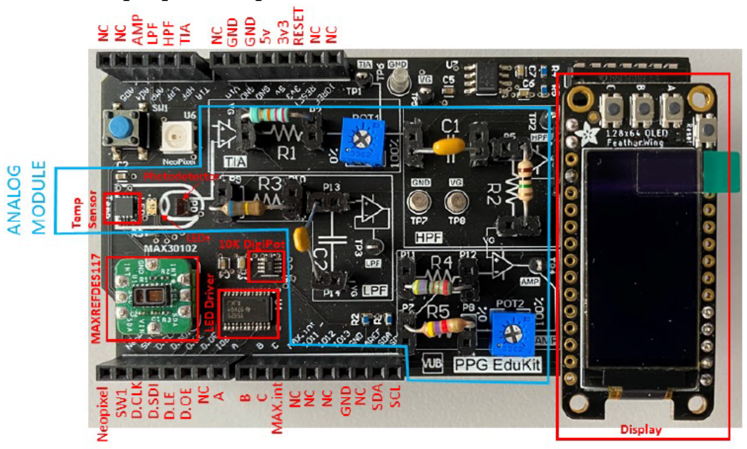
Lab 3: Plot raw PPG signal over UART with Arduino

Introduction

This lab helps the user to get a first interaction with the PPG EduKit platform, both in terms of hardware and software configuration. The goal is to measure the plethysmograph signal using the analog PPG module and send the data through UART in order to visualize the signal.

The PPG EduKit platform is shown below. The module can be used as a shield for any Arduino board as long as the microcontroller runs with a logic level voltage of 3V3. **Do not use the PPG EduKit with microcontrollers that have a logic level voltage of 5V, like the Arduino UNO.** The analog PPG module includes one RGB LED, a photodetector, a transimpedance amplifier, a high pass filter, a low pass filter and an amplification stage. In this lab the PPG signal is acquired from the last amplification stage, thus the board has to be configured with the proper component values.



Objectives

After completing this lab, you will be able to:

- Understand how to use the PPG EduKit Arduino library
- Understand the API functions of the libraries
- Use the PPG EduKit GUI application

Procedure

This lab is separated into steps that provide information on the detailed instructions that follow. Follow these detailed instructions to progress through the lab.

The lab includes 3 primary steps: create a new project and include the PPG EduKit library, write the main application by calling the library functions, and finally run the PPG EduKit GUI application to read and visualize the data.

1 Creation of the Project

- Open Arduino IDE.

- Go to File → New.
- After the sketch is created, save the sketch (File → Save As) as **UartPlot**.
- Copy the PPG EduKit library (PPG_EduKit.cpp and PPG_EduKit.h) in **UartPlot** directory.

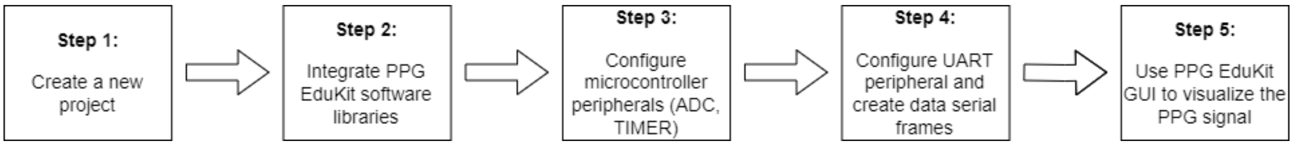
2 Use PPG EduKit library

PPG EduKit library is provided to speed up and to facilitate the development of PPG related applications, such that the user can focus on PPG signal acquisition or other types of algorithms. The library is a wrapper for other libraries, such as OLED library, graphical library, TLC5925 and AD5273 library.

- At the beginning of the sketch, include the **PPG_EduKit.h** header file and create an instance of **PPG_EduKit class**.

```
#include "PPG_EduKit.h"

PPG_EduKit PPG_Shield;
```

- Declare the global variables. The serial frame command is represented by the **frameParams_t** structure type. The variable is passed to the **createSerialFrame** function in order to create the desired data frame. The function returns the address location of the data frame buffer to be sent, and **gpFrame** pointer is used to point to that memory location. The sampling rate variable is used to control the ADC sampling rate in Hz.


Step 1:
Create a new project
Step 2:
Integrate PPG EduKit software libraries
Step 3:
Configure microcontroller peripherals (ADC, TIMER)
Step 4:
Configure UART peripheral and create data serial frames
Step 5:
Use PPG EduKit GUI to visualize the PPG signal

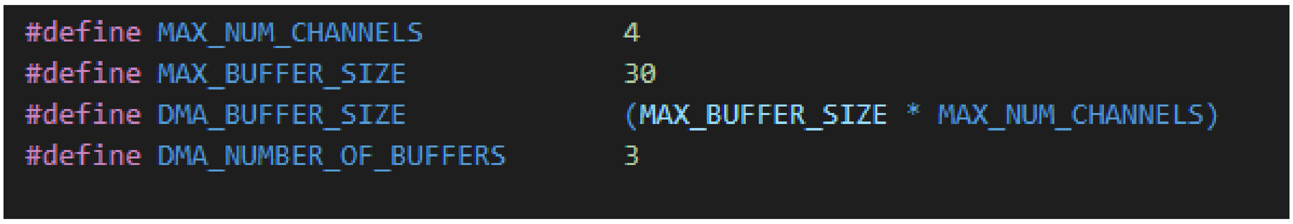
- Use **setup** function to initialize the **PPG_Shield**. The **periphList** structure is used to select the peripherals that should be initialized and configured. In this lab, only the **read_AMP** option should be enabled. **Begin** function should be called to initialize the list of peripherals and to set the ADC sample rate. **EnableLed** function is used to send commands to TLC5925 LED driver. A current of 5mA is configured using AD5273 digital potentiometer.

```
void setup() {

  PPG_EK_Peripherals periphList = {
    .oledDisplay = DISABLE_PERIPHERAL,
    .neoPixel = DISABLE_PERIPHERAL,
    .tempSensor = DISABLE_PERIPHERAL,
    .ppgSensor = DISABLE_PERIPHERAL,
    .read_TIA = DISABLE_PERIPHERAL,
    .read_HPF = DISABLE_PERIPHERAL,
    .read_LPF = DISABLE_PERIPHERAL,
    .read_AMP = ENABLE_PERIPHERAL
  };

  PPG_Shield.begin(&periphList, samplingRate);

  PPG_Shield.enableLed(IR_CHANNEL, 5, true);
}
```

- In main loop, the **readChannel** function is called periodically. The ADC is configured in DMA mode, where the DMA controller, which is a separate peripheral, is configured to automatically send data from one ADC peripheral to memory. The frame type is **CHANNEL_DATA**, the wavelength is set to **IR_CHANNEL**, **tissueDetected** flag should be always true since there is no need to signal this in this particular case. The frame is created by passing the buffer address to **createSerialFrame** function along with the frame structure. The size of the buffer should be the buffer length multiplied by 2 since every sample is represented by 2 bytes (uint16_t). The frame is sent over UART by calling the **sendFrame** function.

```
void loop() {

  uint32_t bufflength = 0;
  uint16_t* cBuf = PPG_Shield.readChannel(ADC_AMP, &bufflength);

  if(bufflength != 0)
  {
    memset(&frameParam, 0x00, sizeof(frameParam));
    frameParam.frameType = CHANNEL_DATA;
    frameParam.params.wavelength = IR_CHANNEL;
    frameParam.tissueDetected = true;
    gpFrame = PPG_Shield.createSerialFrame((uint16_t *) &cBuf[0], (bufflength) * 2, &frameParam);

    PPG_Shield.sendFrame(gpFrame);
  }

}
```

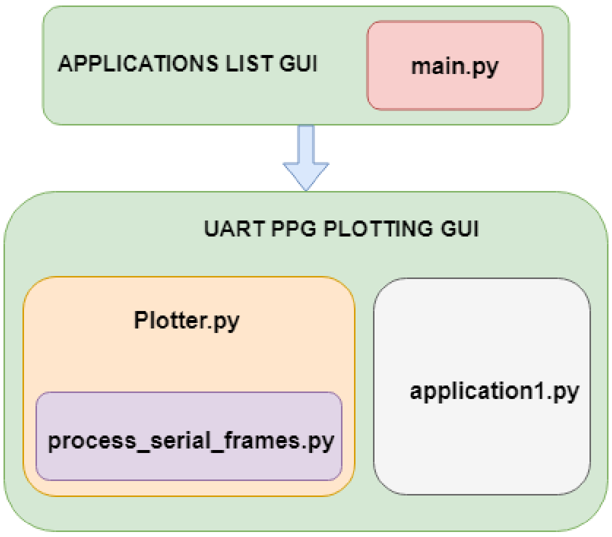
- The DMA uses a number of 3 buffers to store the PPG samples. While reading one buffer and sending it over UART, the DMA controller fills the next buffer without wasting CPU clock cycles! The buffer size and the number of DMA buffers can be configured in **PPG_EduKit.h** header file.



```
#define MAX_NUM_CHANNELS          4
#define MAX_BUFFER_SIZE           30
#define DMA_BUFFER_SIZE           (MAX_BUFFER_SIZE * MAX_NUM_CHANNELS)
#define DMA_NUMBER_OF_BUFFERS     3
```

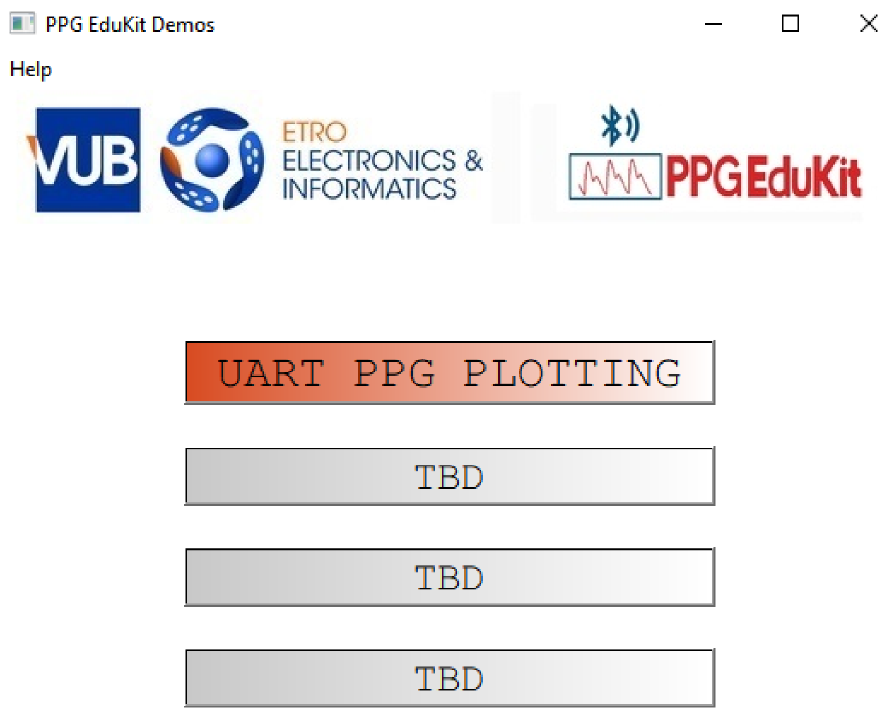
3 Run PPG EduKit GUI application

The PPG EduKit GUI is a Python program used to interface the PPG EduKit platform with different types of applications. The user interface is created using QT Designer and stored in **app1_gui.ui** file. The main window loads the requested application in the list. **UART PPG PLOTTING** application has two source files: **application1.py** and **Plotter.py**. The **process_serial_frames.py** file is used to interface the embedded application that communicates over UART with the PC. If the data frame structure is changed (by adding more functionalities), the change should be included both in the **serial_frame.c** file and in **process_serial_frames.py** script.

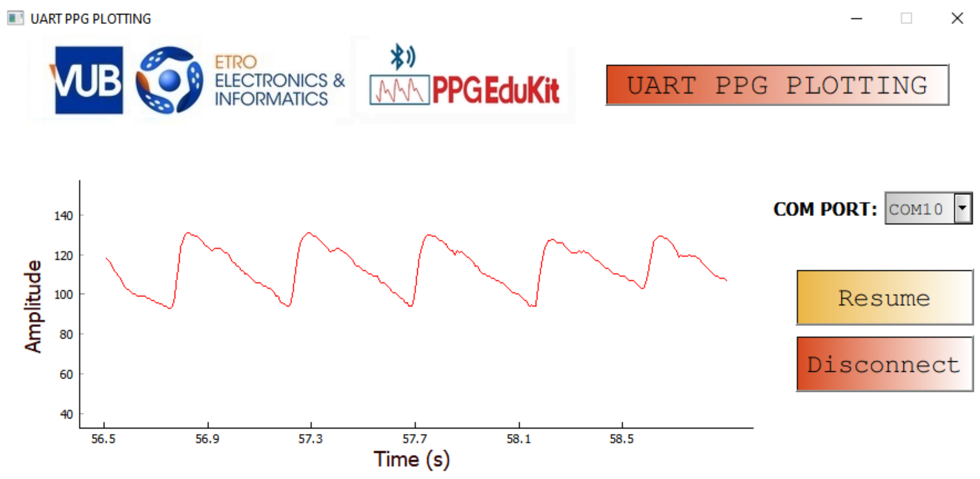


To run the application, one can use just a simple terminal, but the recommendation is to use Spyder IDE or PyCharm.

- Open the command line and go to the location of the PPG EduKit GUI application.
- Execute the command: **pip install -r requirements.txt**. If the pip package is not installed, please refer to [1] in order to set up your environment.
- Open the project in Spyder/PyCharm.
- Run the **main.py** file. In the application list interface, select the **UART PPG PLOTTING** application.



- Select the COM port and click **Connect**. Place your finger over the sensor's surface and visualize the volumetric changes in arterial blood, which is associated with your cardiac activity.



References

[1] https://phoenixnap.com/kb/install-pip-windows