Rutherford Project Manual

Lizzie Bloomfield, Chloe Rey

August 2023

Abstract

This eight week placement involved the research and development of a multi-sensor array of particle detectors used in an updated Rutherford Scattering Experiment for the outreach team at Rutherford-Appleton Laboratories. The experiment uses an ITk pixel sensor with a RD53a chip, two Cosmic Watch detectors made of silicon photomultiplier (SPM) sensors and two SPM Micro sensors laid out in an array around the alpha source. The software applications Prometheus and Grafana were used to process and display the data from the experiment in a way that is accessible to a wide audience.

Contents

1 The Project

This short manual is a guide to the setup and operations of the remodelled Rutherford Scattering Experiment that has been made for outreach at RAL. It will go into the main components of the experiment and guide the user through how each one operates. All code used in the experiment has been collated on a GitHub page, which is linked below in Section 7, and can be used by the experiment operators if need be.

1.1 List of Main Components

- Raspberry Pi 4, IP-address 130.246.41.240
- 2x Cosmic Watch sensors
- 2x SPM Micro V2 sensors (aka John)
- ITk Pixel detector with the DAQ PC (heplnw014) aka "Orange monster", with the IP-address 130.246.43.14

2 Raspberry Pi

The Raspberry Pi (RPi) is what has been used as the main computer in the experimental setup. Data from the SPM Micro V2 (aka John) sensor is received directly by the RPi. Data from the Cosmic Watch includes an Arduino Nano which has its own code uploaded to it which collects the data to the display on the OLED screen attached to the Cosmic Watch. A python script on the RPi then puts this data into a text-file.

Earlier, when testing data inputs and Prometheus HTTP pulls, we created a multi button input to simulate a signal to test sending data to Prometheus and displaying the output in different ways in Grafana. There is a file called histogram_grafantest.py on the RPi which when run can differentiate which button is being pressed and sends that data to the Prometheus and then Grafana page. The circuit for one of these buttons is shown in Figure 3, to create a multi button input simply repeat this circuit using different GPIO pins on the RPi.

2.1 Prometheus and Grafana

The RPi has been fully set up with Prometheus and Grafana and all the necessary libraries downloaded such as Prometheus client which allows us to write python code to send data to Prometheus instead of using promQL, the coding language usually used for Prometheus. Prometheus is a software application used for event monitoring and alerting and is used in this experimental setup to monitor the hit/no-hit rate of alpha particles on all three types of detectors. It records metrics in a time series database built using a HTTP pull model, with flexible queries and real-time alerting. Grafana works in tandem with

Prometheus and is an open-source analytics and interactive visualization web application that allows us to display the real time hits of alpha particles during the experiment.

Prometheus and Grafana are set up in the RPi in a directory called "prometheus-grafana". To run them both, first make sure you are in the correct directory by running the command:

```
cd /home/clss/prometheus-grafana
```

Once you are in the correct directory, you can run both Prometheus and Grafana with the command:

```
docker-compose up
```

This command runs the docker-compose file in the prometheus- grafana directory. This will have Prometheus and Grafana running in the background so if you wish to access them this terminal needs to be left up and running. To access Prometheus on the RPi, you can type localhost:9090 in your web browser, to access Grafana, use localhost:3000 and to access the metrics page, use localhost:8000/metrics. The metrics page is a page where all the input data from the RPi is displayed in text form, so for example, in the case of the button inputs it will say the number of times the buttons have been pressed. To access these pages remotely, simply type the same URL into the browser of your PC but replace 'localhost' with the IP address of the RPi which is '130.246.41.240'. If connecting to any of these pages on a PC that has not previously done so, a firewall issue may occur.

When using Prometheus to push data and/or Grafana to display it, the docker-compose file needs to be running or "up". When done with the run, the docker-compose file can be closed with:

```
docker-compose down
```

2.1.1 Pulling Data to Prometheus

To get data into Prometheus initially, a library called Prometheus_client can be used directly in the Python code for the data input. The "button to prometheus"[1] file on our GitHub page is an example of this where, at the top of the Python script, prometheus_client is imported and the metric type is specified. An example of this method is the code used to pull data from a button click into Prometheus, listed in the Appendix below[8]. In this piece of code, the metric used is Counter although after more tests with the Grafana dashboard, we have decided to used the Gauge metric for all our Prometheus data pulls. If another data source is added to the experiment, it can be pulled to Prometheus using this method and there are other examples of full Python scripts using prometheus_client on our GitHub.

[1]GitHub page: `https://github.com/LizBloomfield/Rutherford-Software`

3 Cosmic Watch

There are two Cosmic Watches (CW) which each include an Arduino Nano with a naming.ino code uploaded to name them. this allows the RPi to identify each detector and receive data from both. In this case, we have named one Fabian and the other Fig, although this can be changed if needed in the naming.ino code. In addition to this, the OLED.ino python code is used to display the number of hits on the OLED screen attached. The CW can be connected to an oscilloscope in order to see hits as big spikes in the voltage that then discharges through a capacitor[2] For the sensor to detect anything reliable (aka not background noise) it needs a scintillator wrapped in black tape to block any exterior light to minimise noise but with a small square unwrapped on the side facing the SiPM on the CW for it to detect any light the aloha particles will make in the scintillator. The power cable is then connected to a USB port in the RPi so a python script labelled import_data_py3.py can be run which will output the data from the CW into a text-file called CW_data.txt. The output in this text-file should show nine different columns: date, time, event, Arduino time, ADC, SiPM, deadtime, temp and name. The column needed in this case is the SiPM column which shows the potential difference of the sensor in mili volts. An increase in voltage suggests a particle has been detector so a threshold needs to be set in a python code so that when a certain voltage is attained a hit is recorded.

4 SPMMicro

The SPMMicro (SPM) also known as 'John' is a essentially a simpler version of the CW. It has a light dependent silicon sensor which increases in voltage with an increase in light. It therefore works with a scintillator to detect alpha particles. There are two wires connected to the SPM, the red is positive and the grey negative or ground. The SPM gives an analogue output which is a format the RPi does not recognise. Therefore, an Arduino Nano with the necessary code uploaded is used to translate the analogue data to digital. The grey wire of the SPM is connected to the ground of the Arduino and the red is connected to the pin labelled A0. The Arduino itself is then connected to the RPi through its power cable and a USB port on the RPi. A python script, called john_output.py on the RPi, has been written to detect increases in voltage over a certain threshold and record that as a hit. The python script can be adjusted to print the data it is receiving which will be needed when testing with alpha particles as the threshold may need to be adjusted accordingly.

[2]These spikes are usually from cosmic rays and muons, however for this project, the CW will be modified to detect alpha particles.

5 ITk Pixel Sensor

The final type of detector used in this experiment is the ITk pixel sensor. Juliette Martin has been kind enough to write a comprehensive guide on how to set up and run the ITk pixel sensor[3] but as it has already been setup and tuned for this experiment, this guide shouldn't be needed for the main experiment but is available for troubleshooting. The bash script ITK_bash.sh has been written and when run, should prompt the ITk to run a noise scan, translate the .json file produced into a simple hit/no-hit text-file and push the data to Prometheus on the RPi. The noise scan itself has a time limit on the scan of 30 seconds although that can be changed by accessing the bash script for the ITk and changing the time limit manually.

The python file promhttp.py translates the .json noise scan into a text-file that is then pushed to Prometheus and is also where the detector threshold is set. Currently, the threshold is set at 200e but this will need to be changed, as this was used to check if the code can differentiate between hit/no-hits. For alpha particles, a threshold of 500e or more would be best as this is higher than almost all of the background noise picked up in the initial noise scan that can be seen in Figure 1.

6 Final Setup

Each detector will have their own OLED screen which will display a histogram bin. When a detector is hit by an alpha particle, the OLED screen can be coded to show a small flash and/or its histogram bin can increment by one. This means that as the experiment progresses, you can see the angular distribution of hits one each detector. Each detector has a different level of sensitivity and so will therefore need to be calibrated to a similar accuracy. Figure 2 shows a rough sketch of how the experiment can be setup.

6.1 Outreach Aims

This whole experiment has been built for outreach purposes at Rutherford-Appleton Laboratories, with the aim to create a working Rutherford Scattering experiment. The placement itself has been used for the research and development of a multi-sensor array to replace the zinc sulphide screen of the original experiment.

7 External Links

Github page: `https://github.com/LizBloomfield/Rutherford-Software`

[3]Yarr document of ITk by Juliette Martin: `https://cernbox.cern.ch/remote.php/dav/public-files/5DPDmtoastE61wF/YARR_instructions.pdf?access_token=null`

Yarr document of ITk by Juliette Martin: https://cernbox.cern.ch/remote.php/dav/public-files/5DPDmtoastE61wF/YARR_instructions.pdf?access_token=null

8 Appendix for Code and Figures

Code 1: Python code to pull data from a button click into Prometheus, an example of the use of prometheus_client

```
import RPi.GPIO as GPIO
import time

from prometheus_client import start_http_server, Counter

#main bulk of button code is here (shortened for simplicity)

# Start Prometheus metrics server
    start_http_server(8000)
    print("Prometheus metrics server started")

    try:
        while True:
            time.sleep(1)
    except KeyboardInterrupt:
        GPIO.cleanup()

if __name__ == '__main__':
    main()
```

Figure 1: Noise Scan of background noise from the ITk Pixel sensor.

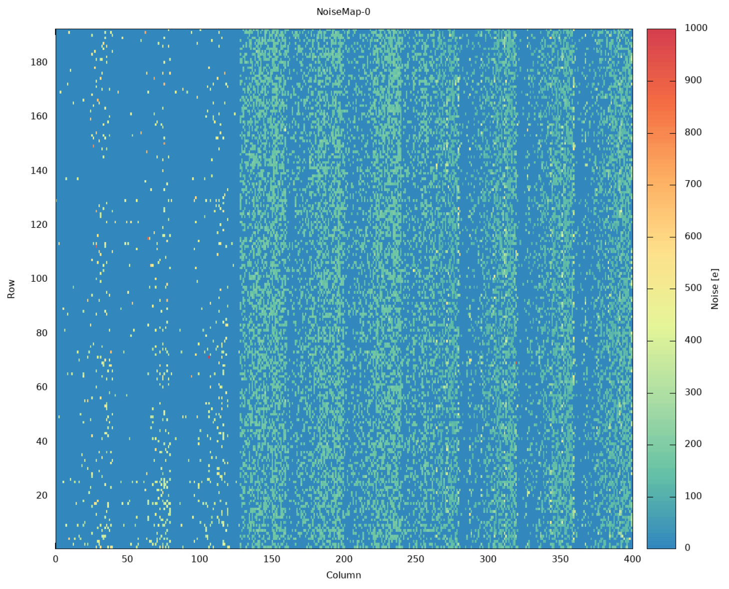
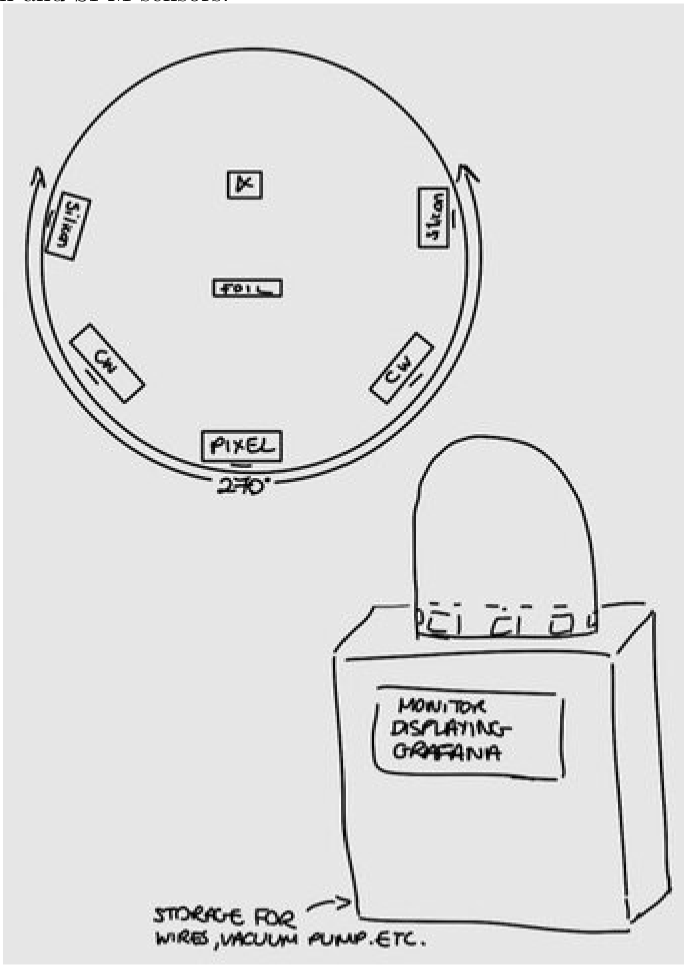
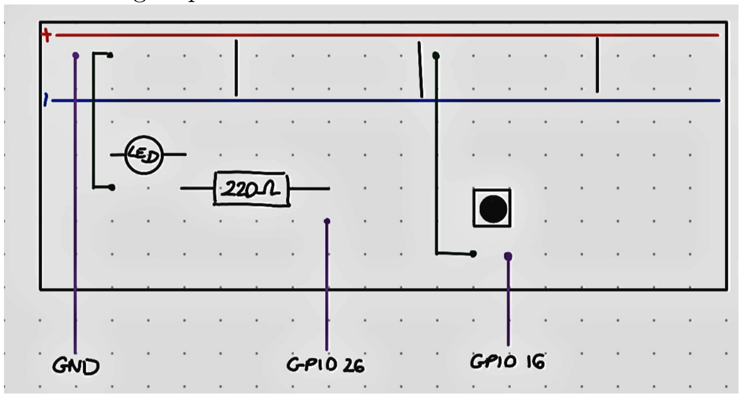


Figure 2: Sketch of layout for final experimental setup, including the ITk, Cosmic Watch and SPM sensors.



IX
Silicon
Silicon
FOIL
CW
CW
PIXEL
270°
MONITOR DISPLAYING GRAFANA
STORAGE FOR WIRES, VACUUM PUMP. ETC.

Figure 3: Circuit on a breadboard used to send button data to prometheus, Purple lines leading to pins on RPi


+
-
LED
220Ω
GND
GPIO 26
GPIO 16