# A C++ API for the Implementation of Software-Defined-Radio NMR Spectrometers

João Diogo Campos Franco
joao.c.franco@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

November 2019

**Abstract**

Recently, home-build NMR systems have become easier to develop due to the emergence of the SDR technology. This article describes the implementation of SDNMR - Software Defined Nuclear Magnetic Resonance - a C++ API envisioned for the development of SDR-based NMR applications and flexible low-cost bench-top NMR systems. This free and open-source API contains robust algorithms designed to operate with any arbitrary SDR interface, as well as any micro controller board for the precise operation of gradients, shims, varactors and switches, if necessary. While currently the software only offers support for the LimeSDR-USB board and Texas Instruments microcontroller boards, it is shown how to add support for new SDR boards and micro controller boards. As a proof-of-concept, an application named LimeNMR was developed with this API, designed solely for the cross-platform operation of a LimeSDR-USB based spectrometer with any additional control given by a Texas Instruments MCU Board. The application is capable of calibrating the spectrometer to a desired condition of operation as well as performing basic routines of probe adjustment, center frequency searching, pulse length determination, $T_1$ and $T_2$ measurement and shims adjustment. It also benefits from a flexible pulse programmer capable of controlling different pulse shapes and timings, gradients and external switches, with a 1 µs time resolution.
**Keywords:** NMR, SDR, Home-built digital spectrometer, Software development

## 1. Introduction

In recent years, several commercial NMR spectrometers have become available but they can still be expensive and application oriented. Furthermore, the hacking of such devices is not always recommended since they are usually not open-source.

The development of home build NMR systems, imposes several challenges related with the design of the transmitting and receiving electronics. These implementations are designed attempting to balance the complexity, the cost and the flexibility of the project. Usually, the broader the range of applications for the NMR system developed, the more complex and expensive the project becomes, hence, the more difficult it is for it to be reproduced.

Some application-oriented solutions describe only the development of modules required by NMR spectrometers, or the design of digital systems based on field-programmable gate arrays (FPGAs). The Open Core NMR project[1] provides an open-source FPGA modular system, however, users still have to recreate the expensive system themselves (US$\$5000$).

With the development of telecommunication systems, Software Define Radios (SDRs) have emerged. SDR describes a family of programmable devices which include DACs, ADCs, amplifiers, mixers and a PC interface. As such, SDR allows radio to become a dynamic software element, while the hardware only has to implement the RF front-end[2].

A NMR spectrometer can be thought as a radio system. Quoting R.R. Ernst, "NMR spectroscopy (...) can be understood based on a sound theory"[3]. Therefore, with the transpose of SDR into the design of NMR systems, one does not need to struggle with the difficulties associated with the electronics design and is not restricted to a limited use of applications only. SDR boards are cheap and allows one to obtain an "off-the-shelf" spectrometer if one has access to some SDR-compatible NMR software.

The proof-of-concept of the direct implementation of an off-the-shelf SDR device into a simple NMR system has been done[4] and, recently, more NMR systems based on SDR technology have been purposed with available commercial boards[5, 6].

There are two main problems with the purposed

SDR based systems so far. First, they have been developed to be device dependent, dealing with the specific details of the SDR board used[6]. Second, the software employed for the development of the digital NMR system was not NMR specific[5], hence, do not provide means for software maintenance and the introduction of new spectrometer features. Furthermore, most NMR open-source software focus on the digital signal processing (DSP) of already acquired NMR data and also on the simulation of NMR spectra, neglecting NMR data acquisition and pulse programming. A flexible, maintainable and complete NMR software suite in LabView was purposed[7], but for direct-acquisition DAQ boards, which operate at fixed low frequencies.

To solve these problems, it was created SDNMR - Software Defined Nuclear Magnetic Resonance - a C++ free open source API for interfacing with arbitrary SDR devices and develop SDR-based NMR applications without the need to know specific details of the SDR board used. It offers algorithms for the calibration, data acquisition, data processing and pulse programming for any generic SDR board. It also offers support for the operation of a peripheral MCU board that can operate gradients, shims, varactors or switches.

SDNMR was designed to accommodate an ecosystem of SDR-based NMR applications so the first application with this framework was also developed, named LimeNMR, designed for the solely operation of a LimeSDR-USB[8] and a LaunchXL-F28337s[9] NMR spectrometer.

## 2. Fundamental Background

When an external static magnetic field $\overrightarrow{B_0}$ is applied to a NMR-sensitive sample, the sample exhibits a bulk magnetization $\overrightarrow{M}$ in the same direction of $\overrightarrow{B_0}$.

As described by the Bloch equations[10], when the sample is subjected to a short excitation pulse of a radiofrequency (RF) magnetic field $\overrightarrow{B_1}$, perpendicular to $\overrightarrow{B_0}$, $\overrightarrow{M}$ is rotated from its equilibrium position and will exhibit a precessing motion with a Larmor frequency $f_0$, given by:

$$f_0 = \frac{\gamma}{2\pi} B_0 \tag{1}$$

with $\gamma$ being the characteristic gyro-magnetic ratio of the sample's nucleus.

The angle by which $\overrightarrow{M}$ is initially rotated is named the tip-angle $\beta$, equated by:

$$\beta = \gamma \tau B_1 \tag{2}$$

with $\tau$ being the duration of the RF pulse applied to the sample.

This Larmor precession motion of the sample's bulk magnetization is detected during an NMR experiment resulting in the NMR signal denominated Free Induction Decay (FID).

For a single-pulse 1D NMR experiment, the maximum NMR signal is obtained for a tip-angle $\beta$ of 90º and the minimum NMR signal is obtained for a tip-angle $\beta$ of 180º.

The experiment's signal-to-noise ratio (SNR) scales with the magnetic field's strength $B_0$ and with the sample's target nuclei density. The characteristics of the detection coil also influence the noise presence of the acquired signal.

## 3. Implementation

### 3.1. The SDNMR Project

The SDNMR is a C++ API designed to interface with any arbitrary SDR device and reduce the development time of new SDR based NMR systems. It is a free and open-source software distributed under the terms of the GNU General Public License (GPL). The project intends to both collect innovative software required in the field of NMR digital systems and to provide diverse user-friendly, efficient and device-independent NMR software compatible with commercially available hardware.

The API consists in a collection of software capable of initiating SDR devices, acquiring and processing NMR data, and with flexible pulse programmer. It also provides support for the addition of a peripheral microcontroller (MCU) board for an extra control of shims, gradients, varactors and switches. It does not aim to provide a single monolithic solution only valid for specific hardware architectures, but rather to be an NMR spectroscopy tool.

The block diagram that represents the structure of the developed API is illustrated in fig.1.

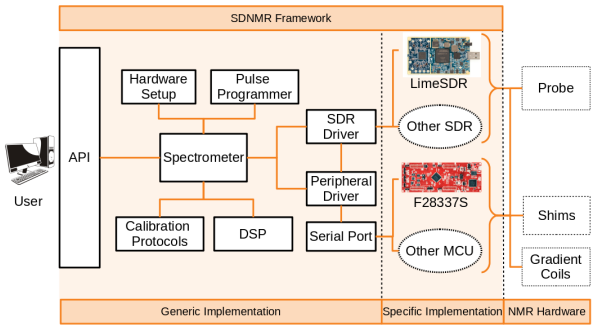


**Figure 1:** Block diagram illustrating the architecture of the SDNMR API. The right side represents the external hardware common in NMR experiments and their connection to the framework developed.

There are five main features provided by the API, namely: arbitrary device support, specification of the experiment's hardware setup, digital signal processing techniques, RF and NMR calibration protocols and a pulse programmer.

### 3.2. Device Support

This feature provides a generic connection interface between the boards and the remaining software, so the API is not compatible to a specific SDR board or microcontroller (MCU) board. Instead, it is generalized and provides support for any MCU board and SDR device capable of both transmitting and receiving. The interfaces are provided by the SDR Driver and the Peripheral Driver, allowing the connection between the SDR boards and the MCUs boards, respectively, to the software suite (fig.1).

To add support for another commercial or custom made SDR board, the SDR Driver class must be informed on how to access the new board's registers and execute transmitting and receiving actions with that access.

Similarly, the Peripheral Driver allows the execution of generic serial communication protocols between any MCU board's custom interface implemented and the remaining SDNMR framework. The LAUNCHXL-F28337s board was programmed to have two 12 bits DAC pulse width modulated (PWMs) varactor controllers, ten 12 bits DAC PWMs shim controllers, three 12 bits DAC gradient channels and two probe switch controllers with several switch logic modes. Furthermore, an additional pin is able to perform tasks upon an external interrupt (edge transition low to high).

### 3.3. Hardware Setup

For the development of flexible NMR systems, the API also offers methods to change the external hardware setup related to the probe, gradient coils and shims, and how are they connected to the SDR board (and the MCU board).

### 3.4. Digital Signal Processing

The API provides methods for the storage, processing and exportation of NMR data in case there is a need to further process it with a third party software.

The DSP features are based on high performance mathematical libraries, such as the FFTW3 library[11] and the GNU scientific library[12]. It contains all the common processing techniques employed in 1D NMR experiments, including phase correction, peak finding, baseline correction, FID apodization and spectrum smoothing.

### 3.5. Calibration Protocols

This feature consists in the collection of routine calibration protocols developed with the ultimate goal of maximizing the SNR of the acquired signal and measure important spectroscopic parameters of the sample.

#### 3.5.1. DC offset and IQ Imbalance Corrections

This routine requires the transmission of a continuous wave (CW) test signal between the calibrating TX and RX SDR's ports. Upon digital processing, this signal is decomposed in its spectral frequencies which will reveal the wanted tone the frequency of the CW signal) and all the unwanted tones resulting from the DC offset and IQ imbalance effects of both the TX and RX paths. A binary search algorithm is applied to the SDR's correction registers until all the unwanted tones amplitudes are minimized and a calibration cache is generated. Fig. 2 illustrates this algorithm.

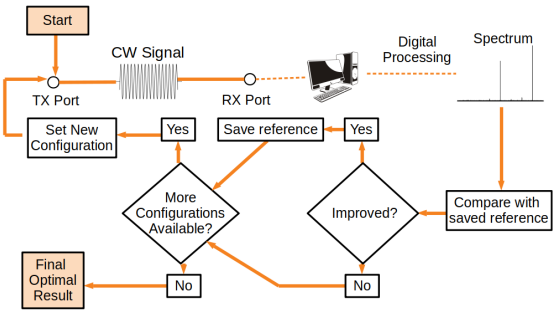


**Figure 2:** Illustration of the DC offset and IQ imbalance correction algorithms. Details are clarified in text.

#### 3.5.2. RX and TX Characterization

The RX calibration requires the connection of the RX port to an RF frequency synthesizer, followed by the acquisition of CW signals of known power and frequency for each RX gain, including a measurement of the thermal noise. A calibration cache is then created for each gain, associating the external power to the amplitude of the CW signal's power spectrum tone. The results are processed to compute the RX sensitivity and linearity for the intended center frequency of operation and to convert the RX port into a measuring receiver so that the TX characterization can take place.

The TX calibration consists in the transmitting of CW test signals from a TX port to a measuring receiver. The power of these signals is then computed for different TX configurations.

#### 3.5.3. Probe Adjustment

This routine requires an RF directional coupler, both the SDR and the MCU board, and a probe with voltage controlled varactors that are connected to the MCU board. It is illustrated in fig. 3.

TX is connected to one end of the RF directional coupler's mainline. The other end of the mainline is connected to the probe's channel intended to be calibrated. The coupled port is connected to RX. RF signals at the desired frequency of tuning are then sent to the probe and the reflected power is acquired in the measuring receiver. The

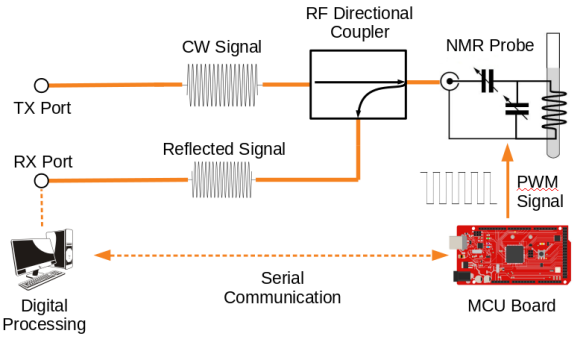


**Figure 3:** Illustration of the automatic probe adjustment routine's setup. Details are clarified in text.

MCU board controls and adjusts the voltage of the probe's varactors until the reflected power is minimal, through a binary search algorithm applied to the probe's varactors controllers.

#### 3.5.4. Center Frequency Searching

The center frequency searching algorithm sweeps a frequency range, determined by the user, while applying a C++ version of the peak finding algorithm described in[13], applied over the magnitude spectrum of the acquired signal instead of the lorentzian spectrum. The peak amplitude threshold and the peak broadness threshold can also be adjusted.

#### 3.5.5. Pulse Length Determination

This routine consists on the transmission of a train of hard pulses with known power and different lengths. The resulting FIDs acquired are processed and a nutation spectrum is generated. The first null point of the nutation spectrum is automatically identified and set as the 180º pulse length.

The nutation spectrum can be generated with either the lorentzian or the magnitude NMR spectra. The integral of the pulses is also computed to avoid the calibration of the pulse length for every pulse shape.

#### 3.5.6. T1, T2 and T2* Measurement Routine

The longitudinal relaxation time $T_1$ is obtained through the execution of an inversion recovery (IR) NMR sequence[14]. The resultant signal intensity $I(\tau)$ as a function of $\tau$, the wait time between the 180º and the 90º pulses of an IR sequence, is fitted to the curve described by:

$$I(\tau) = I_0 \left(1 - 2\exp\left[\frac{-\tau}{T_1}\right]\right) \tag{3}$$

with $I_0$ being the signal's maximum intensity.

The non-linear fitting function to the signal's intensity, is preferable to the linear fit of the log of the intensity after a long evolution time minus the intensity against time. The linear fit in this manner is less accurate than the direct non-linear fit since too much weight is attributed to the noisier points of low intensity.

Similarly, the transversal relaxation time $T_2$ is computed from the non-linear fitting of the data resultant from a CPMG sequence[14]. The signal's intensity $I(\tau)$, as a function of $\tau$, the wait time between the 90º and the 180º pulses of a CPMG sequence are fitted to:

$$I(\tau) = I_0 \exp\left[\frac{-2\tau}{T_2}\right] \tag{4}$$

with $I_0$ being the signal's maximum intensity.

Finally, $T_2^*$ is determined from a linewidth analysis of the lorentzian NMR peak, through equation:

$$\Delta\nu = \frac{1}{\pi T_2^*} \tag{5}$$

with peak's full width half maximum ($\Delta\nu$), obtained by the peak fitting to a lorentzian function.

### 3.6. Pulse Programmer

The pulse programmer (PP) of the SDNMR API is capable of generating arbitrary NMR sequences.

Each NMR sequence can be divided into several sub-sequences and each one of them is characterized by a repetition time and a number of scans. Every sub-sequence is composed by pulse sequences and gradient sequences, which can either be configured or bypassed, resulting in a flexible pulse programmer. A block diagram illustrating the architecture of each NMR sequence is present in fig. 4.

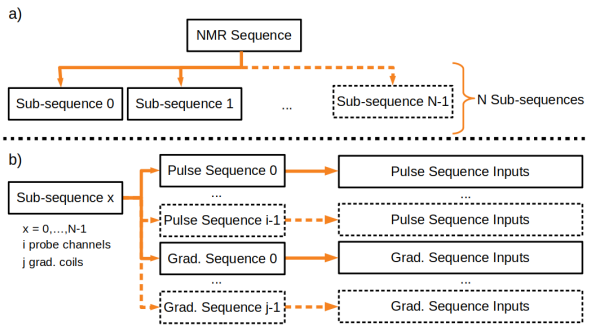


**Figure 4:** a) Each NMR sequence produced with the pulse programmer consists of $N$ sub-sequences, each one characterised by a repetition time, and a number of scans. b) Each sub-sequence is composed by a maximum number of $i$ pulse sequences, dictated by the number of probe channels available and a maximum number of $j$ gradient sequences, dictated by the number of gradient coils available.

Each pulse sequence can be configured as an arbitrary combination of pulse sequence inputs: template excitation pulses or custom pulses defined by the user, wait times and acquisitions. The pulses can be generated with a digital offset frequency from the channel center frequency. Each

gradient sequence can be configured as an arbitrary combination of gradient sequence inputs: gradient values and wait times.

The outcome of the PP is the automatic generation of IQ data buffers ready to be transmitted by an SDR device and both configuration data and timestamps data to be fed to an MCU device.

The actual methods for transmitting and receiving are dependent on the particular API of the devices employed and is carried away by the SDR Driver class and the Peripheral Driver class. Hence, the data generated automatically by the pulse programmer must be treated to match these boards first.

Relative to the MCU device, the configuration data sets its state to certain conditions before the start of an experiment; the timestamps data is used to control the interrupts and actions of the timers during the experiment. For the LAUNCHXL-F28837s, a system with three simultaneously operating timers was implemented: one timer controls the overall experiment timings, the second one controls the switches timings and the third and last one controls the gradients timings.

For the SDR board and the MCU board to operate synchronously, the former emits a trigger signal when the transmitting starts at the beginning of every sub-sequence. This trigger enables the execution of the pre-programmed routine of switches and gradients configuration of the MCU board.

### 3.7. LimeNMR

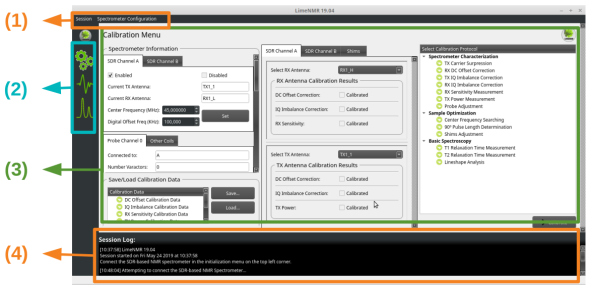


**Figure 5:** Snapshot of the LimeNMR initial menu, the calibration page, upon its execution and spectrometer connection. The legend is described in text.

Currently in its 19.04 version, the LimeNMR interface is presented in fig.5. On the top menubar (1) there are options to connect the spectrometer and to change its settings of operation; on the left toolbar (2), there are three main menus: a calibration menu, for the execution of the calibration routines to the conditions of operation of the NMR experiment (the initial menu); a pulse programmer menu, for the graphical generation of custom NMR sequences; and a spectra view menu, for the visualization and processing of the data acquired with the custom NMR sequences. The options referent to each one of those menus are then presented on the right page (3) while continuous information about the processes occurring during the spectrometer's operation is displayed on the session's log (4).

### 3.8. Implemented NMR Spectrometers

The SDNMR API aims to be a tool for the development of SDR-based NMR, hence, two different systems were implemented and tested: a 45 MHz spectrometer and a 300 MHz spectrometer.

#### 3.8.1. 45 MHz spectrometer

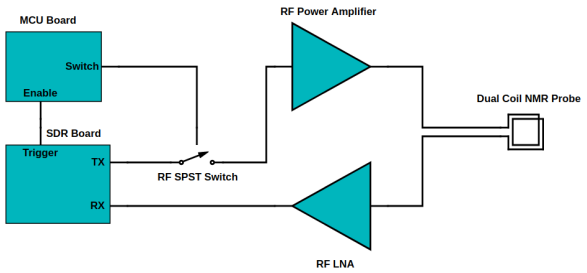


**Figure 6:** Schematic of the 45 MHz NMR spectrometer circuit implemented.

The 45 MHz spectrometer implements a 32 W RF transmitter power amplifier, designed to operate in the range of 1MHz-700MHz, and a 30 dB low noise amplifier (LNA) to enhance the NMR signals.

A pin diode RF single pole single throw (SPST) switch acts as the TX gate and it is controlled by one of the MCU's board switch controllers.

A dual coil probe is used under the Bruker's ICON, a bench-top 1T permanent magnet.

#### 3.8.2. 300 MHz spectrometer

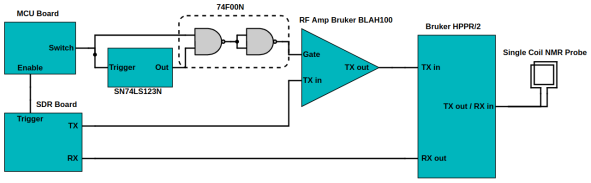


**Figure 7:** Schematic of the 300 MHz NMR spectrometer circuit implemented.

The 300 MHz spectrometer utilizes the 100 W RF transmitter power amplifier (PA) BLAH100 from Bruker, which is capable to operate in the range of 200MHz-600MHz, whose gate is controlled by the MCU board. A monostable protection circuit, to limit the maximum pulse duration to 200 µs, was employed for safety reasons.

The Bruker HPPR/2 module is an RX preamplifier with a TX/RX switch integrated.

## 4. Results

### 4.1. LimeSDR-USB Performance

It is critical to know some performance features of the SDR board selected to operate with the SD-

NMR API, in order to optimize the experiment's results. The measurements obtained relative to the TX ports maximum output power (fig. 8) and the RX ports reflection coefficient (fig. 9), as a function of the channel center frequency, are reported. Both LimeSDR-USB channels shown similar curves for their TX and RX ports, therefore, only the curves referent to the LimeSDR-USB channel A are presented.

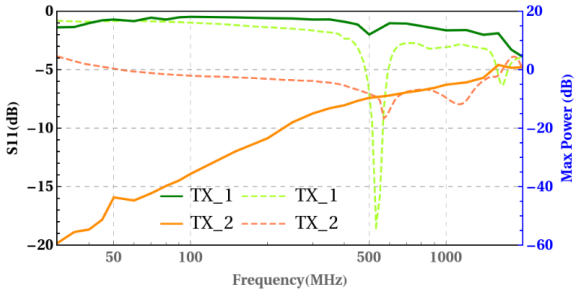


**Figure 8:** Performance of the TX ports of channel A of the LimeSDR-USB. The green data corresponds to the TX1_1 port, while the orange data corresponds to the TX1_2 port. The thick lines indicate the maximum power of each port for a given TX center frequency, while the dashed lines show the antennas S11 curve, obtained with the E5071C ENA Vector Network Analyzer. Generated with *Mathematica*.

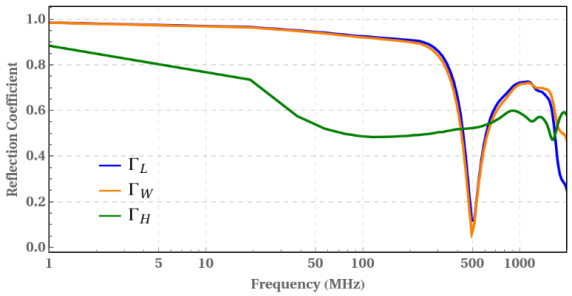


**Figure 9:** Reflection coefficient of the RX ports of channel A of the LimeSDR-USB. Generated with *Mathematica* using S11 data acquired with the E5071C ENA Vector Network Analyzer.

From 30 MHz to 2 GHz, the TX_1 port of each channel always provide a maximum output power higher than TX_2. At 45 MHz, the TX_1 port can output a maximum power around 16 dBm, while the TX_2 port generates a maximum output power of -45 dBm. At 300 MHz, the TX_1 port can output a maximum power around 17.5 dBm. Moreover, the TX front-end is not matched for a 50 $\Omega$ output impedance, except in the frequency region around 500 MHz.

The results of the reflection coefficient of the RX ports, should be considered as a starting point for the evaluation of the RX selection for the intended NMR spectrometer, since they are suitable for different frequency ranges. However, they still do not infer anything about the RX sensitivity and the linearity.

### 4.2. Pulse Programmer

Figure 10 illustrates four different pulse shapes generated with the SDNMR API and transmitted with the LimeSDR-USB: a) an hard pulse with a 8 ms duration and a digital offset frequency of 100 KHz; b) a sinc pulse with 10 lobes, a bandwidth of 5000 Hz and a digital offset frequency of 10 KHz; c) a gaussian pulse with 5 ms, 20% truncated and a digital offset frequency of 20 KHz; d) a custom pulse, whose data has been generated with a third party software (*Mathematica*), corresponding to a decaying exponential with 80 ms and a digital offset frequency of 100 KHz. All the generated pulses are produced with a 1 µs time resolution and in accordance with the mathematical models developed.

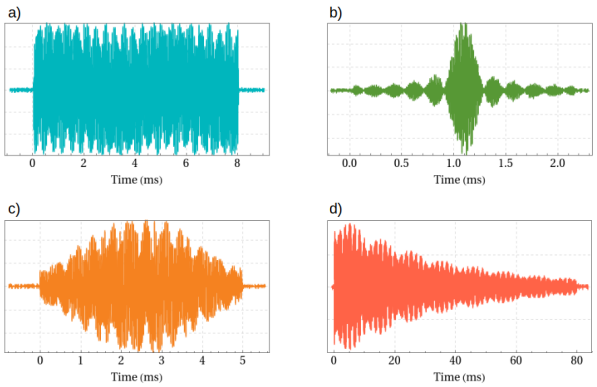


**Figure 10:** Representation of four different pulse shapes generated with the SDNMR API and transmitted with the LimeSDR-USB. Details are clarified in text. The data was retrieved from an oscilloscope.

In particular for the LimeSDR-USB, without digging into the FPGA firmware, the transmitting sequence does not start at the same time that the trigger signal is emitted. Furthermore, this offset time $\delta t_1$ is not constant and acts as a stochastic variable with small variations of up to $\pm$ 0.5 µs.

This fact has two major negative effects in the performance of the pulse programmer using a LimeSDR-USB: first, it prevents the time resolution to be in the order of the sub-microseconds, even after the introduction of a control offset time variable, $\Delta t$, that dictates when does the MCU board starts to execute its routine after the trigger signal has been received; second, the TX pulses may be truncated for a stochastic time $\delta t_2$ of a maximum of 0.5 µs, which can be significant if they are high-power short-duration pulses, as illustrated by fig. 11.

For higher performances it is necessary an FPGA firmware development similar to the one presented in [6]. However, this phase uncertainty only occurs between different sub-sequences of the same NMR sequence. Hence, sum-to-memory coherence is still obtained throughout different averages of the same sub-sequence.

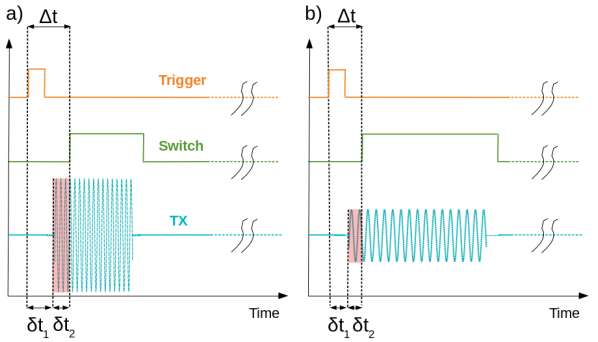


**Figure 11:** Scheme of the pulse truncation effect of the LimeSDR-USB illustrating the $\delta t_1$ and the $\delta t_2$ effects, for an high power pulse (a) and a low power pulse (b). The trigger signal is represented in orange, the gate signal is represented in green and the transmission path is represented in blue for both cases. The area of pulse truncation, represented with a red background, is proportional to the pulse power.

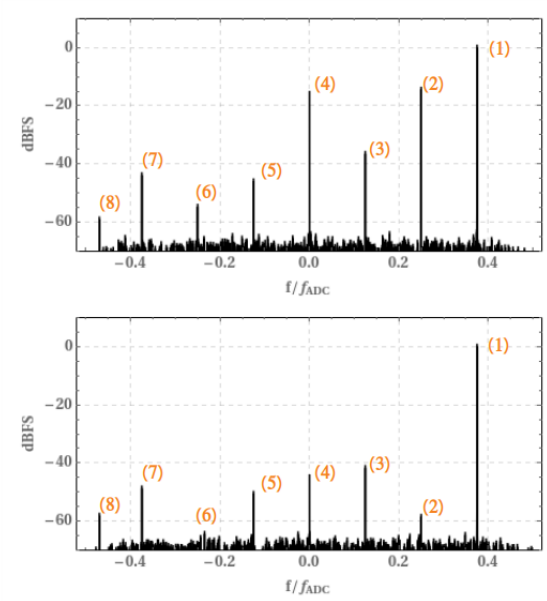


**Figure 12:** Illustration of the correction algorithm for the LimeSDR-USB, at 45MHz. On the top, the spectrum is shown without corrections and, on the bottom, the spectrum is corrected. The peak numeration is also described in the text. Generated with *Mathematica*.

The pulse programmer is capable of automatically synchronizing the switches controllers and both the pulse and the gradient sequences to a $\pm$ 0.5 µs resolution. For optimal synchronization, the sampling rate must be an integer multiple of 1 MHz.

In the case of the LimeSDR-USB, the model developed allows a minimum repetition time of 155$\pm$1 ms. Between repetition times, a sequence buffer containing all the pulse sequence inputs with up to, roughly, $\sim 10^6$ data points can be generated, which corresponds to a 1-2 second sequence using a sample rate of 1 MHz.

Regarding the LAUNCHXL-F28337s developed model, a maximum of 32 state changes can be obtained for each gradient channel and a maximum of 16 state changes can be obtained for each switch controller, individually, per sub-sequence. This MCU board was programmed to operate with a 1 µs time resolution as well.

### 4.3. DC Offset and IQ Imbalance Corrections

The results of fig. 12 were obtained for the 45 MHz spectrometer, since the general spectrometer performance is much more critical at low-field to guarantee the success of the NMR experiment. From right to left: peak (1) is the full scale CW signal, i.e., the wanted tone; peak (2) is the TX carrier, resulting from the presence of DC offset in the TX path; peak (3) is the "ghost" image of peak (1) which results from the IQ imbalance of the TX path; peak (4) is the RX DC offset; peaks (5), (6) and (7) are the mirrored images of peaks (3), (2) and (1), respectively, which appear due to the IQ imbalance of the RX path; finally, peak (8) appears as a sampling frequency artifact.

By comparison between the uncorrected spectra (top) and the corrected one (bottom), the algorithm developed is capable of attenuating all the unwanted tones to below -40 dBFS with respect to the wanted tone. Peak (8) remains unchanged as the algorithm does not correct it, however it is still below -55 dBFS and can be removed with further post-processing.

### 4.4. RX and TX Characterization

The RX characterization results for the 45 MHz spectrometer are shown on figure 13. Both plots are the same, but plot a) measures the SNR of the acquired signals with respect to the thermal noise, while plot b) identifies the optimal zone of operation of the RX port. Each curve of the plots is an isopower curve, representing the same signal measured with different RX gains. The power correspondent to some of the lines is shown on the right side of both slots. The power of the minimum sensitive signal, i.e., the minimum power that is possible to be acquired with an SNR of 3 dB relative to the thermal noise, is also identified, as -130 dBm for the RX_L port of channel A, at 45 MHz. The calibration routine also measures the minimum discernible signal, for the conditions of acquisition, as a function of the RX gain.

The SNR plot, on the right side of fig. 13, shows that the acquired signals SNR can achieve a maximum of close to 111.88 dB, the maximum of a power spectrum obtained with ADCs with an ENOB of 9 bits under quadrature detection.

Saturation is visible for high external powers measured with high RX gains. The thermal noise region (bottom) shows a constant "plateau" for RX gains up to 42 dB, approximately, and it grows linearly after it. This "plateau" occurs due to

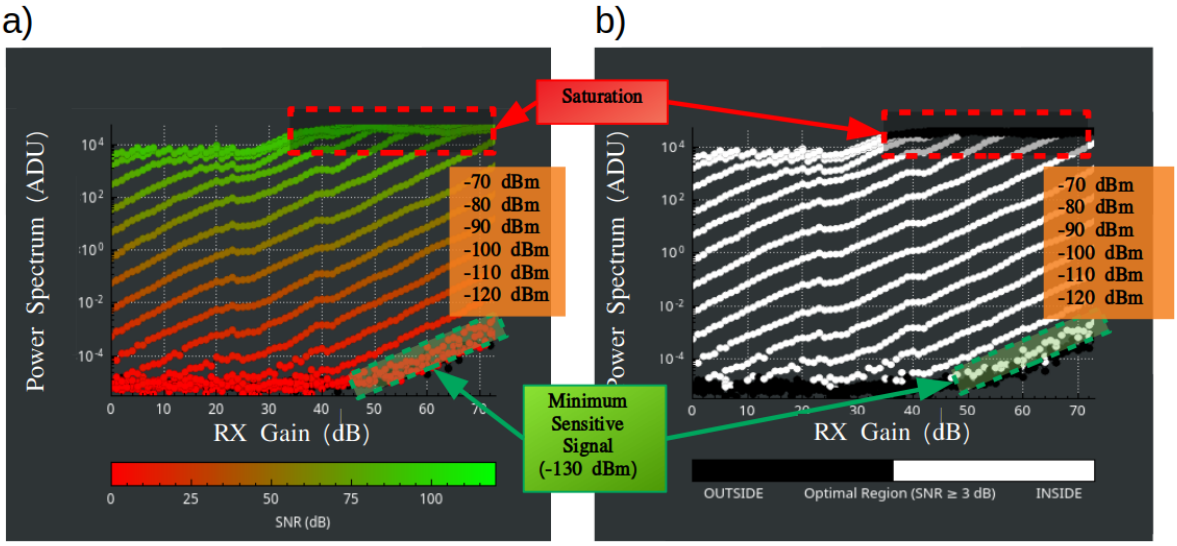


**Figure 13:** RX port characterization results for the LimeSDR-USB RX_L port of channel A, at 45 MHz. a) An SNR plot of several external output powers measured with different RX gains. b) A region plot identifying the suitable area of operation of the RX antenna at the given center frequency. Details are clarified in text. Obtained with LimeNMR using a HM8135 programmable RF synthesizer.

low quantization resolution below 42 dB. After 42 dB, the thermal noise is effectively measured with higher degrees of precision.

The right side of figure 13 shows the linear zone of operation of the RX_L port of channel A, at 45 MHz, defined between the minimum sensitive signal for such operational conditions and the 1 dB compression point.

The TX_1 port of channel A characterization is illustrated in fig. 14, at 45 MHz, for a gain of 50 dB and of 73 dB. In a) the curve is parabolic, since the output power is proportional to the square of the transmitted amplitude, and in b) one can observe a saturation in the TX generation.

This saturation is significant and occurs at high gains which are, usually, the most important for the NMR experiments. The API uses this calibration for the generation of pulses with more precise output power.

The maximum output power measured for the highest TX port gain is, roughly, 0.35 W, or 25.44 dBm, i.e., almost 9.5 dB higher than the maximum output power of 16 dBm presented in fig.8. This occurs because RX_L, the measuring RX port, was calibrated to a 50 Ω impedance RF synthesizer, but the TX_1 port's impedance is not of 50 Ω, as can be seen from fig.8 also.

In order to obtain results relative to a 50 Ω impedance system, it is necessary to employ impedance conversion circuits on the output of the SDR's RF ports.

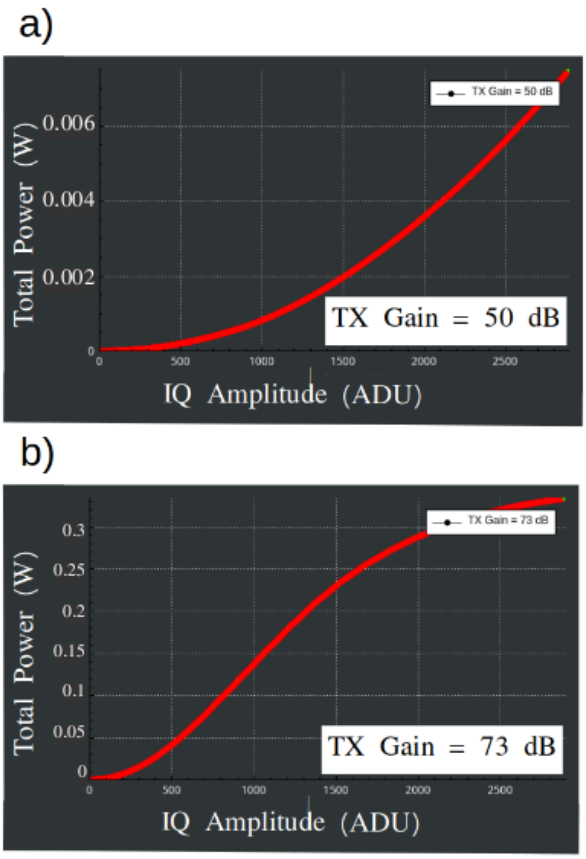


**Figure 14:** TX port characterization results for the LimeSDR-USB TX_1 port of channel A, at 45 MHz. a) Power plot for a TX gain of 50 dB. b) Power plot for a TX gain of 73 dB, the highest possible. Details are clarified in text. Obtained with LimeNMR.

### 4.5. Center Frequency Searching

This routine is able to identify NMR peaks that are present within a frequency range pre-determined,

with an Hz precision, and distinguish them from an unwanted spectrum artifact, provided that the two do not overlap.

Figure 15 shows a snapshot of the LimeNMR Center Frequency Searching Menu identifying a signal from an isopropanol sample and the spectrum of the acquired signal, acquired with the 45 MHz spectrometer.

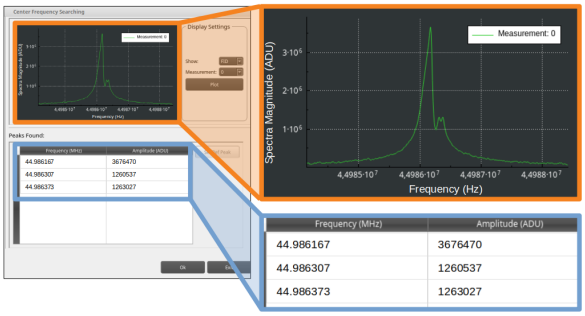


**Figure 15:** Cropped snapshot of the LimeNMR Center Frequency Searching Menu after the sucessful execution of the SDNMR's center frequency searching algorithm with the 45 MHz spectrometer. The sample contains isopropanol.

The acquired signal was obtained under a very inhomogeneous magnetic field. Nevertheless, the system is capable of clear peak identification independently of the NMR signal's linewidth.

### 4.6. Pulse Length Determination

The results here represented are obtained for the 300 MHz spectrometer, since the probe of the 45 MHz spectrometer was not power efficient.

Due to the phase uncertainty inherent to the LimeSDR-USB FPGA's firmware, the calibration routine was executed in the magnitude nutation spectrum mode. Figure 16 shows a nutation spectrum obtained for a water sample, representing two nodes, using the TX1_1 port with 50 dB gain, at 300 MHz. The 90º and the 180º pulse length determined were 47$\pm$0.5 µs and 94$\pm$0.5 µs, respectively.


**Figure 16:** Magnitude nutation spectrum of a water sample acquired with the 300 MHz spectrometer setup, showing two lobes. The first NMR peak was acquired after a 2 µs pulse duration and the remaining NMR peaks with a 4 µs time increment of the pulse duration. Obtained with LimeNMR.

### 4.7. Relaxation Time Measurements

Given the phase uncertainty inherent to the LimeSDR-USB FPGA's firmware, the calibration routine to measure both relaxation times $T_1$ and $T_2$ was updated with the options to be executed with the magnitude spectrum. Hence the experimental data is fitted with the absolute of equations (3) and (4), respectively.

Figure 17 illustrate the resulting measurement of $T_1$ of a sample containing water and copper(II) sulfate, at 300MHz. As observed, the full evolution process of longitudinal relaxation of $M_z$ was registered.

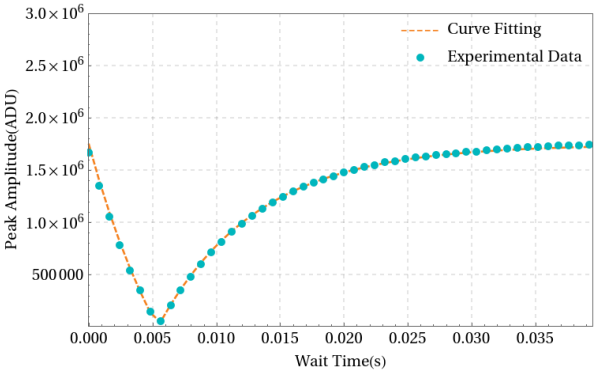


**Figure 17:** $T_1$ measurement results for a water+copper(II) sulfate sample, at 300 MHz, after the execution of an inversion recovery sequence. The results were obtained with LimeNMR and exported to *Mathematica*. The software measured $T_1$ = 7.8 ms.

The obtained value for $T_1$ of 7.8 ms differs approximately $0.2\%$ of the value measured with Bruker's TopSpin software.

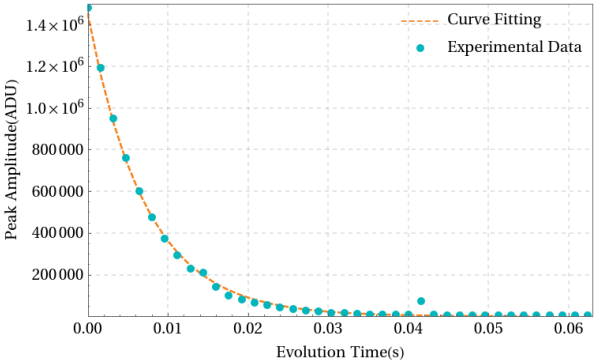


**Figure 18:** $T_2$ measurement results for a water+copper(II) sulfate sample, at 300 MHz, after the execution of a CPMG sequence. The results were obtained with LimeNMR and exported to *Mathematica*. The software measured $T_2$ = 7.2 ms.

Figure 18 illustrate the resulting measurement of $T_2$ of a sample containing water and copper(II) sulfate, at 300MHz. The fitting result shows $T_2$ = 7.2 ms, which differs $0.4\%$ of the same routine executed with Bruker's TopSpin software.

### 4.8. Other NMR Signals

Figures 19 and 20 illustrate the NMR signals obtained for a water sample and an ethanol sample, respectively, with the 300 MHz spectrometer and the acquisition conditions described in the legends.

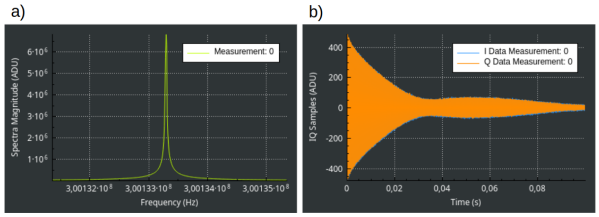


**Figure 19:** a) NMR peak of a water sample, acquired with the 300 MHz spectrometer. b) FID signal of the water sample acquired for 100 ms with a 1 MHz sampling rate. Obtained with LimeNMR.

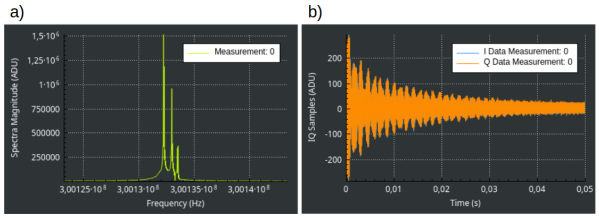


**Figure 20:** a) NMR peak of an ethanol sample, acquired with the 300 MHz spectrometer. b) FID signal of the ethanol sample acquired for 50 ms with a 1 MHz sampling rate. Obtained with LimeNMR.

## 5. Conclusions

### 5.1. Achievements

The developed API's results demonstrate the software's ability to easily generate complex NMR pulse sequences, perform hardware optimization routines, center frequency searching, pulse length determination and relaxation time measurements for NMR samples, for different SDR/MCU interfaces. They can be obtained with a LimeSDR-USB, providing that the transmitting and receiving buffers maximum sizes are considered and its phase uncertainty limitations are respected. These results are also compared to the ones provided by Bruker's software with very small relative deviations. Nevertheless, the limitations found are inherent to the hardware employed and not the software developed. This allows for better performances to be achieved for higher priced hardware, if necessary.

Currently, it offers support only for the LimeSDR-USB SDR board and Texas Instruments MCU boards with the same modules as the LAUNCHXL-F28837s.

The SDNMR and the LimeNMR software source-code, installation instructions and documentation can be found in [15].

### 5.2. Future Work

Some possible future work include: the development of support features for other SDR and MCU devices, the improvement of the already implemented support scripts, the implementation of new and more robust DSP algorithms or new mathematical models for other pulse shapes and, last but not least, the development of more free and open-source API applications.

## References


[1] K. Takeda. Opencore nmr: Open-source core modules for implementing an integrated fpga-based nmr spectrometer. *Journal of Magnetic Resonance*, 2008. doi:10.1016/j.jmr.2008.02.019.

[2] A. Selva et al. Introduction to the software-defined radio approach. *IEEE Latin America Transactions*, 2012. doi:10.1016/j.jmr.2014.01.005.

[3] R. R. Ernst. Nuclear magnetic resonance fourier transform spectroscopy (nobel lecture). 1992. doi:10.1002/anie.199208053.

[4] A. Asfour et al. Software defined radio (sdr) and direct digital synthesizer (dds) for nmr/mri instruments at low-field. *Sensors*, 2013. doi:10.3390/s131216245.

[5] C. Hasselwander et al. gr-mri: A software package for magnetic resonance imaging using software defined radios. *Journal of Magnetic Resonance*, 2016. doi:10.1016/j.jmr.2016.06.023.

[6] C. A. Michal. A low-cost multi-channel software-defined radio-based nmr spectrometer and ultra-affordable digital pulse programmer. *Concepts in Magnetic Resonance*, 2018. doi:org/10.1002/cmr.b.21401.

[7] A. Asfour. *Low-Field NMR/MRI Systems Using LabVIEW and Advanced Data-Acquisition Techniques. Practical Applications and Solutions Using LabVIEW Software*. doi:10.5772/19751.

[8] Lime Microsystems. Lms7002m - fprf mimo transceiver ic with integrated microcontroller.

[9] Texas Instruments. C2000 delfino mcus f28377s launchpad development kit.

[10] F. Bloch et al. Nuclear induction. *American Physical Society*, 1946. doi:10.1103/PhysRev.69.127.

[11] ACM SIGPLAN Conference on Programming Language Design and Implementation. *A Fast Fourier Transform Compiler*, 1999.

[12] Brian Gough. *GNU Scientific Library Reference Manual - Third Edition*. 2009.

[13] Tom O'Haver. *A Pragmatic Introduction to Signal Processing*. 2014.

[14] F. Dalitz, M. Cudaj, M. Maiwald, and G. Guthausen. Low-field permanent magnets for industrial process and quality control. *Progress in Nuclear Magnetic Resonance Spectroscopy*, 2014. doi:10.1016/j.pnmrs.2013.09.001.

[15] J.D.C. Franco. Sdnmr - software defined nuclear magnetic resonance. https://github.com/users/JoaoDFranco/projects/1, 2019.