****

**Improvement of a particle detector prototype used for outreach purposes**

**Proposed by:** Florine Valcher

Master’s degree student in Physics Engineering at Polytech Clermont-Ferrand

**Directed by:** Dr. Veronica Bindi

Associate Professor at the Physics Department of the University of Hawaii at Mānoa

**Principal:** University of Hawaii at Mānoa

Physics and Astronomy Department

2505 Correa Road

96822, Honolulu, HI

United States of America

# I. Getting started with the particle detector

This part will be the subject of a state of the art, in which we will detail the existing elements at the beginning of this project. Some theoretical points about scintillators and photomultipliers will be detailed thereafter.

## State of the art

This state of the art will be focused on the work done before I arrived in the laboratory. My colleagues Giorgia and Matilde had to build a muon detector. At the end of their project, they did not know exactly what kind of particles they detected, that is why the detector has been called particle detector. When I arrived, the detector was already in working order. It is composed by a Raspberry Pi (model 3), a breadboard on which we can connect LEDs. LEDs will serve to see if the detector is working well. The detector is also composed by a GPS sensor (Global Positioning System), and the detection system is a simple webcam.

The main component of the detector is the Raspberry Pi. It is a printed card on which we can find a lot of different modules, as we can see on the Figure 1. It is often used in high schools to teach basic computer science skills.

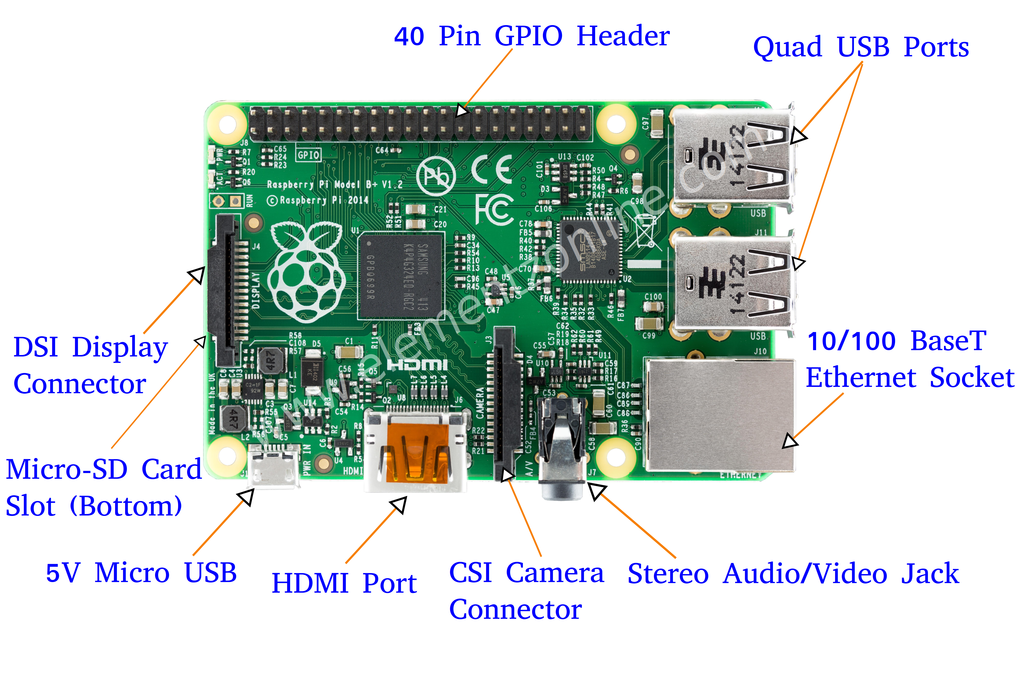


Figure 1: Different modules of the Raspberry Pi

In this project, we will not use all the modules of the Raspberry Pi. Indeed, we will first of all use the Micro-SD card slot, which allows us to install the Operating System (OS) in the Raspberry Pi system. For this project, the OS installed is Raspian Jessie. Then, we use USB and HDMI port to connect the webcam and the monitor. The GPIO Header allows us to connect the Raspberry Pi to the sensors we want to add.

To detect particles, the first choice was to use a simple and inexpensive webcam (Logitech C270, cf. Figure 2). Indeed, we use the Complementary Metal Oxide Semi-conductor sensor (CMOS) of the webcam as a particle detector. This sensor is really sensitive. We want to prevent the light from touching the sensor, because we would like to detect particles, and not visible light. That is why the lens is removed, and the sensor covered with aluminum foil and black tape. By doing this, we are sure we will not detect particles coming from the visible light, but only charged particles. When a cosmic ray enters in the CMOS sensor, it deposits a charge onto the sensor itself and leaves a trace. Thanks to this trace, we will be able to display this with an appropriate code.



Figure 3: Obscured webcam

Figure 2: Logitech C270

The GPS module we used is connected to the Raspberry Pi thanks to a USB port (cf. Figure 3). The antenna of this GPS is not enough performing to acquire satellites data inside a building. That is why, we can replace this GPS with a more performing one in order to improve the detector.



Figure 4: GPS module



We use a radioactive source for our tries. Here we use Cobalt 60.This allows us to check if the detector is working correctly. Indeed, the webcam sensor is not big enough, so it is pretty difficult to detect particles without a radioactive source. I led experiment without it, and this is really random. Indeed, during a 7-hour experiment, I detected 7 particles (July 12th). During a 1-hour experiment two weeks later, I also detected 7 particles (July 27th).

Figure 5: Cobalt 60, radioactive source

The OS we installed on the Raspberry Pi allows us to use the Terminal in Linux in order to install all requirements needed to make the detector working.

After taking notes of the components of this detector, I was able to build mine in order to acquaint oneself with the Raspberry Pi interface. You can see on the Figure below what it looks like.

Figure 6: Particle detector

LEDs + switch button

Ribbon cable

Radioactive source

GPS module

Webcam

As you can see on this Figure, the Raspberry Pi is connected to the breadboard thanks to a ribbon cable. This cable is composed by a lot of wires and each wire is connected to a pin. You can see that the webcam and the GPS module are connected thanks to a USB cable. We also use USB cable to connect a monitor, a mouse and a keyboard, in order to take control of the Raspberry Pi, and to be able to use all functionalities of the OS.

After building the detector, I followed the “How To” tutorial to install all the requirements of each components. After having installed these requirements, I took notes about the Python code my colleagues did, to make the detector working.

To show the particles we detect, the webcam takes a first frame, and we apply a threshold to this frame. Each pixel which has a smaller value than 20 will be considered as a black pixel, and its value will take 0. Each pixel which has a higher value than 20 will be considered as a white pixel, and will take the value 255. After some tests, they found that 20 is low enough to define the black background and the lighter pixels of a particle, but it is also high enough to remove the noise from the CMOS sensor. This first frame is then kept in memory by the code. The webcam takes another frame, we apply the same threshold, and sums the second frame with the first one. We apply this sum until the end of the experiment. So, each particle we detect will appear as a white pixel on the image, as you can see below (cf. Figure 7).

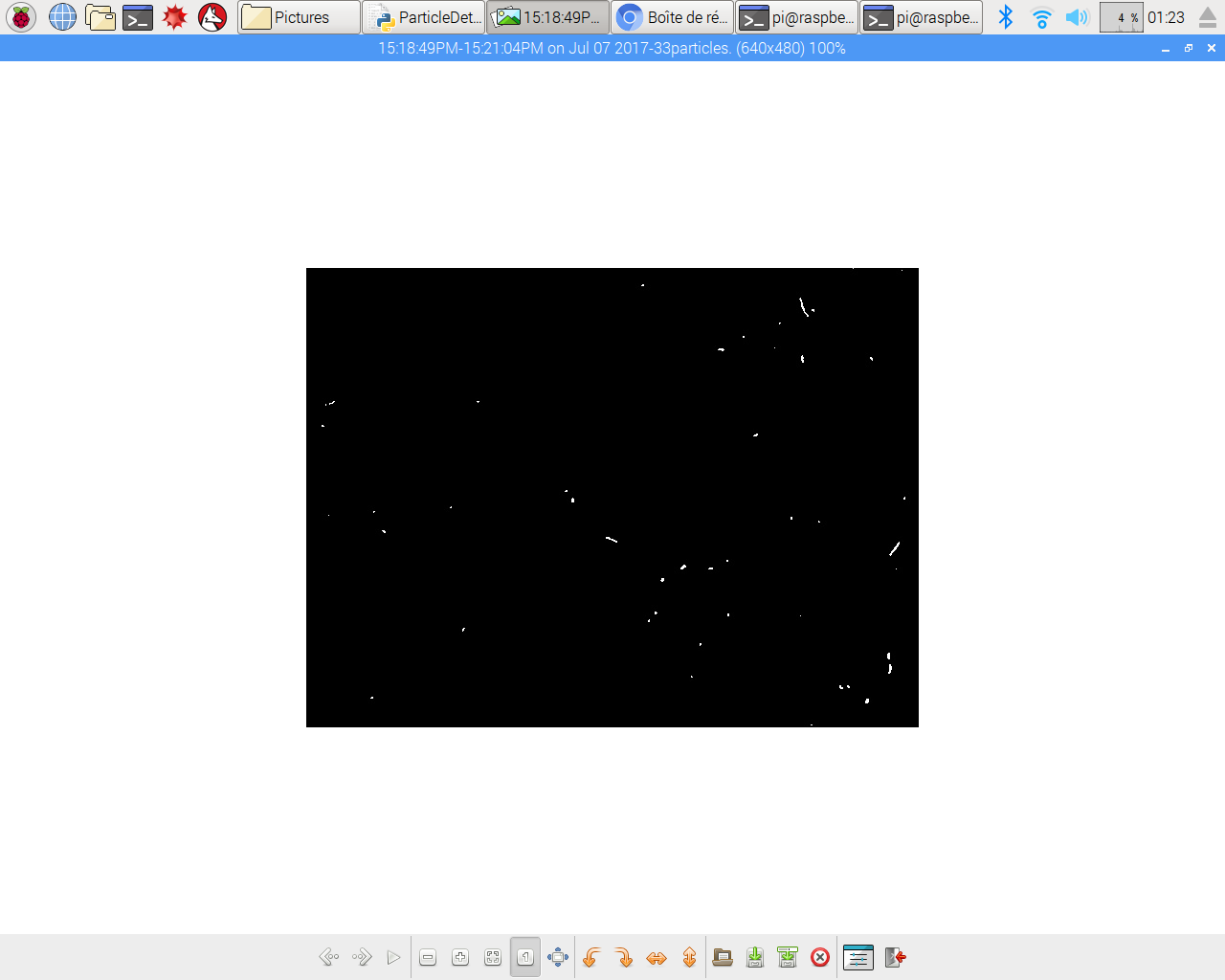


Figure 7: During the experiment, particle detection

On this figure, I took during an experiment, we can see white spots on a black background. With the explanation I gave before, we know that each white spot is a particle which touched the CMOS sensor of the webcam.

After building my own detector and doing some experiments, I took note of my requirements specification.

## Different improvements of the detector

The main goal of the project was improve this detector, to develop it, always with simple and inexpensive components, bearing in mind that it must be as portable as possible. After seeing the functional prototype, as described in the state of the art, axes of improvement have emerged.

Indeed, the first idea was to emit an audible signal each time an event happens, using a buzzer. Subsequently, the existing GPS could be replaced so that satellite data can be acquired inside a building. The addition of a temperature sensor, but also a pressure sensor can be interesting for outside experiments, allowing us to get additional information, which may be interesting for the user. In order to facilitate the interaction with the detector, we could add a display screen which returns the number of events, and some other parameters if desired, such as temperature or pressure.

In order to detect more events, the existing detection system could be changed. Indeed, the webcam used in the first prototype could be replaced with a system containing a scintillator and a silicon photomultiplier. In the next part, I will explain how these two components work.

## Theory about scintillators and photomultipliers (si trop long, posé en annexe)

You cannot use the scintillator without the photomultiplier, and vice versa. Indeed, the scintillator allows you to convert ionizing rays into light, and the photomultiplier allows to convert the light in an electric signal. Let us see how both work more precisely.

There are two main families of scintillators: inorganic scintillators, and organic scintillators. In the inorganic scintillators family, you can find crystal scintillators. Inorganic scintillators emit a lot of light, but have a low rapidity. On the other hand, the organic scintillators are composed by plastic scintillators, which allow to deliver less light than inorganic ones, but are quicker and inexpensive compared to the other family [1]. As we would like to build an inexpensive detector, we will choose this kind of scintillator.

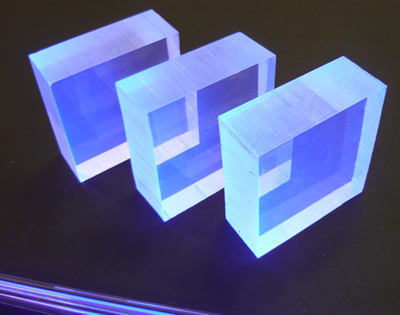
Scintillators emit light (cf. Figure 8) when a charged particle passes through them and deposits a fraction of its initial energy due to electromagnetic interactions. The amount of emitted light is related to the energy of the incident particle and the distance the particle traverses through the scintillator. In this case, the scintillator responds to this energy because the plastic is doped with a fluorescing agent that glows very slightly when some kinetic energy is transferred to the fluorescing molecules. Within nanoseconds, the de-excitation of these fluorescing molecules produces visible light, typically in the 300 to 600 nm wavelength range, that travels through and exits the scintillator [2].

Figure 8: Scintillators emitting light

When the light is emitted, it must be observed using a light detector. We can use photomultiplier tubes (PMTs) or Geiger tubes, but these are too expensive, and bigger than the photomultiplier. That is why we will use a Silicon Photomultiplier (SiPM). A SiPM consists of a large number of microcells, each composed of silicon P-N junctions. Electrons migrate into the P-side and holes migrate into the N-side. This creates a region known as the “depletion region", where the electrons and holes eliminate through recombination. When a photon traverses the depleted region, it can deposit sufficient energy to an electron in the valence band to move it to the conduction band, thereby creating a current. Biasing the P-N junction increases the depletion region and creates an electric field. When a charge carrier accelerates through this field, it can gain sufficient kinetic energy to ionize the surrounding atoms through impact ionization. This creates an avalanche of electron-hole pairs, which can have a high gain [3].

# II. Installation of the various components

As we have seen above, the particle detector built is a prototype which has to be improved, including the addition of new sensors, and changing the exchange interface. This part will be focused on the requirements specification given at the beginning of the project, and will also explain how the components selection was done.

## Components selection

At the beginning of my project, I had a research phase which took me two entire weeks. During this phase, I took notes of the different components I will have to handle. Each component has to be carefully selected regarding the different constraints imposed by the prototype, and by the University. Indeed, the prototype has to be as portable as possible, and inexpensive. The total amount of the order should not exceed $200, and each part had to come from the USA.

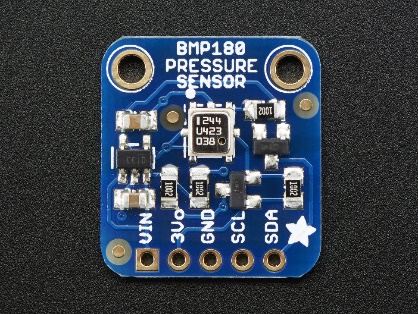
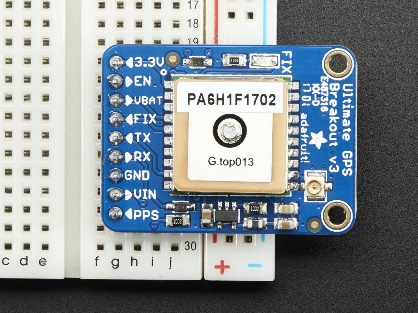
First of all, my research was primarily focused on the temperature sensor, and on the pressure sensor. During my research, I found a component called BMP180, coming from the Bosch company, which acts like a barometer. Indeed, this sensor allows you to get the pressure, the altitude, and the temperature during the experiment. The temperature and the pressure ranges were sufficient and the accuracy was really interesting (temperature range, accuracy) [4]. I chose this sensor thanks to its reliability, and also because in order to make the detector as portable as possible, it seemed really interesting to get two sensors in just one.

Figure 9: BMP180

Then, I looked for a more powerful GPS module. I found one called Ultimate GPS Breakout, coming from Adafruit Industries. This GPS has a big antenna which allows you to get satellites data while you are located inside a building, and has a 1.8 meter accuracy [5]. The antenna was really interesting, that is why I chose this module.

Figure 10: Ultimate GPS Breakout

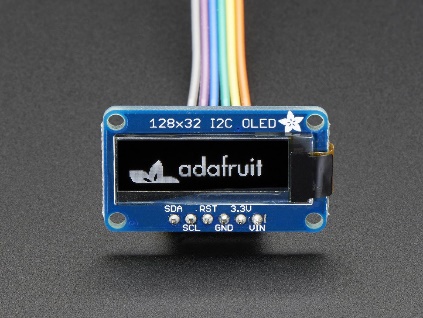
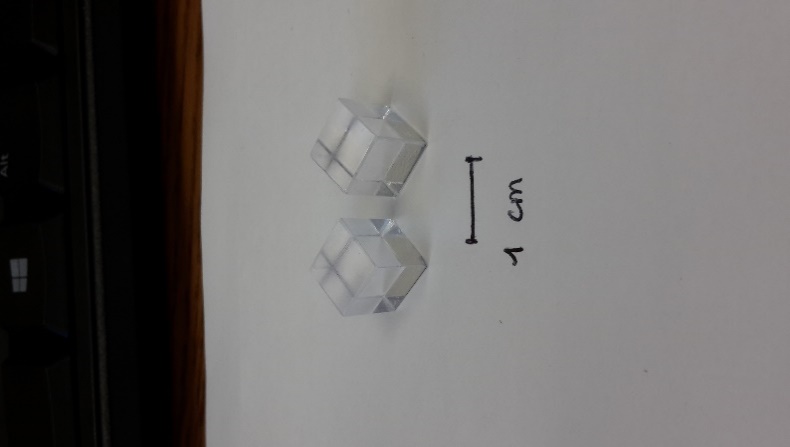
I also had to add a display screen. This screen will allow us to read data in an easier way than in the first version of the prototype. Indeed, we would like to display the number of particles detected in real time, a timer and may be the pressure or temperature values at each event. At the beginning of the project, I thought we just wanted to display the number of particles,that is why I chose a little display screen. Indeed, I chose a monochrome display screen (128x32 pixels) coming from Adafruit Industries. It seemed easy to use with a Raspberry Pi, that is why I decided to choose this one.

Figure  : Monochrome 128x32 display screen



I chose to work with a plastic scintillator because this kind of scintillator is quicker than the crystal one. Also, they are less expensive and it should be sufficient for the goal of this project. However, we had a problem with the order, so I finally found one in the Department. I cut two pieces in the main block, with the dimensions (9x9x10)mm, and then I polished it (cf. Figure 12).

Figure 2: Plastic scintillators

Concerning the SiPM, I chose one which already had pins to connect to the breadboard, easier to use. I also looked at the active area. Indeed, more the area is important, more the price of the component will be high. I finally chose the component called MicroFC-SMTPA-10035 (cf. Figure 13). The active area is really small: 1x1mm². This is a really small piece so, that is why I cut a little piece of scintillator.

Figure 13: MicroFC-SMTPA-10035

For all these parts, I made a part list with all the prices and the datasheets. After choosing these components, I had to install them on the existing prototype. You can find comments about the installation in the next part.

## Comments about the installation

The installation of each component was easy thanks to tutorials I found on the Internet. Thanks to the pictures below, you can easily imagine to what it looks like.

|  |  |  |
| --- | --- | --- |
| buzzer  Figure 4: Buzzer installation | raspberry_pi_UltimateGPSUART_bb.png  Figure 5: GPS module installation | raspberry_pi_BMP085_Breadboard_1K.png  Figure16: BMP180 installation |

On the Figure 14 above, we can see that the buzzer installation is really simple. Indeed, we just need to connect the positive pin to 5V and the other to the ground (GND) [6]. The GPS module needs more wires to be connected (cf. Figure 15). Indeed, we need to connect the TX and RX pins to RXD and TXD, respectively. These ports allow the GPS module to transmit data to Raspberry Pi (TXD or TX) and to receive data from satellites (RXD or RX). We also have to connect the voltage Vin to 5V and the GND pin to the ground [7]. We need the same amount of wires to connect the BMP180 module, which is the temperature and pressure sensor. As you can see on the Figure 16, we will not connect the Vin pin to 5V, but to 3V3 in order to protect the sensor. We also have to connect the SDA and the SCL ports. These ports are characteristic of the I2C method. I2C means Inter-Integrated Circuit. It is typically used for attaching lower-speed peripheral ICs to processors and microcontrollers in short-distance, intra-board communication. For this, we have 2 new parameters, called SDA and SCL. SDA is used for the bi-direction data, and SCL represents the clock signal [8].

The display screen works with the I2C bus, like the pressure and temperature sensor (BMP180), as seen before. We have to connect the SDA and SCL pins to those of the Raspberry Pi. We also have to connect the ground to GND pin and the Vin pin to 5V or 3V3. There is a new pin, called RST (Reset), which can be connected to a pin with a number. I personnally chose to connect RST pin to pin 24, because the library we use to run the display screen uses pin 24 [9].

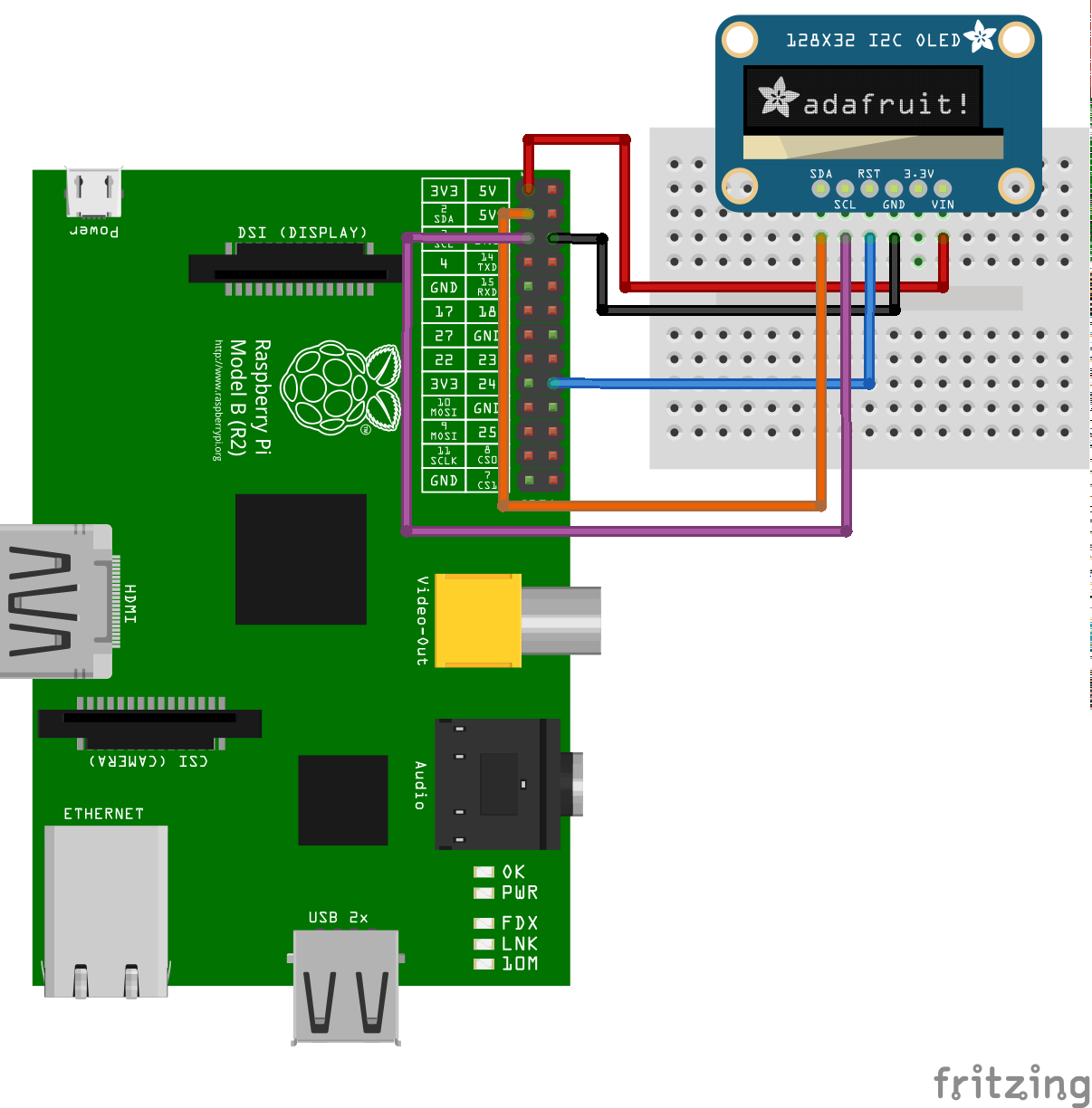


Figure 17: Display screen installation

To install the SiPM on the breadboard, I found the Figure 18 [10], showing how to connect the different pins to the breadboard. First of all, I wrapped the system SiPM + Scintillator with aluminum foil and black tape, as my colleagues did with the webcam’s CMOS sensor. You can see the system wrapped on the Figure 19 below. Wrapping the system allows to prevent the visible light from crossing the scintillator and touching the active area of the SiPM. We only want to detect cosmic rays.

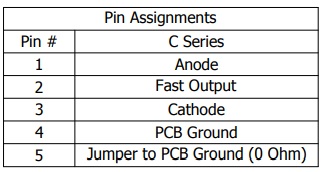


Figure 19: SiPM+ Scintillator wrapped

Figure 18: SiPM pin assignments

As you can see, there are five pins on the SiPM that we needed to connect to the breadboard. I connected the component to the breadboard, but due to a lack of time, I was not able to finish this part of the project. You can see on the Figure 19 the wrapped component, which is really small. To wrap it, I put a layer of aluminum foil on the scintillator, and then I covered it with two layers of black tape.

After having installed the components on the breadboard, I had to program all of these to make the detector working.

## Components programming

In this part, I will explain how I programmed each component. I have to say that I did not write all the Python code. Indeed, my colleagues Giorgia and Matilde Mazzini already wrote the main part of the code. When I arrived, the code was able to run the webcam and the previous GPS module. It was also able to detect the particles on the final frame. I wrote new lines to make my components working.

The first component I installed and I programmed was the buzzer. The buzzer has to emit a sound each time a particle is detected. To do that, I had to understand how the previous version of the code worked. Indeed, as I wrote before, the program takes a first frame, applies a threshold. Then, it takes a new frame, and applies the same threshold. A sum of these two frames is made just after and the result is saved on the first frame. We always display the first frame during the experiment. At the end, all the frames are saved on the same image and that is why we can see the evolution of the experiment in real time. I had to focus on the new frame the program takes and detect if new points are detected. Then, once I understood this, it was simple to write the code.

The second component I installed and programmed was the GPS module from Adafruit. As I chose to connect the component with the UART method, I needed to change the port we used in the previous code. Indeed, the previous GPS module was connected to the breadboard thanks to a USB cable. So, we were using the USB port. Now, we need to use the serial port. On the Raspberry Pi, the serial port is often called **ttyS0**, instead of **ttyUSB0**. I just needed to change this point in the code, at two different places.

Then, I add the temperature and pressure sensor called BMP180. To try the component, I ran different examples from the library I previously installed: Adafruit\_BMP. In one of these examples, I found different lines that allowed me to get the temperature and the pressure. I added these lines in my program, and I made a loop which allows to display in the Terminal the pressure and the temperature at each event, and also on the display screen.

Then, I installed the display screen as we saw before. I installed the library called Adafruit\_SSD1306 which is the reference of the screen. There are different examples in this library, like the pressure and temperature sensor. I ran these examples to see if everything was working fine, and I read how these examples had been programmed. Then, I was able to add few lines in my main code to allow the display screen to return the number of events in real time, the pressure and the temperature at each event.

I found this part easier than I expected because a lot of information were available thanks to the libraries I installed. I just had to think about how to modify and implement the examples in my code.

I presented in this part how I installed the different modules I needed to improve the detector, so now let us see the results and the problems encountered during this 3-month internship.

# III. Presentation of the results and discussion about the problems encountered

In this part, you will find the results of this project and the problems I encountered.

## Presentation of the results

To check if all the components are working well, I had to run the code I improved, called ParticleDetectorImproved.py. I have to say that I improved the code step by step, component by component.

Thanks to that, the buzzer is now emitting a brief sound when particles are detected. I also wanted to improve the exchange with the prototype. The experiment starts itself when the GPS detected satellites. Otherwise, we have to push a button to force the program to start. If we have to push this button, so the buzzer beeps once. To stop the detection, we have to push this button again until the buzzer emits two beeps. It is more convenient for the user to hear sounds from the detector. Also, middle school students could find this friendlier to use. I also improved the display in the Linux Terminal. Indeed, now each event is displayed with the date and the time of the detection. We also can see the temperature and the pressure of each detection. You can see what it looks like below (cf. Figure 20)

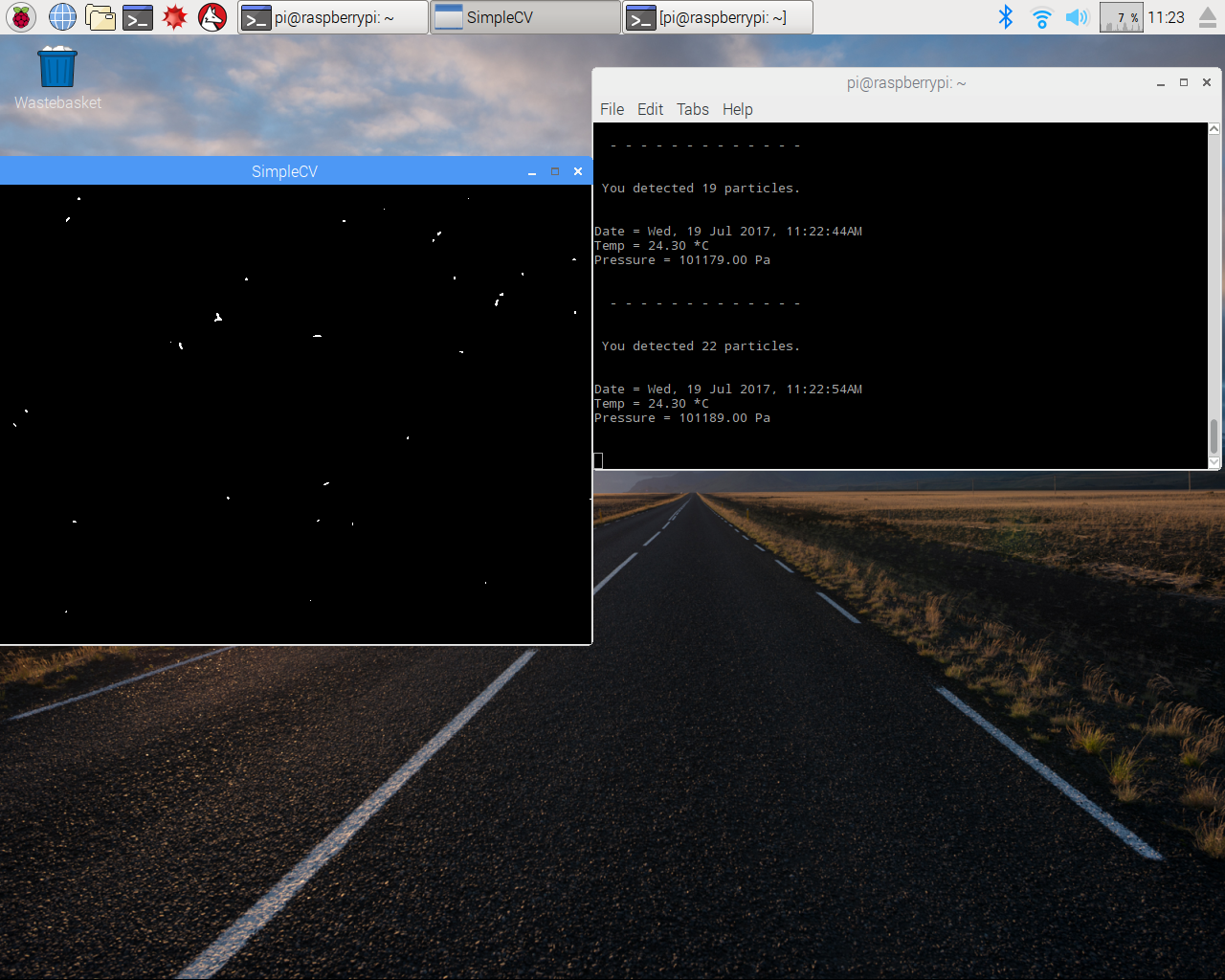
****

Figure 206: Display in the Linux Terminal

As you can see on this figure above, you know exactly how many particles you detected from the beginning of the experiment, and at what time. This can be very useful for the user if he wants to do a report on his experiments. When a new event happens, you can detect more than one particle. During the figure’s experiment, you can see that we went from 19 to 22 particles. It means on the frame we always display in real time, we should have at least 22 white spots. Indeed, we should have at least 22 spots because each white point has to have a certain size to be considered as a particle. This is an arbitrary value. For these experiments, we decided that each particle which has a size equal or superior to one pixel will be considered as a particle. You can see below, on the Figure 21, the display of the particles detected during this experiment.

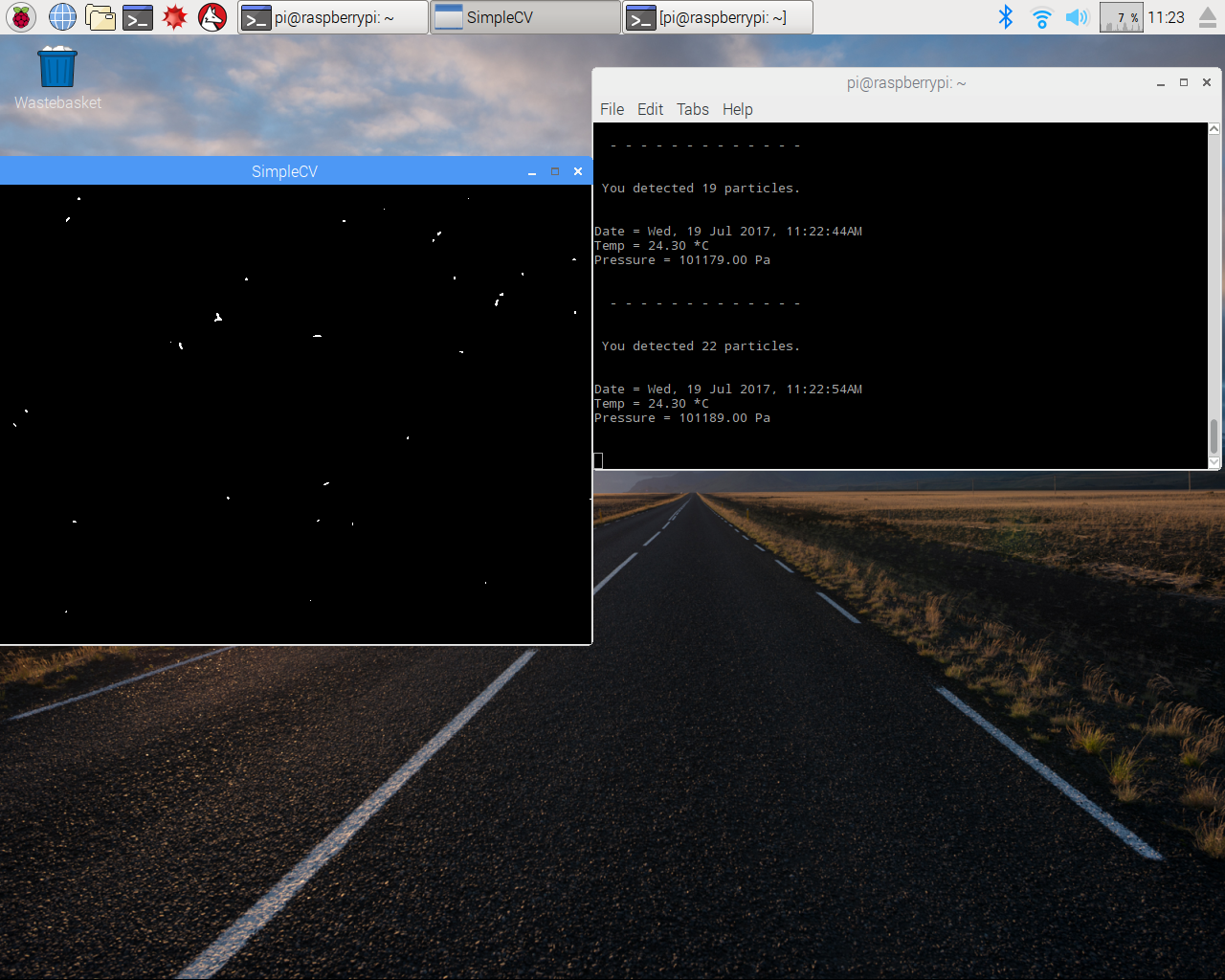


Figure 71: Display of the particles detected

Thanks to the sum of the frames, we are able to see in real time the evolution of the detection. We can also read the altitude of the experiment, know how many satellites we were connected to, thanks to the GPS module, when the program stops the detection (cf. Figure 22).

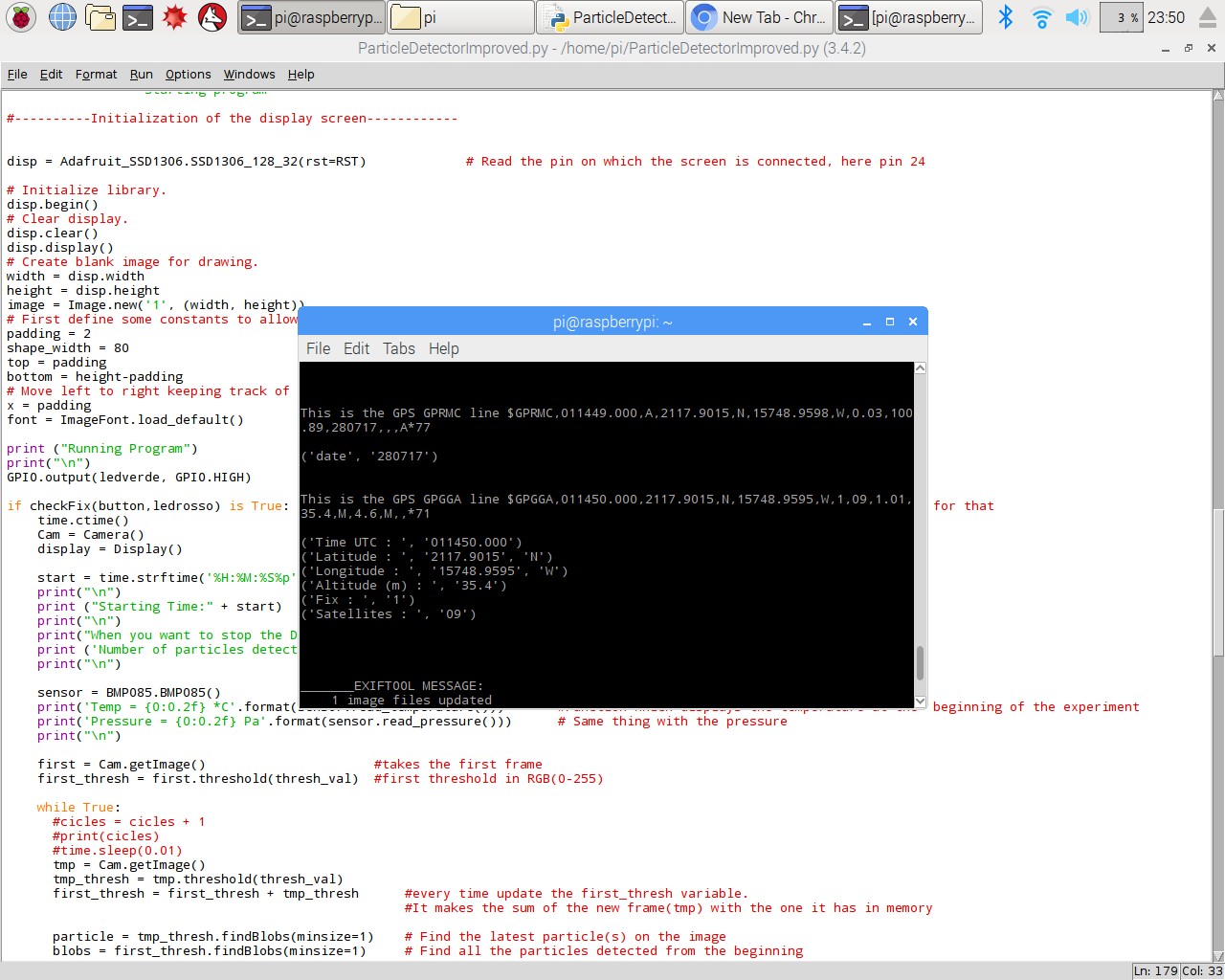


Figure 22: Display of the GPS data

On this figure, you can see the details of the GPS data. Thanks to the program, we are able to determine and display the longitude, latitude, altitude of the experiment, you can also see the number of satellites you were connected during the experiment. The display screen allows us to display some information for the user. Indeed, we are able to display the number of particles detected in real time, the temperature and the pressure of each event. You can see below on the Figure 23 what it looks like.

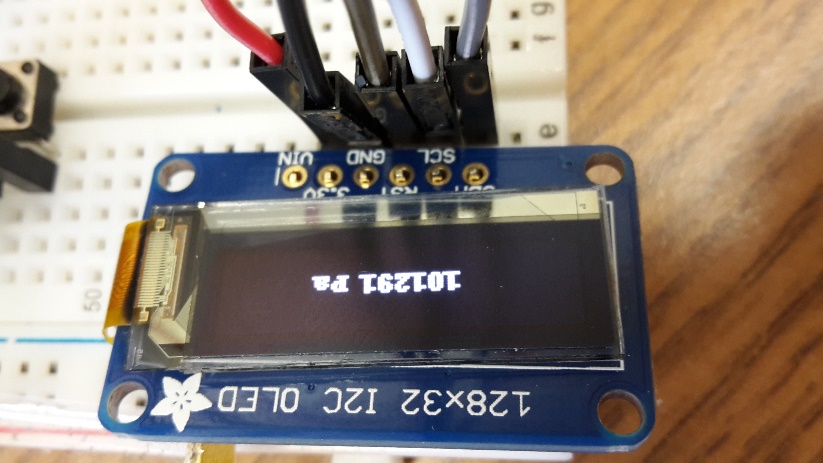
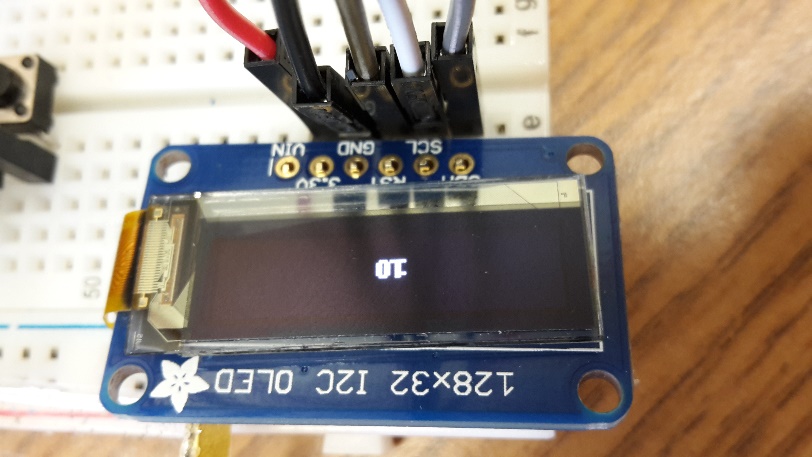
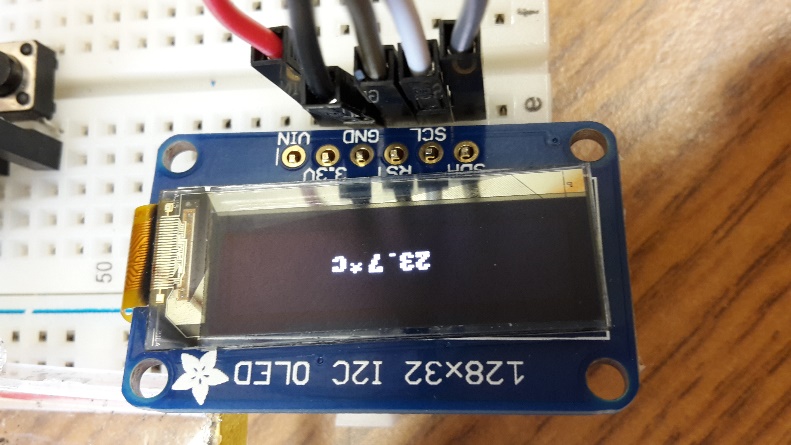


Figure 23: Data on the display screen

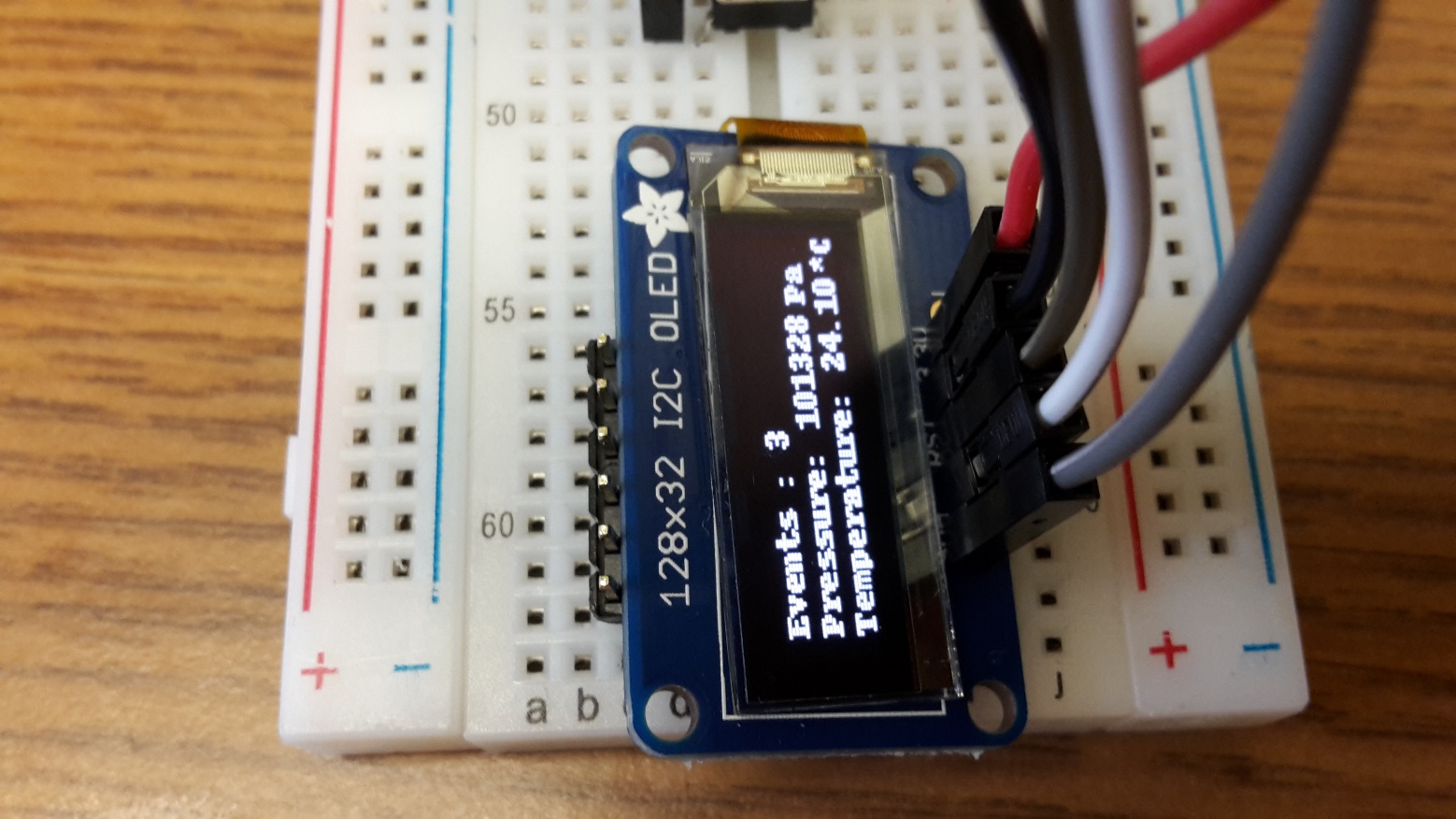
I tried to different ways to display these information. The first one was to display one by one all the information. Indeed, when an event happened, you could read on the display the total number of particles detected during 2 seconds, then the temperature appeared for 2 seconds, and finally the pressure was displayed. This was a problem because during 6 seconds the display screen is working, and if a new event happens during this, the display screen does not consider it. That is why we decided to display all the information together, as you can see below on the Figure 24.

Figure 24:Display of the data

## 

This display allows the user to get all the data without waiting during two seconds. The major drawback is the font which is really small.

## Problems encountered

In this part, I will explain the problems I encountered and how I resolved them. The first problem I encountered was with the buzzer programming. At this time, I did not really know the Python code and I struggled to find how to follow the particle detection in real time. It took me an entire week to figure it out. As I explained in the second part of this report, I had to create a new variable with the threshold we applied on the new frame we took. We apply then a function which allows us to know if particles are detected. If we did detect particles, the buzzer has to emit sound. I created a loop for that, as you can see in the code.

The second problem I encountered was the connection between certain components and the breadboard. Indeed, the pins should have been solder on the components to allow a better connection. As I did not find a soldering iron, I continued without soldering and took time before experiments to check if all my components were connected to the breadboard. The components which have to be soldered are the temperature and pressure sensor (BMP180) and the display screen.

Also, it took a lot of time to order the SiPM and find a scintillator. Indeed, we place an order early July for the SiPM. The company we contacted for the scintillator could not ship the item until the end of July. So, I needed to find one in the lab and to cut and polish it. I had only few days to work on the detection system, and I did not have any progress during this time.

# Conclusion

I learnt a lot during this internship. Indeed, I was not really familiar with Python and electronics components. I had few problems and I was able to fix them just by taking the time to go step by step, and to think in a simpler way. The installation of all the components was easy thanks to the information I found on the Internet. The programming was a bit difficult at the beginning because I did not really understand the code that Giorgia and Matilde did. When I took time to study this code, everything went better. Unfortunately, I did not have time to install the new detection system and to add a timer on the display screen.