HIGH PRECISION VOLTAGE SOURCE FOR QUENCH DETECTION EQUIPMENT TESTING

TE-MPE-EP

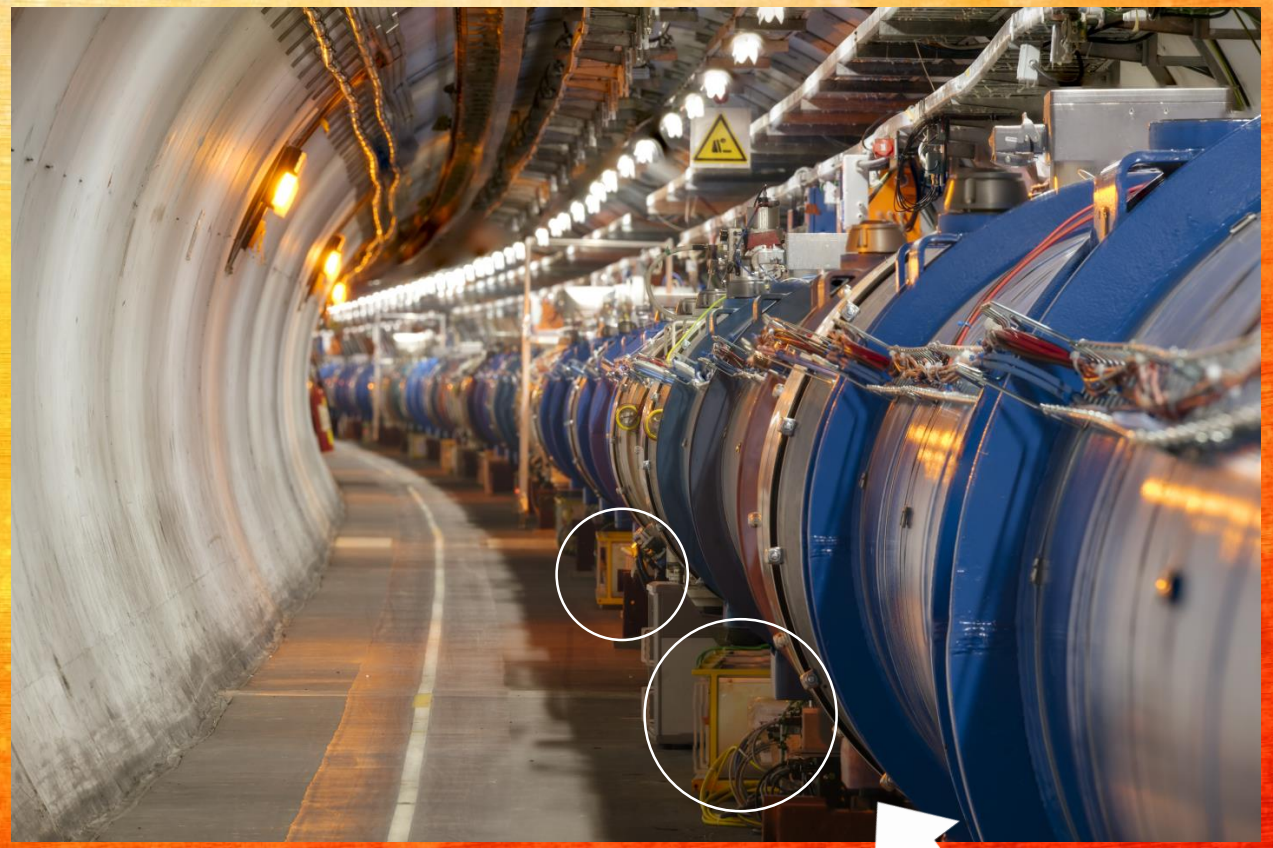
CERN GREEK HSSIP 2021

Hot-spots on superconductors

- Technological and constructive constrains ⇒ weakness, development of hot-spots under presence of high density currents

What is Quench?

- The complex phenomena around that hot-spots in the superconducting electro-magnets
- Loss of superconductivity



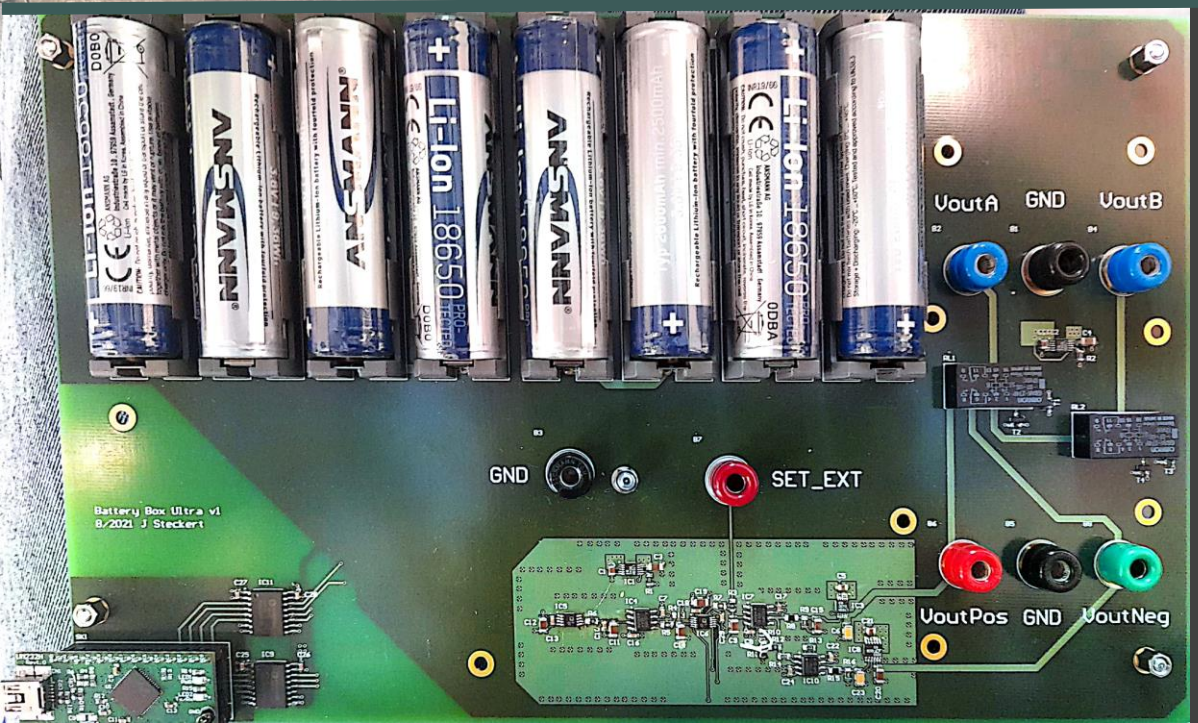
Quench Protection System (QPS)

the system which detects magnet quenching and protects them

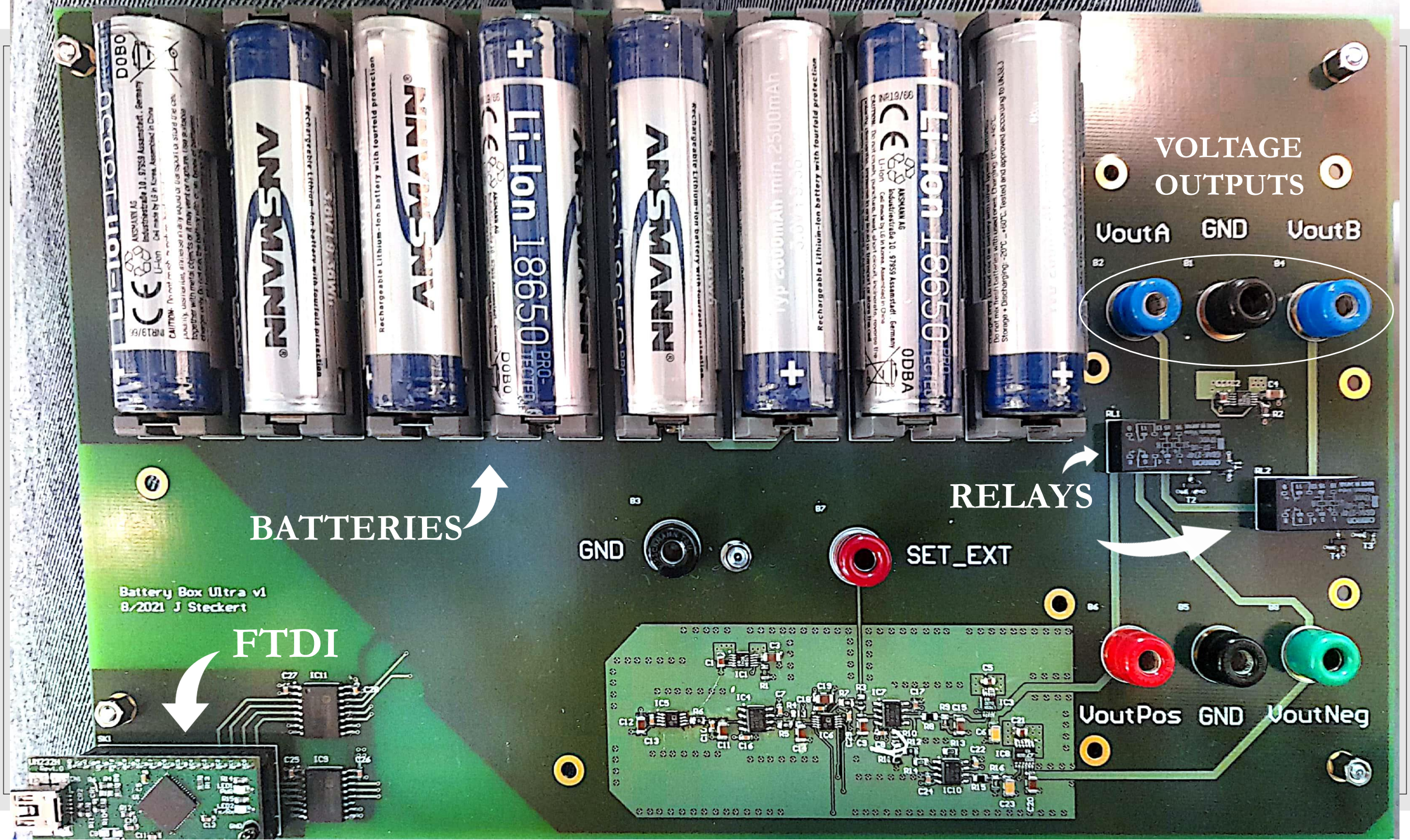
Batteries:
Discrete values of voltage outputs (1.5V step)

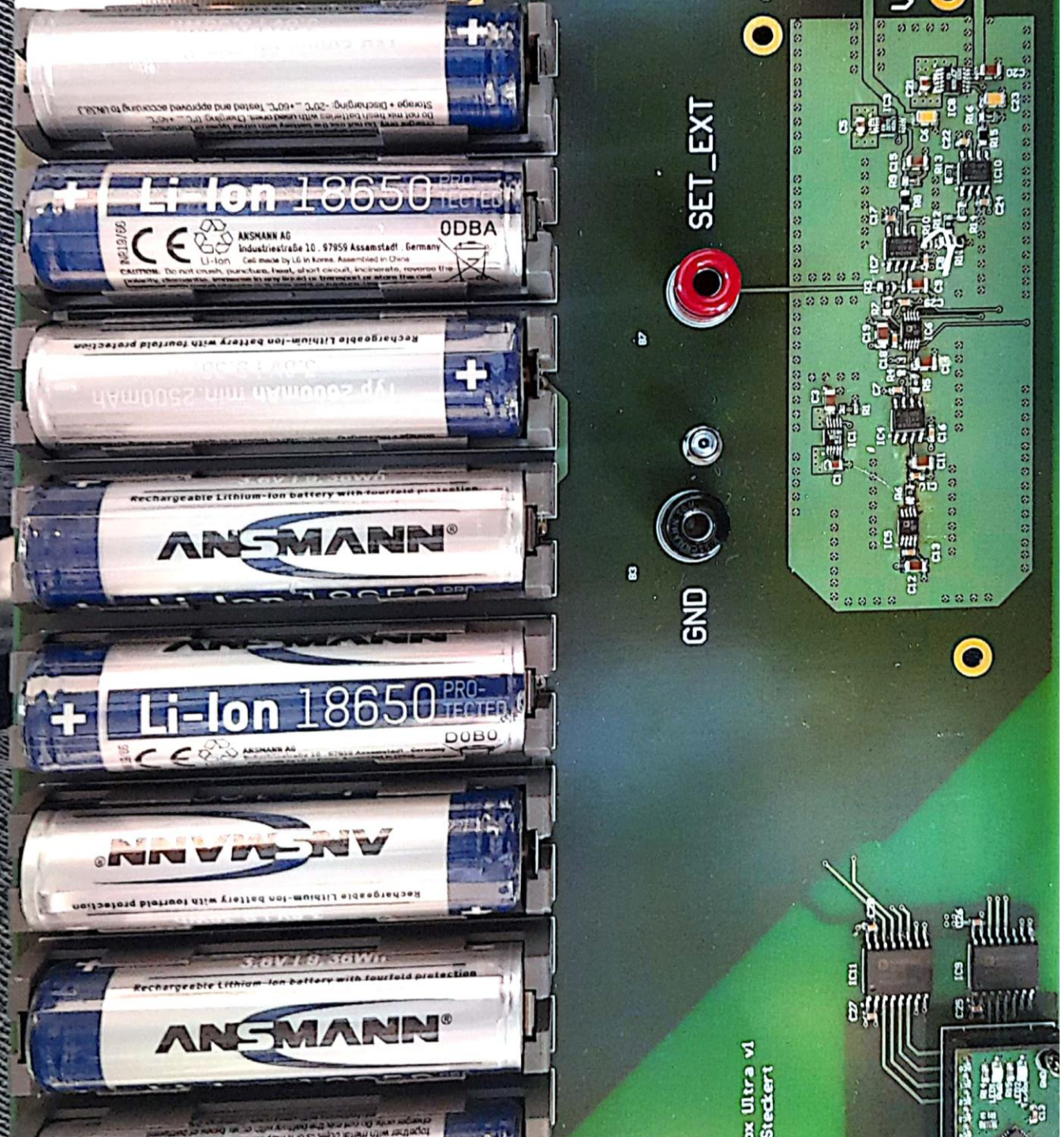
(OLD) BATTERY BOX

ULTRA BATTERY BOX



Additional components (DAC):
Any desirable value in the same voltage range


VOLTAGE
OUTPUTS
VoutA
GND
VoutB
RELAYS
BATTERIES
GND
SET_EXT
Battery Box Ultra v1
8/2021 J Steckert
FTDI
VoutPos
GND
VoutNeg

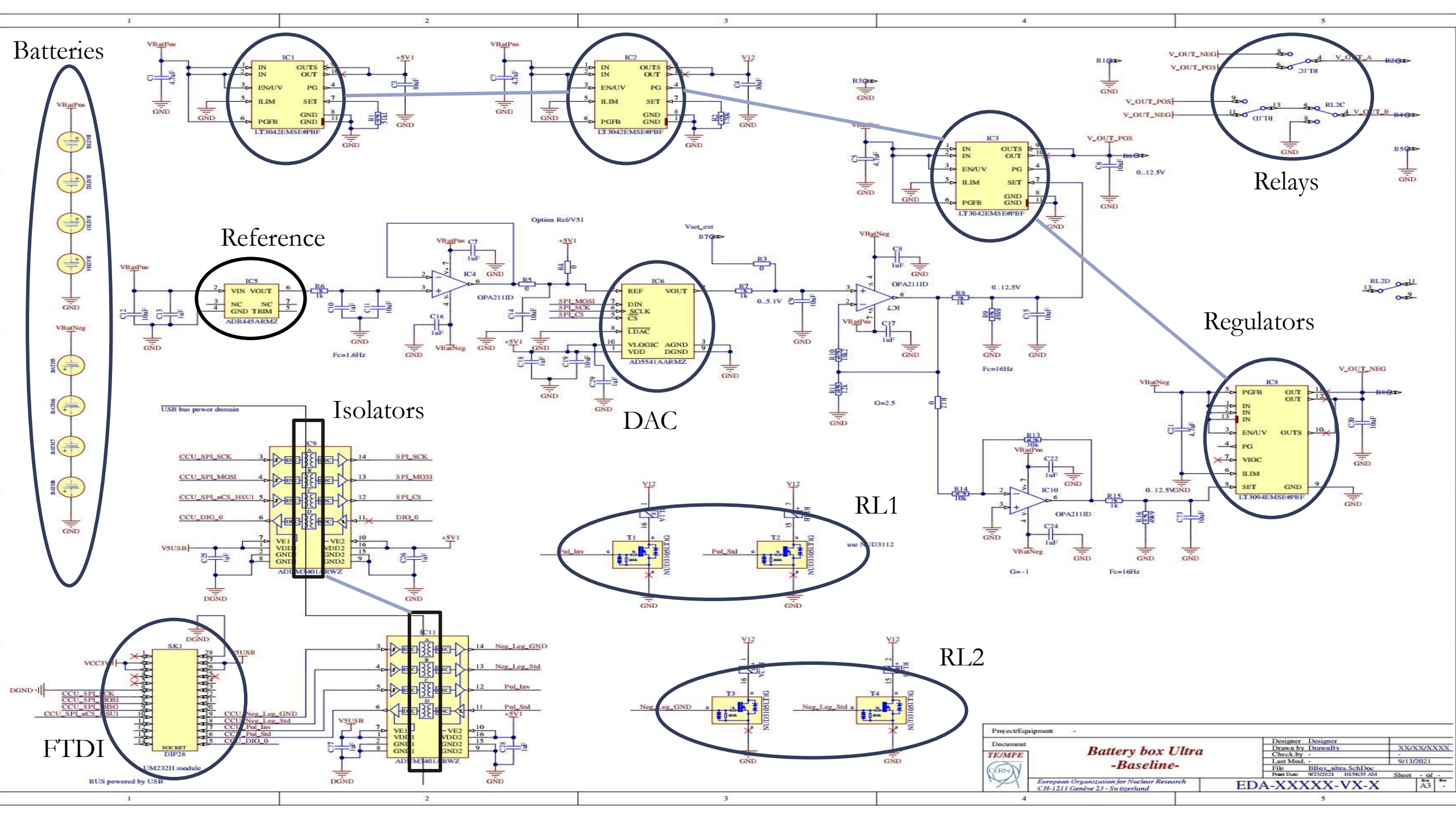


FTDI microchip: (Future Technology Devices International Ltd), FTDI microchip: It's used to add a USB serial port to a device

Batteries: supply of power

Relays: permit to user to swap the polarity on VoutA and VoutB, "switches"

Voltage Outputs: VoutA can be positive/negative, VoutB can be positive/negative/ground


Batteries
Reference
Isolators
DAC
Relays
Regulators
RL1
RL2
FTDI
Battery box Ultra
-Baseline-
EDA-XXXXX-VX-X

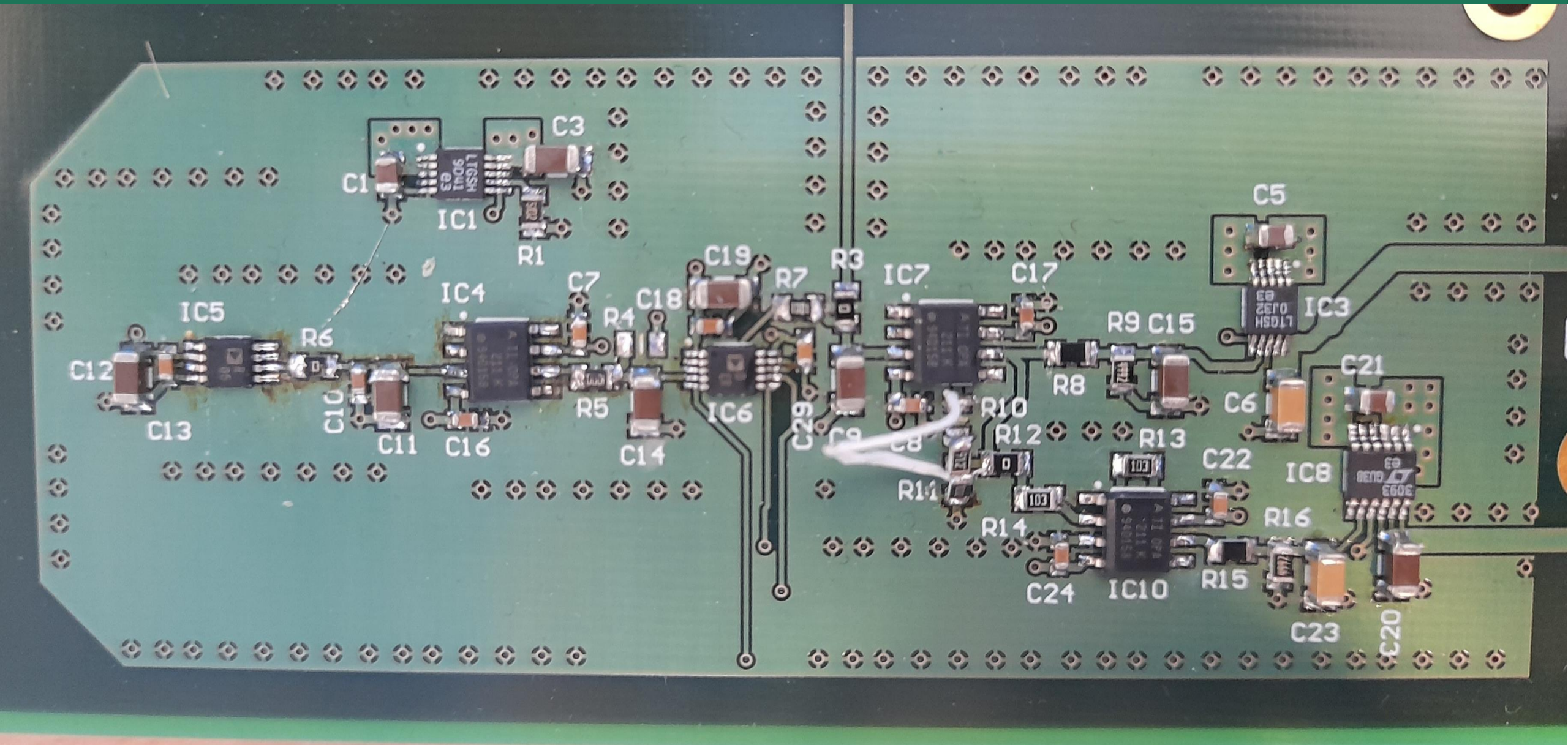
Reference: high-precision voltage reference for the DAC

DAC: Digital-to-Analog Converter

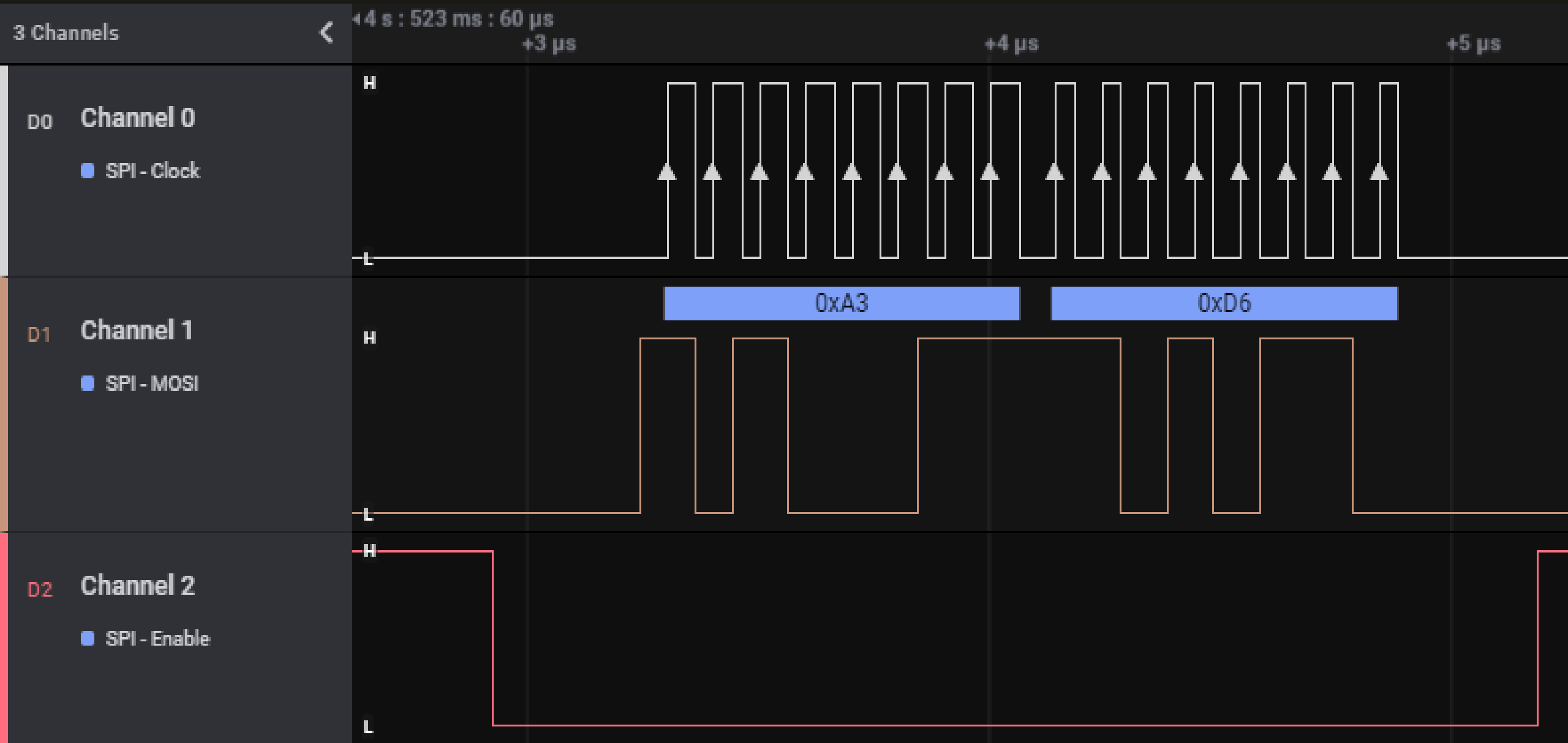
Regulators: voltage and current supply

Isolators: Separate the signals from unwanted interference.

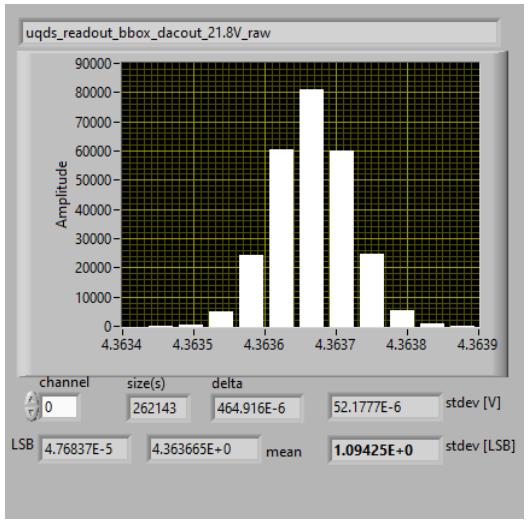
CIRCUIT

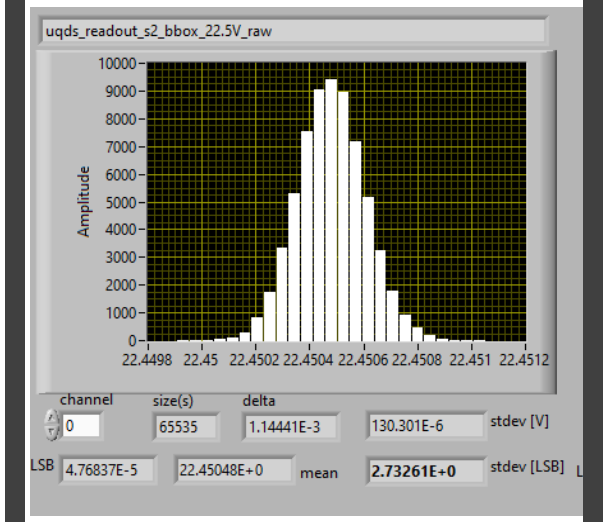

C1
C3
IC1
R1
C19
R7
R3
IC7
C17
C5
IC5
IC4
C7
C18
R4
R9
C15
IC3
R6
C12
R8
C21
C10
R5
IC6
C6
C13
C11
C16
C14
C9
R10
R12
R13
C22
IC8
R11
R14
R16
C24
IC10
R15
C23
C20

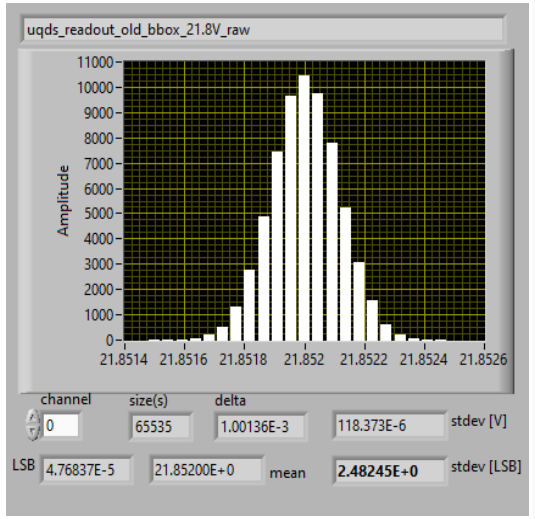
FTDI to DAC signals (SPI)

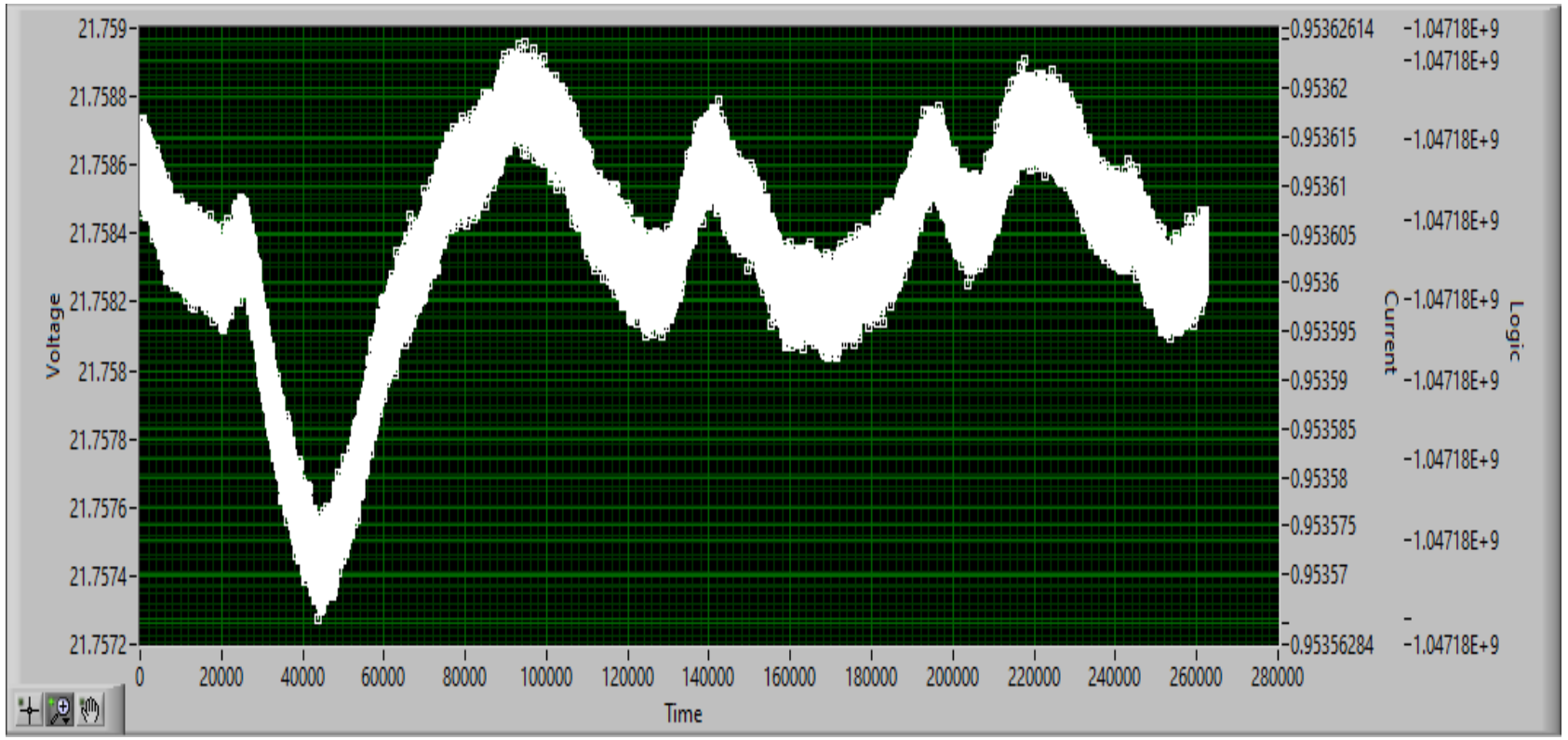


```python
# -*- coding: utf-8 -*-
"""
Created on Tue Sep 21 13:36:07 2021

Small class to control the Battery Box v3.
@authors: dvancea and tpridii

Requirements :
    instal libusb:
    on Windows through Zadig
    #https://eblot.github.io/pyftdi/installation.html#id1

    install pyftdi:
    # python pip3 install pyftdi OR python pip install pyftdi
    OR
    # git clone https://github.com/eblot/pyftdi.git
      cd pyftdi
      # note: 'pip3' may simply be 'pip' on some hosts
      pip3 install -r requirements.txt
      python3 setup.py install

The output voltage is defined by 16bits sent to DAC through SPI.
Two relays (2 coils and 2 switches per each) are controlled
through FT232H gpio ports: AC1, AC2, AC3, AC4, in order to define
polarity and balance on the output.
"""
import time
import pyftdi.spi as spi
from pyftdi.ftdi import Ftdi

SYSTEM = Ftdi.list_devices()

class BatteryBoxV2:
    """Class containing method to control the Battery Box v3.
    The user must input the serial number of the device after he connects it to the computer
    """
    def __init__(self, name, balance):
        self.name = name
        self.ctrl = spi.SpiController()
        self.device_handler = self.get_device_handler()
        self.balance = balance
        self.max_voltage = 25



    def get_device_handler(self):
        """ Establishes a connection to the device based on its name.
        """

        device_name = "ftdi://ftdi:232h:"+self.name+"/1"
        self.ctrl.configure(device_name)
        self.slave = self.ctrl.get_port(cs=0, freq=12E6, mode=0)
        self.gpio = self.ctrl.get_gpio()
        self.gpio.set_direction(0x1E00, 0x1E00)
        if self.name == SYSTEM[0][0][4]:
            return device_name
        else:
            print("Battery box not connected or wrong name given!")
            return None

    def close_connection(self):
        """ Immediately aborts all active tasks associated with a device, disconnects
        any routes, and returns the device to an initialized state. Aborting a task
        immediately terminates the currently active operation, such as a read or a
        write.
        """

        self.set_output_zero()
        self.ctrl.close()

    def set_output_zero(self):
        """ Sets the output of the battery box to 0V.
        """
        self.set_dc_voltage(0)

    def set_balance(self, selection):
        if selection>0 and self.balance=="sym":
            self.gpio.write(0x0A00)
            time.sleep(2)
            self.gpio.write(0x0000)
            print("Vout = VoutA - VoutB =", selection)
            selection = selection/2
        elif selection>0 and self.balance=="asym":
            self.gpio.write(0x1200)
            time.sleep(2)
            self.gpio.write(0x0000)
            print("Vout = VoutA =", selection)
        elif selection<0 and self.balance=="sym":
            self.gpio.write(0x0C00)
            time.sleep(2)
            #self.gpio.write(0x0000)
            print("Vout = VoutA - VoutB =", selection)
            selection = selection/2
        elif selection<0 and self.balance=="asym":
            self.gpio.write(0x1400)
            time.sleep(2)
            self.gpio.write(0x0000)
            print("Vout = VoutA =", selection)
        elif selection == 0:
            print("VoutA =", selection)
        else:
            print("Invalid balance value")
            return 0
        return selection

    def set_dc_voltage(self, selection):
        """ The user sets a voltage with the desired polarity.
        """

        if selection > self.max_voltage or selection<-self.max_voltage:
            print("Invalid Input! Voltage chosen not available! Available range +-25V")
            return 0

        if self.balance=='asym' and (abs(selection) > (self.max_voltage/2)):
            print("Invalid input for asymmertic balance! Maximum assymetric value +-12.5V!")
            return 0

        sel = self.set_balance(selection)
        vout = abs(sel)*65535/12.5

        h, l = divmod(vout,256)
        self.slave.write([int(h),int(l)])

        return 1
```





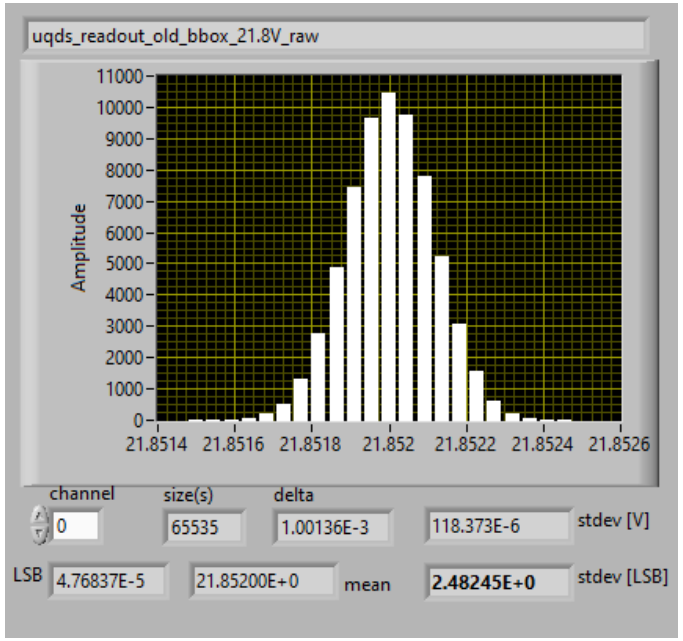


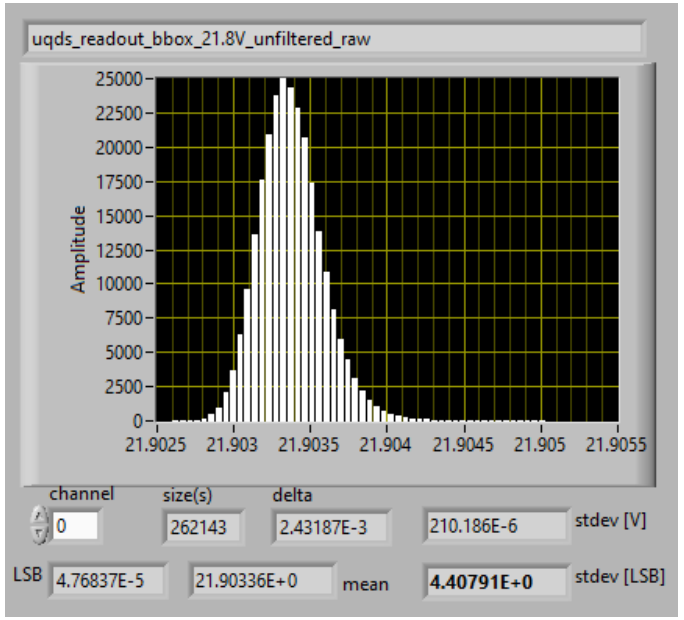


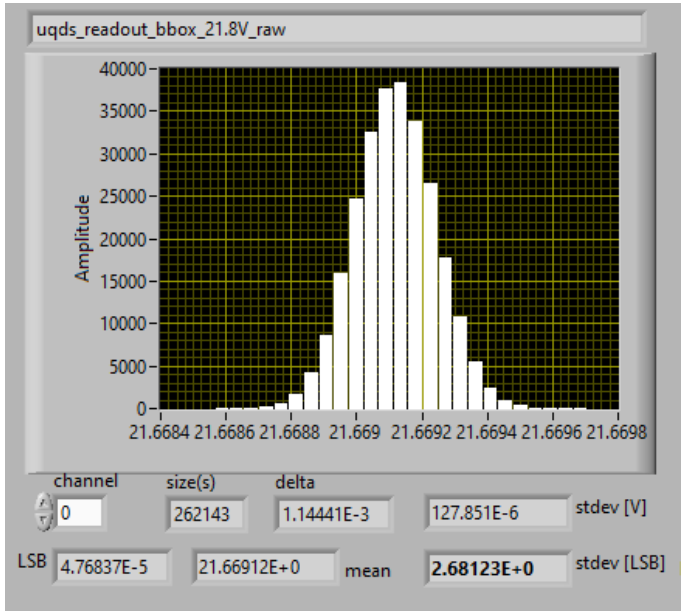
NOISE

THE FLUCTUATION OF VOLTAGE VALUE

NOISE COMPARISON







Thank you!

Maliaris Konstantinos
Stamatelou Stamata
Supervisor:
Tetiana Pridii