

Supervise patient movements in a hospital

A Project made by: Ahmed Mahmoud Ebou El Hassani, Assma Benkaddour, Ghizlane El hirech, Hajar El Khanfri, Hamza Benjelloun, Kawtar Essebani, Oumaima Saber, Oumaima Safi and Youness Jabar.

Departement: Computer science, University Ibn Tofail

National School of applied sciences

Kenitra, Morocco

Introduction:

Health Tracker web application is a platform that guarantee a fast response to emergency calls thanks to numerous alarm systems and a web server page that updates data when the database changes ensuring a track of patients each frame of seconds to lower the max possible of risk's rate.

Our platform is made using **Django** a high-level Python Web framework that encourages rapid development and clean, pragmatic design.

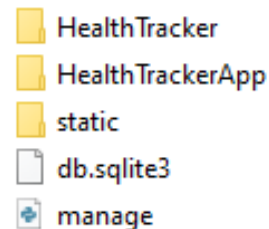
First, we created our project with the command:

```
django-admin startproject  
HealthTracker
```

Then we created our application with the command:

```
django-admin startapp  
HealthTrackerApp
```

Also, we created a new folder for static files with the name static:

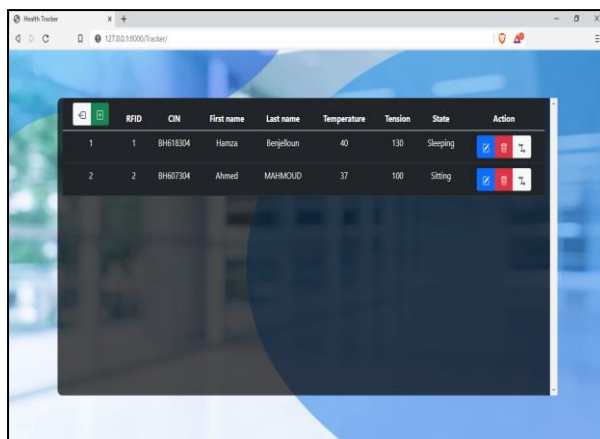


Our project is based on two main pages:

Index (main page)



Track Page (Control Page)



The screenshot shows a web browser window with the URL '127.0.0.1:8000/Tracker/'. The page displays a table with patient information. The table has columns for ID, RFID, CIN, First name, Last name, Temperature, Tension, State, and Action. There are two rows of data.

	RFID	CIN	First name	Last name	Temperature	Tension	State	Action
1	1	BH618304	Hamza	Bejjelloun	40	130	Sleeping	[Icons]
2	2	BH607304	Ahmed	MAHMOUD	37	100	Sitting	[Icons]

Index

It's a page where the admin needs to login and access to the other page.

If the username or password posted is wrong an alert message is shown to tell the user to verify his inputs.



But if the username or password are correct the user is redirected to the track page.

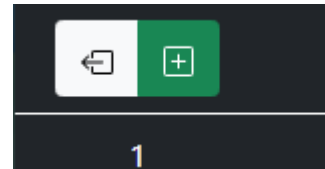
To do these treatments we use the function home defined in view.py

This function is declared each time when the page is load or accessed via the path declared in a file named urls.py

To explain well the project's concept let's play the role of a doctor:

After login with the correct username and password, I have to add the patient.

To do these I need to use the green button with plus sign in the track page.

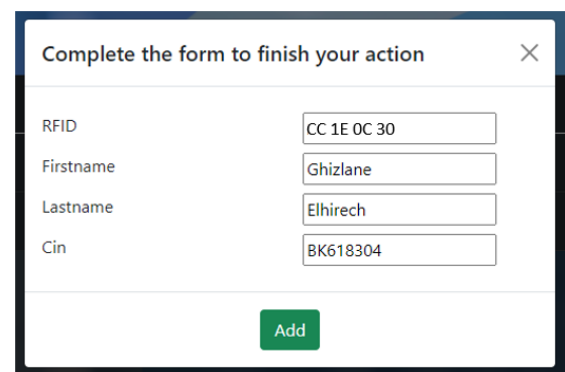


This will show a modal with four fields:

- RFID
- First Name
- Last Name
- CIN

To add a patient, I can enter his data automatically in the fields, but his RFID is unknown.

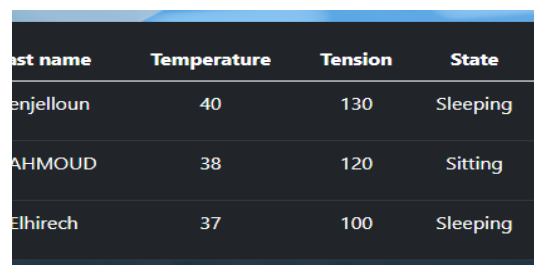
So, I am going to scan his RFID and with some codes in the back end the field of RFID will get automatically the ID of patient, then I will complete the other fields and submit.



A modal form titled 'Complete the form to finish your action'. It contains four input fields: RFID (CC 1E 0C 30), Firstname (Ghizlane), Lastname (Elhirech), and Cin (BK618304). There is a green 'Add' button at the bottom.

When clicking on add not only the database is updated but also the functions that calculate the tension, temperature and state are automatically activated.

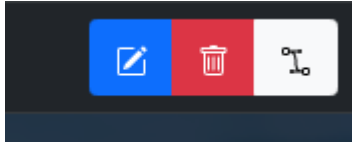
But of course, the data will be sent to our page once the patient enter to his own room.



The screenshot shows a table with patient data. The table has columns for Last name, Temperature, Tension, and State. There are three rows of data.

Last name	Temperature	Tension	State
enjjelloun	40	130	Sleeping
MAHMOUD	38	120	Sitting
Elhirech	37	100	Sleeping

We can delete or edit each patient from the blue and red buttons on the right side, also we can show up a modal containing temperature evolution and tension, state estimation by clicking on the white button on the right side of the desired patient.



After for example deleting the patient ghizlane we can see that the patient is successfully deleted from the database and the web page.

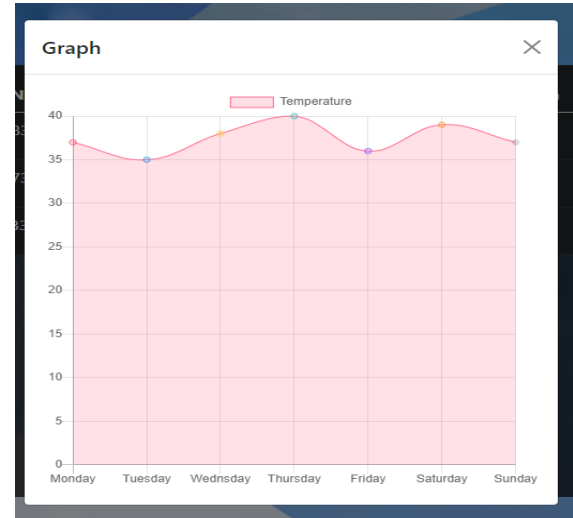
	RFID	CIN	First name	Last name	Temperature	Tension	State	Action
1	1	BH418104	Hanna	Benjeloun	40	130	Sleeping	  
2	2	BH607301	Ahmed	MAIMOOD	38	120	Sitting	  

The concept of deleting or editing use URL handling which means that after one of these two actions (edit or delete) we send a request to a map with the ID of a patient.

With the functions in views, we collect the id and we use the repositories to get the user by his ID (ID sent already as string),

and then the function does the treatment and save changes.

Also, here is an example of an evolution graph modal:



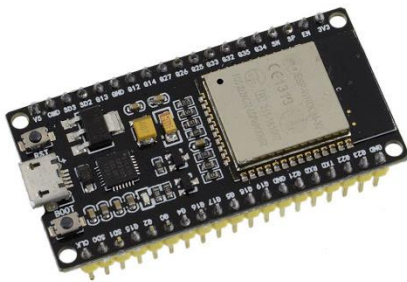
Nodes IOT:

Our project contains three nodes; the first one for patient identification, the second one for measuring identified patient's temperature and pressure and the third one for monitoring patient's movements.

- **Patient Identification Node:**

In this node, we used the following material:

ESP32:



ESP32 is a series of microcontrollers type of system on a chip (SoC) of Espressif Systems, based on the architecture of Xtensa LX6 Tensilica (in), integrating the management of Wi-Fi and Bluetooth (5.0 and up THE 5.1 1) in dual mode, and a DSP . It is an evolution of ESP8266.

RFID Reader:

Radio Frequency Identification (RFID) is the wireless non-contact use of radio frequency waves to transfer data. Tagging items with RFID tags allows users to automatically and uniquely identify and track inventory and assets. RFID takes auto-ID technology to the next level by allowing tags to be read without line of sight and, depending on the type of RFID, having a read range between a few centimeters to over 20+ meters.



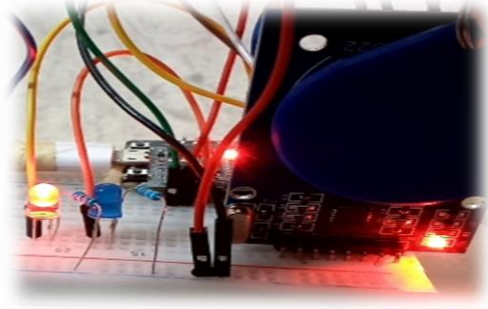
LEDs and Resistances



This node is used for reading the User Id from the patient's card. If the UID exists in our database, which means that this person is a patient in our hospital, he could measure his temperature and pressure using the following node. In that case, the blue LED will be turned on:



Otherwise, the red one will be turned on:



Also, we use this node while adding a new patient in our hospital's database. We write his information in his card also

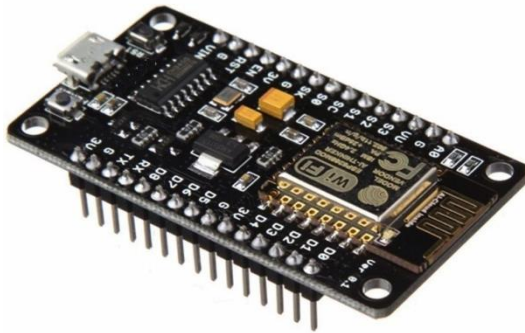
- **Measuring Patient's Temperature and Pressure Node:**

In this node, we used the following material:

ESP8266:

ESP8266 is a wifi SOC (system on a chip) produced by Espressif Systems. It is a highly integrated chip designed to provide

full internet connectivity in a small package.



ambient temperature or the temperature of a specific object.



Buzzer:

A buzzer or beeper is an electromechanical or piezoelectric element which produces a characteristic sound when a voltage is applied to it: the beep. Some require DC voltage, others require AC voltage.

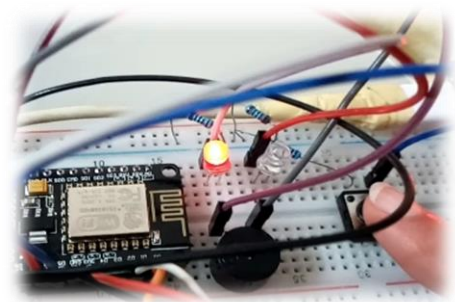


Identified patients in the previous node could measure their temperature. This patient should first click on the push button and then he should put his forehead near the temperature sensor. After measuring his temperature, we estimate his pressure, and we will see if that patient has a normal temperature and pressure. If not; we mean, if the patient had a Low or high temperature, the buzzer and a red LED will turn on.

Push Button:

Push button is a simple switch mechanism to control some aspect of a machine or a process.

Buttons are typically made out of hard material, usually plastic or metal. The surface is usually flat or shaped to accommodate the human finger or hand, so as to be easily depressed or pushed.



• Monitoring Patient's Movements Node:

In this node, we used the following material:

Raspberry Pi 3 B+:

The Raspberry Pi 3 B + is a single board computer that can connect to a monitor, a keyboard / mouse assembly and has Wi-Fi and Bluetooth interfaces.

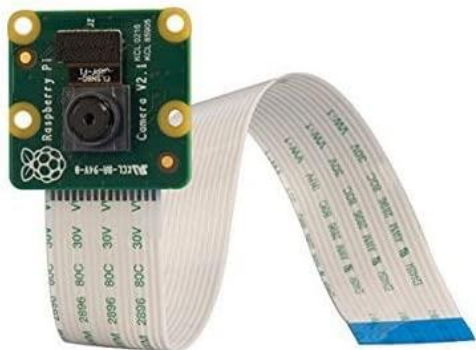
It boots from a micro-SD card and runs on a Linux or Windows 10 IoT OS. It is

supplied without case, power supply, keyboard, screen and mouse in order to reduce the cost and promote the use of recovery equipment.

The Raspberry Pi3 B + model is based on an ARM Cortex-A53 64-bit quad-core 1.4 GHz processor, has 1 GB of RAM memory, a Wi-Fi interface, a Bluetooth interface, 4 USB ports, an Ethernet port, a HDMI port, a micro-SD port and a GPIO connector with 40 I / O pins.



Pi Camera Module:

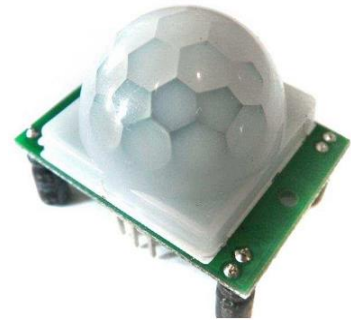


1 camera module is a portable light weight camera that supports Raspberry Pi. It communicates with Pi using the MIPI camera serial interface protocol. It is normally used in image processing, machine learning or in surveillance projects.

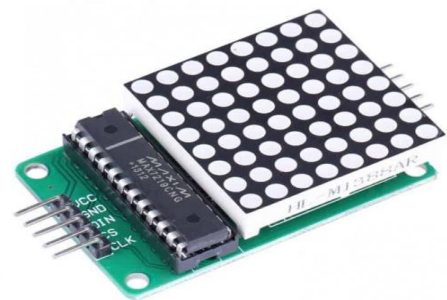
PIR Sensor:

PIR Sensor is short for passive infrared sensor, which applies for projects that need

to detect human or particle movement in a certain range. It is also known as PIR (motion) sensor or IR sensor.

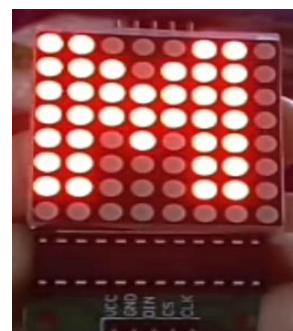


LED Matrix 8*8:

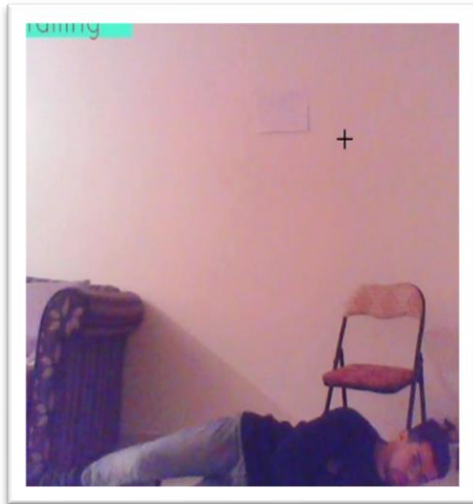


It's a matrix that contains 64 LEDs.

This node consists to detect patient's movements by PIR sensor, if our patient is moving; we turn on the LED Matrix's LEDs:



Also, we will turn on the raspberry camera module to see either the patient is falling:



And finally, we will send this information to the database.

I. IMAGE PROCESSING

Human Pose estimation is an important problem that has enjoyed the attention of the Computer Vision community for the past few decades. It is a crucial step towards understanding people in images and videos. We have implemented our project with python as a programming language with the library for real-time computer vision OpenCV. **Human Pose Estimation** is defined as the problem of localization of human joints (also known as keypoints - elbows, wrists, etc) in images or videos. It is also defined as the search for a specific pose in space of all articulated poses. Human Pose Estimation has some pretty cool applications and is heavily used in Action recognition, Animation, Gaming, etc . For example, a very popular Deep Learning app HomeCourt uses Pose Estimation to analyse Basketball player movements. And our project is also an application of it.

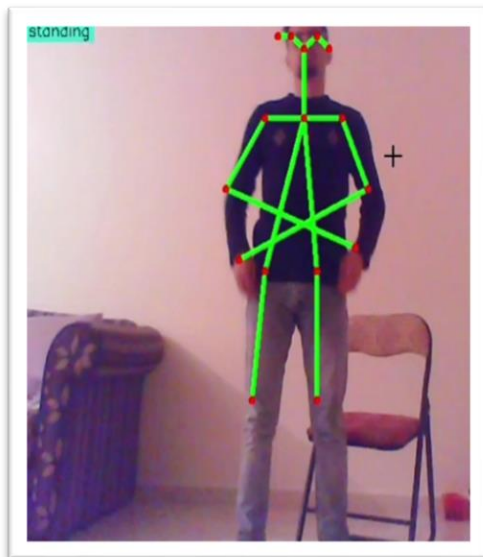
In nutshell, human pose estimation is the localization of the body parts.

We have worked with a tensorflow model trained to know the human pose estimation so we can load the frozen models using opencv , and for overlaying these lines and points correspondingly we will be using opencv in addition our algorithm can successfully estimate the poses : standing ,siting, and falling.

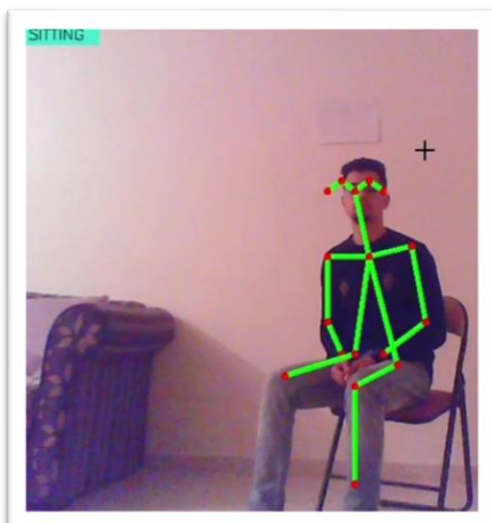
you can see that as long as we have the full view of a body the human pose estimation algorithm can detect all possible 18 points of the body skeleton, similarly in this case we have very less points body with the help of tensorflow deep learning architecture is still it manages to detect all possible points

```
BODY_PARTS = { "Nose": 0,
               "Neck": 1,
               "RShoulder": 2,
               "RElbow": 3,
               "RWrist": 4,
               "LShoulder": 5,
               "LElbow": 6,
               "LWrist": 7,
```

Or standing:



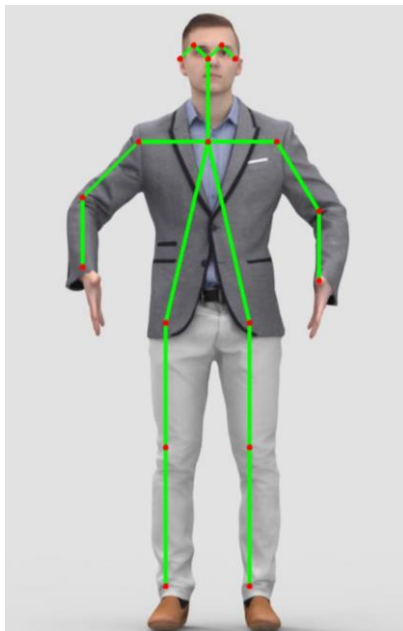
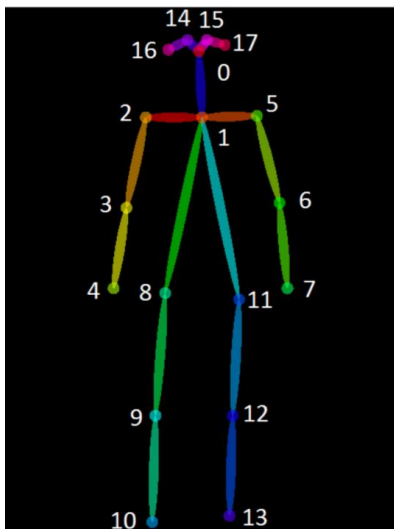
Or sitting:



```

"RHip": 8,
"RKnee": 9,
"RAnkle": 10,
"LHip": 11,
"LKnee": 12,
"LAnkle": 13,
"REye": 14,
"LEye": 15,
"REar": 16,
"LEar": 17,
"Background": 18 }

```



II. OPEN CV

OpenCV is the huge open-source library for the computer vision, machine learning, and image processing and now it plays a major role in real-time operation which is very important in today's systems. By using it, one can process images and videos to identify objects, faces, or even

handwriting of a human. When it integrated with various libraries, such as Numpy, python is capable of processing the OpenCV array structure for analysis. To Identify image pattern and its various features we use vector space and perform mathematical operations on these features.

III. HOW TO INSTALL OPENCV

Step 1: Make Pi up-to-date

Update existing packages.

```
sudo apt-get update
```

Upgrade existing Pi software.

```
sudo apt-get upgrade
```

Update Raspberry Pi firmware.

```
sudo rpi-update
```

Step 2: Install Dependencies

- 1) `sudo apt-get install build-essential cmake pkg-config`
- 2) `sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev libpng12-dev`
- 3) `sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev`
- 4) `sudo apt-get install libxvidcore-dev libx264-dev`
- 5) `sudo apt-get install libgtk2.0-dev libgtk-3-dev`
- 6) `sudo apt-get install libatlas-base-dev gfortran`

Step 3: Setup Python 3 development tools

```
sudo apt-get install python3 python3-setuptools python3-dev
```

Step 4: Setup pip tool

- 1) `wget https://bootstrap.pypa.io/get-pip.py`
- 2) `sudo python3 get-pip.py`

Step 5: Grab OpenCV 3.4.1 and OpenCV-contrib archives

- 1) `cd ~`
- 2) `wget Oopencv.zip`
`https://github.com/Itseez/opencv/archive/3.4.1.zip`
- 3) `wget Oopencv_contrib.zip`
`https://github.com/Itseez/opencv_contrib/archive/3.4.1.zip`
- 4) `unzip opencv.zip`
- 5) `unzip opencv_contrib.zip`

Step 6: Install numpy

```
sudo pip3 install numpy
```

Step 7: Build OpenCV

- 1) `cd ~ / opencv-3.4.1 /`
- 2) `mkdir build`
- 3) `cd build`
- 4) `cmake -D CMAKE_BUILD_TYPE = RELEASE \`

- 5) `-D CMAKE_INSTALL_PREFIX = /usr/local \`
- 6) `-D INSTALL_PYTHON_EXAMPLES = ON \`
- 7) `-D OPENCV_EXTRA_MODULES_PATH = ~/opencv_contrib-3.4.1/modules \`
- 8) `-D ENABLE_PRECOMPILED_HEADERS = OFF \`
- 9) `-D BUILD_EXAMPLES = ON..`

Step 8: Increase swap space

We are going to increase swap size from 100MB to 1024MB, to facilitate compilation of OpenCV on all

four cores of Pi.

Open / etc / dphys-swapfile using nano editor.

Change

the value `CONF_SWAPSIZE` variable to 1024.

Save the edited file and activate the new swap space

using the following commands to restart the service

- 1) `sudo /etc/init.d/dphys-swapfile stop`
- 2) `sudo /etc/init.d/dphys-swapfile start`

Step 9: Compile and Install OpenCV

We are going to compile OpenCV on all four cores. Issue

the following command and compilation should succeed within 2 hours.

- 1) `make -j4`

All that is left is, installing OpenCV using the following commands.

- 1) `sudo make install`
- 2) `sudo ldconfig`

Step 10: OpenCV Installation Test

Fire up the Python 3 interpret on command line and run

the following Python code.

- 1) `import cv2`
- 2) `cv2.__version__`

The python code should output '3.4.1', which indicates

that the installation is perfectly fine.

Remove the downloaded zip files to free up space

- 1) `cd ~`
- 2) `rm -rf opencv.zip opencv_contrib.zip`

Revert back the swap size to its original value of 100MB

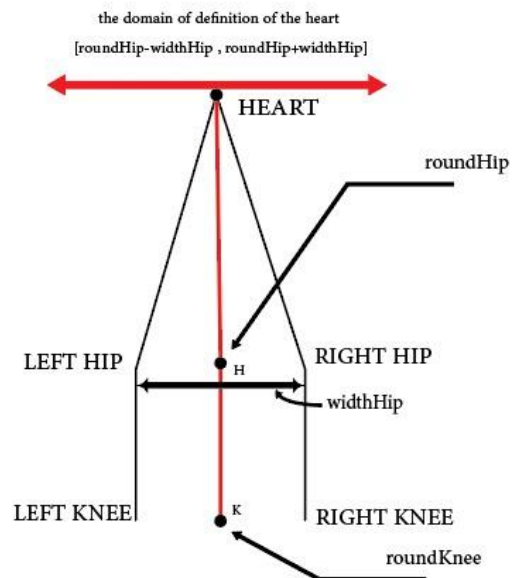
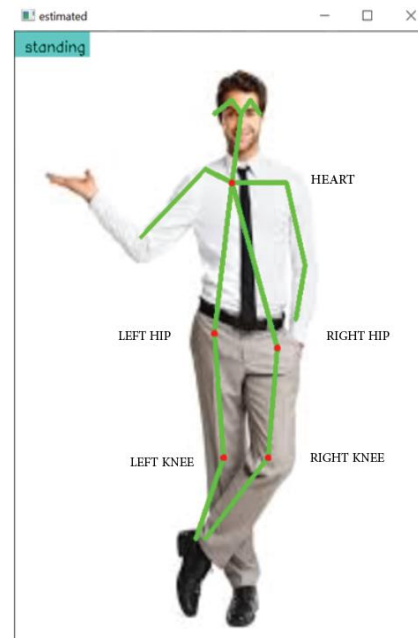
by editing the / etc / dphys-swapfile. And, restart the

service by issuing commands.

- 1) `sudo /etc/init.d/dphys-swapfile stop`
- 2) `sudo /etc/init.d/dphys-swapfile start`

IV. PROGRAM DESCRIPTION OF IMAGE PROCESSING

The determination of the points is not that easy, sometimes the model cannot determine all the points because of some impediments, The treatment that we did focus on these five points on the picture below, we try to find the 3 major points: the heart, the hip and the knee



- The **first major** point is the heart If we don't have the heart point we will get the message "lack of information"
- The **second major** point is the hip it's an approximate point we will call it

“roundHip” we will take the middle of the segment between the “left hip” and the “right hip”

We calculate the width of the hip using the coordinates of left hip and right hip by the formula :

$$\sqrt{[(x_2 - x_1)^2 + (y_2 - y_1)^2]}$$

If we cannot get both points and only one, we will consider roundHip as the only one we have (left hip or right hip) and the width we will take it approximately as the half of the distance between the roundHip and the heart otherwise we will get the message "lack of information"

If we cannot get both points and only one, we will consider roundHip as the only one we have (left hip or right hip) and the width we will take it approximately as the half of the distance between the roundHip and the heart otherwise we will get the message "lack of information"

-The **third major** point is the knee it's an approximate point we will call it “roundKnee” we will take the middle of the segment between the “left knee” and the “right knee”

If we cannot get both points, we will consider it as the only one we have (left knee or right knee) otherwise we will get the message "lack of information".

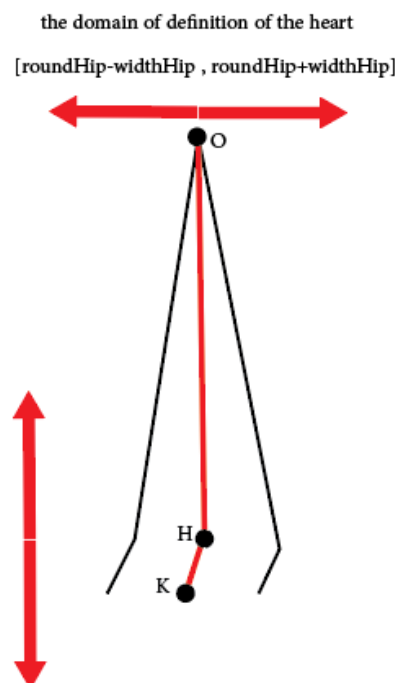
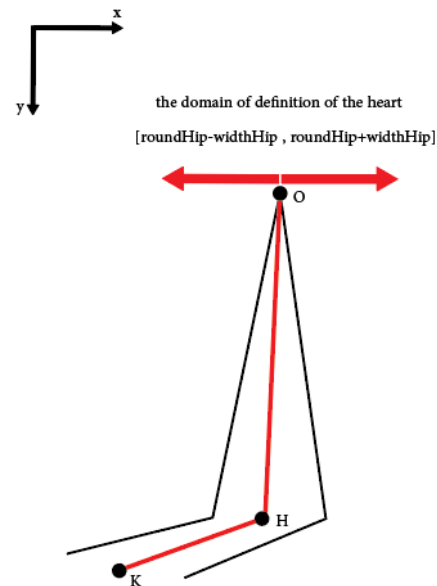
So now after getting the tree points : heart, roundHip and roundKnee and the width of the hip. The first thing we verify is the projection of OH in the axes X and Y

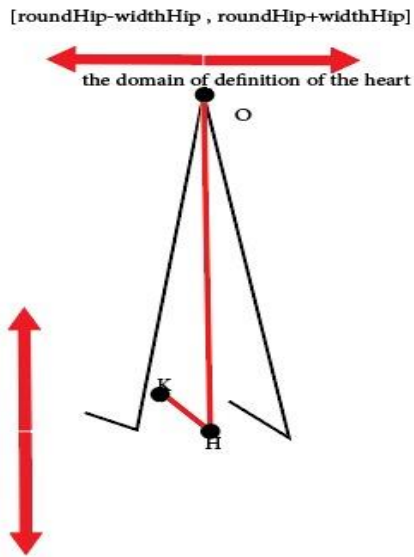
If we have : $OH_{dy} > OH_{dx}$ We can say that the backbone is straight then we will go to verify if he is standing or sitting .

We can say that the person is sitting if we have one of the conditions :

- First we check if $HK_{dx} > HK_{dy}$ if it's true then we can say that the person is sitting
- Otherwise we will check if HK_{dy} exists in the interval $[roundHip - widthHip, roundHip + widthHip]$

Else the person will be standing



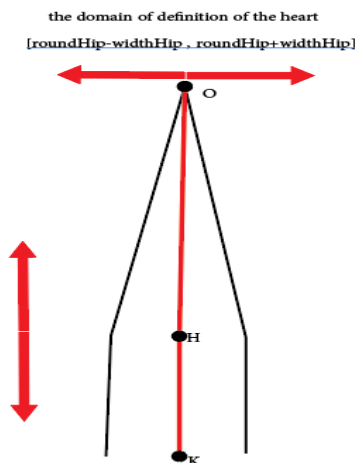


And finally, we have the alarm siren:

Which contains a red lamp, a relay which sends the signal 0 or 1, and the buzzer:



This alarm siren will function when the patient moves in disarray like in the picture below:



To determinate if the person is falling we convert the capture to gray and subtract the background. We keep just the foreground which is the body and we try to find the maximum of body's contours, then determinate the body by drawing a rectangle.

We compare for 5 times if the width is greater than the height. If this condition is TRUE we send the position 'falling' to the database.