Università degli studi di Torino

Sessione di laurea ottobre 2016

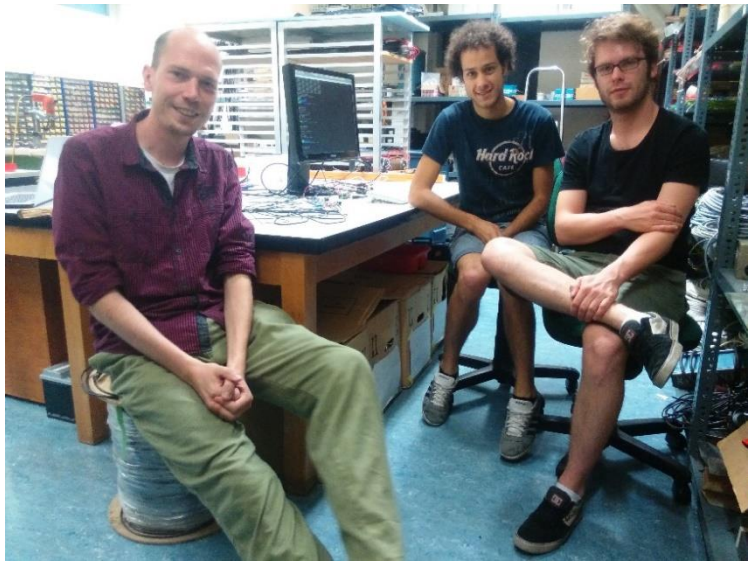
Karlsruher Institut für Technologie

COSMIC REVELATION SHOWCASE

Candidato: Daniele D'Ago

Relatore: Mario Edoardo Bertaina

This project is the experience I did with Erasmus+Traineeship at KIT, Karlshrue in June and July 2016 and in Stuttgart in September 2016.



COSMIC REVELATION

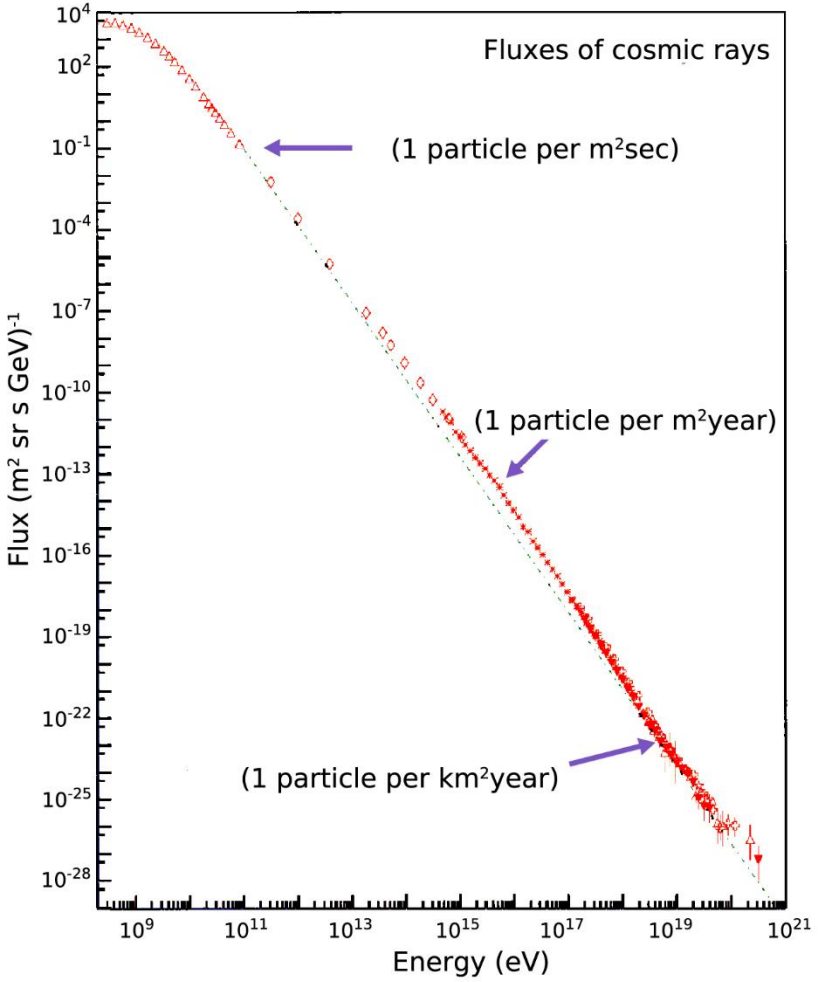
Cosmic revelation is an art project from Tim Otto Roth which involves detection of cosmic showers.

It aims to interest people in physics.

COSMIC RAYS

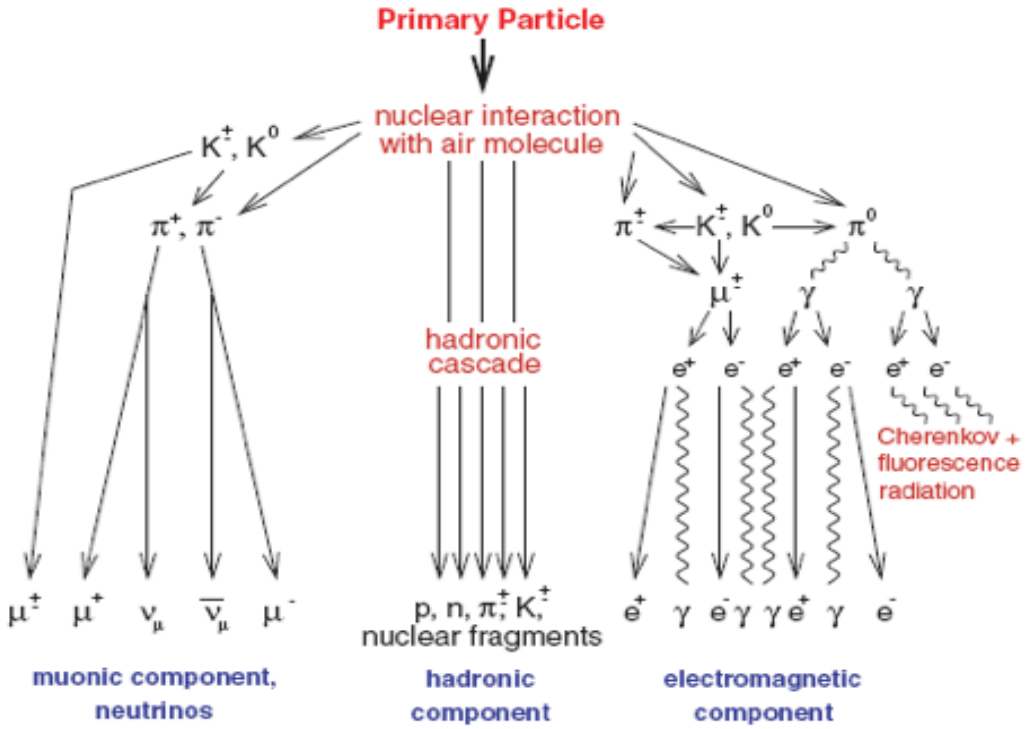
Cosmic rays are high energy nuclei of various chemical elements produced and accelerated in astrophysical environments.

Flux of cosmic rays on earth strongly depends on energy.



COSMIC RAYS

Cosmic rays interact with earth atmosphere, giving birth to «showers» of particles.

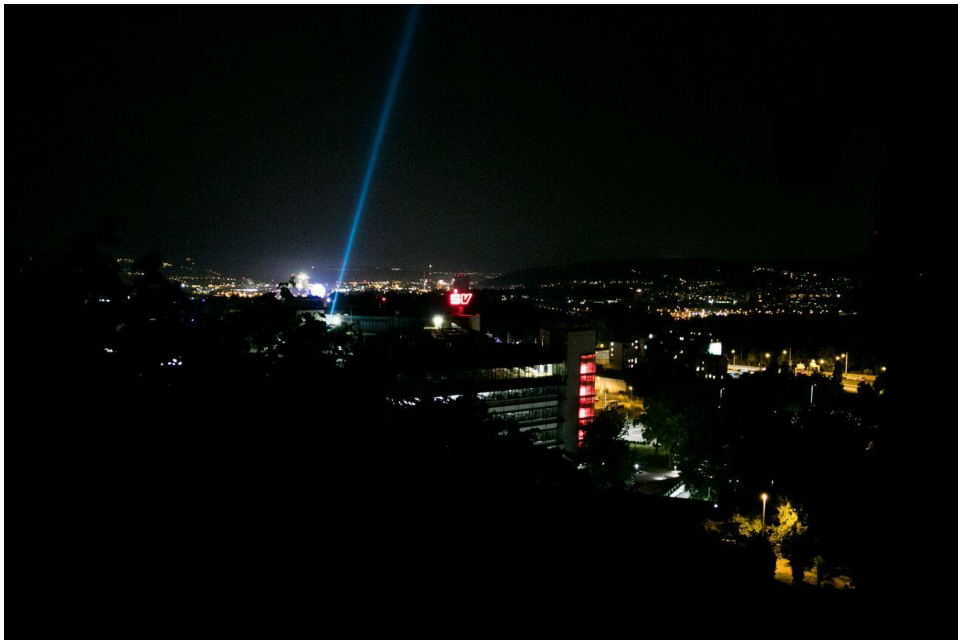


Arrays of detector allow us to reconstruct important parameters, such as the direction of the primary particle.

KASCADE

Kascade was an extensive air shower array to study the cosmic ray primary composition and the hadronic interactions in energy range $10^{14} \div 8 \cdot 10^{16}\ eV$. It was placed at KIT – CN in Karlsruhe, Germany. The array consisted of 252 detector stations arranged in a total surface of 40 000 m^2 (a square with 200 m side).

COSMIC REVELATION



Detectors from KASCADE experiment have been placed on the roof of Sparkasse Versicherung building in Stuttgart. As a cosmic shower hits the roof, lights in the staircases flash and a skybeamer point in the direction of the shower.

WHY THE SHOWCASE?

COSMIC REVELATION does not run during the day

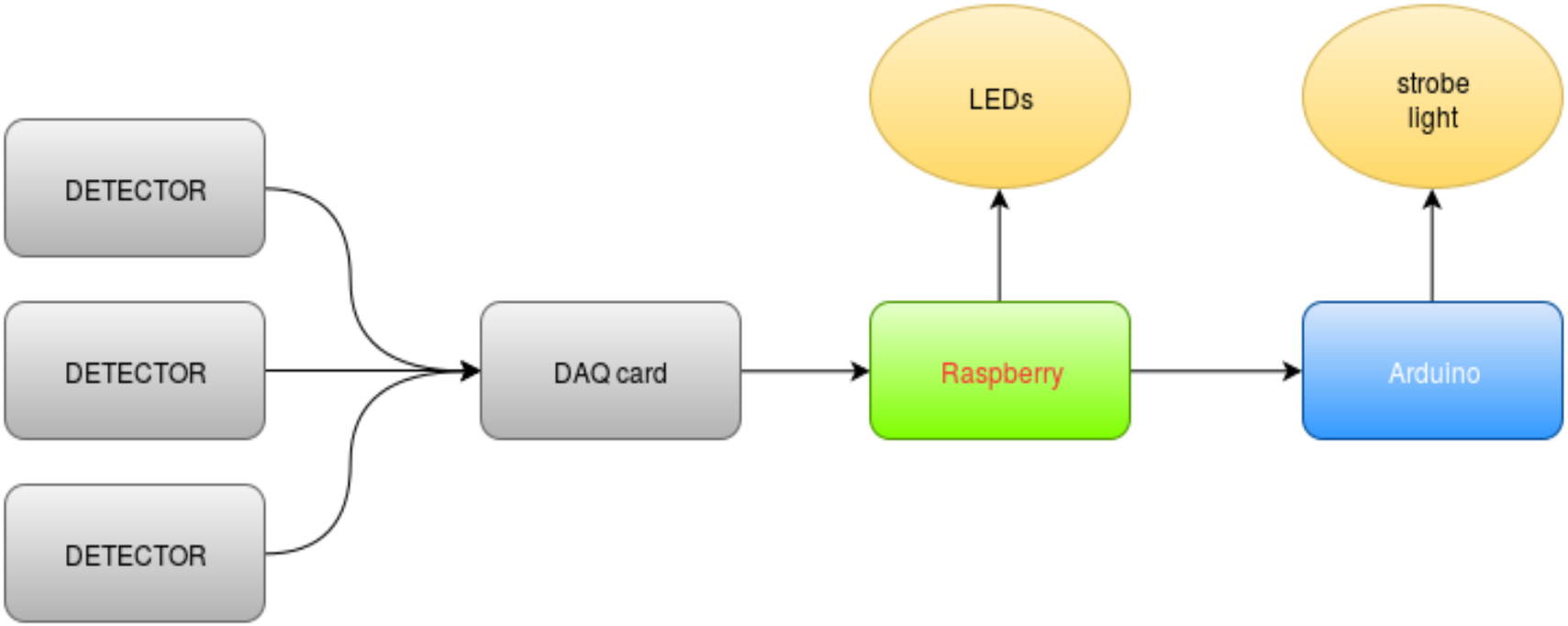
+

Explanation of the project

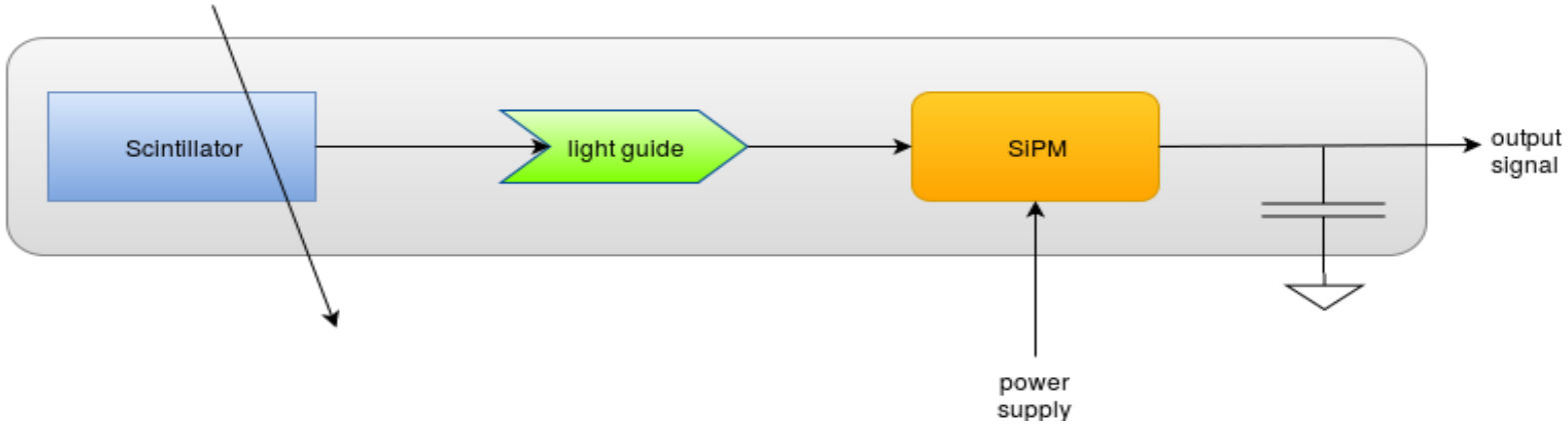
→

SHOWCASE

SHOWCASE SCHEMATIC



DETECTOR - SCHEMATIC



The detectors used are part of CosMo experiment kit, developed at DESY.

We are not able to access the inner parts of the box.

The signal we get is not coming straight from the SiPM, but it is filtered: we have to deal with it!

DAQ CARD

The DAQ card used is the one provided by the CosMO experiment kit.

SIGNAL INPUT is managed through four analog inputs (only three used).

DAQ CARD

SETTINGS INPUT managed through the USB port. Allows the user to manually set:

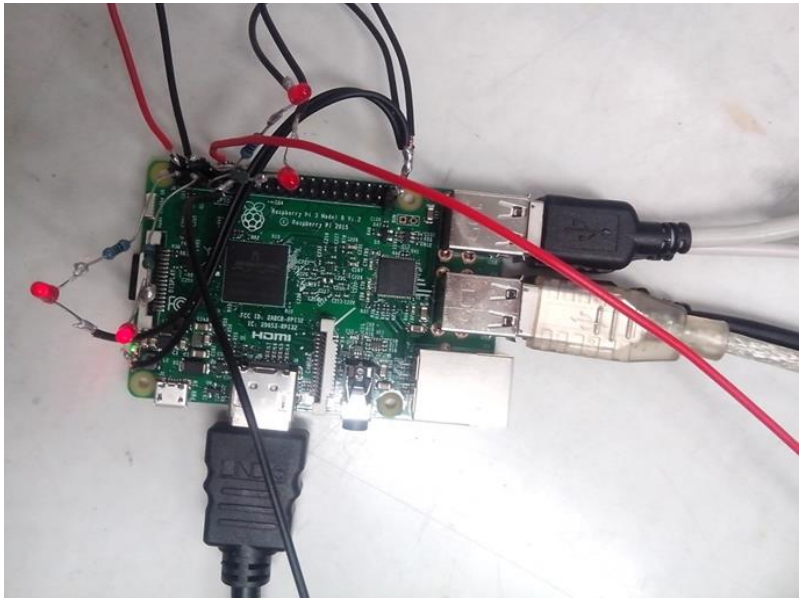
- THRESHOLDS for the three channels
- NUMBER of triggered channels
- WIDTH of coincidence gate

SIGNAL OUTPUT managed through the USB port. Required information need to be extracted by the interface programme.

RASPBERRY PI

A Raspberry PI works as the main controller of the showcase.

- Communication with DAQ card (input and output)
- LEDs control
- Arduino control



During testing and building phase directly controlled by the user.

In the final setting in autoplay mode.

RASPBERRY PI - code

Communication with the serial port is managed using the python library «serial».

- Read
- Write

Communication with LEDs and Arduino are managed by the GPIO pins.

RASPBERRY PI – DAQ inputs

User can directly handle some parameters by sending strings to the DAQ (write function) :

- Thresholds
- Coincidence gate width
- Number of triggered channels

RASPBERRY PI – setting time

Since the DAQ board timer overflows at ~170s the raspberry needs to count the additional cycles.

During running phase in the final setup, the time of capture needs to be set to «infinity». This is provided by a infinite for-loop.

RASPBERRY PI – DAQ outputs

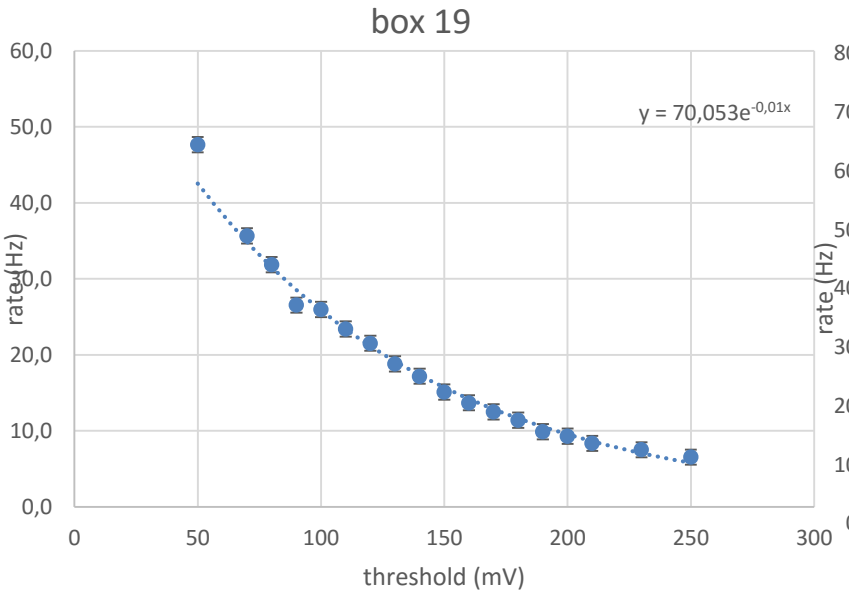
The DAQ output is read by the raspberry Pi through the USB port with the function «readline».

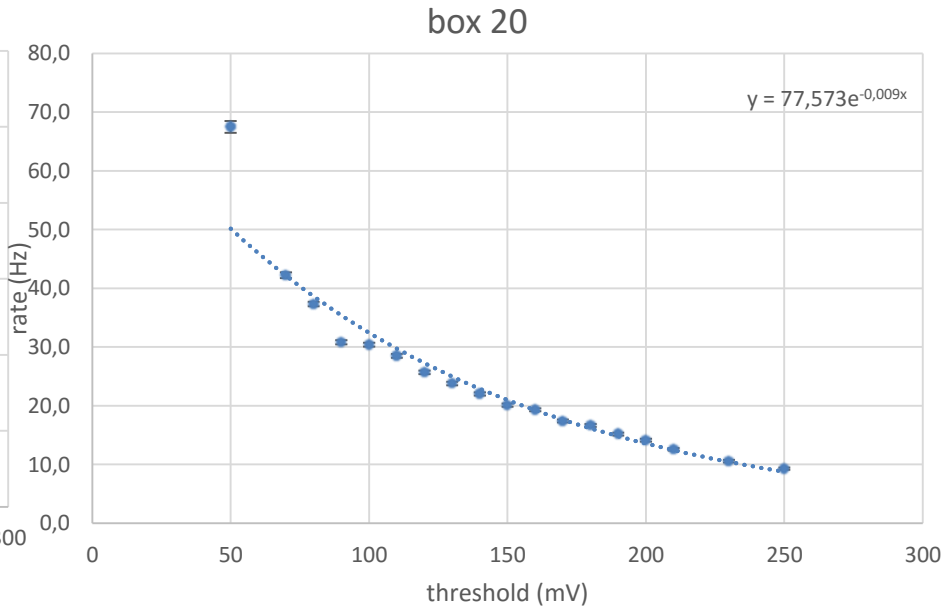
The function «readline» is called during the for loop once every 12 ms (time needed for executing instructions in loop).

```
[# events in ch 0, # events in ch 1, # events in
ch 2, events in ch 3, # coincidences, time]
```

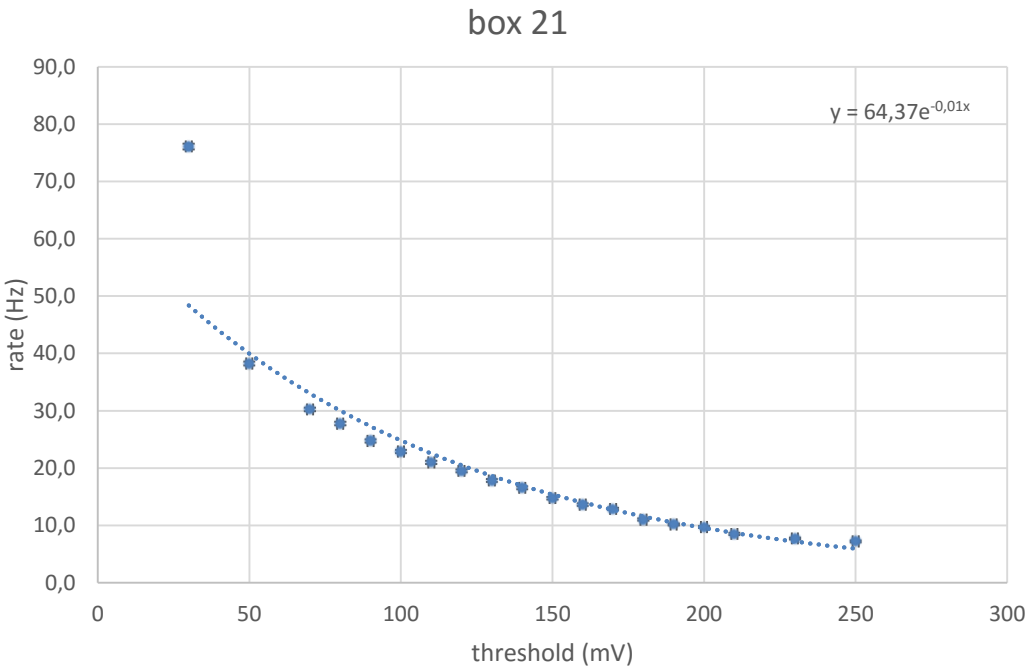
DAQ CARD – THRESHOLDS

Thresholds had been chosen in order to get the expected coincidence rate and to get the same rate in each detector.





DAQ CARD – THRESHOLDS



According to data fit, thresholds selected are:

99 mV	box 19
121 mV	box 20
87 mV	box 21

These values provide a rate $r \approx 20$ Hz

DAQ CARD – THRESHOLDS

Expected flux for muons is $\phi_e \approx 200$ Hz/m^2

Since each scintillator has a surface S = 0.04 m^2, we expect to have a rate $r_e \approx 8$ Hz

The rate selected according to fit is slightly higher. Anyway, this is not a problem for measuring coincidences. We can estimate the rate of accidental coincidence with selected single rates.

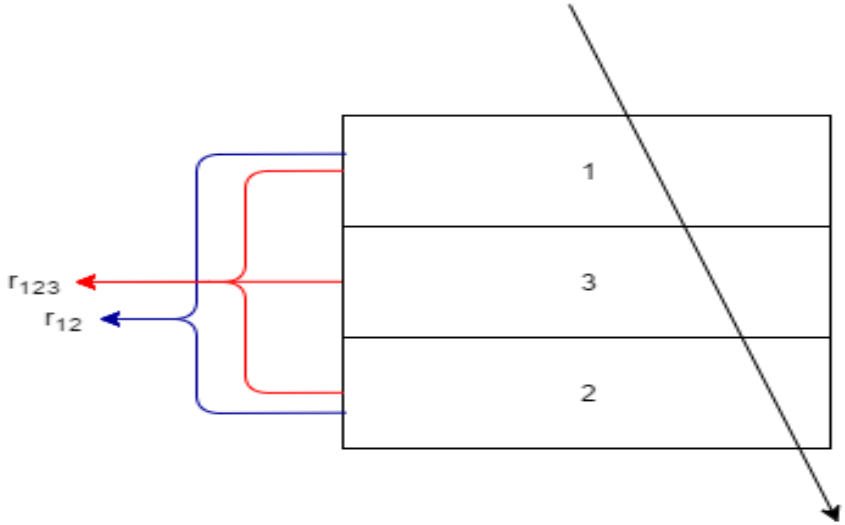
$$r_{acc} = (2\,\Delta t\, r_1 r_2)(\Delta t\, r_3) = 1.6\ 10^{-9}\ \text{Hz}$$

Which is infinitesimal compared to measured coincidence rate

$$r_c = 0.02\ \text{Hz}$$

DETECTOR - EFFICIENCY

To evaluate the efficiency of the detectors we measure rates of double and triple coincidence.



PROBLEM: we can not measure double and triple coincidences during the same data capture.

SOLUTION: 2h data capture time double, 2h triple.

DETECTOR - EFFICIENCY

19
20
21

21
19
20

20
21
19

r_d = 3.84 ± 0.02 Hz
r_t = 3.67 ± 0.02 Hz

ε_{20} = 0.956 ± 0.001

r_d = 3.80 ± 0.02 Hz
r_t = 3.68 ± 0.02 Hz

ε_{19} = 0.969 ± 0.001

r_d = 3.86 ± 0.02 Hz
r_t = 3.73 ± 0.02 Hz

ε_{21} = 0.967 ± 0.001

DETECTOR - EFFICIENCY

We can give an esteem of the efficiency as the mean efficiency.

$$\varepsilon_{tot} = 0.964 \pm 0.007$$

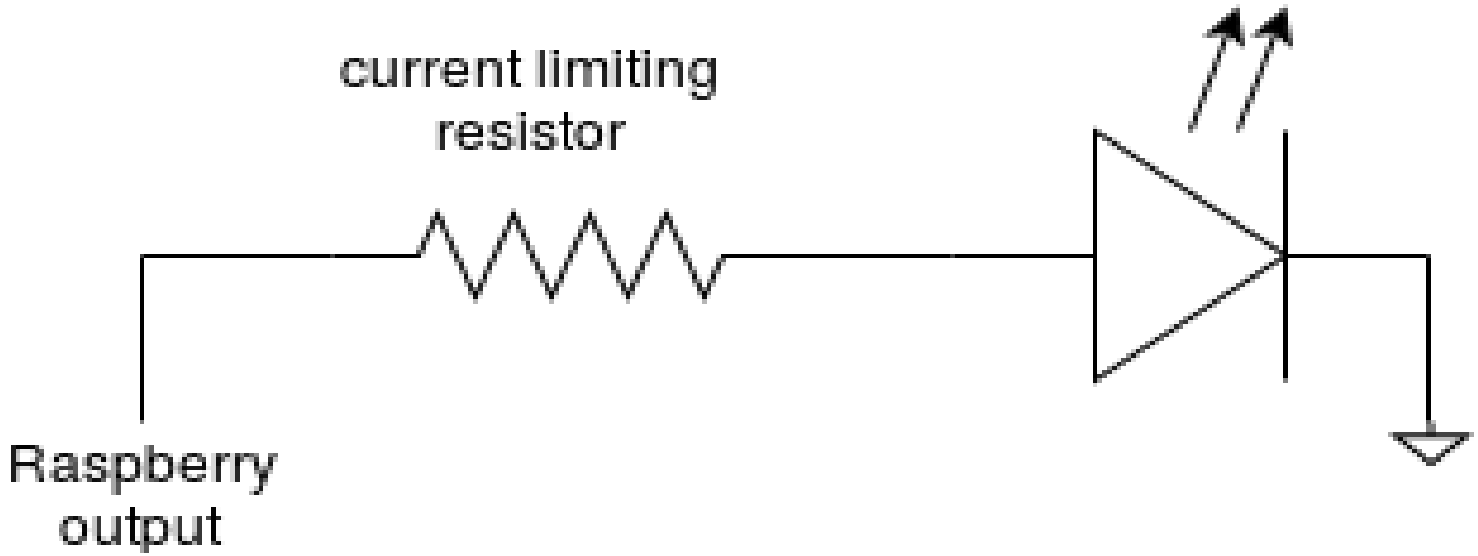
RASPBERRY PI – outputs

For every new DAQ outputlist, Raspberry checks if any of the numbers had changed.

If one of the numbers had changed, no matter what is the new value, Raspberry sets to HIGH the corresponding pin for a 10 ms time, then it sets it back to LOW.

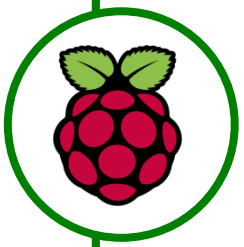
RASPBERRY PI - LEDs

LEDs are connected to the pins referred to the single detectors.



RASPBERRY PI - code

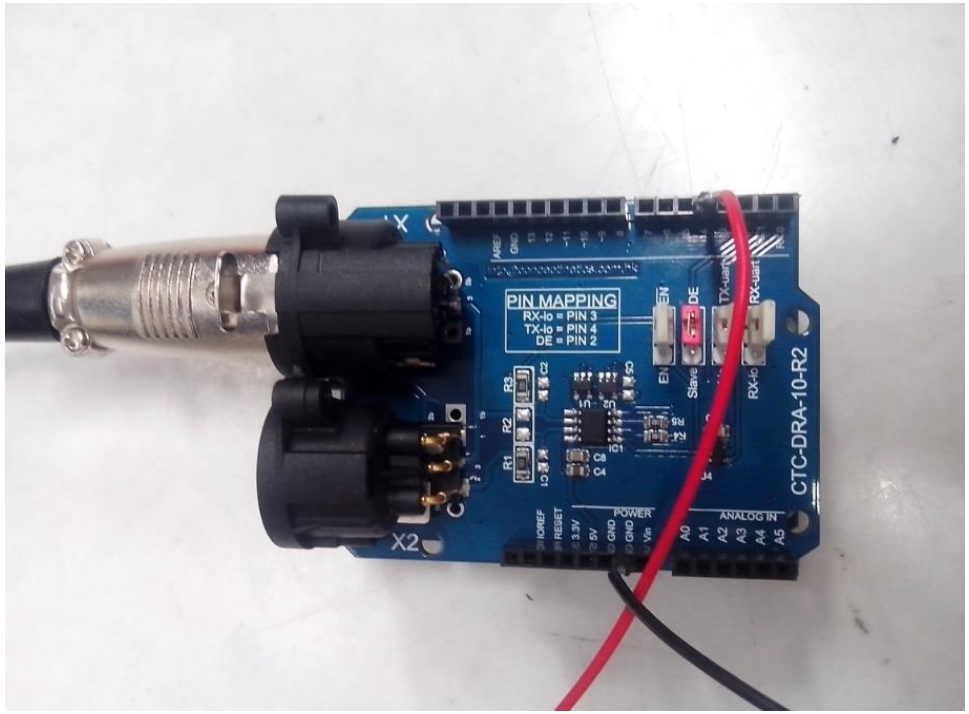
```
import serial
import time
import sys
from gpiozero import LED

class DAQ():

def __init__(self, device):
                self.port = serial.Serial(port=device, baudrate=115200, bytesize=8,
                                          parity='N', stopbits=1, timeout=0.5, xonxoff=True)
                time.sleep(0.5)
                self.write("ST 0", 0.1)
                self.write("VE 0", 0.1)

def read(self):
                output =  self.port.readline()
                return output[:-2]

def write(self, message, wait=0):
                self.port.write(str(message)+"\r")
                time.sleep(wait)

def set_thresholds(self, t1=200, t2=200, t3=200):
                print"\n\n"
                for channel, threshold in zip(range(3), [t1,t2,t3]):
                        print "Threshold for channel %s: %smV" % (channel, threshold)
                        self.write("TL %s %s" % (channel, threshold))

def set_trigger(self, trigger=3):
                self.write("WC 00 27")
                if trigger == 3:
                                self.write("WC 00 27")
                elif trigger == 2:
                                self.write("WC 00 1F")
                else:
                                print "Trigger nust be 2 or 3"
                                sys.exit(0)

def set_coincidencegate(self):
                 self.write("CD",0.1)
                 self.write("WC 01 00",0.1)
                 self.write("WC 02 FF",0.1)
                 self.write("WC 03 FF",0.1)
                 self.write("CE",0.1) # Set coincidence window to 01*24ns
                 self.write ("WT 01 00")
                 self.write ("WT 02 32")
```



```
def measure(self, runtime):
                print "\nStart measurement ... (%ss measurement time)" % runtime
                self.write("CD", 0.1)
                self.write("RB", 0.1) # Reset counter

                start, counter, t, cycle,  = 0, 0, 0, 0
                outputlist = [0,0,0,0,0,0]
                ledlist = [LED(2),LED(3),LED(17)]
                coincled = LED(18)
                coincflash = LED(15)

                a,b = 1,0
                if runtime == 0:
                            a = 0

                while (t<runtime or a==b):
                                self.write("DS")
                                output =  self.read()
                                while not (output.startswith("DS") and len(output) > 5):
                                                time.sleep(0.01)
                                                output = self.read()
                                else:
                                  if output.startswith("DS") and len(output) > 5:
                                      f_outputlist = outputlist
                                      outputlist = [int(f[3:],16) for f in output.split(" ") if len(f)>3]
                                      l_outputlist = outputlist
                                      measuret = t


                                      internal_count = outputlist[-1]
                                      if internal_count<counter:
                                                cycle +=1
                                                l_outputlist = f_outputlist
                                      if start == 0:
                                                start = internal_count - 1
                                      counter = internal_count
                                      total_count = internal_count - start + cycle * 4294967296

                                      for i in range(0,3):
                                                if l_outputlist [i] - f_outputlist[i] != 0:
                                                                ledlist[i].on()
                                                                time.sleep(0.01)
                                                                ledlist[i].off()

                                                if l_outputlist [4] - f_outputlist[4] != 0:
                                                                coincled.on()
                                                                 time.sleep(0.01)
                                                                coincled.off()
                                                t = total_count * 0.000000040

                self.write("CE", 0.1)
                return measuret, f_outputlist
```

ARDUINO

Arduino is used for controlling strobe light via DMX shield.

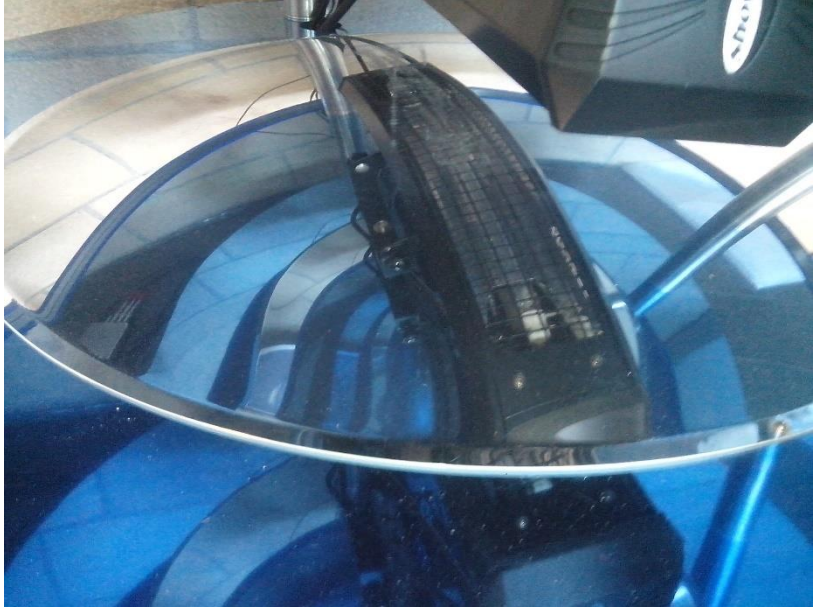


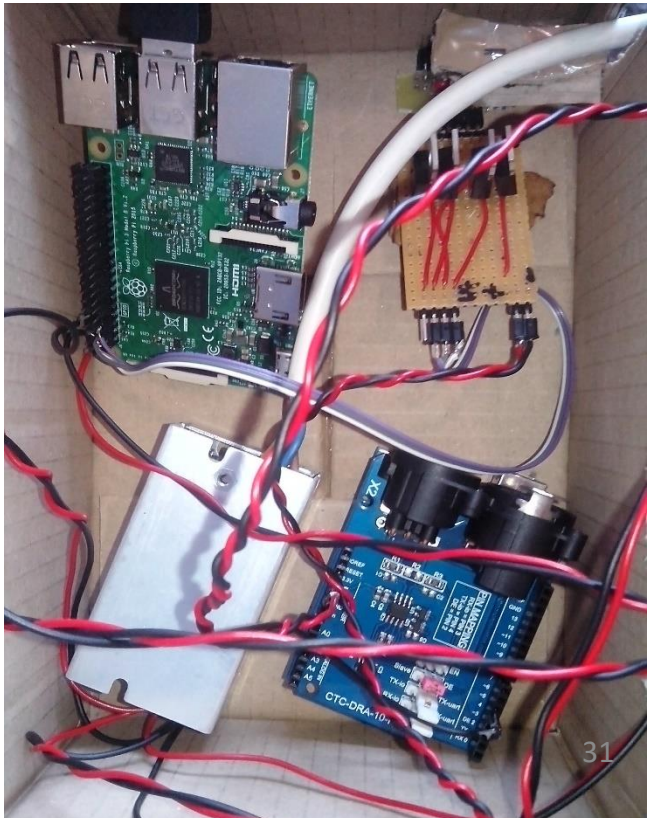
It is connected to Raspberry «coincidence» pin.

ARDUINO - code

Arduino code is basically a for-loop that switches on strobe light if input pin (driven by Raspberry) is HIGH.

It has also been provided a control structure that avoids to trigger twice the same HIGH input.






KIT
KCETA
KIT-Zentrum Elementarteilchen- und Astroteilchenphysik
UNSERE FORSCHUNGSGEBIETE


KIT
KCETA
KIT-Zentrum Elementarteilchen- und Astroteilchenphysik
UNSERE FORSCHUNGSGEBIETE
GRAZIE
PER
L'ATTENZIONE