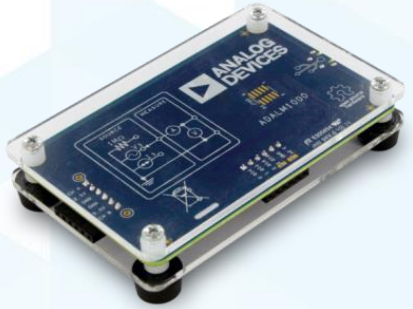
Building a radio with M2K and spare parts

Adrian Suciu

AHEAD OF WHAT'S POSSIBLE™

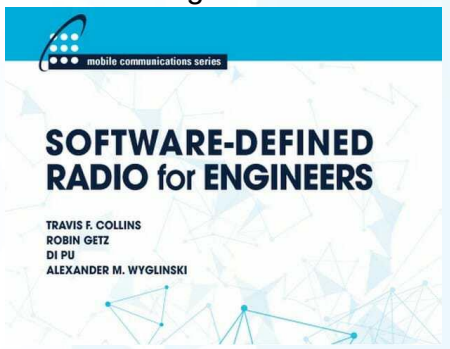
Analog Devices Active Learning Program

- Dedicated to inspiring students to better understand analog real world signals
- Enables integration of technology into course curricula, design and research projects



- Consist of:
  - Active learning modules
    - ADALM-1000
    - ADALM-2000
    - ADALM-PLUTO (SDR)
  - Over 60 free courseware materials & labs https://wiki.analog.com/university/courses/electronics/labs
  - Free books, tutorials, etc.

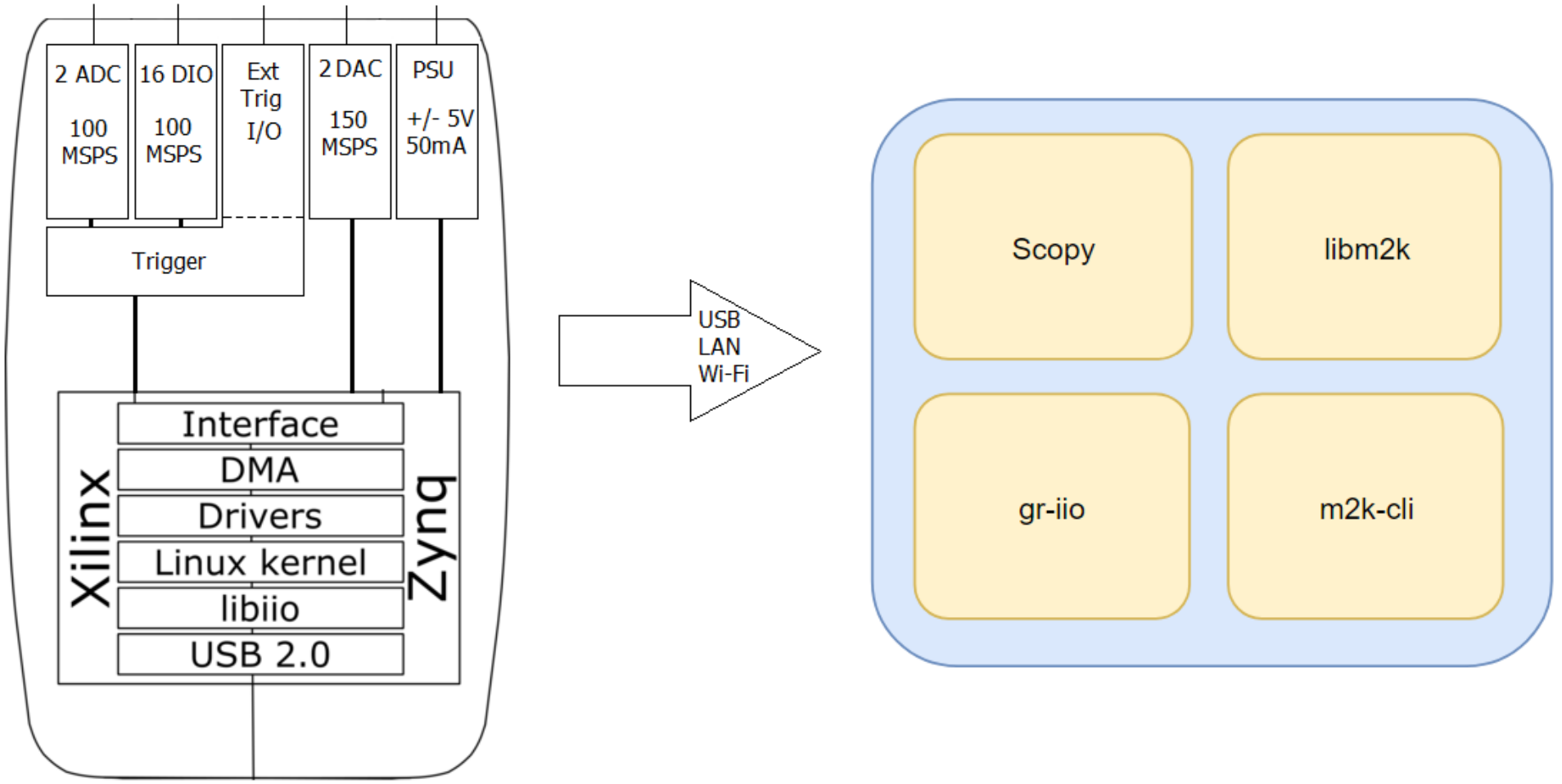
Software-Defined Radio for Engineers Book



The hardware – ADALM2000

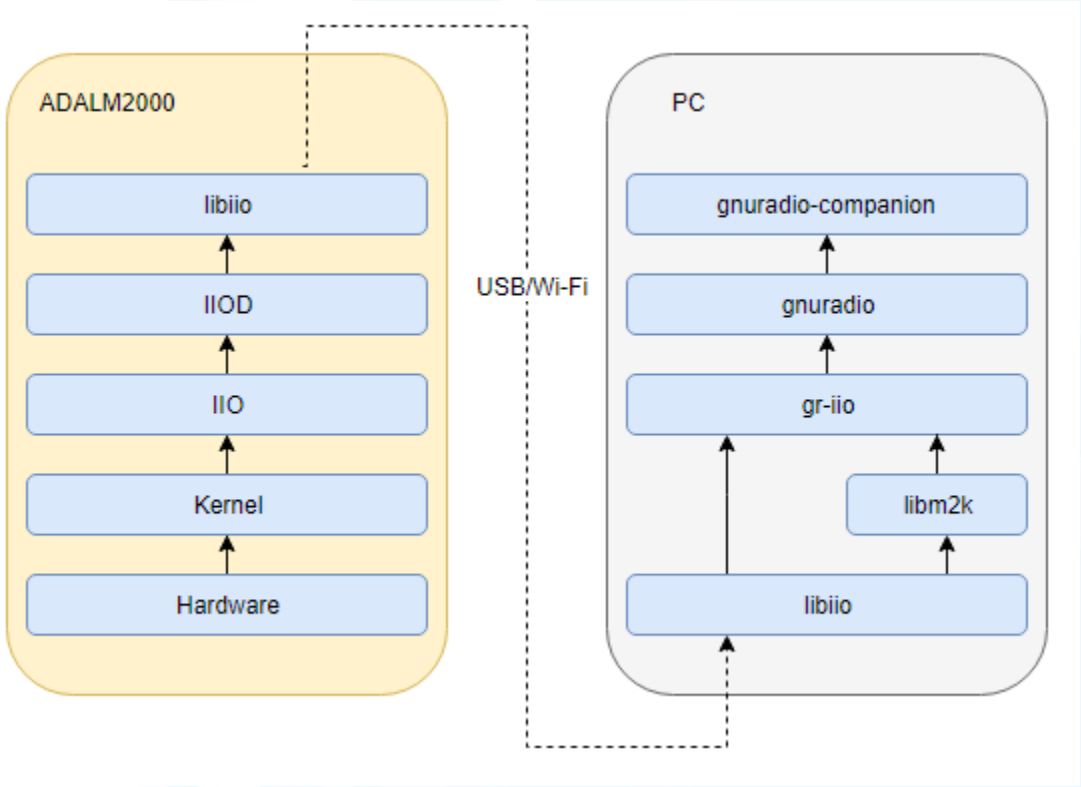
- Affordable USB powered multifunction instrument
- Same infrastructure as the PLUTO
  - ZYNQ Z7010 SoC running Linux
  - AD9963
  - Connectivity options (USB, WiFi, LAN, memory stick)
  - Secondary power connector
- Open source hardware and firmware
  - https://github.com/analogdevicesinc/m2k-fw
  - https://wiki.analog.com/university/tools/m2k/hacking/hardware

M2K block diagram and interface



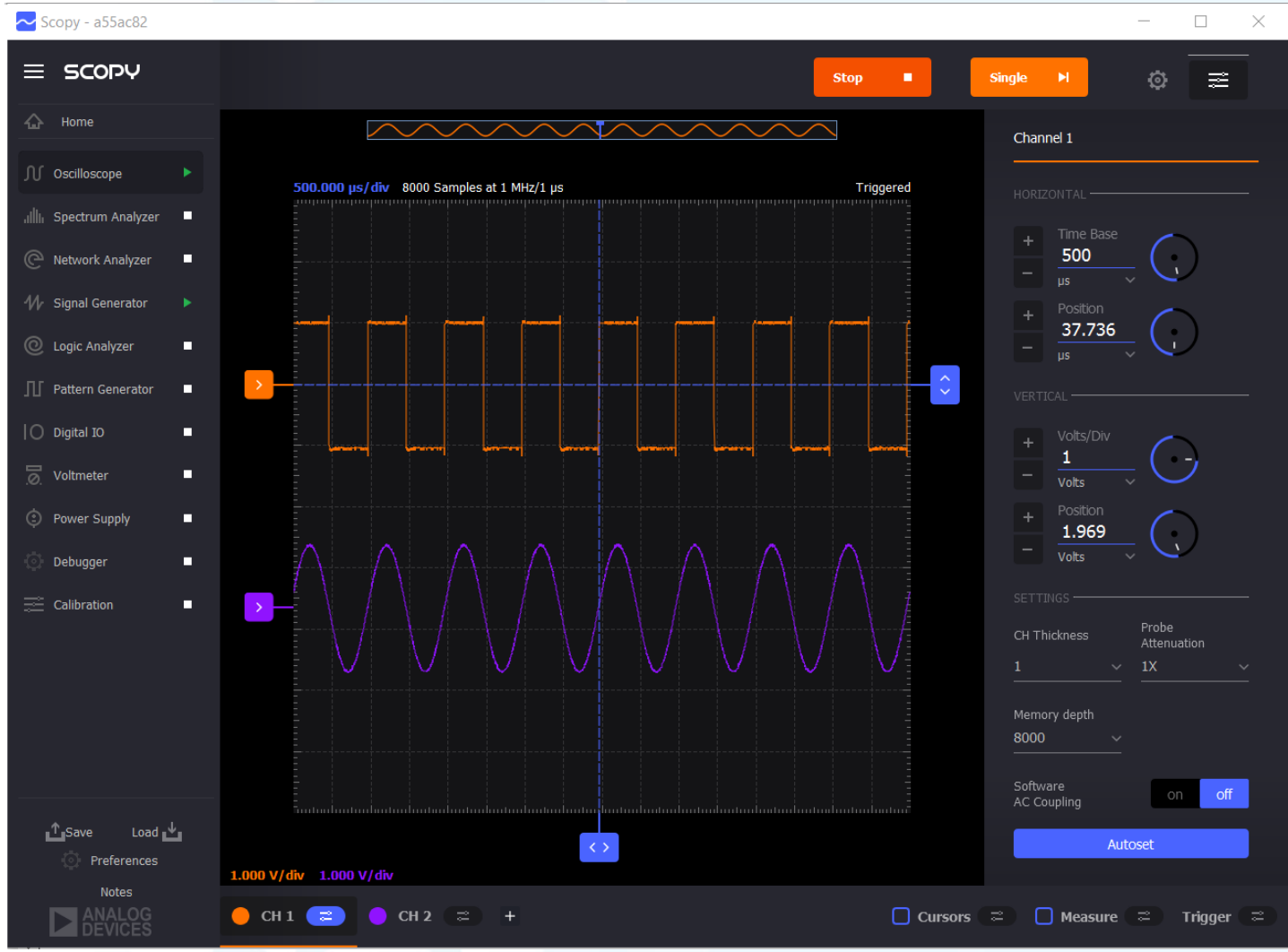
Analog Signal Chain

- ADALM2000 is an embedded Linux host
- Uses IIO subsystem to manage its inputs and outputs
- libiio is a system library that abstracts the low-level details of the IIO subsystem
- IIOD provides IIO data remotely to clients via USB, IP or even Serial
- libm2k uses libiio to implement the M2K interface
- gr-iio is used as an interface between GNU Radio and IIO devices (M2K)
- gnuradio & gnuradio-companion are used for signal processing



The software - Scopy

- Touch-friendly, modern-looking, Qt based application (Windows, Linux, MacOS)
- Features several virtual instruments:
  - Oscilloscope
  - Signal generator
  - Voltmeter
  - Power supply
  - Spectrum analyzer
  - Network analyzer
  - Logic analyzer
  - Pattern generator
- GNU Radio is used for signal processing and acquisition
- Open source - https://github.com/analogdevicesinc/scopy



m2k-cli

- Command line interface tools for the M2K
- Used for
  - calibration
  - configuration
  - push/get samples
- Can be integrated in bash scripts for device configuration and measurement automation

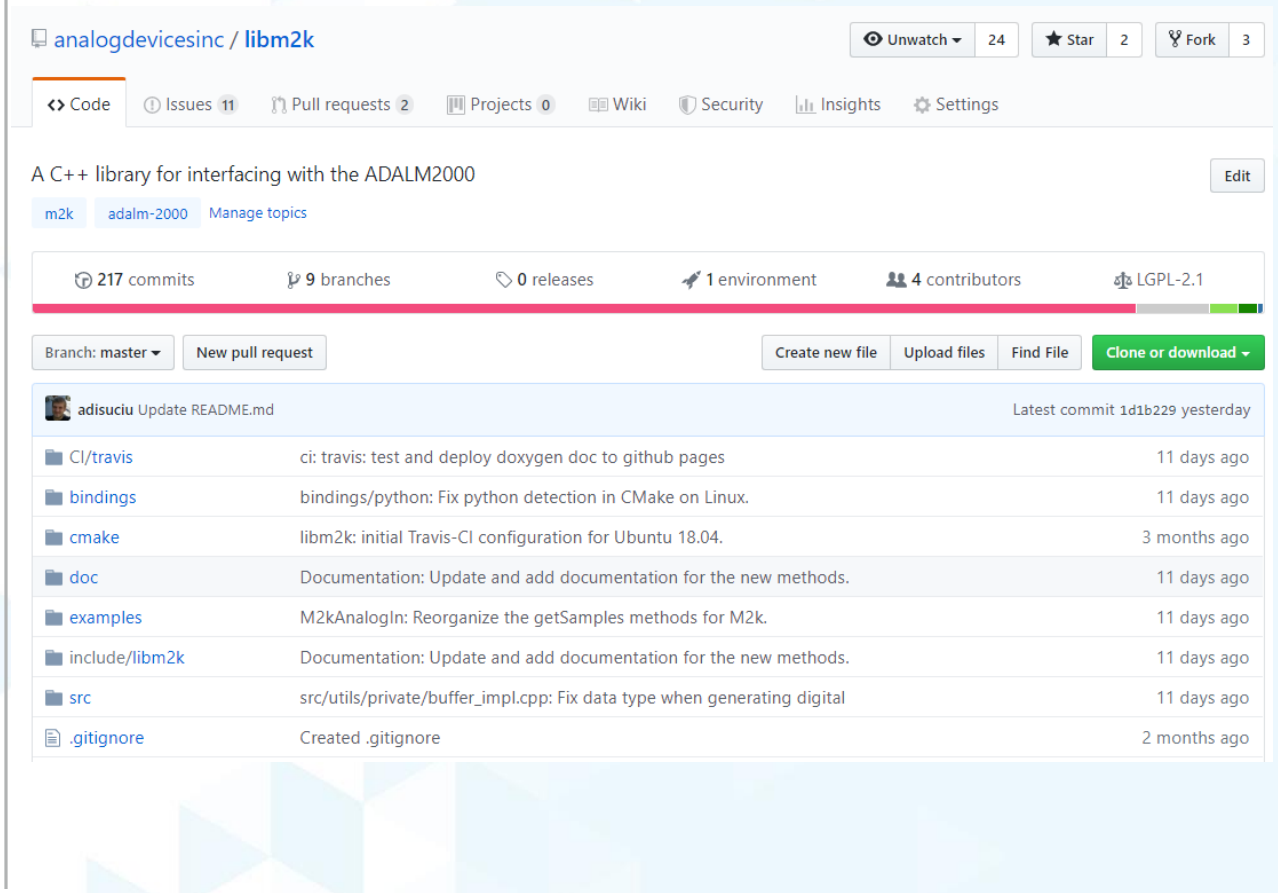
```
teo@teo:~$ m2kcli --help
usage: m2kcli [-h | --help] [-v | --version]

Control the ADALM2000 from the command line

optional arguments:
  -h, --help            show this help message and exit
  -v, --version         return the current version
  --stop-analog         stop the generation of the analogical signal
  --stop-digital        stop the generation of the digital signal

commands:
  These commands represent the components of the ADALM2000

  {analog-in,analog-out,digital,power-supply,dmm}
    analog-in           control the analogical input component
    analog-out          control the analogical output component
    digital             control the digital component
    power-supply        control the power supply
    dmm                 control the digital multimeter
```

The library – libm2k

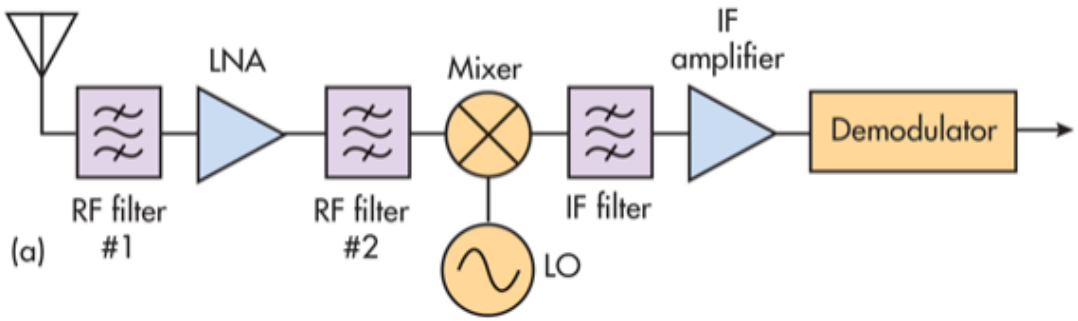
- An C++ library to interface with the M2K
- Abstracts all device initialization, calibration, operation in simple, easy to use methods
- Built on top of libiio, minimal dependencies

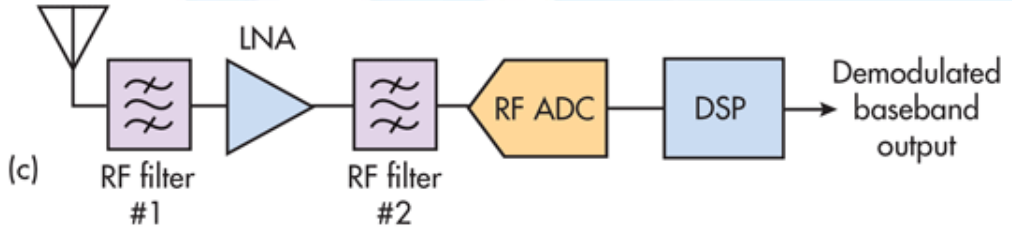
- Build instructions for Windows/Linux/MacOS
- Windows installer
- Bindings for Python, C# and MATLAB (WIP)

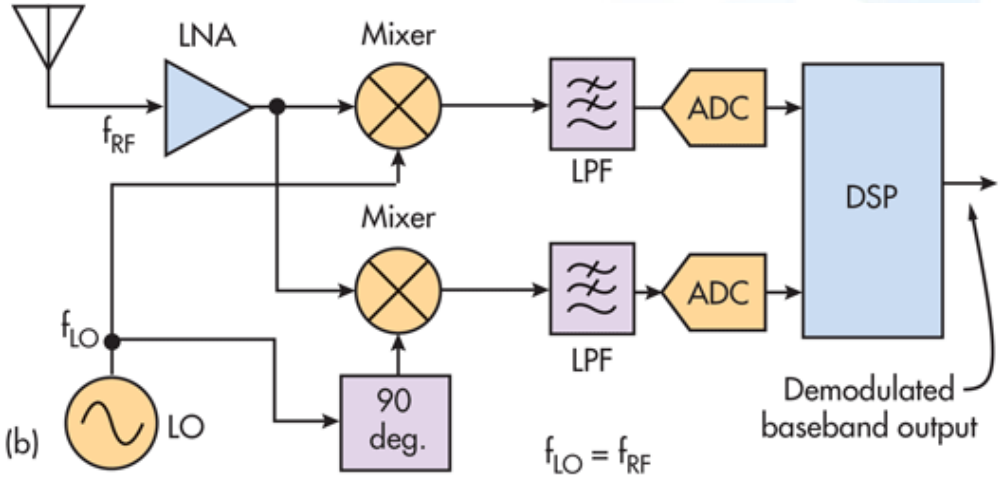
- Opensource licensing LGPLv2.1
  https://github.com/analogdevicesinc/libm2k



Basics: Radio Architectures

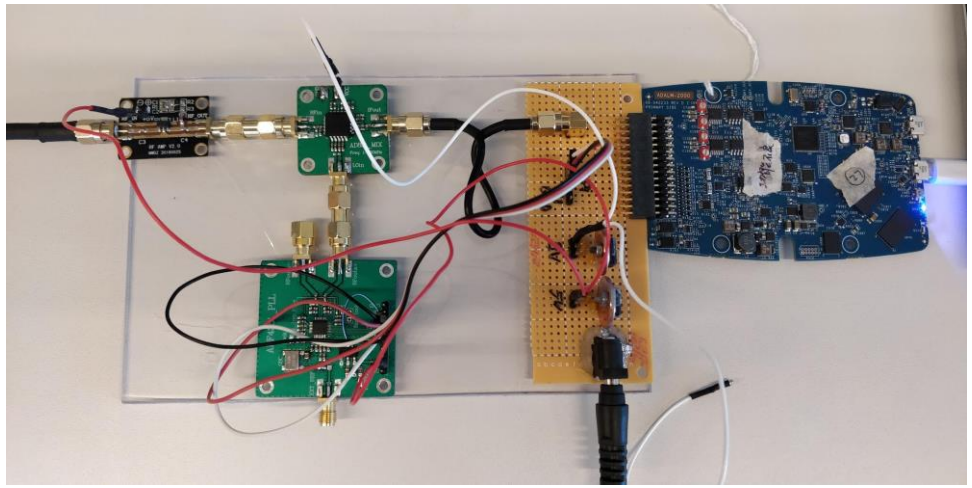


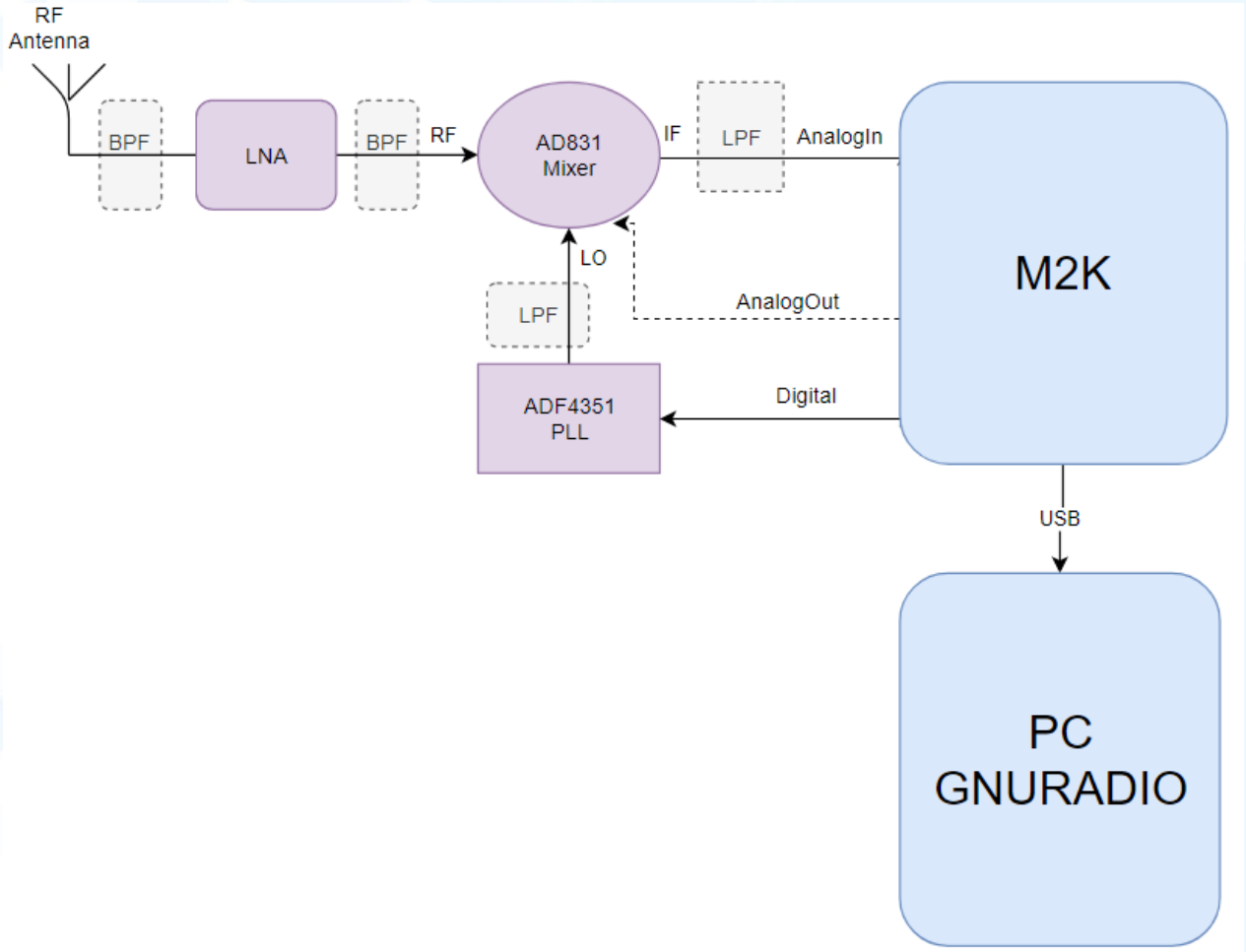




https://www.electronicdesign.com/adc/high-speed-rf-sampling-adc-boosts-bandwidth-dynamic-range
Lou Frenzel | Feb 22, 2017

How to build a radio with the M2K

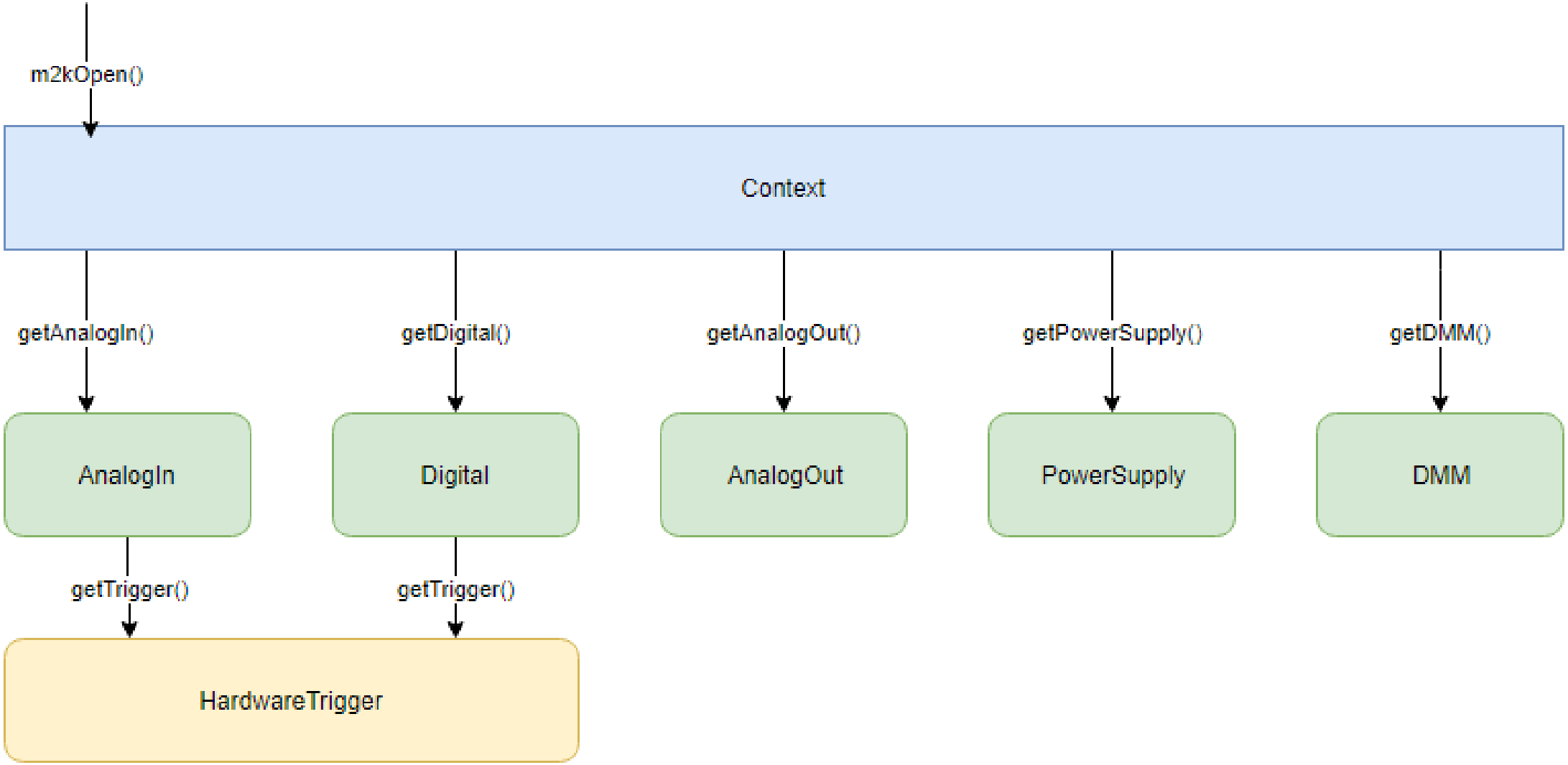




- Parts used – COTS:
  - FM Antenna – 10$
  - LNA – 10$
  - AD831 mixer – 10$
  - ADF4351 PLL – 25$
  - SMA connectors & voltage regulators – 5$

  Total cost ~ 60$

libm2k - block diagram



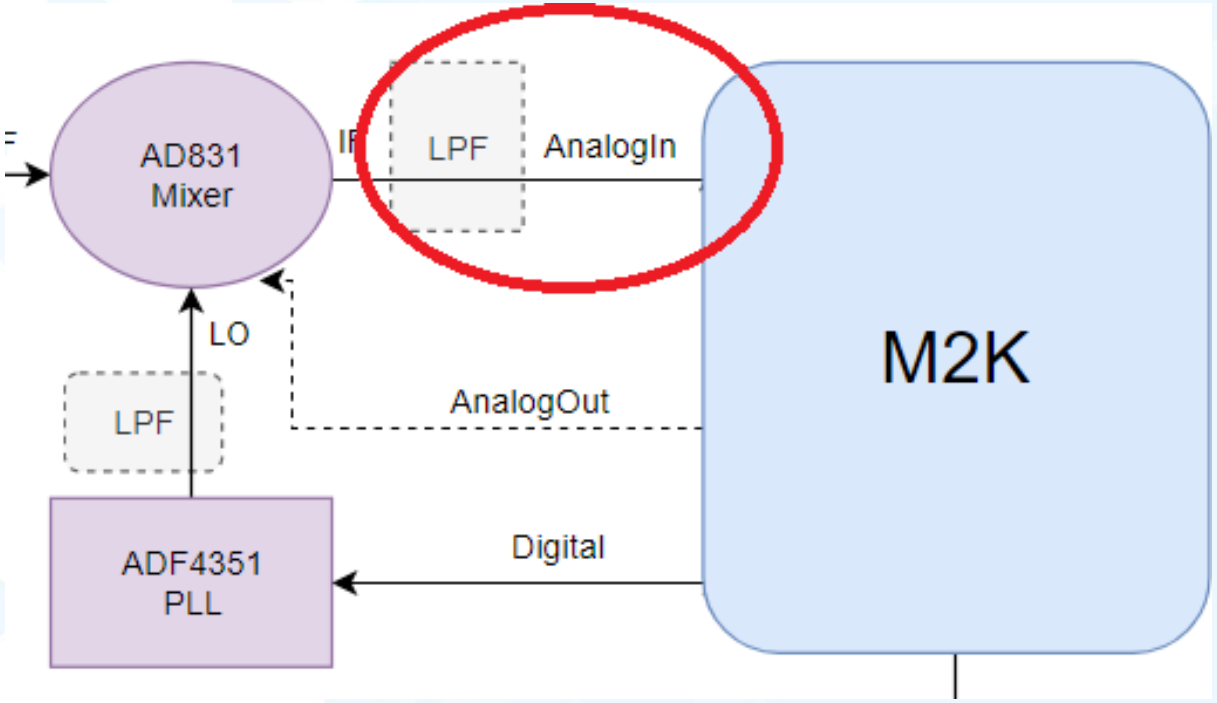
libm2k - Contexts

- A libm2k context is basically a device handle
- An M2KContext provides methods for
  - calibration
  - device initialization
  - device identification
  - handles for all of the instruments
- Multiple devices can be instantiated with libm2k

```
1  import libm2k as l
2
3  channel = 0
4
5  ctx=l.m2kOpen()
6  ctx.calibrateADC()
7  ain=ctx.getAnalogIn()
8  ain.enableChannel(channel,True)
9  print(ain.getVoltage()[channel])
```

libm2k - AnalogIn

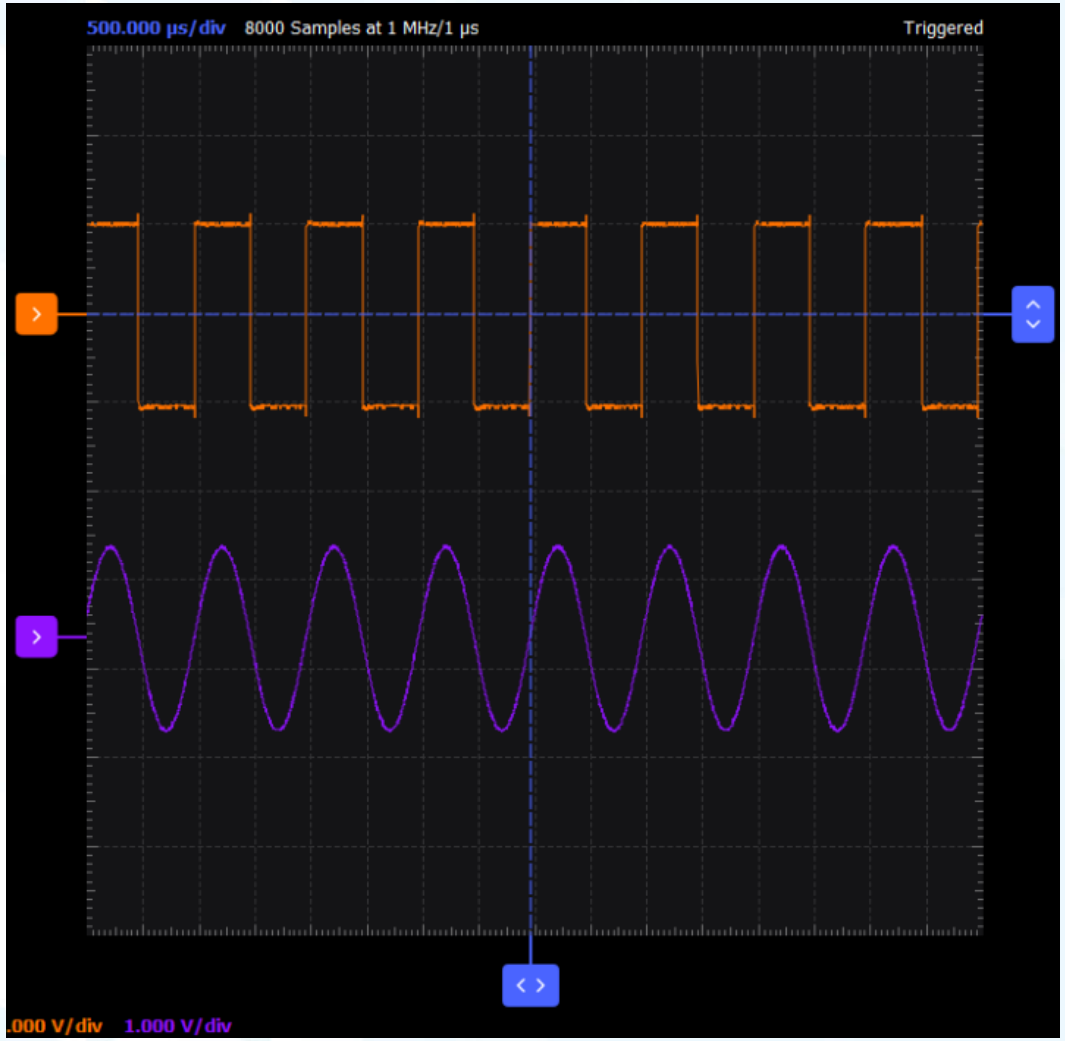
- Interface to the 2-channel 12-bit 100MSPS analog input
- Works at baseband (0-20MHz)
- +/-25V (low gain) +/-2.5V (high gain)
- Provides methods for acquiring raw and converted-to-volts data
- Signal acquisition can be conditioned by the trigger mechanism
- Provides getters and setters for parameters of the AnalogInput such as:
  - sample rate
  - buffer size
  - gain mode
  - others



libm2k - HardwareTrigger

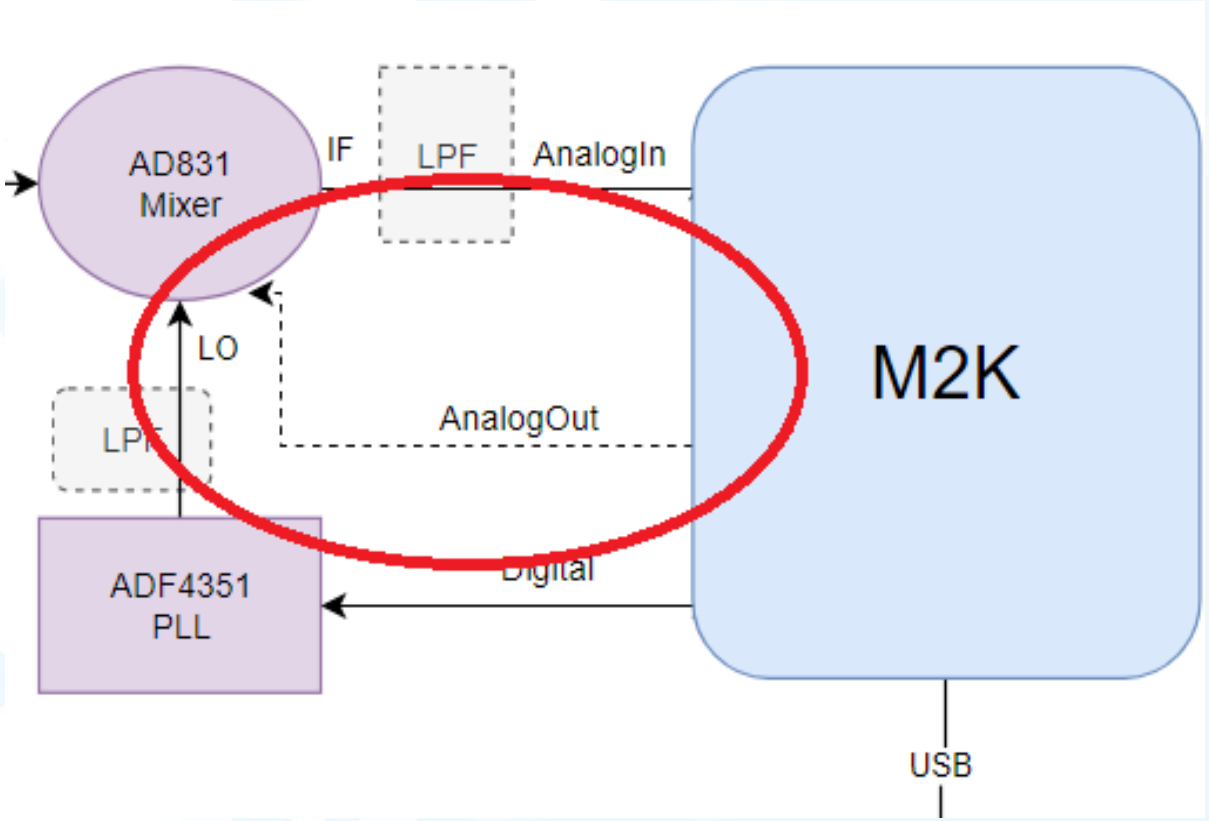
- M2K has a versatile trigger mechanism that can be configured using the HardwareTrigger handle
- Conditions signal acquisition (by edge type or signal amplitude)
- 2 types of triggers – internal and external (TI)
- It is possible to daisychain multiple M2Ks using the trigger out (TO) and connecting it to the trigger in(TI) of another M2K.

- Analog trigger not useful for our project – we want to gather all data continuously, however the digital trigger is.



libm2k - AnalogOut

- The M2K has a 12 bit, 150MSPS, two-channel synchronized DAC
- Similar to AnalogIn, the class provides interfaces to configure and push data(in raw or volt format) to the AnalogOut interfaces
- Can be used to provide LO signal up to 20MHz or provide IF for a radio transmitter.

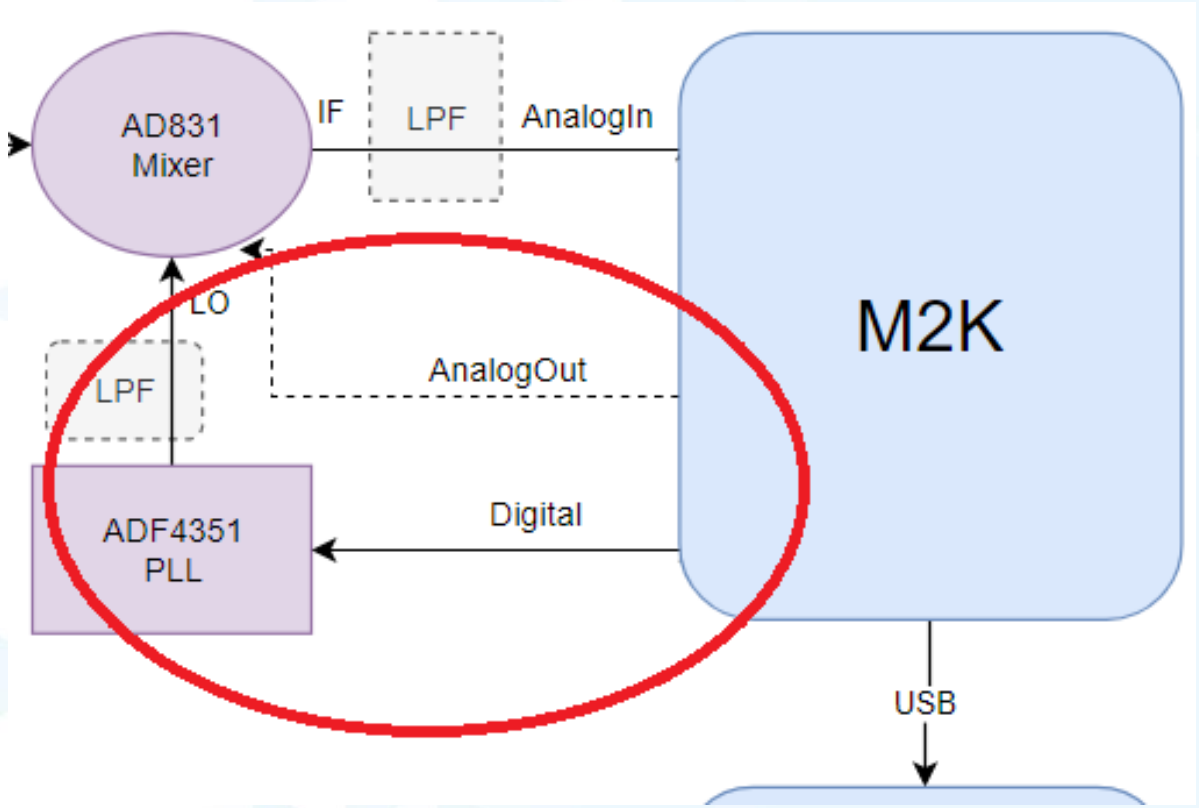


libm2k – PowerSupply & DMM

- The PowerSupply class provides access to a +/- 5V 50mA powersupply on the M2K
- However we couldn’t use this powersupply as the the power requirements for this project are way higher ~ 150mA @12V
- DMM is an interface to M2K internal voltages/temperatures that can be used for device monitoring

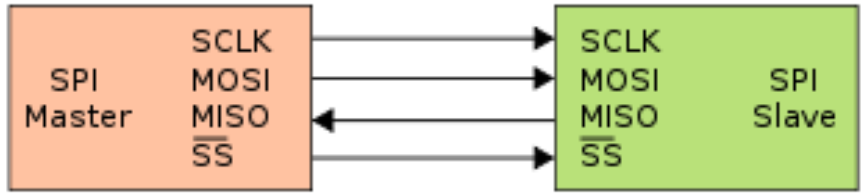
libm2k - Digital

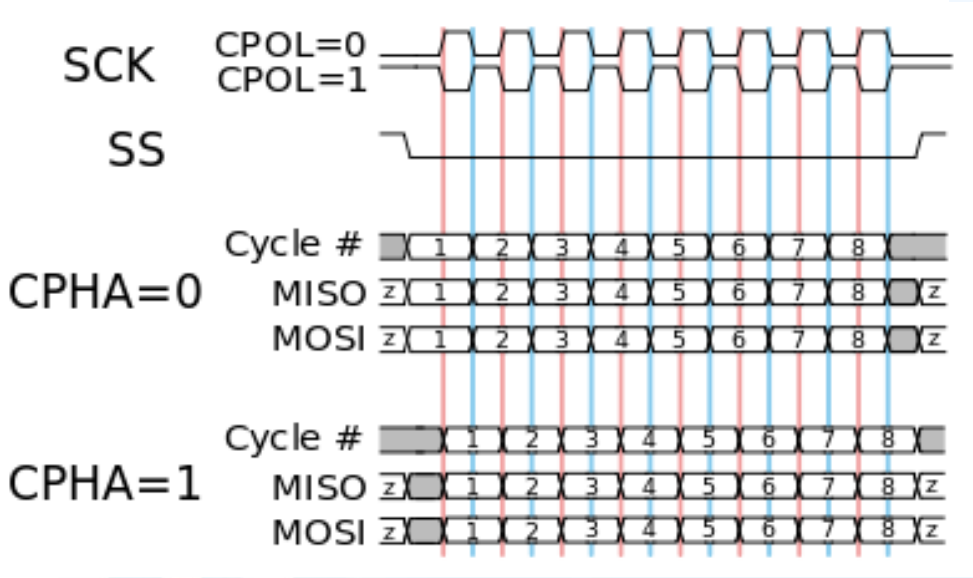
- The digital interface of the M2K provides 16 digital I/Os capable of buffered (100MSPS) and triggered operation
- When configured as output, the pins can be either push-pull or open-drain making them ideal to bitbang a variety of protocols (e.g. SPI/I2C).
- We used the Digital interface to configure the ADF4351 PLL to generate LO frequencies up to 500MHz



M2K SPI Engine

- The SPI protocol can be used to configure a variety of devices
- We created a SPI encoder/decoder using the M2K's digital interface to make interaction between the M2K and SPI devices easier (such as the ADF4351)
- The API is compatible with the drivers from the No-OS repository - developers can use readily available device drivers to configure the devices without any modifications.

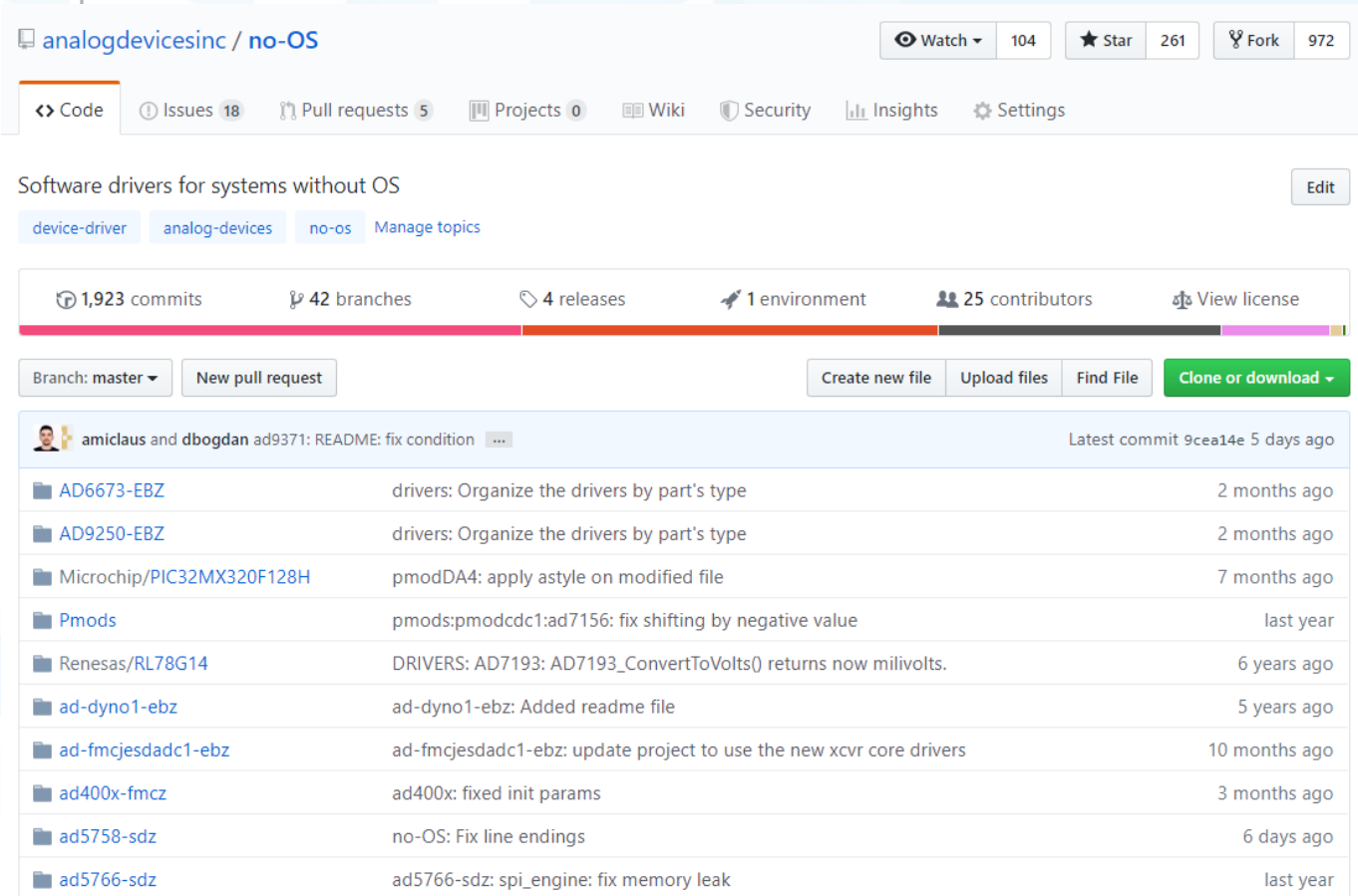




No-OS driver repository

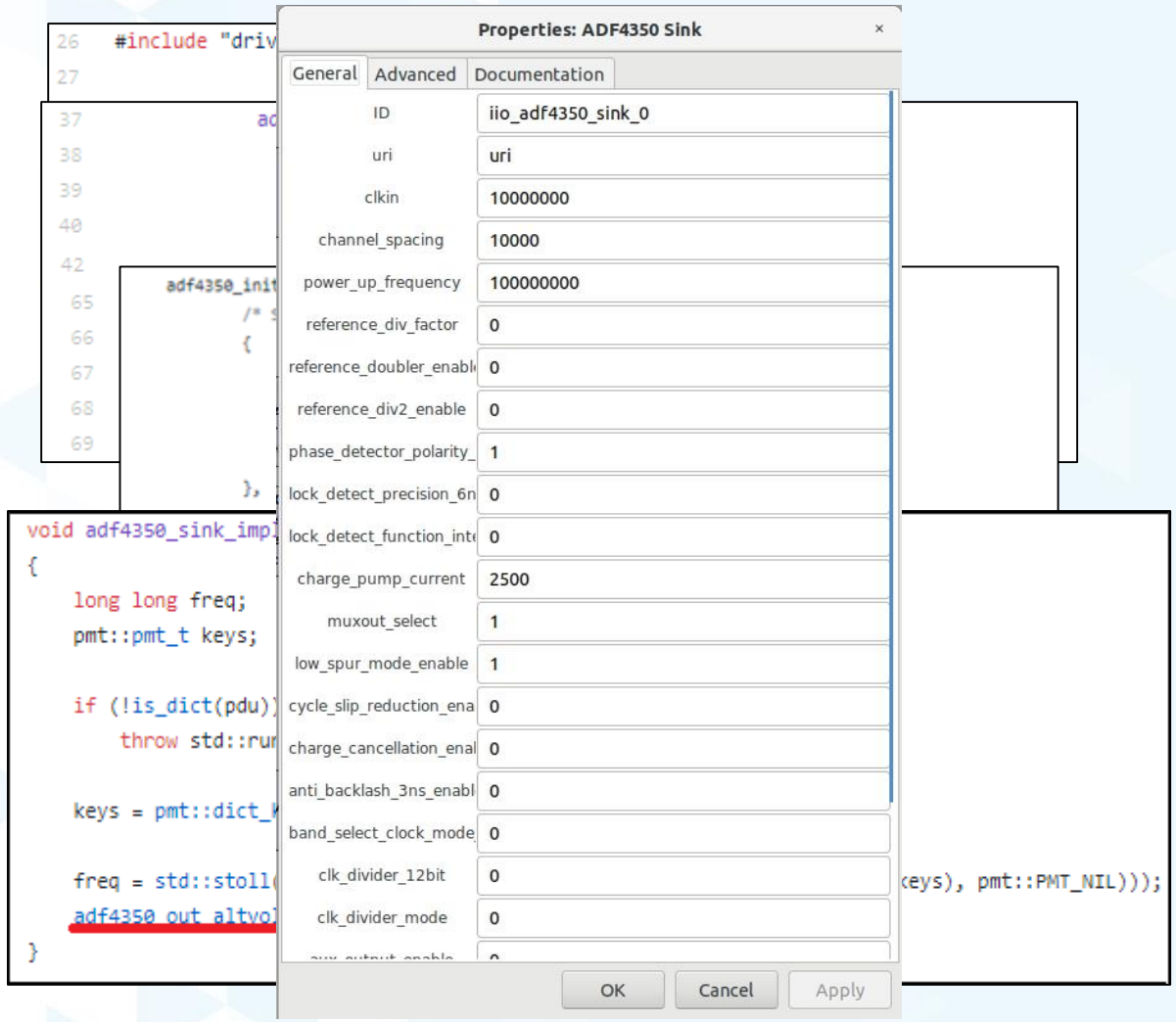
- https://github.com/analogdevicesinc/no-os
- Contains drivers for ADI components that should be used in user-space or microcontrollers without an OS.
- Not targeted to a specific hardware platform
- SPI devices interface through 3 functions:

```
87  /* Initialize the SPI communication peripheral. */
88  int32_t spi_init(struct spi_desc **desc,
89                   const struct spi_init_param *param);
90
91  /* Free the resources allocated by spi_init(). */
92  int32_t spi_remove(struct spi_desc *desc);
93
94  /* Write and read data to/from SPI. */
95  int32_t spi_write_and_read(struct spi_desc *desc,
96                             uint8_t *data,
97                             uint8_t bytes_number);
98
```

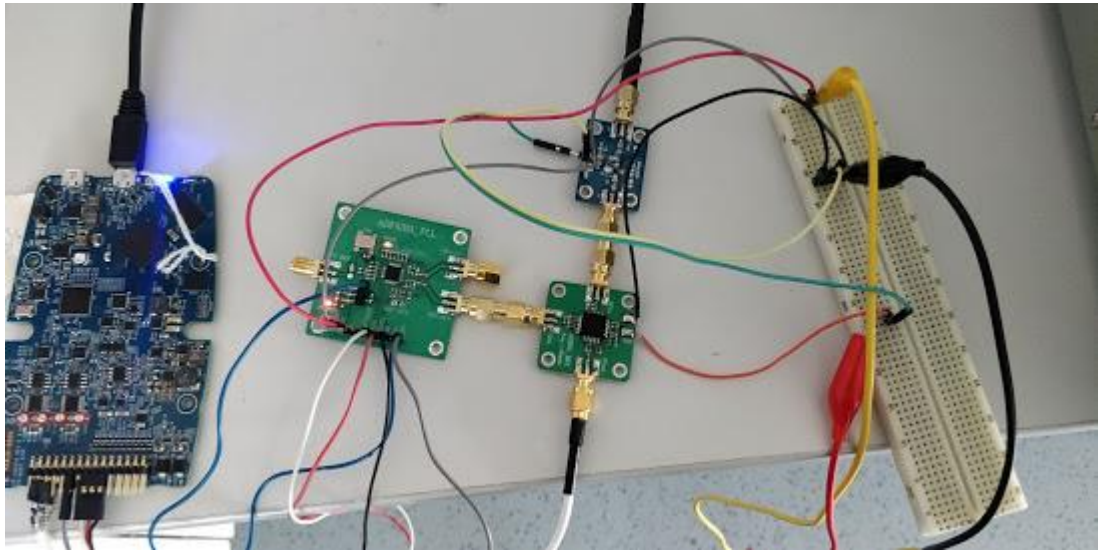


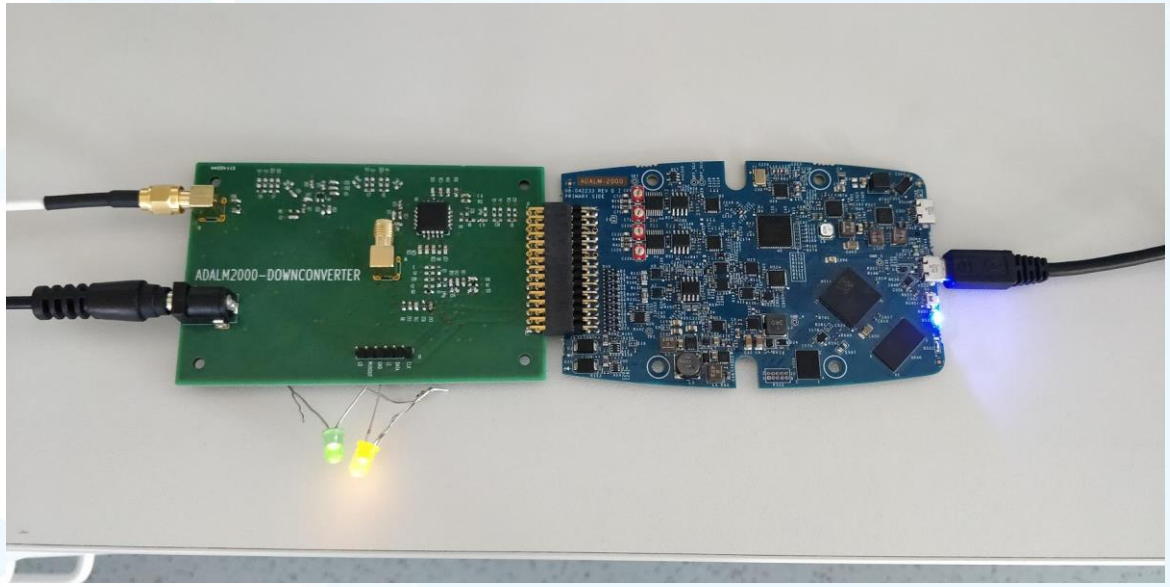
ADF4351 Sink

- ADF4351 – VCO range 35MHz to 4.4GHz
- Output frequency/power configurable through SPI interface
- Easy to use driver in the No-OS repository
- https://github.com/analogdevicesinc/gr-iio/tree/grcon2019_m2k
- With minimal modifications we used the ADF4351 driver to configure the PLL using the M2K SPI engine
- We created a gnuradio block that on message receipt changes the frequency of the PLL (one function call from driver !)



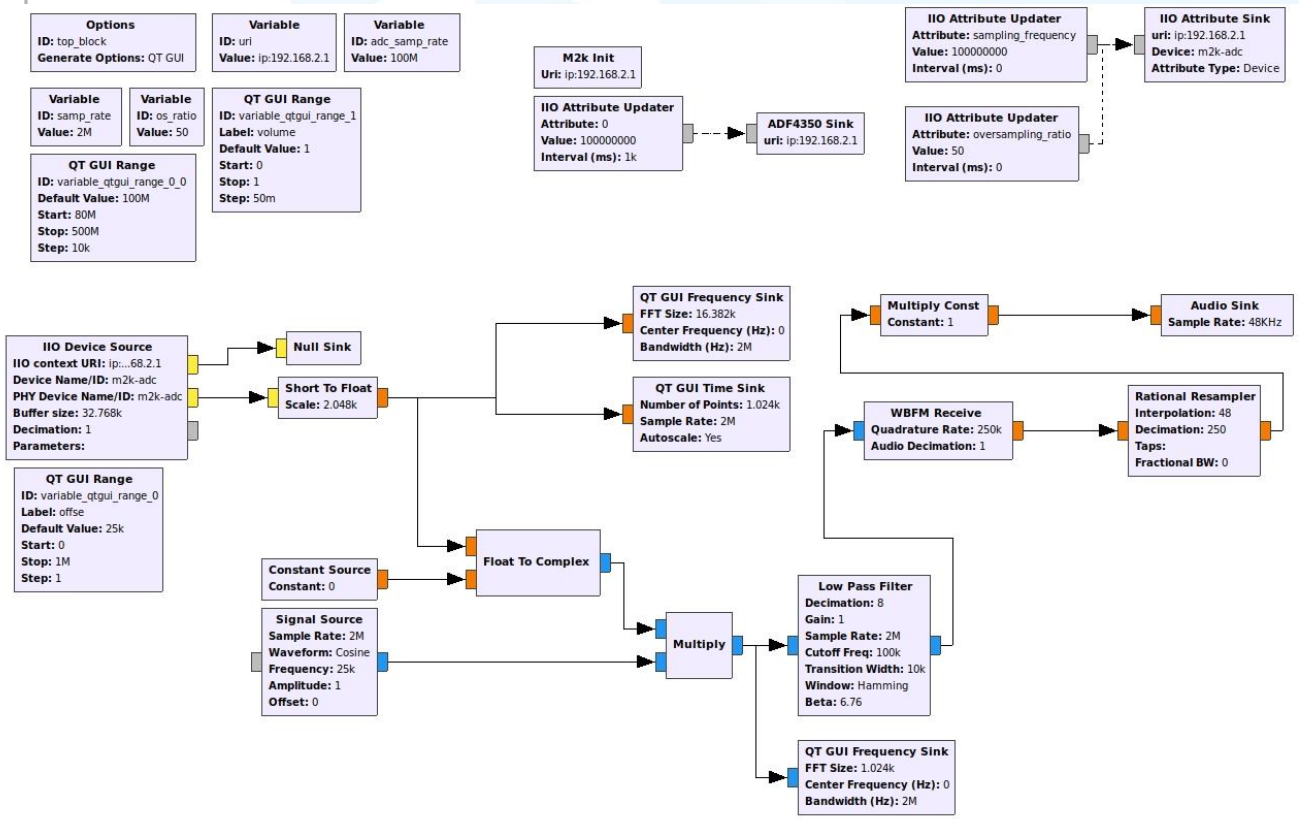
How it all fits together





How it all fits together - II

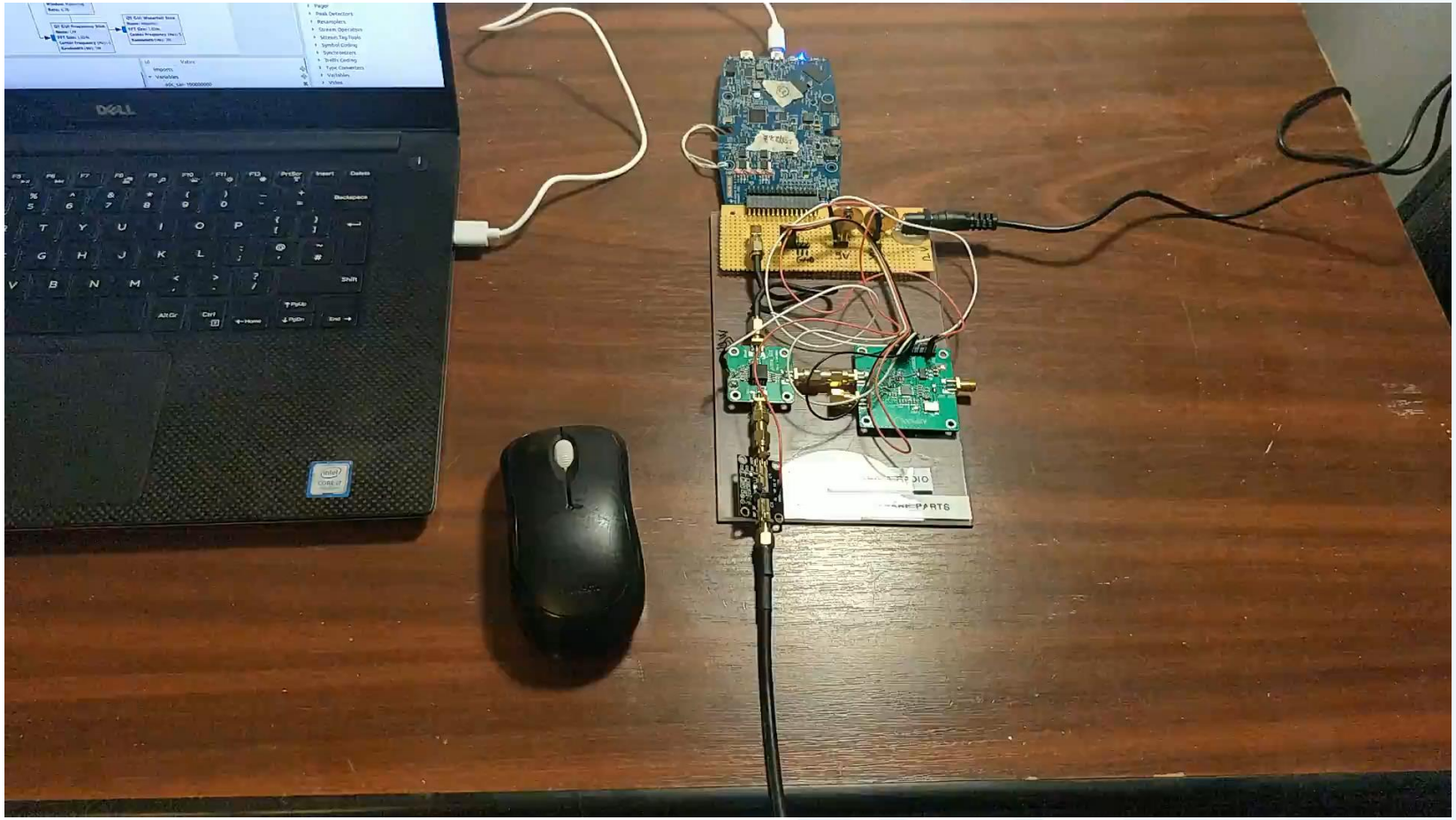
- We created a python script that initializes the libm2k in the gnuradio flow
- Used the ADF4351 sink to configure the PLL
- Use an IIO device source to acquire data from the device
- We converted the signal from short to float to complex, ran it through a LPF, demodulate it with the WBFM Receive block and resampler to get the audio data.



Hello FM Radio world

ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

DEMO



Conclusions & Future plans

- It is possible to use the M2Ks analog front end to capture RF data with proper circuitry
- The M2K and GNUradio can make a good combo to get started with electronics and even SDR

- In the future we want to extend libm2k functionality
  - create a gr-m2k – based on gr-iio – that provides more ways to interact with the M2K through gnuradio
  - create interfaces for I2C and UART protocols for better integration with no-os drivers
  - extend m2k-cli to control digital interfaces such as SPI/I2C/UART
  - add Debian packaging for libm2k and scopy

Thanks !