Part I
Development environment

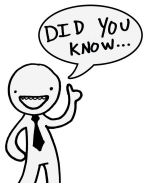
1 Foreword

Computing sessions belong to the educational program of the ESIPAP (European School in Instrumentation for Particle and Astroparticle Physics). Their goal is to become familiar with C++ programming through practical work in the context of high energy physics. The session is designed to be pedagogical. It is advised to read this document section-by-section. Indeed, each section of the document is a milestone allowing to acquire computing skills and to validate them. The sections related to C++ programming are ranked in terms of complexity. The student is invited to **fill a report (.doc)** which includes notes related to questions that will be asked during the computing sessions.

In the document, some graphical tags are used for highlighting some particular points. The list of tags and their description are given below.

The student is invited to perform a practical work by **writing a piece of code** following some instructions.

Analyzing or interpreting task is requested and the results must be reported in the report.



Some **additional information** is provided for extending the main explanations. It is devoted to curious students.

A piece of **advice** is given to help the student in his task.

2 The Raspberry framework

The present computing session must be performed on a specific setup based on a single-board machine, the Raspberry Pi 3 Model B+ released in 2018, and an add-on board containing several sensors, the Sense Hat board. The user will find below all the instructions for handling this particular framework and being ready to develop code.

2.1 Checking the hardware connections

The Raspberry board needs to be linked to other peripheral devices for running properly. The following photography show the different items of the setup:
Please check the connections between the Raspberry board and:

- the screen via a HDMI cable,
- the mouse via a USB cable,
- the keyboard via a USB cable,
- the power supply cable,
- the Sense Hat board via the Ribbon cable.

Finally check that a micro SD card is also inserted in the Raspberry board.

2.2 Booting the Raspberry

For switching on the Raspberry board, the user has just to plug the power supply. The OS (Operator System) contained in the SD card will be executed and an initialization screen must be displayed at the screen. During this phase of initialization, the pixels of the Sense Hat board show rainbow colors.

If you do not managed to have the booting sequence described previously, please warn the supervisors.

2.3 Opening a Linux console

The OS (Operator System) is a Linux distribution called Raspbian which is very similar to Debian in the PC world. According to the Figure 2, you can open a new console by clicking on the third icon of the task bar. It's the perfect environment for developing with Raspberry. All required packages have been *a priori* installed by the supervisors.

If you notice that when you type with your keyboard it displays wrong letters (AZERTY/QWERTY issue), you should configure your keyboard by clicking on the last icon on the right of the task bar.

Good luck with during your computing session!!!

Figure 1: The `Windows` Start button

3 The `Raspberry` framework

The present computing session must be performed on a specific setup based on a single-board machine, the `Raspberry Pi` 3 Model B+ released in 2018, and an add-on board containing several sensors, the `Sense Hat` board. The user will find below all the instructions for handling this particular framework and being ready to develop code.

3.1 Checking the hardware connections

The `Raspberry` board needs to be linked to other peripheral devices for running properly. Photography 2 show the different items of the setup:

Figure 2: Connections of the Raspberry board

Please check the connections between the `Raspberry` board and:

- the screen via a HDMI cable,

- the mouse via a USB cable,
- the keyboard via a USB cable,
- the internet network via an Ethernet cable,
- the power supply cable,
- the `Sense Hat` board via the Ribbon cable.

Finally check that a micro SD card is also inserted in the `Raspberry` board.

3.2 Booting the Raspberry

For switching on the `Raspberry` board, the user has just to plug the power supply. The OS (Operator System) contained in the SD card will be executed and an initialization screen must be displayed at the screen. During this phase of initialization, the pixels of the `Sense Hat` board show rainbow colors.

If you do not managed to have the booting sequence described previously, please warn the supervisors.

3.3 Opening a Linux console

The OS (Operator System) is a `Linux` distribution called `Raspbian` which is very similar to `Debian` in the PC world. According to the Figure 3, you can open a new console by clicking on the fourth icon from the left of the task bar. It is the perfect environment for developing with `Raspberry`. All required packages have been *a priori* installed by the supervisors.

Figure 3: The task bar of a ***`Raspbian`*** *session*

If you notice that when you type with your keyboard it displays wrong letters (AZERTY/QWERTY issue), you should configure your keyboard by clicking on the last icon on the right (English, US or French flag) of the task bar.

3.4 Setting the environment

To load the work environment, you can issue the command below at the shell prompt.

```
bash$source␣/home/pi/tools/setup.sh
```

If the system is properly installed, the version of each tool to study should be displayed at the screen like below. If you have an error, please call the supervisors.

```
-----------------------------------------------
              ESIPAP environment
-----------------------------------------------
```

```
 - GNU g++  version 4.9.1
 - ROOT     version 6.06/00
 - Geant4   version 10.2.0
-----------------------------------------------
```

Good luck in your computing session!!!

Part II
Data acquisition with the Sense Hat board

4 Hello world!

In this section, a very simple example of main program (the so-called *hello world* example) is supplied and explained in order to help beginners in C++ programming. More experimented students should be a little patient: challenging tasks are coming soon.

4.1 Main program source

The main program will be contained in a source file called `main.cpp`. It displays at the screen the message *"Hello World!"*.

```
// STL headers
#include <iostream>
using namespace std;

// Main program
int main(int argc, char** argv)
{
  // Display messages at screen
  cout << "Hello World!" << endl;

  // Normal program termination
  return 0;
}
```

Listing 1: A first main program

4.2 Building the main program

To build with `g++` compiler an executable file from `main.cpp` file, the simpliest command to type at the shell prompt is:

```
bash$g++ main.cpp
```

If the `main.cpp` file compiles properly, an executable file with the default name `a.out` is created. Of course, we invite the students to use `g++` in more advanced way by adding three items:

- giving a proper name to the executable program
- splitting the compilation step from the linking step
- specifying some compilation options

To avoid retyping several times during the session the building commands, a shell script can be written. This is an example of a such file called `mymake`:

```
g++ -W -Wall -ansi -pedantic -o main.o -c main.cpp
g++ -o main main.o
```

Listing 2: A first building script

It is necessary to make this script executable before launching it.

```
bash$chmod +x mymake
```

4.3 Work to do

- **Recopy the content of the `main.cpp` and `mymake` files.**
- **Build the program and test that the *"hellow world"* message appears properly when you launch the executable.**

- **Explain the compilation options used for generating the object file `main.o`.**

5 Handling the Sense Hat board

5.1 Sense Hat board

Sense HAT is an add-on board for Raspberry Pi connected from the 40 GPIO pins and developed in the context of the Astro-Pi project. It has several integrated circuit based sensors that can be used for many different types of experiments and applications. The board is equipped with many sensors which allow to measure the pressure, the humidity, the temperature and the orientation. In order to interact with the environment, the board is equipped with a 8X8 LED matrix display and a small 5 button joystick.

5.2 Using the Sense Hat board as a weather station

In the context of this computing sessions, we will use this setup as a possible setup to monitor the environmental conditions under which a silicon sensor detector would be operated. Further, the key ingredient for us will:

- Monitoring the temperature
- Monitoring the relative humidity
- Displaying messages through the LED matrix

The C++ library RTIMULib have been developed and installed on the setup. It contains several classes useful to communicate with the Sense Hat elements. In order to access those functionalities, it will be required to link this library during the compilation.

In the goal of simplifying the access to the Sense Hat board, we have developed an new class called `SenseHat` which uses several classes contained in libRTIMUlib.so.

5.3 Handling the class SenseHat

- **Header file:** Open the header of the `SenseHat` class. Browse it and make sure you understand its structure and all the keywords it contains. In it is not the case, you can firstly refer to the lecture and in a second time, contact the teachers to go further. %item **Compilation:** You need to create an object file (.o) from `SenseHat.cpp`.
- **Demo program:** Extend the "Hello world" example by instantiating an object of the class `SenseHat` and by initializing it. In order to compile our program, you need to link our executable with the `libSenseHat.so` library. If you don't know how to proceed, you can refer to the appropriate section in the lecture. If you fail, don't hesitate to contact the teachers.

- **Compilation:** In order to "automatize" and simplify the compilation, you can add the compilation commands in a bash script: `mymake`. We will learn later how to create a `Makefile`

- **Source file:** If you want to go further, you can check the source code of the class `SenseHat` and take note about elements which may appear as unclear to you.

5.4 First acquisition program

You must write a program `FirstDAQ.cc` reading the temperature and relative humidity measured by the sensors.

- **Step 1:** Read 10 times both temperatures measured by the pressure and humidity sensors every 5 seconds and display them on the terminal (using std::cout)
- **Step 2:** Make the program more configurable by asking the user the number of measurements and the delay.

- The delay can be generated by using a method of the class `SenseHat`. It uses the UNIX command `sleep` which use delay expressed in μs.
- Use std::cin to retrieve parameters entered by the user.
- It is also possible to retrieve values through the arguments given in the command line (arguments of the function `main`)

5.5 Exporting the measurements

It is often useful to disentangle the acquisition program from the data analysis one. The goal of the acquisition program `FirstDAQExport.cc` to be written is to store data in a given numerical format. For the sake of simplicity, we will store the data in a CSV (Comma Separated Value) format. This imply that variables of a given measurement (once per line) are separated by a comma.

- **In the files `CSVExport.h` and `CSVExport.cc`, Write respectively the prototype and the implementation of a function that will store in a CSV file the following values:**
  - the time in μs since the first measurement.
  - the pressure [bar]
  - the relative humidity
  - the temperature measured by the pressure sensor [°C]
  - the temperature measured by the humidity sensor [°C]
  - both temperatures from the CPU and GPU of the Raspberry Pi [°C]
- The code should be protected for non available measurement and a default value (-9999.) should be given in case of failure, allowing offline treatment.
- **The function will take as argument the delay between measurements and the number of measurements.**
- Write the compilation instructions in a script `mymake` and compile the project
- Test if the program is working properly. The integrity of the CSV file can even be tested by loading in with `OpenOffice Calc`. Change the delimiter for csv file in the configuration to use "," and not ";".

5.6 Measurement campaigns

Measurements in various conditions can be done, allowing dedicated analyzes in the next computing sessions. In order to facilitate the execution of several runs with varying conditions, modify the main program to read from the command line the delay, the number of measurements and the output csv filename.

The list of measurements is described below in the text and in the table 1. You will find in the last column indications to know which kind of measurements you need to perform. We could use by default a delay 1.5 secondes.

- **Stable conditions:**

  As the setup is running since several minutes and as the environment conditions are stable, one can take data in stable conditions as reference. This run will be used later to inter-calibrate the sensors.

- **High rate:**

  Perform a high rate acquisition,*i.e.*with a short delay, in order to estimate the error rate.

- **Humidity:**

  Launch an acquisition of 2 minutes. After 30 secondes, start blowing on the Sensor Hat.

- **Temperature:**

  Launch a 2 minutes acquisition and after 30 secondes, turn on the hair dryer close (but not too) the SenseHat board.

- **Influence of the CPU/GPU activity:** Stop the raspberry and wait few minutes in order to decrease the temperature. Turn on the raspberry and start an acquisition with a duration of 2 minutes. Meanwhile, run on another terminal, the command *python hot.py* which is a *CPU intensive* program. The file can be downloaded from Repeat the same procedure while launching a video *video.avi*. The sequence will be repeated by putting directly the SenseHat on top of the raspberry pi.

- **Measurements outside:** We can use a USB battery to move the setup from inside to outside. More details must be asked to the teachers. This will allow us to work in varying conditions (P,T,RH)

Name	Duration [min]	Comments	Who
Stable conditions	5	delay 2 sec	all
High rate	0.25	scan short delays [ms to μs]	1-2
Humidity	3	blow on the SenseHat board after 30'	3-4
Temperature	3	run an air dryer after 30'	5-6
hRPi-1	4	run *hot.py*	7-8
hRPi-2	4	launch *video.avi*	9-10
hRPi-3	4	run *hot.py* - SenseHat on top of RPI	11-12
hRPi-4	4	launch *video.avi* - SenseHat on top of RPI	13-14
Outside	6	move the setup from inside to outside	15-16

Table 1: Summary of the runs to be launched

Part III

For going further...

6 Using the LED matrix.

If you have reached that point, it means that you have achieved the mail goal of this computing session: congratulations ! You will propose several options to go further.

6.1 Display measurements with the LED matrix

The Sense Hat board is equiped with a LED matrix. It is possibe to change the color of each of the individual pixels via methods of the class `SenseHat`. We will use this functionality here to represent the value of the measured observables (temperature, humidity) on a 8 LED line. You are free to achieve this goal and you can exchange ideas with the teachers. As guideline, think about defining a range, a color, a delay for refreshment, etc.

6.2 Light warning

An other option offered by the presence of the LED matrix, is to warn the user about conditions beyond predefined threshold such as high temperature, high humidity, etc ... You can define select variable(s) use to trigger warning, define thresholds, and develop a tool to make the LED matrix blinking.

6.3 Making the program configurable

Extend the previous project by passing the configuration parameters in a dedicated configuration file. You are free to define the format of the file. An option could be to have lines structure with keywords and values like this:

```
...
WarningVar humidity
WarningMin 30
WarningMax 60
## Frequency in Herz
WarningFreq 2
## Duration in sec.
WarningDuration 5
...
```

6.4 Output format

For the sake of simplicity we used the CSV format as it it human readable and easy to import in many programs. However this format is not convenient for large dataset. You can decide to save your data in binary format. This assume that you know in advance, in both the producer and the analyzer programs, what is the format. In that section, you can modify our program to save in a binary format the data and write the corresponding program to decode and analyze them.

Part IV
Appendix

A Class SenseHat

You will find below the content of the header file of the class SenseHat.

```
    #ifndef SENSE_HAT_H
#define SENSE_HAT_H

// Header for SenseHat colors
#include "SenseHatColor.h"

// Define implicitly the SenseHatCore class
class SenseHatCore;

// Definition of the SenseHat class
class SenseHat
{
 protected:
  SenseHatCore* core_;

 public:

  // Constructor without argument
  SenseHat();

  // Destructor
  ~SenseHat();

  // Initializing the SenseHat board: to call first
  bool Initialize(bool verbose=false, const char* cfgfilename = "RTIMULib.ini");

  // Displaying a nice logo on the pixels
  void DisplayESILogo();

  // Delaying the execution for a specified time in microseconds
  void Sleep(unsigned long time_microseconds);

  // Getting P and T from the pressure sensor in bar and in Celsius_deg
  bool GetPressureSensorData(float& pressure, float& temperature) const;

  // Getting H and T from the humidity sensor in % and in Celsius_deg
  bool GetHumiditySensorData(float& humidity, float& temperature) const;

  // Getting GPU T in Celsius_deg
  bool GetRaspberryTemperatureGPU(float& temperature) const;

  // Getting ARM T in Celsius_deg
  bool GetRaspberryTemperatureARM(float& temperature) const;

  // Getting the version of the library
```

```
  const char* GetVersion() const;

  // Getting pixel color at coordinate (x,y)
  SenseHatColor::Color_t GetPixel(const unsigned char& x, const unsigned char& y) const;

  // Setting pixel color at coordinate (x,y)
  void SetPixel(const unsigned char& x, const unsigned char& y, SenseHatColor::Color_t c

};

#endif
```