

# **The structure of the FABY Documentation**

## **Introduction**

how we think in this project

how we select the project

how we did the story board

how we deal with issues

## **material selection**

the material I choose

why I choose it

## **software selection & machine using**

### **cad design**

design the body on solid works

prepare the body (STL)file on cura

print the body

### **circuit design**

wiring diagram using fritzing

design PCB on eagle (hat for rasperrypi 3 model B)

### **computer vision coding**

definition of computer vision

OpenCV library and it's uses

Installing important libraries

Face recognition explanation

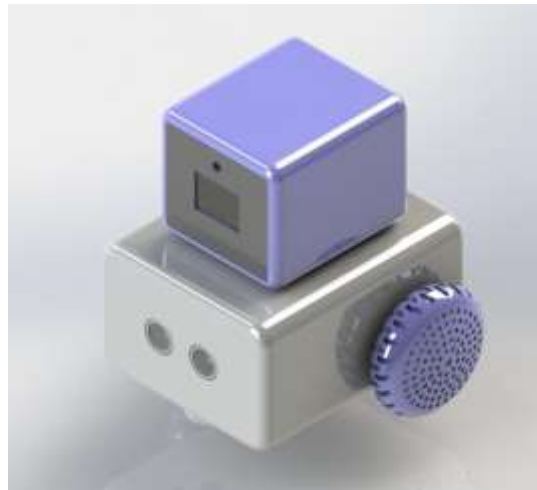
Hand recognition explanation

The whole code

### **google assistant code**

### **mobile app**

## **Introduction**



**As usual, before I go in details in this project, I will discuss with you the process before making this project, the process I love to call on it, selection process**

**Selection process contains**

how we think in this project

how we select the project

how we did the story board

how we deal with issues

technical selection

**during the diploma we learnt how we study the project as it's known as (project management)**

**it's organized procedures that let you study project very carefully and accurately, and I will try to give you thoughts about what happened behind sense**

**how we think in this project**

when we decided to make a project (at first, we didn't decide yet which project we will choose) we intend to make it like ROMBA (it's cleaning robot)



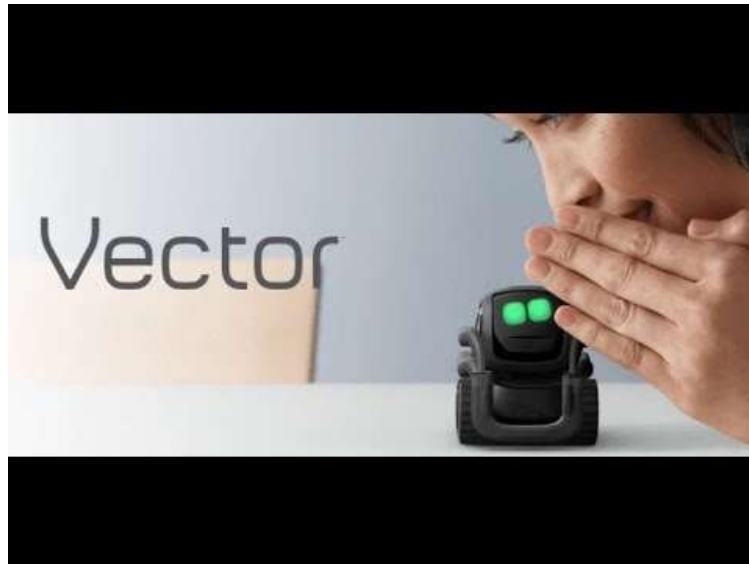
We want to make automatic smart robot.

But after we bought ROMBA and made reverse Engineering on it (that examine every part in ROMBA robot and determine the function of every part)

We watch ad for robot called Vector



What make us admire in this project that it's interactive Robot, small, and very cute (I will leave link video of this robot)



**So we were confused which one we will choose to make it and then we got a wonderful idea.**

### **Selection of project**

As I said earlier, we had a smart idea, the team and I decided to make a combination with both of them

We decided to make our FABY ROBOT

FABY ROBOT is not like both of ROMBA and VECTOR, it's had some features of ROMBA and some others of VECTOR



Se were excited to start build story board and study of project and guess what, that's exactly what we did

## Story board

Simply what we want from this robot to do is

To be interactive

Connected with mobile application

Connected with WIFI

Connected with google assistant



So, we starting to warming up and make some prototypes to see how it works and absolutely we faced issues and problems

And as I make in all of my documentations, I will tell you our failure before success, to avoid every thing wrong we did

## Material selection

It's the main and most important process of them all, because if you didn't select your component carefully and accurately the whole project will fall down

The main and important thing the microprocessor ( Raspberry pi 3 model B )



We need resberri pi camera to make face recognition and camera tracking “computer vision”



2 Metal gear motors ratio 1 :1000 because it has a huge power that can hold the whole body and move fast



OLED screen to display the emotion and messages of FABY to make it interactive will with hymen



Lipo battery 1300 mA, it has large capacity that can supply the motors and servo and OLED screen



Servo motors to make a mechanism for head of FABY to make it track the person and look up and down for more interactive



Ultrasonic sensor to detect any obstacles in front of it and avoid it



Infrared sensor: to detect if there is ground or not so that it doesn't fall from high ground





**USB microphone: to recognize speeches**



**Speakers: to make the robot speaks**



**Caster wheel: to support the body**



## Software selection & machine using

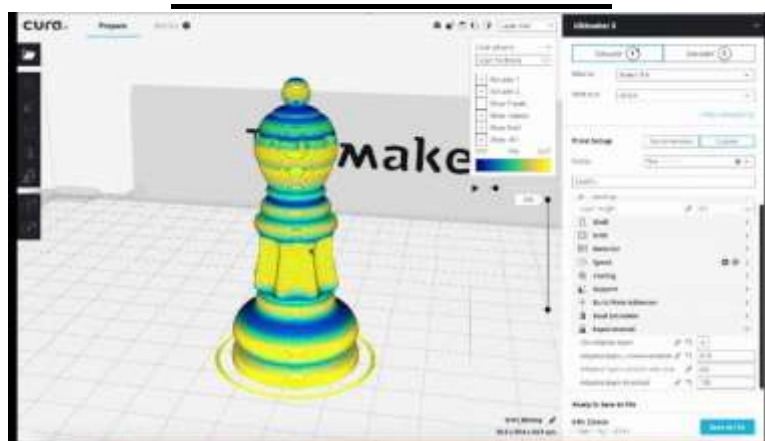
Fusion 360 or solid works any of them will be satisfying for making the body



Eagle : for making PCB of the FABY Robot

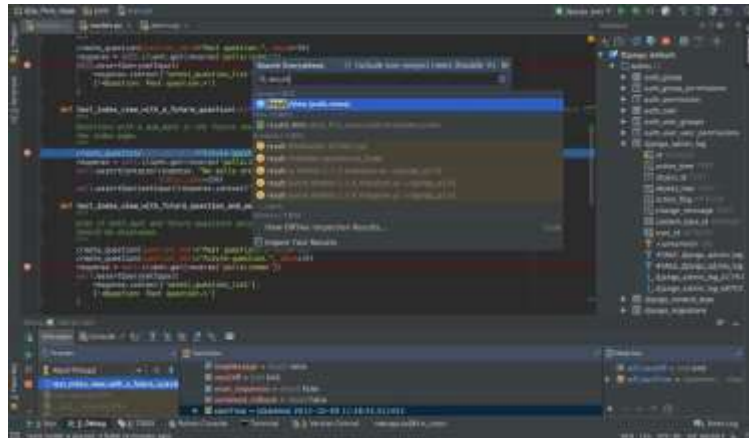


Cura ultimaker 3D printer (to make infill and slice)



For programming we will use pycharm to use python





**3D printer machine**



**PCB milling machine**



## Fritzing software



## Cad design

In this part we will know how to design the robot on cad software

How we think in the body?

Before you think at any thing you should know this, you must take the measurements of your components to know how the body dimension will be figure (1),(2),(3)



As smaller it's, as better it was

As we finished the measurements and preparing our dimensions now, we can imagine the body, so we had to body to be smaller and artistic so we found a one looks exactly as we describe

Figure (4)



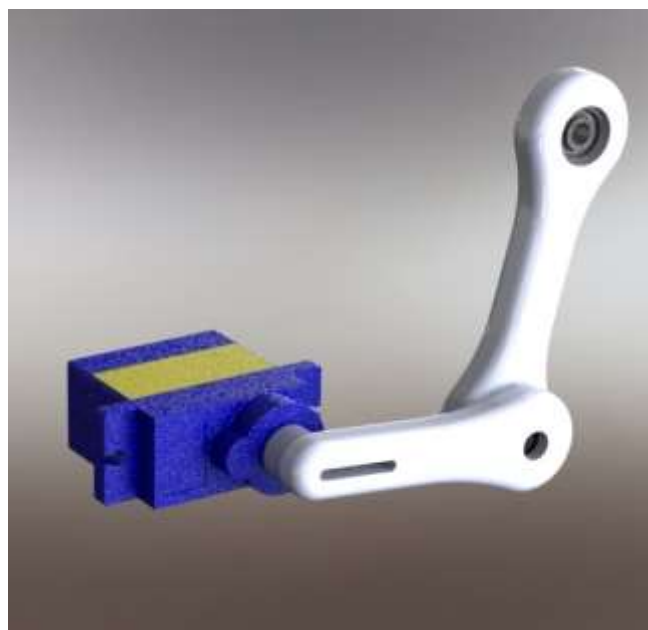
So, we will make our body look close like that

We had the measurements and a vision of the body so, let,s build our FABY Robot

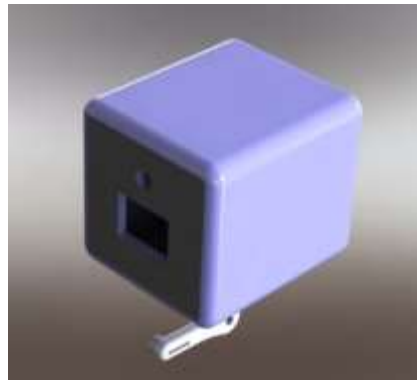
Let's begin with the head, the head must include the camera and OLED screen so will design the head to contain all of these components in it figure (5)



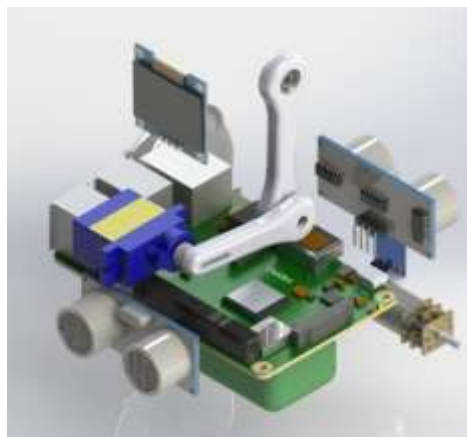
now we need to think in a mechanism that allow the head to move up and down, luckily for you we will give you this mechanism figure (5)



So let's see our mechanism on the head figure (6)



Let's preparing our components and put it in the body figure (7)



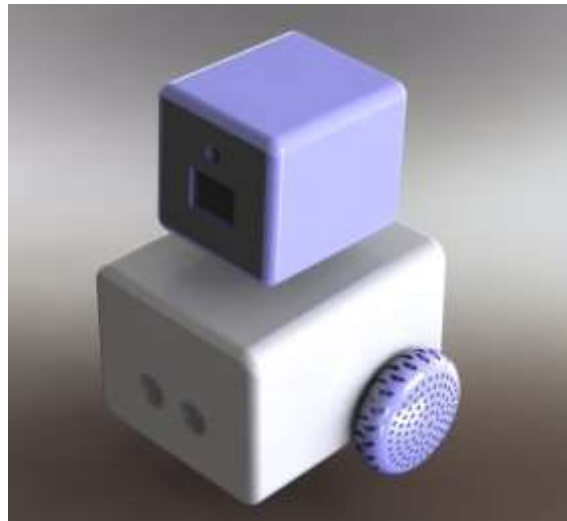
Let's see our body now figure (8)





Design the wheel and put it to the final shape figure (9)

Don't worry I will leave all files down so you can download it



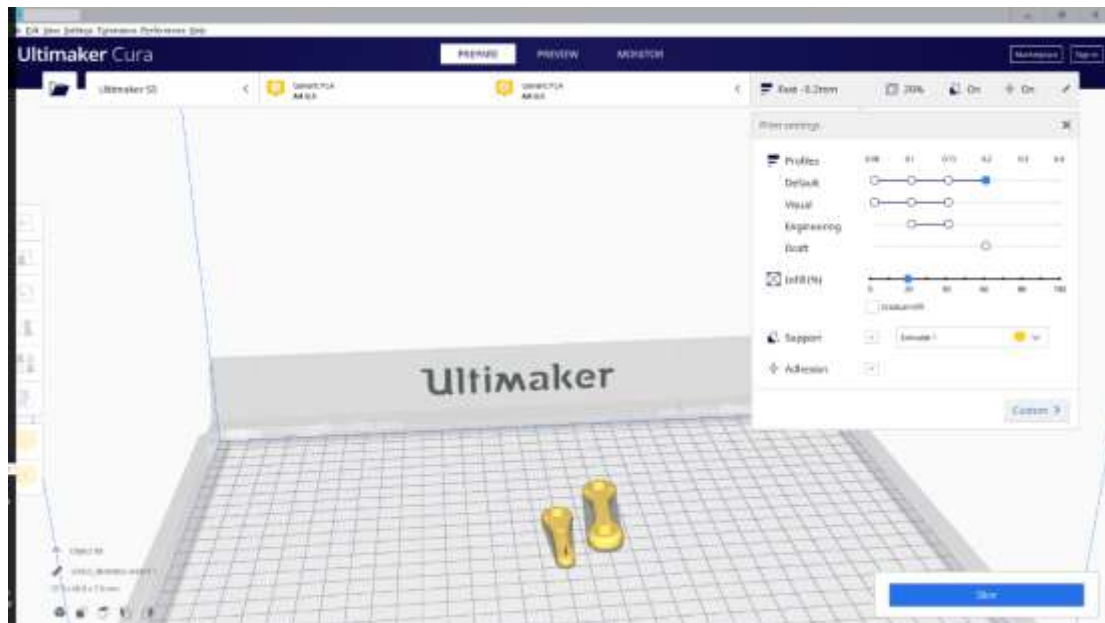
After finish designing the body it's time to fabricate it

To fabricate the body, you need the main thing that you will print with the filament I prefer PLA filament figure (10)

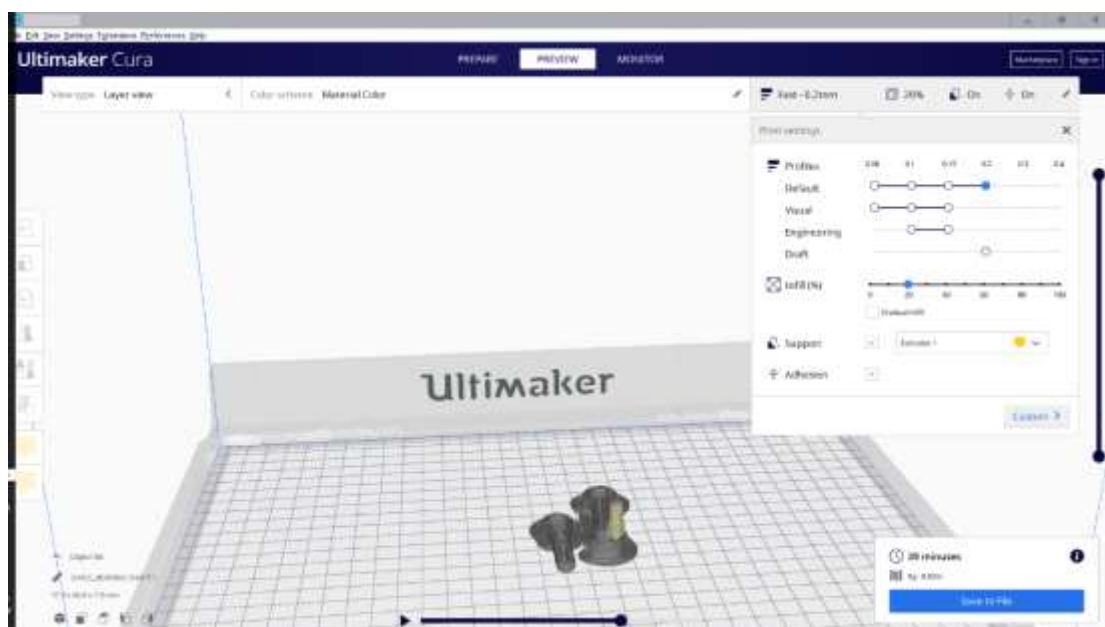


So, after you save the parts as STL files, open Cura and import these files, I will give you two examples

The first one is the mechanism of the head figure (11)



after you import it edit the infill percent and the draft and press slice It will calculate the specific time it will finish printing this part figure (12)



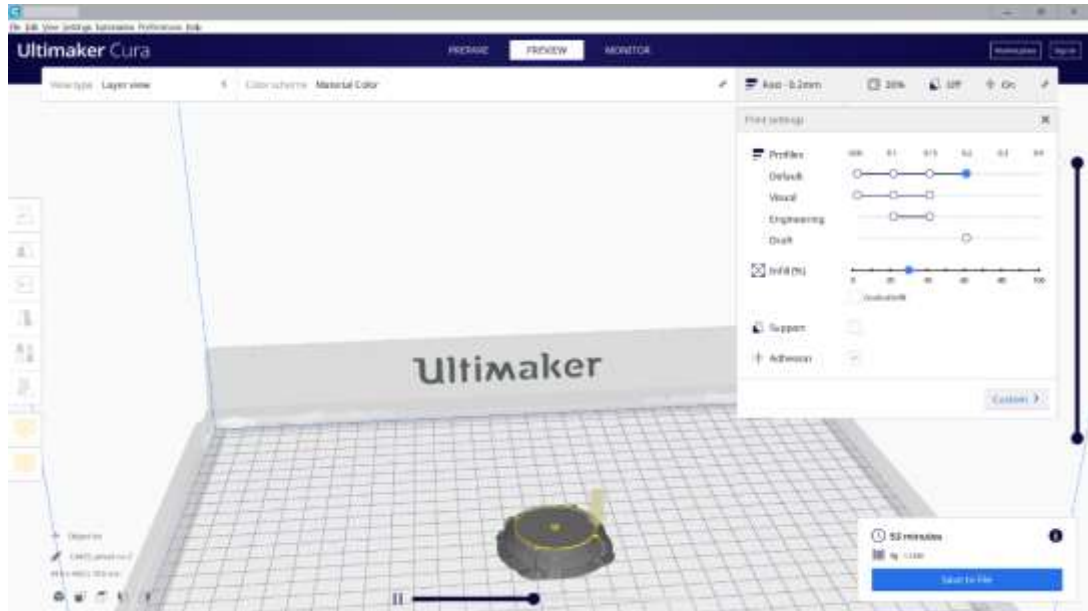
And then press save to file and take this file to flash drive and put it on flash drive and put it in the 3D printer to begin to print Figure (13)



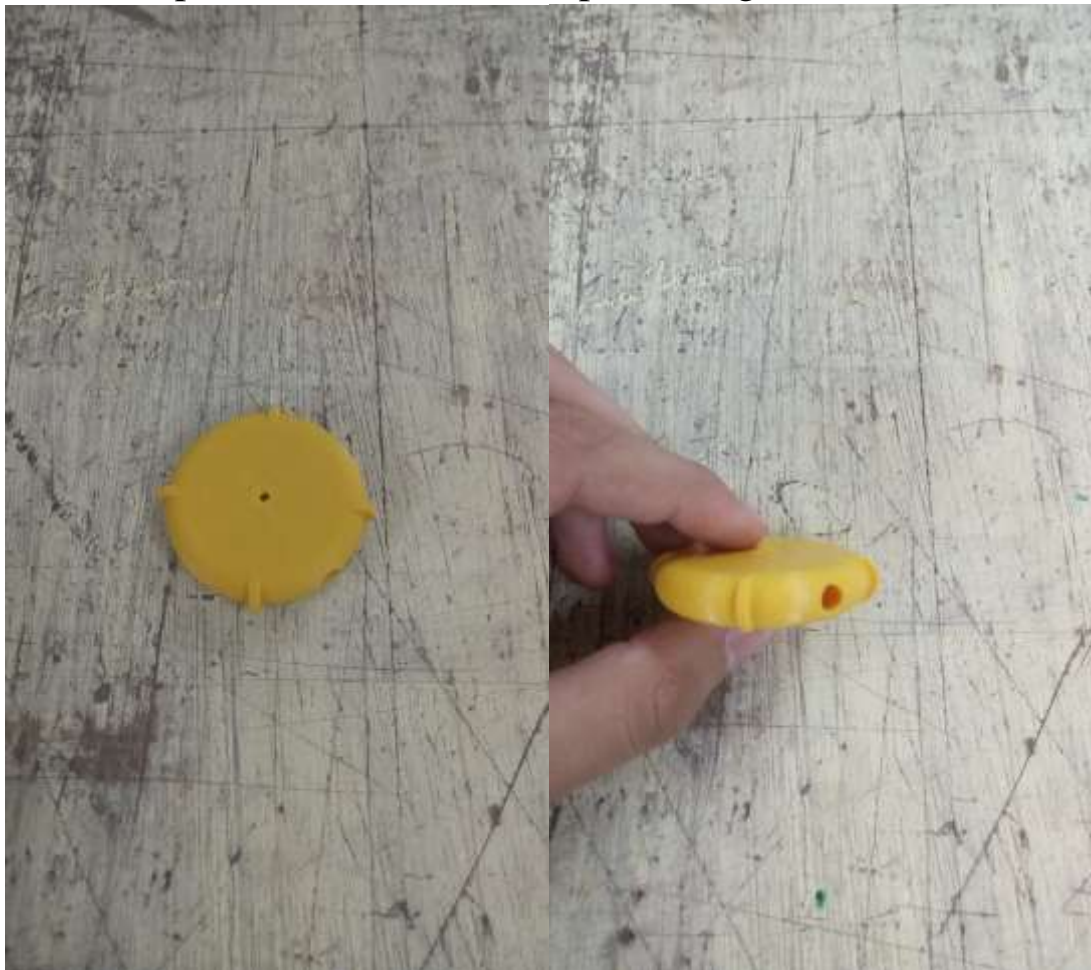
Let's move on to the second part that we will print it which is the body of the wheel without the tire, again repeat the same steps, import the file in cura figure (14)



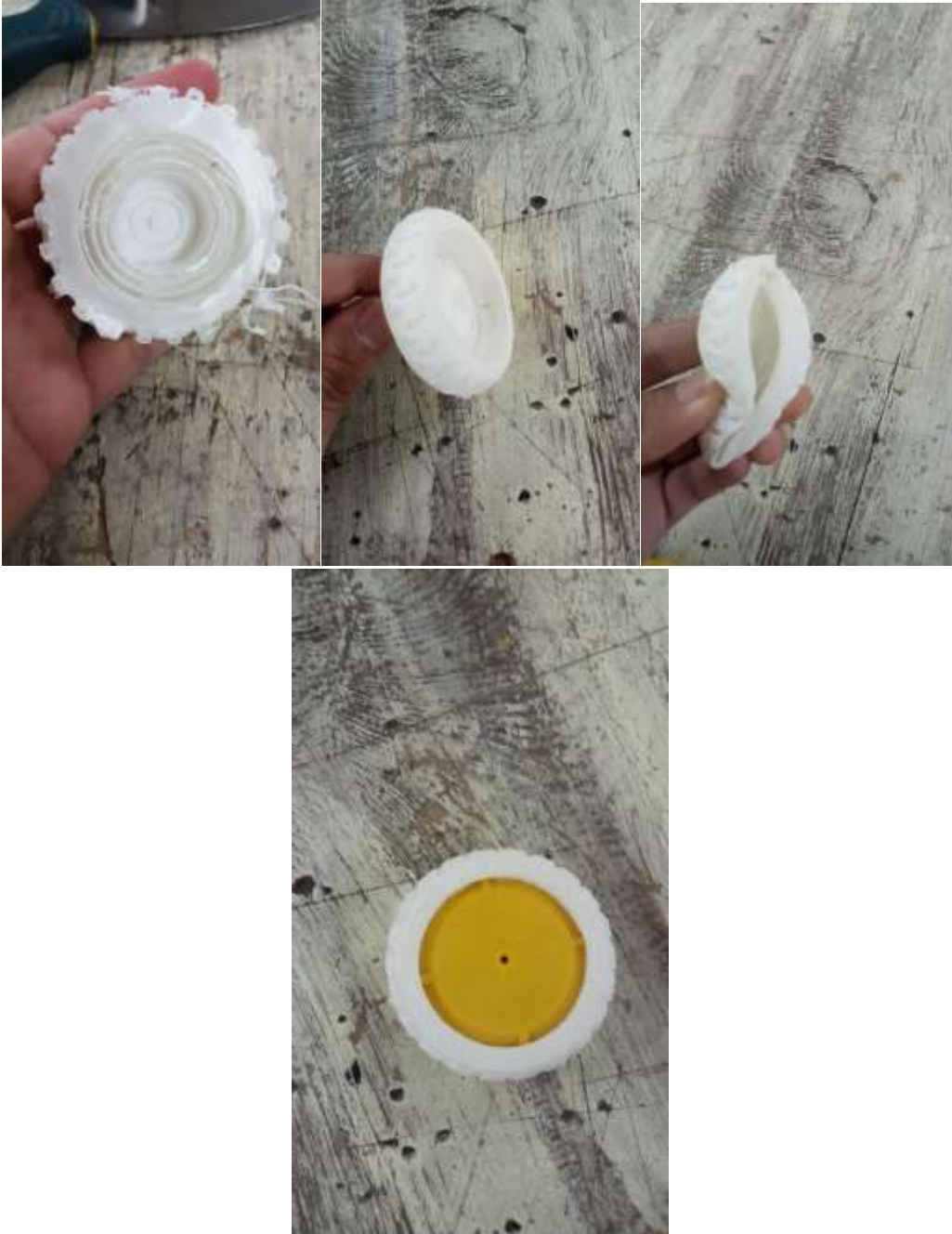
Edit the filament and supports and draft as you see and press slice, you will see the average time that the part will be 3D printed in figure (15)



Then press save the file, and put it on flash drive and upload it to the 3D printer, and left it to be printed figure (16),(17)



Then add to the wheel the rubber part which we made it by ourselves figure (18), (19), (20), (21)



For more information about cad design and the methods visit [Mohamed Gamal](#) website, the original content

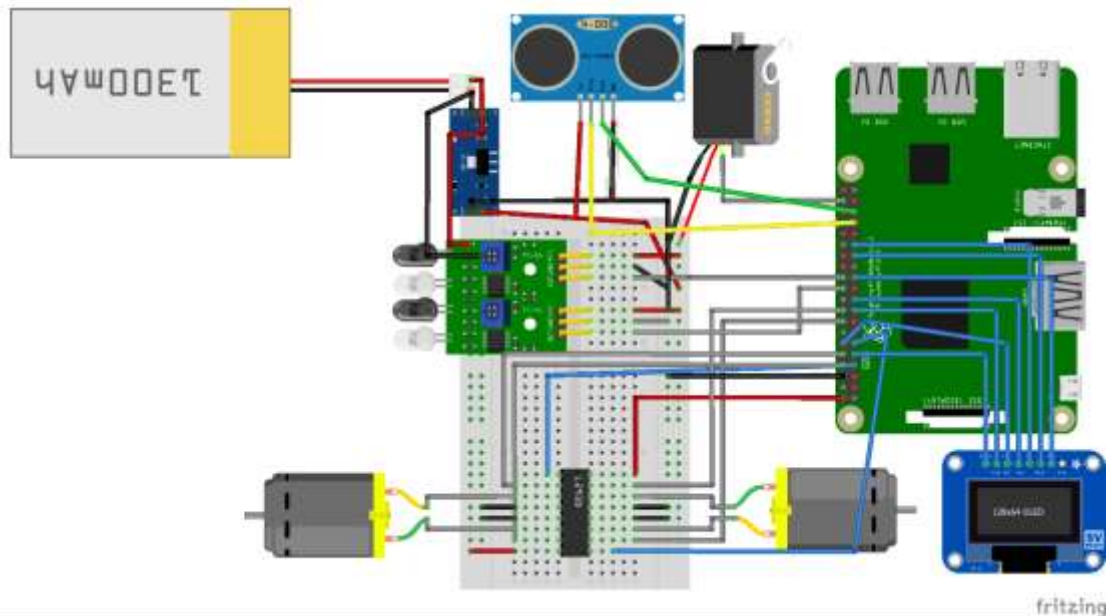












after we made the wiring diagram it's the time to move to the eagle cad, we will have some fun

Now we will design a hat for resberri pi 3 model B

1-Select your components.

2-arrange them in the most suitable way that the pins are clos to each other as possible to avoid overlap of wires and routes.

3-export your schematic to board and get your components arranged.

\*Since We're using rasperry pi board as a controller , we need to make sure that the board will not overlap the pi and fit well over it .

that's why we used the component "RASPBERRYPI-40-PIN-GPIO\_PTH\_REFERENCE".

Components:

- 7.4 lipo battery
- step down dc-dc converter
- 2 IR sensors
- L293D motordrive IC.
- 2 metal gear DC motor.

-Raspberry pi 3 model B.

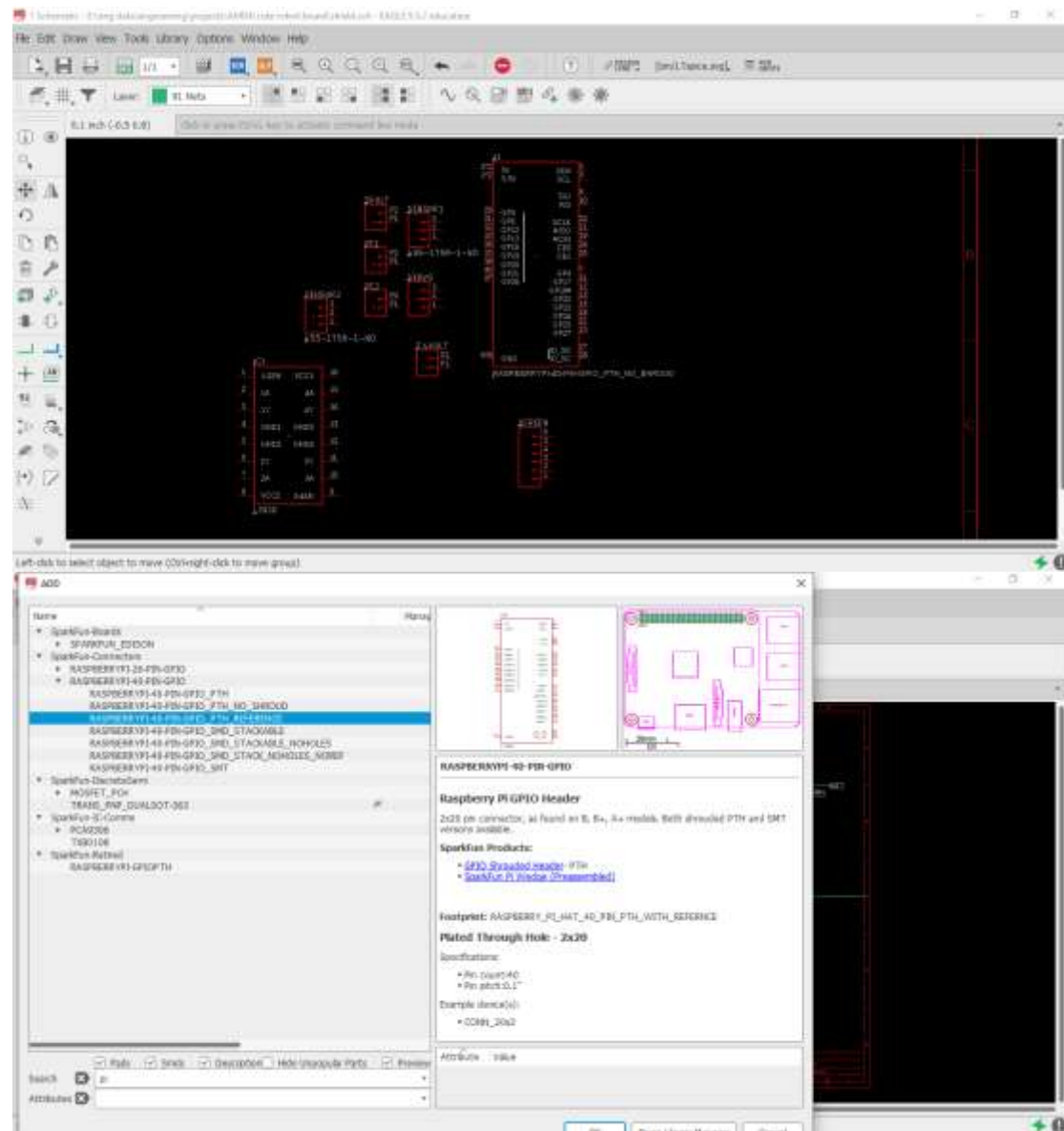
-1 Ultra sonic sensor

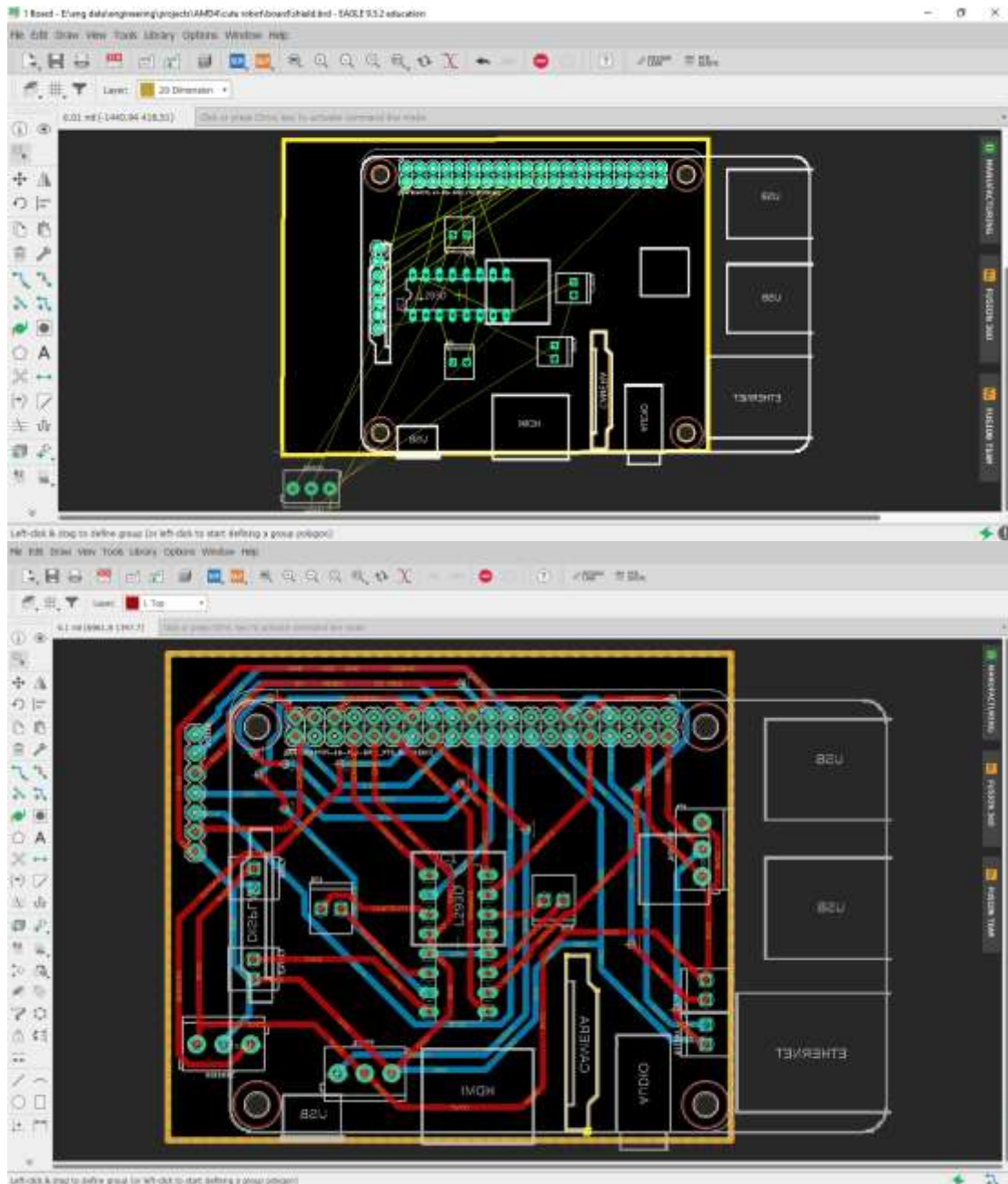
Shield board components :

-5 of 2 wire 5mm terminals.

-3 of 3 wire 5mm terminals.

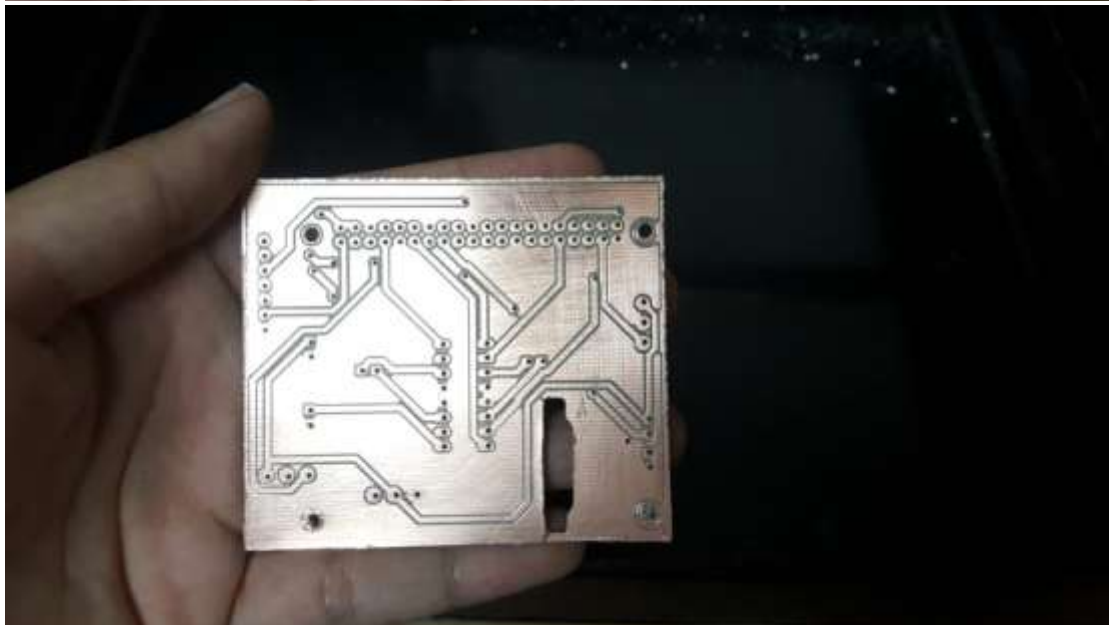
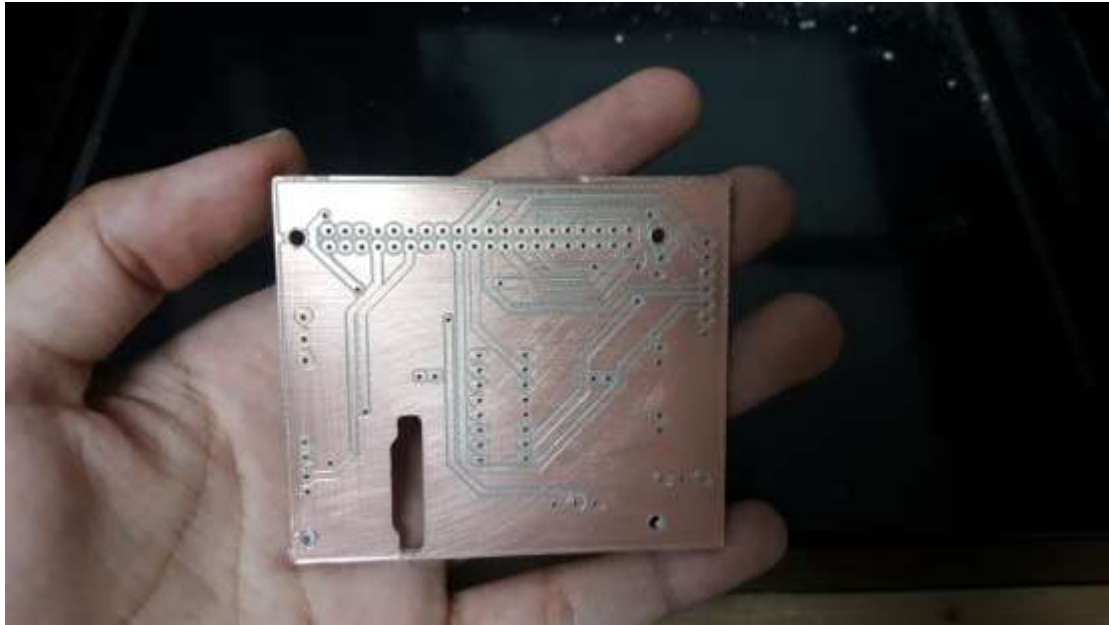
-6 pin headers for the screen





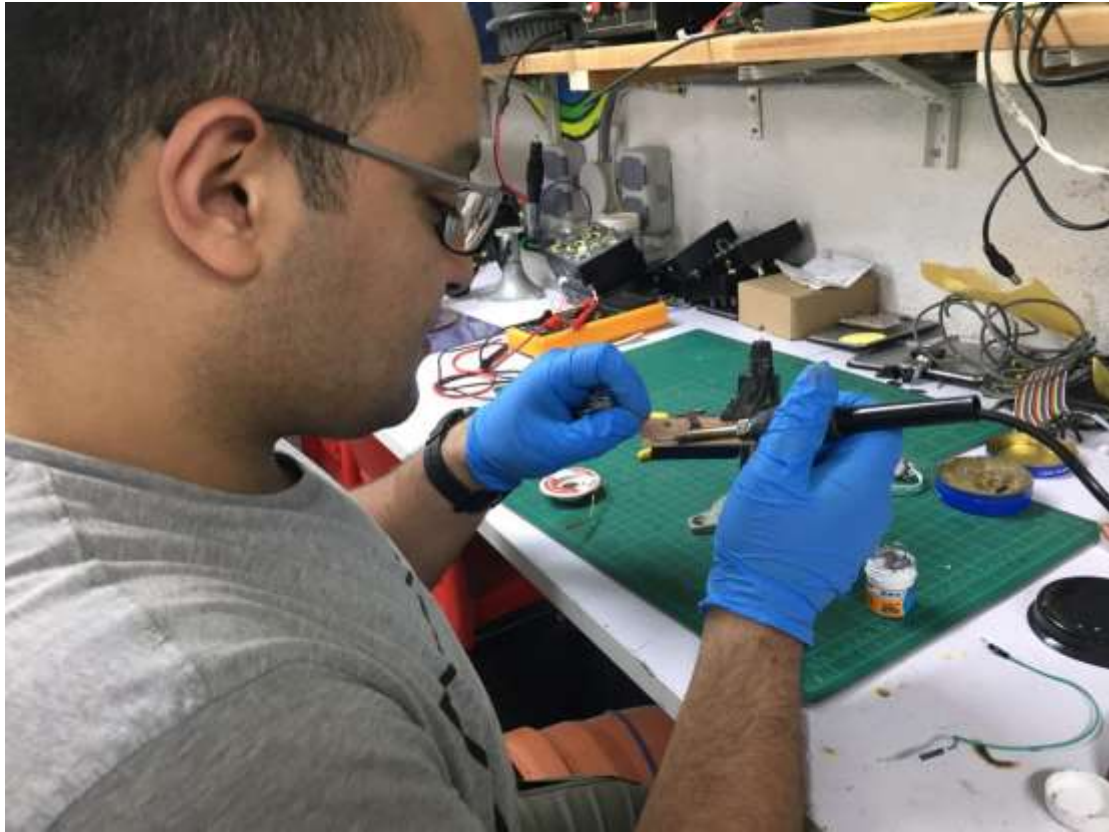
After that we will send it to lab to be fabricated and note that (the mine track width is 0.5mm and mine diameter hole 0.8 mm )

After we send it to FAB LAB it's fabricated and the result was fantastic



and again, don't worry I will leave the board and schematic down in files, just click on it and download it

After we finish the PCB fabricate it's now time for soldring the component on the PCB



Note: Thanks for Alaa saber for providing us with all resources, we did the PCB design with his effort so thank you again

If you want to visit the original content of circuit design please click on [Alaa Saber](#)

## Computer vision



this part is not easy, it's needed a lot of concentration and codes to understand how face recognition and hand recognition works so be aware in everything I will say and I will try to give you the conclusion not whole idea

- note: special thanks to our lovely Amani that help us all by herself in computer vision she explain all the procedures with marvelous words, and explain all the errors she had faced, I recommend to read her article in computer vision you will find interesting videos with simple explanation just click on [Amani Ayman Muhammad](#)

**Computer Vision:** Computer Vision is the broad parent name for any computations involving visual content – that means images, videos, icons, and anything else with pixels involved. But within this parent idea, there are a few specific tasks that are core building blocks:

- In **object classification**, you train a model on a dataset of specific objects, and the model classifies new objects as belonging to one or more of your training categories.
- For **object identification**, your model will recognize a specific instance of an object – for example, parsing two faces in an image and tagging one as Tom Cruise and one as Katie Holmes

to make face & hand recognition we should install important library which called **OpenCV**.

### **What is OpenCV library?**

is an open-source for computer vision and digital image processing and machine learning software library

I highly recommend to you if you don't know what is face detection is and didn't deal with it you should visit the websites, I will leave now

### **[Face detection with OpenCV and deep learning](#)**

### **[OpenCV Face Recognition](#)**

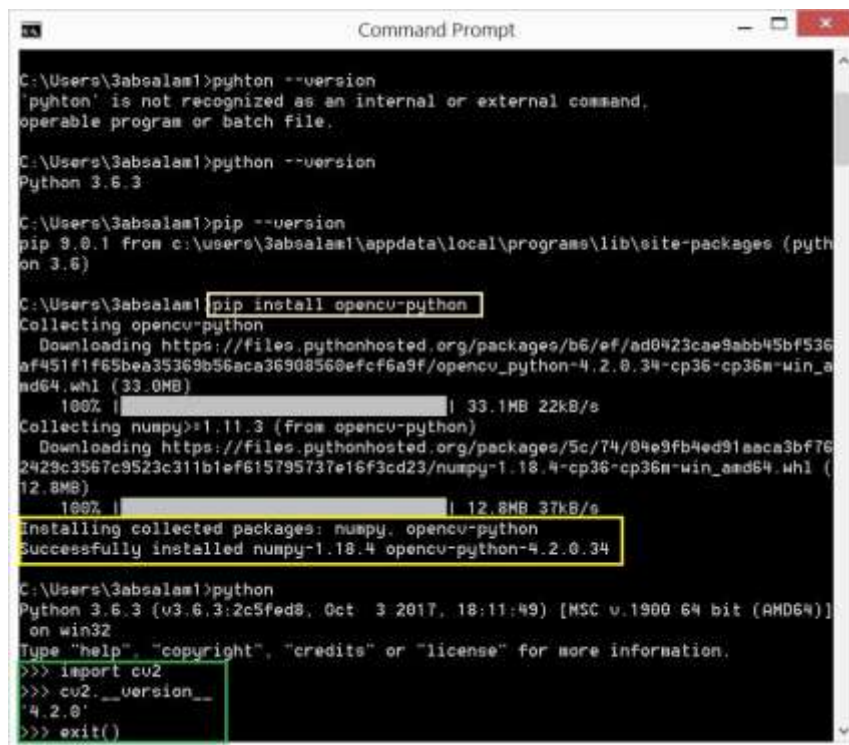
These two websites will explain every small detail about using deep learning with OpenCV library to make face detection, I shall not go in detail in this documentation because it will be very long and hard instead of that I will give you short summary of it.

### **How to install OpenCV library?**

I worked on the command prompt in windows and just typed this one line



## pip install opencv-python



```
C:\Users\3absalam1>pyhton --version
'pyhton' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\3absalam1>python --version
Python 3.6.3

C:\Users\3absalam1>pip --version
pip 9.0.1 from c:\users\3absalam1\appdata\local\programs\lib\site-packages (python 3.6)

C:\Users\3absalam1>pip install opencv-python
Collecting opencv-python
  Downloading https://files.pythonhosted.org/packages/b6/ef/ad8423cae9abb45bf536af451f1f65bea35369b56aca36908560efcf6a9f/opencv_python-4.2.0.34-cp36-cp36m-win_amd64.whl (33.0MB)
    100% |#####| 33.1MB 22kB/s
Collecting numpy>=1.11.3 (from opencv-python)
  Downloading https://files.pythonhosted.org/packages/5c/74/04e9fb4ed91aeca3bf762429c3567c9523c311b1ef615795737e16f3cd23/numpy-1.18.4-cp36-cp36m-win_amd64.whl (12.8MB)
    100% |#####| 12.8MB 37kB/s
Installing collected packages: numpy, opencv-python
Successfully installed numpy-1.18.4 opencv-python-4.2.0.34

C:\Users\3absalam1>python
Python 3.6.3 (v3.6.3:2c5fed8, Oct 3 2017, 18:11:49) [MSC v.1900 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> import cv2
>>> cv2.__version__
'4.2.0'
>>> exit()
```

I assume that you have **OpenCV** installed on your system.

## Dlib and the face\_recognition packages.

***Note:** For the following installs, ensure you are in a Python virtual environment if you're using one. I highly recommend virtual environments for isolating your projects — it is a Python best practice. If you've followed my OpenCV install guides (and installed virtualenv + virtualenvwrapper )*

*then you can use the workon command prior to installing dlib and face\_recognition*

## Installing dlib without GPU support

If you do not have a GPU you can install dlib using pip by [following this guide](#):

```
Face recognition with OpenCV, Python, and deep learning
```

```
$ workon # optional
```

```
$ pip install dlib
```

Or you can compile from source:

```
Face recognition with OpenCV, Python, and deep learning
```

```
$ workon <your env name here> # optional
```

```
$ git clone https://github.com/davisking/dlib.git
```

```
$ cd dlib
```

```
$ mkdir build
```

```
$ cd build
```

```
$ cmake .. -DUSE_AVX_INSTRUCTIONS=1
```

```
$ cmake --build .
```

```
$ cd ..
```

```
$ python setup.py install --yes USE_AVX_INSTRUCTIONS
```

## Installing dlib *with* GPU support (optional)

If you *do* have a CUDA compatible GPU you can install dlib with GPU support, making facial recognition faster and more efficient.

For this, I recommend installing dlib from source as you'll have more control over the build:

```
Face recognition with OpenCV, Python, and deep learning
```

```
$ workon <your env name here> # optional
```

```
$ git clone https://github.com/davisking/dlib.git
```

```
$ cd dlib
```

```
$ mkdir build
```

```
$ cd build
```

```
$ cmake .. -DDLIB_USE_CUDA=1 -DUSE_AVX_INSTRUCTIONS=1
```

```
$ cmake --build .
```

```
$ cd ..
```

```
$ python setup.py install --yes USE_AVX_INSTRUCTIONS --yes DLIB_USE_CUDA
```

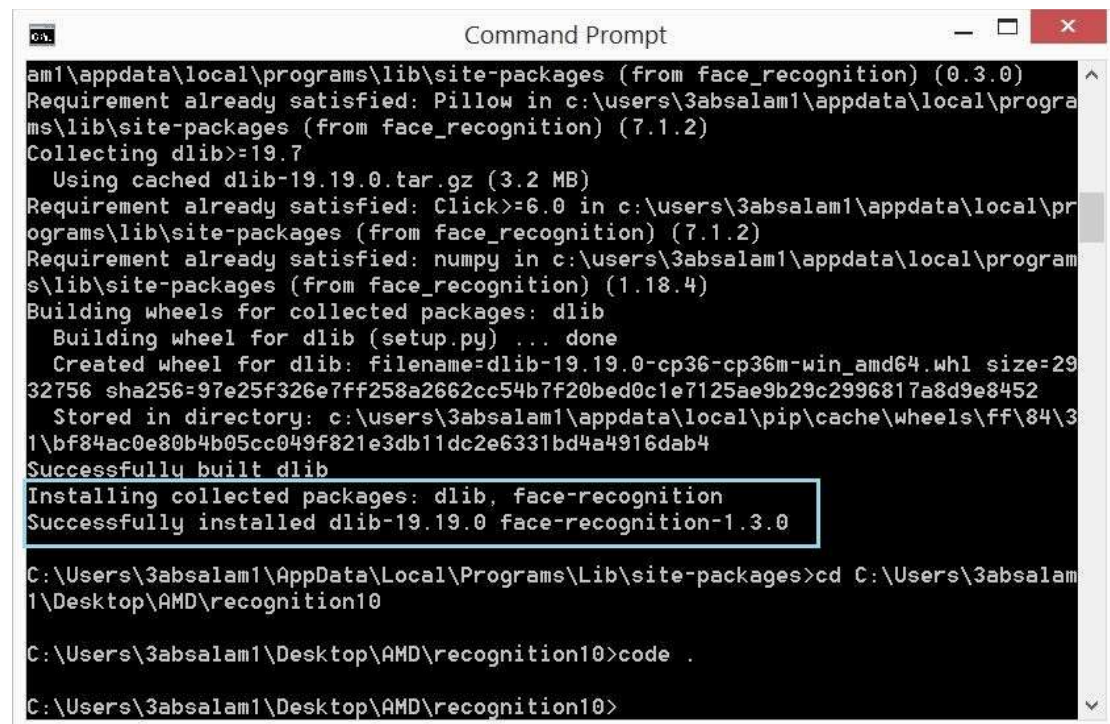
## Install the `face_recognition` package

The `face_recognition` module is installable via a simple pip command:

Face recognition with OpenCV, Python, and deep learning

```
$ workon <your env name here> # optional
```

```
$ pip install face_recognition
```



```
Command Prompt
am1\appdata\local\programs\lib\site-packages (from face_recognition) (0.3.0)
Requirement already satisfied: Pillow in c:\users\3absalam1\appdata\local\progra
ms\lib\site-packages (from face_recognition) (7.1.2)
Collecting dlib>=19.7
  Using cached dlib-19.19.0.tar.gz (3.2 MB)
Requirement already satisfied: Click>=6.0 in c:\users\3absalam1\appdata\local\pr
ograms\lib\site-packages (from face_recognition) (7.1.2)
Requirement already satisfied: numpy in c:\users\3absalam1\appdata\local\program
s\lib\site-packages (from face_recognition) (1.18.4)
Building wheels for collected packages: dlib
  Building wheel for dlib (setup.py) ... done
  Created wheel for dlib: filename=dlib-19.19.0-cp36-cp36m-win_amd64.whl size=29
32756 sha256=97e25f326e7ff258a2662cc54b7f20bed0c1e7125ae9b29c2996817a8d9e8452
  Stored in directory: c:\users\3absalam1\appdata\local\pip\cache\wheels\ff\84\3
1\bf84ac0e80b4b05cc049f821e3db11dc2e6331bd4a4916dab4
Successfully built dlib
Installing collected packages: dlib, face-recognition
Successfully installed dlib-19.19.0 face-recognition-1.3.0

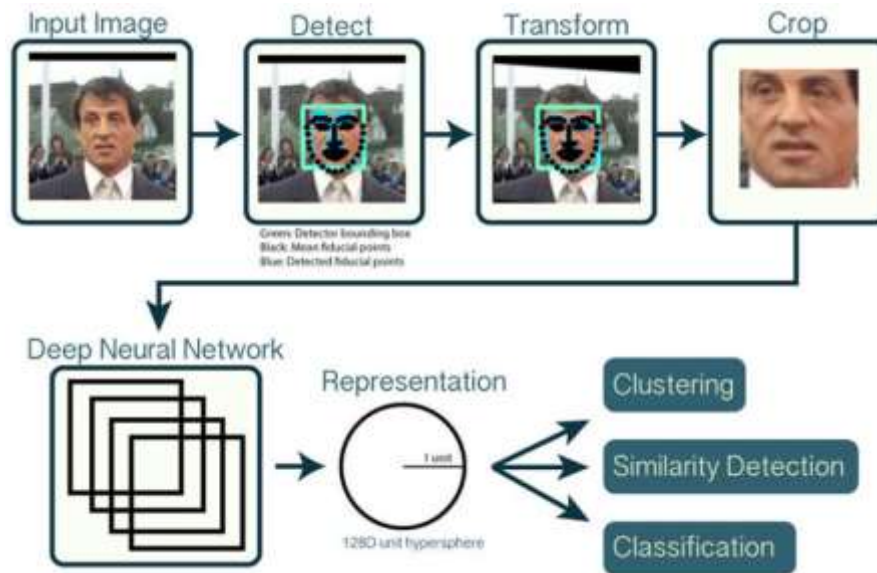
C:\Users\3absalam1\AppData\Local\Programs\Lib\site-packages>cd C:\Users\3absalam
1\Desktop\AMD\recognition10

C:\Users\3absalam1\Desktop\AMD\recognition10>code .

C:\Users\3absalam1\Desktop\AMD\recognition10>
```

## How face recognition works?

In order to build our OpenCV face recognition pipeline, we'll be applying deep learning in Reviewing the entire FaceNet implementation is outside the scope of this tutorial, but the gist of the pipeline can be seen in Figure 1

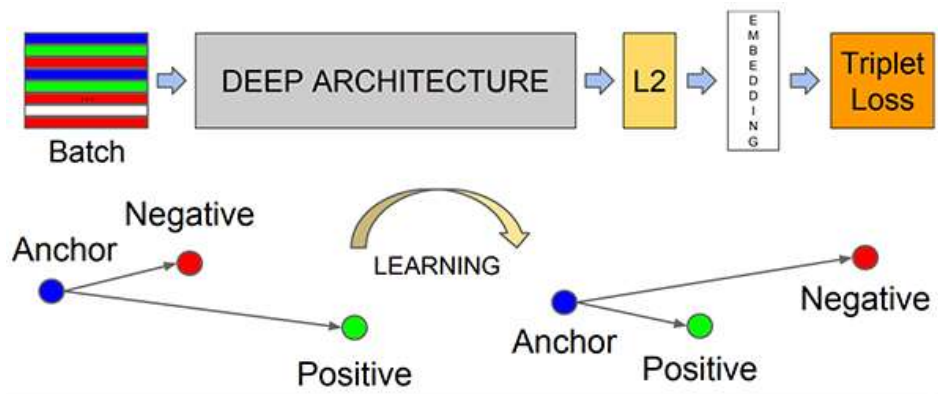


Face alignment, as the name suggests, is the process of

- (1) identifying the geometric structure of the faces and
- (2) attempting to obtain a canonical alignment of the face based on translation, rotation, and scale

While optional, face alignment has been demonstrated to increase face recognition accuracy in some pipelines

After we've (optionally) applied face alignment and cropping, we pass the input face through our deep neural network:



The FaceNet deep learning model computes a 128-d embedding that quantifies the face itself.

That's the way face recognition works.

Here is a sample of face recognition code

```

import cv2
import numpy as np
import sys
import os

# Define the path to the deep learning model
MODEL_PATH = "models"
DATA_PATH = "data"
EMBEDDING_DIM = 128

# Load the model
model = cv2.FaceRecognizerEngine()

# Load the training data
faces = []
names = []
for i in range(1, 100000):
    img = cv2.imread(os.path.join(DATA_PATH, "img%d.jpg" % i))
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    img = cv2.resize(img, (112, 112))
    img = img.astype(np.float32) / 255.0
    faces.append(img)
    names.append("img%d.jpg" % i)

# Train the model
model.train(faces, names)

# Test the model
img = cv2.imread("img1.jpg")
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
img = cv2.resize(img, (112, 112))
img = img.astype(np.float32) / 255.0

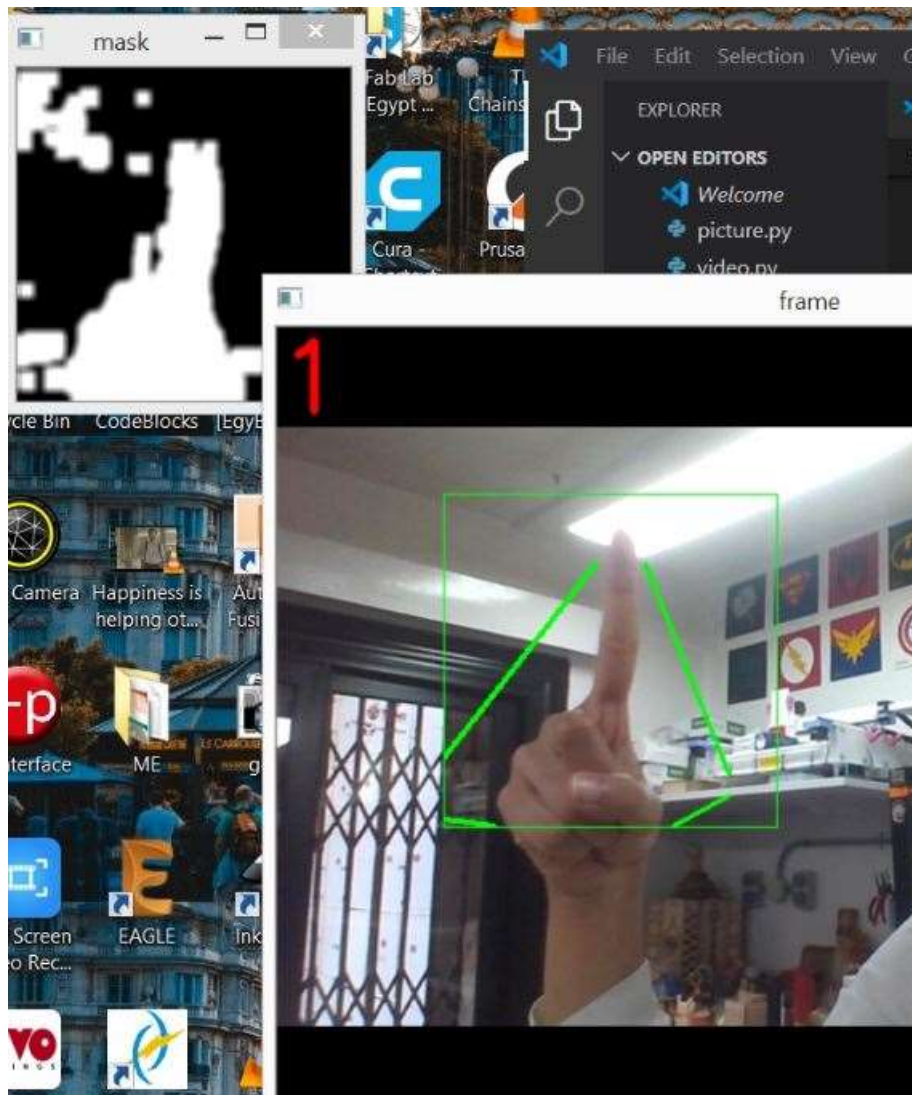
results = model.recognize(img)
print("Results: %s" % results)

```

And here is video for Amani trying the code and it works yeah

[Amani video](#)

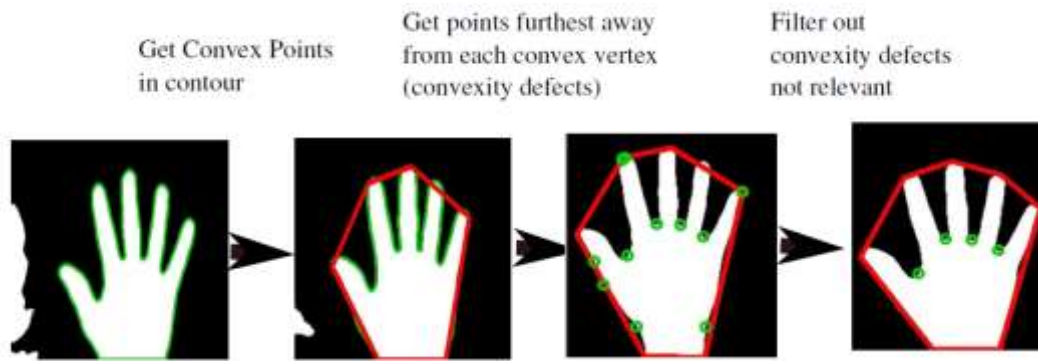
## Hand track and counting



In order to make hand track we need to install library called **math** it's easy to install I will leave links and codes down

### How hand track works?

In order to detect fingertips, we are going to use the Convex Hull technique. In mathematics, Convex Hull is the smallest convex set that contains a set of points. And a convex set is a set of points such that, if we trace a straight line from any pair of points in the set, that line must be also be inside the region. The result is then a nice, smooth region, much easier to be analysed than our contour, that contains many imperfections.



**To detect the Fingers and count them.**

Find the roi(region of interest)

Hand Segmentation : Convert the video frame from BGR to HSV(or Gray)

Perform a Gaussian blur

Perform a Threshold

Find the Biggest Contour(this will be our hand)

Perform a convexHull and mark the ROI(region of interest)

Count the no. of countours

Display it

**Samples**

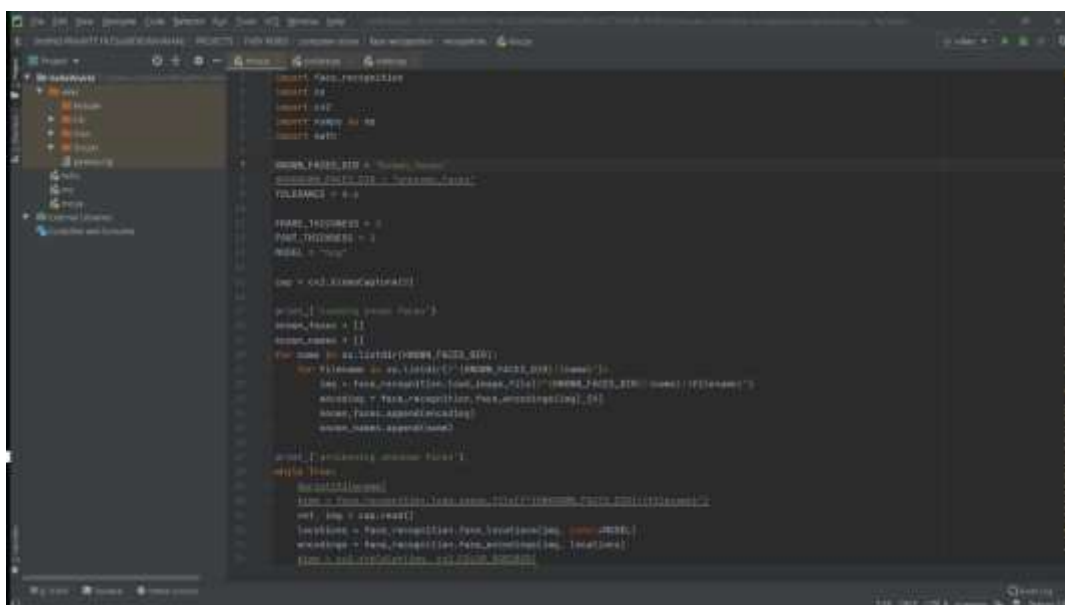


very well. [You can find the original code here](#). Then, found another source that illustrates how the counting of hand fingers' code works. [You can find it here](#).

Now for the final step: mix the two codes together (face recognition and hand tracking)

Just bring the face recognition code and merge it with hand tracking code in one big code, don't forget to edit the variables

Sample of code



```
import cv2
import numpy as np
import sys
import os

# Load the face recognizer
face_recognizer = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

# Load the hand detector
hand_detector = cv2.CascadeClassifier('haarcascade_hand_26stage.xml')

# Parameters
MIN_FACE_SIZE = 100
MIN_HAND_SIZE = 100
TOLERANCE = 0.5

# Face recognizer variables
FACE_WIDTH = 100
FACE_HEIGHT = 100
FACE_COLOR = (0, 0, 255)

# Hand detector variables
HAND_WIDTH = 100
HAND_HEIGHT = 100
HAND_COLOR = (0, 255, 0)

def main():
    # Open the camera
    cap = cv2.VideoCapture(0)

    # Read the first frame
    ret, frame = cap.read()

    # Detect faces
    faces = face_recognizer.detectMultiScale(
        frame,
        scaleFactor=1.1,
        minNeighbors=5,
        minSize=(MIN_FACE_SIZE, MIN_FACE_SIZE)
    )

    # Detect hands
    hands = hand_detector.detectMultiScale(
        frame,
        scaleFactor=1.1,
        minNeighbors=5,
        minSize=(MIN_HAND_SIZE, MIN_HAND_SIZE)
    )

    # Draw bounding boxes
    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x+w, y+h), FACE_COLOR, 2)

    for (x, y, w, h) in hands:
        cv2.rectangle(frame, (x, y), (x+w, y+h), HAND_COLOR, 2)

    # Show the frame
    cv2.imshow('Face and Hand Tracking', frame)

    # Wait for a key press
    cv2.waitKey(1)

    # Close the camera
    cap.release()

    # Destroy the window
    cv2.destroyAllWindows()

if __name__ == '__main__':
    main()
```

And here is a video for Amani after she finished the whole code [Here is a video of the mixed code running](#)



## Google assistant software

In this part we will learn about google assistant and how to provide our FABY Robot with google assistant

Before anything it's prefers to work on the rasperry pi software (Linux raspbian distribution)



Make sure you plugged microphone in the rasperry pi

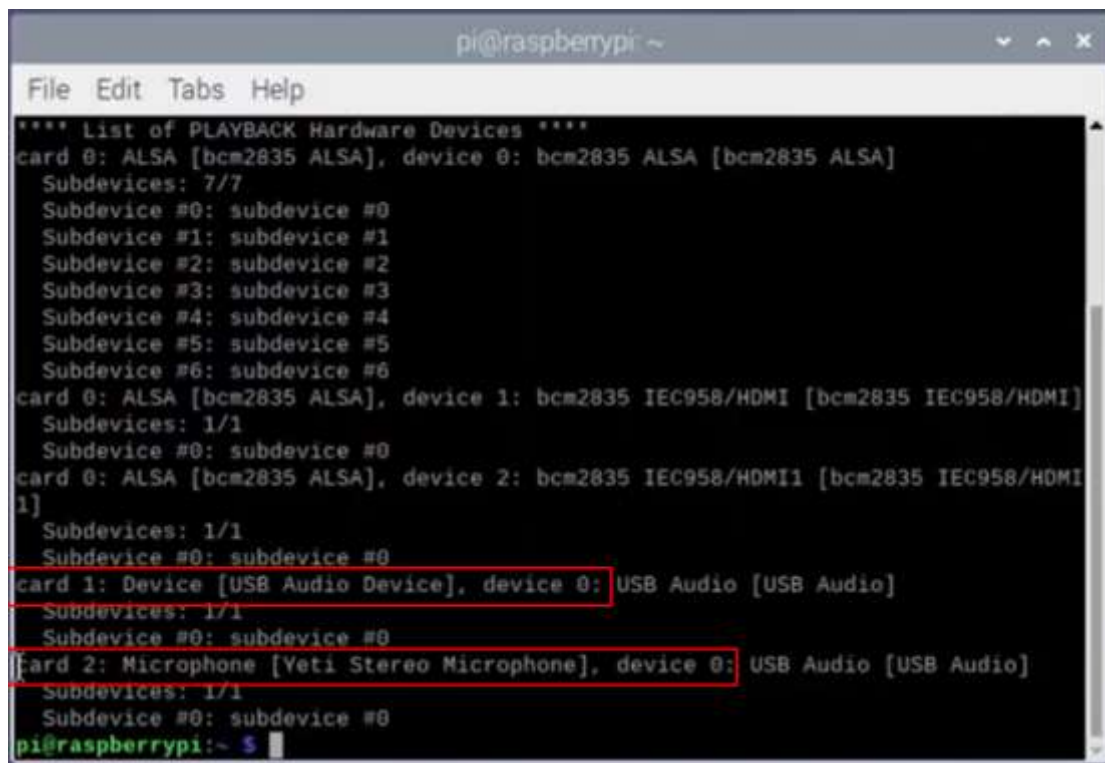
By the end of 2019 and the beginning of 2020 google posted a tutorial about how to implement google assistant on your rasperry pi

<https://developers.google.com/assistant/sdk/guides/library/python>

n, I followed all the tutorial steps, so I managed to make it works In this documentation I will talk about these steps, so you can follow me here or go to Google tutorial.

Configure and Test the Audio Test the microphone and speaker to make sure they are working well. To do so, Frist, open the terminal then write “aplay -l” to find the card number and device number allocated for the microphone and speaker. Don’t

forget to write down these two numbers for microphone and speaker as we will need them later.



```
pi@raspberrypi ~  
File Edit Tabs Help  
**** List of PLAYBACK Hardware Devices ****  
card 0: ALSA [bcm2835 ALSA], device 0: bcm2835 ALSA [bcm2835 ALSA]  
  Subdevices: 7/7  
    Subdevice #0: subdevice #0  
    Subdevice #1: subdevice #1  
    Subdevice #2: subdevice #2  
    Subdevice #3: subdevice #3  
    Subdevice #4: subdevice #4  
    Subdevice #5: subdevice #5  
    Subdevice #6: subdevice #6  
card 0: ALSA [bcm2835 ALSA], device 1: bcm2835 IEC958/HDMI [bcm2835 IEC958/HDMI]  
  Subdevices: 1/1  
    Subdevice #0: subdevice #0  
card 0: ALSA [bcm2835 ALSA], device 2: bcm2835 IEC958/HDMI1 [bcm2835 IEC958/HDMI  
1]  
  Subdevices: 1/1  
    Subdevice #0: subdevice #0  
card 1: Device [USB Audio Device], device 0: USB Audio [USB Audio]  
  Subdevices: 1/1  
    Subdevice #0: subdevice #0  
card 2: Microphone [Yeti Stereo Microphone], device 0: USB Audio [USB Audio]  
  Subdevices: 1/1  
    Subdevice #0: subdevice #0  
pi@raspberrypi:~ $
```

Second, Create the (.asoundrc) file by writing this command  
“sudo nano /home/pi/.asoundrc”

on the terminal

Third, replace all the text inside it by the following text “  
pcm.!default {

type asym capture.pcm "mic"

playback.pcm "speaker"

}

pcm.mic

{

```
type plug slave
```

```
{ pcm "hw:,"
```

```
}
```

```
}
```

```
pcm.speaker {
```

```
type plug slave {
```

```
pcm "hw:,"
```

```
}
```

```
}
```

Fourth, replace, with the numbers that you wrote down earlier.



```
pi@raspberrypi: ~
File Edit Tabs Help
GNU nano 3.2 /home/pi/.asoundrc Modified
pcm.!default {
  type asym
  capture.pcm "mic"
  playback.pcm "speaker"
}
pcm.mic {
  type plug
  slave {
    pcm "hw:2,0"
  }
}
pcm.speaker {
  type plug
  slave {
    pcm "hw:1,0"
  }
}
Get Help Write Out Where Is Cut Text Justify Cur Pos
Exit Read File Replace Uncut Text To Spell Go To Line
```

Fifth, save and exit the file by clicking (Ctrl + x) then (y + Enter)

Sixth, type “alsamixer” on the terminal and raise the sound level of the speakers

Seventh, test the speakers by typing “speaker-test -t wav” on the terminal, by clicking enter you should

hear (left, Front) from the speakers then click (Ctrl + c) to stop .it

Eighth, test the microphone by recording sound by typing -- “arecord --format=S16\_LE --duration=5

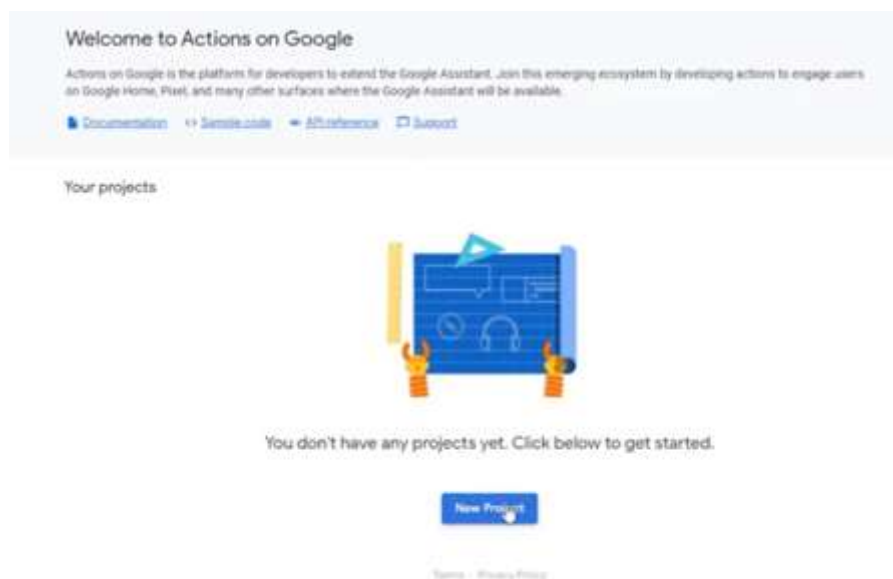
rate=16000 --file-type=raw out.raw” on the terminal

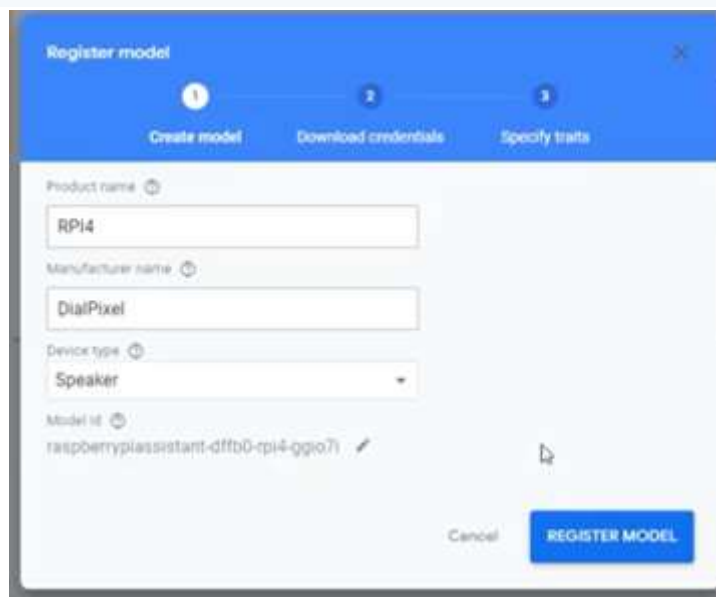
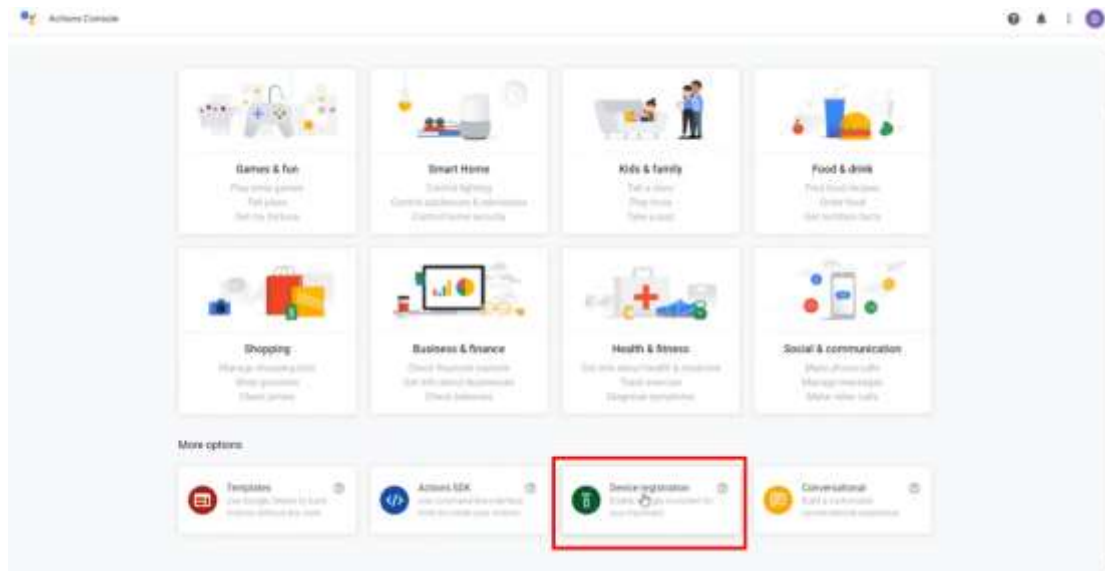
Ninth, play the sound that you recorded to make sure the -- microphone works well by typing “aplay

format=S16\_LE --rate=16000 out.raw” on the terminal

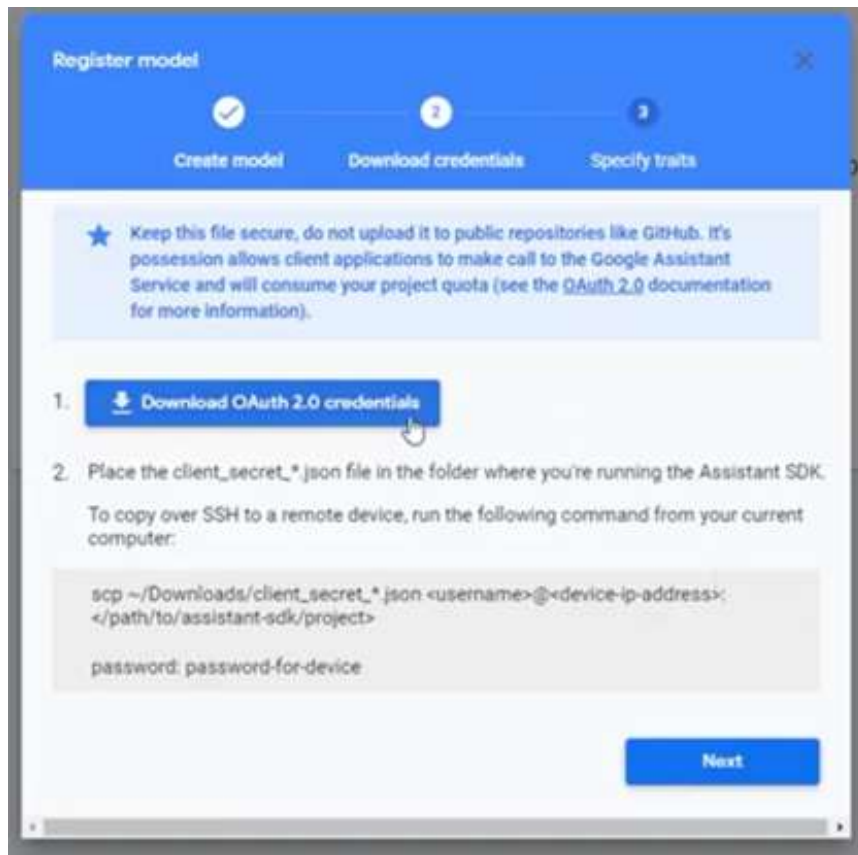
## **Configure a Developer Project and Account Settings and Register the Device Model**

First, open your internet browser then go to “console.actions.google.com” then select (new project) and enter the name of your project



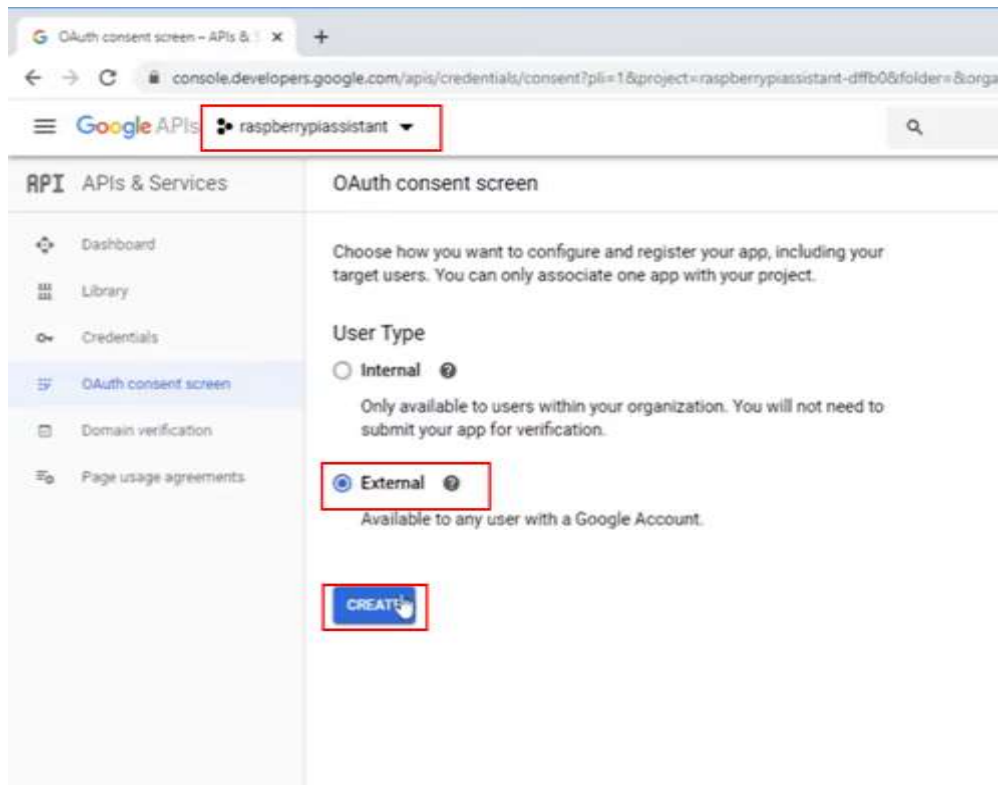


Third, after clicking on (REGISTER MODEL), click on (Download OAuth 2.0 credentials), save the json file as we will need it later, then skip the specify traits options



Fourth, go to “[console.developers.google.com/apis](https://console.developers.google.com/apis)” then click on (Enable API and services), search for google assistant API and enable it

Fifth, go to the authorization console screen as shown in the figure below, then select (External) then (CREAT), then confirm your Email and save the settings



Sixth, go to “myaccount.google.com/activitycontrols” and make sure all the following are turned on

Web & app activity

Location history

Device information

Voice and audio activity

#### **Step4: Install the SDK and Sample Code**

First, open the terminal then type “sudo apt-get update” then “sudo apt-get upgrade” then install

python3 virtual environment by typing “sudo apt-get install python3-dev python3-venv” then write

python3 -m venv env” to enable the virtual environment“

Second, update the pip by typing “env/bin/python -m pip install --upgrade pip setuptools wheel”, then

activate the python virtual environment using the source  
”command by typing “source env/bin/activate

Third, type in the terminal the following “sudo apt-get install  
portaudio19-dev libffi-dev libssl-dev”, then

install google assistant SDK by typing “python -m pip install --  
” upgrade google-assistant-sdk[samples]

Fourth, copy the json file that you downloaded and put it in  
.”/home/pi” directory then copy its path

Fifth, back to the terminal and make sure the virtual  
environment is activated then type “python -m pip

”install --upgrade google-auth-oauthlib[tool]

Sixth, type in the terminal “google-oauthlib-tool --scope  
https://www.googleapis.com/auth/assistant-sdk-prototype --save  
--headless --client-secrets /home/pi/<credential-file-  
name>.json”, Don’t forget to

.replace <credential-file-name> with the path of the json file

Seventh, now you must see a URL on the terminal, open it and  
copy the code then paste it in the

terminal

Eighth, to activate google assistant type “googlesamples-  
-- <assistant-pushtotalk --project-id <project-id

device-model-id <model-id>” on the terminal and Don’t forget  
to replace both <project-id> and <modelid> with their values  
from the action dash board then go to the project settings page



## **Mobile Application**

For Mobile App design, I used MIT App inventor, because it's easy to learn, and easy manual. Actually, MIT App inventor is a great starter program for app building.

There are several challenges should be included in the App:

1. Make an attractive design.
2. Converting the voice into words for commands.
3. Send a command to take a photo or record a video.
4. The robot motion control.
5. Connect to google assistant API.

So that, I watched several videos on YouTube, as:

- [Android Tutorial: Convert Speech To Text | Speech Recognition](#)
- [Converting speech to text: How to create a simple dictation app using speech recognition](#)
- [10 iPhone Apps for Converting Voice to Text](#)
- [speech to text : convert speech to text and vice versa](#)
- [Speech to Text : Voice Notes & Voice Typing App](#)
- [The 7 Best Android Dictation Apps for Easy Speech-to-Text:](#)
- [Speech notes](#)
- [Voice Notes](#)
- [Speech Texter](#)
- [Voice Notebook](#)
- [Google Assistant](#)
- [Speech to Text](#)
- [OneNote](#)
- [Android Speech to Text Tutorial](#)
- [Speech To Text - Android Studio - Java](#)
- [Android Speech to Text Tutorial](#)
- [Speech Recognition using Python](#)
- [Speech to Text Conversion using MIT App Inventor 2](#)
- [speech recognition app using mit app inventor](#)
- [speech recognition android app using MIT app inventor](#)

- [MIT App Inventor 2 - Voice Recognition App](#)
- [App Inventor: Speech Recognizer and Text to Speech](#)
- [Convert voice commands to text - App-inventor Android Arduino](#)
- [converting text into orders using google assistant](#)
- [Text To Speech MIT APP Inventor](#)

The design not worked well in the first, so produced several versions as the following:

- Version 1:
  - The first Screen:
    - The buttons for voice [Button3] was added, with the speech organizer to can talk as Non-visible components, beside a text box for writing the voice words.
    - Added ON [Button4]and OFF [Button5] button to control the raspberry pi GPIO pin, by adding android thingsboard1 and Android thingsGPIO1 as Non-visible components.
    - Forward [Button2] was added to go to screen 2.
  - The Second screen:
    - The buttons for voice [Speak Button] was added, with the speech organizer to can talk as Non-visible components, beside a text box for writing the voice words.
    - Added Backward [Button1] to back to Screen 1, and forward [Button2] to go to Screen 3.
  - The third Screen:
    - The buttons for voice [Button1] was added, with the speech organizer to can talk as Non-visible components, beside a text box for writing the voice words.
    - Bluetooth
    - Added a bluetooth picture [ListPicker1 button]to open the mobile bluetooth to contact with the robot which controled by adding

bluetoothClient1 and Clock as Non-visible components.

- Added Backward button [Button2] to back to Screen 1.
  - The Cons of the design:
    - It is designed for converting speaking into words, but it is so boring with poor design.
  - [The video testing](#)
- 
- Version 2:
    - here I downloaded the Vector app to follow the ideal design. But the home icons not fully organized and sending the commands still not working.
- 
- Version 3:
    - It is cover all the previous challenges.
      - The First Screen { **Welcome screen** }:
        - Involves one button to go to home screen [Screen 2]
      - The Second Screen { **Home Screen** }:
        - Includes six button to open camera, Stats, Entertainment, Question and Answer mode, Interact, About.
      - The Third Screen { **Camera Screen** }:
        - Contacted with firebase database by adding in Non-visible components, by sending code {1} when press camera picture [Snap button] to open the robot camera and take a capture, and sending code {2} when press the video picture [Video button] to open the robot camera and record a video.

- The firebase also tested by adding Name and Age in the text boxes and press save to save on the database.
- Camera test button added to shows the firebase response, which did by adding Ev3touchsensor1 and camera1 as non-visible components.
- The Fourth Screen {**Entertainment Screen**}:
  - Includes four arrows {left, right, up, and down} to control the robot motion.
- The Fifth Screen {**Question and Answer mode Screen**}:
  - Which will be connected with Google Assistant API
- The Sixth Screen {**Interact Screen**}:
  - Contains the buttons for speaking [Hi\_FABY], with the speech organizer to can talk as Non-visible components, beside a text box for writing the voice words.
- The Seventh Screen {**About**}:
  - That includes the information about the robot and the working team and the fabricated place.
- [The video testing](#)
- [The video Testing the commands sending to firebase.](#)

