Manual for the XUV-detector control

This is the manual for the control software of the XUV-detector at ESR. This includes the control for: 1. the motion of the cathode, controlled by a Trinamic stepper motor. 2. the (EA) power supplies, which supply the main and secondary coil of the detector.

Components

Raspberry Pi4 with a Pixtend V2S extension board

All programs are run from a Raspberry Pi4 using a command shell and python. A Pixtend V2S extension board is used to access several analog and digital inputs and outputs.

Trinamic stepper motor

A Trinamic stepper motor (TMCM-1161 V1.0) is used to move the cathode from the park position towards the beam and offers a various programming possibilities, including setting the speed, acceleration, end switches and more.

EA power supplies

Two EA power supplies (EA-PS 3016-10 B) are used to power the coils around the detector supporting up to 10 A and 16 V.

Connection of the hardware and ring signals

The stepper motor is connected via an USB connection and has to be powered by 24 V.

The Ea power supplies are connected with serial hubs. The connected Pins are:
pin 2: set current value,
pin 10: real current value,
pin 4: ground.
Further pins could control the voltage but the Pixtend board is limited to 2

analog outputs and inputs.

Trigger signals need to be positive TTL-signals on digital_IN_0 (Trigger IN) and on digital_IN_1 (Trigger OUT). The underlying handling of these signals are done by the Pixtend board. This requires the pulses to be at least 30 ms long, as this is the maximum expectable time for an internal cycle. Otherwise, trigger signals can theoretically be missed with the corresponding consequences. . .
The digital_OUT_0 is set active whenever the detector is not in the *Park position*.

Software

The software consists of several Python scripts. The general XUV-detector control not only sets up a PyQt interface and manages most of the detector but also starts all sub processes: The stepper motor control and the Pixtend board control. The later is connected with a MQTT eclipse Mosquitto broker.

Starting the software

Usually, the user will have to connect to the Raspberry Pi4 first. To do so open a terminal on any Linux GSI computer or on the linux laptop provided by Danyal (Passwort: 2020gwmx). Connect to the Raspberry Pi4 using ssh **-X pi@140.181.91.15** and enter the password: **SPARCcpu**. Subsequently change into the control software folder with **cd xuv_control_v1** and start the GUI with: **python Xuv-control.py**.

If the motor is not connected correctly the GUI wont start but fail with a connection error.

The GUI

The stepper motor which controls the mirror position is located on the left side and offers several options:

- If the motor looses connection during a measurement it will be displayed in the GUI and a **Connect:** button offers the possibility to try a

reconnect. In a normal run this will be not necessary.

- The **Current Position** is always displayed in mm together with the information if the detector is in the park or active position. As soon as the motor leaves the end switch the detector is labeled as **Active**, otherwise **Parked**.

- The buttons **Active position** and **Park position** move the detector to the designated positions. The default active position is set to 10 mm and the Park position moves the Detector as far out of the beam line as possible.

- **Set new active-pos** can be used to set a new position the detector moves to when activated. The maximum is 14.4 mm.

- **Automation** toggles if the detectors movement is controlled manually or by the ring signals. If the automation is turned on the detector will move in **30 s** after each *detector in signal* and move out immediately after each *detector out signal*.

- **Set new Acceleration** and **Set new speed** do exactly what the buttons say. Their default values are 100 and 2000.

- **left stop reached** and **right stop reached** are shown when ever an end switch is activated.

- **STOP** stops the movement of the detector immediately. Behind the stop button the current status of the motor is displayed.

- **Ring signals** are displayed at the bottom of the GUI

The coils are controlled by the right side of the interface and only offer limited settings:

- The smaller coil closer to the beam is labeled **secondary coil** while the bigger coil is labeled **Main coil**. Both can be controlled with **Set new current** with the suggested default values shown next to the Buttons: 2.8 for the secondary and 9.2 A for the main coil.

- The actual current is displayed for both coils with a precision of 0.1 A.

Generally the interface should be closed with the **Exit** button or the **x** in the upper right corner. closing the interface with strg-c might lead to errors and should be avoided. If the GUI crashes the detector is set to move to the park position but to be sure the GUI should be restarted in such a case.
Additional buttons like a possibility to define the Park position can be easily integrated. Each other parameter, for example the time delay of the automation, has to be set in the .py files but could be easily integrated into the GUI if necessary.