Hardware Connection

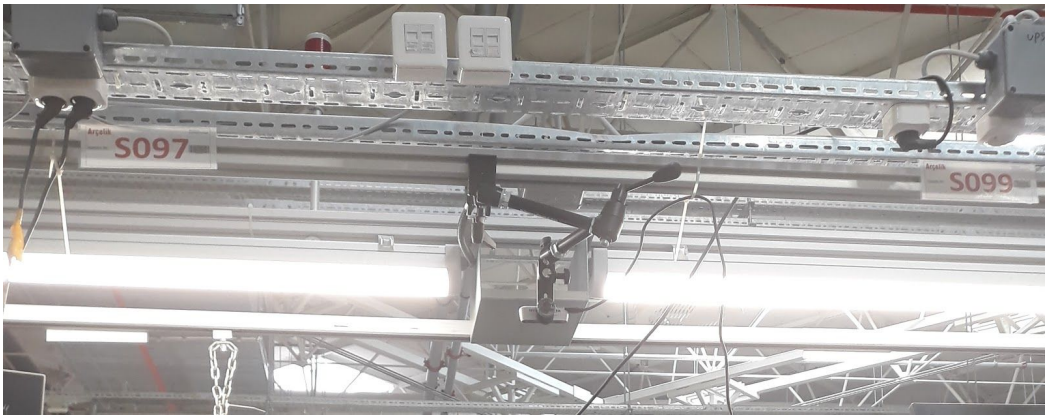
The camera intel realsense D435 needs a USB3 connection, USB2 doesn't work. The camera is provided with a USB3.0 cable that is 1m long. In order to record from a long distance, you have to use a USB3.0 extension cable, like this one:

You can buy it here:
https://www.amazon.fr/AmazonBasics-Rallonge-C%C3%A2ble-m%C3%A2le-femelle/dp/B00NH12O5I?th=1

Camera mount

The camera comes with a tiny tripod:

But to record in uncontrolled environments such as a factory, it is recommended to use an articulated mount, that is rigid and stable. For instance:



This mount is a “manfrotto 143 magic kit”, you can buy it here:
https://www.amazon.co.uk/Manfrotto-143-Magic-Arm-Kit/dp/B0011EHRDM
or here:
https://www.manfrotto.fr/bras-magique-avec-support

Documentation

The documentation for the camera can be found online:

- Datasheet : https://realsense.intel.com/wp-content/uploads/sites/63/Intel_RealSense_D400_Family_Datasheet_Jan2019.pdf
- Github wiki and documentation (this is a good starting point): https://github.com/IntelRealSense/librealsense/tree/master/doc
- General tutorials : https://software.intel.com/en-us/realsense/documentation

SDK

The camera has an SDK supported in Windows, linux and Mac OS called *librealsense2*.
There are 2 ways to install the SDK :

- Building the SDK from source:

https://github.com/IntelRealSense/librealsense/blob/master/doc/installation.md

- Installing packages using apt-get install (if you work under linux ubuntu):

https://github.com/IntelRealSense/librealsense/blob/master/doc/distribution_linux.md

From my experience, when you install the packages using apt-get under ubuntu 16.04 LTS, the installed software is unable to receive data streams from the camera. For this reason, I recommend building the SDK from source.

Code samples

The camera comes with code samples (refer to the folder “examples”). When building the SDK, make sure to build the samples. You can compile them separately using g++. For instance:
g++ -std=c++11 rs-save-to-disk.cpp -o save_to_disk -lrealsense2
(make sure to put the file stb_image_write.h in the same folder as rs-save-to-disk.cpp)
To compile a c++ program (here record.cpp) using librealsense2 and opencv under linux, use the following command:
g++ -std=c++11 record.cpp -o record `pkg-config --cflags --libs opencv` -lrealsense2

Third-party libraries

OpenCV can be used to save and load frames.

Recording with the camera

1. Record a video in .bag format

Either using realsense-viewer or a script (such as rs-record-and-playback.cpp that you can find in the folder librealsense/examples/record-playback).
There is a tool provided with the sdk to convert bag files to other formats:

in librealsense/tools/converters, when building librealsense from source, you should build examples too (using cmake ../ -DBUILD_EXAMPLES=bool:true). Using this tool (**rs-convert**), one can extract depth and color images from a bag file.

Here is a command to extract color images from a bag file:
<path_to_rs-convert> -i <path_to_bag_file> -p <path_to_folder_for_extracted_frames> --color
In order to extract depth images, replace **--color** with **--depth**
From my experience, the frames extracted from a .bag file are not synchronized, for this reason, the number of extracted RGB frames is not always equal to the number of extracted depth frames.

2. Record frames

It is possible to record RGB, depth and infrared frames and their metadata using a script. Librealsense provides an example : rs-save-to-disk.cpp.

Synchronization between depth and RGB streams
For the hardware synchronization between the depth sensor and the RGB sensor to be possible, **both sensors should operate at the same frame rate**.

Correlation between resolution and frame rate
When the resolution is high, the frame rate drops automatically, and the frame rate that you specify in the C++ script is not respected. For example, for an RGB image of resolution 1280x960, the frame rate is 15fps, even when you specify higher frame rates such as 30fps as you create the rgb stream. To make sure of that, print the metadata of the frames.
I recommend a resolution of (640x480) for depth and RGB to be able to reach and keep a 30fps (actually it's 29fps) for both streams.