Basics 03 : SLab Setup

This document describes the setup of the hardware board that will be the basic instrument in the SLab Python project. Next the SLab environment is described.

BOM	1x MCP6002 Opamp 1x MCP6004 Opamp 1x red LED 2x 47kΩ resistors 1x 470Ω resistor

V1.0(19/3/2017) © Vicente Jiménez. License information is at the end of the document

Index

Introduction

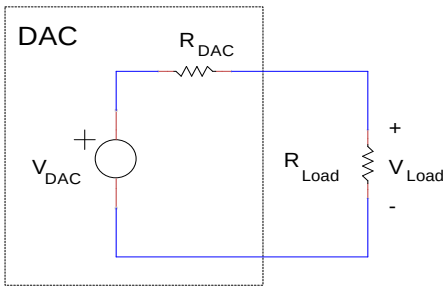
At the end of the previous document we had a working SLab Python environment, we programmed the hardware board with a SLab firmware and we checked that it can communicate with the PC.

Unfortunately, typical microcontroller boards have poor ADC and DAC performance under non ideal conditions. This is due, as we will explain, to loading effects. This document describes a buffer setup that greatly improves the performance of the hardware board.

Need for a buffer setup

Microcontroller (MCU) devices usually include a CPU, flash memory, RAM memory and several peripherals in a single integrated circuit. Most MCU input/outputs work with very low current levels because current drivers take-up too much space. That means that ADCs and DACs are quite sensitive to loading effects. If you are not convinced about that, the rest of this section gives a more detailed explanation. If you don't want to mess with the details, you can skip to the next section.

Loading effects appear when a load takes current from a source. Take as an example the following circuit where a DAC that we use to generate a voltage V_{DAC}, with an output resistance R_{DAC}, is connected to a resistive load R_{Load}.



The voltage at the load will be:

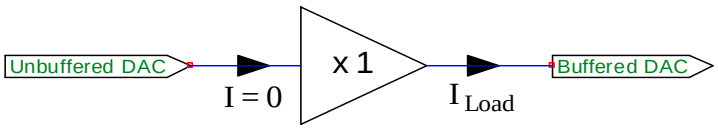
$$V_{Load} = V_{DAC} \frac{R_{Load}}{R_{Load} + R_{DAC}}$$

The voltage at the load won't be the same as the value we programmed on the DAC. Let's calculate the how high the load resistance needs to be so that the error is less than 1%:

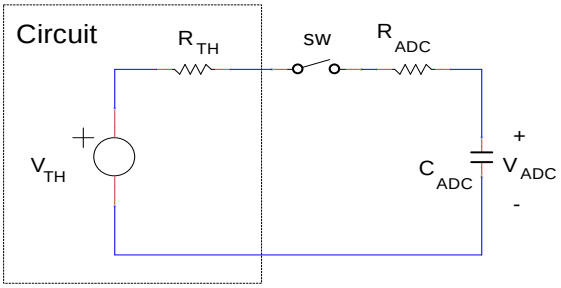
$$\frac{V_{Load}}{V_{DAC}} \geq 0{,}99 \qquad R_{Load} \geq 99 \cdot R_{DAC}$$

The F303RE board features a DAC with a maximum specified output resistance of 15kΩ. That means that, in order to have a less than 1% error, the load must be over 1,5MΩ. Moreover, as in this board the DAC has 12 bit resolution, we need to have an error of less than about 0,01% for it to be negligible. That will increase the load requirements to over 150MΩ. The end result is that, if our budget error is small, we cannot connect the DAC to any load at all.

The solution to the problem is to add a buffer with unity gain on the output of the DAC that does not load it:



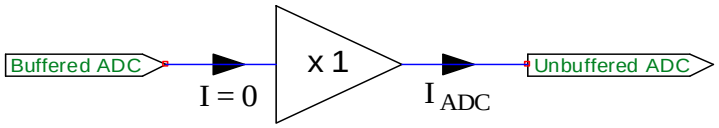
We have a similar problem with the ADCs. The MCU ADCs are used to read the voltage on a circuit node. We can model any DC circuit with a Thevenin equivalent. The model of the ADC is more complex that the one of the DAC, however. Typical SAR ADCs can be modeled as a RC network and a switch. Whenever you want to read an ADC input, you close the switch so that the V_{ADC} equals the V_{TH} voltage we want to measure.



It takes time to load the C_{ADC} capacitance to a voltage close enough to the open circuit V_{TH} voltage. In the case of the F303RE board, the manufacturer recommends R_{TH} to be always below 47kΩ. At maximum sample rates, however, it shall be below 18Ω if we want the full 12 bit resolution of the ADC.

The C_{ADC} capacitor is depleted on each reading so we need to drain some charge, in other words, some current during some time, from the circuit every time we read it with the ADC.

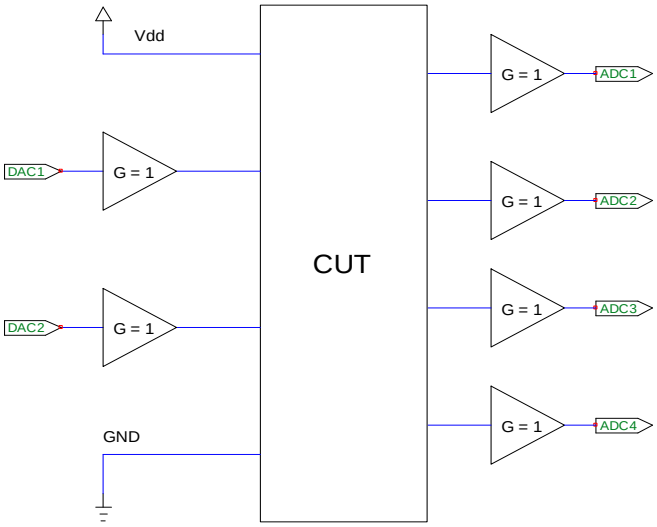
In general, although at low sample frequencies the problem is not as big as in the DACs, we also need to drive the ADC inputs with a unity gain buffer:



That way, we consume zero current from the circuit, and the MCU ADC is driven by a low output resistance buffer.

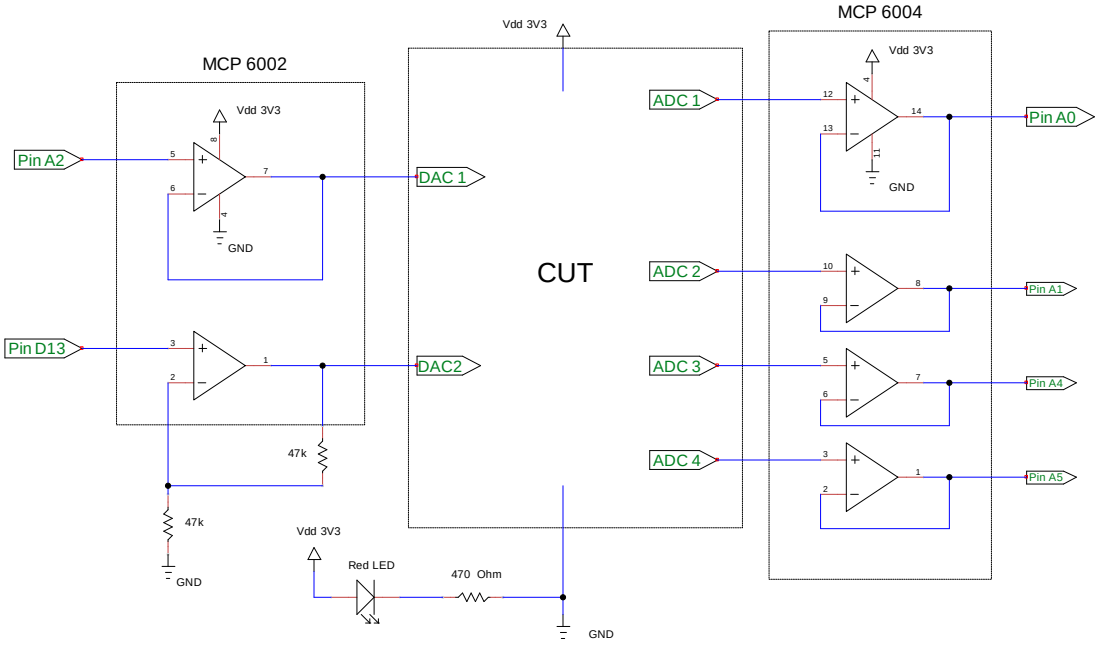
SLab buffer setup

In the end, as we need to buffer the 2 DACs and 4 ADCs, ideally we will need 6 unity gain buffers as shown in the following figure:



The rest of the document deals with the buffer setup in the F303RE or L152RE boards. If you have any other board, refer to its documentation for the details of the needed buffer setup.

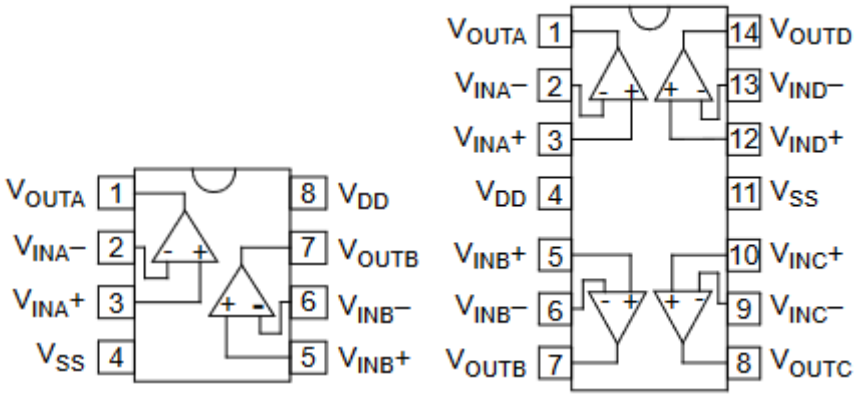
The easier way to implement the buffers is to use follower circuits implemented with operational amplifiers. The following figure shows the proposed buffer circuit for the F303RE board:



We use 6 operational amplifiers: 2 for de DACs, included in a MCP6002 dual opamp chip and 4 for the ADCs, included in a MCP6004 quad opamp chip.
We have also included a red LED and a resistor so that it lights up when the board is powered.

Observe that in the second DAC channel, connected to pin D13, a gain 2 non inverting amplifier configuration is used thanks to the two 47kΩ resistors. In the F303RE board, pin D13 is connected to the user LED. We need to keep voltages at this pin below half Vdd to prevent the LED to affect the DAC response.

Don't worry if you don't understand the circuit. You don't need to understand it to implement its functionality. The following figure shows the two opamp chips we will use. We just need to implement the proper connections.



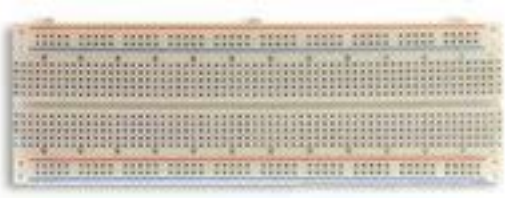
Note the pin numbers, we will refer to them later.

The MCP600x opamps are nearly full rail but they cannot guarantee forcing voltages too close (less than 25 mV) to the supply rails. So, operating next to the rails will be unreliable. Also, the maximum output current of the opamps is about 15 mA when powered from a 3.3V supply. That limits the current we can get from the buffered DAC channels. In our experiments we will try to require less than 10 mA from the DAC outputs to have some current margin.

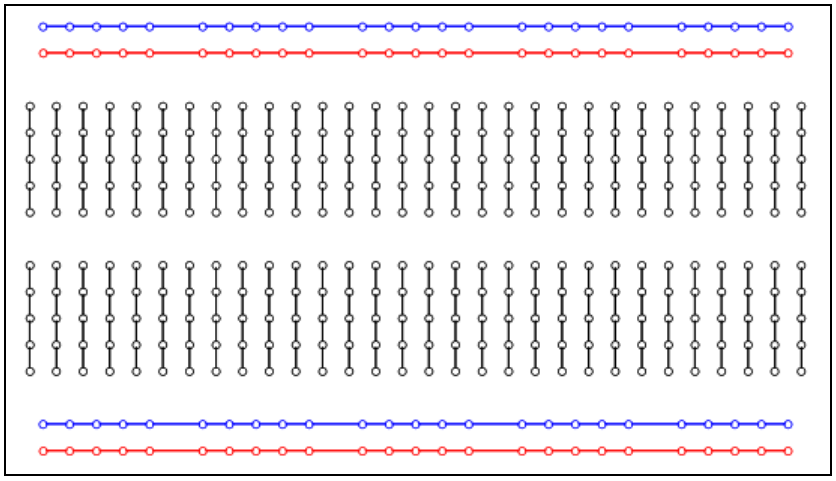
Implementing the buffer setup

As we remember, from previous documents, we will use a solderless breadboard to build our circuits. To ease the build of the buffer circuit, we will also implement it on the same board.

Our solderless breadboard will be something like that one:



The board includes multiple connecting points. Some connecting points are connected together as shown in the following figure.

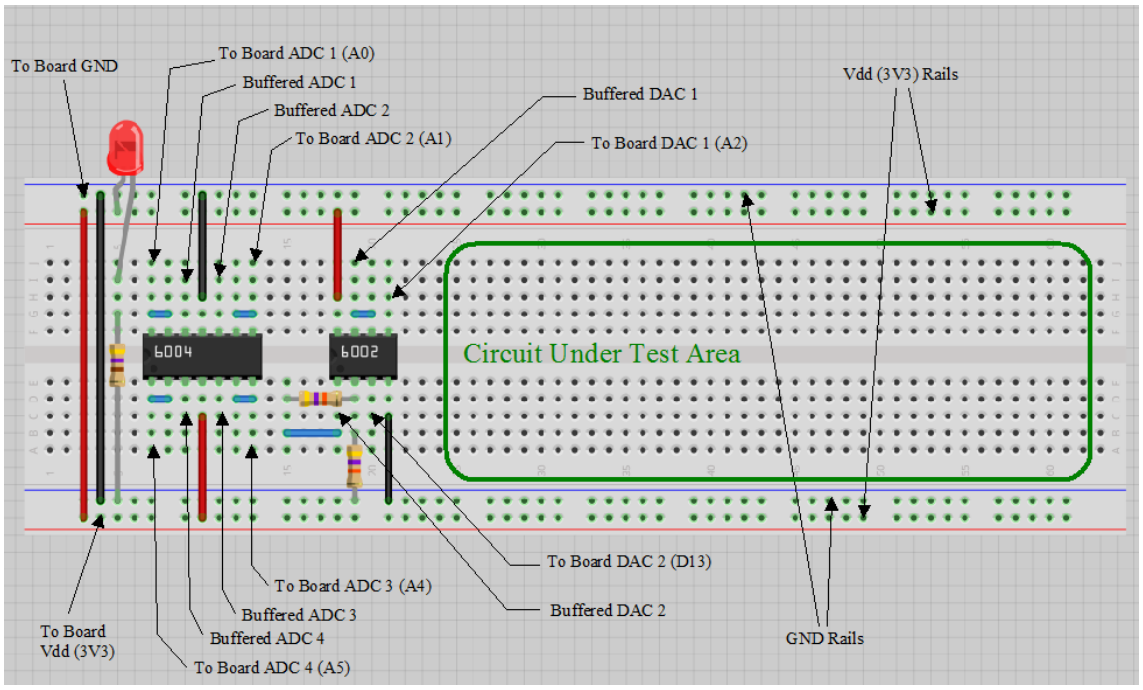


In order to implement our buffer circuit we can put the components over the board as in the following figure. Take your time to see that the connections correspond to the previous circuit schematic.

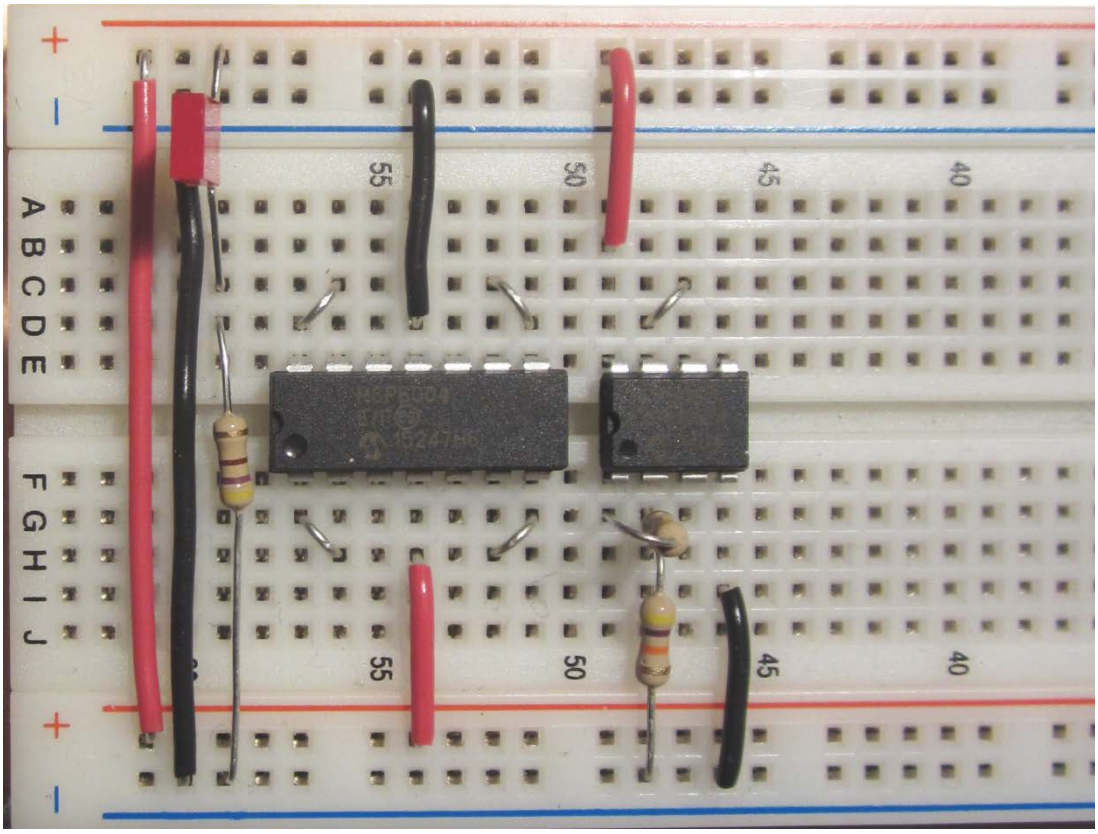
Observe that we have shorted together the upper and lower red Vdd rails and the upper and lower blue GND rails.

In the case of the red LED, the long terminal is the one that needs to be connected to the positive rail. If you connect it wrong, it just won't lit.

If you put the driver components to the left of the board your will have more than half of the board space to implement the circuits we will test in later documents.

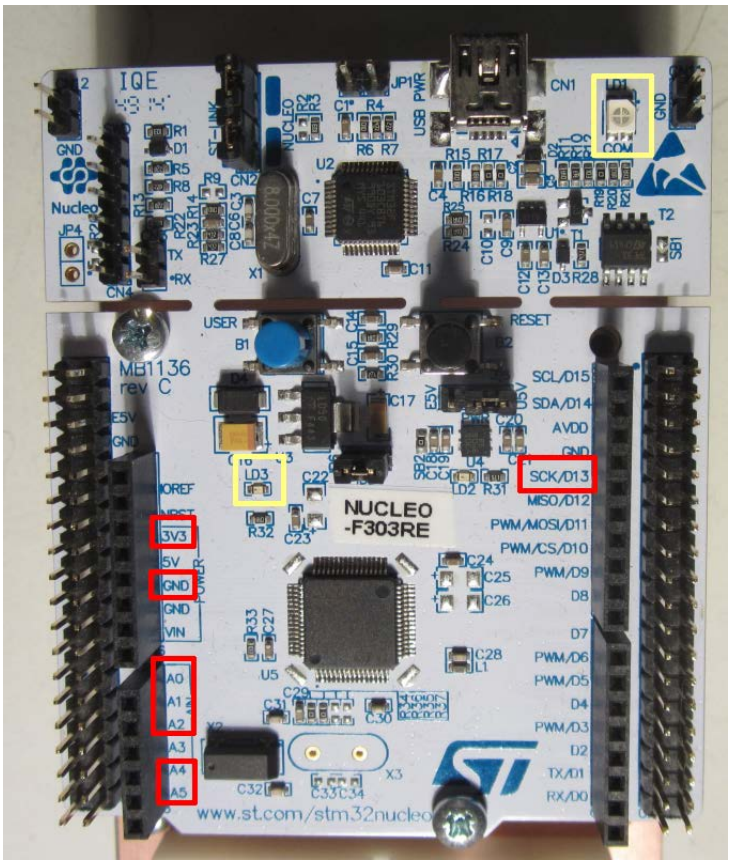


Below you can see a photograph of the breadboard with all the connections. Note that one of the 47kΩ resistors is oriented in vertical to reduce the used space. Note also that in this case the red LED is square. Using trimmed wires instead of jumper wires gives a clean polished result.



Now it is time for you to implement the circuit in the breadboard. Take your time and trim the component terminals if needed.

Next step is to connect the breadboard to the F303RE hardware board. The following figure shows the F303RE board with the relevant terminals, in female headers, marked in red. Two board LEDs, LD1 and LD3 are marked using yellow boxes.



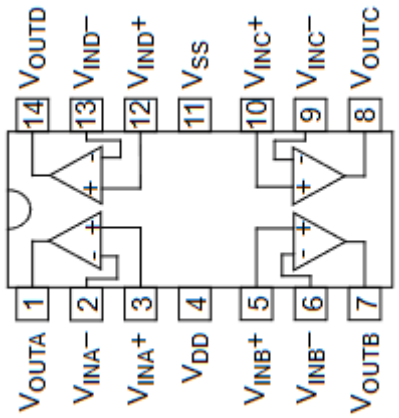
Connect the board to a PC. Led LD1 at the top of the board should light on red or green. Led LD3 in the middle of the board should light in red.

We have to implement 8 connections starting with the power rails. Connect one jumper wire from pin labeled GND (There are several and all are the same) to the blue negative rail. Then, connect another jumper wire from the pin labeled 3V3 to the red positive rail. The red LED should light up. If not, check that the LED is not backwards. Revise the connection until it works.

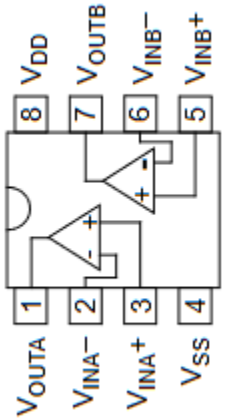
Now, we must connect all DAC and ADC cables from the board to hardware F303RE board to the breadboard.

The following table shows the connections to implement. Use a jumper wire for each one. You can also refer to the previous breadboard figures.

Name	F303RE Pin	Breadboard Pin
Unbuffered DAC 1	A2	MCP6002 pin 5
Unbuffered DAC 2	D13	MCP6002 pin 3
Unbuffered ADC 1	A0	MCP6004 pin 14
Unbuffered ADC 2	A1	MCP6004 pin 8
Unbuffered ADC 3	A4	MCP6004 pin 7
Unbuffered ADC 4	A5	MCP6004 pin 1

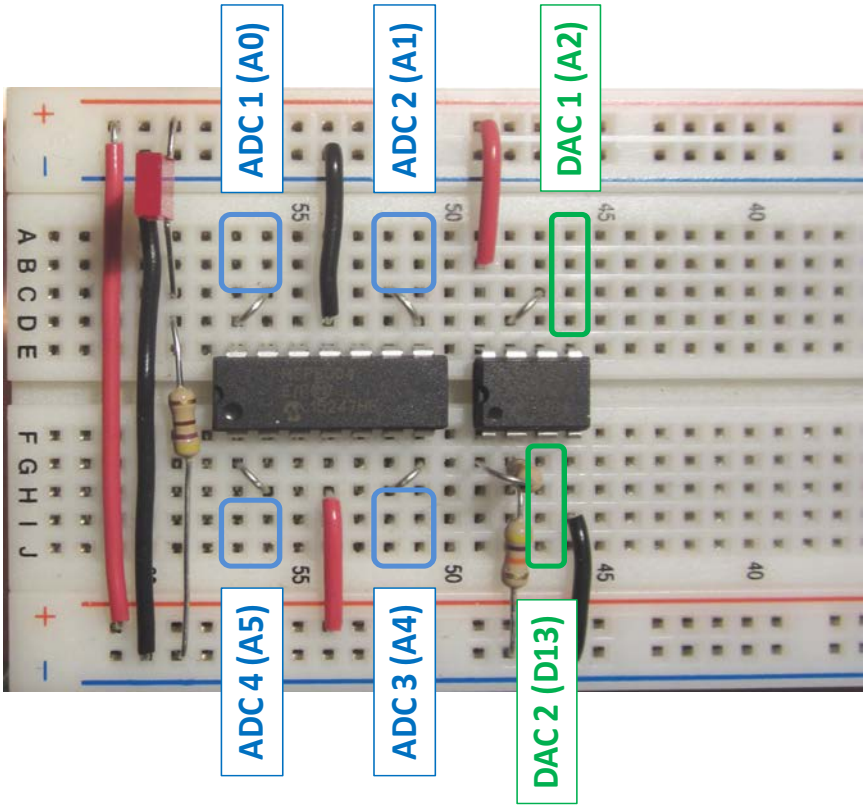


MCP 6004

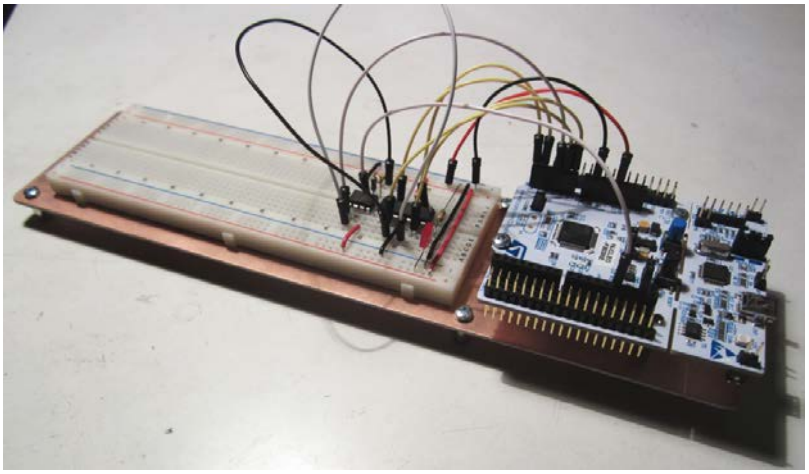


MCP6002

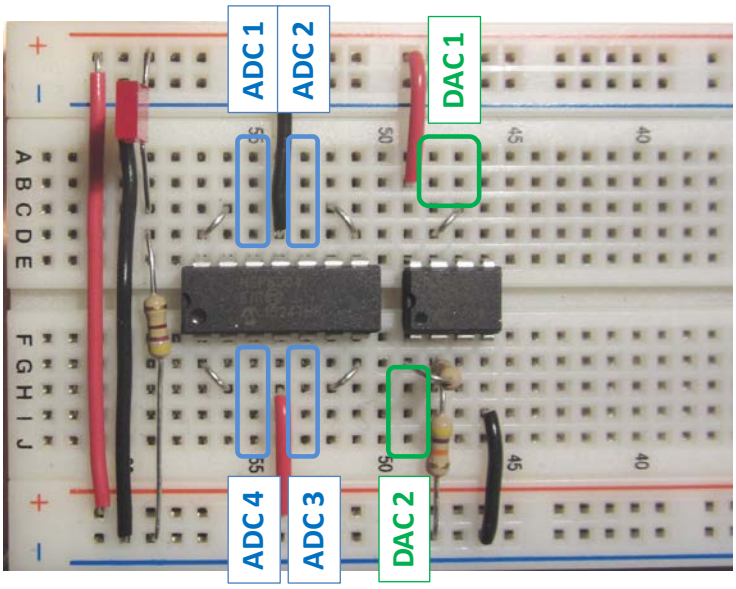
The following figure shows the points to connect on the breadboard.



At this point you have all the connections between the breadboard and the hardware board. The image shows the F303RE board and the breadboard mounted over a copper clad so that they can be carried together. We call this setup the ***Long Board***.



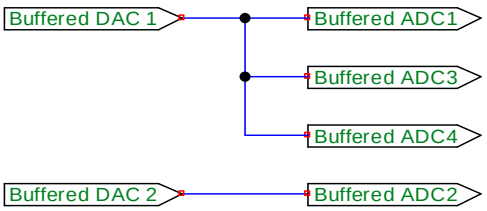
The following figure shows the location of the **buffered connections** we will use to interface our circuits. Note that the unbuffered connections that go to the hardware board have been removed to simplify the image. Don't disconnect them yourself.



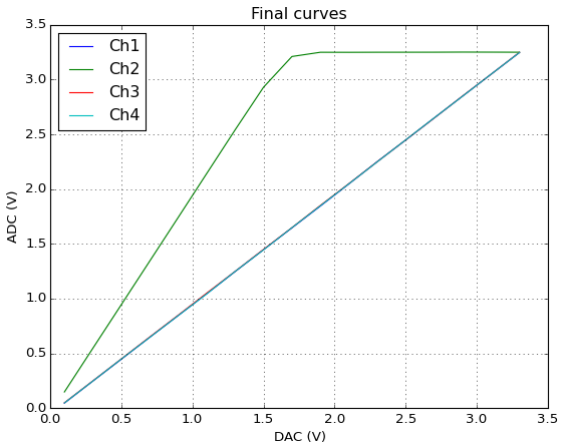
The SLab system can also use digital I/O lines. For now, however, we only need to deal with the previously described analog lines.

First test of the setup

At this point we have all buffered DAC and ADC connections available on our board. We will connect the buffered DAC 1 output to buffered ADC 1, 3 and 4 inputs. We will also connect the buffered DAC 2 output to buffered ADC 2 input.



Now, go to the ***Code*** folder and execute the ***Calibrate4.py*** script. It will request you to hit RETURN, after that, it will perform some measurements and generate a curve like the one below. If the curve does not resemble to the figure, revise the connections.



The above curve should be a set of straight lines, all with the same values on both axes. Curves for channels 1, 2 and 4 are good but not perfect. Curve 2 is very bad. Next document will deal with calibration so that we can rely on the measures we perform using the SLab system.

Curves above are for a fresh uncalibrated system. If you have executed the ***Calibrate4.py*** script in a properly calibrated system all curves should have the same X and Y values.

References

SLab Python References

Those are the reference documents for the SLab Python modules. They describe the commands that can be carried out after importing each module.
They should be available in the **SLab/Doc** folder.

TinyCad

Circuit images on this document have been drawn using the free software TinyCad
https://sourceforge.net/projects/tinycad/

Fritzing

The Fritzing program has been used to draw the breadboard images.
http://fritzing.org/home/

Copyright © Vicente Jiménez (2017)
This work is licensed under a Creative Common Attribution-ShareAlike 4.0 International license. This license is available athttp://creativecommons.org/licenses/by-sa/4.0/