Ollscoil Teicneolaíochta an Atlantaigh

Atlantic Technological University

# Smart Mirror

Project Engineering

Year 4

Conor Keane

Bachelor of Engineering (Honours) in Software and Electronic Engineering

Atlantic Technological University

2023/2024

Welcome

# Declaration

This project is presented in partial fulfilment of the requirements for the degree of Bachelor of Engineering (Honours) in Software and Electronic Engineering at Galway-Mayo Institute of Technology.

This project is my own work, except where otherwise accredited. Where the work of others has been used or incorporated during this project, this is acknowledged and referenced.

_______________________________________________

# Acknowledgements

I want to acknowledge all the ATU Lecturers and staff members who have helped me and given me guidance through out this project and over the years. In particular I would like to thank Michelle Lynch for helping and supervising me with this project.

I would also like to thank my class members who I have built a strong and life-long friendship with over the years.

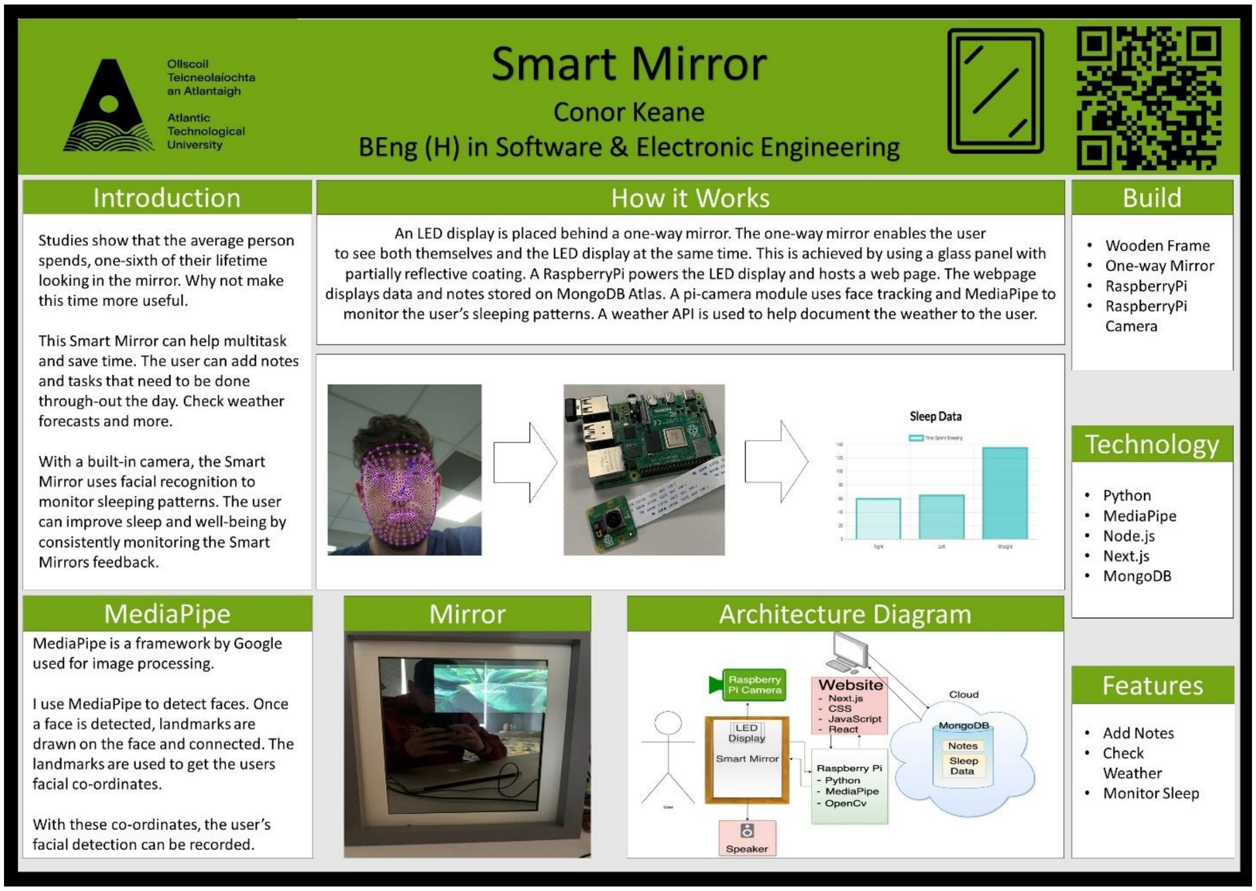
Finally, I would like to thank Greaney Glass Products Limited Galway who helped cut the glass I needed for this project and gave me insight into the physics of one-way glass.

# Table of Contents

# 1 Summary

For a while now I have been looking at smart mirrors online. Smart Mirrors are a popular project online, so much so that you can install open-source software like MagicMirror² [1], which will turn a display into a ready to go smart mirror website. I wanted to make my own smart mirror with my own software and design. To make this mirror stand out when compared to other smart mirrors online, I wanted to add a sleep monitoring feature. I had suffered from sleeping problems and not getting significant amounts of sleep. I wanted to integrate this into my project so I could use it on myself and monitor my sleeping patterns. I have also been interested in Artificial Intelligence (ai) and wanted to integrate this into my project. My smart mirror would sit on a locker in my bedroom, monitor my sleeping patterns and give data based on my sleeping patterns. It would have other helpful features like being able to take notes, play music and display a weather forecast. I would achieve monitoring sleeping patterns using a Raspberry Pi [2] and a camera module. Ai would then detect any sleeping patterns using facial recognition. A web page would be displayed with the help of the Raspberry Pi. Here I can check the weather, listen to music thanks to Spotify [3] and weather [4] API's.

## 2 Poster



Ollscoil Teicneolaíochta an Atlantaigh
Atlantic Technological University

# Smart Mirror

Conor Keane
BEng (H) in Software & Electronic Engineering

### Introduction

Studies show that the average person spends, one-sixth of their lifetime looking in the mirror. Why not make this time more useful.

This Smart Mirror can help multitask and save time. The user can add notes and tasks that need to be done through-out the day. Check weather forecasts and more.

With a built-in camera, the Smart Mirror uses facial recognition to monitor sleeping patterns. The user can improve sleep and well-being by consistently monitoring the Smart Mirrors feedback.

### How it Works

An LED display is placed behind a one-way mirror. The one-way mirror enables the user to see both themselves and the LED display at the same time. This is achieved by using a glass panel with partially reflective coating. A RaspberryPi powers the LED display and hosts a web page. The webpage displays data and notes stored on MongoDB Atlas. A pi-camera module uses face tracking and MediaPipe to monitor the user's sleeping patterns. A weather API is used to help document the weather to the user.

### Build

- Wooden Frame
- One-way Mirror
- RaspberryPi
- RaspberryPi Camera

### Technology

- Python
- MediaPipe
- Node.js
- Next.js
- MongoDB

### MediaPipe

MediaPipe is a framework by Google used for image processing.

I use MediaPipe to detect faces. Once a face is detected, landmarks are drawn on the face and connected. The landmarks are used to get the users facial co-ordinates.

With these co-ordinates, the user's facial detection can be recorded.

### Mirror

### Architecture Diagram

### Features

- Add Notes
- Check Weather
- Monitor Sleep

# 3 Introduction

## 3.1 Project Goals

The goals of my project were the following:

- To differentiate my Smart Mirror from other Smart Mirror projects found online.
- Add features to the Smart Mirror that I would use or have an interest in.
- Implement a form of artificial intelligence into the mirror.
- Complete the project within the time frame given to me.

## 3.2 Motivation

I had seen many tutorials on how to build a smart mirror online [5]. My motivation for this project was to build my own version. I wanted to learn more about the Python language, using a RaspberryPi, and working with some form of artificial intelligence. Finding the time to work on my own version of a smart mirror would have been expensive and time consuming. With the ability to do a final year project, this gave me the motivation to work on a smart mirror with little time limitations and expenses. I also have had sleeping problems in the past. I looked into physical sleeping monitor devices but had read that they don't actually help. One reporter from the New York Times actually said it made him grumpier [37]. I also found a lot of devices only track heart rate and time slept, not positions or if your mouth is open etc. Sleeping positions or if someone is sleeping with their mouth open, can actually tell you a lot about the sleep their getting and how it will affect them [38] [39]. These were some factors I wanted to take into account.

## 3.3 Scope

The project is to be placed in a user's bedroom. High enough that it can oversee the user's bed. This way the mirror can record data on the user's sleeping patterns.

# 4 Background

## 4.1 Raspberry Pi 4 Model B

The Raspberry Pi 4 Model B [6] is the backbone of the Smart Mirror. 4GB of RAM, 64-bit quad-core Cortex-A72 processor, 2 Micro HDMI ports, 2 USB 3.0 ports and 2 USB 2.0 ports. The Pi runs on Debian version 12 (Bookworm). With these features it was the ideal board for a smart mirror. Multiple ports would allow more features to be incorporated. The Pi is also known for its ability to run python scripts. This would be helpful when trying to implement any form of artificial intelligence or machine learning, as Python is one of the best languages for image processing [7].

## 4.2 OpenCV

OpenCV is a real-time library for image processing and performing computer vision tasks [8]. This library is an easy install and would be helpful when working with capturing and image processing any images taken of the user sleeping.

## 4.3 Raspberry Pi Camera Module 3

To monitor the users sleeping patterns a camera module is needed to capture this data. The Raspberry Pi Camera Module 3 [9] is ideal for this. With its easy set up and compatibility with the Raspberry Pi. With a resolution of 11.9 megapixels, Phase Detection Autofocus and 1080p50 video mode, this camera module will be able to detect the user and report back information so that their sleep positions can be monitored.

## 4.4 MediaPipe

MediaPipe is an open-source framework for machine learning created by Google [10]. This framework can detect a user’s face and place landmarks onto their face. These landmarks use an x, y and z axis, each landmark has a co-ordinate. With these co-ordinates a user’s facial positioning can be detected.

## 5 Project Architecture

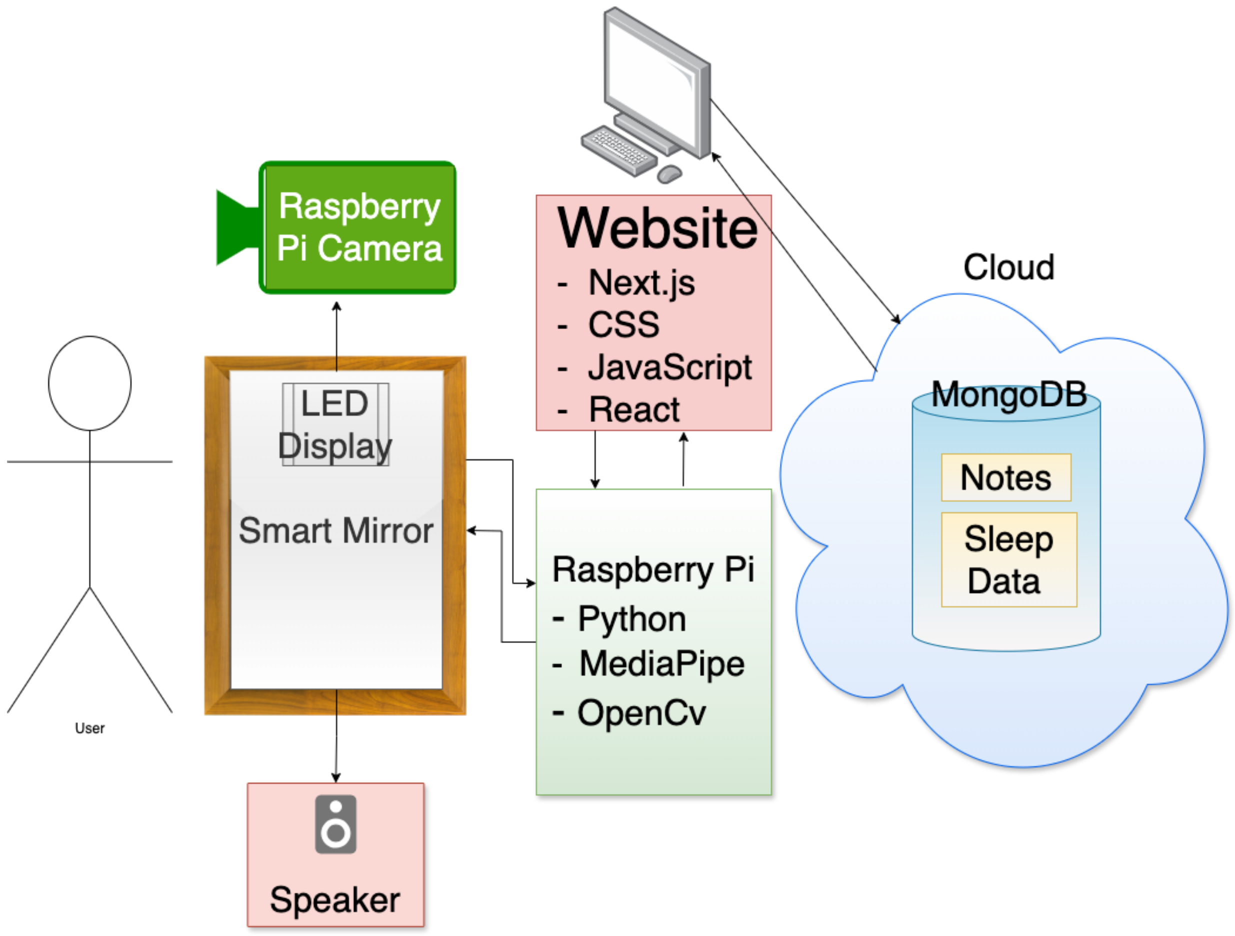


Figure 5-1 Architecture Diagram

# 6 Raspberry Pi

The Raspberry Pi is the heart of this project, running code, displaying the webpage and more.

## 6.1 Features

The Raspberry Pi is a very popular board for its value for money, wide range of flexibility and its simplicity in programming. I chose this board because of its wide range of ports, its vast amount of documentation, its popularity when working with image processing projects, its ability to run python. The Pi board was also compatible with the display and camera I was going to use. The boards size was ideal as it would be easy to attach to the mirror. Its USB-C power supply port would also make it so I could power the board with a portable charger. The board version, Raspberry Pi model 4 B 4GB was picked because it had enough power and storage to run the mirror and do facial detection.

## 6.2 Setting up the Pi

Out of the box the Pi needs to be set up. On my laptop I downloaded the Raspberry Pi Imager software from the official Raspberry Pi website. With this software I can choose the board I'm using and the operating system I want. I insert a microSD card into my laptop download the file given to me by the Raspberry Pi Imager and place it into my Raspberry Pi.

## 6.3 Installing Software

With my Raspberry Pi now running, I can install any needed software. Throughout my project I downloaded and used multiple software. Due to Raspberry Pi recently updating their operating system to Debian 12 Bookworm [11], this meant some software wasn't compatible or supported anymore. It also made installing needed software more difficult. I ran into trouble when installing software like TensorFlow [12] and MongoDB [13]. As a result, I needed to find work arounds. I used OpenCV and MongoDB Atlas as alternatives.

# 7 Mirror

Building the mirror was an important step in my project. The mirror had to have a modern design to fit in with other home automation tools that fit into modern homes.

## 7.1 Construction

The mirror is broken into several parts. The frame, the glass, an LED display, a cardboard panel, and electronic components such as wires, speakers, the Raspberry Pi. When putting together the mirror I wanted to give it a sleek design. Hiding any wires from the user. A portable charger is used to power the Pi board. The other electrical components are then connected to the Pi.

## 7.2 One-Way Mirror Glass

Mirrored glass is usually made with aluminium coating. This gives the mirror the reflection affect. One-way mirrors or “privacy” mirrors are more complicated. “Truly there is no such thing as a one-way mirror” [14] as they are actually glass, with partially reflective coatings, which have a reflective-high reflection from the coated side and a relatively low light transmission. To get a one-way mirror affect from these glass panels, the ratio of the amount of light reflected from the glass needs to exceed the amount of light transmitted through the glass. What this means is to get the one-way mirror affect to work with my mirror, the light in the room that the user is in needs to be brighter than the cardboard panel that will be at the back of the mirror. To achieve this the cardboard panel will be painted black.

## 7.3 LED Display

With the one-way mirror affect achieved, an LED display is placed behind the glass. This way the light from the display can pass through the glass panel. The user can now see themselves and the display. The display is connected to the Raspberry Pi board with a HDMI cable and a micro-USB cable.

## 7.4 Input and Sound

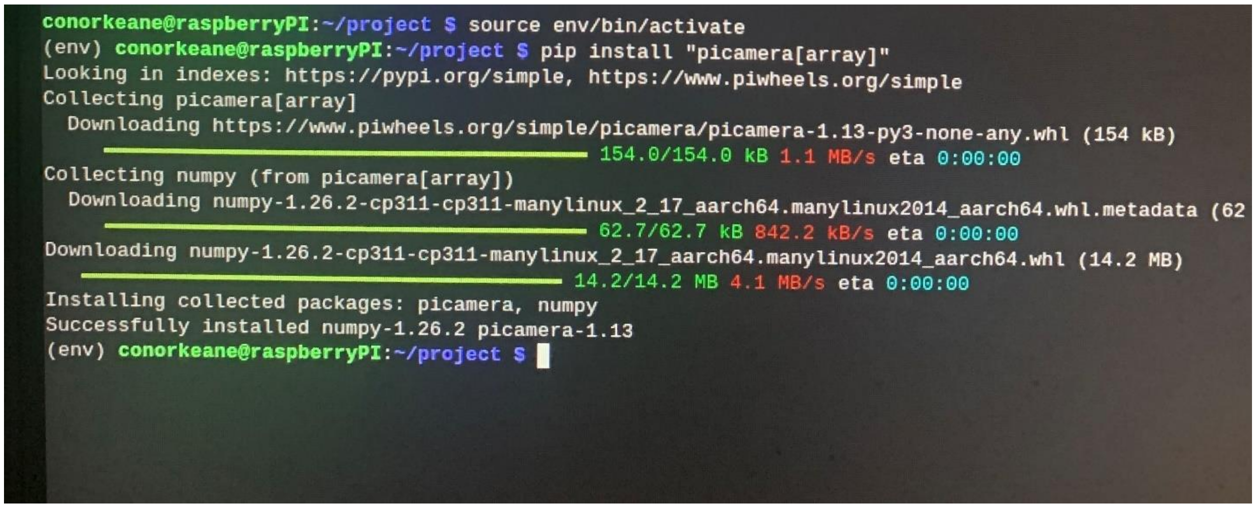
For sound a USB-wired speaker is connected to the Raspberry Pi. For the user’s input, the smart mirror a Bluetooth USB keyboard and mouse is connected to the Pi also. While the LED display used in the smart mirror is touch screen compatible, the glass is too thick for the display to detect the user’s input. As a result, the user can use a keyboard and mouse. This navigation system does make the mirror seem less advance. On the other hand, the use of a keyboard and mouse with the mirror makes it so the glass on the mirror will have less smudges, and dirt from consistently being pressed on.

## 8 Raspberry Pi Camera Module 3

The Pi Camera Module would help capture any images I needed to detect the users sleep. The camera is connected to the Pi through the board’s camera port.

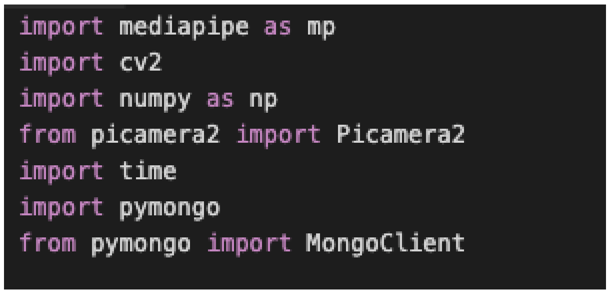
### 8.1 Installation and Usage

To install the software needed to run the camera module, a library needs to be downloaded. I did this in the terminal.



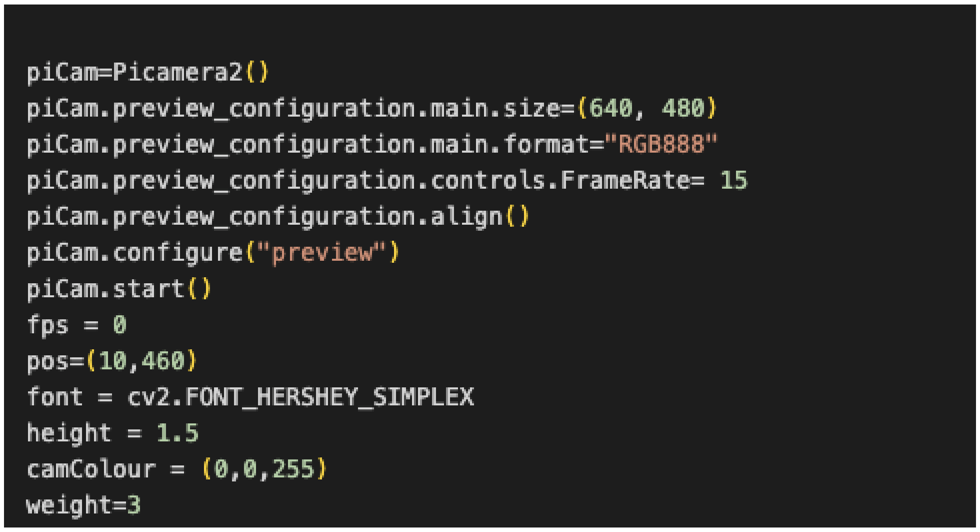
```
conorkeane@raspberryPI:~/project $ source env/bin/activate
(env) conorkeane@raspberryPI:~/project $ pip install "picamera[array]"
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting picamera[array]
  Downloading https://www.piwheels.org/simple/picamera/picamera-1.13-py3-none-any.whl (154 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 154.0/154.0 kB 1.1 MB/s eta 0:00:00
Collecting numpy (from picamera[array])
  Downloading numpy-1.26.2-cp311-cp311-manylinux_2_17_aarch64.manylinux2014_aarch64.whl.metadata (62
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 62.7/62.7 kB 842.2 kB/s eta 0:00:00
Downloading numpy-1.26.2-cp311-cp311-manylinux_2_17_aarch64.manylinux2014_aarch64.whl (14.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 14.2/14.2 MB 4.1 MB/s eta 0:00:00
Installing collected packages: picamera, numpy
Successfully installed numpy-1.26.2 picamera-1.13
(env) conorkeane@raspberryPI:~/project $ 
```

From here the library Picamera2 can be imported into my code



```python
import mediapipe as mp
import cv2
import numpy as np
from picamera2 import Picamera2
import time
import pymongo
from pymongo import MongoClient
```

From here I can use the camera to create a frame or image of a live feed. This is done by the pi taking in an array of pixels.



```
piCam=Picamera2()
piCam.preview_configuration.main.size=(640, 480)
piCam.preview_configuration.main.format="RGB888"
piCam.preview_configuration.controls.FrameRate= 15
piCam.preview_configuration.align()
piCam.configure("preview")
piCam.start()
fps = 0
pos=(10,460)
font = cv2.FONT_HERSHEY_SIMPLEX
height = 1.5
camColour = (0,0,255)
weight=3
```

I can choose what colour I want the camera image to be in (RGB), the camera quality (640p by 480p), the framerate and more.

## 9 OpenCV

Open Computer Vision Library or OpenCV is an open-source computer vision software library. It is the worlds biggest computer vision library, with over 2500 algorithms [8]. Released back in in June 2000, 23 years ago, OpenCV is still popular to this day. With lots of documentation and useful forms and videos, I decided to use OpenCV. This library would help me with image processing and give an output image of any facial detection I was doing.

### 9.1 Image Processing

I used OpenCV to help create an image window. This window would display an array of pixels gotten from the Raspberry Pi Camera Module. This array of pixels with the help of OpenCV could display any text I need to put onto the image. For example, in the code below I have gotten the image (in this case called frame) from the camera module. I can now use OpenCV to add text to the image. This will come in handy for debugging code and overall appearance.

```
cv2.line(image,p1,p2,(255,0,0),3)

cv2.putText(frame,text,(20,50), cv2.FONT_HERSHEY_SIMPLEX,2,(0,255,0),2)
cv2.putText(frame, "x: " + str(np.round(x,2)),(500,50),font,1,(0,0,255),2)
cv2.putText(frame, "y: " + str(np.round(y,2)),(500,100),font,1,(0,0,255),2)
cv2.putText(frame, "z: " + str(np.round(z,2)),(500,150),font,1,(0,0,255),2)
cv2.putText(frame, "Mouth: {}".format(mouth_status), (20, 100), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)

timeEnd = time.time()
loopTime = timeEnd-timeStart
fps =.9*fps + .1*(1/loopTime)

cv2.putText(frame, str(int(fps))+ ' FPS', pos, font, height, camColour, weight)
```

OpenCV also lets me then display the image.

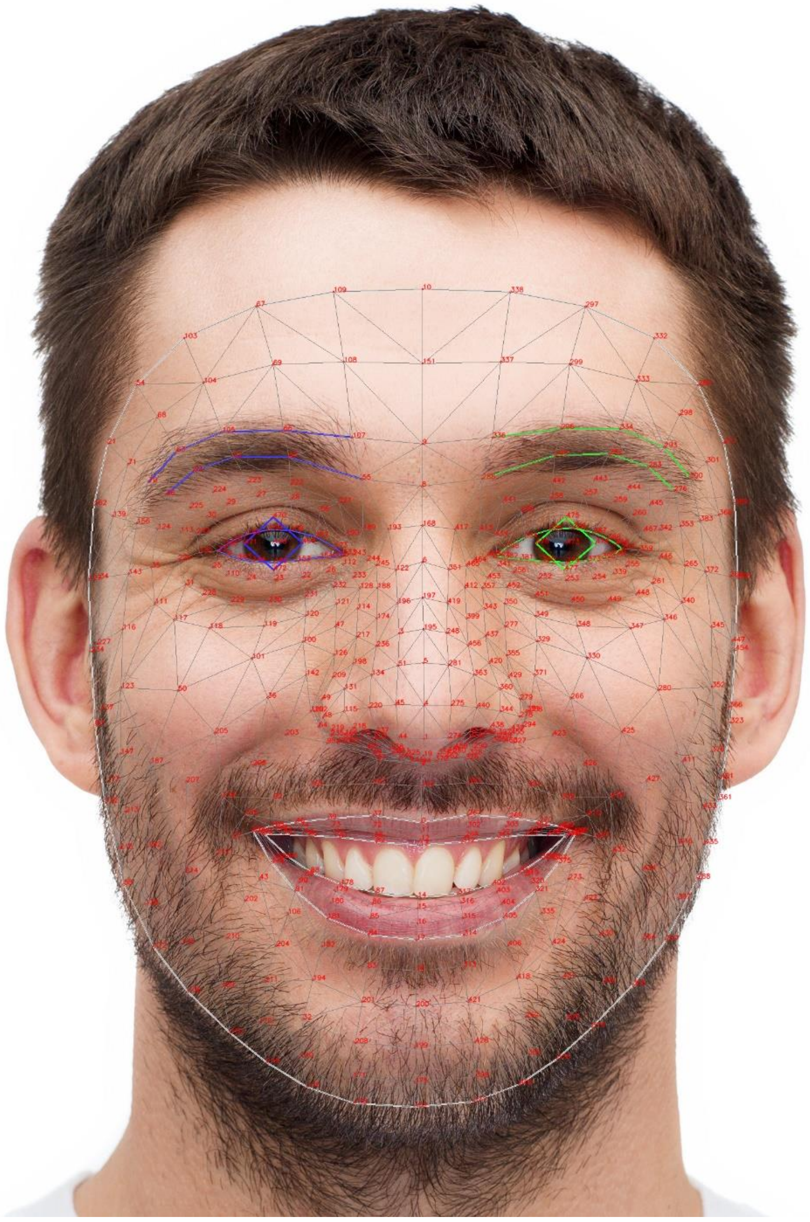
```
# mp_drawing.draw_landmarks(image, results.face_landmarks, mp_holistic.FACEMESH_CONTOURS, drawing_spec)
cv2.imshow("piCam", frame) # Uncomment to see clear camera
#cv2.imshow("piCam", image) #Uncomment to see Landmarks on camera
```

# 10 MediaPipe

MediaPipe is a framework made by Google to help with building machine learning and artificial intelligence programs. MediaPipe offer different solutions and uses such as being able to detect, body poses, hand gestures, eye tracking, facial detection and more. For the smart mirror the Face Landmark Detection solution is used. MediaPipe offer code examples, guides and other documentation [15].

## 10.1 Landmarks

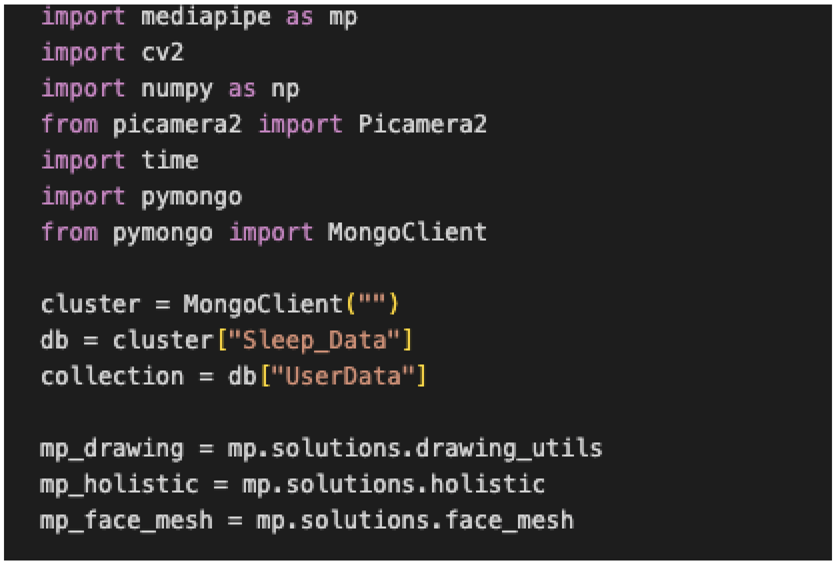
To understand how the user face co-ordinates are gotten a basic understanding of landmarks is needed. MediaPipe Landmark Detection is originally intended for face expressions, however as each landmark has a co-ordinate, and is on an x, y and z 3-dimensional graph, this can help provide insight and an estimate into the way the user is facing. From this image found on MediaPipes documents [16], each landmark has a number.



Thanks to this feature I can get the co-ordinate value of certain areas of the face, store these values. If the values change, then I can detect if the user is facing left right or straight ahead.

## 10.2 Set-up

MediaPipe is installed on the Pi using a pip install command. It is then imported into the code.

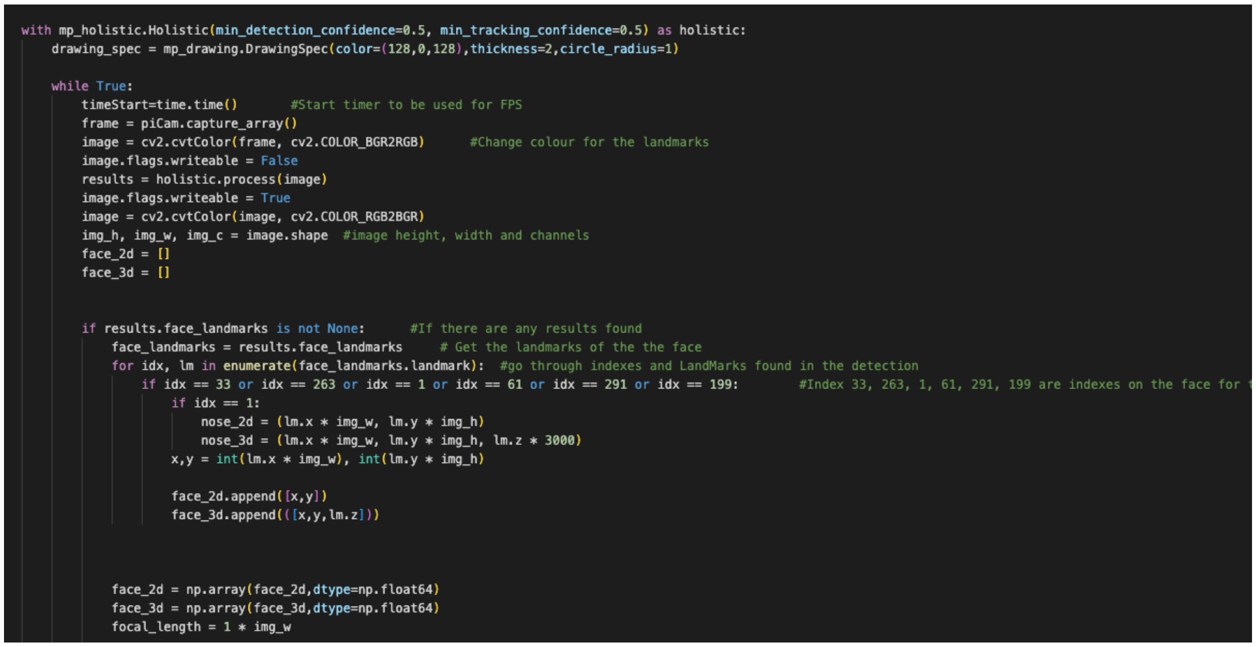


```
import mediapipe as mp
import cv2
import numpy as np
from picamera2 import Picamera2
import time
import pymongo
from pymongo import MongoClient

cluster = MongoClient("")
db = cluster["Sleep_Data"]
collection = db["UserData"]

mp_drawing = mp.solutions.drawing_utils
mp_holistic = mp.solutions.holistic
mp_face_mesh = mp.solutions.face_mesh
```

Once imported, features like drawing_utils and fac_mesh are set up. This needs to be done so the landmarks can be drawn on any face that is detected.

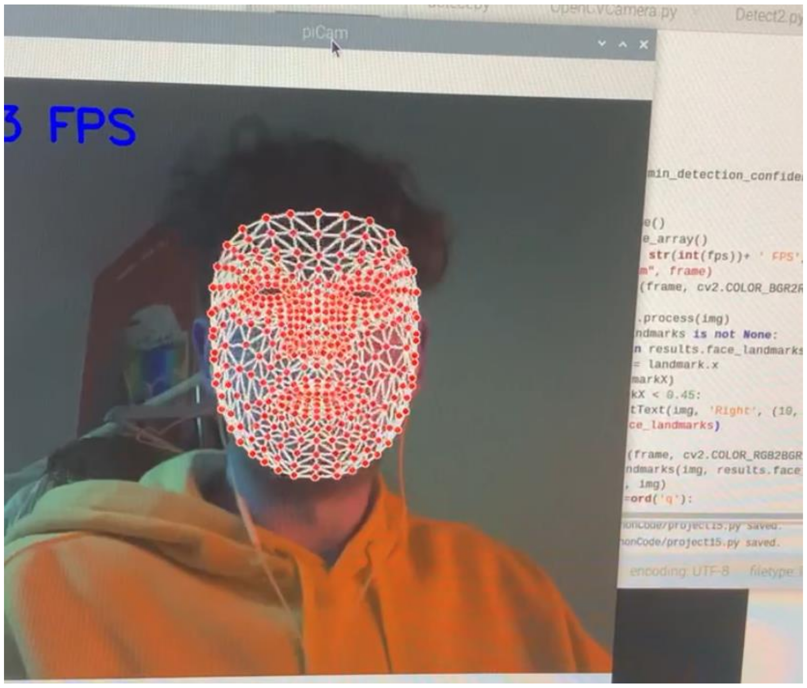
### 10.3 Detection and Drawing

Next, I define the size, colour and thickness of the landmarks that will be drawn. The colour of the image that I will be capturing will also need to be changed from its original RBG888 to BGR. This needs to be done so MediaPipe can detect the face and draw the landmarks. Once complete the image can be changed back. I then check to see if any facial detection is found, if so, landmarks are drawn. In my code I use the landmark points 33, 263, 1, 61, 291, 199. These make up the nose, ears forehead and other places. If the landmark 1 is detected (the nose) then I take in the co-ordinates of the face.



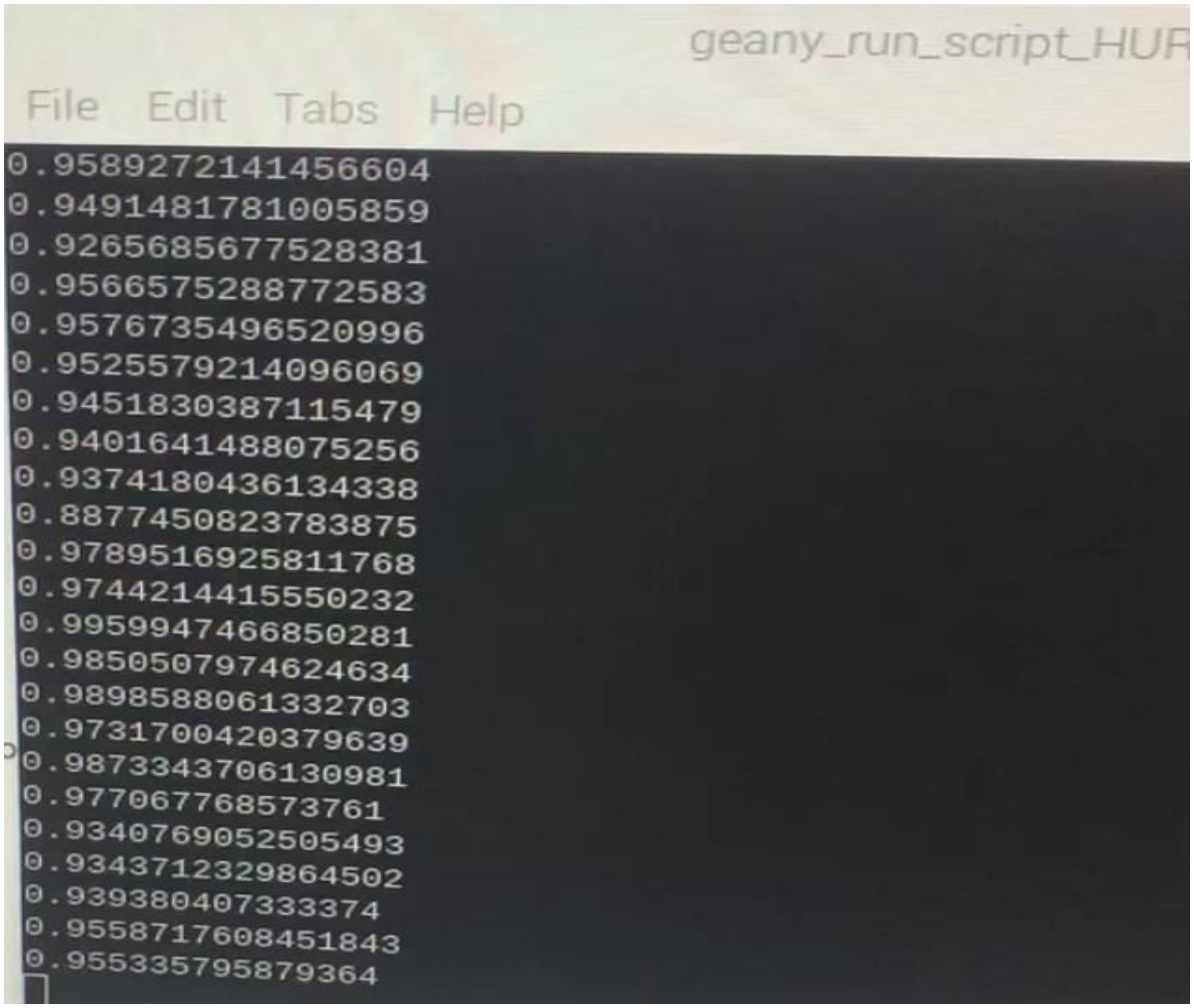
```
with mp_holistic.Holistic(min_detection_confidence=0.5, min_tracking_confidence=0.5) as holistic:
    drawing_spec = mp_drawing.DrawingSpec(color=(128,0,128),thickness=2,circle_radius=1)

    while True:
        timeStart=time.time()       #Start timer to be used for FPS
        frame = piCam.capture_array()
        image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)      #Change colour for the landmarks
        image.flags.writeable = False
        results = holistic.process(image)
        image.flags.writeable = True
        image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
        img_h, img_w, img_c = image.shape  #image height, width and channels
        face_2d = []
        face_3d = []


        if results.face_landmarks is not None:      #If there are any results found
            face_landmarks = results.face_landmarks     # Get the landmarks of the the face
            for idx, lm in enumerate(face_landmarks.landmark):  #go through indexes and LandMarks found in the detection
                if idx == 33 or idx == 263 or idx == 1 or idx == 61 or idx == 291 or idx == 199:        #Index 33, 263, 1, 61, 291, 199 are indexes on the face for
                    if idx == 1:
                        nose_2d = (lm.x * img_w, lm.y * img_h)
                        nose_3d = (lm.x * img_w, lm.y * img_h, lm.z * 3000)
                    x,y = int(lm.x * img_w), int(lm.y * img_h)

                    face_2d.append([x,y])
                    face_3d.append(([x,y,lm.z]))



            face_2d = np.array(face_2d,dtype=np.float64)
            face_3d = np.array(face_3d,dtype=np.float64)
            focal_length = 1 * img_w
```

This is what the landmarks look like during the drawing stage.



## 10.4 Calculating

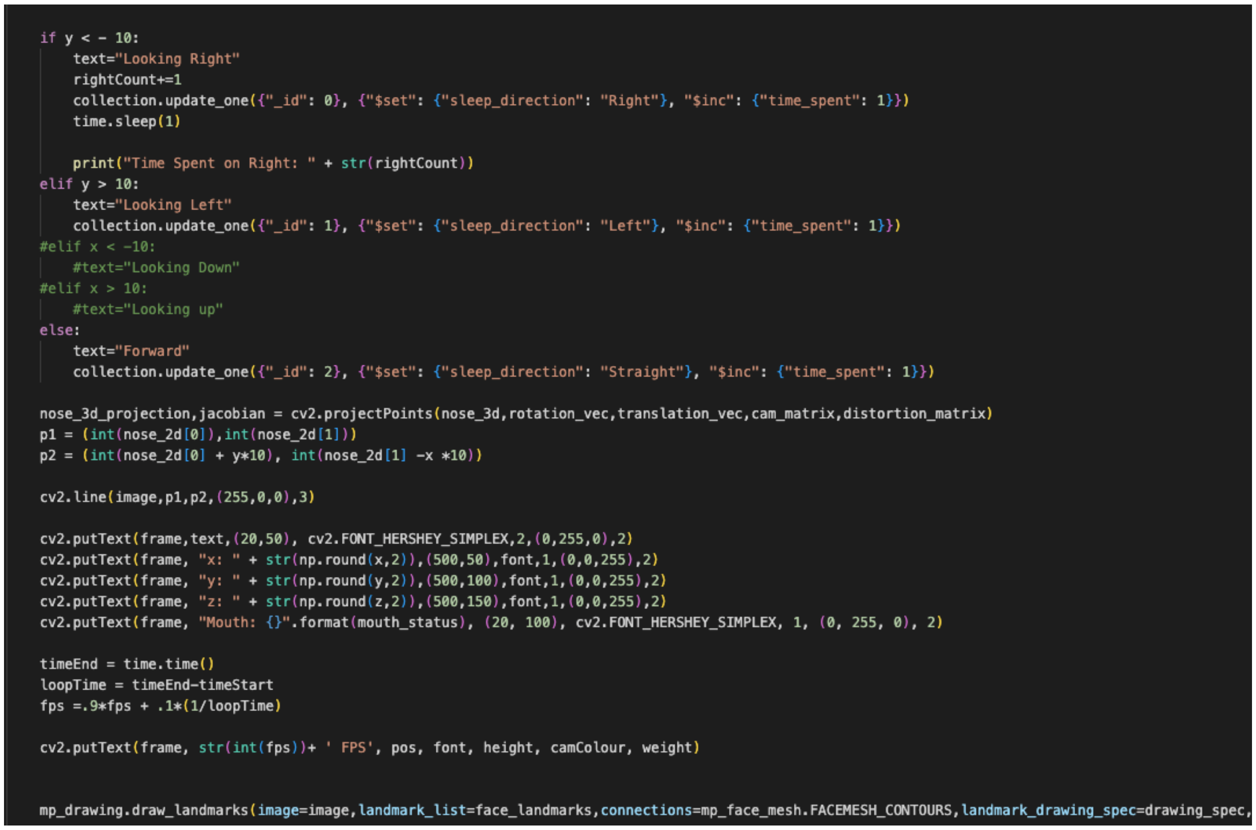
The after getting the co-ordinates, they are converted into a NumPy array for further equations. I get the focal length with the image width. I then get the camera matrix with the focal length, image width and height. The image then needs to be checked for distortion. The facial pose is gotten using cv2.solvePnP, the rotation vector is then turned into a rotation matrix. The values up to this stage are very small. To get better readings they need to be multiplied.



```
0.9589272141456604
0.9491481781005859
0.9265685677528381
0.9566575288772583
0.9576735496520996
0.9525579214096069
0.9451830387115479
0.9401641488075256
0.9374180436134338
0.8877450823783875
0.9789516925811768
0.9744214415550232
0.9959947466850281
0.9850507974624634
0.9898588061332703
0.9731700420379639
0.9873343706130981
0.977067768573761
0.9340769052505493
0.9343712329864502
0.939380407333374
0.9558717608451843
0.955335795879364
```

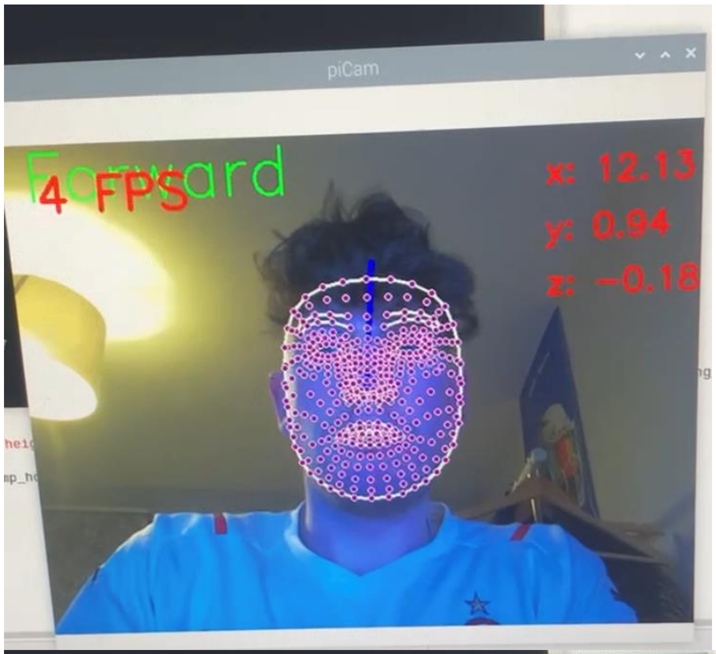
## 10.5 Display and Record

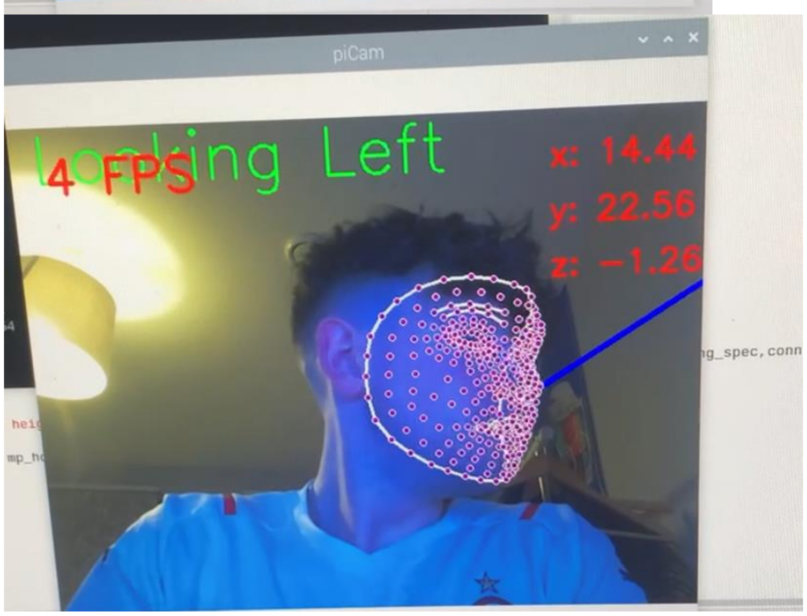
With more accurate co-ordinate values a simple if statement can work to see if the user is facing left, right or straight ahead.



```
if y < - 10:
    text="Looking Right"
    rightCount+=1
    collection.update_one({"_id": 0}, {"$set": {"sleep_direction": "Right"}, "$inc": {"time_spent": 1}})
    time.sleep(1)

    print("Time Spent on Right: " + str(rightCount))
elif y > 10:
    text="Looking Left"
    collection.update_one({"_id": 1}, {"$set": {"sleep_direction": "Left"}, "$inc": {"time_spent": 1}})
#elif x < -10:
    #text="Looking Down"
#elif x > 10:
    #text="Looking up"
else:
    text="Forward"
    collection.update_one({"_id": 2}, {"$set": {"sleep_direction": "Straight"}, "$inc": {"time_spent": 1}})

nose_3d_projection,jacobian = cv2.projectPoints(nose_3d,rotation_vec,translation_vec,cam_matrix,distortion_matrix)
p1 = (int(nose_2d[0]),int(nose_2d[1]))
p2 = (int(nose_2d[0] + y*10), int(nose_2d[1] -x *10))

cv2.line(image,p1,p2,(255,0,0),3)

cv2.putText(frame,text,(20,50), cv2.FONT_HERSHEY_SIMPLEX,2,(0,255,0),2)
cv2.putText(frame, "x: " + str(np.round(x,2)),(500,50),font,1,(0,0,255),2)
cv2.putText(frame, "y: " + str(np.round(y,2)),(500,100),font,1,(0,0,255),2)
cv2.putText(frame, "z: " + str(np.round(z,2)),(500,150),font,1,(0,0,255),2)
cv2.putText(frame, "Mouth: {}".format(mouth_status), (20, 100), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)

timeEnd = time.time()
loopTime = timeEnd-timeStart
fps =.9*fps + .1*(1/loopTime)

cv2.putText(frame, str(int(fps))+ ' FPS', pos, font, height, camColour, weight)


mp_drawing.draw_landmarks(image=image,landmark_list=face_landmarks,connections=mp_face_mesh.FACEMESH_CONTOURS,landmark_drawing_spec=drawing_spec,
```

These values can then be printed to the image for a better understanding and debugging of the code.





These readings of if the user is facing left, right or straight ahead are saved to MongoDB Atlas. A cloud version of MongoDB. This data can then be used on the webpage to display the data to the user.

## 10.6 Forums and Sources for Code

For help on my landmark code, I used the following. This YouTube video showed how to get the co-ordinates of the user's head and landmarks [17]. He also posted a page about the code [18]. OpenAI was also used to help code and debug [19].

# 11 Web Page

On the LED display behind the glass panel is a simple web page. I designed this web page using Next.js [20] and Node.js [21]. A majority of this code was done in my college module Full Stack Development [22]. Changes and alternative code that was then put in from modules like Cloud Computing [23] and App Development [24]. These modules gave me a good understanding of how to display a web page to a user. To make my mirror a better user experience I wanted to add more features to the mirror.

## 11.1 Weather API

To let the user check the weather I added a weather feature. This weather page uses an API from Open Weather [4]. To help understand this API I used YouTube videos [25] and documentation [26]. The weather page asks a user to enter a location and then gives a detailed page of the weather in that location.

## 11.2 Spotify API

I added a Spotify feature to allow the user to listen to music. This API proved to be a lot more difficult to set up due to its authentication, keys and tokens. To help with these problems I used YouTube videos [27] and API documentation [3]. When debugging the problems I used stack overflow [28] documentation and ai assistance [19].

## 11.3 Displaying Data

To display data to the user I used the react charts library [29]. I also watch YouTube videos [30] on how to present graphs to the user. With any bugs or issues again, I used forums and ai assistance [19].

## 11.4 Design

I tried to improve my webpage design by adding gifs and images as backgrounds. I used animated CSS designs found online [31] [32].

# 12 GitHub

All my code and work were pushed to GitHub throughout the project. I had 2 main GitHub repositories. One for my raspberry pi scripts, the other for my web page. I changed repositories halfway through the semester due to adapting to next.js and using code my lecture had given [22], during lectures. This code used a good next.js structure and already had connected to a local MongoDB database. As a result, I made a new repository where I worked on changing this code. Merging my pervious code with this new code[43].

# 13 AWS

The Node.js backend is ran on a AWS Ec2 server. This was set up during a cloud computing lecture.

# 14 Problem Solving

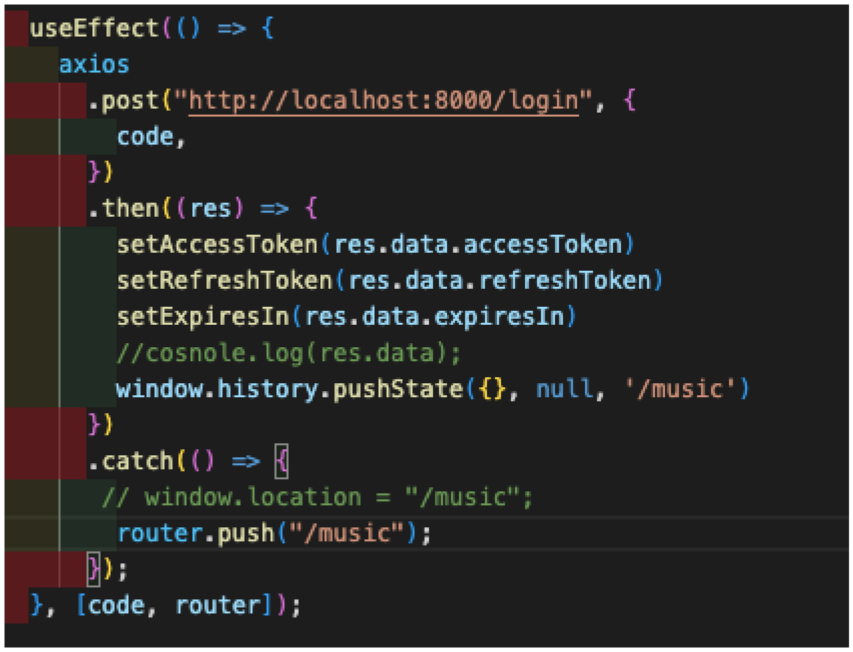
Throughout my project I ran into problems. To debug these problems, I looked at relevant documentation, YouTube tutorials, forms, and used ai assistance.

## 14.1 Installation Problems

When working with the Raspberry Pi I needed a lost of frameworks and libraries. The Raspberry Pi got a recent update from Debian 11 to 12. This update made a lot of libraries no longer supported. These problems resulted in me trying to find work arounds. TensorFlow and MongoDB were popular problems I came across. MongoDB not even being supported on Debian 12 [33] meant that I would have to create a virtual environment on the PI [34] and then downgrade the software and proceed with the installation. This work around seemed time consuming. As a result, I opted for using MongoDB Atlas [35] as a quicker work around.

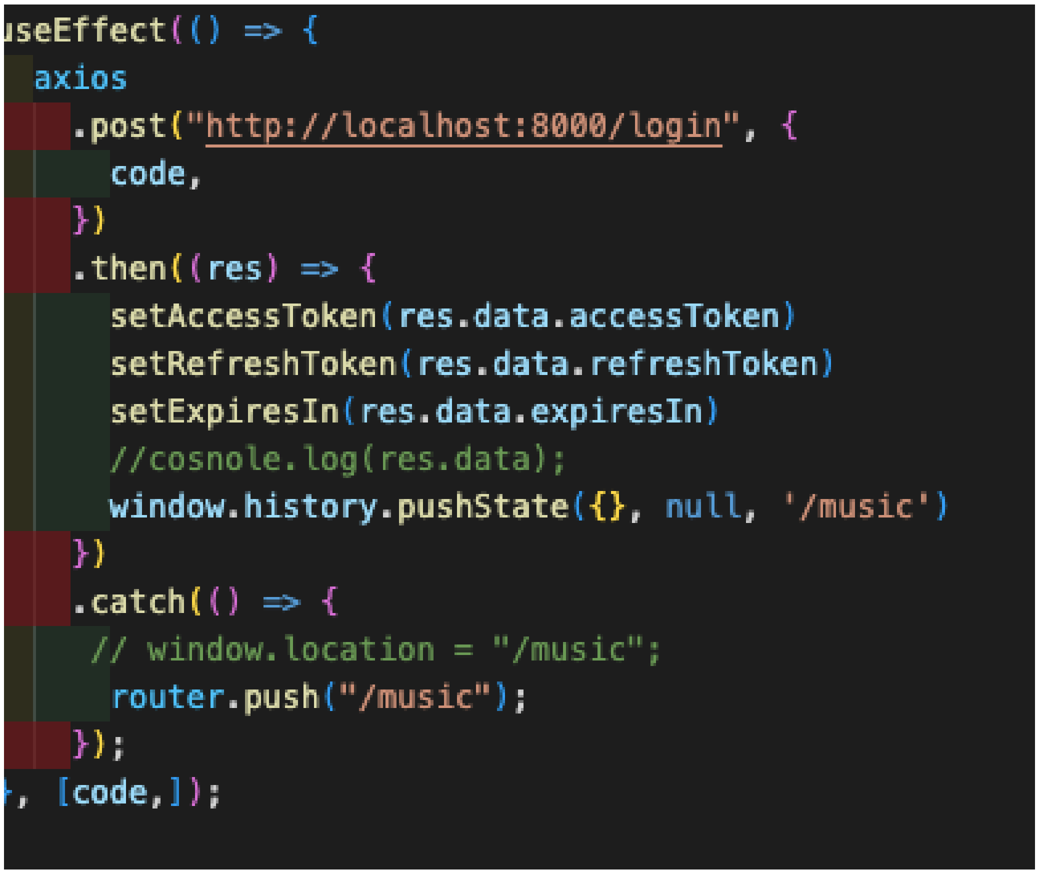
## 14.2 Spotify API

Trying to get Spotify working on my web application took time and a lot of problem solving. The video I followed [27], used React. I found react and next.js similar and as a result believed taking the react code and integrating it into next.js would be easy. I ran into many bugs. Next.js focuses heavily on its file set up. This meant integrating the react code to my Next.js code would have to be filed correctly. Another issue with Spotify was the token I was given. This token resets very quickly meaning that the token could expire before the user gets a chance to log in. Resulting in a failed log in. The YouTube tutorial shows a work around by refreshing the token every few minutes. This solution did not work with my react code. To get the new token you would have to refresh the page and take the token from the URL. For example, if your webpage is http://localhost:3000/music, the API token is inserted into the end like this http://localhost:3000/music{token API key}. The YouTube videos work around was to refresh the page and extract the token from the URL using window.history.pushState({}, null, '/music'). Then navigating the user back to the correct URL window.location = '/music';. In next.js this did not work as next.js uses routing to navigate. As a result, I had to change the code to router.push("/music"). This worked until Next.js gave a warning to include the router in its useEffect(). Like so



```
useEffect(() => {
  axios
    .post("http://localhost:8000/login", {
      code,
    })
    .then((res) => {
      setAccessToken(res.data.accessToken)
      setRefreshToken(res.data.refreshToken)
      setExpiresIn(res.data.expiresIn)
      //cosnole.log(res.data);
      window.history.pushState({}, null, '/music')
    })
    .catch(() => {
     // window.location = "/music";
      router.push("/music");
    });
}, [code, router]);
```

This resulted in a bug. When the user pressed the log in button it would successfully log them in. However, once logged in the http://localhost:8000/login post request would be sent to the back end every second. This resulted in thousands of post requests. I tried to find the solution to this bug online but could not find a solution for something so small yet devastating. I decided to use AI assistance [19] to debug the problem. Using console.log and its recommendations I was able to restructure my code, back track and log the problems to the console. Eventually the bug was resolved I took the router out of the use effect [code, router] now became just [code], and decided to live with the Next.js warning when building my code.



```
useEffect(() => {
  axios
    .post("http://localhost:8000/login", {
      code,
    })
    .then((res) => {
      setAccessToken(res.data.accessToken)
      setRefreshToken(res.data.refreshToken)
      setExpiresIn(res.data.expiresIn)
      //cosnole.log(res.data);
      window.history.pushState({}, null, '/music')
    })
    .catch(() => {
     // window.location = "/music";
      router.push("/music");
    });
}, [code,]);
```

## 15 Ethics

When capturing and displaying a person’s data, everyone always asks about how safe is that data from being stolen. Many people I talked to about my project found it creepy or unsafe that a camera would be watching them sleep. For my project I did not want the user to feel unsafe. As a result, there is no physical recording of the user in their sleep. The code simply logs to the data base if the user is sleeping left, right or straight ahead. This means the user does not have to worry about any potential person stealing a recording of them sleeping. With the Spotify feature, the user does not use their own account credentials. Again, this means the user does not have to worry about potentially losing their account of data.

## 16 Conclusion

From my final year project, I have gained a lot of experience and knowledge on areas I previously had not. I learned the importance of time management, teamwork, planning, research and more. For the first time, while working on a college assignment I engaged with people outside of my college for guidance and help []. My project thought me not just electronics and coding but also physics. Coming away from this project I also have something I want to continue to work on. I put my own money into this project and happy I did so, as I can continue to work on the project once graduating. In my personal opinion I have built an actual product that people would buy. I know this because people have actual bought peoples smart mirrors online [36]. While the product might not be as good as theirs, it does stand out with its use of machine learning and artificial intelligence. My goal was to be able to monitor a user's sleep in some format and I achieved this. Using a framework that had not been used for monitoring something like sleep. Many other projects had to train ai models to monitor sleep [], I feel proud that when faced with the challenge of trying to detect a persons head position, I was warned on the difficulty this would bring and that it might not be possible. Through research and creative thinking I was able to achieve this despite the doubts and concerns.

## 17 References

[1] Magicmirror.builders. (2016). MagicMirror².[online]Available at: https://magicmirror.builders/.

[2] Author, P.F.I. the lead, guides, owner of R. com M. goal is to help you with your R.P. problems using detailed, life, tutorials I. real and experience, I. a L. system administrator with web developer (2022). *What Is A Raspberry Pi? (Hardware, Software, Goal & Usage)*. [online] RaspberryTips. Available at: https://raspberrytips.com/what-is-a-raspberry-pi/#:~:text=A%20Raspberry%20Pi%20is%20a%20small%20computer%20that%E2%80%99s.

[3] Developer.spotify.com. (n.d.). *Getting started with Web API | Spotify for Developers*. [online] Available at: https://developer.spotify.com/documentation/web-api/tutorials/getting-started.

[4] OpenWeatherMap.org (2015). *Weather API - OpenWeatherMap*. [online] Openweathermap.org. Available at: https://openweathermap.org/api.

[5] www.youtube.com. (n.d.). *Smart Mirror Touchscreen (with Face ID) using Raspberry Pi 4 | Full Tutorial*. [online] Available at: https://www.youtube.com/watch?v=RWjvJq4Zabk&t=182s [Accessed 29 Apr. 2024].

[6] Raspberry Pi (n.d.). *Raspberry Pi 4 Model B specifications*. [online] Raspberry Pi. Available at: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/.

[7] GeeksforGeeks. (2018). *Image Processing in Python*. [online] Available at: https://www.geeksforgeeks.org/image-processing-in-python/.

[8] OpenCV (2018). *About OpenCV*. [online] OpenCV. Available at: https://opencv.org/about/.

[9] Ltd, R.P. (n.d.). *Raspberry Pi Camera Module 3*. [online] Raspberry Pi. Available at: https://www.raspberrypi.com/products/camera-module-3/.

[10] Authors, T.M. (n.d.). *mediapipe: MediaPipe is the simplest way for researchers and developers to build world-class ML solutions and applications for mobile, edge, cloud and the web.* [online] PyPI. Available at: https://pypi.org/project/mediapipe/.
[11] Author, S.L. Bookworm – the new version of Raspberry Pi OS. [online] Raspberry Pi. Available at: Bookworm — the new version of Raspberry Pi OS - Raspberry Pi

[12] The MagPi magazine. (2018). *How to install TensorFlow on Raspberry Pi*. [online] Available at: https://magpi.raspberrypi.com/articles/tensorflow-ai-raspberry-pi.

[13] GeeksforGeeks. (2022). *How to Install MongoDB on Raspberry pi?* [online] Available at: https://www.geeksforgeeks.org/how-to-install-mongodb-on-raspberry-pi/ [Accessed 29 Apr. 2024].

[14] Connaughton, P. (2023), E-mail to Conor Keane, 24 November.

[15] Boesch, G. (2023). *MediaPipe: Google's Open Source Framework for ML solutions (2023 Guide)*. [online] viso.ai. Available at: https://viso.ai/computer-vision/mediapipe/#:~:text=Computer%20Vision%20Teams-.

[16] Google for Developers. (n.d.). *Face landmark detection guide | MediaPipe*. [online] Available at: https://developers.google.com/mediapipe/solutions/vision/face_landmarker#get_started.

[17] AI (2021). *Head Pose Estimation with MediaPipe and OpenCV in Python - OVER 100 FPS!!! YouTube*. Available at: https://www.youtube.com/watch?v=-toNMaS4SeQ [Accessed 27 Feb. 2024].

[18] R, J. (2023). *Real-Time Head Pose Estimation FaceMesh with MediaPipe and OpenCV: A Comprehensive Guide*. [online] Medium. Available at: https://medium.com/@jaykumaran2217/real-time-head-pose-estimation-facemesh-with-mediapipe-and-opencv-a-comprehensive-guide-b63a2f40b7c6 [Accessed 21 Feb. 2024].

[19] OpenAi ChatGPT. (2024). ChatGPT response to Conor Keane

[20] Vercel (2024). *Next.js by Vercel - The React Framework*. [online] nextjs.org. Available at: https://nextjs.org/.

[21] Node.js (2023). *Node.js*. [online] Node.js. Available at: https://nodejs.org/en.

[22] B. O'Shea. "fullsStackWeek11lab2", Lecture, Full Stack Development, Atlantic Technological University, Galway, 2023.

[23] B. O'Shea. "week10promises", Lecture, Mobile App Development, Atlantic Technological University, Galway, 2024

[24] B. O'Shea. "week5c", Lecture, Cloud Computing, Atlantic Technological University, Galway, 2023

[25] Code Commerce (2022). *Build A React JS Weather App - OpenWeatherMap API - Tutorial*. *YouTube*. Available at: https://www.youtube.com/watch?v=UjeXpct3p7M [Accessed 24 November. 2023].

[26] openweathermap.org. (n.d.). *Current weather data - OpenWeatherMap*. [online] Available at: https://openweathermap.org/current.

[27] www.youtube.com. (n.d.). *How To Build A Better Spotify With React*. [online] Available at: https://www.youtube.com/watch?v=Xcet6msf3eE [Accessed 24 Apr. 2024].

[28] Stack Overflow. (n.d.). *Routing with Spotify API*. [online] Available at: https://stackoverflow.com/questions/68871481/routing-with-spotify-api [Accessed 24 Apr. 2024].

[29] react-chartjs-2.js.org. (n.d.). *Vertical Bar Chart | react-chartjs-2*. [online] Available at: https://react-chartjs-2.js.org/examples/vertical-bar-chart [Accessed 2 March. 2024].

[30] www.youtube.com. (n.d.). *How To Make Beautiful Charts In Next.js*. [online] Available at: https://www.youtube.com/watch?v=15qMh8C1Wzo [Accessed 26 Apr. 2024].

[31] Juviler, J. (n.d.). *24 Creative and Unique CSS Animation Examples to Inspire Your Own*. [online] blog.hubspot.com. Available at: https://blog.hubspot.com/website/css-animation-examples.

[32] Cool CSS Animation. (n.d.). *Cool CSS Animation | Resources and tutorials for designers*. [online] Available at: https://coolcssanimation.com/.

[33] gamer_girl_2007_nah (2024). *How to install MongoDb on debian 12?* [online] Available at: https://www.reddit.com/r/debian/comments/1ab60vn/how_to_install_mongodb_on_debian_12/.

[34] Freek_Dijkstra (2023). Debian 12 Bookworm Support [online] Available at: https://www.mongodb.com/community/forums/t/debian-12-bookworm-support/253430/3.

[35] MongoDB (n.d.). *Managed MongoDB Hosting | Database-as-a-Service*. [online] MongoDB. Available at: https://www.mongodb.com/atlas.

[36] Virlos Magic Mirror V4-2 Way Mirror with Internal LCD Screen for Smart Mirror Projects-Great for Raspberry Pi. Available at: https://www.amazon.co.uk/Vilros-Internal-Mirrors-Projects-Great-Raspberry/dp/B0BJXD68JV/ref=sr_1_3?crid=3Q8Y60V8ZM4GZ&dib=eyJ2IjoiMSJ9.KEdm5-_sMwDtPCZS1CpTl6W2YeSnmAdjE8bEFXWLEqi26krtHikPGSH7MxHDtOKMqNWuSrXt3QCTfAAvrJfW3s-ktaexkCTxzZcvjkkFjcJnVoLPSp5G5dEfHx99-yDirBOVcOOxKqZSJwFa6QaxS_RbtaD6L4I_vtsJXa5UXh3iXDCyDEJ8lPczM6WCBlUvOSYD5WBt9Xk6NuPcxUgWmA.GZ2N2TSDgvb8LUpiA47WX9RuTxVPSV6Z7LiGXrlbUns&dib_tag=se&keywords=raspberry%2Bpi%2Bsmart%2Bmirror&qid=1714361755&sprefix=raspberry%2Bpi%2Bsmart%2Bmirror%2Caps%2C79&sr=8-3&ufe=app_do%3Aamzn1.fos.d7e5a2de-8759-4da3-993c-d11b6e3d217f&th=1

[37] The Sad Truth About Sleep-Tracking Devices and Apps. (2019). *The New York Times*. [online] 17 Jul. Available at: https://www.nytimes.com/2019/07/17/technology/personaltech/sleep-tracking-devices-apps.html.

[38] Restivo, J. (2023). *Sleep paralysis: Causes, symptoms, and treatments*. [online] Harvard Health. Available at: https://www.health.harvard.edu/diseases-and-conditions/sleep-paralysis-causes-symptoms-and-treatments.

[39] Healthline. (2017). *Mouth Breathing*. [online] Available at: https://www.healthline.com/health/mouth-breathing#what-is-mouth-breathing.

[40] conorkeane01 (2023). *conorkeane01/FYP-*. [online] GitHub. Available at: https://github.com/conorkeane01/FYP- [Accessed 29 Apr. 2024].

[41] conorkeane01 (2024). *conorkeane01/FYP_Rasp*. [online] GitHub. Available at: https://github.com/conorkeane01/FYP_Rasp [Accessed 29 Apr. 2024].

[42] conorkeane01 (2024). *conorkeane01/SmartMirrorWebsite*. [online] GitHub. Available at: https://github.com/conorkeane01/SmartMirrorWebsite [Accessed 29 Apr. 2024].

[43] O'Shea. B ("2024) Cloud Computing EC2 lecture. Atlantic Technological University Galway.