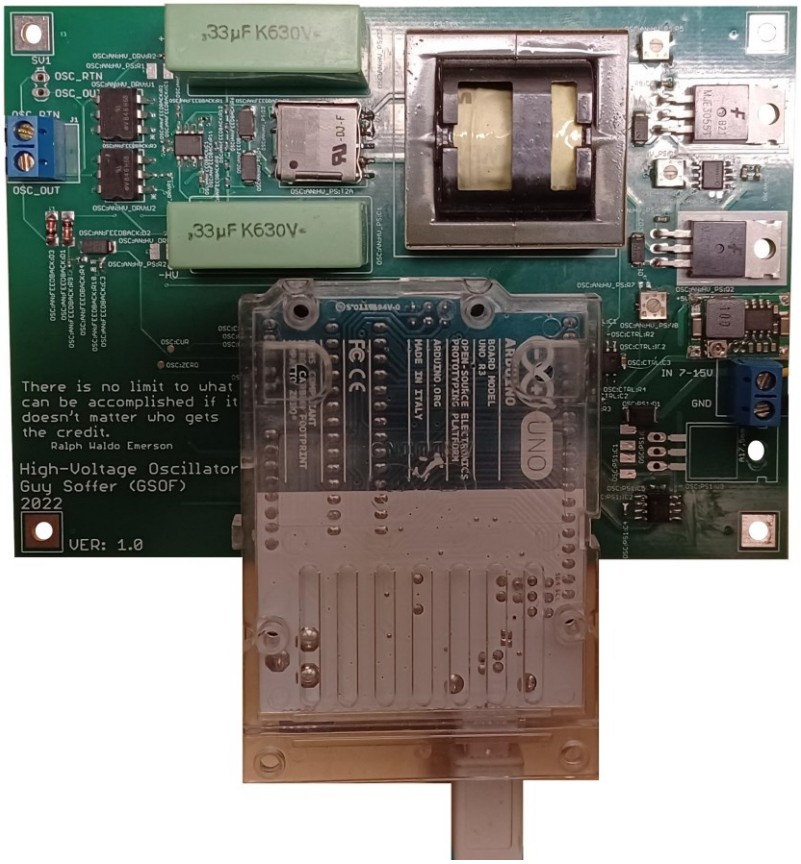
User manual for
Programable-High-Voltage Oscillator



Revision 0.3

Acknowledgements

Special thanks to James Perry for providing electrical requirements and assisting with beta testing.

Introduction

The Programable High Voltage Oscillator (PHVO) is a programmable square wave oscillator capable of output voltages between 20V to 300V (peak-to-peak) and 1Khz to 15Khz output frequency. The HVO is a hardware expansion board for the Arduino Uno R3. The Arduino controller is running a dedicated firmware that regulates the output voltage, generates the square wave output, and communicates with the PC using the GSOF_ArduBridge protocol stack (over USB).

A short video demonstrating the HVO can be watch in the link below:
https://www.youtube.com/watch?v=4fjpdIp5VfY

Top-level overview:

The chapter will briefly overview the hardware and software capabilities.

High Voltage Oscillator hardware

The hardware includes:

- Reverse polarity protection.
- Internal voltage regulators.
- Bipolar, high voltage step up power supply.
- Bipolar high voltage output stage.
- Output current measurement circuit.
- Output voltage monitoring

A unique capability is the ability to measure the average output current. This enables the user to estimate the impedance of the connected load. In Digital MicroFluidics (DMF) the impedance is measured between any of the DMF electrodes and the Indium-Tin-Oxide (ITO) plate. This is extremely useful for detecting faults in droplets and estimating droplet volume. Figure **1** shows the top-level architecture of the HVO.

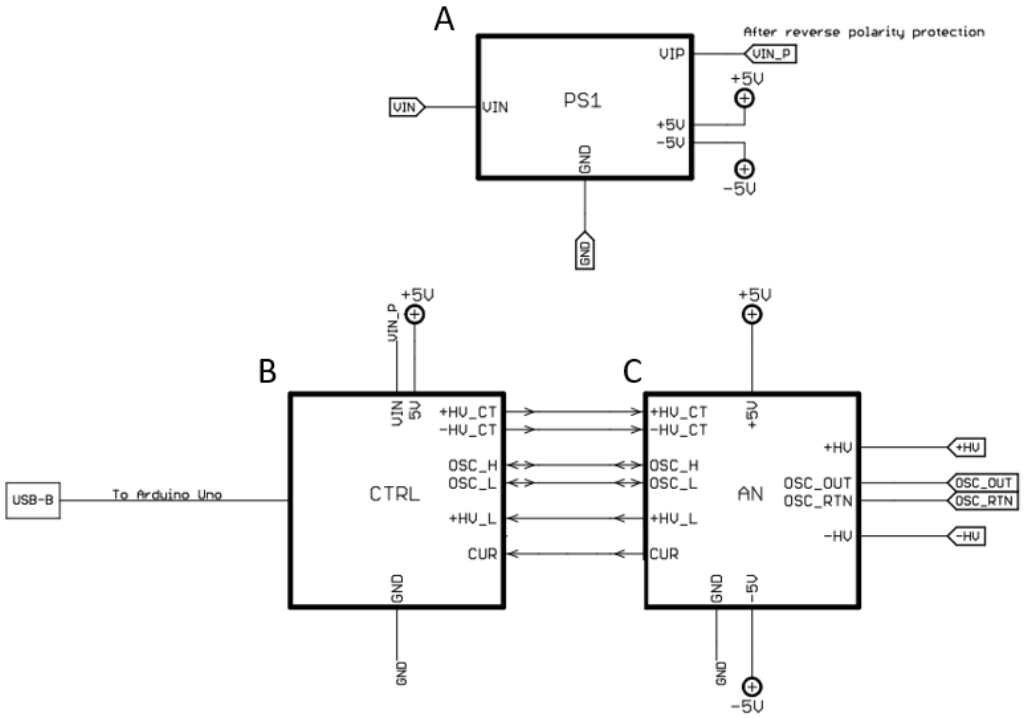


Figure 1 - Top-level overview of the High-Voltage Oscillator design. The HVO includes (**A**) +/- 5V voltage regulator that feeds the Arduino Uno controller (**B**), and the oscillator analog section (**C**). The analog section consists of the programmable, bipolar high voltage step up circuitry, the output stage and output current measurement circuit. The arduino controls the output voltage and output stage to generate the periodic square wave. The Arduino connects to the PC via USB-B connector. The connection enables the user to configure the desired output signal by means of frequency and peap-to-peak voltage.

Python class

The HVO class has four (4) public methods.
The init() method accepts the port and baud rate used for communication.
The setFreq_Khz(int khz) sets the output frequency in Khz resolution (0 to 30).
The setVoltage(int volt) accepts integer value between 0 to 255.
The getVoltage(), returns the peak-to-peak output voltage in Volts.
The main includes the ***osci*** global object to access the HVO hardware (**Figure 2**).

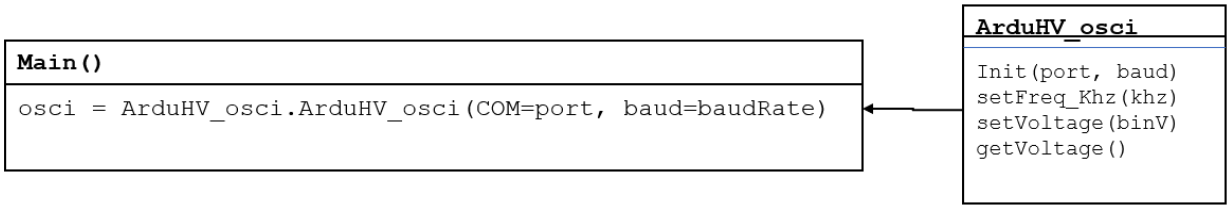


Figure **2** - Global ArduBridge system object

NOTE:	The setVoltage() method accepts unitless integer values between 0 to 255. 0 is the lowest and 255 is the highest possible output voltages. The current firmware does not include a close loop output voltage regulator. Hence, the output voltage will vary under different loads and output frequencies. It is up to the user to monitor the output voltage, using the getVoltage() method (or external equipment) and increase or decrease it using the setVoltage() method.

Installation and hardware setup

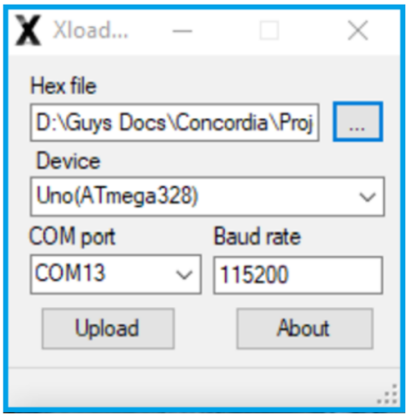
The HVO software has two parts, the firmware that runs on the Arduino and the Python module that runs on the PC. Each part should be installed independently. The Firmware installation procedure is usually called programming or uploading.

Firmware upload

To program the Arduino with the compiled firmware (FW), use the xloader application in the ArduBridge folder and follow the steps below:

1. Open the xloader application (Figure 4A).
2. Select the COM port that the Arduino Uno R3 is connected to.
3. Select the Hex file "HVO_V01.hex" (or the most updated FW you have) (Figure 4B).
4. Press the "Upload" button and wait for the programming operation to finish after a few seconds.

A



B

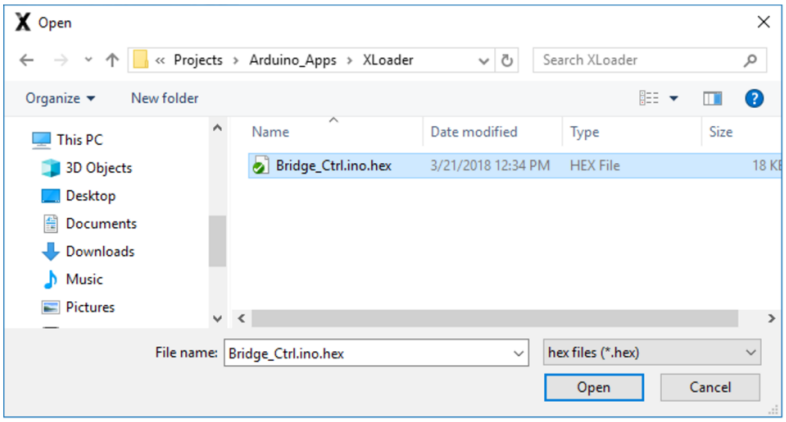


Figure **4** - Xloader application

Python modules installation

The *ArduHV_osci* module depends on the ArduBridge. First make sure you have Python 3.1 (or above) installed together with the latest *pyserial* module. Installation of the *ArduBridge* can be done in two ways:

1. By double clicking the setup.bat batch file (Figure 5A).
2. By running the setup.py script in a command prompt ("CMD") window (Figure 5B).

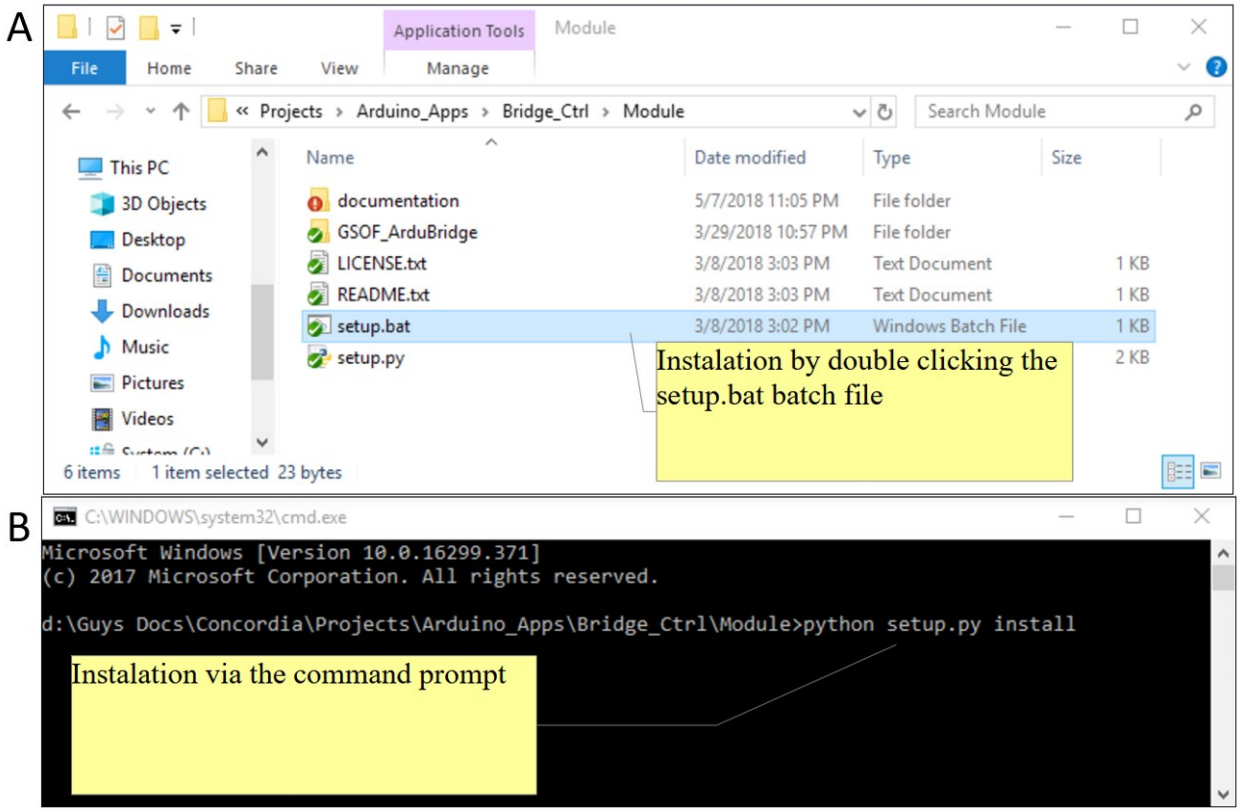


Figure **5** - Two options to install the ArduBridge Python module.

Connection

The HVO operates from a single 7.0V to 12V / 1A power supply. The Arduino board is driven by its USB power.

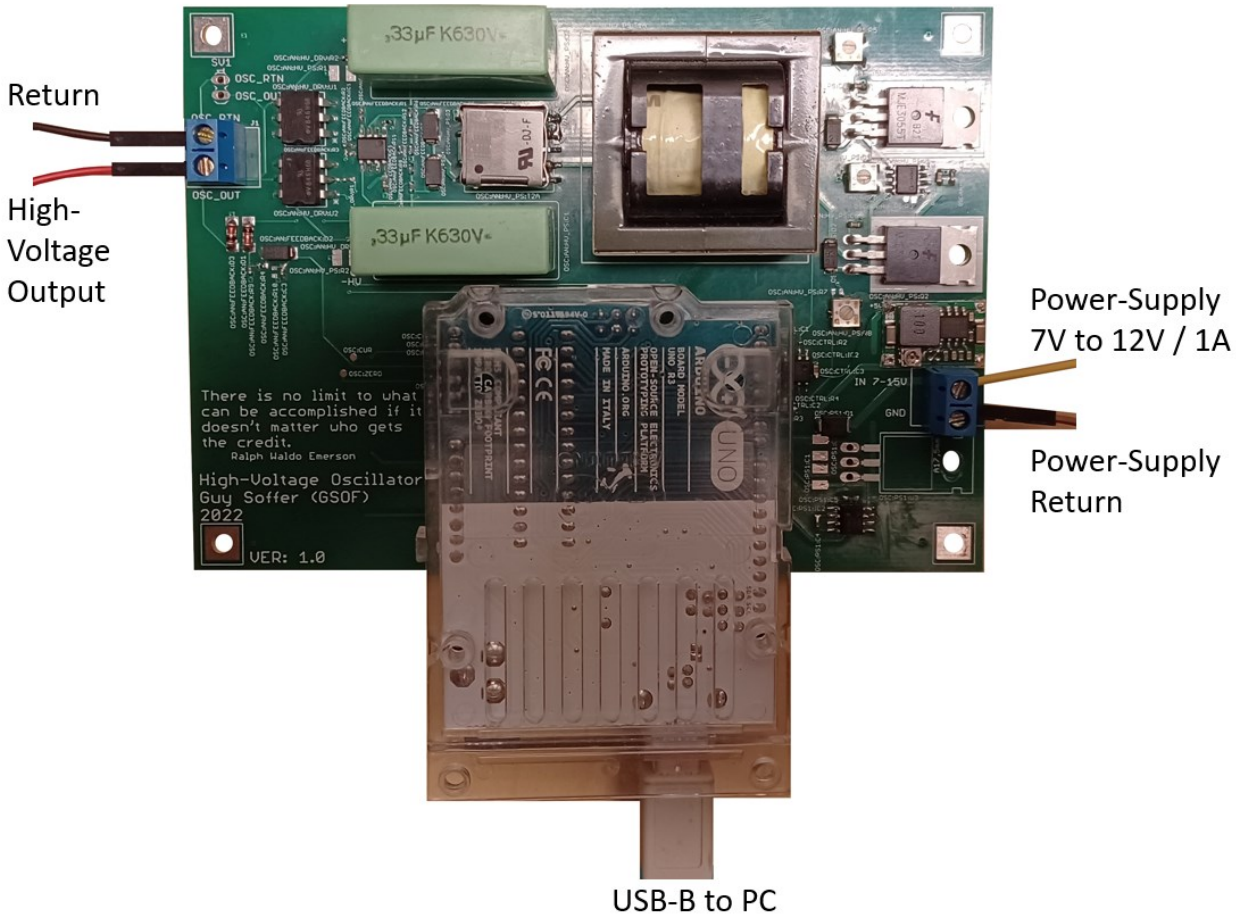


Figure **6** - Example of ArduShield power supply configurations

Expansion connector and stack support

The ArduShield can be connected to any expansion board that uses I2C communication. The expansion connector includes supply voltage, ground (+5V and +5_RTN), and special +V and +V_RTN signals for additional power (from the HV input connector). The ITO_RTN pin is dedicated for impedance measurement for DMF setup. The SDA and SCL are the standard I^2C bus signals. Figure 7A shows a stack up configuration in which the ArduShield is connected to two extension boards using only the expansion connector SV2 (Figure 7B). This eliminates cables and reduces the overall size of the setup.

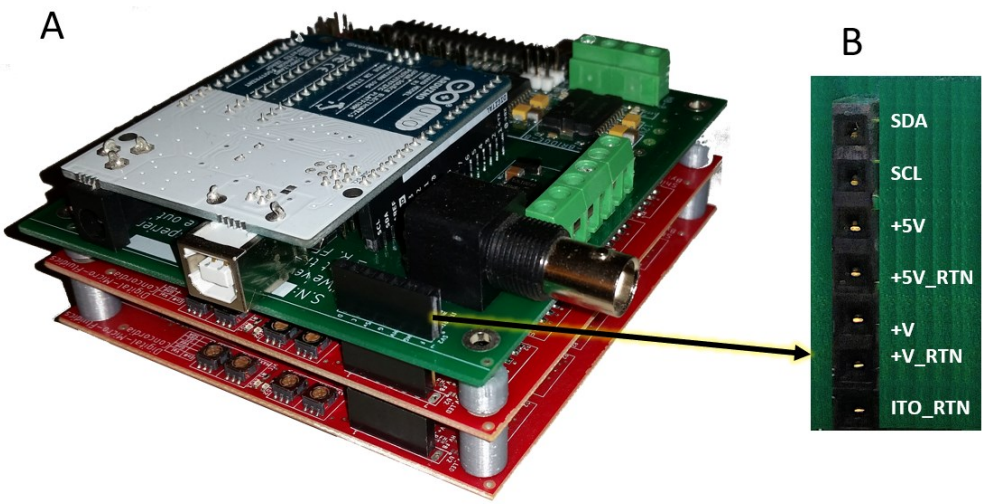


Figure **7** - Example of ArduShield and expansion boards in stack configuration
(**A**) Stack of three boards, bottom two (red) are DMF drivers and on top (green) is the ArduShield.
(**B**) The pin arrangement and signal names of the expansion connector.

NOTE:	The SDA and SCL signals are also on the GPIO header. The SDA and SCL signals are connected to AN4 and AN5 (in the Arduino Uno). Remove any filtering capacitors and disconnect any loads from AN4,5.

Main configuration script

This chapter shows how to build and initialize the PHVO.

Initializing the Programable High Voltage Oscillator

Configuring the PHVO is simple and mostly automatic. Only the COM or DEV parameter should be manually edited to match the assigned value by the operating system (OS). A template code that builds the global '`osci`' object and starts communication is included (Figure **8**). It can serve as a starting point for any project. To complete the project, the user should add relevant imports, update the '`port`' variable and add his code at the bottom.

```
#!/usr/bin/env python
"""
Script to build an High Voltage Oscillator object and configure it.
To customize the environment, you will need to change the parameters
in the "PARAMETER BLOCK" in the __main__ section

By: Guy Soffer
Date: 21/Sep/2022
"""

#Basic modules to load
import time
import ArduHV_osci as OSCI

def close():
    ardu.OpenClosePort(0)
    print('COM port is closed')

if __name__ == "__main__":
    #\/\/\/ CHANGE THESE PARAMETERS \/\/\/
    port = "COM3"          #<--Change to the correct COM-Port to access the Arduino
    baudRate = 115200*2  #<--Leave as is
    #/\/\/\   PARAMETERS BLOCK END  /\/\/\

    print('Using port %s at %d'%(port, baudRate))
    osci = OSCI.ArduHV_Osci( COM=port, baud=baudRate )

    print('Discovering the High Voltage Oscilator on port %s'%(port))
    if osci.OpenClosePort(1):
        print('HVO is ON-LINE.')
    else:
        print('HVO is not responding.')

    osci.cfg()

    print("To set the output frequency to 5 Khz use")
    print("osci.setFreq_Kz(5)")
    print("To change the output voltage use")
    print("osci.setFreq_Kz(b) #< where b is integer between 0 to 255")
    print("To read the output voltage use")
    print("osci.getVoltage()")
```

Change the '***port***' to the assigned 'COM" (under windows) or '/dev/' (under Linux)

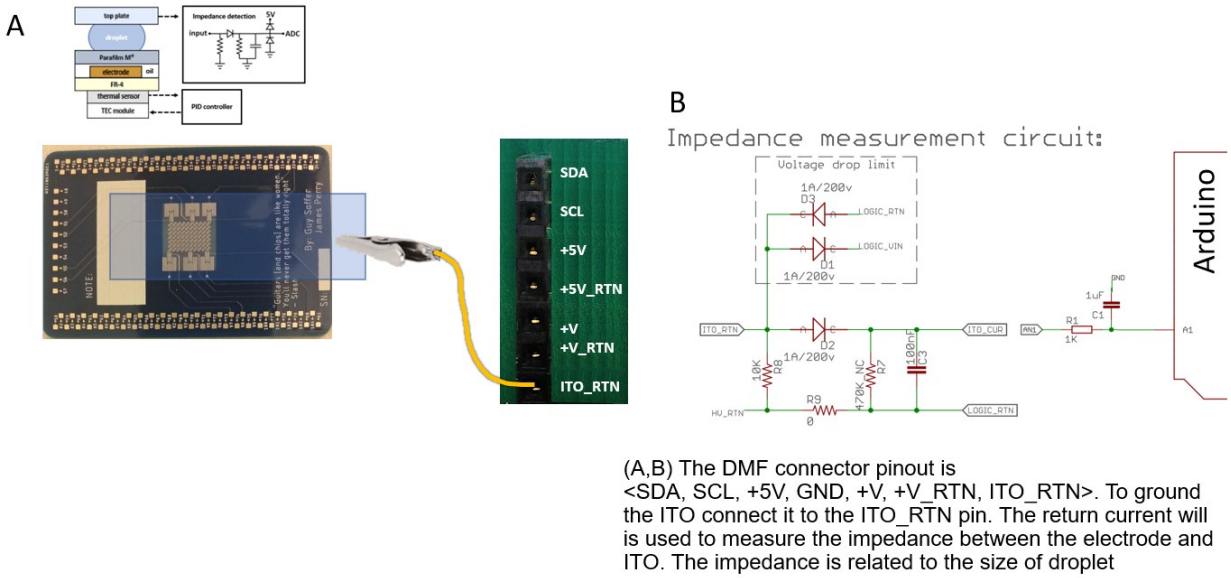
Build the HVO object under the name '***osci***'

Prints "Quick help"

Figure **8** - Example of the HVO main script that initializes the HVO and enables the user to control it.

Digital Microfluidics (DMF) impedance measurement circuit

TBD



(A,B) The DMF connector pinout is <SDA, SCL, +5V, GND, +V, +V_RTN, ITO_RTN>. To ground the ITO connect it to the ITO_RTN pin. The return current will is used to measure the impedance between the electrode and ITO. The impedance is related to the size of droplet

Figure 9 - The ArduShield DMF impedance measurement circuit and connectors

Typical setup with DMF electrode driver board

The typical setup shown in figure-xx and the DMF setup shown in figure-xx can be combined and further expanded with additional sensors and cooling fans (not shown).

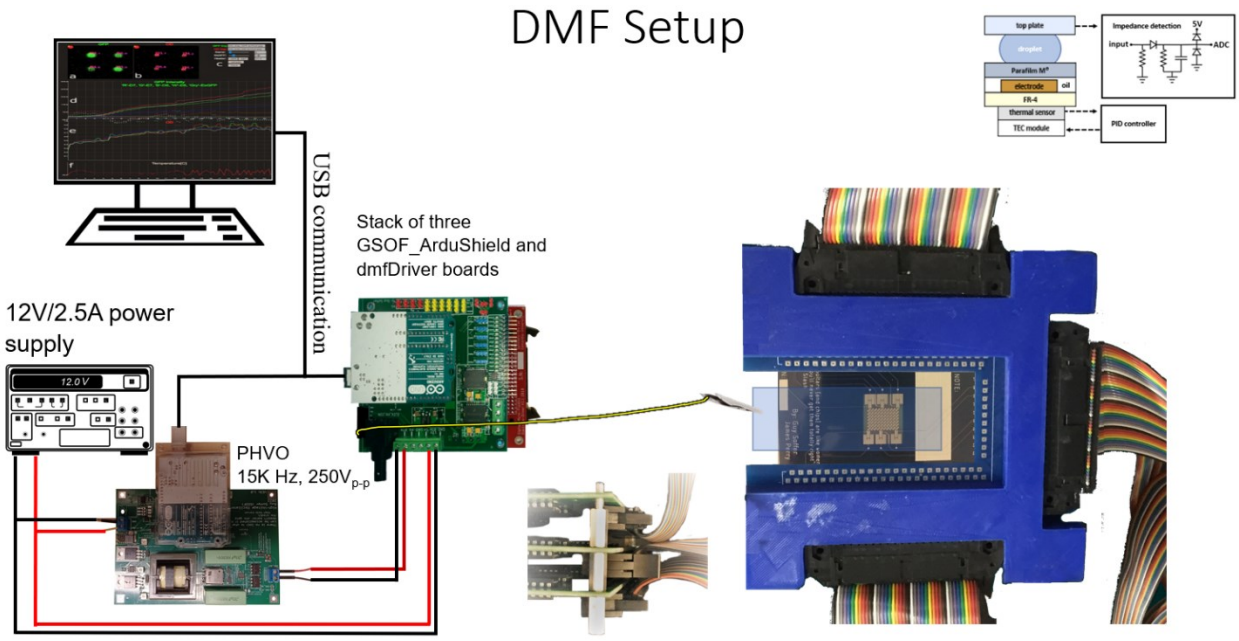


Figure 10 - Example of typical DMF setup using the ArduShield

NOTE:	The ITO ground (ITO_RTN) is connected to the logic ground via the impedance measurement circuit.