Single Particle Tracking System setup

Overview of the System :

1. The sample emits photons that are received by the PMTs which convert them to electric analog pulses.
2. The pulse is shaped to TTL 3.3 V and 20 ns wide by the Constant Fraction Discriminator (CDF) in the Correlator Box.
3. The Pulses are acquired by the FPGA in the Arduino and transmitted to the Microcontroller which execute the position estimation and the feedback voltages.
4. Data for feedback voltage is (48MHz PWM) sent to an analog circuit which operate smoothing and shifting of the signal according to the specifications of the Mirrors drivers
5. In Parallel to 4, Estimated position and experimental data is transmitted to the PC and written to an XLS file via the python program.

A diagram of a computer system

Description automatically generated

Figure 3 All components of the system

Connections required

The Arduino MKR Vidor 4000 is the core of the system, its replacing the ISS DAC.

It’s placed on a breadboard circuit on which the analog electronic circuit is implemented.

The System should be put in correlator mode, meaning the SMAs from the PMTs should be connected to the Correlator.

* Two BNCs are connecting ISS Controller X/Y input to the ISS DAC.
* Disconnect the BNCs from the ISS DAC and connect them to the BNC of the breadboard circuit.
* Connect the Wall Charger connected to the breadboard power unit (see fig2)
* Connect the Arduino by USB to a PC (can be placed on the Optical Table, see fig 1).
* Output BNC from the correlator should be connected on the Arduino (purple in fig1).

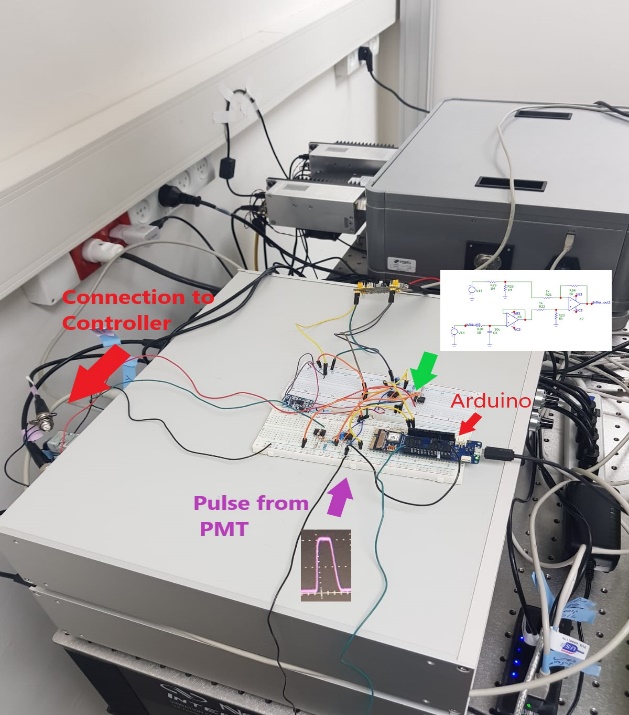


Figure 1 Overview on the connections of the system to the Arduino

A close-up of a circuit board

Description automatically generated

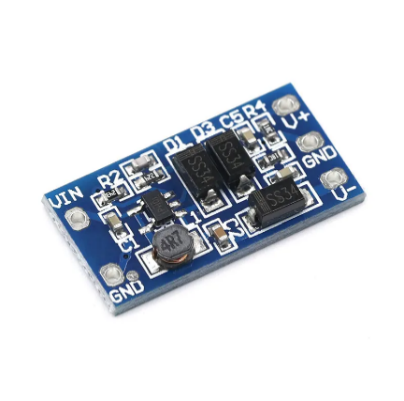


Figure 2 Wall Charger and Breadboard Power Unit

How the SW control the acquisition ?

A python program is controlling the whole system. It accepts commands for configuration and is in charge of execution of the data acquisition procedures.

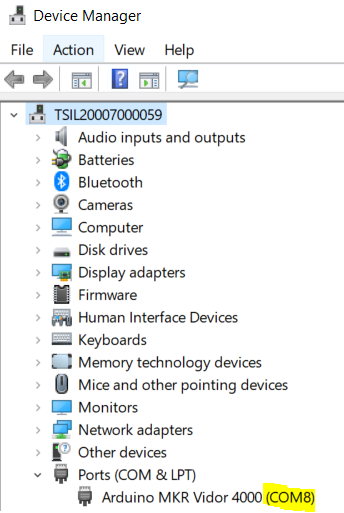
To use Real Time Burst Search which is implemented in the FPGA, config values (in **green**) should be configured by doing **Set** commands. **Background** command can be used to determine the proper Timing Window corresponding to the Background rate. Then a **BurstSearch** command can be used to make a burst search procedure. In both case an Excel with the measured timestamps data (in **blue**) is provided in the Working directory. The **Tracking** procedure consists of raster scanning the sample while executing RTBS and when the condition is met (burst\_flag = 1, in **yellow**) initiating the orbital rotation, collect # of photons received each ms (in **red**), estimate the position after each orbit and move the center of the orbit. This process is repeated until the object is considered as lost (burst\_flag = 0).

Its important to understand that while background and BS use timestamps data (blue track), Tracking is using the number of photons received in the 1ms interval (red track).

A diagram of a process flow

Description automatically generated

Figure 4 : From Up to Bottom, Python commands, code executed in MCU and HW implemented in the FPGA

You need to have python installed.

In Windows open “Command Prompt” Application.

Navigate to the folder containing the python script by using *cd C:\Users\dmergy\Downloads*

After plugging the Arduino by USB it will be connected to a port with a name like COM4, COM8 which is different between PCs and USB ports.

On windows, you can get the name from the Device Manager

This port name should be given to the python program as input.

If you are using Arduino IDE for further development close serial monitor.

Run:

*python Integration\_HWBurstSearch\_OrbitalTracking.py <USB Port Name>*

List of accepted commands:

* **background <value> msec/sec/min/photons**

Receive photons from the sample with a fixed laser, report timestamps in an Excel and compute the Background count rate (see [Burst Search in FRETBursts — FRETBursts 0.7.1+0.gc51b documentation](https://fretbursts.readthedocs.io/en/latest/burstsearch.html))

* + input (int): Time of measurement OR # of photons in the measurement
  + output : Excel File with photons timestamps measured

In the end of background procedure, a **set** command will be proposed with T from Background.

* **set T/N/F <value>**

Configure the HW for executing real time Burst Search

* + T (int): Timing Window to be used in BS
  + N (int): # of photons in the Timing Window
  + F (int): Multiplier of the BG rate to be used

For more explanation see : [Burst Search in FRETBursts — FRETBursts 0.7.1+0.gc51b documentation](https://fretbursts.readthedocs.io/en/latest/burstsearch.html#the-core-algorithm)

* **get T/N/F**

Print the config value used in the Burst Search algorithm.

* **bsearch <value> msec/sec/min/photons"**

Receive photons from the sample with a fixed laser, report timestamps in an Excel and perform real time Burst Search.

* + input (int): Time of measurement OR # of photons in the measurement
  + output : Excel File with photons timestamps measured and Burst Flag

* **start\_finding**

Receive photons from the sample while raster scanning with the laser, performing real time Burst Search until it a Burst is detected. Then begins an Orbital Tracking until the object is lost.

* + output : Excel File with # photons received at each point and estimated positions