

# Data-acquisition with MDSplus and custom devices

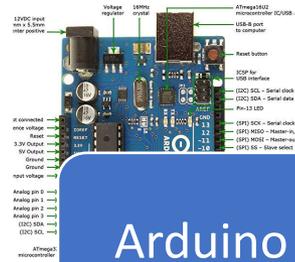
Fernando Santoro

Thomas Fredian, Stephen Lane-Walsh and Joshua Stillerman

PSFC at MIT

- Hardware (the experiment)
- Software (MDSplus Device class for data acquisition)
- Real-time visualization
- Data analysis
- Jupyter Lab

# Jupyter Lab (as the development environment)

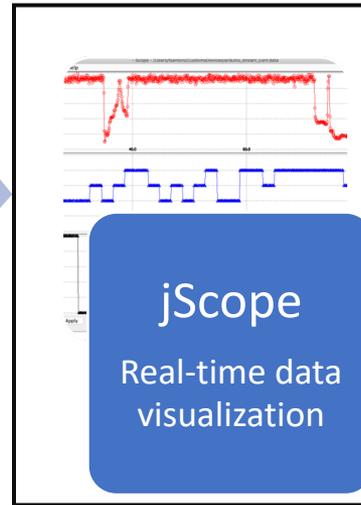


Arduino  
Lidar, T/H sensors

Custom Device  
(as the experiment)

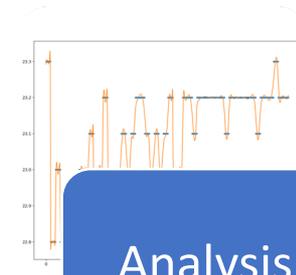


MDSplus  
device class



jScope  
Real-time data  
visualization

in a subprocess



Analysis  
Scipy, Matplotlib,  
etc

# Arduino device and sensors

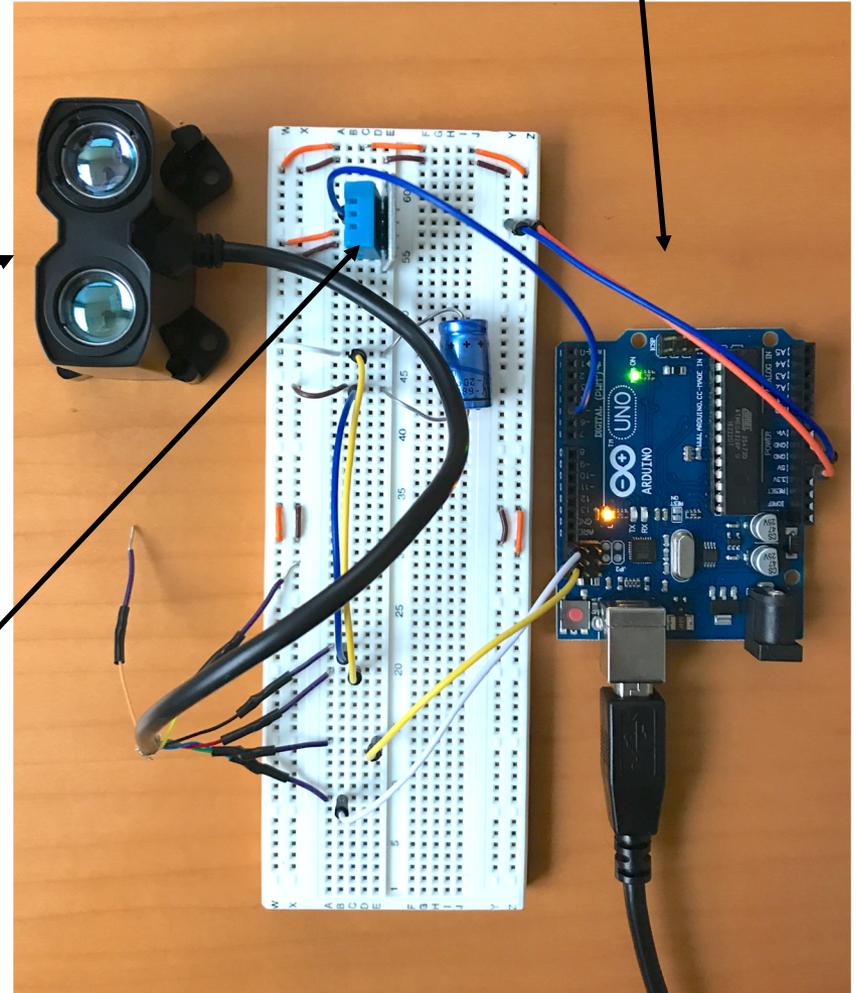
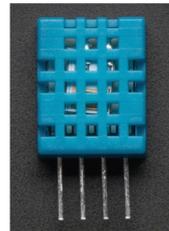
Arduino UNO  
microcontroller



Garmin Lidar v3HP



Adafruit Temperature  
Humidity sensor



# MDSplus and Jupyter Lab

The screenshot shows a Jupyter Lab environment with two main windows. The left window, titled 'MDSPLUSARDUINODEVICE.IPYNB', contains a Python script for defining an MDSplus device. The right window, titled 'MDSplusArduinoRun.ipynb', contains a notebook with instructions on creating a model tree and launching jScope.

```

In [ ]: import sys
        print(sys.version)

In [ ]: import MDSplus
        from MDSplus import Event
        import serial
        import time
        import datetime
        import threading
        import numpy as np
        import jdc
        import pdb

        class ARDUINO(MDSplus.Device):
            parts=[
                {'path':'COMMENT','type':'text','options':{'no_write_shot'}},
                {'path':'MODE','type':'text','value':'','options':{'no_write_shot'}},
                {'path':'BAUD','type':'numeric','value':9600,'options':{'no_write_shot'}},
                {'path':'SEG_LENGTH','type':'numeric','value':5,'options':{'no_write_shot'}},
                {'path':'MAX_SEGMENTS','type':'numeric','value':1000,'options':{'no_write_shot'}},
                {'path':'TREND_EVENT','type':'text','value':'ARDUINO_TREND','options':{'no_write_shot'}},
                {'path':'STREAM_EVENT','type':'text','value':'ARDUINO_STREAM','options':{'no_write_shot'}},
                {'path':'RUNNING','type':'any','options':{'no_write_model'}},
            ]

            parts.append({'path':'DISTANCE','type':'signal','options':{'no_write_model','write_once'}})
            parts.append({'path':'TEMPERATURE','type':'signal','options':{'no_write_model','write_once'}})
            parts.append({'path':'HUMIDITY','type':'signal','options':{'no_write_model','write_once'}})
    
```

**Creation of the MDSplus model tree**

Before executing the python device and start the data acquisition (as a systemd service or streaming data directly), the MDSplus tree needs to be defined. In other words, the model tree needs to be created. This is done by calling the Tree() method from the MDSplus library, and create a 'NEW' tree.

**Environment Variables**

```

In [1]: import os
        os.environ['arduino_path'] = '//Users//fsantoro//CustomDevices'
        os.environ['MDS_PYDEVICE_PATH'] = '//Users//fsantoro//CustomDevices'

!envy UDP_EVENTS

Out[1]: 'yes'
    
```

**Launching jScope as a subprocess**

This allows for the execution of the jScope application without blocking JupyterLab.

```

In [2]: import subprocess
        p = subprocess.Popen(['jScope', '/Users/fsantoro/CustomDevices/arduino_stream_conf.data'], stdout=subprocess.PIPE)
    
```

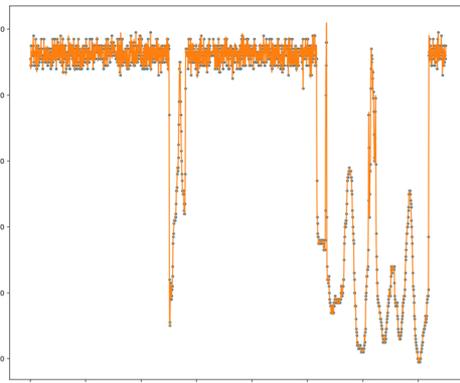
**Creation of an MDSplus model tree for this experiment**

```

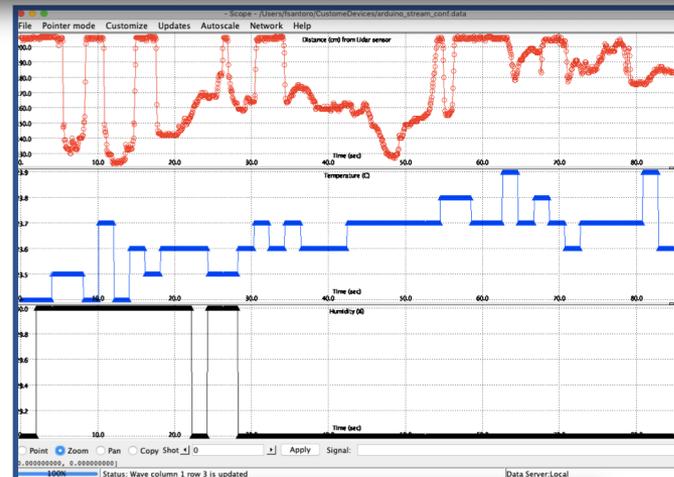
In [3]: from MDSplus import Tree
        import mdsarduino

        tree = Tree('arduino', -1, 'new')
        tree.addDevice('arduino_node', 'arduino')
        tree.write()
        tree.close()
    
```

Data  
Analysis



matplotlib and scipy



jScope

Real-time  
visualization