DuoMAG™ TMS

MAGNETIC | STIMULATION

Python Toolbox Manual

Document Version

Version 2.1 | 21 June 2025

Contact

https://brainbox-neuro.com/ support@brainbox-neuro.com

Contents

1. Introduction

The DuoMAG Python Toolbox is a Python class designed to control the DuoMAG XT-100 and DuoMAG MP devices.

These methods allow you to:
- Set the stimulator intensity.
- Trigger pulses.
- Read the status of the DuoMAG device.
- Adjust the recharge delay after a pulse.
- Adjust the TTL OUT delay after a pulse.

The toolbox also contains a number of demo scripts as guidance. Additional commands and functionalities may be provided with the toolbox.

2. Setup & Requirements

The DuoMAG device must be properly set up before attempting to communicate with it. Ensure that all necessary physical connections (such as the serial cable to the COM port) are securely in place and that the device is powered on.

For all Python methods to work properly, the following library is required:
- pySerial: This is used for serial communication between the DuoMAG device and your computer. You can install it using the following command:

  ```
  pip install pyserial
  ```

Ensure that Python version 3.7 (or later) is installed on your system for compatibility. You can download Python from the official website: https://www.python.org/downloads/.
Device Communication

Once the setup is complete, you can use the pySerial library to send and receive commands from the DuoMAG device over a serial connection. For detailed instructions on communication protocols and commands, refer to the DuoMAG user manual.

3. Installing Deymed Drivers

The first step is to install the drivers for the DuoMAG.

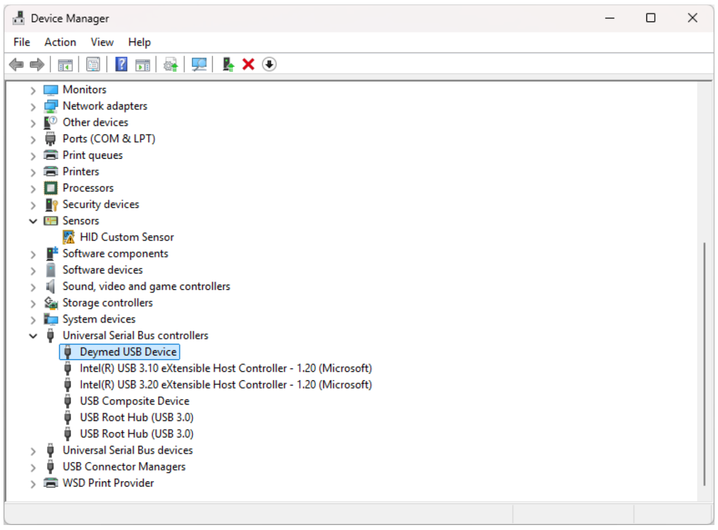
1. Connect the DuoMAG to your computer via a USB cable

2. Download the Deymed drivers
    - Find the drivers on the website:

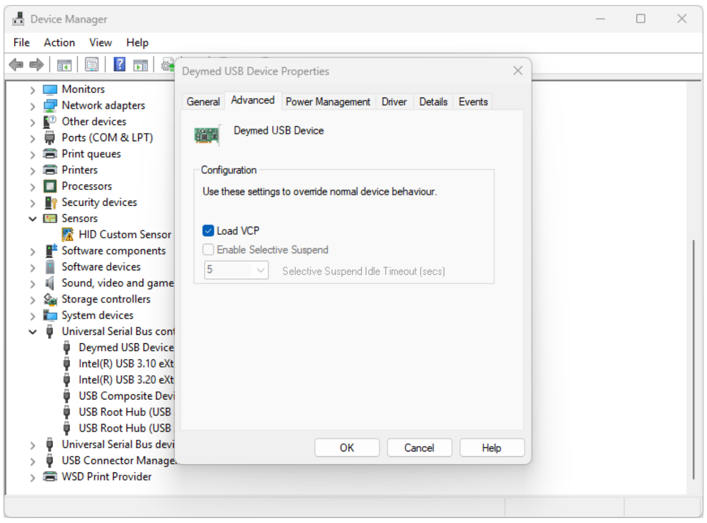
https://cloud.deymed.com/f/3064bae1bf9a45e5b6e0/

    - Download as ZIP (via green button in the top right)
    - Unzip the folder (should be named 'Deymed Drivers')

3. Open Device Manager (Windows) and find the DuoMAG, this should be listed under 'Universal Serial Bus controllers" as "Deymed USB Device"



4. Enable Virtual COM Port
    - Right click the device and go to 'Properties'
    - Click the 'Advanced' tab and tick 'Load VCP' then click 'OK'



Deymed USB Device

Configuration

Use these settings to override normal device behaviour.

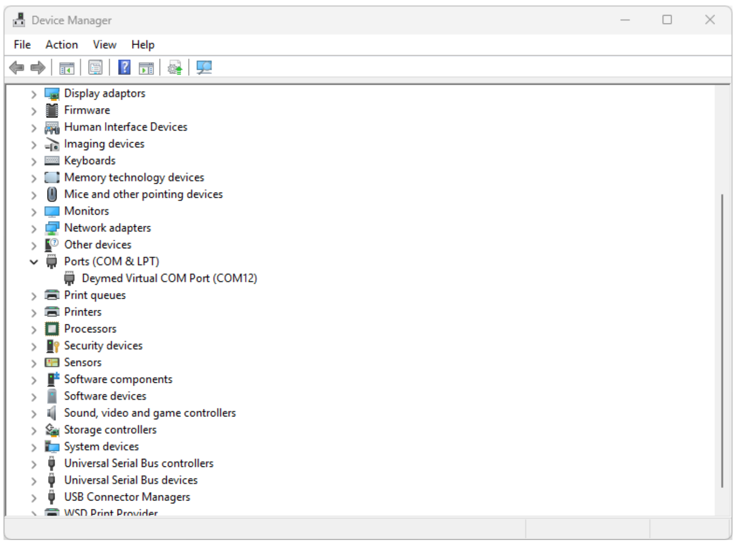
☑ Load VCP

☐ Enable Selective Suspend

5 Selective Suspend Idle Timeout (secs)

5. Restart your computer **(not optional!)**

After following these steps, the device should now also appear under “Ports (COM & LPT)” in Device Manager as Deymed Virtual COM Port (COMX). Depending on the age of the device, you may also need to install FTDI drivers.



```
> Display adaptors
> Firmware
> Human Interface Devices
> Imaging devices
> Keyboards
> Memory technology devices
> Mice and other pointing devices
> Monitors
> Network adapters
> Other devices
v Ports (COM & LPT)
      Deymed Virtual COM Port (COM12)
> Print queues
> Printers
> Processors
> Security devices
> Sensors
> Software components
> Software devices
> Sound, video and game controllers
> Storage controllers
> System devices
> Universal Serial Bus controllers
> Universal Serial Bus devices
> USB Connector Managers
> WSD Print Provider
```

For the DuoMAG MP Dual, both units will need to be installed separately following this process. This COM port is used when creating the DuoMAG object, so make note of it.

4. Python Requirements

Communicating with the DuoMAG device requires the **pySerial** library for serial communication. This can be installed using the following command:

```
pip install pyserial
```

To check whether pySerial is installed on your machine, you can run the following command in your Python environment:

```
pip show pyserial
```

This will display the installed version of pySerial. If it is not installed, you can install it using the pip command mentioned above.

5. DuoMAG Constructor

Before the rest of the functions can be used, a serial port object for the DuoMAG device must be created with the function DuoMAG.

The function is called in the following way:

```
import DuoMAG
controller = DuoMAG.DuoMAG("COMX")
```

Controller is the DuoMAG object returned by the function. This object is used to call further toolbox methods.

Note: Ensure that the DuoMAG device is connected via USB before calling this function.

Pulse

The Pulse method sets the intensity and requests pulses from the DuoMAG device.
The method is called in the following way:

Pulse(intensity, pulse)

- The input argument intensity should be a scalar value between 0 and 100. Any decimal values are rounded to the nearest integer.
- The input argument pulse should be set to 1 to request a pulse at the new intensity value. If no input for pulse is provided, the intensity is set without triggering a pulse request.
- If only the DuoMAG object is given as input, a pulse is requested at the current intensity of the device.

Example:

```
import DuoMAG
controller = DuoMAG.DuoMAG("COMX")
# Set the intensity to 25% and requests a pulse
controller.Pulse(intensity=25, pulse=True)
```

Note: The Pulse method checks if the coil is charged before generating a pulse. So if it called multiple times in a row, it will block the script until the coil is charged. Likewise, if the intensity is increased by a large number, there will be a delay between setting the intensity and the pulse while the coil charges.

6. ReadStatus

The ReadStatus method allows users to retrieve the current state of the DuoMAG device via the serial port connection. The state information is returned in a structured format, providing a detailed overview of the device's operational parameters.

The function is called as follows:

```
import DuoMAG
controller = DuoMAG.DuoMAG("COMX")
status = controller.ReadStatus()

for key in status:
    print(key + " : " + str(status[key]))
```

Purpose:

- This method is used to monitor the real-time status of the DuoMAG device, such as the current intensity level, potential overheating, or other operational states. It also checks for any errors related to voltage, fan operation, or coil disconnection.

This function is critical for ensuring the device is functioning within expected parameters during use and helps detect any issues early by reading various status indicators.

7. Recharge

The Recharge method is used to set the recharge delay for the DuoMAG device, allowing users to customise the delay between stimulation pulses. This function takes the current state of the device and adjusts the recharge time based on specified parameters.
The function is called as follows:

Recharge(steps, stepSize)

- steps: A scalar value between 0 and 127 that defines how many steps the delay should include.
- stepSize: A value in milliseconds used to multiply the steps input, defining the total delay. Supported values for stepSize include 0.05, 0.10, 0.20, 0.50, 1.00, 2.00, 5.00, and 10.00 ms.

```
import DuoMAG
controller = DuoMAG.DuoMAG("COMX")
DuoMAG.Recharge(DuoMAG, 60, 2)  # This sets the recharge delay to 120 ms
```

Purpose:

- This method allows for precise control over the timing of stimulation protocols by adjusting the recharge delay, which is crucial for achieving the desired effects in repetitive Transcranial Magnetic Stimulation (rTMS) applications. By customising the recharge delay, users can optimise the stimulation regimen for their specific experimental or clinical needs.

8. TTL

The TTL method is designed to set the TTL (Transistor-Transistor Logic) output delay for the DuoMAG device, providing users with the ability to customise the timing of TTL signals generated by the device. This method allows for precise timing control in experiments requiring accurate synchronisation of stimulation events.

The method is invoked as follows:

TTL(steps, stepSize)

- steps: A scalar value between 0 and 127 that determines the number of steps for the TTL OUT delay.
- stepSize: A value in milliseconds that multiplies the steps input to establish the desired total delay. Supported values for stepSize include 0.05, 0.10, 0.20, 0.50, 1.00, 2.00, 5.00, and 10.00 ms.

```
import DuoMAG
controller = DuoMAG.DuoMAG("COMX")
worker = DuoMAG.DuoMAG("COMY")

controller.Pulse(50) # Sets intensity of the controller to 50%
worker.Pulse(60) # Sets intensity of the controller to 60%

controller.TTL(DuoMAG, 60, 1)  # This sets the TTL OUT delay to 60 ms

controller.Pulse() # The worker should then pulse 60ms after
```

Purpose:

- By adjusting the TTL OUT delay, users can effectively synchronise the timing of external devices or processes with the stimulation output of the DuoMAG. This capability is essential for ensuring the accuracy and reliability of experimental protocols, particularly in applications involving repetitive Transcranial Magnetic Stimulation (rTMS) where timing precision is critical.

9. Installing FTDI Drivers

After installing the Deymed drivers, the drivers from FTDI need to be installed to allow the USB to function as a serial port.

1. Download the FTDI VCP drivers
    - Find the driver on the website: https://www.ftdichip.com/Drivers/VCP.htm
    - Unzip the folder (should be named 'CDM v2.12.28 WHQL Certified')

2. Open Device Manager and find the DuoMAG
    - This should be listed under 'Other Devices' as 'USB Serial Port'

3. Update the driver software

- 'Browse my computer for driver software' and select 'Let me pick from a list of device drivers on my computer'
- On the menu 'Select your devices type…', select 'Ports (COM & LPT)' and click 'Next'
- Select 'Have Disk…' and on the 'Install From Disk' tab click 'Browse'
- Find the folder downloaded from FTDI website named 'CDM v2.12.28 WHQL Certified', and select the file named 'ftdiport' and click Open
- On the 'Install From Disk' menu Click 'Ok'
- On the 'Select the Device Driver' tab click 'Next'
- Click 'Yes' when the warning appears – this should now install properly
- The device should now appear under 'Ports (COM & LPT)' as 'USB Serial Port' followed in brackets by the COM port name (e.g. COM3).

For the DuoMAG MP Dual, both units will need to be installed separately following this process.