|  |
| --- |
| Authors: Lyudmila Grigoryeva, Ikhlas Jenfi, Roberta Tiriticco  12-8-2019 |

|  |
| --- |
| Turku University of Applied Sciences |
| Final Report |
| SECURITY SYSTEM |

Contents

[1. Introduction and Project Goal 1](#_Toc26730656)

[2. Project Outcome 1](#_Toc26730657)

[3. Hardware 1](#_Toc26730658)

[4. Project’s Success 2](#_Toc26730659)

[5. Future Improvement 2](#_Toc26730660)

# Introduction and Project Goal

Our project goal was to create a prototype of a security system designed to notify the owner if someone tries to access a secured space without permission. To ensure a good notification service, when the credential inserted are wrong, the system takes a picture, analyzes it and reports it to the owner through a private message.

# Project Outcome

The outcome of the project is a simple prototype that simulates a security system that notifies the owner of each attempted access.

The prototype consists of three main parts:

* **Hardware** for taking pictures and giving visual feedback with LEDs
* **Flask server** for login interface and executing commands
* **Image recognition** library
* **Connection to the user’s phone via a Telegram bot**

User is able to access a web page, insert the credentials and unlock some device. If the credentials provided are not correct, the device takes a picture of the user trying to unlock the device and after analyzing it, the program sends the photo through a private message to the owner and it will inform who or what tried to access the device.

# Hardware

To simulate the locking and unlocking of a security system, we decided to use visual feedback. If the access has been granted a green LED will turn on, otherwise, if the credential provided are wrong, a red LED will blink.

The hardware part was build using electronic components and coded using C.

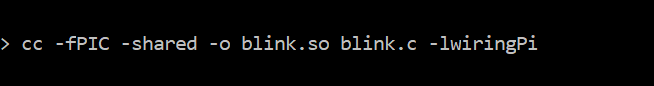
We chose this specific programming language since C allows programmers to gain easy access to the hardware by directly manipulating memory registers, I/O ports, interrupts and much more.

However, we decided to use Python, a different programming language, to program the other parts of the project. Therefore, we had to find a way to connect the Python script with the C script.

Therefore, the first step is to write the function in C. In this case the code has been saved in a file named *blink.c*.

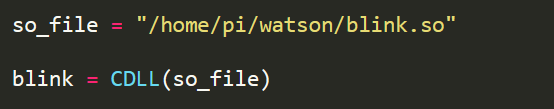


Secondly, we use the following program to generate shared library of the file using the C compiler.  
Be sure to type also the command to include the WiringPi library.

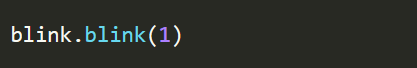


We import the *ctypes* (foreign function interface) library to call C function from the Python script.

Then we specify the location of the blink.so file as follow.



Finally, we are able to call the C function from the Python script passing the parameter accordingly to the test case.

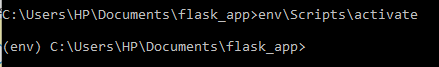


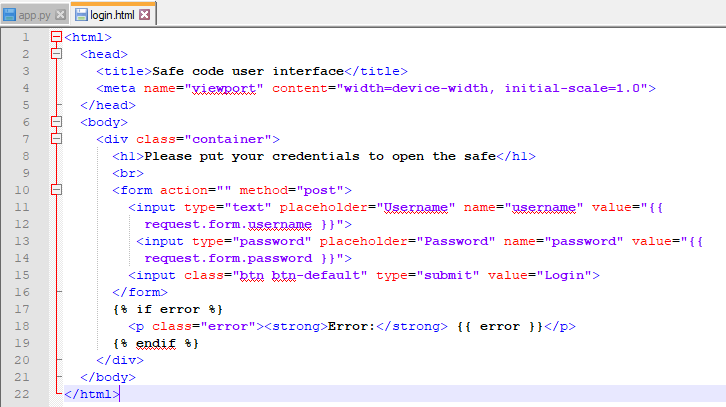
# Server

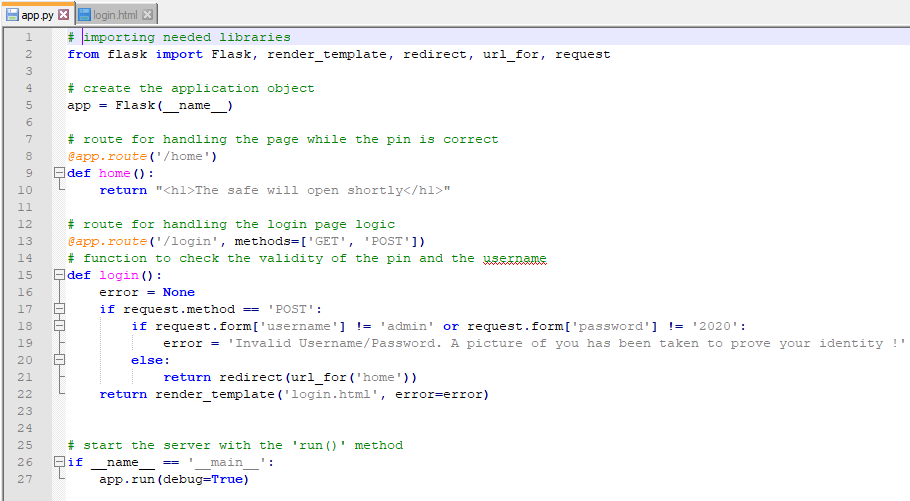
To start with our project, setting up Flask is the first thing to do. Flask was chosen due to being lightweight, easy to setup and for using Python.

This Flask app checks if the credentials provided are correct or not. HTML was used to write a simple web interface.

To start with downloading Flask, we needed first to create a folder for it then accessing that folder in the terminal. Once in the main folder of the flask app:

* To install Flask virtual environment:  following by a name for its folder.
* To activate to environment:  . Once “enter” is pressed, this is how it looks while it’s activated: 
* Now, it’s time to install Flask: 
* Our Flask is set now. After that the HTML and python code are followed:

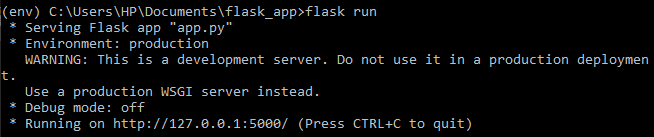




* After that, we needed to set the main app file “app.py” in the virtual environment:



* To run and test our code: 
* If all correct it won’t throw error, but it will give an address where your app is running:



* To stop the running app: Ctrl + C shall be pressed.
* To deactivate the virtual environment: 

# Image Recognition

Once the Flask server is up and running, the next goal of the project was to implement image recognition into the system. Since writing an image recognition algorithm from scratch lies beyond the project scope, it was decided to use an existing image recognition library.

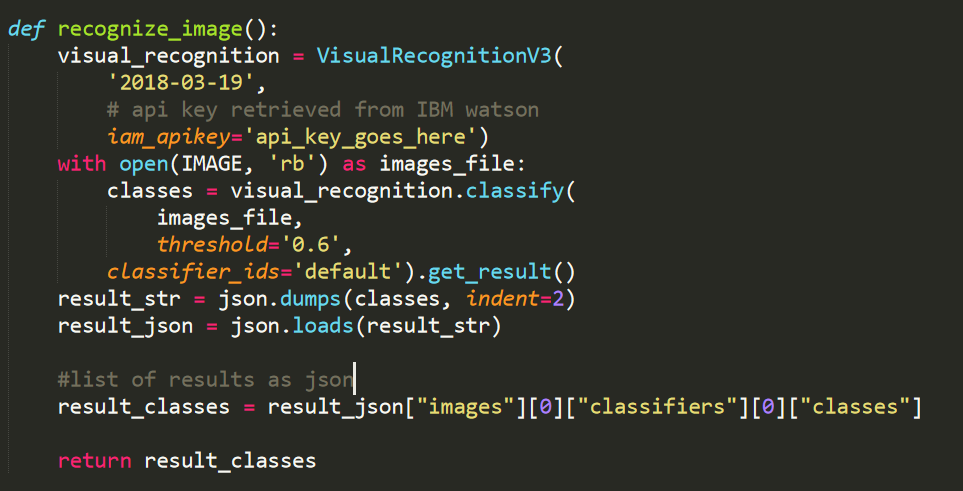
IBM Watson is a collection of AI services and tools provided by IBM. With Watson you can implement visual recognition, natural language classification and machine learning. Besides training models, Watson visual recognition service also provides built-in models for quick image analysis without any training. To use those models we will need to:

* Create an IBM Watson account (free tier available)
* Set up a Visual Recognition service in IBM Watson Studio
* Build a classifier

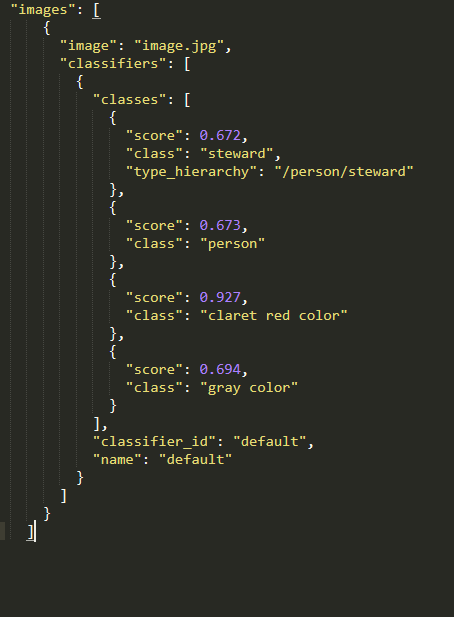
Watson has 4 different built-in models for different image recognition: the General model for default classification, the Explicit model for detecting image inappropriate for general use, the Food model for food images, and the Text model for extracting text from images. We will use the General model in this project since we do not need to build a custom model.

In IBM Watson studio you can also test images with the built-in models without writing any code, which is quite handy. But for implementing the classifier into our project we will use Python.

Using the built-in libraries is very straightforward if one can handle basic operations in Python.

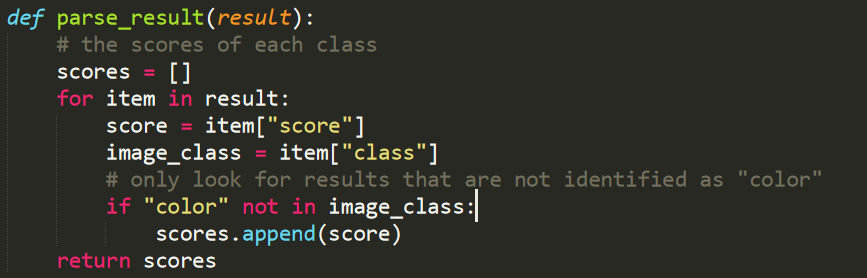


This is the result that we get:

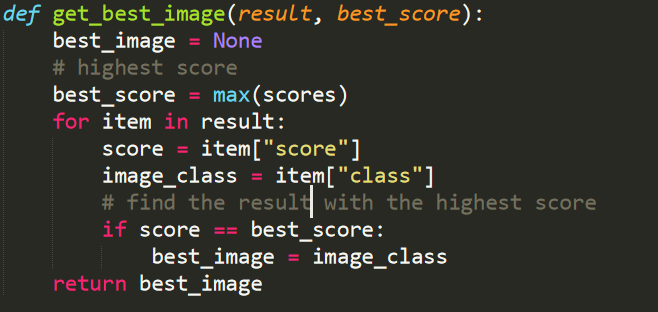


We need to parse the result since we are not interested in the colors that the image classifier identifies. To do that we just ignore any “class” result that equals to “color”.

We also need the scores of each class in order to compare them and find the highest score. It will point out to the most accurate result.



Now we just have to find the highest score in the *scores* list. We can do that by using a built-in Python function *max()*.



The result *best\_image* is a string that is passed on to IFFFT/Telegram.

# Connection to the phone

Finally, we used Telegram to send the pictures to the owner of the safe device. Telegram supports bots, so it was an easy choice (compared to Whatsapp, for example).

All we needed to do is import the telepot library (Python library for Telegram integrations) and use BotFather (Telegram’s Bot account for creating your own bot and accessing its API key).



The chat\_id value is received by the bot when user sends a specific message. In our case we had to send a message every time we wanted to receive the photo, which is not the best option in a real-life environment. Perhaps, there should be a further study of how Telegram manages this issue.