Arduino 10bits network analyser

Contents

[1) Stand-alone use 2](#_Toc454546554)

[2) Workflow 4](#_Toc454546555)

[3) Network analyzer schematics and Arduino docking 5](#_Toc454546556)

[4) Detailed workflow 8](#_Toc454546557)

[5) User settings 9](#_Toc454546558)

**Note:**

In these notes “network analyser” refers to the board holding the ADC and SRAM chips. This board is to be plugged to the Arduino microcontroller. The system showed similar performance on a solderless breadboard (short wires must be used).  
This is a tutorial to build, modify and use this Arduino 10bits network analyzer.

**Main features:**   
10bits resolution over 0 - 5 V range (300uV precision).  
~0.03 Hz temporal resolution (can be changed programmatically).  
3000 Hz spectral width (can be changed programmatically).  
BNC input.  
XXX daq pts 10bits.

**IC components:**  
[Arduino Uno R3](https://www.arduino.cc/en/Main/ArduinoBoardUno) (microcontroller)  
[MCP3002](http://ww1.microchip.com/downloads/en/DeviceDoc/21294E.pdf) (10bits ADC chip)  
[23LC1024](http://ww1.microchip.com/downloads/en/DeviceDoc/25142A.pdf) (SRAM chip)  
  
**Needed softwares:**  
Arduino IDE V1.7.10 (freeware)  
Processing V3.1.1 (freeware)  
Matlab (Signal Processing Toolbox required)

Arduino script (analog sampling, data handling, and transfer to PC)  
Processing script (creates buffer files containing digitalized voltage values)  
Matlab script (buffer FFT computation)

1. Stand-alone use
2. Plug the Arduino to the PC (USB)
3. Plug the network analyzer to the Arduino  
   \*See section 3 for plugging instruction
4. Plug the BNC analog output into the network analyzer BNC analog input

\*Voltage input must be within 0-to-5 V range

1. Compile and upload the Arduino script
2. Run the Processing script
3. Run the Matlab script (Matlab script must be in the folder containing the Processing script or Matlab path must be that folder)

Matlab will then continuously show the last set of data available, its FFT and peak frequency.

When running, Matlab will plot the first 200 sample points and display the log of the full FFT as in figure1.

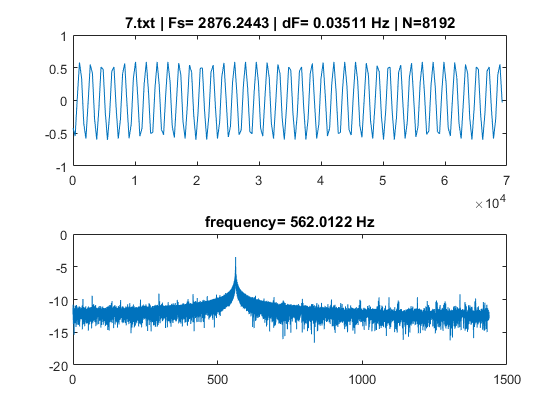


Figure 1 : Run on a 562Hz sinwave, Vpp=1V.   
Fs: Sampling frequency, dF: spectral resolution. N: number of samples in the buffer.  
Subplot1: first 200 points of the signal (amongst N).  
subplot2: log FFT, and peak frequency.

1. Workflow

Main functions:

*[Arduino+Network analyser]* are recording the data from the ADC and sending them to the PC, *[Processing]* is getting the data and putting them in a one column file, *[Matlab]* is reading the data file and doing the computations (FFT etc.).

Simple workflow:

*Loop until user termination:*{

**Arduino side:**

1. *loop until SRAM is filled with desired number of data points:*

{  
-ADC chip acquires voltage value (2bytes)  
-voltage digital value is written to the SRAM chip (2bytes)  
}

1. Once the SRAM is filled, Arduino will read each byte and transfer it to the PC buffer via serial connection (the Processing script is used to communicate with the Arduino)

**PC side:**

1. Once all the SRAM bytes are transferred to the PC buffer, Processing will save the digital voltage values into a text file: #.txt (# goes from 1 to 9, circularly).
2. As soon as a new file #.txt is available, Matlab will read the file, compute the FFT and plot the data

}

# Network analyzer schematics and Arduino docking

Circuit schematics: network analyser + Arduino

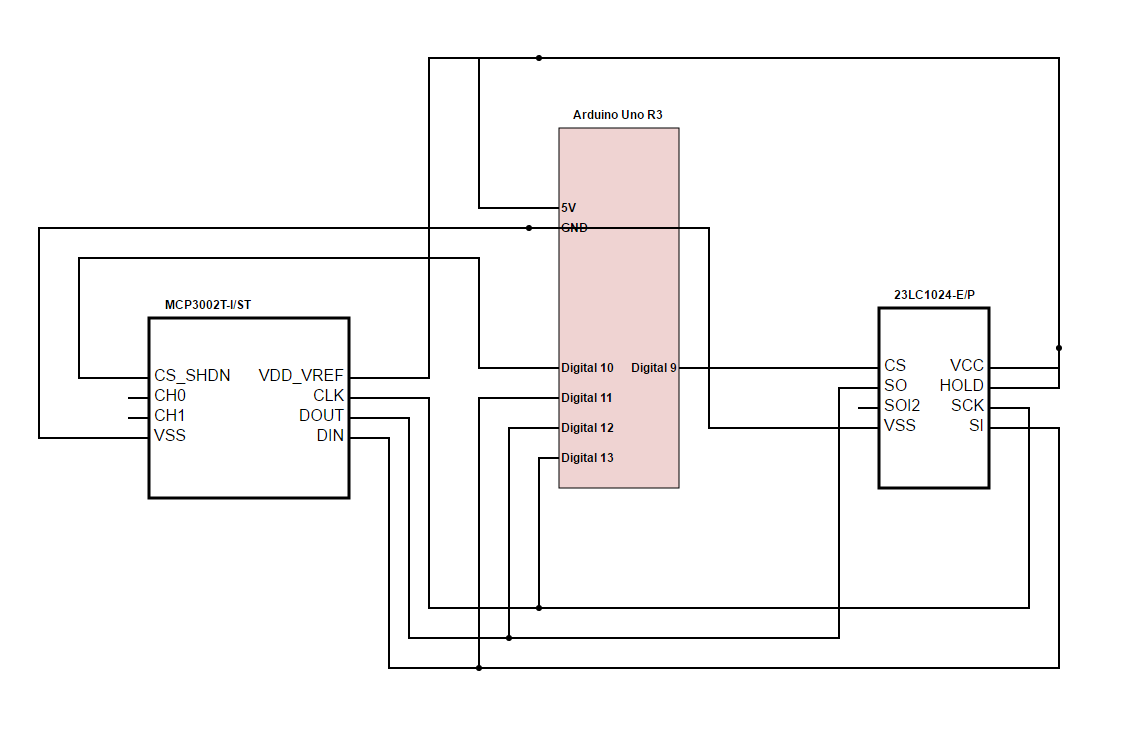


Figure 2 : Network analyser circuit+Arduino

Pin 9 to CS SRAM  
Pin 10 to CS ADC  
Pin 11 to Data in (DIN & SI)  
Pin 12 to Data out (DOUT & SO)  
Pin 13 to Clock (CLK & SCK)  
5V Arduino to Vdd, Vref, HOLD  
GND arduino to Vss

CH0: BNC input (negative BNC input is grounded)

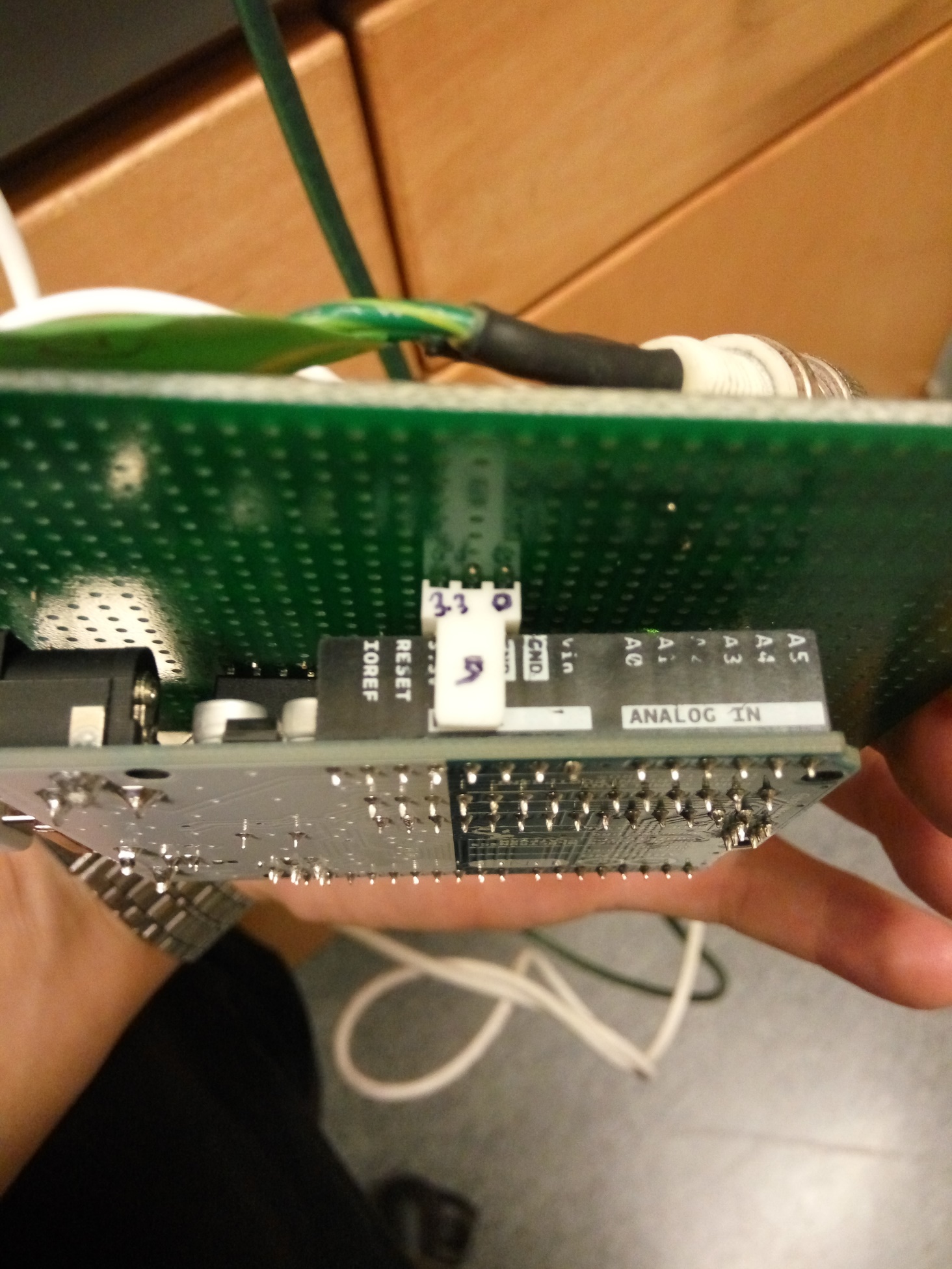
Docking instruction: network analyser (soldered version) + Arduino :

Figure 3 : Center 3pin-Molex set to Arduino’s 5V pin. See figure2 for pin details.

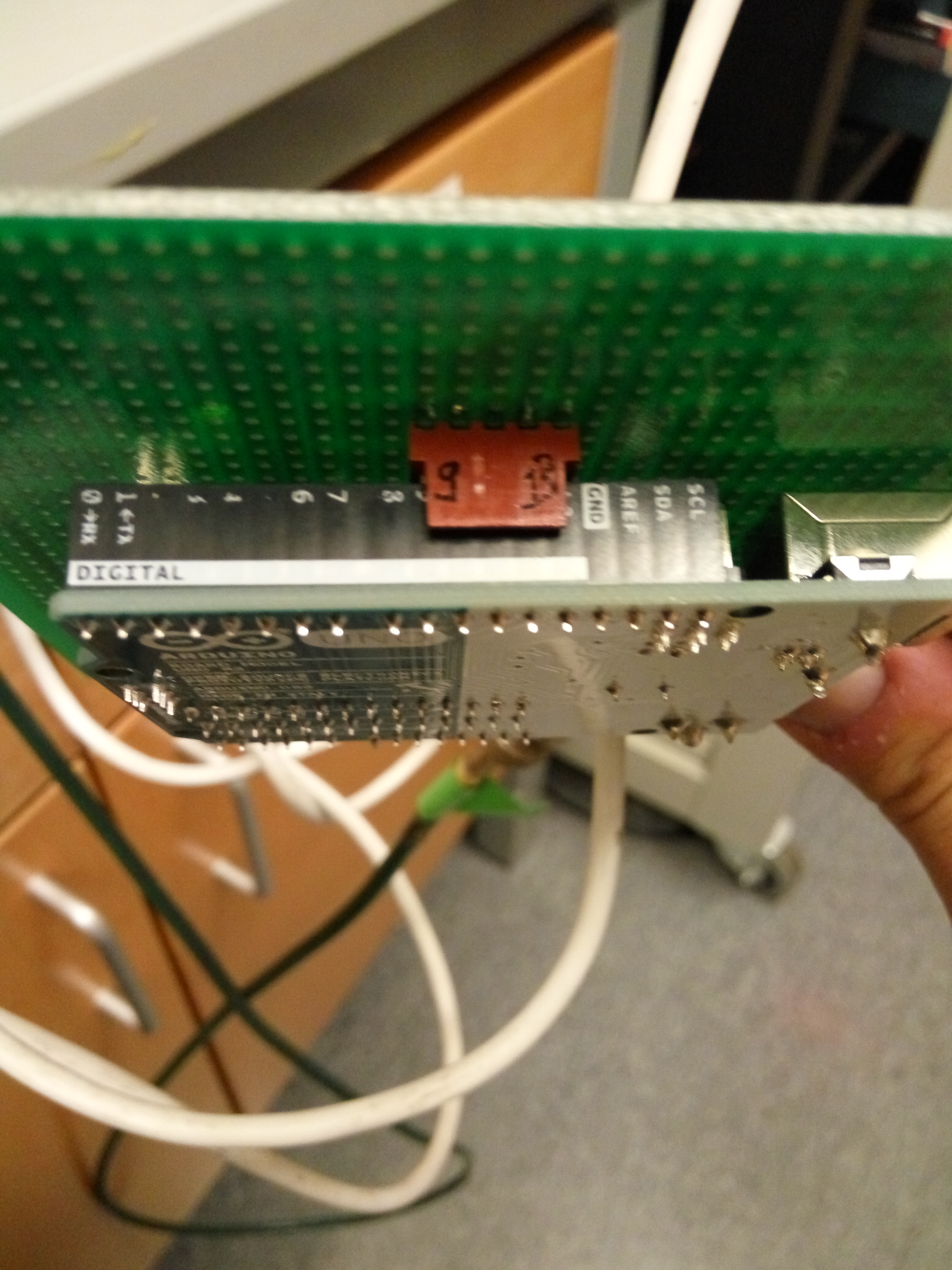


Figure 4 : Right-most 5pin-Molex is plugged to Arduino’s digital pin 13. See figure2 for pin details.

1. Detailed workflow
2. Arduino script  
   Arduino, MCP3002 ADC and 23LC1024 SRAM are communicating to the Arduino via SPI. In this network, Arduino is the SPI master (running the SPI clock), MCP3002, 23LC1024 are slaves.

-Arduino initializes SPI and Serial protocols

*loops until SRAM is filled with desired number of data points:*

{

-Arduino selects MCP3002 (setting voltage HIGH on pin 10)

-Arduino receives 2bytes from MCP3002

-Arduino selects 23LC1024 (setting voltage HIGH on pin 9)

-Arduino writes the 2bytes in 23LC1024

}

*once the SRAM is filled with the desired number of data points:*

-Arduino records number of microseconds (timestamp) elapsed during the acquisition.

-Arduino selects 23LC1024

-Arduino transfers every bytes from 23LC1024 to the PC buffer

-Arduino transfers the timestamp to the PC buffer

-Arduino sends a “return” (\newline) to the PC buffer

1. Processing script  
   *once Processing receives the “return” command from Arduino:*-Processing writes every bytes from the PC buffer in a text file “#.txt”, where # runs from 1 to 9. Digital values are written in one column, the last value is the timestamps
2. Matlab script  
   -Once a new text file is available, Matlab loads the text file  
   -Matlab computes the FFT. Sampling rate (used to compute actual frequencies) is computed using the total acquisition time (timestamp/Number\_Of\_Sample).  
   -Matlab plots the signal and the log(FFT), showing the peak frequency.

# User settings

1. Buffer size (number of sample):  
   The number of sample between for FFT can be adjusted in the Arduino code:

“int BufferSize” (default=8192, max=XXX)

Each sample is 2 bytes long, total SRAM memory is 1Mbits (125’000Bytes).

1. Frequency resolution and spectral width:  
   The frequency resolution can be adjusted by changing the time delay between samples (this has the effect of adjusting the total acquisition time) in the Arduino code.

“int ControlDelay” (default=300) (in microseconds)

1. Matlab options:

-config.Decay : numerical exponential decay of the signal.  
(=1 : unabled, =0 : disabled)

-config.ZeroPadingSize: zero filling option. Will multiply the total size of the signal, adding zeros at the end. Useful to increase the FFT frequency resolution.

(Choose an integer value, greater than 1; 1=no filling)

-config.FreqCorrection, config.TimeCorrection: Calibrate the frequency of the signal, should be done once. May improve the precision on the peak frequency.

Note that the serial transfer to PC is a bottleneck. With current settings, recording 8192 point at a sampling frequency of 3000Hz and sending them back to the PC takes approximately 4seconds.