turn it off and on again".

PRESETS/BANKS

The huge 1Ko Arduino EEPROM memory allows us to store 25 presets:

- there are five BANKS from A to E,
- with each five PRESETS from 1 to 5.

I found this old synth presets style quite convenient. I personally use a bank for each band I play the Bidule in.

You can change your presets with the Bidule's editor window, sending directly a Midi Program change to your Bidule via USB or, of course, with the famous 'Menu' button.

MENU BUTTON

This unique Menu button will be your best friend

quick press

- Press it quickly once, you get Preset 1
- Press it quickly twice, you get Preset 2
- and so on until 5

preset/bank selection

Some of the tactile keys of you Bidule become shortcuts, named "A,B,C,D,E,1,2,3,4,5" when Menu is pressed.

If you need to select a specific Preset or Bank, or both:

- press and maintain the Menu button
- touch and release the desired Bank key if desired
- touch and release the desired Preset key if desired
- release Menu button

trick: try it connected to the editor, in order to see the results

slide transposition (if enabled)

All the instrument's designs can be very different, so try it at home before the first gig with your Bidule !

Octave transposition

- press and maintain the Menu key
- touch slide from 1 to 5, or 5 to 1
- release the Menu key

Semitone transposition

- press and maintain the Menu key
- touch slide from A to E, or E to A
- release the Menu key

long time press

In some cases you will want to recalibrate your instrument, recall the base sound, or save the current preset changes (transposition for example) without going through a computer.

- Calibration : boring 4 seconds Menu press
- Base sound : touch a key during a boring 4 seconds Menu press
- Save : touch a key during a boring 6 seconds Menu press

geeks

- with custom setups, the quick press increments presets from A1 to E5 without worrying about banks.
- if Backward is set at compilation time, a defined key press with the menu button decrements presets

EFFECTS CONTROL

Depending on your instrument, you'll have :

- up to two "Control" buttons (touch or switch),
- up to two "Sensor" buttons (analog sensor, touch, or switch)
- up to two touch zone with pressure values, "Touch"

You can alternatively activate these controllers through MIDI :

- CC#1 for Control I
- CC#2 or Channel Pressure for Control II

FXs can also be set to ON all the time, or even triggered thanks to a split note, or a delay time after playing.

Take a look at "Split/Time" in the [MAIN] section.

INSTRUMENTS BEHAVIOURS

WIND

This behaviour was the first one.

In fact, for a long time, the Bidule was just an electronic bagpipe, built for my own needs, so that I could control a talkbox while playing saxophone fingerings. This was inspired by my late Casio DH-100, which had this usefull breathless mode. Then it gave up the ghost.

I found it fun to build my own, then add a breath controller, settings, an interface... But the program was too big even though it wasn't finished, and the bugs were becoming audible. When the Ictus Ensemble did me the honor of ordering me instruments for a creation with François Sarhan and William Kentridge, I started over from scratch, with many more possibilities, more efficient programming and above all, a lot of customizations and different instrument behaviours. And here we are !

The sax fingerings are inspired by Yamaha and Casio wind controllers, especially the way removing the first finger allows your to change octave - because I always thought that the octaves on the Akai Ewi where a p-i-t-a :) I also added a lot of microtonal/quarter tones fingerings, sometimes close to saxophone's ones, sometimes easier or more consistent.

- PLUCK/TOUCH : plays only when you touch it
- B(L)OW : blow and move fingers ! it behaves like a regular electronic wind controller
- BAGPIPE/GURDY : ...close to a bagpipe. if Special2 is superior to 0, the hold note will be the last played, instead of the open fingering one
- SYNTH : a chromatic and polyphonic (midi out only) keyboard
- KIRK-JACQUEMIN : allows you to play two melodie with left&right hands, as Roland Kirk did. The idea of that behaviour was given by another music genius, Pierre-Hervé Jacquemin. The base note of each hand can be choosen with Special 2 & 3. The audio output plays only the upper hand, whereas the midi output plays both hands, on separate midi channels though.

Please note that any value superior to zero for 'Special 1' will turn microtonal fingerings off

STRINGS

Not really a guitar nor a violin, but a funny tactile slide e-guitar.

For each preset, the editor's Special settings allow you to tune each virtual string independently if needed.

- **PLUCK** : the Bidule will play until you stop touching it. Not good for short notes, but nice for trills and tapping
- **B(L)OW** : the Bidule will play only when you touch the "string" . Good for rhythms and repeated notes
- **SYNTH** : the Bidule acts like a chromatic fretted keyboard, and the "strings" change the base note of the keyboard
- **BAGPIPE** : a never ending slide guitar / if Special1 is set to 0, the hold note will be the last played, instead of the open string note
- **KIRK-JACQUEMIN** : same as bow, with a chromatic keyboard on left hand

nota bene : Synth & Kirk-Jacquemin have polyphonic midi outputs

KEYBOARD

- **PLUCK & SYNTH** : a regular keyboard
- **B(L)OW** : should behave like a monophonic keyboard
- **BAGPIPE** : monophonic keyboard, but neverending of course
- **KIRK-JACQUEMIN** : ?

HARP

- **PLUCK** : a "regular" harp or kalimba, which plays sounds after releasing a contact after a certain amount of contact time, or if you touch a lot of contact at the same time, sounds will stop and won't be played at release contact (in order to imitate the way you stop all the strings with your hands on a harp, or avoid a note to be played)
- **BAGPIPE** : same as pluck, without stopping sound of course
- **SYNTH & BOW** : the instruments behaves like a regular keyboard, or e-bow ; it plays at contact (and not at release)
- **KIRK-JACQUEMIN** : a special behaviour that plays 'Note On' at contact on the selected midi channel, and 'Note Off' plus 'Note On' on midi channel+1 at release. Try it with a smooth attack sound for channel 1, and a more percussive sound for channel 2.

SPECIAL BEHAVIOURS

DRUMS

cheap 1-bit sounds, and associated Midi mapping drums, in order to quickly record patterns with your sequencer or expander

CONTROLLER

turns your Bidule in a midi controller

- by default, it sends a different CC value for each key, in proportion to the pressure
- if 'Special 1' is superior to 0, it sends a unique CC (the one selected with Special1), but the whole instrument becomes a precise slider

note on Bagpipe

at some point, the music has to stop ...

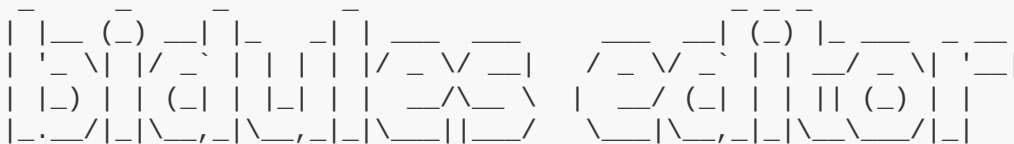
... So just press your good 'Menu' button!

in this specific case, it will stop without changing preset

Other instruments

Hurdy Gurdy, Harp and tactile Horn are almost done.

Any of your geek friends will be able to easily invent a new one, by taking a look at existing ones.



BIDULES EDITOR

Bidules Editor only needs a web browser, since it's a single html page. If you lost the file, here it is:

[<http://www.systole.fr/bidules.html>]

And to save it to your computer... just press ctrl+S (or cmd+S on Mac).

It works with Chrome, Chromium, Vivaldi, Brave, Edge, on Linux, Mac, W!ndo\$, Android, iOS... Just open the html file without doing anything else. You may have to enable the Web Midi Api or Experimental features on Chrome: [<chrome://flags/#web-midi>]

If your USB Midi device is not recognized, you may specify a part of its name by clicking on the blinking 'add one ?'.

By typing keys when nothing is selected, you'll also be able to send midi notes to your Bidule, in order to try the sounds, or transform quickly your computer in a small synth. Left and right Shift key will also send Control I && II controlers.

Fine settings can be achieved with arrow keys, or by zooming on the sliders on tactile devices.

PRESET BANK/NUMBER

There are 25 presets: banks from A to E, with 5 presets for each. To select one, depending on your instrument type, you can:

- click in the editor's list
- push the Bidule's menu button several time
- maintain the Menu button and touch the Bank and or Number keys
- send a Midi program change to the instrument (from 1 to 25)
- type keys in the editor

Keys 1 to 5 : Numbers / 6 to 0 : Banks

Backspace or - / Space or + : next / previous Preset

PageDown / PageUp : next / previous Bank

Home / End : A0 / E5

HELP!

Help about how to enable the mandatory Web Midi API

INFOS

Informations about your Bidule (instrument type, synth, controllers...)

REFRESH

Refreshes both editor & Bidule current preset

PANIC

The classic and useful Midi panic that stops all sounds

STAGE

A special page for stage performance purpose, with easy preset selection and view, midi flow and panic button [!!]

You can even edit the name of your Banks/Presets and save into the editor, to make easier live performances (but it won't be in the Bidule).

FACTORY

Some presets to try sounds, ideas, fxs, plus a dangerous Randomizer !

EXPORT/IMPORT

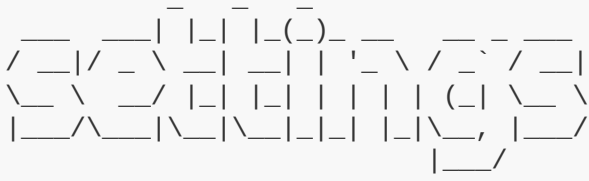
Exports/imports the current settings as a midifile, that you can use here later, keep as a backup, send to a friend, or embed in a midi sequencer to change settings in realtime

COPY/PASTE

Copy / paste the current settings in memory

STORE

After any change, the word "store" appears, and will slightly move until you press it to store the setting in the internal memory



SETTINGS

MAIN

- transpose : transposition by semitones
- behaviour : the different behaviours of your Bidule
- midi channel : enables midi output & sets channel from 1 to 15
16 is reserved for communication with the editor,
and polyphonic Bidule chain
- program : optional midi program change on preset load
- temperament : temperament (applied to both synthesizer and midi)
- base note : base note of the temperament
- bend range : pitch wheel range, to match midi output and bend fxs

SYNTH

- volume : internal synth volume
- attack : attack time
- release : release time
- fixed length : optional fixed length for each note
- tuning : internal synth tuning
- legato : legato mode for envelope, vibrato, glides, fxs

MODULATION

- vibrato : vibrato depth
- frequency : vibrato frequency
- shape : vibrato shape
- tremolo : tremolo depth
- frequency : tremolo frequency
- external : optional external voltage output

OSC I

First oscillator / pwm & pulse

- steps : division of the period for pwm sound, from 2 to 8. 1 is the special pulse mode
- timbre : width of the pwm or pulse sound. In pulse mode, 7 is random
- sub steps : an optional pwm oscillator, which gives you a natural sub harmonic = sub steps / steps
- sub timbre : width of the optional pwm second sound
- glide : 1st oscillator glide time

OSC II

2nd oscillator / square, clic & noise

- algorithm : harmonic / fixed note / shifter / noise & clic
- A : first value for the algorithm
- B : second value for the algorithm
- glide : 2nd oscillator glide time
- bit crusher : hack the 1st oscillator pin, which crushes nicely

SPECIAL

Optional values specific to your Bidule like strings tuning, or midi cleaning latency.

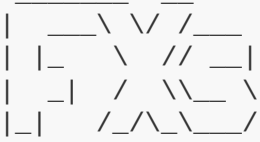
split/time : split note, or delay time, that can triggers an fx. It was inspired by the good old Casio DH-100, whose nice delayed vibrato had to be honored !

latency

Adds a latency, proportionally to 'value', that will clean unwanted notes. This is useful for wind like instruments with midi outputs, release sounds, hold option etc.

Special 1 to 4

- strings : alternate tuning for "string" instruments (0=default)
- harp : tuning pedal positions
- winds : value superior to 0 disable microtonal fingerings



EFFECTS

You've got 3 FXs slots, which allow you to combine a lot of sounds and behaviours. However, please avoid using the same effect on the three slots at the same time, unless you want unpredictable outputs !

SOURCE

"how to enable the effect"

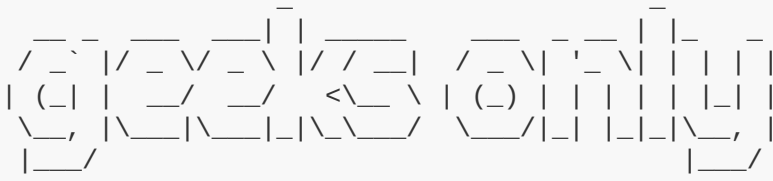
- off : off
 - on : always on
 - control I : 1st control if built (contact, pedal...)
 - control II : 2nd control if built
 - sensor I : 1st sensor (knob, breath...), toggle control I if not
 - sensor II : 1st sensor, toggle control II if not
 - touch I : 1st touch pressure value, invert control I if not
 - split/time/touch II : see below
1. trigger by split/time range if enabled,
 2. or 2nd touch if configured,
 3. or "Invert control II" if not

VALUE

Value of the Fx, shown after the slider (number, notes, percentage depending on the fx.

TYPE

- tremolo : tremolo depth change
- tremolo freq. : volume tremolo frequency change
- vibrato : vibrato depth change
- vibrato freq. : vibrato frequency change
- external : optional external output change
- glide I : glide I value change
- glide II : glide II value change
- bit crusher : switches temporarily the bit crusher
- osc II trigger : enables the 2nd oscillator with 'value' superior to 64, it acts as a toggle
- osc II pitcher : pitch bend of the second oscillator, +/- one octave with various settings
- glitch : if you need unpredictable sounds (depth = 'value'). With value set to 126, a random midi Program Change will be sent at trig, and at 127, unpredictable Program Change will occur ('Program' value = random range)
- dynamics : dynamics (value to 127, 64=random)
- dynamics (low) : dynamics (1 to value, 64=random)
- transpose : do I really need to explain !? (pitch = value)
- bend up / down : pitch bend up or down, on both synth & Midi output
- hold note : hold the notes like a sustain to fake a Midi polyphonic output
- shake : a musical tremolo, speed selected with value - doesn't work well with Latency enabled
- trigger : repeater, double repeater or stopper+custom release (BagPipe mode, harp...)
- random pitch : random pitch for each note, range set with value
- random octave : random octave for each note, range set with value
- harmonics : transpose to false tempered harmonics, proportionally to 'source x value' depth
- micro/macro : transforms the pitch of a semitone according to value, by 1/32 semitones. It also affects the Midi Output, a cool tool to play microtonal ! As it changes all the range of the instrument, you may adjust it with the 'transpose'
- joe's keyboard : simple midi mirror, Zawinul's style !
- xtra midi note : sends a midi Note On on channel+1 (pitch = value)
- cc (switch) : sends a midi Controller (CC=value, on-127 / off-0)
- cc (linear) : sends a midi Controller (CC=value, depth=source)
- modulation cc : sends a midi Modulation Controller (depth = value)
- temporary preset : switches (until release) to the 'value' preset. Save before trying !
- switch to preset : switches to the preset defined by value. Save before trying !
- to last preset : switches to the last preset. Save before trying !
- next channel : temporarily plays midi notes to the next channel. Useful with a Split source for example



GEEKS ONLY

OPTIONAL OPTIONS :)

For those knights of modern times who would have built their Bidule on their own, these options shall be useful.

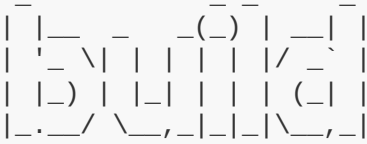
SERIAL CONNECT

If you haven't flashed the UNO with Systole/MocoLufa firmware, if you use an old Arduino 2009 or a cheap copy with CH940 serial controller, or just want to keep a serial connection for programming/debbuging, the editor won't of course detect a Midi device... and "connect to serial ?" will begin to blink :

Just click on it, select the device and choose the desired speed or press enter. The editor would normally connect and behave as with a regular midi device. You can even connect the instrument to a midi input thanks to the "thru" tab. You may have to enable Experimental Web Platform features in your browser.

SET TO SERIAL

Sends a reset message to midi devices flashed with the special Mocolufa+Systole firmware, in order it to behave as a standard Serial USB device and upload a new version of the Bidule. Select everything with Ctrl+A, then "serial reset" appears in the device's menu. Click on it and re-plug your Bidule.



BUILD

The firmware is an Arduino based program. You'll need to compile and upload it to the board. Feel free to adapt it to powerful processors or old Z80 !

Requirements

- Arduino IDE 1.8 - <https://www.arduino.cc>
Other versions may also work, but some generate a too big firmware and stop with an error message
- download & unzip the [Bidules files](#)
- in a text editor, open : INSTRUMENTS/MY-INSTRUMENT.h
- Include the configuration file by typing: `#include "Your_Instrument_File.h"`

put `//` at the beginning of the other `#include` lines to disable them

- You can create a new one by studying the options below and the examples
- Open the Bidule_Firmware_X-XX.ino file in the Arduino software
- Connect your board and select it
- Compile and upload (ctrl/cmd+U - install the required libraries if asked)

BUILDING

Each instrument has its configuration file that defines everything. Thanks to all these options, you can build a lot of different instruments, from the midi processor/interface or the analog standalone synth, to a full playable Bidule with sensors, pedals, capacitive pressure...

Here are the `#define` instructions for the configuration files. Please have a look to the existing ones. Depending on your project, you may need to install the TimerFour library - the Arduino IDE will notify you with a nice error message.

Tips

- you can print this tiny file and put it into the instrument, in case of...
- you can even transform it to a QR code embedded in it (see TOOLS folder)
- comment the file (beginning lines by `//`) to explain how to build it

Nota bene

- pin means "hardware pin number" (D0, D1, D11...)
- key means "touchPin number" (0,1,2,3, corresponding to the PIN list)

TOUCH

```
#define MAXPIN          3          total number of touch pins
#define TOUCH_INIT      3, 8      touch initialization
                                   (sensitivity, oversampling/debouncing)

#define PINS            A1,D2,D4   touch pins list

[optional]

#define turns           10         number of pin scan for each main loop
#define EXTRA_SENSITIVITY      increase sensitivity variable
                                   (fast CPUs or few keys needed)

#define LOW_SENSITIVITY      decrease sensitivity
                                   (slow CPUs, switches, or lot of touch keys)

#define LATENCY         (5 - optional)  adds "cleaning" latency only for mono instruments
                                   (useful with wind fingerings for example)
                                   if defined, the latency value is set with
                                   the 'Latency' setting if a number is also defined,
                                   it becomes the default value

nota : this will work with any metal stuff connected to the pin, or any switch between
the pin and GND see [GEEK] for more options
```


MATRIX

Allows to use a matrix of switches as an input (such as second hand computer or music keyboards, or diy switches+array of diodes), standalone or in combination with the tactile keys, PS2 inputs ...

```
#define MATRIX_MAP
```

```
#define NB_RECEIVE 4
```

```
#define RECEIVE_PINS A1,A2,A3,A4
```

```
#define NB_SEND 12
```

```
#define SEND_PINS D6,D5,D4,D3,D2,A5,12,11,D9,D8,D10,7
```

optional :

```
#define MatrixBuffered
```

to handle the notes events by yourself, with ReadKey() in your custom Bidule() function. take a look at the "RawMatrix.h" example in the "CUSTOM" folder.

```
#define REPERAGE
```

useful to debug key numbers and edit the key maps. When defined, open the Arduino serial screen to get the numbers. Remove it for the final instrument, because this option slows down a lot the rest of the program.

tip : be careful of the wiring direction with diode matrixes.
If nothing happens, try to invert RECEIVE and SEND pins

SYNTH

```
#define audioPinA      D11      oscillator A pin
#define volumePin      D13      volume control pin
```

```
#define audioPinB      D12
```

oscillator B pin - if not defined, osc. A will be used but sounds much mor dirty.
tip : place Led A close to the photoresistor, and Led B a few millimeters away

```
#define filterPin      A3
```

PWM output or whatever extension pin originally designed for a filter, but abandoned with no other options = frequency dependant PWM

optional:

```
#define filterServo
```

the filter Pin controls a Servomotor

```
#define filterOnOff
```

the filter will only be On/Off

```
#define filterMinMax  30:90
```

min and max values for Servo ou real PWM
For true PWM, specify the range (eg: 0:255) and please use a PWM pin

```
#define BULB
```

volume control curve for a bulb (legacy synth, A+B -> vactrol)

```
#define INVERT_LAMP
```

inverts the voltage of the volume output

```
#define INVERT_FILTER
```

inverts the voltage of the filter output

```
#define LAMP_ALWAYS_ON
```

faster attack when using bulb vactrols (noisier)

```
#define ENVELOPE_RATIO 7
```

Envelope time multiplier (for Attack & Release)

INSTRUMENT TYPE

#define Bidule_Include "_Keyboard.h"

"_Keyboard.h"

```
#define KEYS          16          number of note keys (touch pin 0 -> 15)
#define baseNote      48          midi note of the first key
```

"_Gurdy.h"

```
#define KEYS          16          number of note keys
#define baseNote      24          midi note of the virtual string
```

"_Wind.h"

```
!! Breath Controller has to be assigned to Sensor I !!

#define FINGERING      "fingerings.h"    file with fingerings
#define OCTAVE          9                octave key

#define EXTRA_RULE

if(touchPin[1])seminote--
1/2 ton lower if Bb key pressed

#define lowest_note      36              lowest note of the fingerings
(to be defined before Bidule_Include, otherwise
the default base_note of the fingerings.h
file will be used)

#define BFLAT            10              B flat touch key (saxophone fingerings)

#define KJ_OFFSET        4              fingers used in Kirk-Jacquemin mode (default 4)

#define send_fingerings    raw fingerings as Control Changes (89,90 & 91)
(Drums and Controller behaviour and full Latency)
for raw fingerings for the editor ("Tool" folder)
```

"_Horn.h"

```
#define harmonics      3
#define pistons         3
#define baseNote       26
```

"_Strings.h"

```
#define nbStrings      X strings
#define nbNotes        X keys
#define stringNotes    55,62,69,76    default midi note for each string

OPTION

#define FINGERING_CLEANING_TIME 100    time in ms to detect a left hand finger
                                      movement after changing "strings"
```

"_Harp.h"

```
#define KEYS            7

OPTIONS
#define baseNote        48
#define SCHEME_LENGTH   7
#define SCHEME_PITCH    12
#define SCHEME           0,2,4,5,7,9,11
#define minStringTime   3
#define maxStringTime   5000

PEDALS

- Celtic / "Palettes" style
  Select natural/sharp with a SCHEME * single switches
  #define PEDALS_PINS    9,9,9,9,9,9,9

- Concert harp / "Pedal" style
  Select flat/natural/sharp with a SCHEME * double switches
  #define PEDALS_PINS    9,9,9,9,9,9,9
  #define SharpsPin      10

- Tactile style

  Select flat/natural/sharp by touching the strings + the "Sharp" or "Flat",
  or both (neutral) switch touching the lowest string resets all strings
  to the selected alteration switch

  #define Sharps          6            pin number for Sharps switch
  #define Flats           7            pin number for Flats switch
  #define SharpsFlatsAreSwitches
  #define SharpsFlatsNoPullUp
```

other

```
#define BEHAVIOUR       1            behaviour by default ("pluck", if not defined)
                                      1-pluck, 2-b(l)ow, 3-synth, 4-bagpipe
                                      5-drums, 6-controller, 7-kirk-jacquemin

#define BIDULED          optional Led pin (can be shared with the switch pin)

#define SPARE_MEMORY     removes drums sounds, glitch & 2nd controler behaviour,
                                      in case of the desired compiled firmware is to big
                                      if not enough, you can also remove TOUCH and BIDULED
```

MENU

```
#define MENU 12 touch menu key
OR
#define SWITCHMENU 7 menu switch pin

OPTIONAL :
#define MENU_STATE LOW fixed pushed state (HIGH/LOW, no detection)

#define MENU_NO_PULLUP for switch + external pullup resistor
#define PUSH_MENU_FIRST_TO_CALIBRATE waits until push the Menu switch at power up

#define MENU_INTERRUPT MENU key will be handled by an interruption
on the SWITCHMENU pin, which means more precision,
but more code. Choose a pin that accepts interrupts

#define numOptionsTouch 10 number of touch keys for Preset Changes and
options
#define numAfterBank 5 split number between Bank and Numbers
#define Backward_Key 5 key for Backward preset

#define optionsPins 0,1,2,3 touchPins for advanced menu (banks,transpose)

#define MENU_TRANSPOSE allows slide touch transpose (semitones,octaves)
#define SILENT_MENU_TOUCH allows to touch keys before pushing "Menu"
#define EGG
#define MENU_NO_INCREMENTATION disables preset incrementation
#define NO_BANK_LOOP A5 -> B1 if defined, otherwise A5 -> A1
#define BANK_LOCK_TIME 5000 locks to the current Bank if selected just
after power up (max time in ms)

#define SWITCHRULE delay(50) code inserted during use of the menu switch

#define MenuLufa Menu button signal from MocoLufa firmware on
Arduino UNO 16u2 communication chip
(MIDI on:0xF4, off:0xF5)

#define MENUSHIFTER Menu button will become a shift key for your
Key Map (only with Matrix, Key_Pot, or PS/2)

#define MenuShiftRetrigger Re-attacks currently played keys when MenuShift
changes (push/pull accordion for example)
Works with any Matrix and KeyPot

#define disableNextPresetTimer 150 Minimum time in milliseconds between program
incrementation with menu switch
```

TOUCH PRESSURE

```
#define TOUCH_RULE_I          0:5    first:last keys for touch pressure value I
#define TOUCH_RULE_I_THR      30      trigger threshold (0-127)

    and/or

#define TOUCH_RULE_II         0:5    first:last keys for touch pressure value II
#define TOUCH_RULE_II_THR     30      trigger threshold (0-127)

#define single_touch_rule_divisor 50    correction when only one key is pressed (%)
```

CONTROL

```
#define OPTIONI              10       option I key
    OR
#define SWITCHI              6        option I switch pin

OPTIONAL:
#define SWITCHI_NO_PULLUP    for switch + external pullup resistor
#define SWITCHI_STATE        HIGH     fixed pushed state (HIGH/LOW, no detection)

#define OPTIONII             11       idem
    OR
#define SWITCHII             7

OPTIONAL:
#define SWITCHII_NO_PULLUP
#define SWITCHII_STATE        HIGH

    !! if not defined, ControlIII becomes a reverse ControlI switch !!
```

CONTROLLER

```
#define CONTROLLERS          list of the default 12 Midi CCs for the Controller mode
                             default: 75,76,77,78,79,80,81,82,32,33,34,35,36,37,38,39

OPTIONAL:
#define firstKeyCONTROLLER    first touch key assigned to controllers
#define nbCONTROLLERS         11      number of touch keys assigned to controllers
```

SENSORS

```
#define VOLTAGE                voltage reference for analog input :  
                                EXTERNAL, INTERNAL etc. (see Arduino manual)  
  
#define ANALOG_DEPTH          1023    analog input depth  
  
#define SENSORI               A0      sensor pin (linear analog) by default / or  
touchPin  
  
for AC sensors  
#define SENSORI_COUNT          10      closing latency  
#define SENSORI_THR            2      opening threshold  
  
for switches  
#define SENSORI_SWITCH                is a simple switch  
  
OPTIONAL:  
#define SENSORI_NO_PULLUP            for switch + external pullup  
#define SENSORI_STATE                HIGH    fixed pushed state (HIGH/LOW, no detection)  
  
tactile  
#define SENSORI_TOUCH                simple touchPin value  
  
#define SENSORII              3      sensor pin (analog) by default / or touchPin  
  
!! if not defined, SensorII  
    becomes ControlI toggle !!  
  
for AC sensors  
#define SENSORII_COUNT          11      idem  
#define SENSORII_THR            4  
  
for switches  
#define SENSORII_SWITCH  
  
OPTIONAL  
#define SENSORII_NO_PULLUP  
#define SENSORII_STATE          LOW  
  
tactile  
#define SENSORII_TOUCH
```

CC pots

For those that also need to send a lot of Control Changes

```
#define MAXCCPOTS      4          number of pots
#define CCPOTS         A0,A1,A2,A5 pots Analog pins list
#define CCPOTSNUMBERS  92,83,54,12 Control Change list
```

Optional :

```
#define CCPOTSTURNS    4      cycles between each check (default=1 / higher=slower)
#define CCSMOOTH       2      smoothing value (default=3/higher=smoother)
#define CCPOTSCCHANNEL  4      fixed Midi Channel for CC datas (0 to 15)
#define CCZEROCUT              don't send CC when Midi channel is off
```

Please use linear pots of 10 kOhms or higher,
Use the VOLTAGE and ANALOG_DEPTH options if needed

KEY POT ARRAY

Another option to control presets and commands is potentiometers, array of resistors with switches, or matrix keyboards with resistors into a single analog input.

Then you have to use KEY_POT

example :

```
#define KEY_POT    P1,0,P2,100,P3,200,B1,500,B2,600
#define KEY_PIN    A0
#define KEY_POT_POSITIONS 5
```

optional :

```
#define KEY_POT_SPEED  50      (refresh time in milliseconds / 50 ms by default)
```

KEY_POT_POSITIONS gives the number of positions (or keys etc.)

KEY_POT is a list :

- note or command, then minimal analog value (* X positions)

for more informations on the commands, take a look at "Matrix and PS2 Mapping"

DRUMS BEHAVIOUR

```
#define DRUM_KEY_OFFSET    key number of the first Drum key
#define DRUM_SOUND_OFFSET  offset key number of the first Drum sound
#define DRUM_MAX_SOUNDS    number of sounds used (12 maximum)
```

MIDI on Arduino UNO, Mega

It is possible to flash the 16u2 communication chip of the Uno and Mega board to transform it into a Midi device.

It will also allow to connect extra buttons (Ctrl I, Ctrl II, Menu...) on the ICSP Pins or even connect a PS2 keyboard to the 16u2, that will send all the datas to the main processor

Take a look at the script in the "ModifiedUnoMegaMocoLufa" folder, and then the [SET TO SERIAL] section.

MIDI

```
#define SINGLE_MIDI_CHANNEL_INPUT    8      Channels from 1 to 15
                                         (the Bidule will only respond to the commands on channel 8)

#define OMNI_MODE_CONTROL             allows to control all the settings by CC without
                                         using the channel 15 - use it at your own risk

#define NO_USB_THRU                   disables usb in -> midi processing -> usb out

The following options was designed for 32u4 processors (Leonardo family),
because on Arduino UNO, Mega, Nano... pin 0(rx) and 1(tx) handle Midi by default
(and you can't change it).

#define Hardware_Midi                 6      Hardware midi output on the defined pin number

#define EXTERNAL_MIDI_INPUT   Serial1  Uses the Serial1 (or more) as an hardware input
```

SERIAL

```
#define Serial1noRX                   disables the Serial1 input to free the 0 pin
#define SERIAL1SPEED                  Option Serial1 midi speed (otherwise = standard
31250)

#define SERIAL1SPEED                   56000    serial1 speed for unusual Midi connections

!!! If you have more than one Midi input (most of time the USB Midi input),
    don't use them at the same time !!!
```


TEMPERAMENT

```
#define CUSTOM_temperamentI      custom temperament in cents*100
#define CUSTOM_temperamentII     idem
#define CUSTOM_temperamentIII    idem
```

(eg. Bach-Lehmann: 587,391,196,391,-196,782,196,391,391,0,391,0)

CREATE NEW BEHAVIOURS

```
#define BIDULE_INIT - will call your custom void BiduleInit () at power on
```

```
void Bidule () {} - main routine
```

```
byte touchPin[0 to MAXPIN-1] - current value of touch key (0 to 127)
bool monophonic_instrument
```

```
On (byte note, [ byte expression, int pitch ]) - plays a note
Off (byte note = lastnote, bool to_usb = true, bool music_stop = true,
     bool synth_stop = true, byte tail_stop = p[Release]) - stops a note
```

simply use Off() to stop sound in monophonic modes - and no need to write Off()/On()

```
ReadKey () - use with MatrixNotes - note = bits 0 to 6, on = bit 7, otherwise 0
SpecialBehaviour () - for Drums and CC modes with touchPin instruments
```

In order to facilitate custom codes and regular updates.

```
#define EXTRA_CODE_TOP      MyFunction()    extra code, added before the whole code

#define EXTRA_CODE_BOTTOM          extra code, added between the whole code
                                   and the specific instrument Bidule section

#define EXTRA_CODE_PARAMETERS          extra code for other CC inputs from the
                                   editor on channel 15 (tuning of a CV/Gate
                                   for example) these input variables are :
                                   byte controler, byte value

#define EXTRA_CODE_OPTIONS          things to do in the options routine
                                   (Menu, Control etc.)

#define EXTRA_CODE_INIT            things to launch at startup
```

```
#define TouchBypass :
```

- if you wanna add virtual touchPins (controlled by a Matrix, Key Pots, PS/2keyboard, or MIDI) +SW = Switch, +InvSW = inverted switch
OR
- if you have a lot of switches, and need unusal polarity or more processing speed

EXTERN = triggered externally

example : #define PINS D1, EXTERN, EXTERN, D4+SW, D5+InvSW, D6, D7

This method allows for example to mix the values of a keyboard and real tactile keys in the same instrument, or the use of Drums and Controllers behaviours even with a PS/2 keyboard by routing the keys to the desired values.

!! if you only need "virtual"/EXTERN touchPins, just define the number of pins with MAXPIN, and please don't define TOUCH_INIT !!

```
#define CustomOutput          "INSTRUMENTS/CustomOutput-example.h"
```

Specifies code for custom outputs (CV/Gate, oldies, proprietary protocols, lights...)

```
#define MatrixUserDefined    "INSTRUMENTS/KEY_MAPS/MatrixUserDefined-example.h"
    If the Key Mapping commands are not enough, you can define up to ten
    custom commands (user_key_defined, user_key_defined+1, user_key_defined+2 ...)
```

```
#define DEBUG    Enables debugging
    Use DEBUG(variable or text,true for line break) to serial debug
    and TIMER(milliseconds) { DEBUG("timed debug",false) } to avoid overflow
```

```
#define LOWNOISE 5
    Divisor that slows touch detection to avoid digital noise when not playing
```

```
#define disableNextPresetTimer 400
    Safety duration in millisecond between two presets when using the presets FXs
```

Have a look at the examples, and put additional files there for future updates

KEYS MAPPING

If you are using Matrix, Key_Pot, PS2 or standalone key mapping, you'll have to specify which type of commands you will need, in order to reduce the code size :

```
for Midi notes :  
    #define MatrixNotes
```

```
for presets, menu and control options :  
    #define MatrixMenu
```

You can of course use these options at the same time.

I your matrix array (see examples), give two numbers for each key (normal & shifted):

```
-- none (or 0) for nothing
```

```
-- Midi notes numbers from 1 to 127
```

or you can also write them with their respective names :

```
-- Notes (s for sharp, f for flat)  
    C, Cs, Df, D, Ds, Ef, E, F, Fs, Gf, G, Gs, Af, A, As, Bf, B
```

```
-- Octaves  
    00, 01, 02, 03, 04, 05, 06, 07, 08, 09, 010
```

```
    [ example for C#4 : Cs+04 ]
```

```
-- Virtual/extended touch pins : TouchBypassOffset + number (0 to 55)
```

```
    [ nota bene : D1 switches to Bank D automatically ]
```

```
-- Exact preset : MenuPreset
```

```
    [ example : Preset number 8 (pB3) : MenuPreset + 8 ]
```

```
-- Options  
    single_transpose  
    single_semitone  
    single_octave  
    semitone_plus (long press gives semitone_minus)  
    octave_plus (long press gives octave_minus)  
    semitone_minus  
    octave_minus  
    menu_shift (put the value in normal AND shift mode)  
    increment  
    decrement  
    loop_in_bank_increment  
    loop_in_bank_decrement  
    controlI_toggle  
    controlII_toggle  
    sensorI_toggle  
    sensorII_toggle  
    controlI  
    controlII  
    sensorI  
    sensorII
```

```
-- Presets inside bank
    Pr1, Pr2, Pr3, Pr4, Pr5

-- Presets
    pA1, pA2, pA3, pA4, pA5
    pB1, pB2, pB3, pB4, pB5
    pC1, pC2, pC3, pC4, pC5
    pD1, pD2, pD3, pD4, pD5
    pE1, pE2, pE3, pE4, pE5

-- Matrix User Defined Commands :
    user_key_defined
    user_key_defined +1
    ...

#define PS2MocoLufa    "INSTRUMENTS/KEY_MAPS/Accordion.h"

    On Arduino UNO, the special SystoleMocoLufa firmware flashed to the 16u2
    communication chip (in the TOOL folder) allows to use the 3 extra pins of the 16u2
    as two switch Controllers for example, and a Menu button, or a PS/2 keyboard input.

#define PS2Bidule      "INSTRUMENTS/KEY_MAPS/Accordion.h"

    Same as PS2MocoLufa but with PS2 handled by the Arduino - Will often cut the sound
    if used as a note keyboard, but can control Menu, Presets, Transpose...
    The PS2MocoLufa method is more complex (Arduino UNO+special MocoLufa flash),
    but safer.

#define PS2Mouse        ??????
#define PS2MouseButton1 controlI
#define PS2MouseButton2 05+As
#define PS2XAxis        sensorI
#define PS2YAxis        controlII

mandatory if you use PS2Bidule or PS2Mouse !
#define PS2_DATA_PIN    D4
#define PS2_CLOCK_PIN   D7

optional
#define PS2_SHARED_DATA_PIN - put this if the pin is shared with another one

nota: impossible to use PS2Bidule, PS2Mouse, or MENU_INTERRUPT at the same time...

#define MenuShift        Declares that keys can be shifted,
                          and launch another note or function
                          Then you have to declare 2 datas for each key
                          in your key mapping file

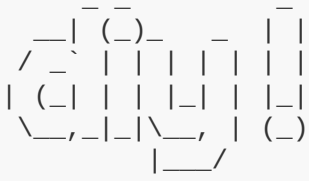
    [ if you put it in the Matrix definition file (recommanded),
      be sure to write it at the end, after the list - see example files ]

#define SpecificCustom 04+C (56)

    Base note of the 5 consecutive semitones that will be changed by Specific1 to
    5 (latency) values to create optional custom pads or pitched keys in presets

#define MatrixBuffered / #define Key_PotBuffered
#define PS2MocoLufaBuffered / #define PS2BiduleBuffered

    Handle the notes events by yourself with ReadKey() in the custom Bidule() function
    Take a look at the "RawMatrix-example.h" example in the "CUSTOM" folder.
```



DIY!

BE ABLE TO BUILD AND REPAIR YOUR INSTRUMENT !

BE ABLE TO CUSTOMIZE IT !

BUILD OTHER ONES !

BE PROUD OF YOURSELF INSTEAD OF BUYING !

Requirements

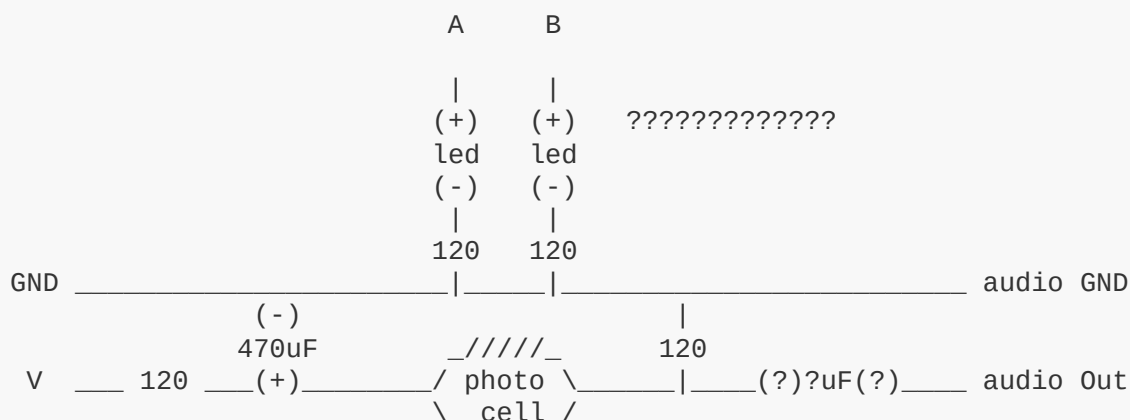
ARDUINO Leonardo (or UNO, Leonardo, 2009, MEGA, clones etc.)

If you already have a PCB, you'll need to solder male and female headers : put them on your Arduino, then put the PCB over the whole stuff and solder. If you use a screw shield instead, or use wires directly or through "dominos", please glue or tape the wires.

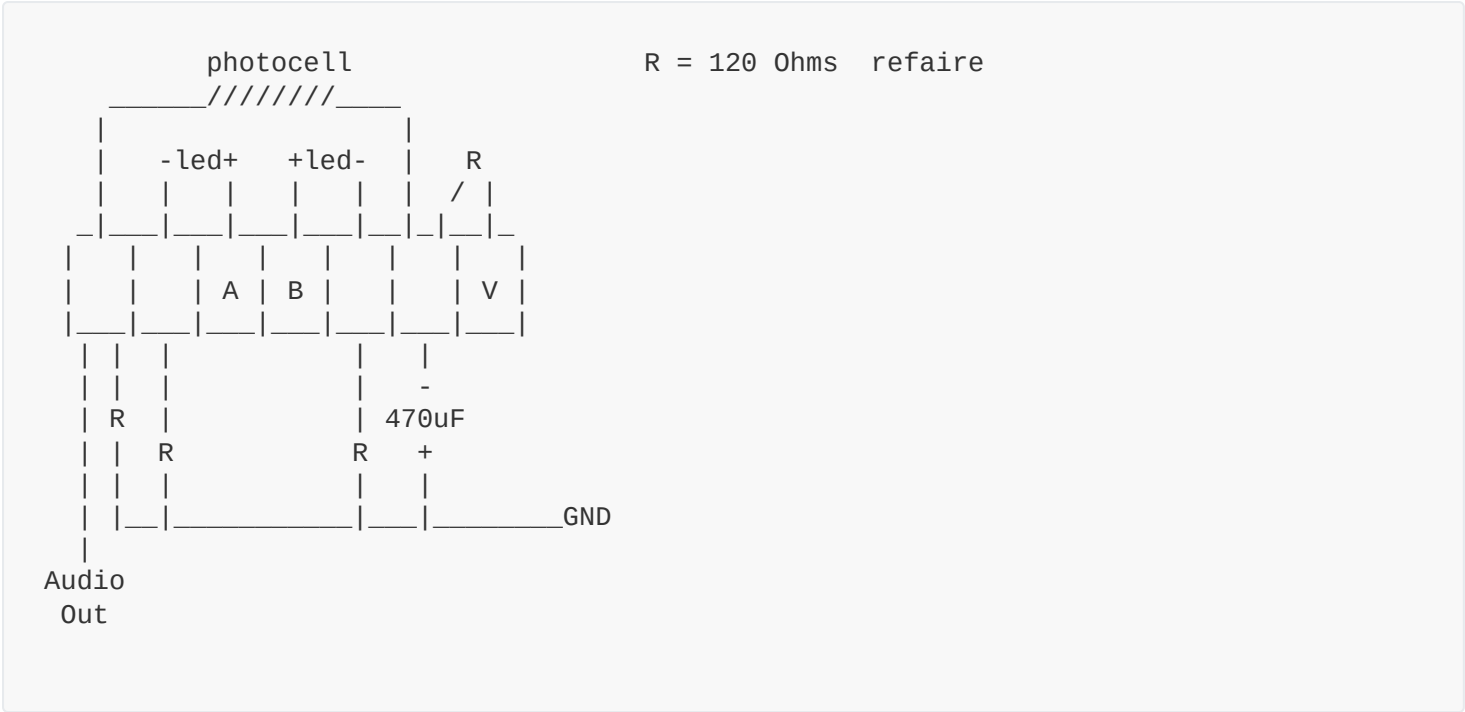
SYNTH

- 3x 120 Ohms resistors (R)
- 2x 470 uF capacitor (C)
- a photocell
- an audio jack
- 2x white LEDs

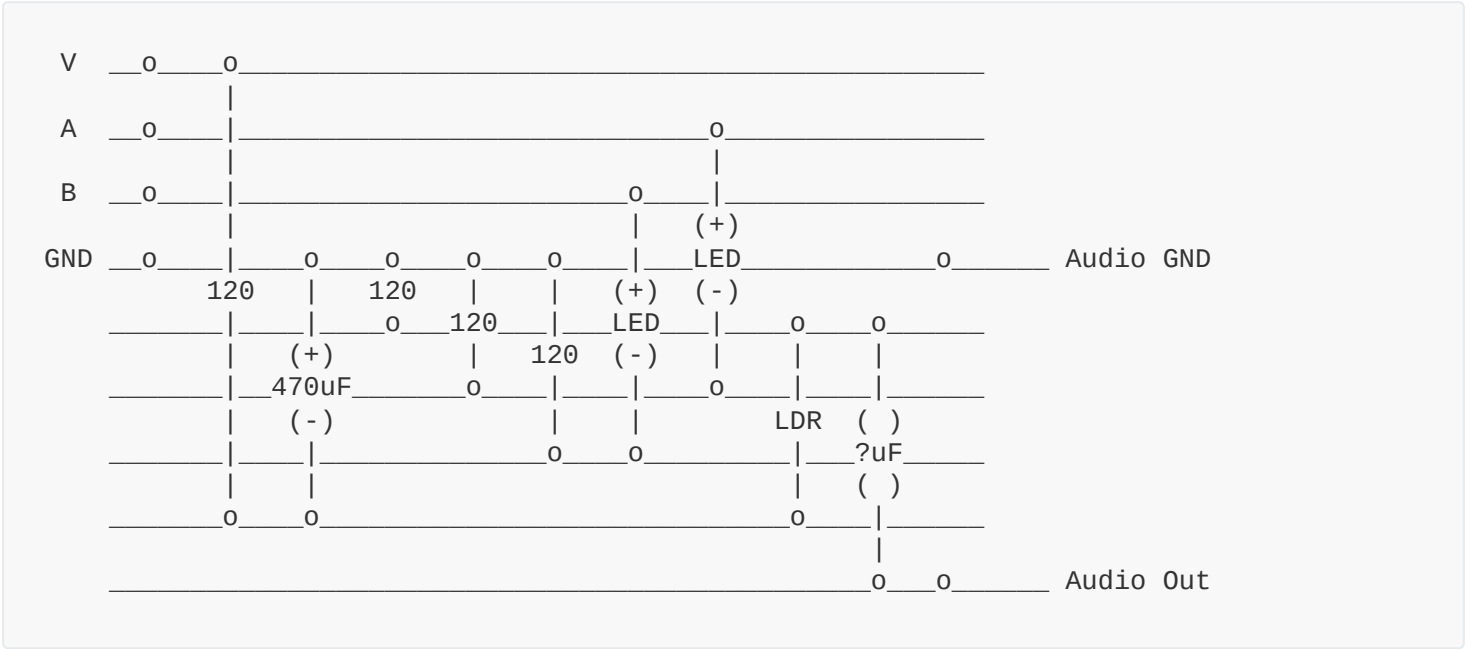
schematic (ajout résistance)



dominos (refaire)

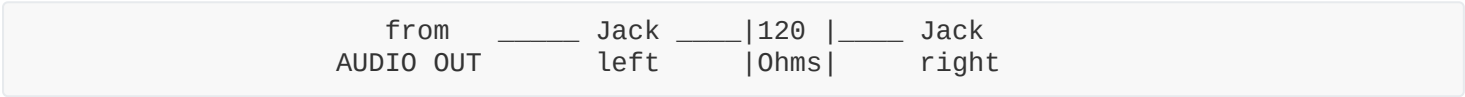


one strip board (11, 12, 13, GND, cond. liaison)



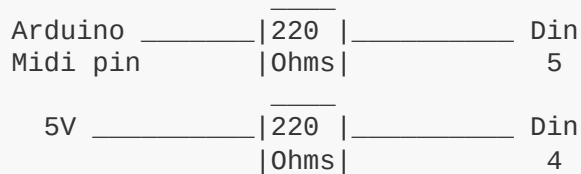
hybrid output

allows stereo and mono connection to work at the same time



MIDI OUTPUT

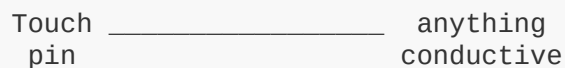
- a MIDI Din jack
- 2x 220 Ohms resistors



switches



tactile parts



SENSORS

electret mic

dynamic mic

on/off

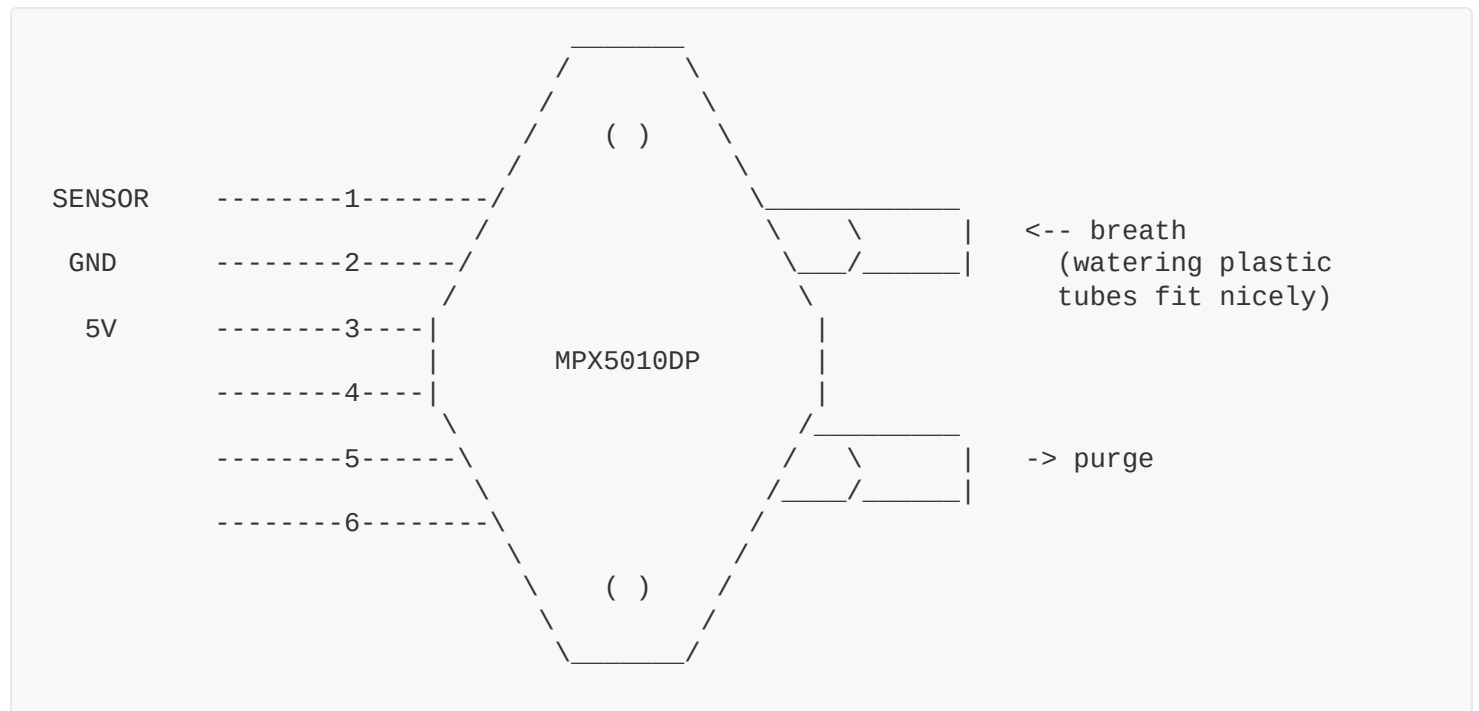
dynamo

potentiometer

breath sensor

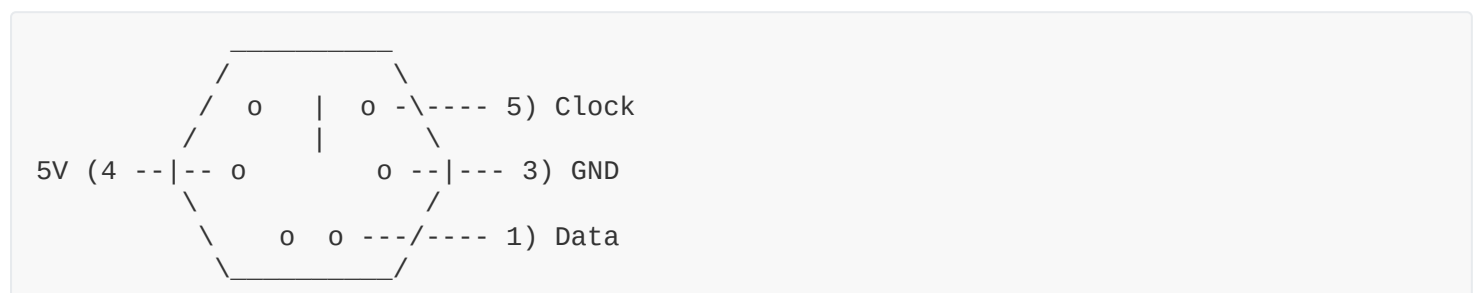
Please take care of the Arduino and other electrical part, since residual water and moisture could happen around the sensor.

The MPX5010DP pressure sensor works fine, other versions may be ok, and SENSOR pin has to be an Analog Pin

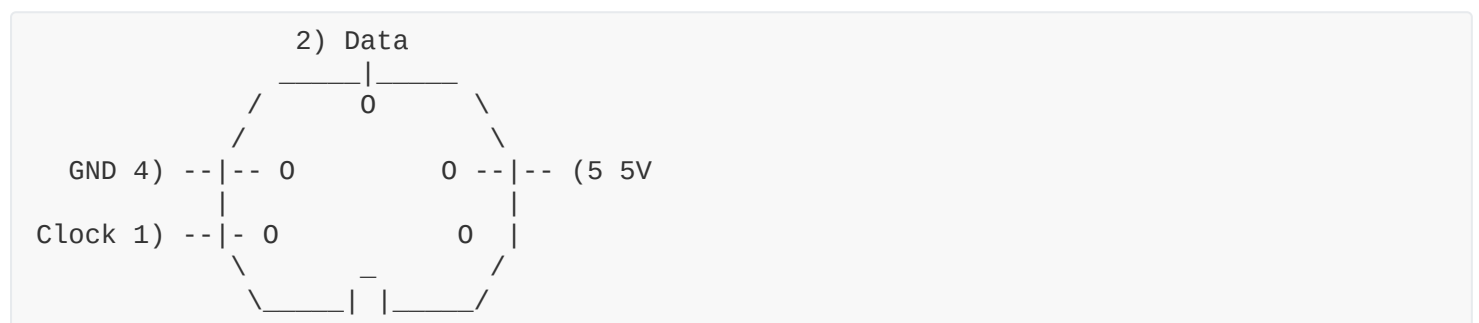


others (front view)

PS/2



AT DIN5



ideas

- use a recycled smartphone as :
 - a midi expander
 - an audio multi-effect
 - a battery
 - a song player while playing your Bidule (thru fxs, midi...)
- put an integrated loudspeaker & amplifier
- put an integrated FM emitter
- stereo or multiple outputs (oscillators on left/right or separate jack output)
- a standalone synthesizer - midi input -> jack output
(may take it's power from the midi plug ?)
- a simple usb/midi interface with midi processor capabilities
- try to change some components values
- make a stereo output with 2 photocells