

Hardware II Seminar

Sensors and Data Analysis

Quick recap

Making an actual sensor setup

Interfacing with hardware for sensing

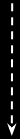


Sensors and computers

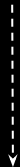
Making an actual sensor setup



SENSOR



COMPUTER I



COMPUTER II

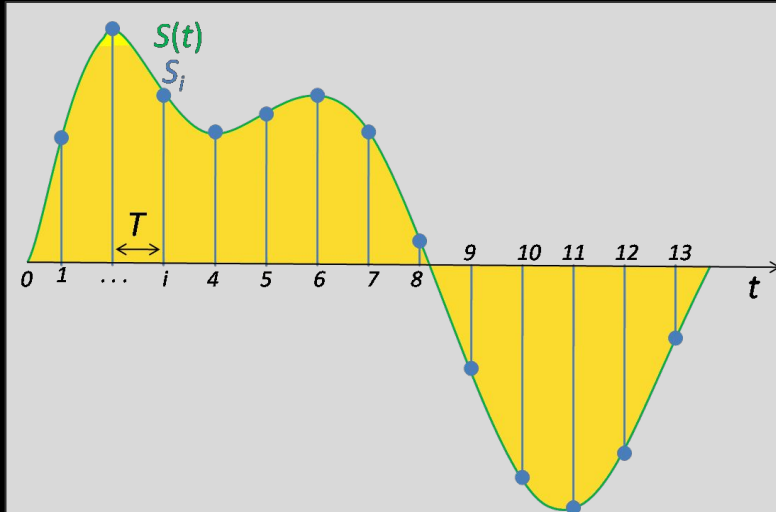
- *How do we read the sensor? Analog, or digital sensor?*
- *Can the computer read it?*
- *Are there libraries for it?*
- *Do we need an additional computer for storing the data?*
- *Or for post-processing it?*
- *How do both computers communicate? Cable or wireless? Serial, WiFi... ?*

Making an actual sensor setup - Streaming data



Three important frequencies to take into account:

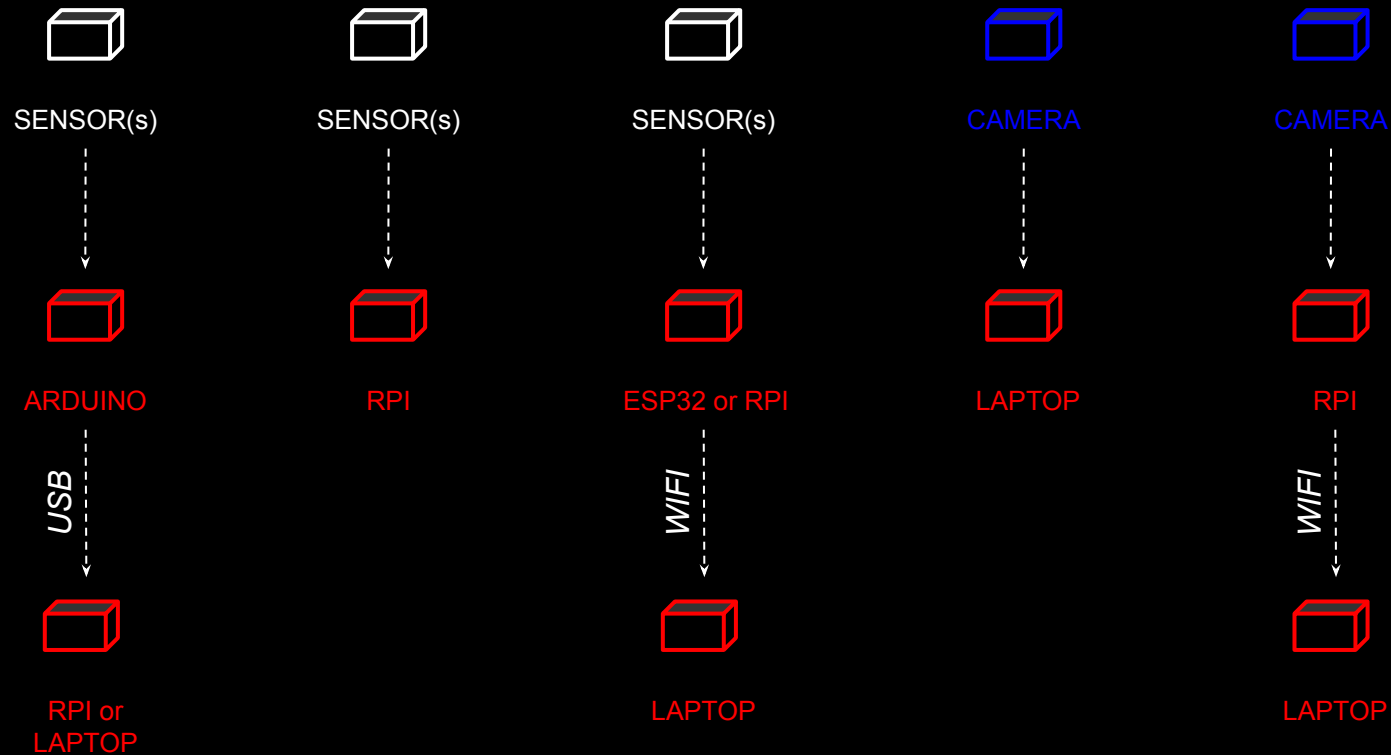
1. *The frequency at which the sensor gives us data*
2. *The frequency at which we (can) sample the sensor*
3. *The natural frequencies that we want to capture*



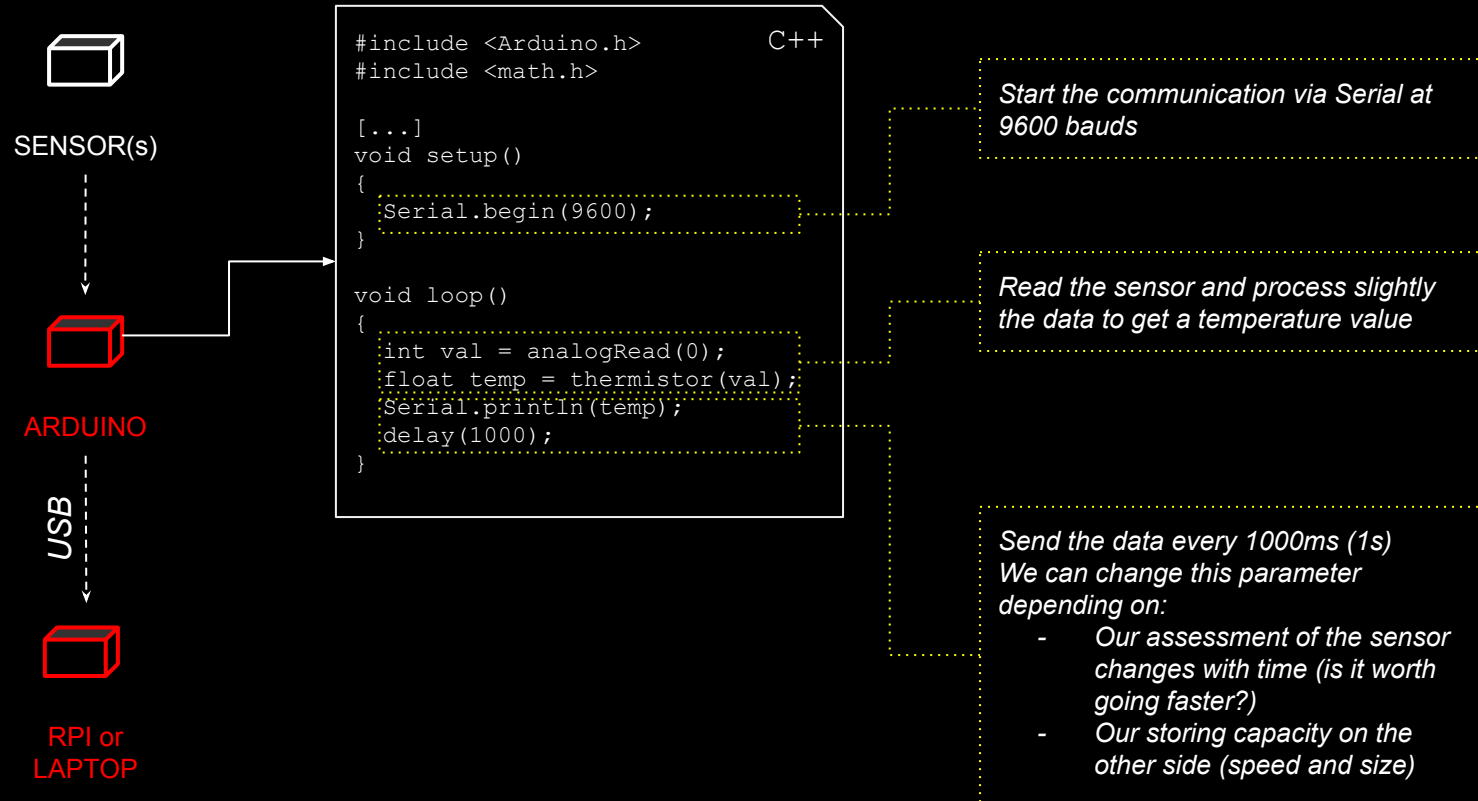
Key takeaways:

1. *We don't need to sample faster than the sensor gives us data*
2. *We need to sample at least twice as fast as the Nyquist frequency*
([Nyquist-Shannon theorem](#))

Making an actual sensor setup - Some use cases



Making an actual sensor setup - Some use cases

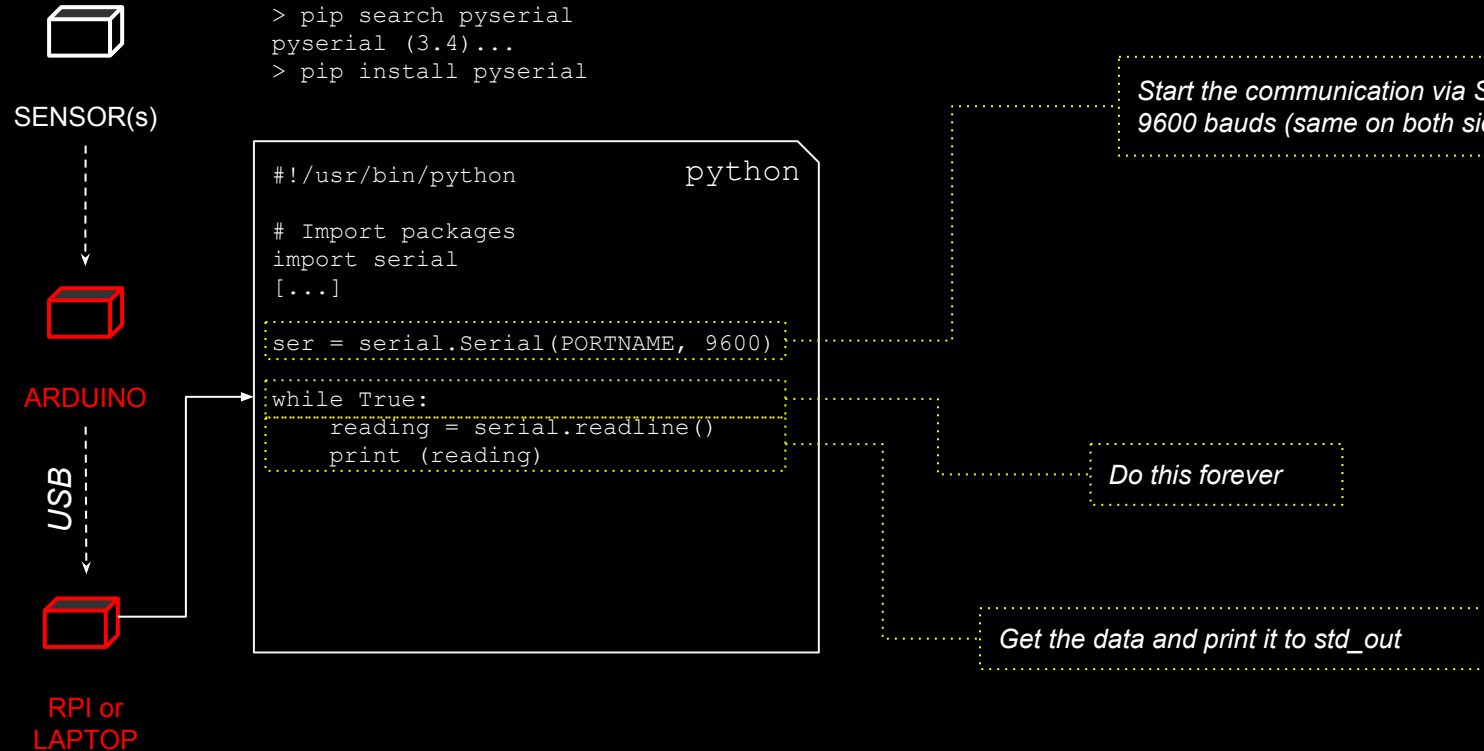


Making an actual sensor setup - Some use cases



First, in the terminal

```
> pip search pyserial  
pyserial (3.4)...  
> pip install pyserial
```



Quick note

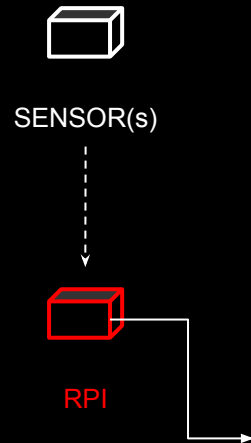


Environments in python

*In any computer, you can use `python` in different ways. You can install libraries **system-wide** or in a **virtual environment**. The best one depends on your use case, but normally, it's a good idea to use **virtual environments**. Some environment managers are:*

- Anaconda (*easy, with GUI, but quite heavy*): <https://www.anaconda.com/products/individual>
- Virtualenv (*official*): <https://pipenv.kennethreitz.org/en/latest/install/#make-sure-you-ve-got-python-pip>
- Virtualenvwrapper (*the best, IMHO*): <https://virtualenvwrapper.readthedocs.io/en/latest/>

Making an actual sensor setup - Some use cases



First, install necessary packages

```
sudo apt update
sudo apt upgrade
sudo apt install -y i2c-tools python3-smbus
```

Second, enable your I2C interface in raspi-config and go to Interfacing options

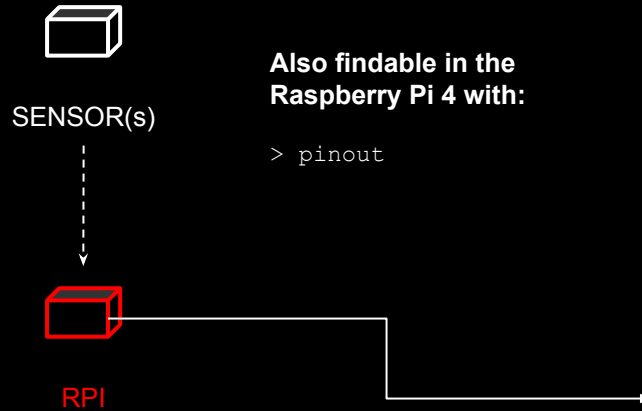
```
> sudo raspi-config
...
> sudo reboot
```

Then, connect the sensor following the pinout (see next slide). Make sure voltage levels are ok for your sensor - the raspberry pi *talks* I2C at 3V3

Then, detect your sensor with (SHT31 address is 44):

```
> i2cdetect -y 1
    0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
10:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
20:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
30:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
40:  --  --  --  --  44  --  --  --  --  --  --  --  --  --  --
50:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
60:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
70:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
```

Making an actual sensor setup - Some use cases



Also findable in the
Raspberry Pi 4 with:

> pinout

Pi Zero

Pi 3

The image shows two Raspberry Pi boards, Pi Zero and Pi 3, with their pinouts. Red and blue dashed lines connect specific pins on the boards to the corresponding entries in the pinout table.

Pin No.	
1	2
3	4
5	6
7	8
9	10
11	12
13	14
15	16
17	18
19	20
21	22
23	24
25	26
27	28
29	30
31	32
33	34
35	36
37	38
39	40

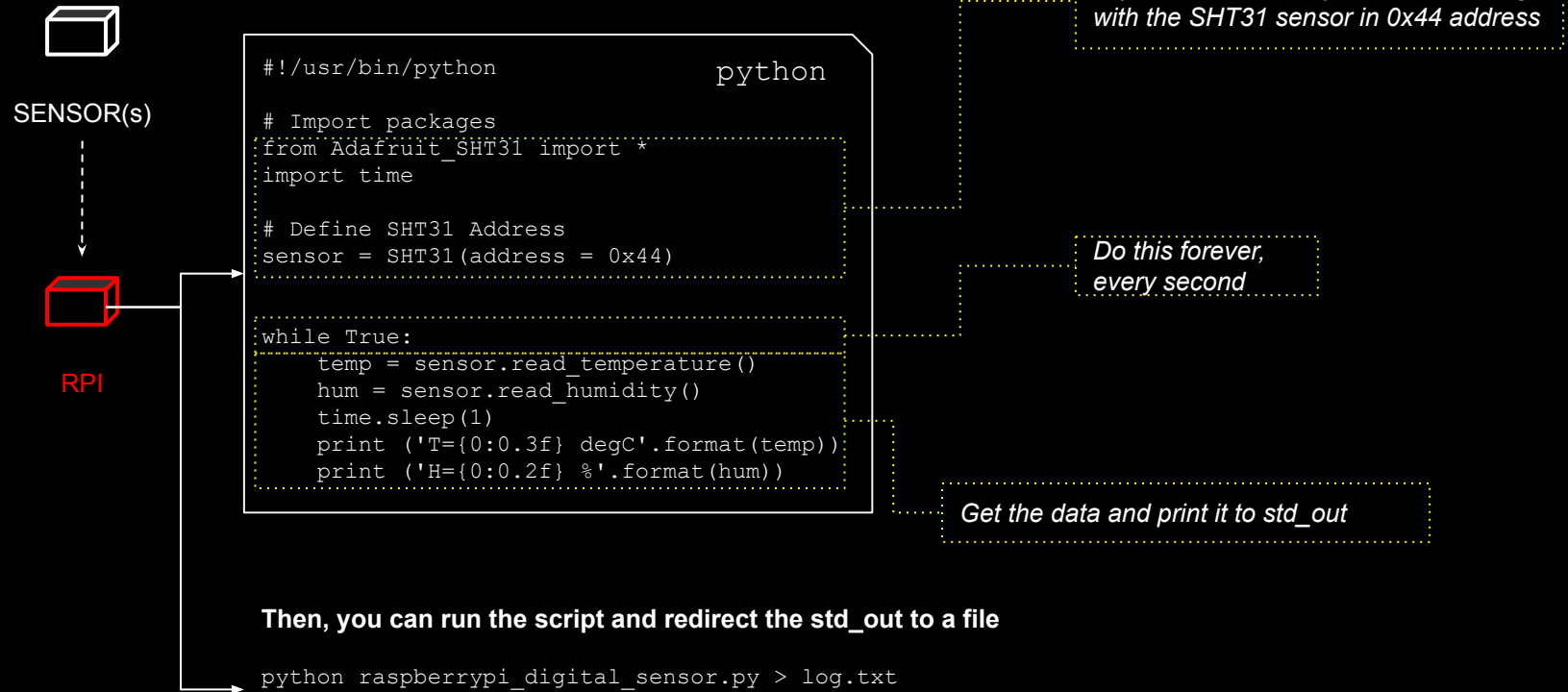
Pin	Pin	Pin	Pin
3.3V	GPIO2	5V	5V
GPIO3	GPIO4	GND	GND
GPIO17	GPIO14	GPIO15	GPIO18
GPIO27	GPIO18	GND	GPIO23
GPIO22	GPIO24	GND	GPIO25
3.3V	GPIO9	GPIO11	GPIO8
GPIO10	GPIO11	GND	GPIO7
GPIO9	DNC	DNC	DNC
GPIO11	GPIO5	GPIO6	GPIO12
GND	GPIO13	GPIO19	GPIO16
GND	GPIO26	GPIO20	GPIO21

Key

Power +	UART
GND	SPI
I ² C	GPIO

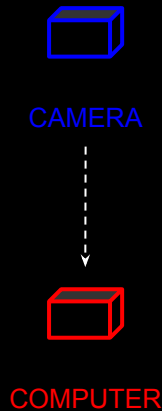
Source: Building the Web of Things: book.webofthings.io
Creative Commons Attribution 4.0

Making an actual sensor setup - Some use cases



And magically, the data will be saved in the file. You can go on and modify the above pseudocode to add a timestamp in your file

Making an actual sensor setup - Some use cases



When using a camera, always look for documentation on the hardware:

- **Kinect:** https://openkinect.org/wiki/Main_Page
- **Intel REALSENSE:** <https://dev.intelrealsense.com/docs/opencv-wrapper> (*openNI* too)
- **Raspberry PI camera:** <https://projects.raspberrypi.org/en/projects/getting-started-with-picamera>

Mainly, we will use `OpenCV` to interact with most of them:

OpenCV (Open Source Computer Vision Library: <http://opencv.org>) is an open-source BSD-licensed library that includes several hundreds of computer vision algorithms.



Making an actual sensor setup - Some use cases



CAMERA



COMPUTER

There are many ways to interact with cameras using OpenCV.
For using depth cameras, we'll use [freenect](#)

OpenCV with python



Two options
(full-fledged) or
pre-built in python
(enough for
non-super-mega
cutting edge).

Easy to use, well
documented.

processing



Easy to use for new
coders. Well
documented and
running easily for
RGBD camera.

openframeworks



Seemly more
complex at the
beginning. Need of
some notions to
compile C++ and
build the code.
Robust and well
documented.
See [Lewis Lepton
videos to get started](#)

Making an actual sensor setup - Some use cases



CAMERA



COMPUTER

Follow instructions on

https://docs.opencv.org/4.3.0/d4/d65/tutorial_table_of_content_introduction.html for a cutting edge OpenCV installation or use the opencv-python package

Recommended, create a virtualenv for opencv and:
pip install opencv-python

Basic example (RGB camera capture):

```
#!/usr/bin/python                                python
import cv2 as cv
cap = cv.VideoCapture(0)
while(True):
    ret, frame = cap.read()
    cv.imshow('Output', frame)
    if cv.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()
cv.destroyAllWindows()
```

Open video capture on device 0
(webcam in my case)

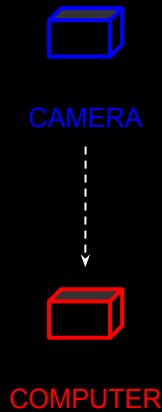
Capture a frame and display it

Close everything if q is pressed

OpenCV with python



Making an actual sensor setup - Some use cases



Follow instructions on

https://docs.opencv.org/4.3.0/d6/d65/tutorial_table_of_content_introduction.html for a cutting edge OpenCV installation

Installing OpenCV on the Pi:

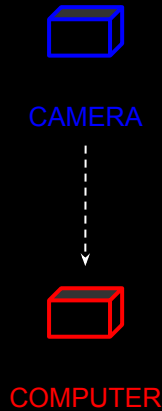
*<https://pimylifeup.com/raspberry-pi-opencv/>
(perfect guide to help you set it up)*



OpenCV with python



Making an actual sensor setup - Some use cases



Video capture and storage of RGB camera example:

```
import cv2                                     python
cap = cv2.VideoCapture(0)
if (cap.isOpened() == False):
    print("Unable to read camera feed")

frame_width = int(cap.get(3))
frame_height = int(cap.get(4))

out =
cv2.VideoWriter('out.avi',cv2.VideoWriter_fourcc('M',
', 'J', 'P', 'G'), 10, (frame_width,frame_height))

while(cap.isOpened()):
    ret, frame = cap.read()
    if ret==True:
        frame = cv2.flip(frame,0)
        out.write(frame)
        cv2.imshow('frame',frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    else:
        break
cap.release()
out.release()
cv2.destroyAllWindows()
```

OpenCV with python



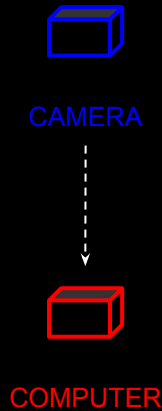
Open video capture on device 0
(webcam in my case)

VideoWriter (see
<https://www.fourcc.org/fourcc.php>)

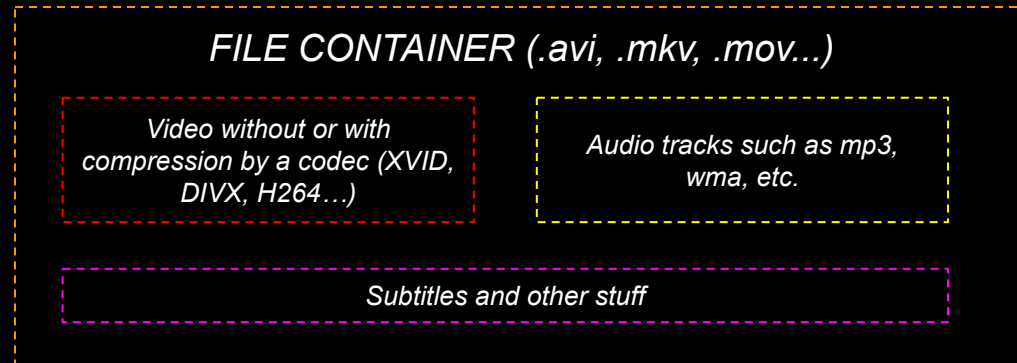
Capture, store and show frame

Close everything if q is pressed

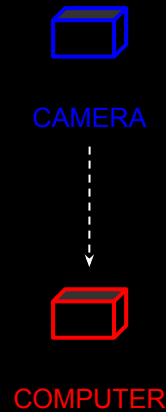
A word about video files and video codecs



Every video file in itself is a container. The type of the container is expressed in the files extension (for example avi, mov or mkv). This contains multiple elements like: video feeds, audio feeds or other tracks (like for example subtitles). How these feeds are stored is determined by the codec used for each one of them. In case of the audio tracks commonly used codecs are mp3 or aac. For the video files the list is somehow longer and includes names such as XVID, DIVX, H264 or LAGS (Lagarith Lossless Codec). The full list of codecs you may use on a system depends on just what one you have installed.



Making an actual sensor setup - Some use cases



Kinect openCV python

python

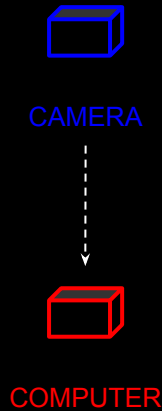
MAYBE NOT WELL IMPLEMENTED -
SEE
<https://github.com/OpenKinect/libfreenect/tree/master/wrappers/python/>
and better use OPENFRAMEWORKS
OR PROCESSING



OpenCV with python



Making an actual sensor setup - Some use cases



Basic example of video capture with RGB camera:

```
java
import processing.video.*;

Capture cam;

void setup() {
  size(640, 480);

  String[] cameras = Capture.list();
  cam = new Capture(this, cameras[0]);
  cam.start();
}

void draw() {
  if (cam.available() == true) {
    cam.read();
  }
  image(cam, 0, 0);
}
```

Init camera

Capture RGB image and then show it



Making an actual sensor setup - Some use cases



CAMERA



COMPUTER

Follow instructions on <https://shiffman.net/p5/kinect/> for setting up processing for Kinect. For REALSENSE, refer to <https://github.com/cansik/realsense-processing>

Basic example of Depth camera (Kinect):

```
import org.openkinect.freenect.*;
import org.openkinect.processing.*;

Kinect kinect;

void setup() {
  kinect = new Kinect(this);
  kinect.initDevice();
  kinect.initDepth();
  kinect.initVideo();
}

void draw() {
  background(0);
  image(kinect.getVideoImage(), 0, 0);
  image(kinect.getDepthImage(), 640, 0);
}
```

java

Init Kinect device

*Capture RGB image and depth image,
and place it in the output window*

processing



Making an actual sensor setup - Some use cases



CAMERA



COMPUTER

Check examples in openframeworks repository:

https://github.com/openframeworks/openFrameworks/tree/master/examples/computer_vision/kinectExample

Basic example of Depth camera (Kinect):

```
TOO LONG TO SHOW HERE, BUT C++  
MUCH BETTER CONTROL OF  
EVERYTHING...
```



openframeworks



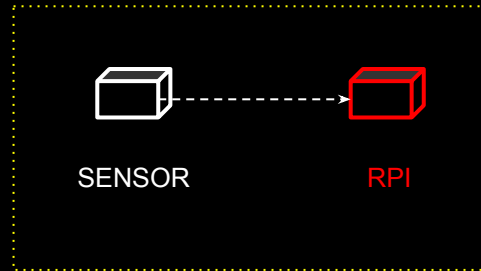
Making an actual sensor setup - Some use cases



Full example using zeromq in the repo

https://github.com/oscgonfer/sensors_dsp_lectures/tree/master/02_datalogging/examples also, more on mqtt [here](#)

Computer 1 - (streamer)



Computer 2 - (viewer)



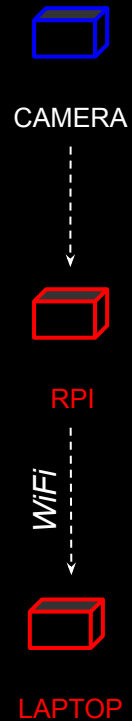
`python StringStreamViewer.py -p PORT`

`python StringStreamer.py -s IP-OF-COMPUTER-2 -p
PORT`

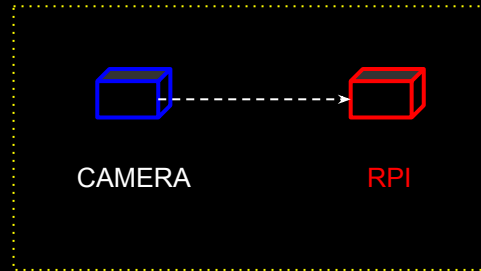
Making an actual sensor setup - Some use cases



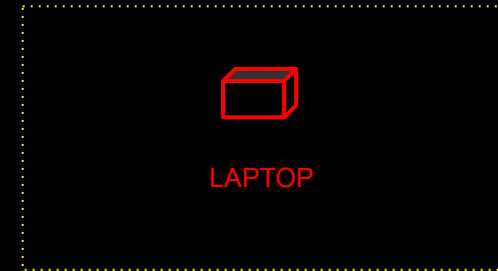
Check example in this repository:
<https://github.com/CT83/SmoothStream>



Computer 1 - (streamer)



Computer 2 - (viewer)



```
python StreamViewer.py -p PORT
```

```
python Streamer.py -s IP-OF-COMPUTER-2 -p PORT
```