Report from the TeFeNICA 2020 Student Internship

Date: 12.X.2020

Name: Paweł Pietrzak

E-mail: pawel.pietrzak7.stud@pw.edu.pl

Division: Faculty of Power and Aeronautical Engineering

Period: 13 July 2020 – 30 September 2020

Supervisors: Marcin Bielewicz and Arkadiusz Chłopik

Subject of Practice:

Cosmic ray measurements in automation cycle using Python programming.

1. Introduction and motivation

A new accelerator complex, the Nuclotron-based Ion Collider fAcility (NICA) [1], is being constructed at the Joint Institute for Nuclear Research in Dubna to perform heavy ion collision experiments. The Multi-Purpose Detector(MPD) [2] was designed to track particles that are products of these collisions. The additional MPD Cosmic Ray Detector (MCORD) [3][4][5] was proposed as an upgrade of sub-detectors set of the MPD. This project proposes to surround the MPD barrel by one layer of scintillators. It can be used to test and calibrate MPD sub-detector systems and for identification of muons that are part of background cosmic radiation and those that are created inside the collider.

As part of work on the MCORD project, a set of simple CosmicWatch [6][7] muon detectors was used to test the methodology of cosmic rays measurements and to examine the influence of the MPD experimental hall concrete walls on the cosmic ray flux. The measurements have been carried out for two years as a part of student internship programmes. To improve their repeatability, a mobile platform for detectors was constructed (Fig. 1). It allowed the systematic control over the relative position of the detectors, which affects the measurement results. Creation of additional measurement automatization program was proposed for further optimisation of the data collection process. This program will allow to perform and monitor measurements in real time. Previously, the results of the measurements could not be read without stopping the detectors and there was no feedback information about successful start of the data collection process. The program will also assist in initial data analysis of the results by visualising them on graphs. The new mobile platform together with the software developed in this project will significantly improve the quality and repeatability of the measurements.

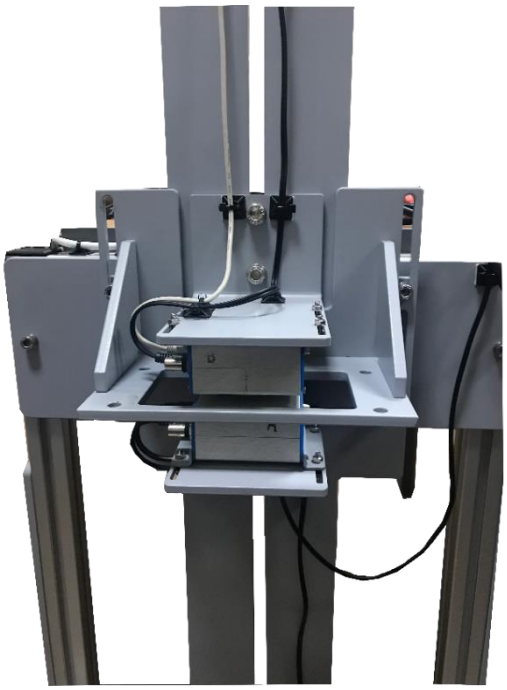

Figure 1: The platform for repeatable CosmicWatch measurements allowing the distance and angle regulation.

2. Background physics

Cosmic rays are high energy particles and gamma quants travelling throughout the universe. In case of particles, they are mostly protons (about 80%), alpha particles and some heavy ions. They originate from the Sun and objects outside of our solar system like other stars, supernovas or active galactic nuclei. Most particles come from the Sun, however many of them do not have enough energy to penetrate Earth's magnetosphere. Extrasolar and extragalactic cosmic rays are less common but are more energetic [8][9].

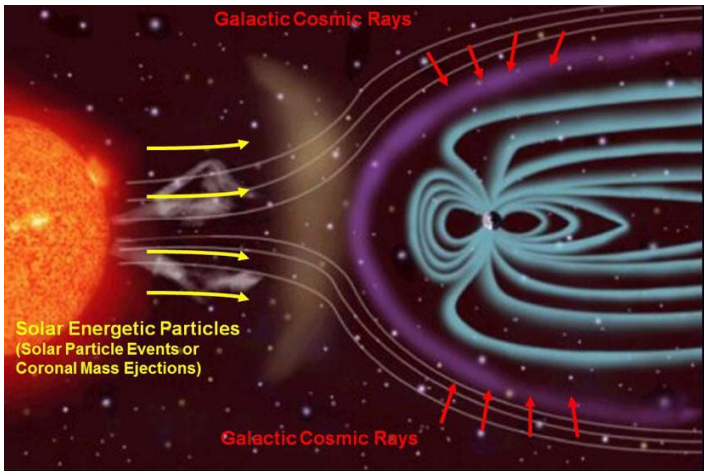

Figure 2: Cosmic radiation impact on Earth's magnetosphere.

When a primary cosmic particle collides with an atmospheric particle it produces many energetic hadrons, which decay on the way through the atmosphere. This causes a cosmic shower – a cascade of particles and electromagnetic radiation. Composition of this secondary radiation depends on the altitude above the sea level. On the ground level it consists mainly of electrons, gamma rays and muons.

Out of secondary particles created in the cascade, muons are the most frequently and most easily measured. They are charged particles and can excite a scintillator when passing through it. After excitation the plastic material emits photons of light which can be detected by other devices. In our program, for making such measurements, we have proposed to use simple and ready to use CosmicWatch devices, whose features and design will be described in the next chapter.

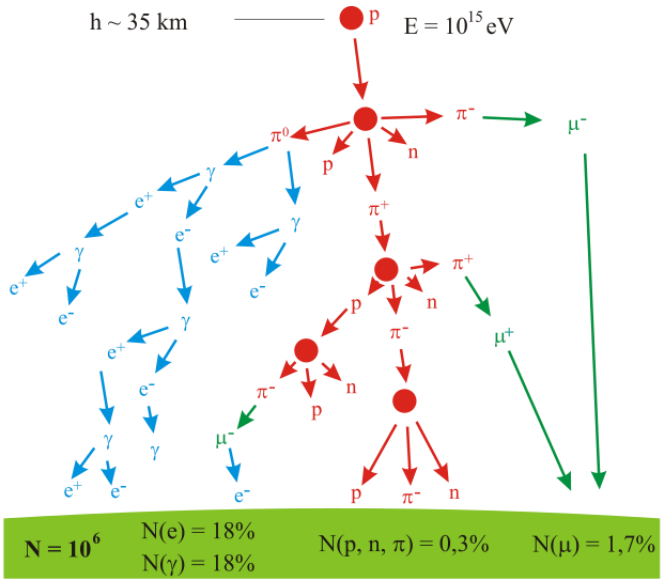

Figure 3: Cosmic shower composition at ground level.

3. CosmicWatch detectors

CosmicWatch is a desktop muon detector project developed in collaboration between the Massachusetts Institute of Technology (MIT) and the National Centre for Nuclear Research (NCBJ). CosmicWatch devices use plastic scintillators and silicon photomultipliers (SiPM) [10][11] to detect passing muons. They can be used to make normal physics measurements (it is important to remember that this equipment does not have high efficiency and accuracy) or as an educational tool for teachers and students.

Figure 4: Two CosmicWatch devices with a set of all required cables: a male to male 3,5 modified Audio Jack cable and two USB A-Mini cables.

Measurements can be saved on a SD card, shown on OLED screen and/or sent to a computer via USB bridge and received on a serial COM port. Data saved on the SD card and sent over port are the same and consist of the following elements and information: the first eight to ten lines are a header with information about detector. Next lines store the measurement data and include: event number, time in milliseconds from the beginning of measurement, digital signal value, analogue amplitude of signal in millivolts, deadtime of a detector in milliseconds and temperature inside the detector in °C respectively. Every value is separated by a space and a dot is used as a decimal separator.

3.1. CosmicWatch hardware

The main components of a CosmicWatch detector are:

- **Plastic scintillator bar** (5x5x1cm) – when muons pass through it they deposit a fraction of their energy. In response, the scintillator emits light (photons). The amount of light depends only on the distance it travelled inside the scintillator. The scintillator is almost entirely wrapped in a reflective foil so the light can only leave out through a single opening.
- **Silicon photomultiplier sensor** – it is attached next to the opening in the reflective foil. It produces a current pulse in response to the absorption of photons emitted by the scintillator. The pulse lasts only for fraction of a microsecond and has low amplitude.
- **Signal modulation circuits** – signal from the SiPM must be modulated so it can be detected by the microcontroller's 10-bit analogue to digital converter (ADC). An amplifier increases the amplitude around tenfold. Then, a pulse stretcher detects the peak amplitude and extends the duration of the amplified signal.
- **Arduino Nano Board** – it is used to perform several tasks:
    - Set the detection threshold on the ADC to ignore SiPM noise
    - Calculate the original pulse amplitude based on the ADC readout
    - Record the time of the event and its number
    - Control the OLED screen and the LED light; OLED displays detection count, measurement time, rate of detections and the latest signal strength; both can be turned off
    - Save the data on the SD card
    - Send the data to the USB port
    - Measure the dead time after each detection by subtracting the time it takes to execute each command from total measurement time; main sources of dead time are the OLED screen and the LED light

Figure 5: Left: The machined scintillator sitting on top of the reflective foil. Right: the covered scintillator with a 2 ×2 cm opening for the SiPM.

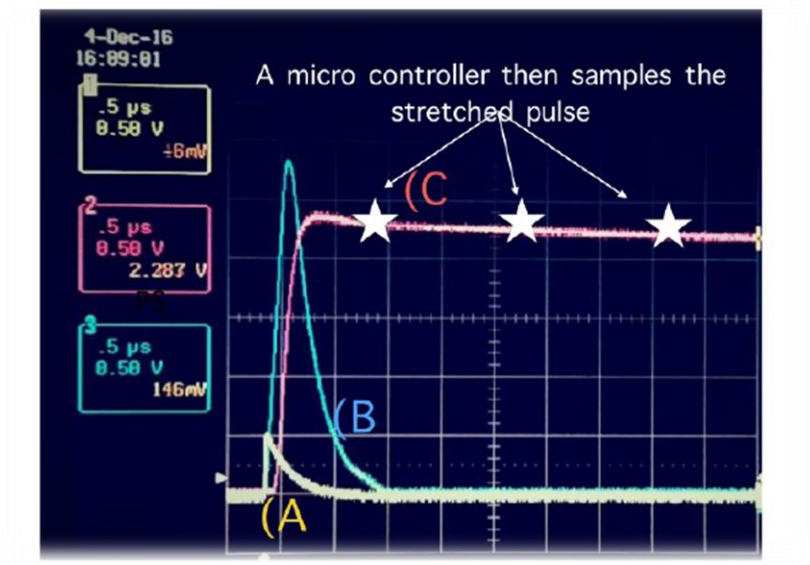

Figure 6: Oscilloscope readout in different parts of the circuit.(A) – original SiPM signal. (B) – amplified output signal. (C) – pulse stretcher output signal.

3.2. Coincidence mode of detectors

Two detectors can operate in a coincidence mode. To enable it, the two detectors must be connected together with a 3,5 mm audio jack cable and turned on in quick succession. In this mode, one detector will be assigned as a Master and the other as a Slave. Master will send a trigger signal through the audio cable after each detection of a particle. Slave will only register a detection if it receives a signal from Master within 30µs.

In coincidence mode (Fig. 7) only muons that have passed through both detectors are counted. This limits the amount of false detections caused by hardware anomalies and detecting other charged particles like electrons or high energy gamma quants. It also allows measurements to be made on a given angle range determined by the distance between used detectors.

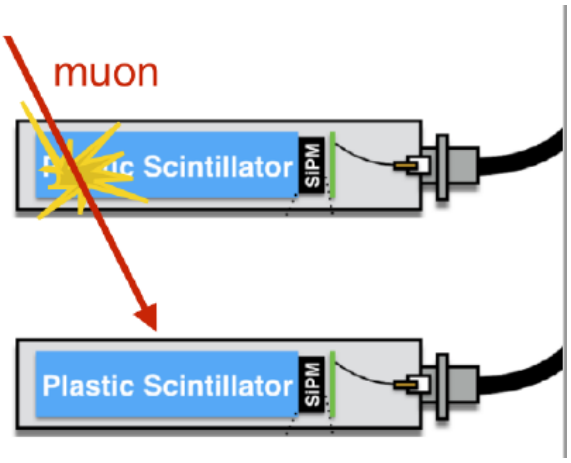

Figure 7:The coincidence mode of two CosmicWatch detectors.

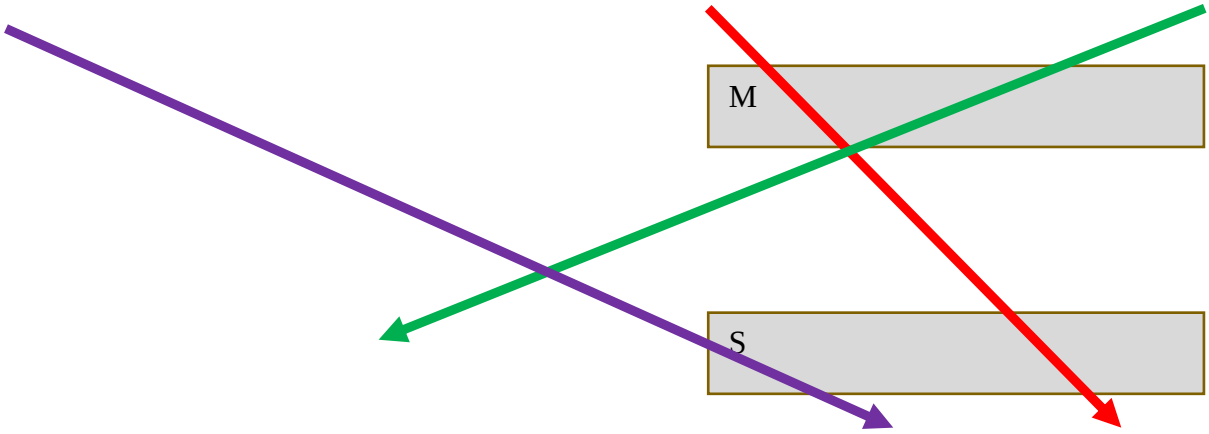

Figure 8: The principles of work in coincidence mode.

The principles of work in coincidence mode are shown in Fig. 8. Three muons are symbolised by coloured arrows. Master would register all muons that passed through it (green and red). Slave would only register muons that passed through both detectors (red). Muons that have passed only through Slave (purple) would not be registered.

4. Software

4.1. Project goal

The goal of this project is to automate the CosmicWatch data gathering process in a way that allows for taking the convenient and reliable measurements and easy data archiving and visualization for both single detector and detectors in coincidence mode.

There are already three existing ways to collect CosmicWatch measurements:

- **Using built-in SD Card Reader/Writer**– this method requires no additional software to be run on a computer, but it has some disadvantages: the data cannot be time-stamped by a more accurate computer clock; the SD Card has to be manually replaced when it gets full and its data files must be manually extracted; running SD Card Reader/Writer and OLED screen together will cause a significant increase in dead time, so there is no

display of live data. There is no way of checking if the data acquisition works correctly without ending it.

- **Using the official data archiving program** – compared to the previous method this one is far more automated, as live data is automatically saved to a specified file, it is timestamped and displayed in the terminal. Data in the terminal may be lost because of the unrelated computer problems and the data saving process is interrupted in case of any problem in communication between the devices.
- **Using the official web application** – it allows for data visualization in a form of various graphs, however, it requires a constant Internet connection. There is no control over an unforeseen measurement interruption and it is not possible to resume measurement in this case (no pause/resume options).

None of these three methods allows the user to save an additional data to files (e.g. distance between detectors) and all of them require the coincidence mode to turn on by manually resetting the detectors at the same time.

4.2. Used software

The project program was developed in Python 3.7 [12]. It is an interpreted high-level programming language with dynamic semantics. It stands out with its extensive error handling capabilities, simple and quick debugging and intuitive, easy to read syntax. These traits make the development time of Python programs shorter and easier to maintain. Another Python's advantage is a huge selection of modules.

PyCharm Community Edition 2020 was chosen to be used as an integrated development environment (IDE) for Python's program editing and debugging. It has an integrated debugger and an intelligent code editor allowing for fast refactoring and quick error identifying. Other important features are easy code navigation, that allows to save time on code edition, and extensive management.

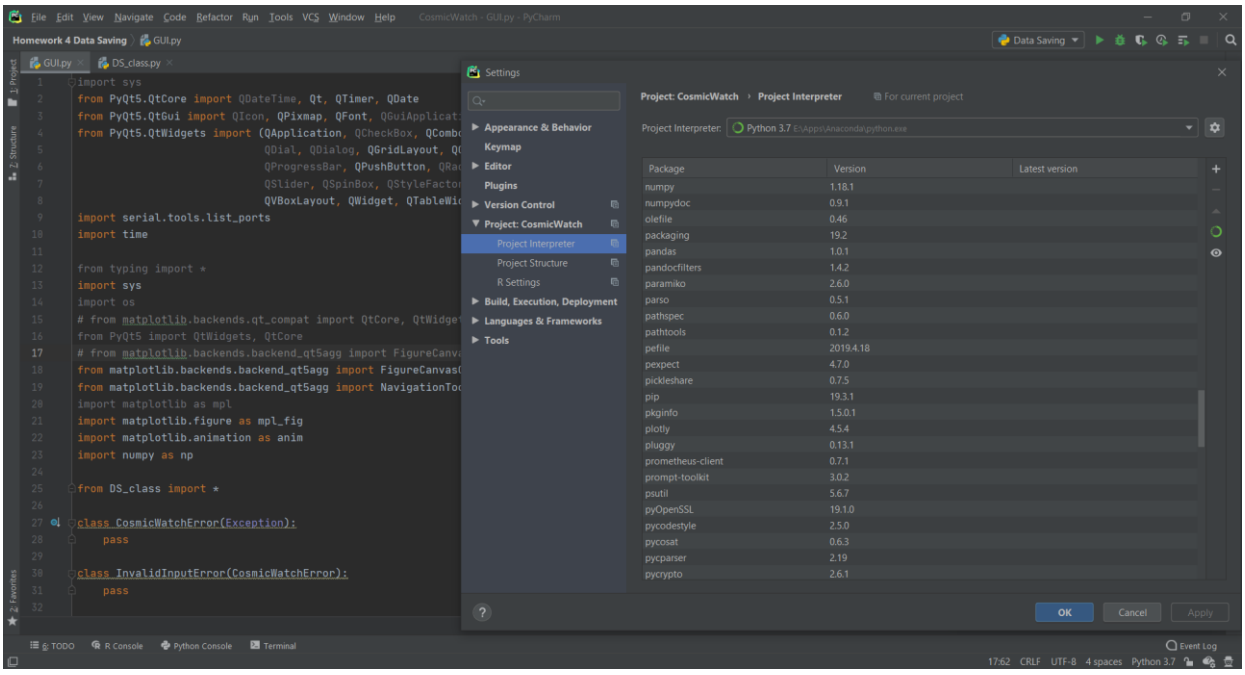

Figure 9:PyCharm IDE with opened module manager.

For developing the program code several Python modules were used:

- **PySerial** [13]– a library which provides support for serial communication; we use it to initialize CosmicWatch detectors and read the measurements over the USB ports
- **PyQt5** [14] – a plugin allowing to use Qt framework in Python language; Qt is a free and open-source cross-platform C++ framework used to construct graphical user interfaces (GUI) and multi-platform applications
- **Matplotlib** [15] – a comprehensive library for creating static, animated and interactive data visualizations; it supports many operating systems and cooperates with many frameworks for creating the GUIs; we use it to create data charts
- **NumPy** [16] – a package for scientific computing featuring support for multidimensional arrays, matrices and mathematical operations on them
- **Pandas** [17] – a library for data manipulation and analysis
- **Time** [18] – a module with time-related functions; we use it to read the computer time used for time-stamping measurements and calculating live and dead time

4.3. Program usage

The program graphical user interface consists of two windows, the main one (Fig. 10) and the chart window (Fig. 11). The main window is divided into five groups:

- **Detector settings** – in this panel user selects angle and inputs distance between detectors in appropriate boxes at the top of the panel; below are two combo boxes for selecting COM ports associated to the detectors – detector assigned to the port selected in the lower box will be initialized as Slave if detectors are in coincidence mode; “Scan ports” button in the middle serves for refreshing the list of existing COM ports
- **Control panel** – consists of “Start”, “Stop”, “Pause” and “Resume” buttons; “Start” begins the measurements and opens a live chart in a new window for each connected detector; “Stop” completely stops the measurements in a safe way and closes serial port connections; “Pause” orders the program to ignore incoming data on serial ports until the “Resume” button is pressed
- **Table** – it displays the last measurements from connected detectors in seven columns: detector name, status (Master/Slave), signal amplitude in millivolts, UTC time of the detection, rate, rate error and number of detections
- **Additional functions** – gathers three other subgroups:
    - “File reading”– consists of two buttons, both open a file selection dialog box that only accepts .txt and .csv files; upper button opens the file as an offline chart; lower opens it in a default system text editor
    - “Live charts” – a single button that opens live charts in a new window for each detector in case they were accidentally closed
    - “Time” tab – it has three timers of measurement process, showing the real, live and dead times.
- **Information box** – a text box at the very bottom of the window, that displays program errors, warnings and messages

Figure 10: Program main window.

Chart window consists of three elements:

- **Navigation toolbar** – a matplotlib widget; it is constituted of eight buttons at the top that can be used to: reset view, undo the last action, redo the last action, pan around the chart, zoom chart, adjust chart size, adjust chart axes, save histogram as a graphic file
- **Histogram chart** – it can be built by either live or offline measurements; the vertical axis shows the number of detections in logarithmic scale; horizontal axis can show analogue signal amplitudes or digital signal values in linear or logarithmic scale
- **Additional settings** – below the chart there are three pairs of radio buttons and a combo box: first two pairs control horizontal axis scale values, the last pair, together with the combination box, controls histogram bars colour and filling

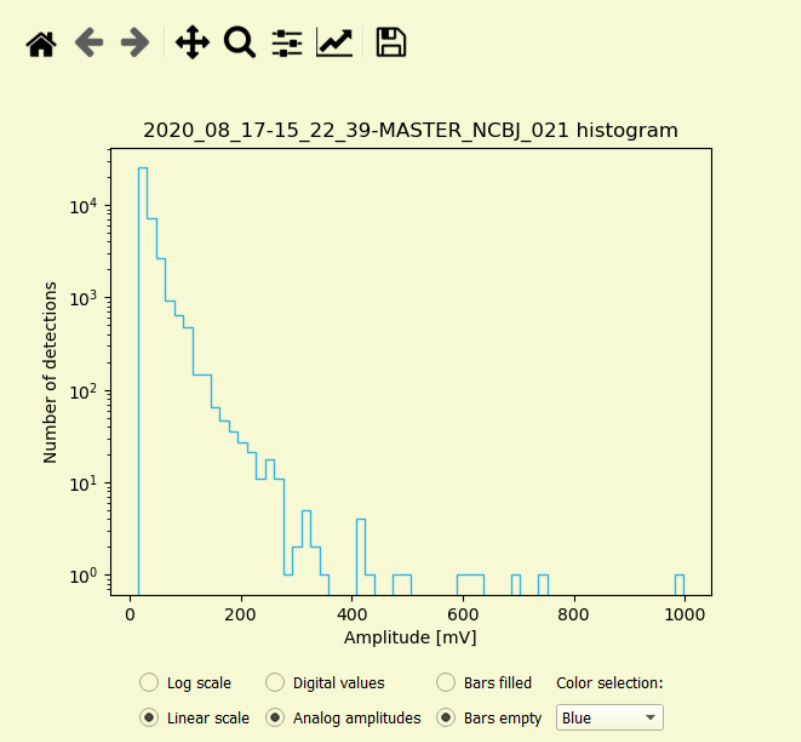

Figure 11: Program chart window showing offline chart loaded from a data file.

4.4. Saving files

After the beginning of a measurement process a text data file for each detector is created in CSV (Comma Separated File) format. The file name is based on its creation time, detector name and working mode. The format of this file can be seen in Fig. 12. The file starts with a header which describes the content of data columns, the device name, the device mode and data entered by the user: distance and angle. Data is separated from the header by an empty line. First two columns are a time-stamp for the UTC time zone based on the computer time. This time is added because Arduino clocks are inaccurate and two connected devices may desynchronize over a few seconds per day. Remaining columns are the same as those sent by the CosmicWatch over USB port (see chapter CosmicWatch detectors, paragraph 3).

```
################################################################################################
### CosmicWatch: The Desktop Muon Detector
### Questions? saxani@mit.edu
### Distance: 5 cm; Angle: 0.0 degrees
### Comp_date Comp_time Event Ardn_time[ms] ADC[0-1023] SiPM[mV] Deadtime[ms] Temp[C]
################################################################################################
DetectorID: NCBJ_026
DetectorMode: Master

2020-09-29 14:38:03.901 1 3233 143 25.74 0 24.23
2020-09-29 14:38:03.923 2 3263 71 19.06 5 24.12
2020-09-29 14:38:04.864 3 4195 396 85.28 10 24.23
2020-09-29 14:38:05.350 4 4689 316 56.81 15 24.23
2020-09-29 14:38:05.737 5 5044 387 81.05 20 24.23
2020-09-29 14:38:06.368 6 5680 60 17.43 25 23.80
2020-09-29 14:38:07.443 7 6787 89 21.05 30 23.80
2020-09-29 14:38:08.058 8 7401 207 35.26 35 24.12
```

Figure 12: An example file saved by the program.

4.5. Program operation

The program performs most actions using two custom objects. All GUI-related operations are performed by the **GUIControl** class. The COM port communication and data reading is handled by the **CosmicWatch** class. To avoid GUI interruptions on cross-thread actions, pyqtSignal object (from PyQt module) is used for communication between the two custom classes.

After the program is started, the GUI is built using the Qt framework. Then the program operates in four main stages (see block diagram of the program – Fig. 13):

- **(I) Collecting initial data** – the program collects user data (distance between the detectors and angle between them and the zenith, the used COM ports) and waits until the "Start" button is pressed
- **(II) Initializing detectors** – it starts with user data validation; if data is incorrect (e.g. no COM ports selected) then program goes back to the previous stage; if the validation is successful, a CosmicWatch object is initialized in a separate thread for each selected COM port; then serial port is opened (which causes detectors soft reboot) and start time is read from the computer time; afterwards headers sent by the detectors are read and analysed in search for detector name and mode; after the header is read completely a signal is sent to the **GUIControl** to update the data table and the data file is created and filled with the header
- **(III) Data logging loop** – each CosmicWatch object reads data portion (lines) sent to them by the detectors; lines beginning with a "#" are immediately saved to the file; other lines are time-stamped and analysed for data; **GUIControl** is signalled to update the data table and charts with new data
- **(IV) Safe program stop** – when the "Stop" button is pressed, the program is closed or if an exception in data reading occurs, COM ports get closed, data reading is stopped, files are saved and closed and data logging threads are terminated to free up the memory

5. Cooperation with NCBJ's Education and Training Division

CosmicWatch detectors used during program development were borrowed from the NCBJ's Education and Training Division. During the lending process, we have learnt that the Education Division is also developing a program for CosmicWatch data reading. Their program has different goals. It is intended to be used by students for educational purposes. Therefore, it does not support, for example, the recording of an angle or a distance between detectors and it may use a web application. Nevertheless, we established rules for output files format (see chapter **4.4**) to make the programs compatible on the data level and files from one program can be loaded up in to the other. The cooperation will be continued in form of information on the development work and experience exchange.

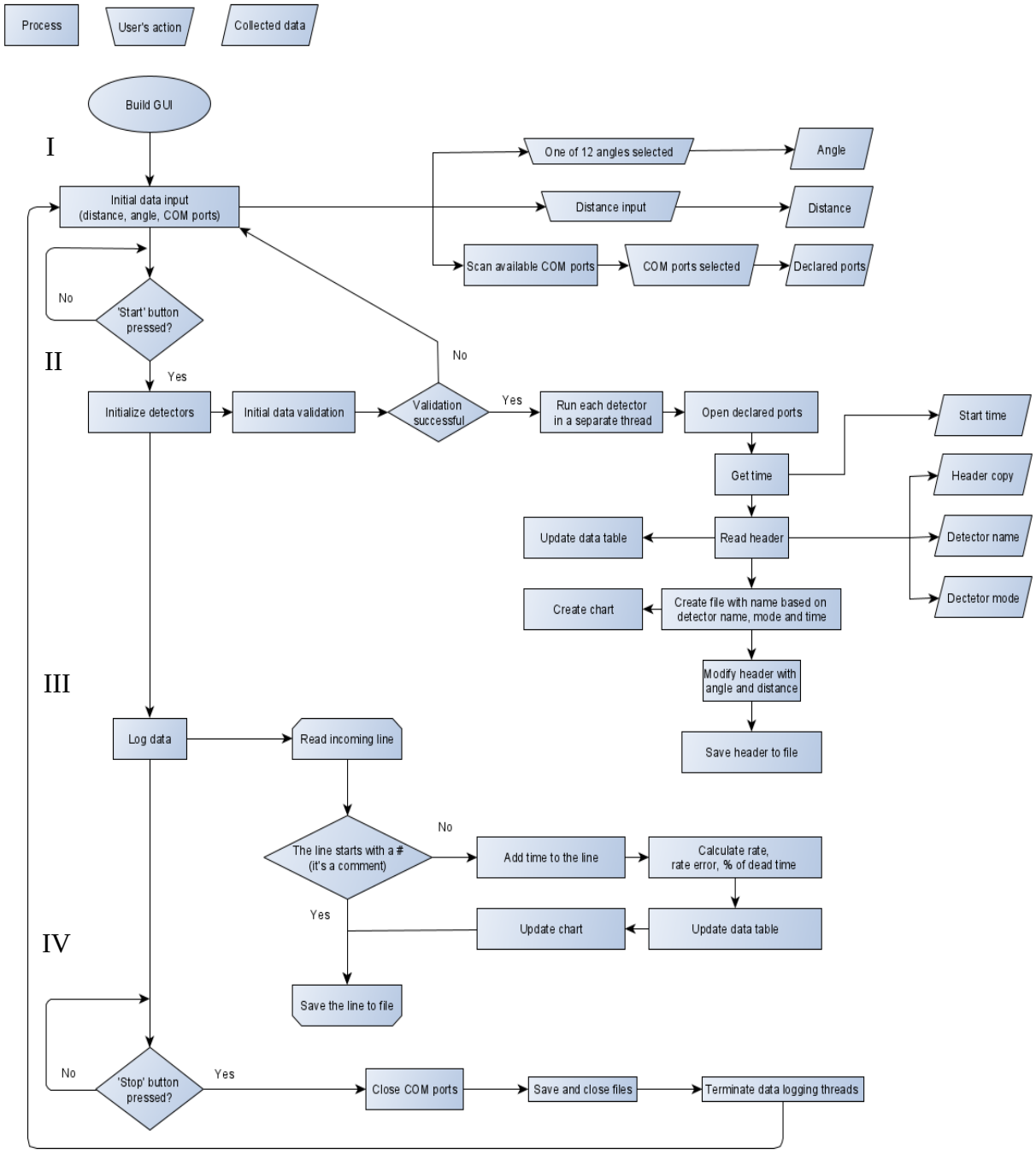

Figure 13: Program block diagram. The four main program stages are marked with Roman numerals.

6. Acknowledgements

Thanks to the people from the NCBJ Education and Training Division it is possible to use their two CosmicWatch detectors in our work on this project. We thank them very much for the help and support provided during the program development.

7. Bibliography

[1] V. Kekelidze et al. (2014). The NICA Project at JINR Dubna. *EPJ Web of Conferences. 71. 00127.*

[2] V. Golovatyuk, V. Kekelidze, V. Kolesnikov, O. Rogachevsky, & A. Sorin. (2016). The Multi-Purpose Detector (MPD) of the collider experiment. *Eur. Phys. J. A*, 52: 212.

[3] M. Bielewicz et al. (2018). MCORD: MPD cosmic ray detector for NICA. *Proc. SPIE Vol. 10808 No. 1080847.*

[4] M. Bielewicz et al. (2019). MCORD - MPD Cosmic Ray Detector a new features. *EPJ Web of Conferences 204, 07016* .

[5] M. Bielewicz et al. (2020). The cosmic ray detector for the NICA collider. *EPJ Web of Conferences 239, 07004.*

[6] Axani, S. N., Conrad, J. M., & Kirby, C. (2017). The Desktop Muon Detector: A Simple, Physics-Motivated Machine- and ElectronicsShop Project for University Students. *American Journal of Physics 85, no. 12,* 948-958.

[7] Axani, S. N., Frankiewicz, K., & Conrad, J. M. (2018). *CosmicWatch: The Desktop Muon Detector Instruction Manual.* Retrieved from https://github.com/spenceraxani/CosmicWatch-Desktop-Muon-Detector-v2/blob/master/Instructions.pdf

[8] T. Stanev. (2009). High Energy Cosmic Rays, 2nd edition. Springer.

[9] Z. Strugalski. (1993). *Promieniowanie Kosmiczne.* Warszawa: Oficyna Wydawnicza PW.

[10] M. Grodzicka-Kobylka, M. Moszyński, & T. Szczęśniak. (2017). Comparison of SensL and Hamamatsu 4×4 channel SiPM arrays in gamma spectrometry with scintillators. *Nucl. Instrum. and Meth. in Phys. Res. A, Vol. 856*, 53-64.

[11] M. Grodzicka-Kobylka, M. Moszyński, & T. Szczęśniak. (2019). Silicon photomultipliers in gamma spectroscopy with scintillators. *Nucl. Instrum. and Meth. in Phys. Res A, Detectors and Associated Equipment*, Vol. 926, 129-147.

[12] M. Lutz. (2013). *Learning Python, 5h edition.*O'Reilly Media, Inc.

[13] *PySerial documentation*. (2020, October). Retrieved from https://pyserial.readthedocs.io/en/latest/index.html

[14] *PyQt5 documentation*. (2020, October). Retrieved from https://www.riverbankcomputing.com/

[15] *Matplotlib documentation*. (2020, October). Retrieved from https://matplotlib.org/3.3.2/index.html

[16] *Numpy documentation*. (2020, October). Retrieved from https://numpy.org/doc/stable/index.html

[17] *Pandas documentation*. (2020, October). Retrieved from https://pandas.pydata.org/

[18] *Time module documentation*. (2020, October). Retrieved from https://docs.python.org/3/library/time.html