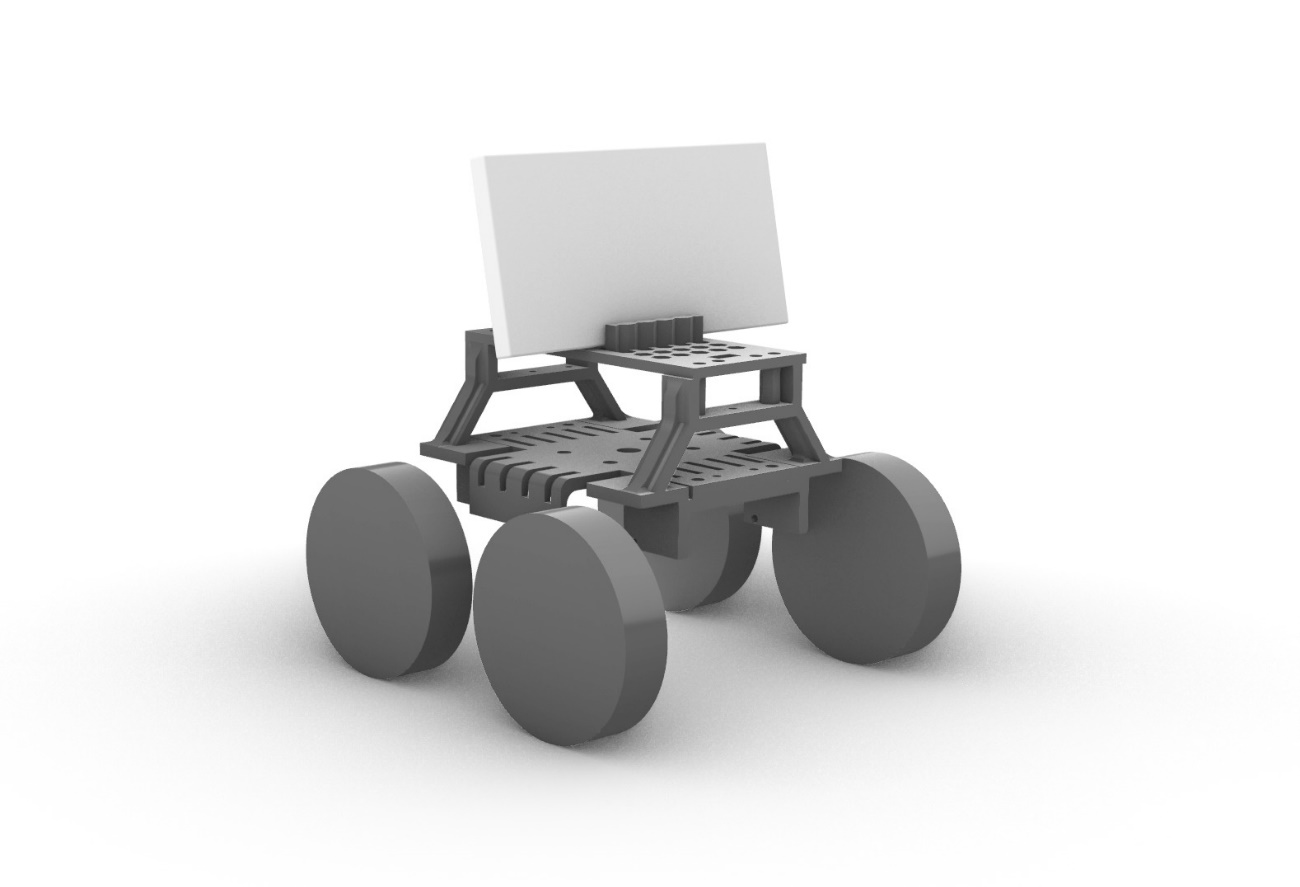
SECTION 1

# MODEL OF PATROLLING ROBOT

## Physical Appearance

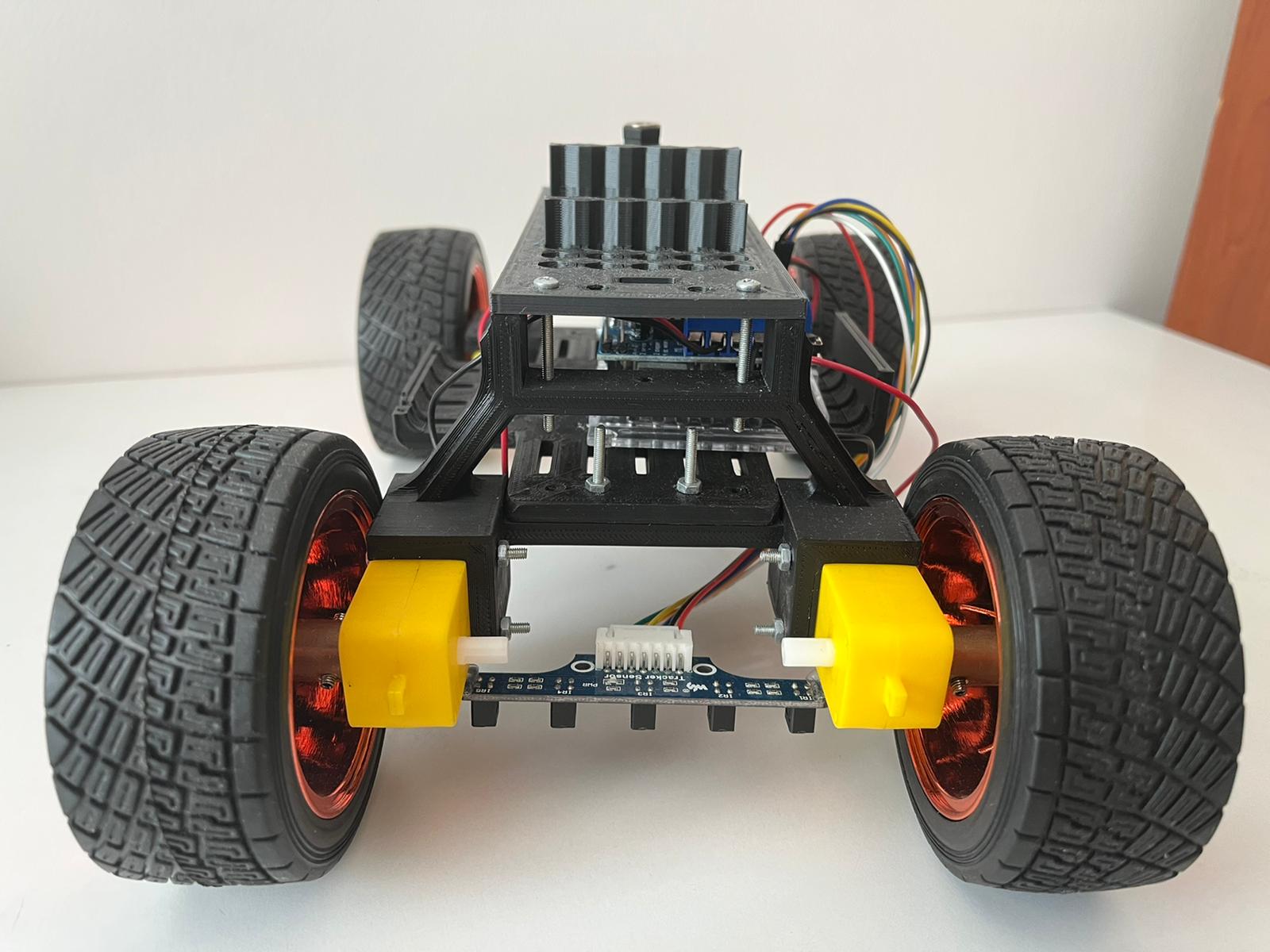
Industrial design techniques were used in vehicle design. The resulting design is shown in Figure 1-1. This design was produced by 3D printing method.

*Figure 1‑1: 3D Vehicle Model*

In this design, the vehicle chassis consists of 4 different parts. It is planned to carry the electronic parts of the vehicle, 1 of these parts is the Body, 2 of them are the Support and the other is the Roof. There are 3 separate zones for components in the design. The area in the lower part of the body is for placing the batteries and protrusions have been added so that the battery beds can be held there. The area between the Body and the Roof is designed to be a safe place for Arduino Uno and Motor Drive Shield. Since there will be a lot of Jumper cable connection between the Shield and the motor and the sensor, holes were left in the Body for fixing the Arduino to the Body. Parts have been added to fit the design integrity so that a Video Camera or phone can be placed on the platform above the Ceiling. Finally, the chassis connection slots of the engines were designed to be a limb of the support parts, and screw holes were added to fix each part to each other.



*Figure 1‑2: Realization of the Model*



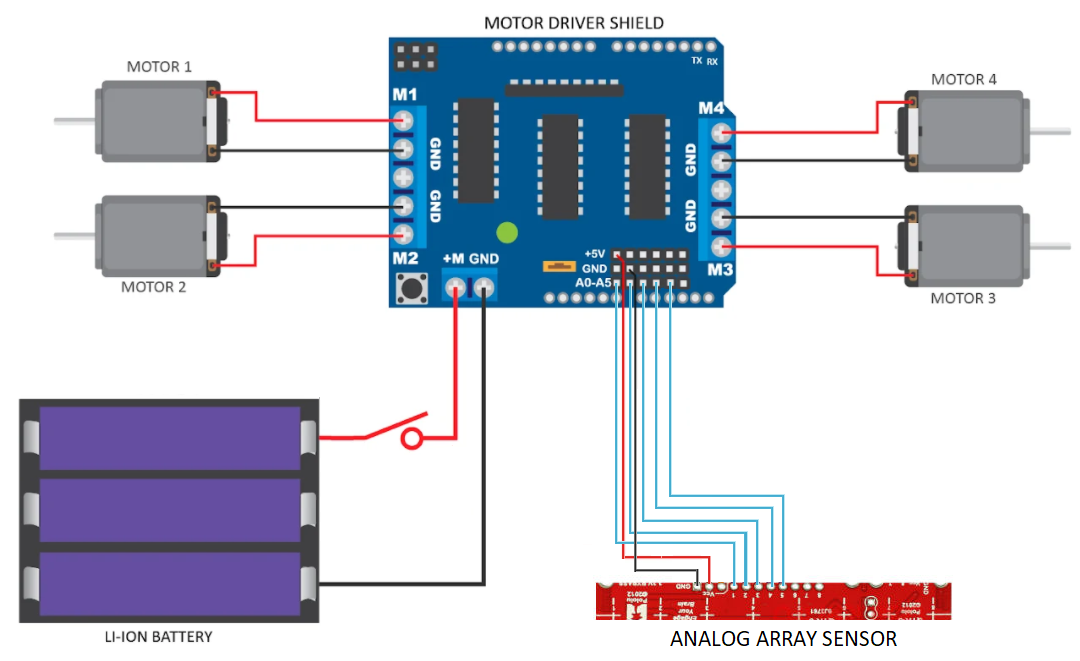
*Figure 1‑3: Front View*

SECTION 2

# ELECTROMECANICAL PARTS

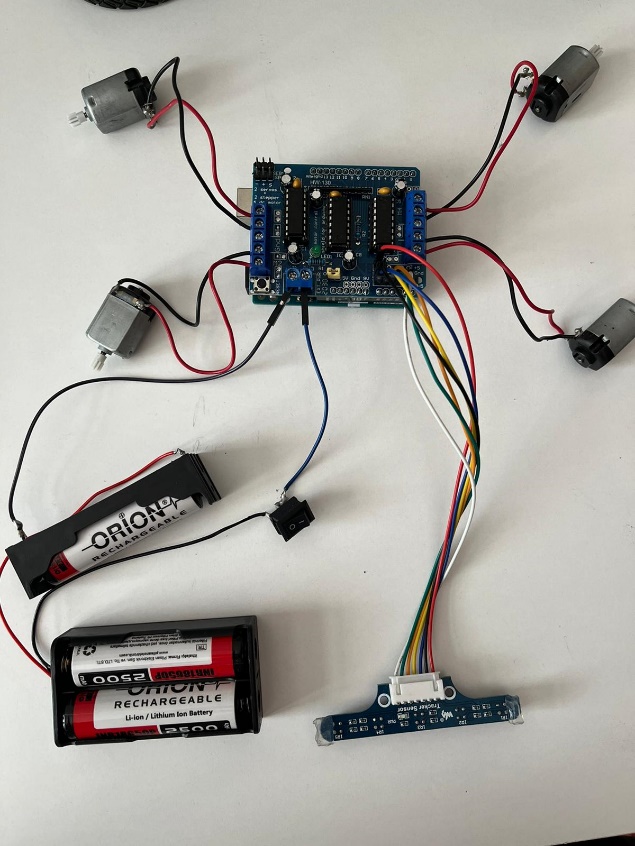
In this part of the project, a robot is designed to follow a specified line. The 4 wheels of this vehicle are designed to be supported by 4 engines and to carry the video camera on it..

## Circuit Diagram



*Figure 2‑1: Circuit Diagram*

*Figure 4-1 shows the circuit diagram used in the vehicle. While creating this circuit, 4 DC motors, 1 analog array of 5 sensors, 1 Motor Driver Shield compatible with Arduino Uno and 3 lithium ion batteries with 3.7 volts supply were used.*

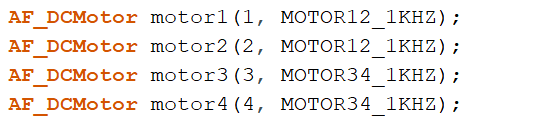
*The connection of the elements used with each other is shown in figure below.*

## Arduino Code

## In this embedded system, an Arduino-based software was used to create the motor operating principle according to the data from the sensors.

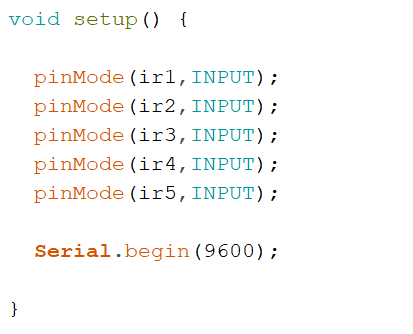
## 

*Figure 2‑4: Defining Sensors*

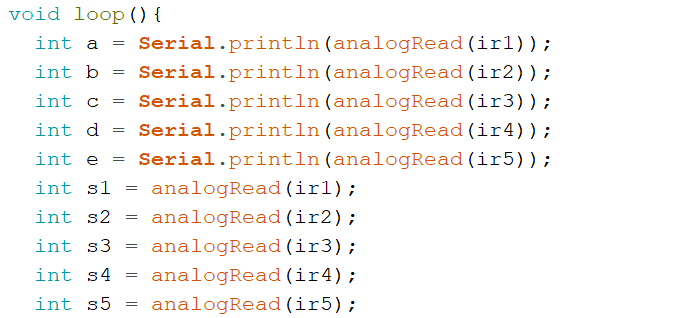
The AFMotor library is called for Arduino to recognize the Motor Driver used. After the mountain, each sensor variable connected to the pins was defined.

*Figure 4‑5: Defining Motors*

Parallel to these, each engine was also defined.

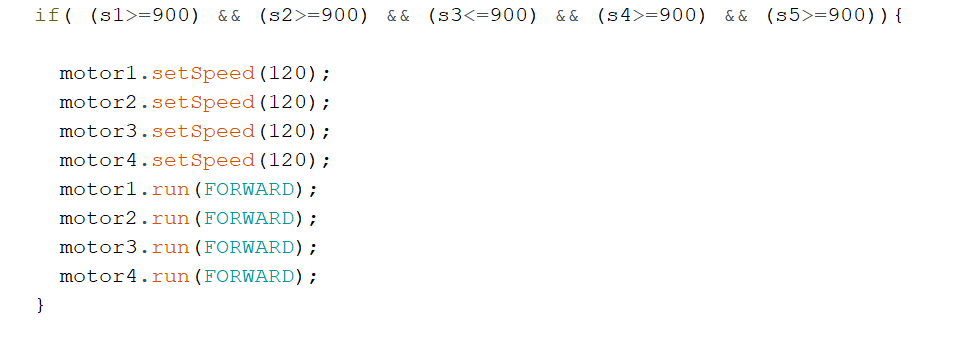


It was informed that the pins to which the variables defined for the sensors are connected are the input pins, and the serial communication command is written.



*Figure 2‑7: Variable assingments*

A continuous loop was started with the “void loop()” and the value of each pin was transferred to the variables a, b, c, d and e, respectively, in order to display the analog data coming from the sensor in this loop, but also the same values are s1, s2, s3, for later use. It is kept to the s4 and s5 variants.



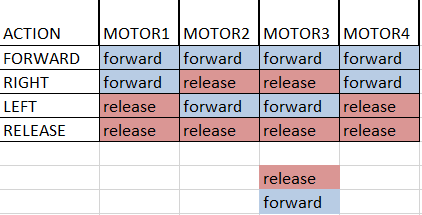
*Figure 2‑8: Sensor-Motor Conditions*

Again in the same cycle, there are motor movement codes designed in a certain draft. In this code structure, the comparison of the numerical value of the color of the line to be followed, which was previously obtained from analog values, according to certain situations, is described in the if block. The upper limit value, which is critical to follow for analog values, is 900. If all values except sensor3 read by Array sensor are above 900, it means that the black line in question is detected only by the 3rd Sensor and the rest of the sensors detect the white color. Therefore, the vehicle should move straight. In this case, the engine speeds were determined as 120 and the command to move forward was sent to all engines..

## Working Principle of Orientations

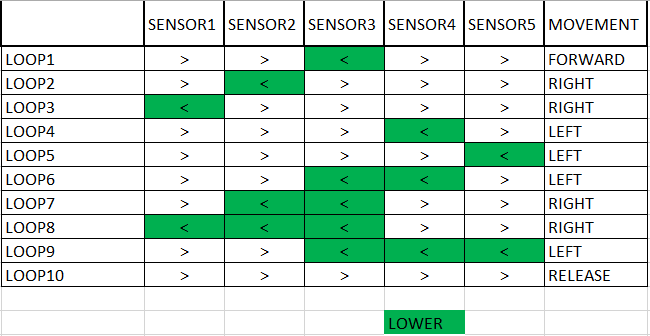
In order to ensure that the vehicle moves to the right or left or to move straight, it should be determined which engines will work according to the specified conditions. In the design of the vehicle's movement, the two wheels on the left stop to move to the left, while the two wheels on the right move. Parallel to this, the 2 wheels on the right wait and the 2 wheels on the left move.

Since the motors and wheels are directly connected to each other, the scenario of movement according to the conditions is given in Table 4-1.



*Table 2‑1: Motor-Orientation Correlation*

The information from the sensors also needs to be interpreted according to the conditions. At this point, the following logic has been followed. A black line (that is, a value less than 900) to be detected from the left while the vehicle is moving is the signal required to turn the vehicle to the left. Therefore, if at least one less than 900 signal is received in any of the left 3s on the Sensor, it is necessary to turn the vehicle to the left. Table 4-2 was created in line with this assumption.



*Table 2‑2: Orientation-Sensor Corelations of Loops*

According to the table, the relations of the sensors with each other are written in the if block in the code and the action that the vehicle should do in each case is specified in the if commands.

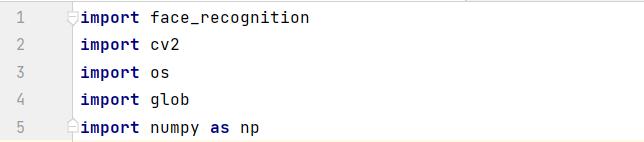
SECTION 3

# FACE RECOGNITION

Object Oriented Programming elements in Python were used to perform this operation. A Class structure was created and the necessary methods and objects were defined in this module. While creating the methods, some functions in the Face Recognition library were used in the methods for the necessary purposes.

## Class Module

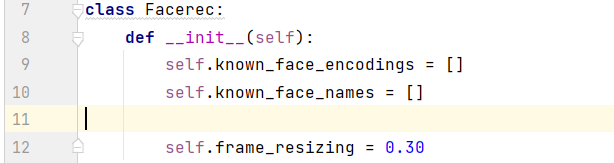
For this, the necessary libraries were called primarily in the Python Project.



*Figure 3‑1: Libraries*

Face\_recognition for face recognition, cv2 to use image processing elements, os to easily call image files from specified folders, globs to list image files in the specified folder, and numpy libraries to work with arrays more easily have been added.

In order to create the Class structure, a Class element and 2 Class Arrays were defined in the Class and its constructor structure.

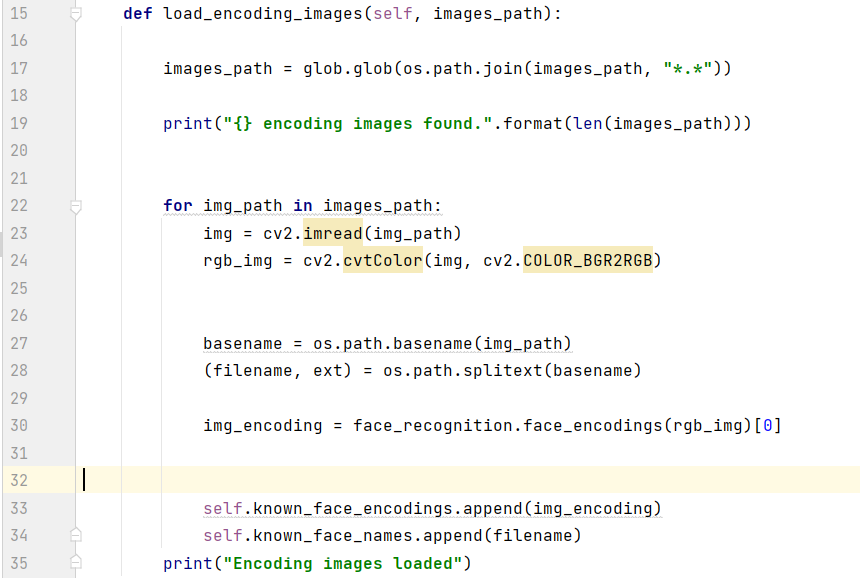


*Figure 3‑2: Defining Classes*

An array called known\_face\_encodings was needed to keep the coding information of the faces to be defined, and an array called known\_face\_names was needed to keep the identity information of the faces corresponding to these faces.

In addition, the phone camera on the vehicle records video at 60 frames per second. So that means 1.6ms per frame. Since the sampling time of the algorithms in the Python code cannot be this short, an external delay was added by interfering with the system. The variable of this delay is assigned as frame\_resizing.

### Identification of Images in folders by The System



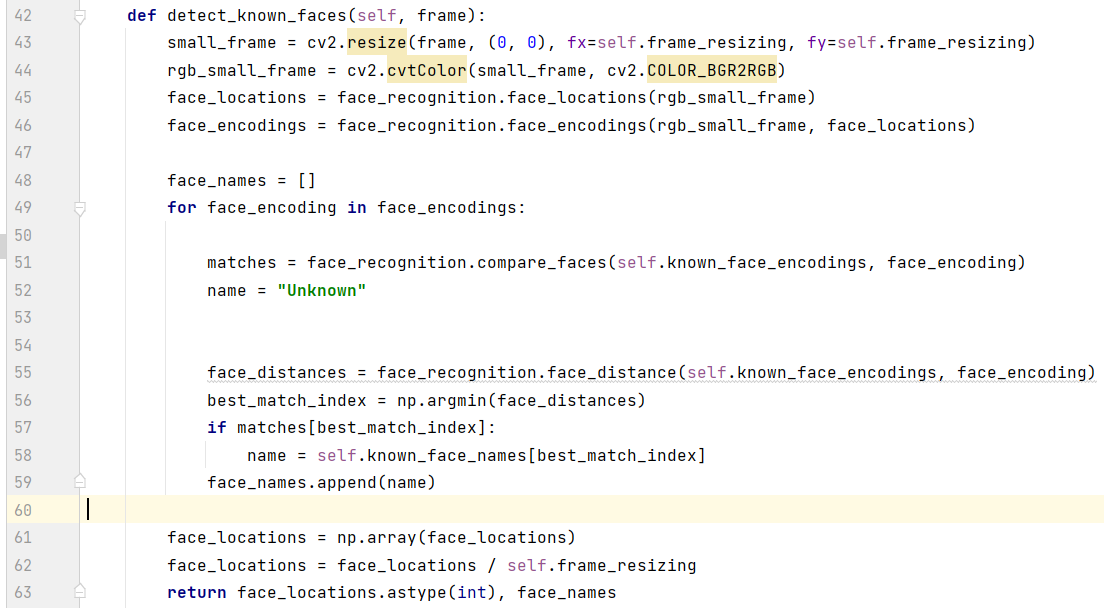
*Figure 2‑3: load\_encoding\_images Function*

A function named load\_encoding\_images has been defined in the Header File to be run directly in the main code. This function scans all images in a specified folder sequentially and stores the coding information of the faces in each image together with their IDs.

The path of the said folder in the host device is defined in the 17th line. Then, depending on the length of this variable, the terminal is informed that there are image files.

In the for loop, each image in the images\_path variable is processed sequentially. First, the image must be converted to RGB channels depending on the software structure of the face recognition library. After this process is done, the identity information of the image file in the process (it is assumed that the name of each file is recorded as identity information) is kept in the variable named basename. Then, using the face recognition library, the coding information of the face in each picture is obtained, and after the code, these coding information and identity information are recorded in the required Class elements. Finally, the information that this function has finished working is printed on the terminal and the load\_encoding\_images function is ready for use.

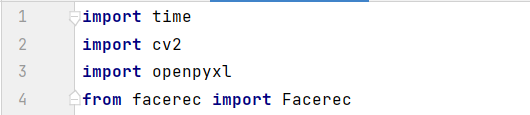
### Comparison of Face Images both Captured by Camera and Identified by System



*Figure 3‑4: detect\_known\_faces Function*

A function is defined above for the video camera to detect faces in each frame. In this function, the pixel numbers are reduced by reducing the size of the video camera frames that will enter the function in the 43rd line, so that the process can be performed in a shorter time. Then, the images from the video camera were converted to RGB channels and the coding information of the faces and their positions were assigned to various variables. Finally, within the for loop, the coding information of the faces in the video camera is compared, and the coding information of the faces stored in the system is compared, and the function software is completed so that if there is a match, the identity information of the matching face will be the output of the function.

## Python Program



*Figure 3‑5: Calling libraries*

Open CV library has been added. Python-Excel library has been added to save the identities of the faces to be detected in the excel file. Finally, the user-defined facerec module has been added.

*Figure 3‑6: Excel rows*

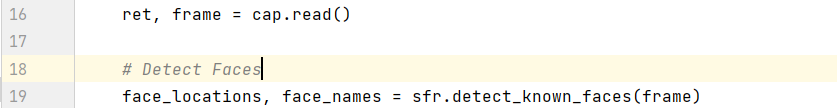
First the Excel file where the credentials will be printed, then the 1st page of this file is shown as the target.

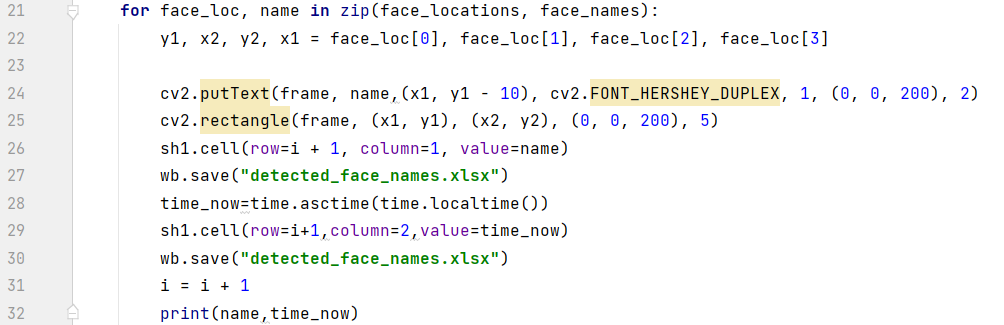
*Figure 3‑7: Defininf Face Images in the folder*

The variable named “sfr” is defined as a Class element and the coding information of the images containing the faces to be recognized in the system is assigned to this element.



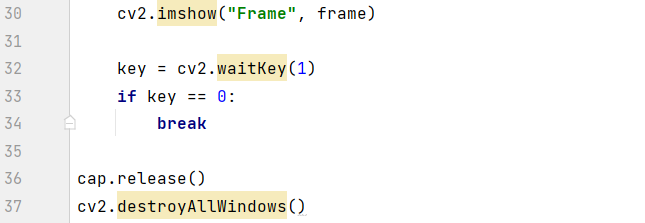
The frames recorded by the video camera are instantly encoded to be assigned to a variable via the VideoCapture function in the Open CV library. If the value of this function is changed to 0.1 or 2, other image sensor devices connected to the host device can also be switched.

The code fragments from here are in a while True loop.

The read function is used to use the frames in the cap object in image processing. The frame object, that is, the faces in the object where the camera's snapshots are kept, and the faces defined in the system are compared with the detect\_known\_faces function to be called by the sfr Class object. At this point, if the match is made, the identities and positions of the faces captured on the frame variable are assigned to the necessary variables. If the match does not occur, a value cannot be observed in the said variables.

*Figure 3‑8 Drawing squares on faces and saving informations of faces in Excel*

Using the face positions and identity information of the faces obtained in a for loop, a frame is drawn on the frame object, that is, on the images obtained in the camera, in a way that frames the faces, and the identity information of the face is printed on this frame.

After this process, the identity information of the detected face is instantly written to the Excel file to be recorded in the system. In line 25, the identity information is printed in the relevant cell, and then the target file is saved. The loop variable is incremented to print the credential on the bottom line at the next sampling time and the despit state is printed to the terminal.

*Figure 3‑9: End of the Program*

Finally, each instantaneously processed frame is displayed on the screen and the video camera image is made viewable in real time.

### Saving Face Images and names Captured by Camera

As can be seen in figure 1, openxl and time libraries have been added to perform this operation. The excel file shown as the target in figure 2 was selected, and in figure 6 it was specified which information should be written on which line. This is done as follows: If a face is detected by algorithms while the camera is recording, the identity information of this face is kept in the name variable. This credential is printed to the target excel file with the help of the openxl library functions. At each sampling time, the variable “i” defined for the while loop is incremented by one, which means go down one line. In addition, parallel to the printed ID information, the time information of that moment is recorded on the side line of the cell containing the ID information recorded at that moment in the loop, via the time library time functions. In this way, the information of which faces the video camera sees at which moment is recorded.

In addition, if a face that is not defined in the system is viewed by the video camera, the cell where the identity information of this face is recorded is painted in red for easy recognition. The recorded information is shown in Table 5-1.

*Tablo 5‑1: Record in Excel*