Quantum Sensing Contribution DIFA/INFN

Matteo Franchini on behalf of the Q-team:
Bellagamba, Bruni, De Castro, Franchini, Rinaldi, Semeria, Semprini, Sioli, Travaglini

Istituto Nazionale di Fisica Nucleare

Quantum Sensing - Status

* Developing the **RFSoc-DAQ system** for the winter single-qubit data-taking and qubit characterisation. Partially done for May data-taking.

* **Analysing data taken** in May with a single planar qubit (tunable). Betchelor thesis of *Carlo Lovecchio*.

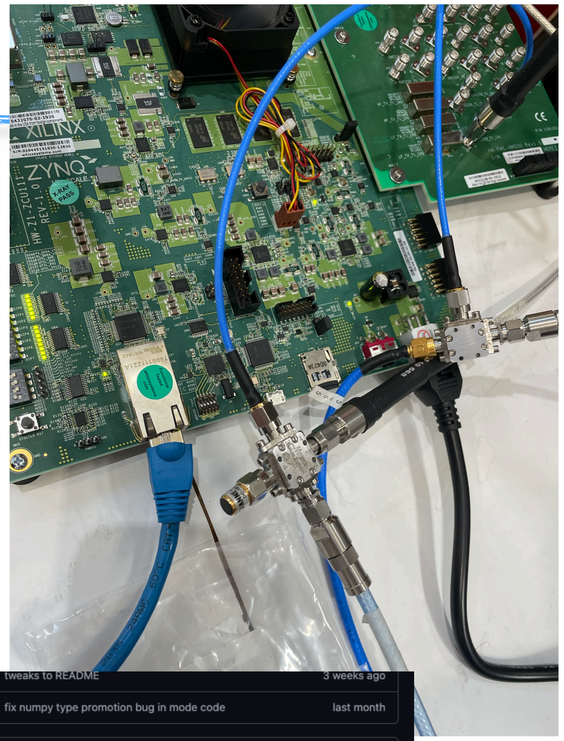
* **Future:** help developing the **RFSoc-DAQ** system for QUART&T prototypes + interested in chip **architecture** studies and **simulation** in collaboration with theorists.

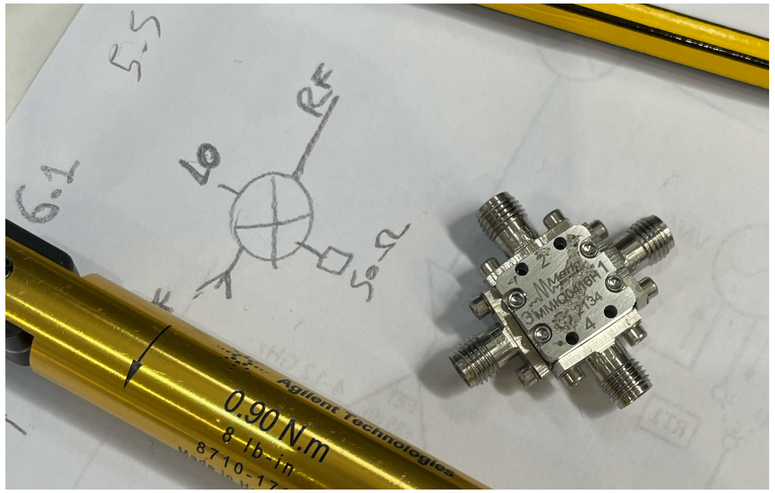
RFSoc DAQ

base.py 3

Users > matteofranchini > MyScripts > QubIT > boarTest > base.py > ...

```python
import json
import socket
from qibosoq.client import execute,convert_commands,connect
from qibosoq.components.base import (
    Qubit,
    OperationCode,
    Config
)
from qibosoq.components.pulses import Rectangular


Run Cell | Run Below | Debug Cell
#%%
HOST = "qubit1.bo.infn.it"
PORT = 6000
Run Cell | Run Above | Debug Cell
#%%
pulse_1 = Rectangular(
            frequency = 500, #MHz
            amplitude = 1000,
            relative_phase = 0,
            start_delay = 0,
            duration = 40,
            name = "drive_pulse",
            type = "drive",
            # shape="Gaussian(5)",
            dac = 6,
            adc = None
)

pulse_2 = Rectangular(
            frequency = 640, #MHz
            amplitude = 0.05,
            relative_phase = 0,
            # gain = 10,
            # phase = 0,
            # freq = 640,
            start_delay = 0,
            duration = 0.1,
```





README.md tweaks to README 3 weeks ago

setup.py fix numpy type promotion bug in mode code last month

README MIT license

QICK

QICK: Quantum Instrumentation Control Kit

The QICK is a kit of firmware and software to use the Xilinx RFSoC to control quantum systems.

It consists of:

New in v0.2.12 New quantum information features and simulation enhancements!

Qibo: an open-source middleware for quantum computing

An end-to-end open source platform for quantum simulation, self-hosted quantum hardware control, calibration and characterization.

$ pip install qibo

Documentation

QICK

* QICK is an open-source qubit controller to be used with Xilinx RFSoC development boards.

* QICK supports the ZCU111, ZCU216, and RFSoC4x2 development boards
  - **recommend using the newer generation of RFSoCs (ZCU216 and RFSoC4x2) for better overall performance**

* Python package, which includes the interface to the firmware
* Jupyter notebooks demonstrating usage

In [1]:
```
# Import the QICK drivers and auxiliary libraries
from qick import *
%pylab inline
```

```
Populating the interactive namespace from numpy and matplotlib
```

In [2]:
```
# Load bitstream with custom overlay
soc = QickSoc()
# Since we're running locally on the QICK, we don't need a separate QickConfig object.
# If running remotely, you could generate a QickConfig from the QickSoc:
#     soccfg = QickConfig(soc.get_cfg())
# or save the config to file, and load it later:
#     with open("qick_config.json", "w") as f:
#         f.write(soc.dump_cfg())
#     soccfg = QickConfig("qick_config.json")
soccfg = soc
print(soccfg)
```

```
QICK configuration:

        Board: ZCU111

        Software version: 0.2.181
        Firmware timestamp: Wed Aug 16 13:39:03 2023

        Global clocks (MHz): tProcessor 384.000, RF reference 204.800

        7 signal generator channels:
        0:      axis_signal_gen_v6 - tProc output 1, envelope memory 65536 samples
                DAC tile 0, blk 0, 32-bit DDS, fabric=384.000 MHz, f_dds=6144.000 MHz
        1:      axis_signal_gen_v6 - tProc output 2, envelope memory 65536 samples
                DAC tile 0, blk 1, 32-bit DDS, fabric=384.000 MHz, f_dds=6144.000 MHz
```

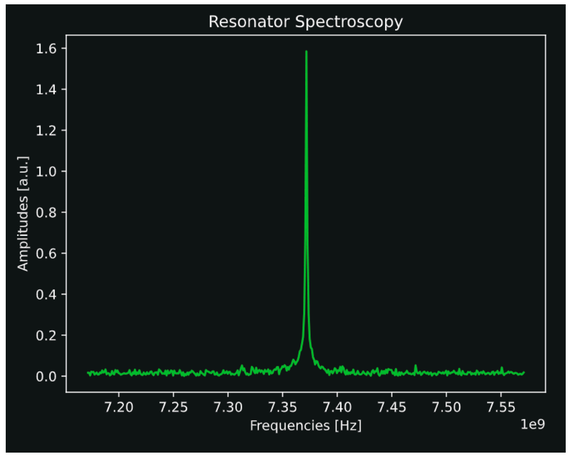
QUIBO

* An open-source full stack API for quantum simulation and quantum hardware control.

* Simply create routines to perform various qubit experiments.
  - Pulse and circuit execution
  - Calibration experiments
    - ☆ Resonator spectroscopy
    - ☆ Qubit spectroscopy
    - ☆ Single shot classification

* Customize instruments and experiments

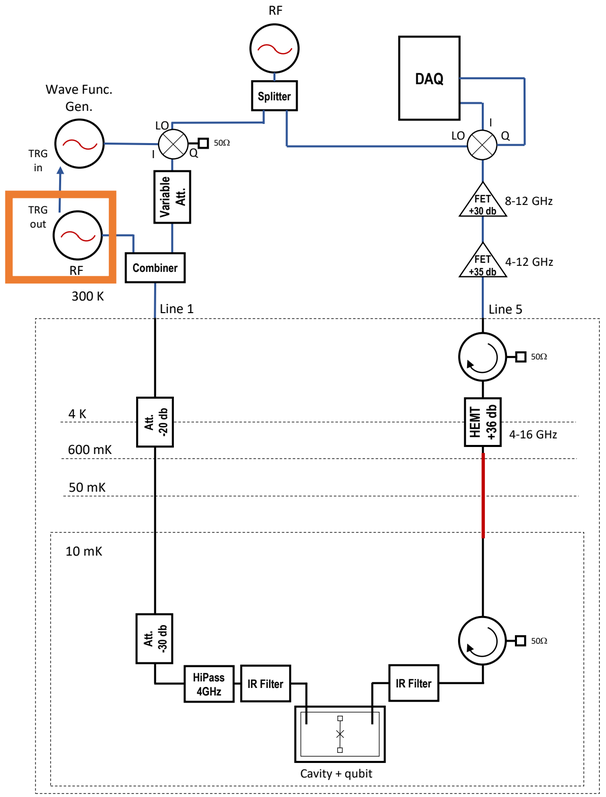
```python
# my_platform/platform.py

import pathlib

from qibolab import (
    AcquisitionChannel,
    Channel,
    ConfigKinds,
    DcChannel,
    IqChannel,
    Platform,
    Qubit,
)
from qibolab.instruments.qm import Octave, QmConfigs, QmController

# folder containing runcard with calibration parameters
FOLDER = pathlib.Path.cwd()

# Register QM-specific configurations for parameters loading
ConfigKinds.extend([QmConfigs])


def create():
    # Define qubit
    qubits = {
        0: Qubit(
            drive="0/drive",
            probe="0/probe",
            acquisition="0/acquisition",
        )
    }
```



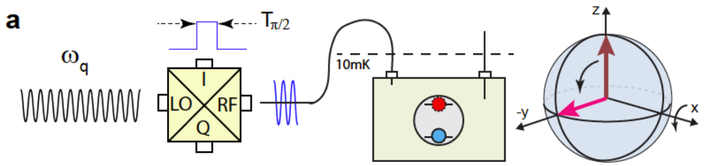
And the we can define the runcard `my_platform/parameters.json`:

```json
{
    "settings": {
        "nshots": 1024,
        "relaxation_time": 70000
    },
    "configs": {
        "0/drive": {
            "kind": "iq",
            "frequency": 4833726197
        },
        "0/drive/lo": {
            "kind": "oscillator",
            "frequency": 5200000000,
            "power": 0
        },
        "0/probe": {
            "kind": "iq",
            "frequency": 7320000000
        },
        "0/probe/lo": {
            "kind": "oscillator",
            "frequency": 7300000000,
            "power": 0
        },
        "0/acquisition": {
            "kind": "qm-acquisition",
            "delay": 224,
            "smearing": 0,
            "threshold": 0.002100861788865835,
            "iq_angle": -0.7669877581038627,
            "gain": 10,
            "offset": 0.0
        }
```

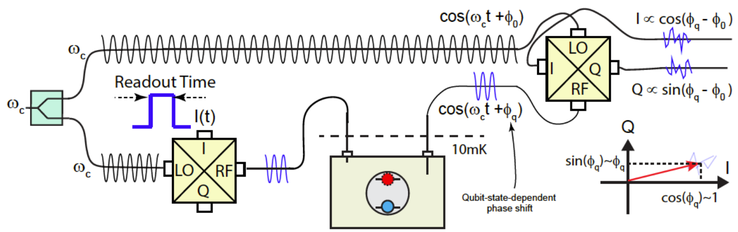
Experimental Setup

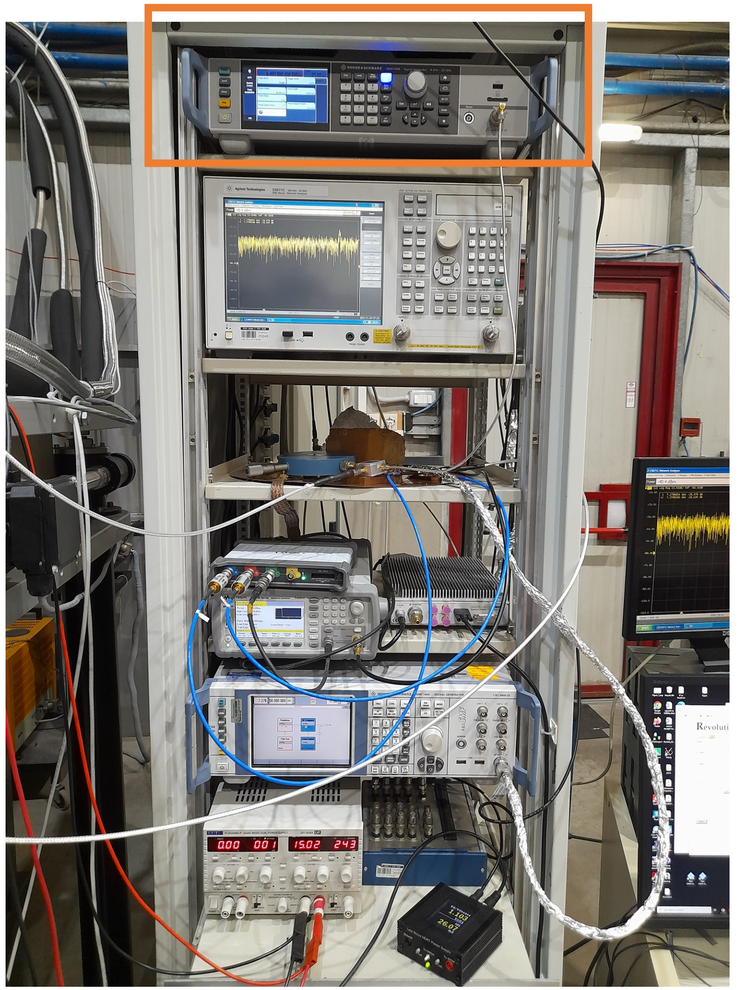


Qubit Control

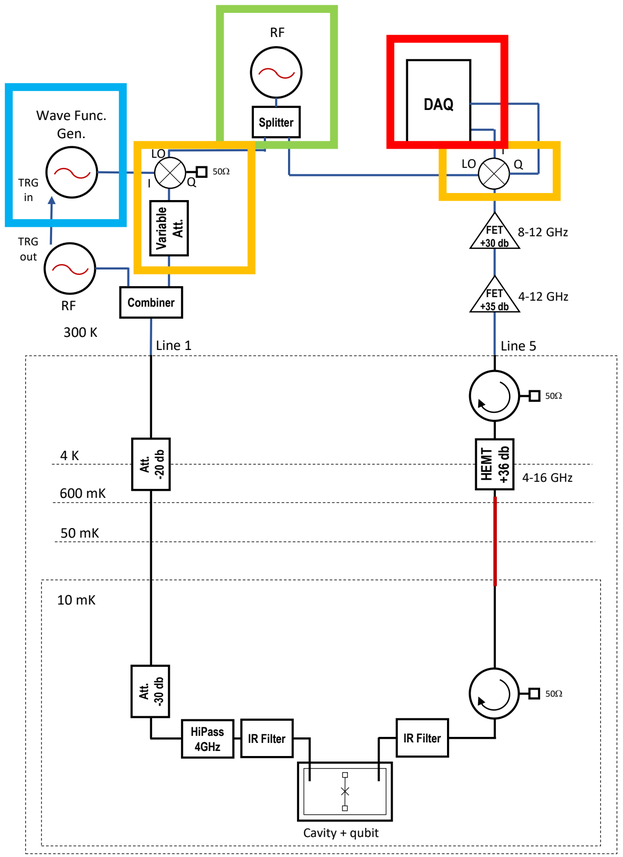


Qubit Readout

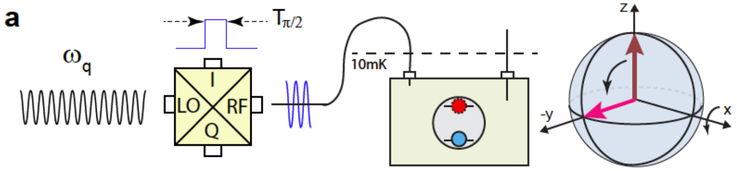




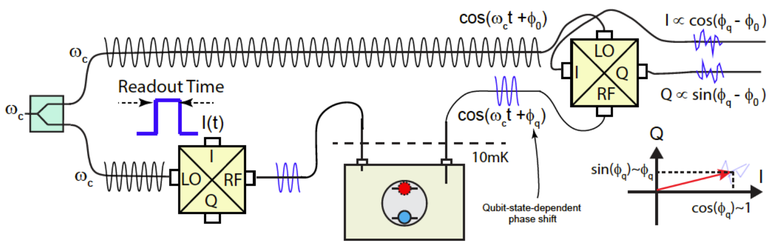
Experimental Setup

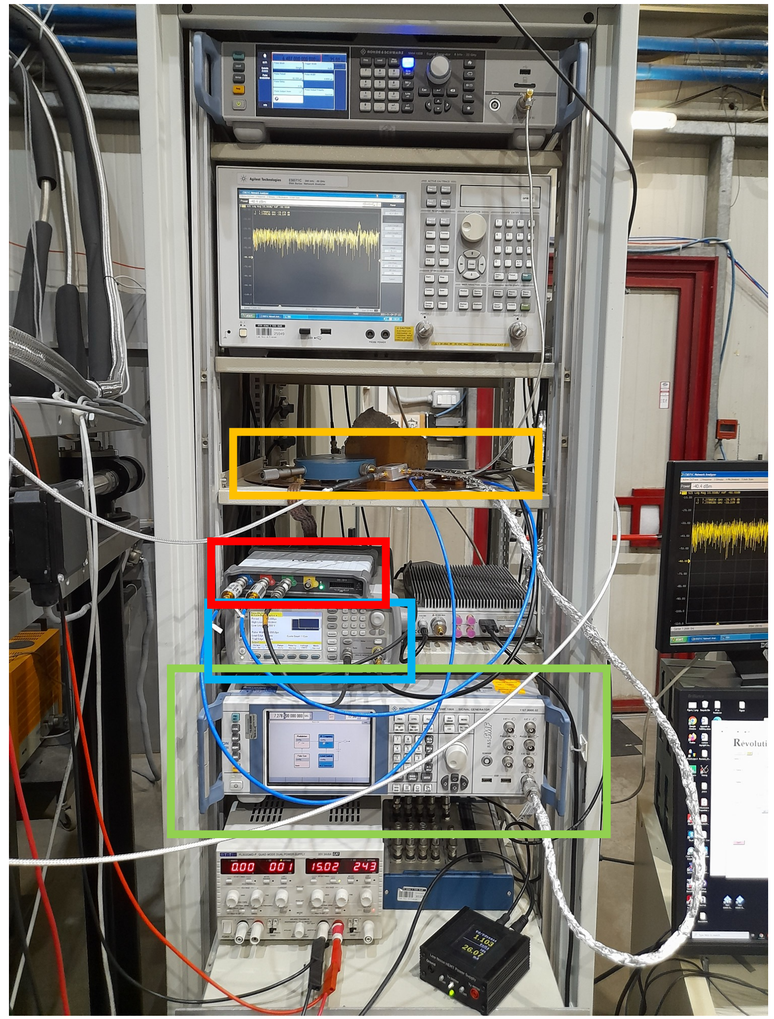


Qubit Control



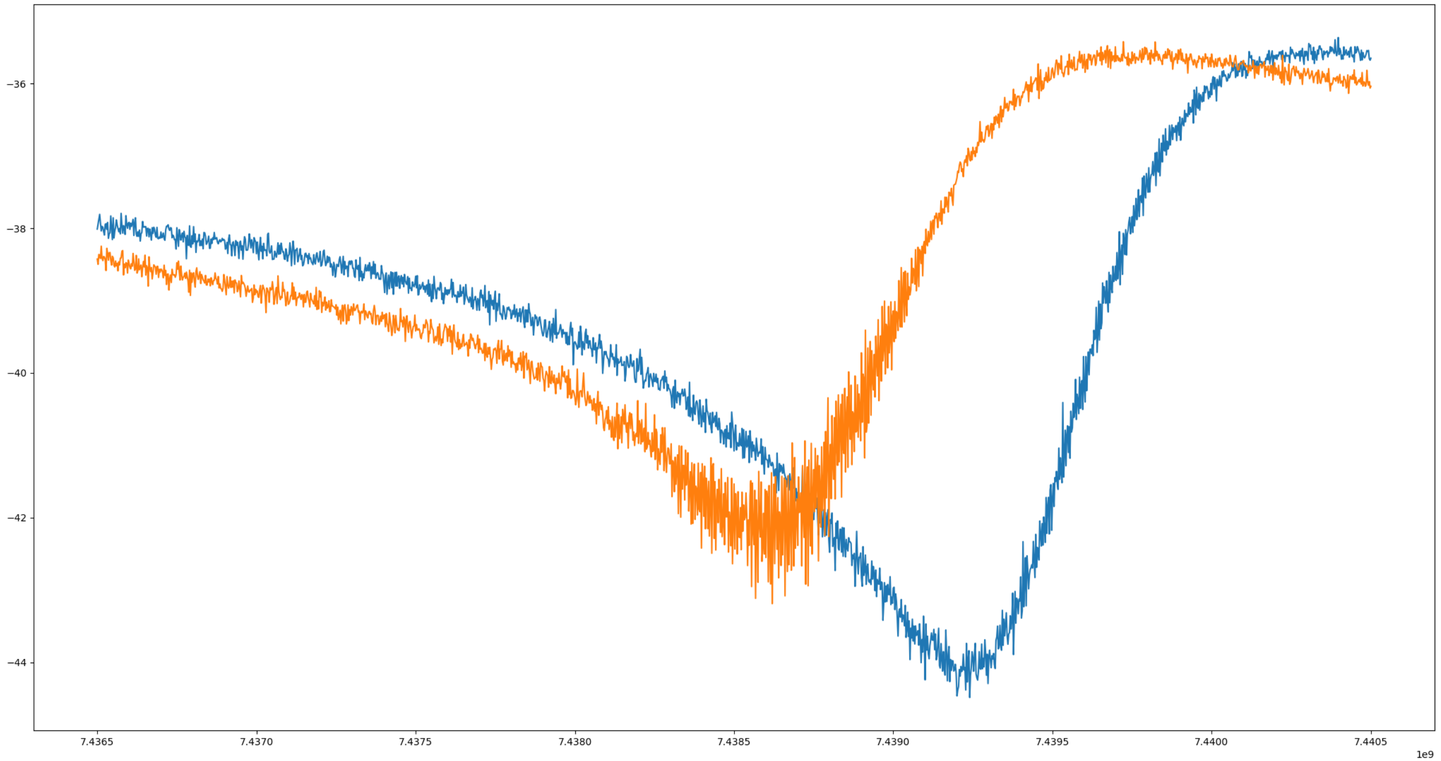
Qubit Readout





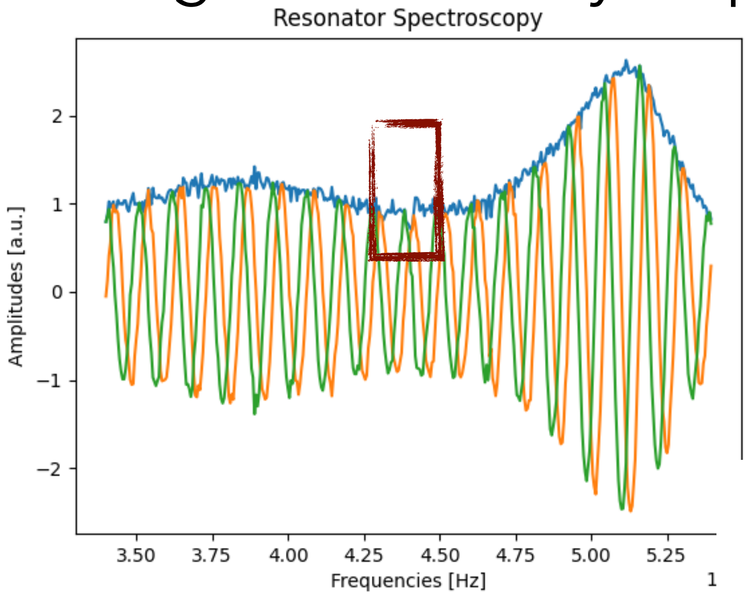
Cavity Resonance

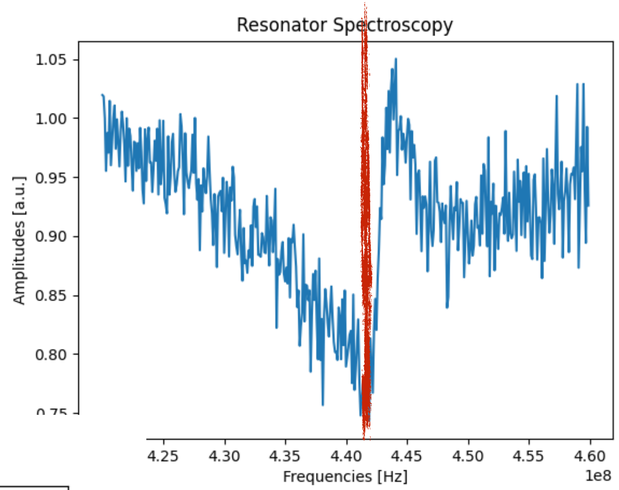
* Searching for the cavity + qubit |0> — Tunable frequency



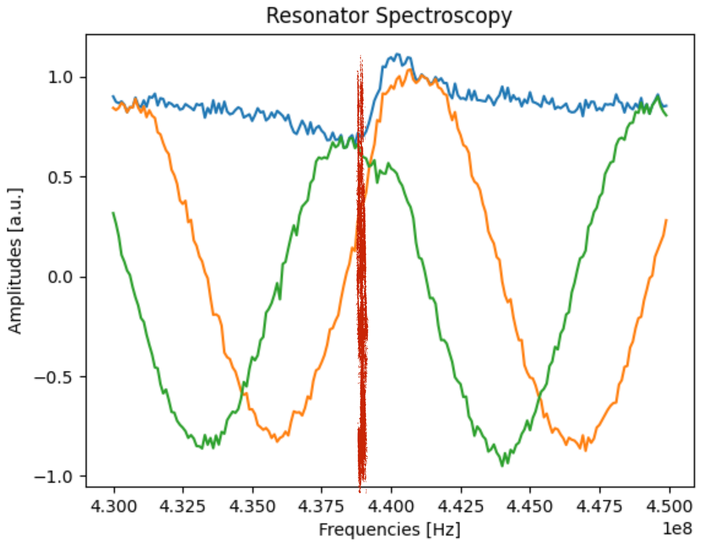
Cavity Resonance

* Searching for the cavity + qubit |0>



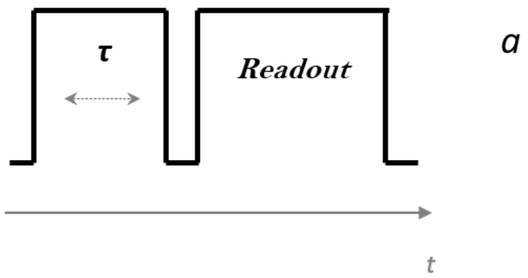


* Switching on the drive

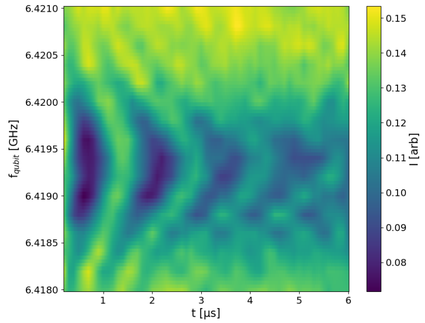


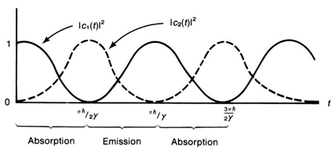
Qubit Characterization

Rabi oscillations

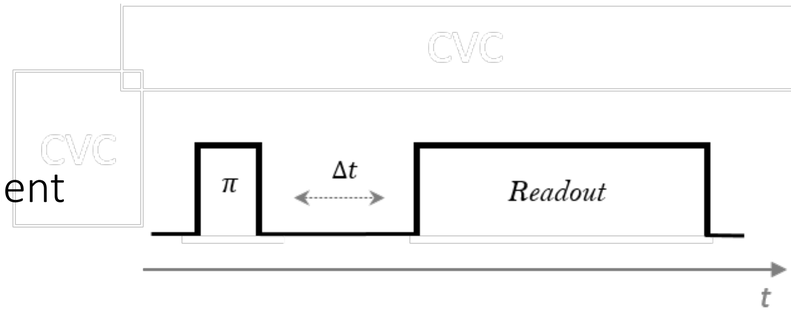


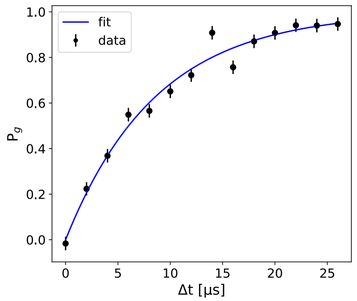
a



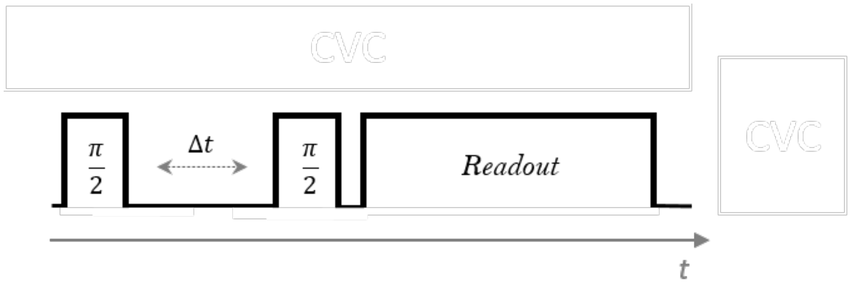


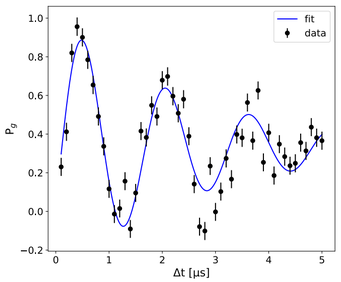
Qubit lifetime measurement





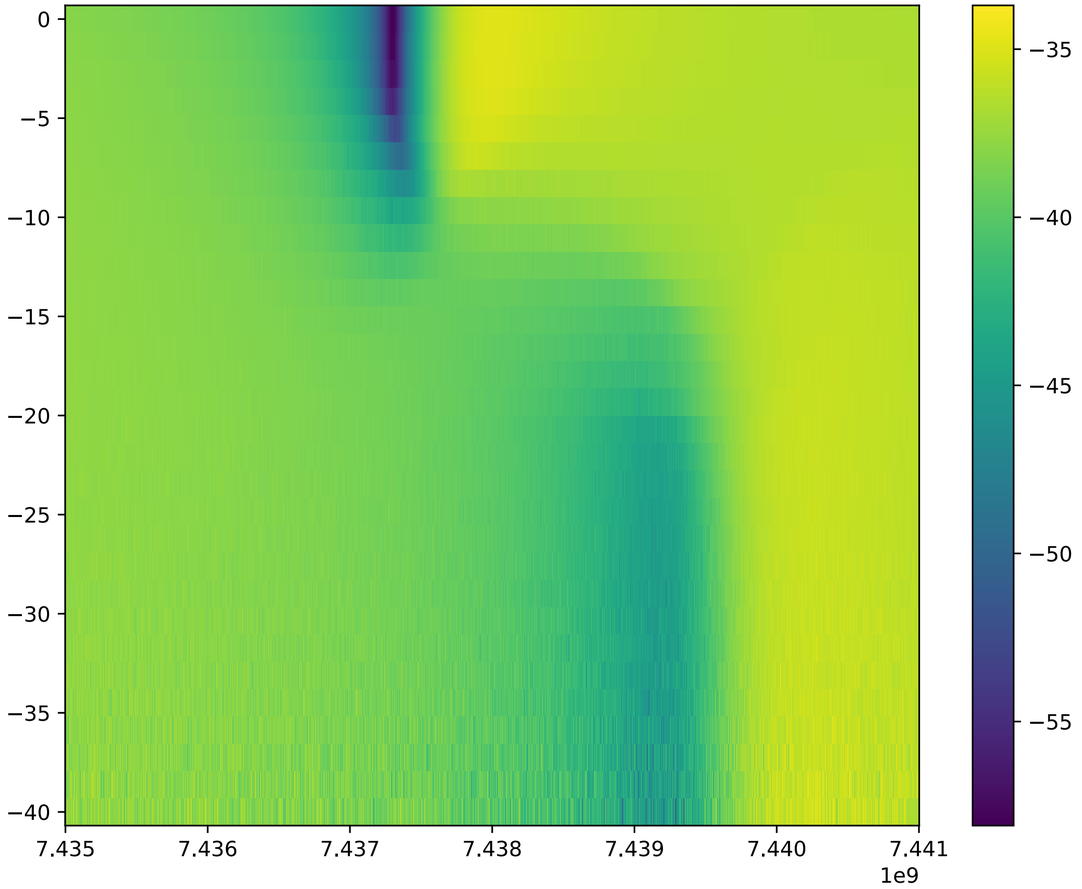
Ramsey Spectroscopy
and
T2 measurement





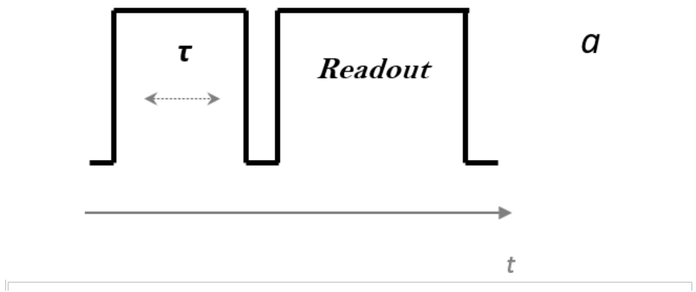
Spectroscopy

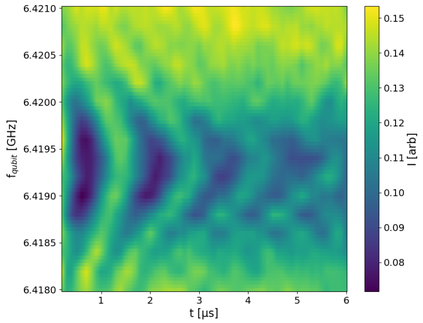
* 1 Tone Spectroscopy (Readout scan, different Drive w.p.s)

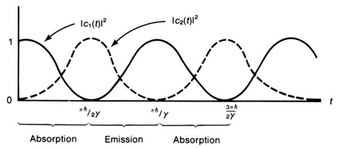


Rabi oscillation

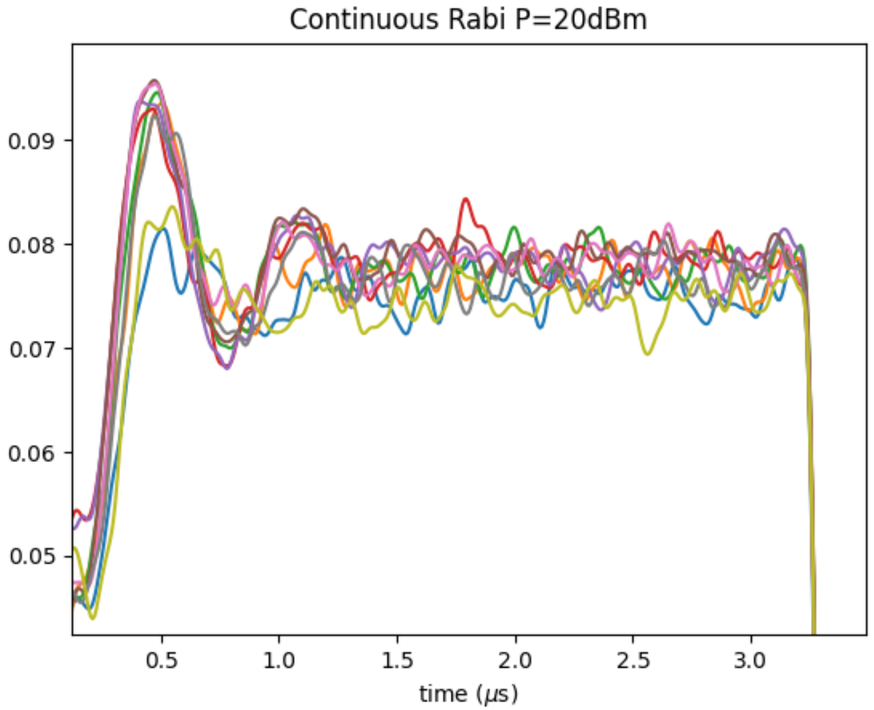
Rabi oscillations





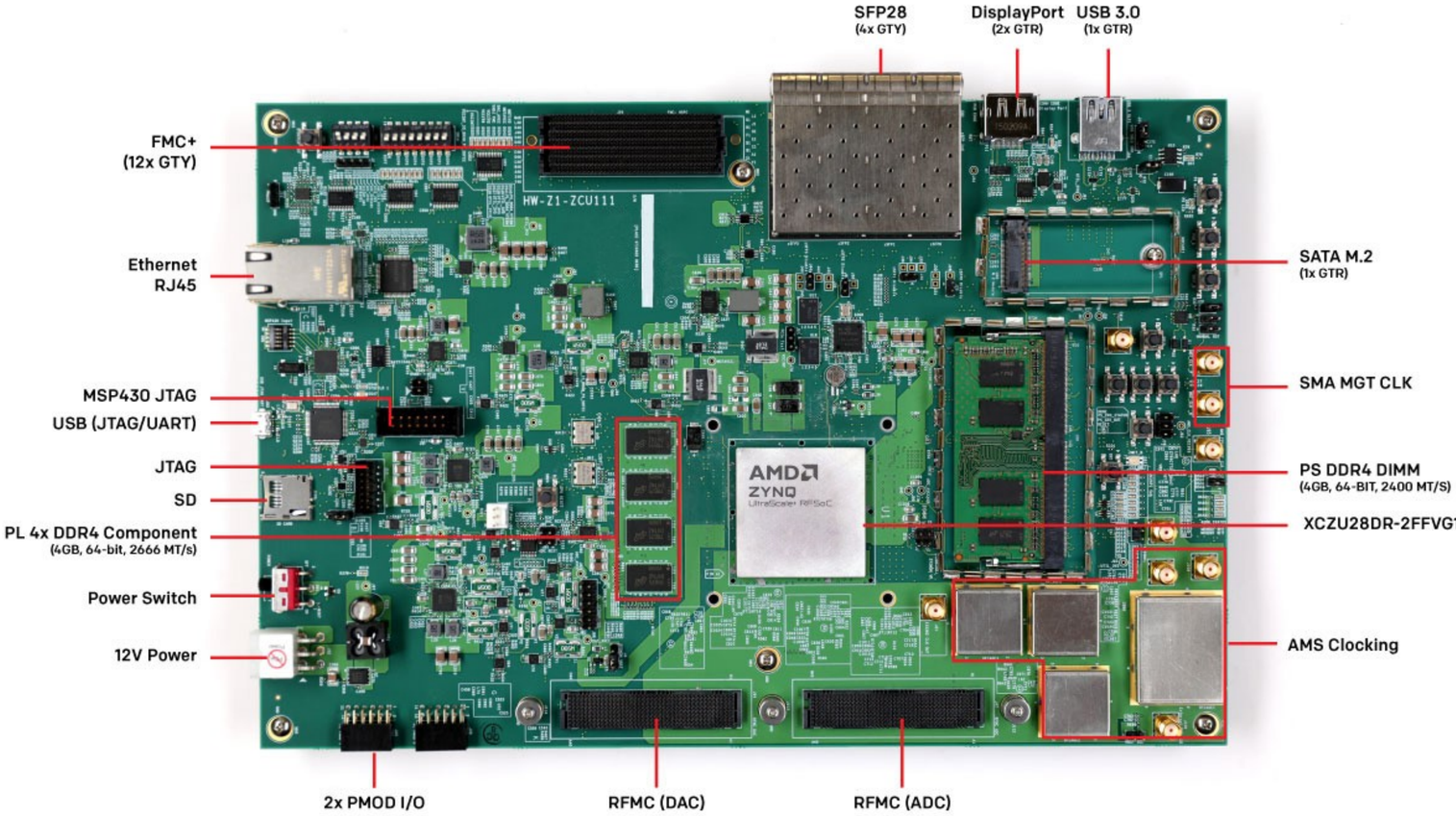


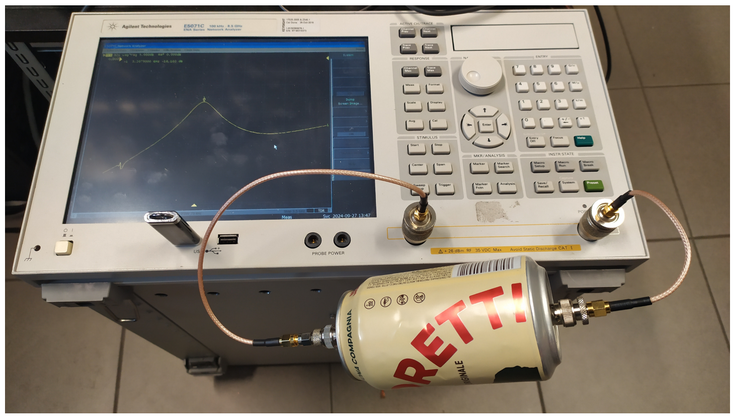
* Ongoing…

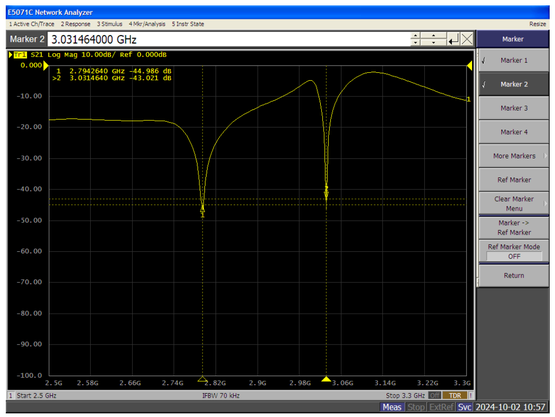


Local setup

* Xilinx ZCU-111 - generate and acquire RF signal
* Tested with oscilloscope and custom classic LC circuits
* Possible set up of a dedicated box for the board with connectors to access ADC and DACs (contacts with the Milano group)
* Possible to build dedicated and portable alimentation box







Conclusion

* We focus our effort on RFSoC board configuration, testing and usage for qubit calibration and experiments
  - Open to contribute to other tasks in QART&T.
* Foreseen to collaborate and assist Frascati team with DAQ and control.
* Possibile collaboration with Ferrara on firmware development
* First positive experience with single qbit data taking - in contact for next experiment
* Testing the board locally in Bologna.