APPAS Sybil – COLLOT
DUPRE Anatole – CASTANIE Léonard

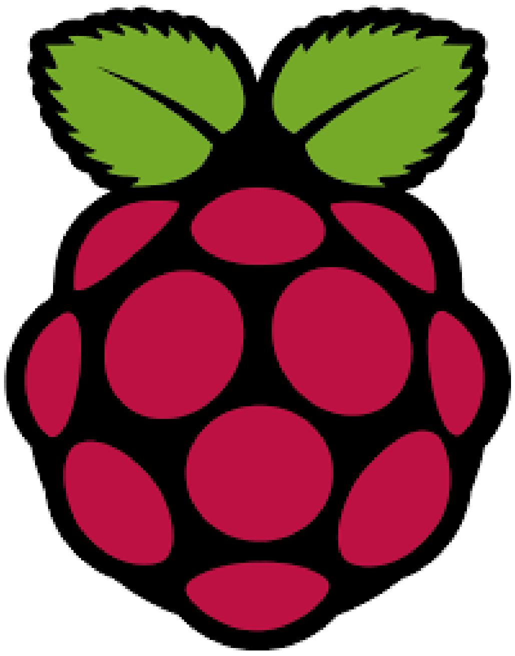
ÉCOLE D'INGÉNIEURS PARIS-LA DÉFENSE

Embedded Systems

Proof of Concept :

Design of a connected babyphone

using a raspberry pi3



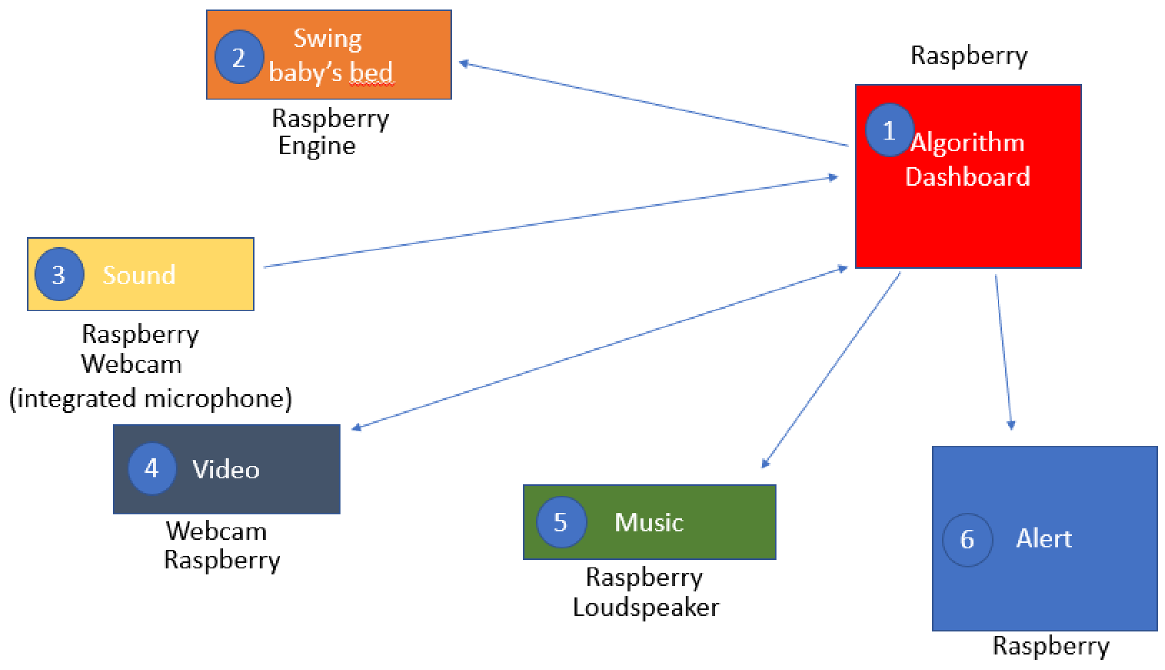
Summary

Introduction

According to our friend Wikipedia a "Proof of Concept" (PoC) is a realization of a certain method or idea in order to demonstrate its feasibility, or a demonstration in principle with the aim of verifying that some concept or theory has practical potential. So we reasoned like that to find our POC subject, we looked for key everyday objects, used by a large majority that could be improved by us by adding additional functions to improve the quality of life of the person (s) who use it. After a long brainstorming session we unanimously decided to do a POC on the connected babyphone. Indeed it is an object that can be improved, can be connected with the parents that are not in the same room than their child.

The first step of the poc was to agree on the different features that our babyphone would have, in order to do it we did a block diagram :

- swing the cradle when the baby starts to cry
- film the baby and transmit it on the phone's parents
- broadcast music whenever the parents want
- send an alert on the parent's phone when the baby starts to cry



The material needed

As you can understand, our baby phone allows the parents to see, swing, listen and also to interact with the child by sending lullaby and receiving alert. In order to create all those features we have combined:

A raspberry pi3 :

A servomotor:

A webcam with a microphone:

A speaker:

We turned to a Raspberry board rather than Arduino because we needed severals features working at the same time. Indeed, the Raspberry card allowed us to combined all this device and connect the device to internet. In addition, on contrary to Arduino board that accepts only the C language, we decided to encode in Python because we needed many librairies available in this framework which allowed us to run all our functionalities at once and moreover to group them on a fully united dashboard.

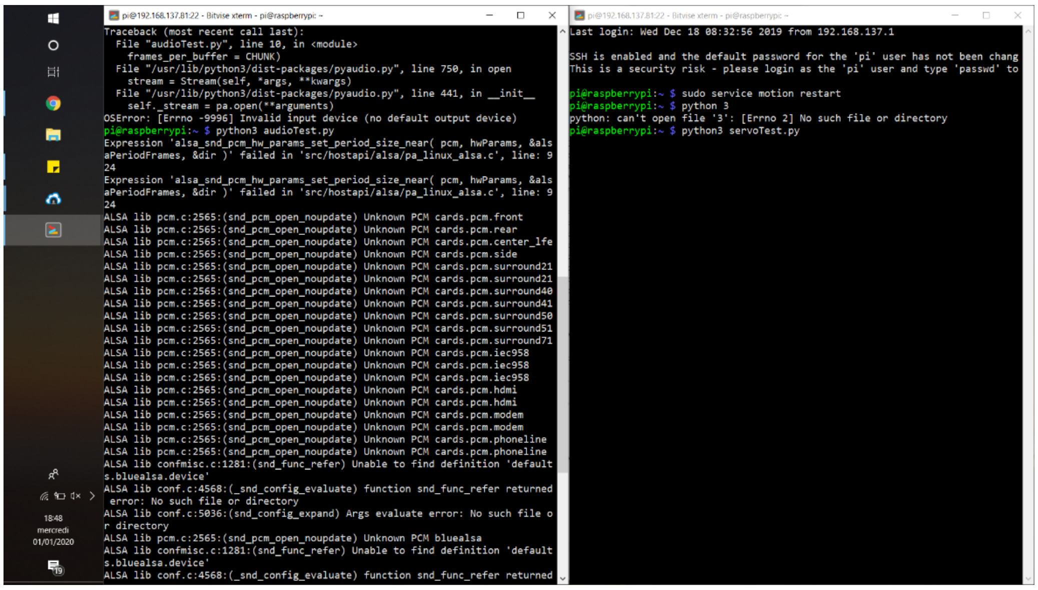
Functionalities

The use of a webcam

In this part, our goal is to read continuously the value of the sound intensity through the microphone of our webcam. Then when the sound intensity will exceed a certain value (for example when the children is crying) we want to activate the camera of our webcam to keep an eye on the child.

One of our big problem in this part is that we didn't arrive to read the value of the sound intensity in the same time than activate the camera. Our first idea was to read the value and only when the children is crying to activate the camera and let the user switch on or off the camera via a web interface and restart reading the sound intensity value. It was very difficult because we had to switch on and off the camera while the code was running and to do that, we have to be super user(sudo) and it was impossible through our interface because when we execute a python code via a php web page we run it as a simple user. We also try to write a value in a folder and then execute the start and stop of the camera via the raspberry directly, but it didn't work probably due to a problem of permission too.

Finally, we found possible to read the value of the sound intensity and use the camera in the same time. We had to run the commands on two different terminal consoles. So we decided to let's the camera on during all the time.

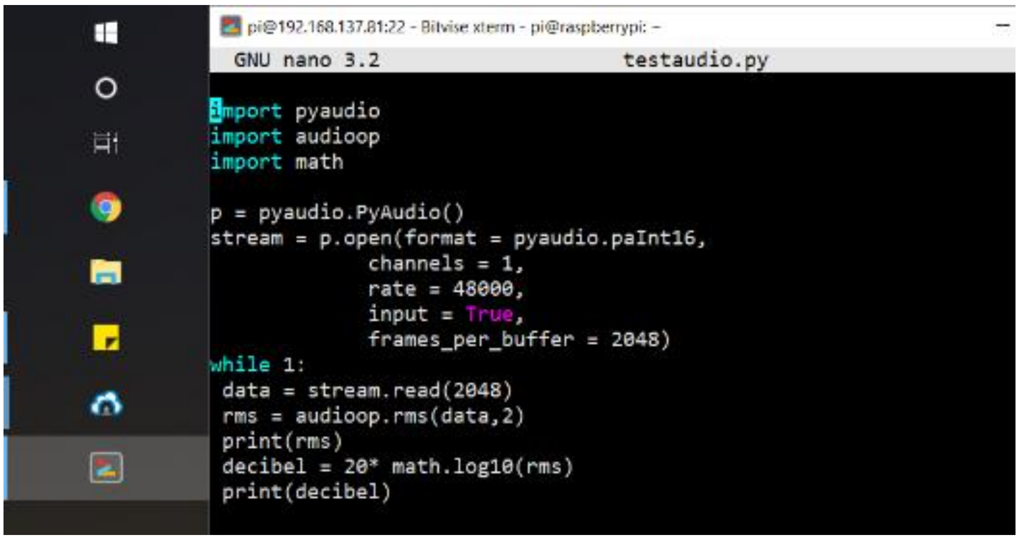

pi@192.168.137.81:22 - Bitvise xterm - pi@raspberrypi: ~
18:48
mercredi
01/01/2020
pi@192.168.137.81:22 - Bitvise xterm - pi@raspberrypi: ~

```
Traceback (most recent call last):
  File "audioTest.py", line 10, in <module>
    frames_per_buffer = CHUNK)
  File "/usr/lib/python3/dist-packages/pyaudio.py", line 750, in open
    stream = Stream(self, *args, **kwargs)
  File "/usr/lib/python3/dist-packages/pyaudio.py", line 441, in __init__
    self._stream = pa.open(**arguments)
OSError: [Errno -9996] Invalid input device (no default output device)
pi@raspberrypi:~ $ python3 audioTest.py
Expression 'alsa_snd_pcm_hw_params_set_period_size_near( pcm, hwParams, &als
aPeriodFrames, &dir )' failed in 'src/hostapi/alsa/pa_linux_alsa.c', line: 9
24
Expression 'alsa_snd_pcm_hw_params_set_period_size_near( pcm, hwParams, &als
aPeriodFrames, &dir )' failed in 'src/hostapi/alsa/pa_linux_alsa.c', line: 9
24
ALSA lib pcm.c:2565:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.front
ALSA lib pcm.c:2565:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.rear
ALSA lib pcm.c:2565:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.center_lfe
ALSA lib pcm.c:2565:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.side
ALSA lib pcm.c:2565:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.surround21
ALSA lib pcm.c:2565:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.surround21
ALSA lib pcm.c:2565:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.surround40
ALSA lib pcm.c:2565:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.surround41
ALSA lib pcm.c:2565:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.surround50
ALSA lib pcm.c:2565:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.surround51
ALSA lib pcm.c:2565:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.surround71
ALSA lib pcm.c:2565:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.iec958
ALSA lib pcm.c:2565:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.iec958
ALSA lib pcm.c:2565:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.iec958
ALSA lib pcm.c:2565:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.hdmi
ALSA lib pcm.c:2565:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.hdmi
ALSA lib pcm.c:2565:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.modem
ALSA lib pcm.c:2565:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.modem
ALSA lib pcm.c:2565:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.phoneline
ALSA lib pcm.c:2565:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.phoneline
ALSA lib confmisc.c:1281:(snd_func_refer) Unable to find definition 'default
s.bluealsa.device'
ALSA lib conf.c:4568:(_snd_config_evaluate) function snd_func_refer returned
 error: No such file or directory
ALSA lib conf.c:5036:(snd_config_expand) Args evaluate error: No such file o
r directory
ALSA lib pcm.c:2565:(snd_pcm_open_noupdate) Unknown PCM bluealsa
ALSA lib confmisc.c:1281:(snd_func_refer) Unable to find definition 'default
s.bluealsa.device'
ALSA lib conf.c:4568:(_snd_config_evaluate) function snd_func_refer returned
```

```
Last login: Wed Dec 18 08:32:56 2019 from 192.168.137.1

SSH is enabled and the default password for the 'pi' user has not been chang
This is a security risk - please login as the 'pi' user and type 'passwd' to

pi@raspberrypi:~ $ sudo service motion restart
pi@raspberrypi:~ $ python 3
python: can't open file '3': [Errno 2] No such file or directory
pi@raspberrypi:~ $ python3 servoTest.py
```

Capture the sound volume

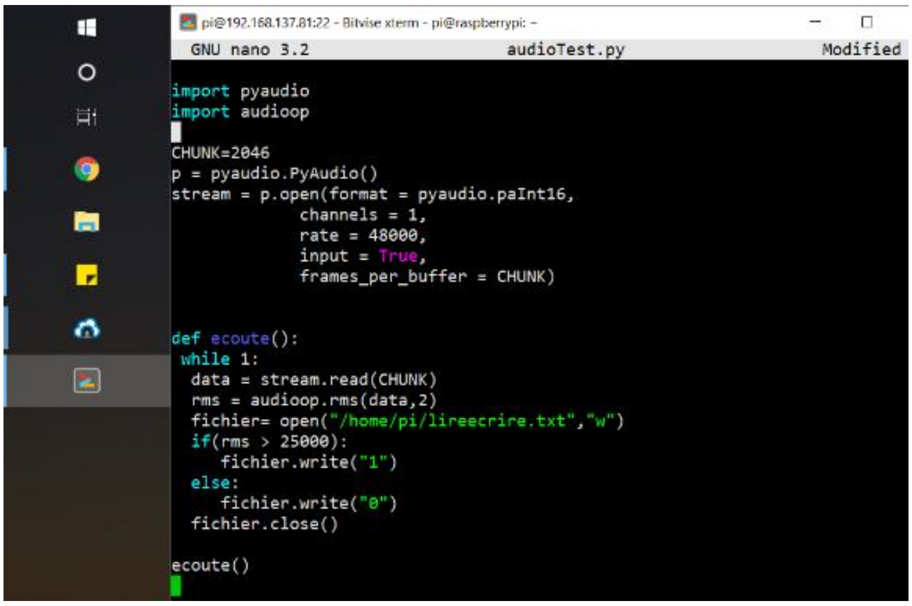
Our goal in this part is to capture a value of the sound intensity using the webcam. Firstly, we have to set up the connection with the microphone of our webcam. We use the library pyaudio. It's a library who allows us to communicate with the audio port and more specially to record audio. So, in our code we create a way of communication between the micro and the raspberry with the first line: *p= puaudio.PyAudio()* and then we defined some parameters of our recording. In our prototype we want to record the sound during a long period. But we don't have enough memory to do that, so we use a CHUNK which is a list with a certain size. Instead of recording the sound during a long period we record continuously the sound but in the same time we delete our old recording. The consequence is that we don't waste too much memory. But we have to set up the chunk and its size to avoid some errors of overflowed. Once the way of communication is well defined we use the function *stream.read(chunk)* of pyaudio to read audio data from the stream. We also need the library audioop, it contains some useful operations on sound fragments. It allows us to have an average value of the sound intensity with the function *audioop.rms(data,2)* which return the measure of the power in the audio records data. Then we convert our value in decibel with a formula: dB=20 * log10(rms) to fix our threshold.



pi@192.168.137.81:22 - Bitvise xterm - pi@raspberrypi: ~

```
GNU nano 3.2                    testaudio.py

import pyaudio
import audioop
import math

p = pyaudio.PyAudio()
stream = p.open(format = pyaudio.paInt16,
             channels = 1,
             rate = 48000,
             input = True,
             frames_per_buffer = 2048)
while 1:
 data = stream.read(2048)
 rms = audioop.rms(data,2)
 print(rms)
 decibel = 20* math.log10(rms)
 print(decibel)
```

This was our first code about the capture of the sound volume, so we print the value rms and decibel to see if the code works (especially the connection with the micro). In the demonstration of our prototype I also use this code to be sure that the microphone of the webcam and the raspberry pi3 are well connected. In order to integrate this part in the all prototype we decided to write in a folder a value "1" when the value of the sound is higher than the threshold and 0 when the value is lower. At the beginning we wanted to run directly in this program the other code to send an email and activate the servomotor, but we had problem. In fact we started to have a big code with many parts so we supposed that some of them were not compatible and some steps take different duration, so it happened problem.



pi@192.168.137.81:22 - Bitvise xterm - pi@raspberrypi: ~

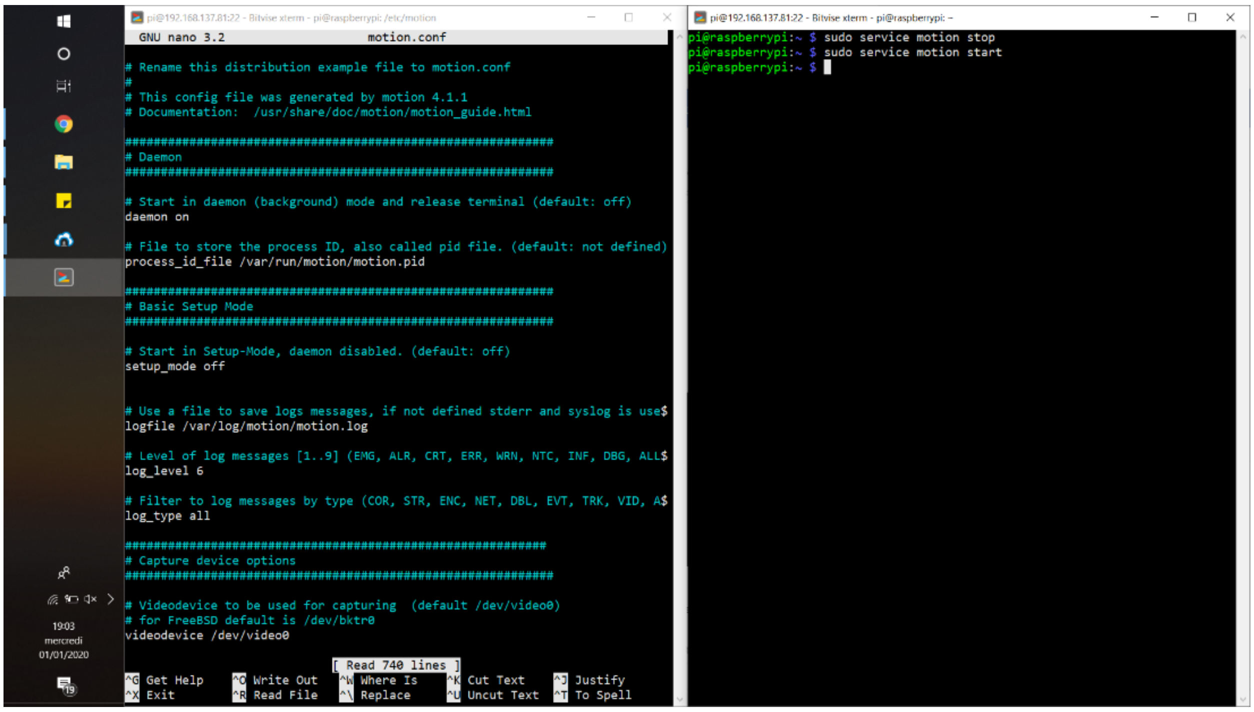
```
GNU nano 3.2                    audioTest.py                    Modified

import pyaudio
import audioop

CHUNK=2046
p = pyaudio.PyAudio()
stream = p.open(format = pyaudio.paInt16,
             channels = 1,
             rate = 48000,
             input = True,
             frames_per_buffer = CHUNK)


def ecoute():
 while 1:
  data = stream.read(CHUNK)
  rms = audioop.rms(data,2)
  fichier= open("/home/pi/lireecrire.txt","w")
  if(rms > 25000):
     fichier.write("1")
  else:
     fichier.write("0")
  fichier.close()

ecoute()
```

Use the camera

We followed the tutorial giving for the TP n°4 “Raspberry Video”. So, we installed the webcam server software motion, then we did the modification mentioned in the tutorial. We also change the size of the image and we add a password in order than only the parents can have access to the web page. This software is used as a motion sensor, for example we can set threshold value representing the number of pixels who change on each image. If the value is higher than the threshold, we can activate an alert. This software doesn’t give us access to the sound.

To launch/stop the video we just need the command: *Sudo service motion start/stop*

And to have access to the camera we can connect us to this address: http://192.168.137.81:8081/



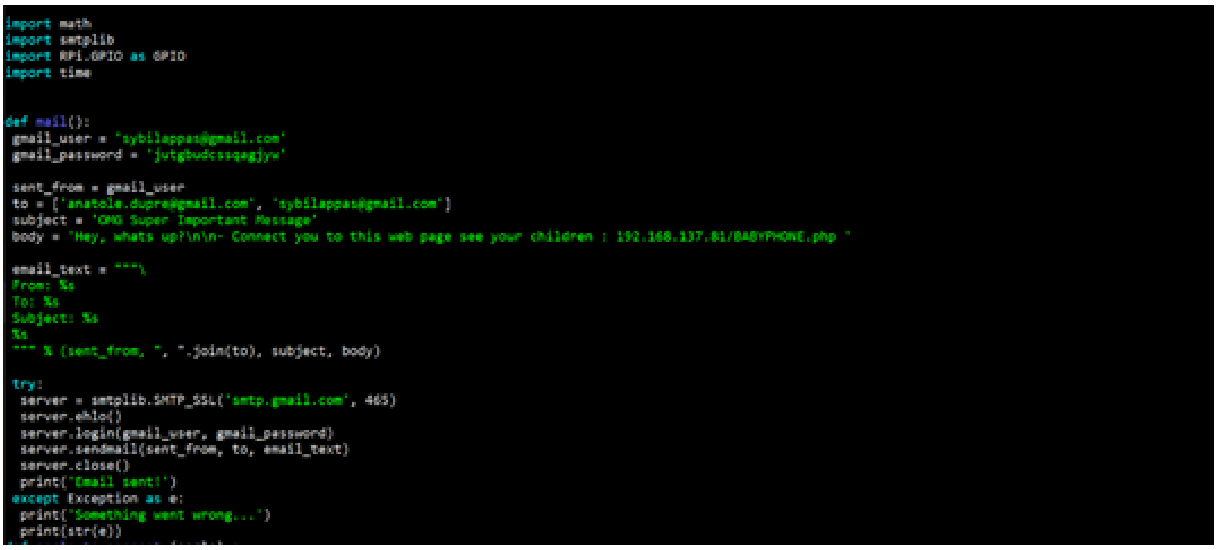
pi@192.168.137.81:22 - Bitvise xterm - pi@raspberrypi: /etc/motion

```
  GNU nano 3.2                     motion.conf

# Rename this distribution example file to motion.conf
#
# This config file was generated by motion 4.1.1
# Documentation:  /usr/share/doc/motion/motion_guide.html

############################################################
# Daemon
############################################################

# Start in daemon (background) mode and release terminal (default: off)
daemon on

# File to store the process ID, also called pid file. (default: not defined)
process_id_file /var/run/motion/motion.pid

############################################################
# Basic Setup Mode
############################################################

# Start in Setup-Mode, daemon disabled. (default: off)
setup_mode off


# Use a file to save logs messages, if not defined stderr and syslog is use$
logfile /var/log/motion/motion.log

# Level of log messages [1..9] (EMG, ALR, CRT, ERR, WRN, NTC, INF, DBG, ALL$
log_level 6

# Filter to log messages by type (COR, STR, ENC, NET, DBL, EVT, TRK, VID, A$
log_type all

###########################################################
# Capture device options
############################################################

# Videodevice to be used for capturing  (default /dev/video0)
# for FreeBSD default is /dev/bktr0
videodevice /dev/video0

                             [ Read 740 lines ]
^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify
^X Exit        ^R Read File   ^\ Replace     ^U Uncut Text  ^T To Spell
```

pi@192.168.137.81:22 - Bitvise xterm - pi@raspberrypi: ~

```
pi@raspberrypi:~ $ sudo service motion stop
pi@raspberrypi:~ $ sudo service motion start
pi@raspberrypi:~ $ 
```

Send an alert by email



```
import math
import smtplib
import RPi.GPIO as GPIO
import time

def mail():
 gmail_user = 'sybilappas@gmail.com'
 gmail_password = 'jutgbudcssqagjyw'

 sent_from = gmail_user
 to = ['anatole.dupre@gmail.com', 'sybilappas@gmail.com']
 subject = 'OMG Super Important Message'
 body = "Hey, whats up?\n\n- Connect you to this web page see your children : 192.168.137.81/BABYPHONE.php "

 email_text = """\
 From: %s
 To: %s
 Subject: %s
 %s
 """ % (sent_from, ", ".join(to), subject, body)

 try:
  server = smtplib.SMTP_SSL('smtp.gmail.com', 465)
  server.ehlo()
  server.login(gmail_user, gmail_password)
  server.sendmail(sent_from, to, email_text)
  server.close()
  print('Email sent!')
 except Exception as e:
  print('Something went wrong...')
  print(str(e))
```

The first function sends an email to the recipient of our choice. The sender here has a gmail address, we had to activate the two steps validation in order to generate a special password that authorise our raspberry to connect and send emails.

For this function we imported the library smtplib that has all the functionalities to send an email. SMTP = Simple Mail Transfer Protocol. SMTP server is used to send emails. When we send an email, it will connect to the SMTP server that is configured and send the text.

Then, the SMTP server will send the mail to the recipient. If the message must pass between different servers, it is always the SMTP protocol that will be used to send the message from servers to servers, the servers then use SMTP relays. (https://www.culture-informatique.net/cest-quoi-un-serveur-smtp/)

To use this library, we have to define what type of sender it is (**sent_from = gmail_user**), we create a list “to” of the email address of the recipients. We also have to create the subject of the email and the “body” of the mail, where we can write what we have to say.

Then we create a variable called "server" created with the instance of the class smtplib : SMTP_SSL An SMTP_SSL instance encapsulates an SMTP connection. It has methods that support a full repertoire of SMTP operations. (https://docs.python.org/3/library/smtplib.html). With this variable we will be able to use all the methods we need to send an email. The Instance SMTP_SSL has as arguments, the type of host (here it is gmail.com) and the port of the server smtp : 465.

Server.Ehlo() : Identify yourself to an ESMTP server using `EHLO`

Server.Login() : Log in on an SMTP server that requires authentication. The arguments are the username and the password to authenticate with (**gmail_user**, **gmail_password**).

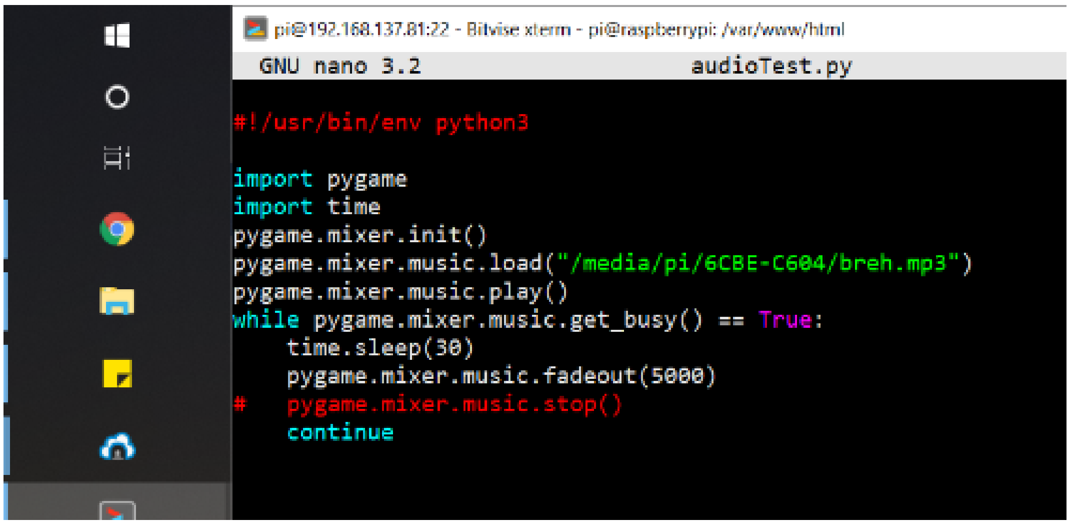
Server.Sendmail() : Send mail. The required arguments : from-address string (here **gmail_user**), a list of to-address strings and a message string (here **to**).

Server.Close() : close de procedure.

We had a real difficulty with the authorisations of our google account to send an e-mail via our raspberry. After a long period of reflection and various tests we have decided to adopt the two steps validation of Gmail to generate a special password.

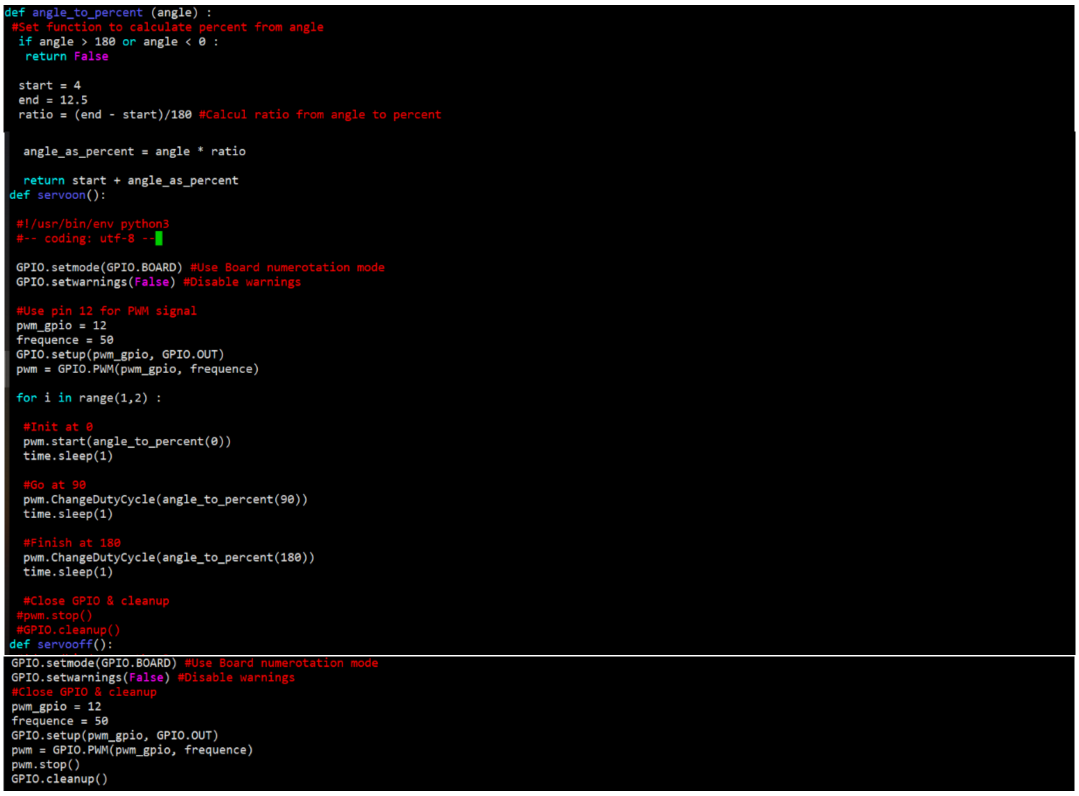
The use of a speaker to play nursery rhyme

In this part we want to play a music through a speaker in order to put the child to sleep. To do that we use the library pygame which is a huge python library using for video games. This library is composed with many sub libraries like mixer adapted for music. Firstly we initialize the mixer module with the command *pygame.mixer.init()* then we download our folder "breh.mp3" which is save on a USB key named *"6CBE-C604"*. Secondly we play the music with the command *pygame.mixer.play()*. Our playlist is quite long so we play the music for thirty seconds with a while and a *time.sleep(30).* When the song has been played for thirty seconds the volume decreases little by little during 5 seconds until the song stops playing.



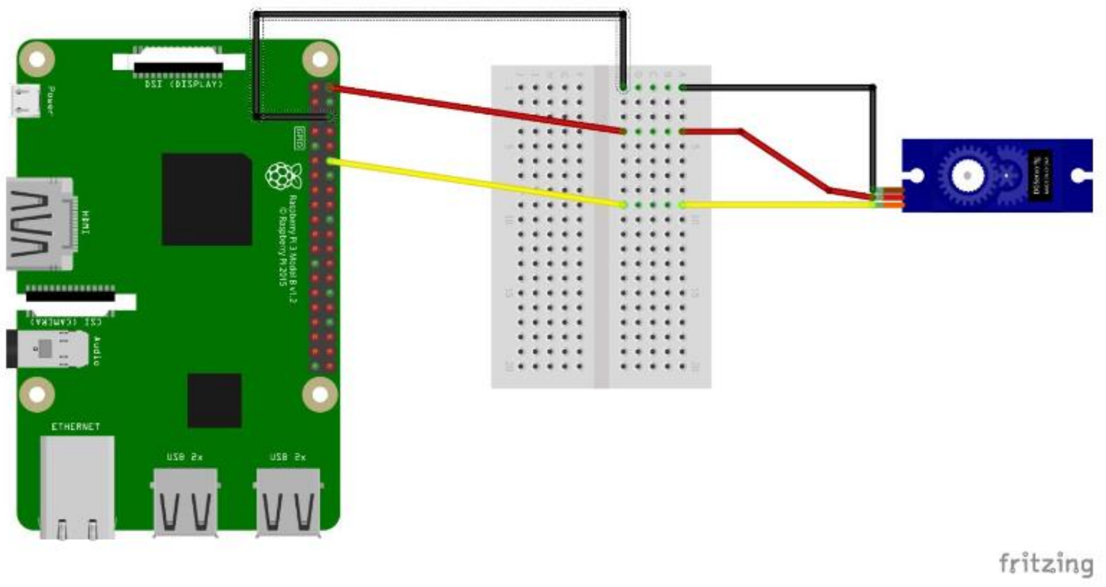
```
pi@192.168.137.81:22 - Bitvise xterm - pi@raspberrypi: /var/www/html
  GNU nano 3.2                              audioTest.py

#!/usr/bin/env python3

import pygame
import time
pygame.mixer.init()
pygame.mixer.music.load("/media/pi/6CBE-C604/breh.mp3")
pygame.mixer.music.play()
while pygame.mixer.music.get_busy() == True:
    time.sleep(30)
    pygame.mixer.music.fadeout(5000)
#    pygame.mixer.music.stop()
    continue
```

The use of a servomotor to rock the baby



```
def angle_to_percent (angle) :
 #Set function to calculate percent from angle
  if angle > 180 or angle < 0 :
   return False

  start = 4
  end = 12.5
  ratio = (end - start)/180 #Calcul ratio from angle to percent

   angle_as_percent = angle * ratio

   return start + angle_as_percent
def servoon():

  #!/usr/bin/env python3
  #-- coding: utf-8 --

  GPIO.setmode(GPIO.BOARD) #Use Board numerotation mode
  GPIO.setwarnings(False) #Disable warnings

  #Use pin 12 for PWM signal
  pwm_gpio = 12
  frequence = 50
  GPIO.setup(pwm_gpio, GPIO.OUT)
  pwm = GPIO.PWM(pwm_gpio, frequence)

  for i in range(1,2) :

   #Init at 0
   pwm.start(angle_to_percent(0))
   time.sleep(1)

   #Go at 90
   pwm.ChangeDutyCycle(angle_to_percent(90))
   time.sleep(1)

   #Finish at 180
   pwm.ChangeDutyCycle(angle_to_percent(180))
   time.sleep(1)

   #Close GPIO & cleanup
  #pwm.stop()
  #GPIO.cleanup()
def servooff():
 GPIO.setmode(GPIO.BOARD) #Use Board numerotation mode
 GPIO.setwarnings(False) #Disable warnings
 #Close GPIO & cleanup
 pwm_gpio = 12
 frequence = 50
 GPIO.setup(pwm_gpio, GPIO.OUT)
 pwm = GPIO.PWM(pwm_gpio, frequence)
 pwm.stop()
 GPIO.cleanup()
```

Three functions were created, one to turn on the servomotor, another to turn it off and the last one defines the angle of the servomotor. We had to import the library RPI.GPIO that contains all the functionalities we need to manipulate a GPIO pin.



- **Angle_to_percent(angle)** : this function converts the rotation angle into a percentage of duty cycle.

- **Servoon()** : The Raspberry Pi allows two numberings, that of the screen printing of the card connector (GPIO.BOARD), or the electronic numbering of the chip (GPIO.BCM). Here we choose the GPIO.BOARD with the method setmode(GPIO.BOARD). Then we use the method GPIO.setwarnings() which allows error messages to be deleted. The gpio is set to pin 12 and OUT, so that the raspberry sends the information to the servo motor. Then with GPIO.PWM (PWM = Pulse Width Modulation) allows to give very fast impulses to the motor. This is why we set a frequency, because GPIO.PWM takes as arguments, the gpio pin and the frequency. The servo angle is controlled according to the percentage of the duty cycle during which the signal is ON.

  GPIO.start(angle_to_percent(0)) allows us to initialize the PWM object with a duty cycle of 0% (angle 0°). http://bricolosciences.over-blog.fr/2017/03/controler-un-servo-en-python-avec-un-raspberrypi.html

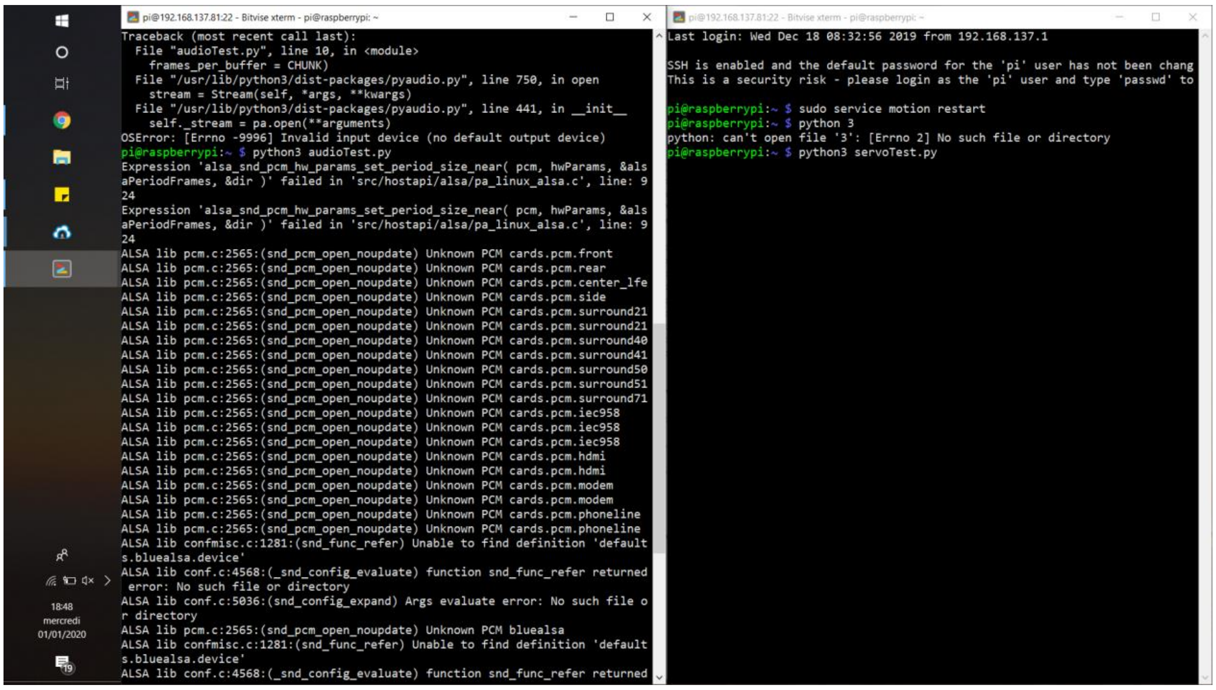
  GPIO.changedutycycle(angle_to_percent(90)) allows us to change the duty cycle to change the servo angle.

- **Servooff()** : Turn off the servo by stopping the PWM and reset any ports we have used in this program back to input mode with GPIO.cleanup().

We have faced some difficulties with our servo motor, we did not know of it was working, so we first tested it with an Arduino program. Then when we knew our servo motor was working and we had to look for the problem in our python program.

How we gathered all the codes to have a functional prototype

As you can see, we decided to code every function in Python language to make it easier to group in a single program. In order to make the video work at the same time than the microphone of the webcam, we have to run two different programs on two different terminals.



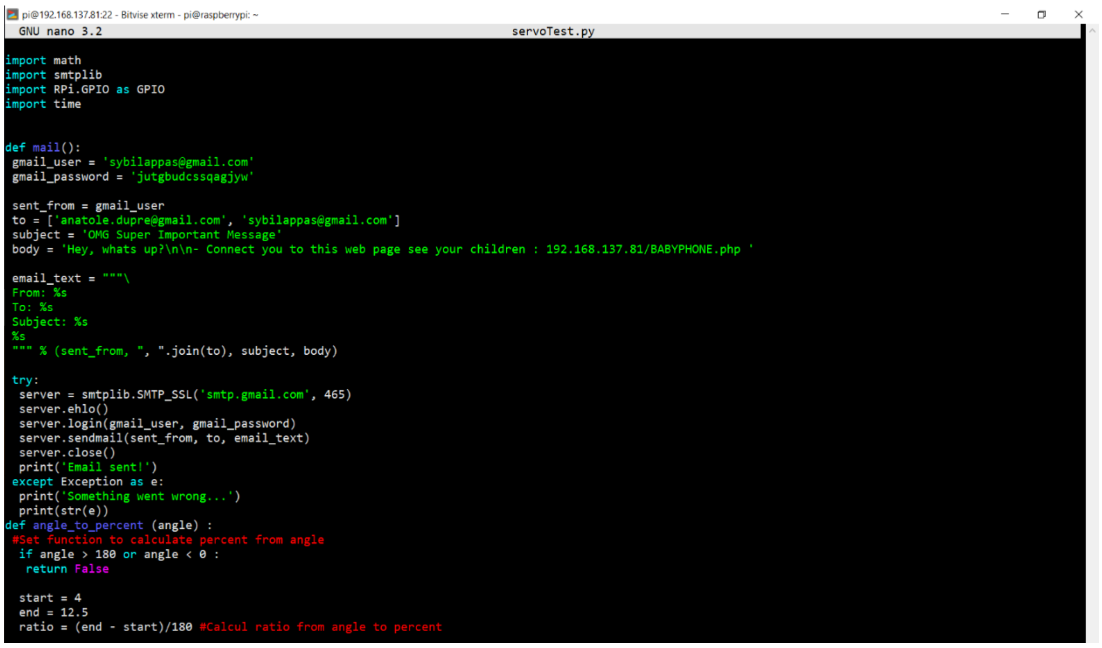
pi@192.168.137.81:22 - Bitvise xterm - pi@raspberrypi: ~

```
Traceback (most recent call last):
  File "audioTest.py", line 10, in <module>
    frames_per_buffer = CHUNK)
  File "/usr/lib/python3/dist-packages/pyaudio.py", line 750, in open
    stream = Stream(self, *args, **kwargs)
  File "/usr/lib/python3/dist-packages/pyaudio.py", line 441, in __init__
    self._stream = pa.open(**arguments)
OSError: [Errno -9996] Invalid input device (no default output device)
pi@raspberrypi:~ $ python3 audioTest.py
Expression 'alsa_snd_pcm_hw_params_set_period_size_near( pcm, hwParams, &als
aPeriodFrames, &dir )' failed in 'src/hostapi/alsa/pa_linux_alsa.c', line: 9
24
Expression 'alsa_snd_pcm_hw_params_set_period_size_near( pcm, hwParams, &als
aPeriodFrames, &dir )' failed in 'src/hostapi/alsa/pa_linux_alsa.c', line: 9
24
ALSA lib pcm.c:2565:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.front
ALSA lib pcm.c:2565:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.rear
ALSA lib pcm.c:2565:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.center_lfe
ALSA lib pcm.c:2565:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.side
ALSA lib pcm.c:2565:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.surround21
ALSA lib pcm.c:2565:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.surround21
ALSA lib pcm.c:2565:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.surround40
ALSA lib pcm.c:2565:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.surround41
ALSA lib pcm.c:2565:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.surround50
ALSA lib pcm.c:2565:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.surround51
ALSA lib pcm.c:2565:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.surround71
ALSA lib pcm.c:2565:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.iec958
ALSA lib pcm.c:2565:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.iec958
ALSA lib pcm.c:2565:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.iec958
ALSA lib pcm.c:2565:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.hdmi
ALSA lib pcm.c:2565:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.hdmi
ALSA lib pcm.c:2565:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.modem
ALSA lib pcm.c:2565:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.modem
ALSA lib pcm.c:2565:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.phoneline
ALSA lib pcm.c:2565:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.phoneline
ALSA lib confmisc.c:1281:(snd_func_refer) Unable to find definition 'default
s.bluealsa.device'
ALSA lib conf.c:4568:(_snd_config_evaluate) function snd_func_refer returned
 error: No such file or directory
ALSA lib conf.c:5036:(snd_config_expand) Args evaluate error: No such file o
r directory
ALSA lib pcm.c:2565:(snd_pcm_open_noupdate) Unknown PCM bluealsa
ALSA lib confmisc.c:1281:(snd_func_refer) Unable to find definition 'default
s.bluealsa.device'
ALSA lib conf.c:4568:(_snd_config_evaluate) function snd_func_refer returned
```

pi@192.168.137.81:22 - Bitvise xterm - pi@raspberrypi: ~

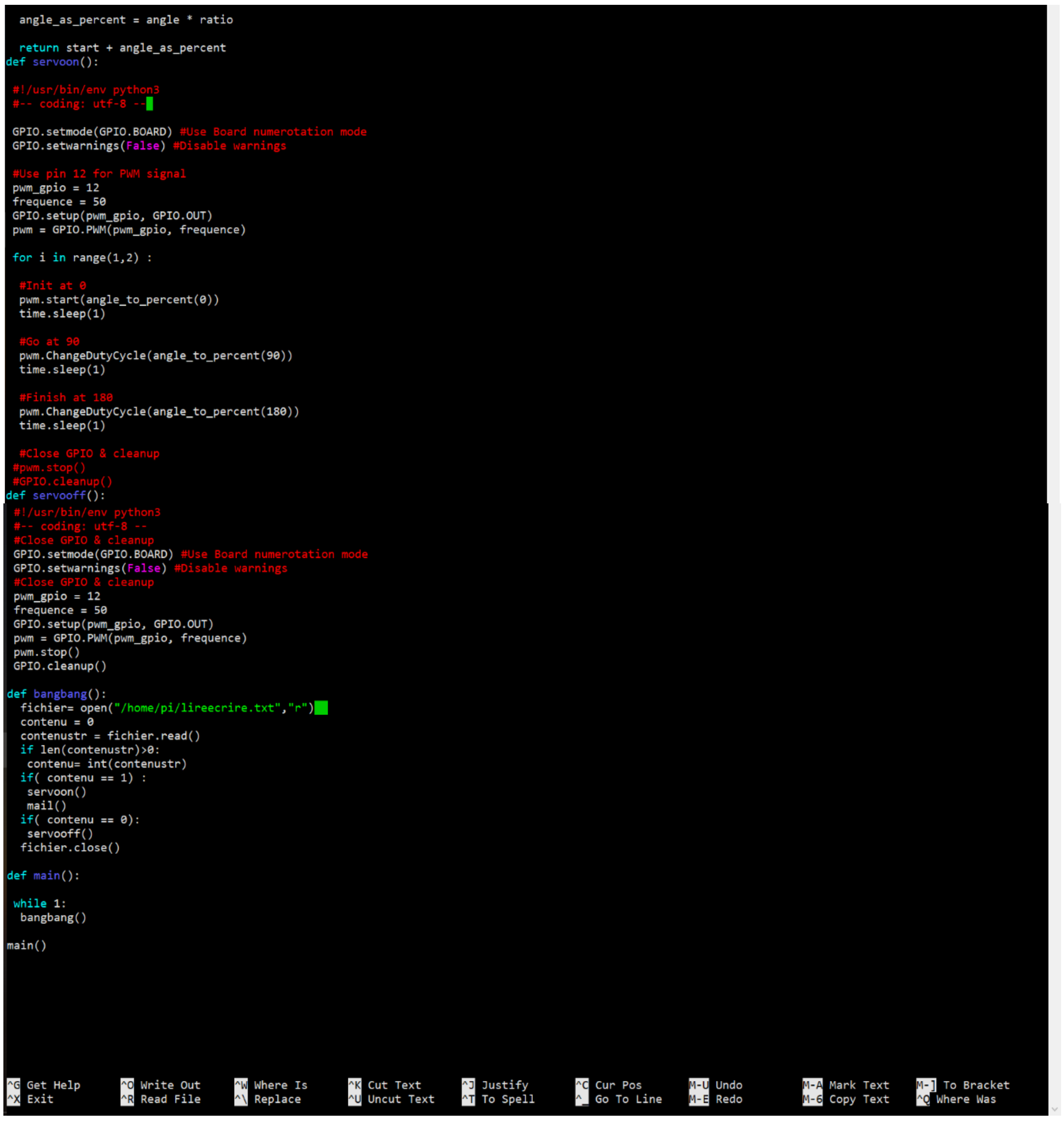
```
Last login: Wed Dec 18 08:32:56 2019 from 192.168.137.1

SSH is enabled and the default password for the 'pi' user has not been chang
This is a security risk - please login as the 'pi' user and type 'passwd' to

pi@raspberrypi:~ $ sudo service motion restart
pi@raspberrypi:~ $ python 3
python: can't open file '3': [Errno 2] No such file or directory
pi@raspberrypi:~ $ python3 servoTest.py
```

As you can see on the left we run the program **audioTest.py** (explained above) to record the sound of the baby. On the right we run the program **ServoTest.py** :



pi@192.168.137.81:22 - Bitvise xterm - pi@raspberrypi: ~

```
  GNU nano 3.2                                servoTest.py

import math
import smtplib
import RPi.GPIO as GPIO
import time


def mail():
 gmail_user = 'sybilappas@gmail.com'
 gmail_password = 'jutgbudcssqagjyw'

 sent_from = gmail_user
 to = ['anatole.dupre@gmail.com', 'sybilappas@gmail.com']
 subject = 'OMG Super Important Message'
 body = 'Hey, whats up?\n\n- Connect you to this web page see your children : 192.168.137.81/BABYPHONE.php '

 email_text = """\
 From: %s
 To: %s
 Subject: %s
 %s
 """ % (sent_from, ", ".join(to), subject, body)

 try:
  server = smtplib.SMTP_SSL('smtp.gmail.com', 465)
  server.ehlo()
  server.login(gmail_user, gmail_password)
  server.sendmail(sent_from, to, email_text)
  server.close()
  print('Email sent!')
 except Exception as e:
  print('Something went wrong...')
  print(str(e))
def angle_to_percent (angle) :
 #Set function to calculate percent from angle
  if angle > 180 or angle < 0 :
   return False

  start = 4
  end = 12.5
  ratio = (end - start)/180 #Calcul ratio from angle to percent
```



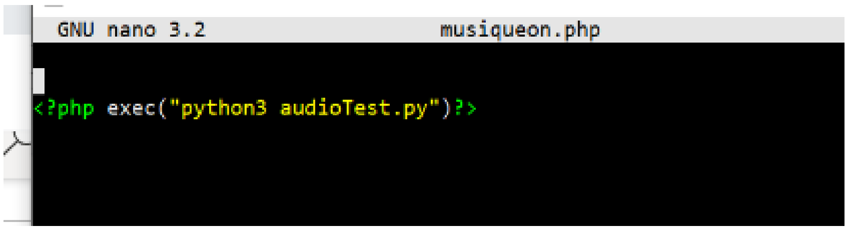
```
  angle_as_percent = angle * ratio

  return start + angle_as_percent
def servoon():

 #!/usr/bin/env python3
 #-- coding: utf-8 --

 GPIO.setmode(GPIO.BOARD) #Use Board numerotation mode
 GPIO.setwarnings(False) #Disable warnings

 #Use pin 12 for PWM signal
 pwm_gpio = 12
 frequence = 50
 GPIO.setup(pwm_gpio, GPIO.OUT)
 pwm = GPIO.PWM(pwm_gpio, frequence)

 for i in range(1,2) :

  #Init at 0
  pwm.start(angle_to_percent(0))
  time.sleep(1)

  #Go at 90
  pwm.ChangeDutyCycle(angle_to_percent(90))
  time.sleep(1)

  #Finish at 180
  pwm.ChangeDutyCycle(angle_to_percent(180))
  time.sleep(1)

  #Close GPIO & cleanup
 #pwm.stop()
 #GPIO.cleanup()
def servooff():
 #!/usr/bin/env python3
 #-- coding: utf-8 --
 #Close GPIO & cleanup
 GPIO.setmode(GPIO.BOARD) #Use Board numerotation mode
 GPIO.setwarnings(False) #Disable warnings
 #Close GPIO & cleanup
 pwm_gpio = 12
 frequence = 50
 GPIO.setup(pwm_gpio, GPIO.OUT)
 pwm = GPIO.PWM(pwm_gpio, frequence)
 pwm.stop()
 GPIO.cleanup()

def bangbang():
  fichier= open("/home/pi/lireecrire.txt","r")
  contenu = 0
  contenustr = fichier.read()
  if len(contenustr)>0:
   contenu= int(contenustr)
  if( contenu == 1) :
   servoon()
   mail()
  if( contenu == 0):
   servooff()
  fichier.close()

def main():

 while 1:
  bangbang()

main()
```

In this program we have grouped together the mail and servo motor functions in the function **bangbang().** The difficulty we had here was that we had to find a solution so that the buffer in our **audioTest()** function is not full after only a few second. As explained above, this is why in the **audioTest()** function we write 0 or 1 in a file depending on the sound level. In the **banbang()** function we read this file, and if we read 1, we are able to run the servomotor and send an e-mail. To be able to recognise the number written on the file, after reading it with **fichier.read()**, we had to convert it into an integer with the **int()** method. Finally we call the **bangbang()** function in the **main().**

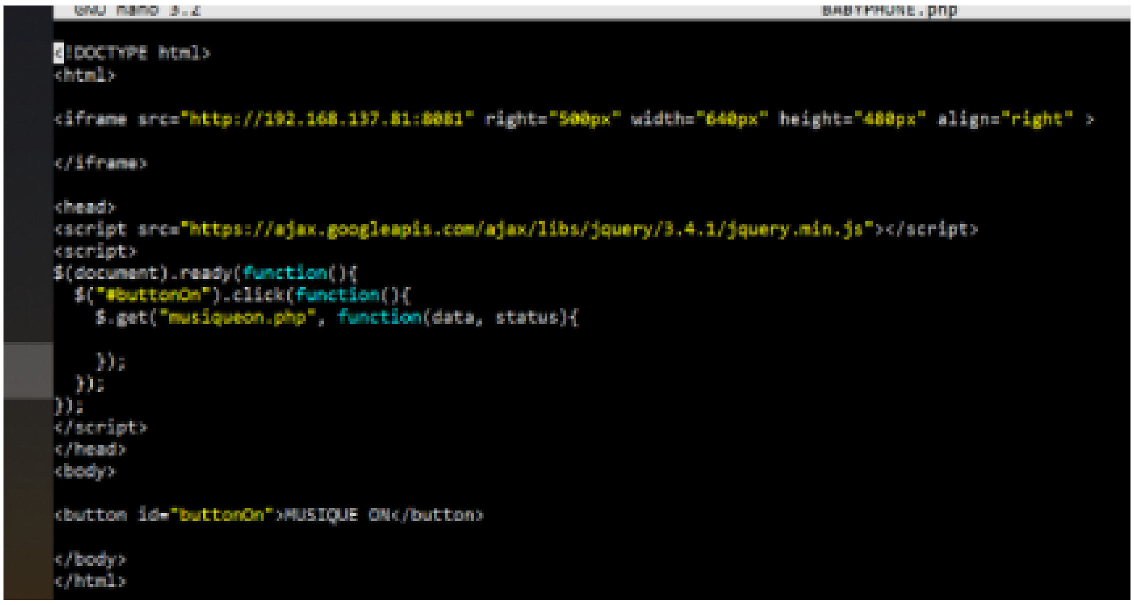
Creation of a web page for the user experience

We decided to create a web page to improve the users experience. The goal of this web page is to allow parents to see their child and if the child is awake, they can play a song through the speaker. In the code part we reuse the same structure of our previous code with the button on and off to switch on and off a LED. To integrate the image of the webcam in this web page we use an IFrame which allow us to add a window into another. In the code line we just write the IP address of the web page that we want to integrate, in our case is the address : *http://192.168.137.81:8081*. Then we add some parameters like the definition of the image, the size, and the place, here we choose to put the webcam on the right.

Then in the following part of the program I will define what's gone happened when I click on the button. For that I used the function *click(function(){}* in this line I say that when I will click on the button : *"buttonOn"* the following instructions will happened, here I call the PHP web page: "musiqueon.php". In this code I just execute the code who play a song that I explained before. To use all this function a add previously a specific library: https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js



```
  GNU nano 3.2                    musiqueon.php

<?php exec("python3 audioTest.py")?>
```



```
  GNU nano 3.2                                                      BABYPHONE.php

<!DOCTYPE html>
<html>

<iframe src="http://192.168.137.81:8081" right="500px" width="640px" height="480px" align="right" >

</iframe>

<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("#buttonOn").click(function(){
    $.get("musiqueon.php", function(data, status){

    });
  });
});
</script>
</head>
<body>

<button id="buttonOn">MUSIQUE ON</button>

</body>
</html>
```

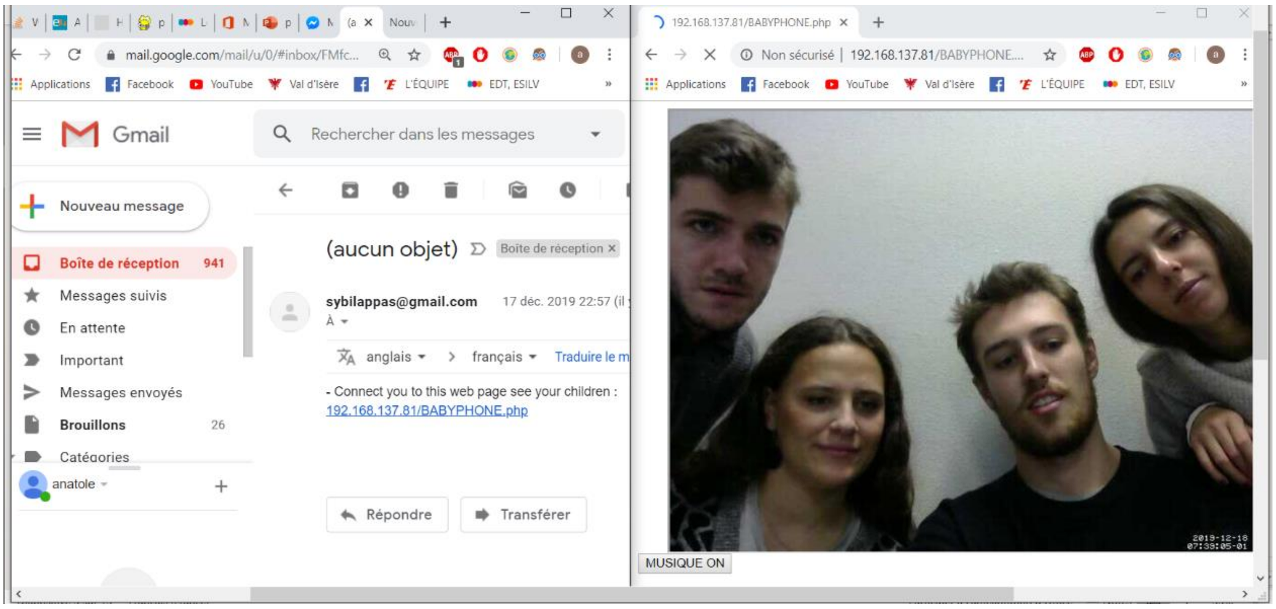
In this part we had a lot of problem to execute the python code through a php page due to some permission. Firstly, we had to give the users the permission to have access to the usb key and the music. Then we also have to add the user www-data to audio with the command: *sudo nano adduser www-data audio.* To see the different permission, we use the command: *cat /etc/group**

All the three programs BABYPHONE.php, musiqueon.php and audioTest.py are saved in the following folder */var/www/html* in order that the web page works and that the programs can call the others.

Conclusion

The project has been well realized and we are proud of teamwork that leads to our connected baby phone, howsoever we noted some areas of improvement:

It could be great if parents could set the sound-level threshold easily, to be sure that they will be awaken by baby crying and not by something else. A Spotify account connection would be useful to play some playlists for baby, to make him asleep with his favourite songs. The user experience would be better with a high-resolution webcam, which will allow parents to see if baby is sleeping well, and if everything is alright in baby's room.



(aucun objet)

sybilappas@gmail.com 17 déc. 2019 22:57

- Connect you to this web page see your children :
192.168.137.81/BABYPHONE.php