**MS-Project**

**Valentin Lumezeanu**

**Comiati Ovidiu**

**Hardware components**

**-Raspberry Pi 3**

**-jumper cables**

**-5V relay**

**-digital temperature sensor**

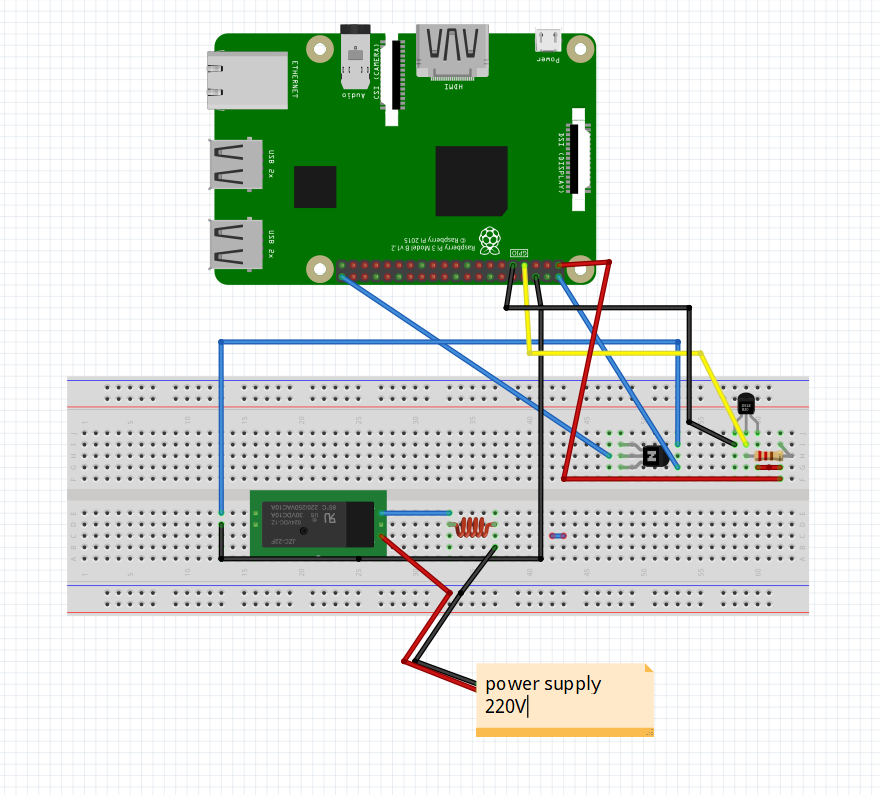
**-power supply**

**-transistor**

**-4.7k ohm resistor**

**-heater**

The objects mentioned above have been combined according to the diagram below…

****

**Pi Server**

The programming language used for manipulating the low level interaction with the raspberry pi is python. For achieving our projects goal we made use of a number of python packages and frameworks (ex:

Flask, RPi.GPIO, multiprocessing,etc)

**Flask**

**Flask** is a micro [web framework](https://en.wikipedia.org/wiki/Web_framework) written in [Python](https://en.wikipedia.org/wiki/Python_(programming_language)). It is classified as a [microframework](https://en.wikipedia.org/wiki/Microframework" \o "Microframework) because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself.

Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools. Extensions are updated far more regularly than the core Flask program.

Applications that use the Flask framework include [Pinterest](https://en.wikipedia.org/wiki/Pinterest" \o "Pinterest), [LinkedIn](https://en.wikipedia.org/wiki/LinkedIn),  and the community web page for Flask itself.

Page: <http://flask.pocoo.org/docs/1.0/>

Installing :

Flask can be easily installed using the command ( considering you have pip installed )

” $ sudo pip install Flask”

**RPi.GPIO**

A powerful feature of the Raspberry Pi is the row of GPIO (general-purpose input/output) pins along the top edge of the board. A 40-pin GPIO header is found on all current Raspberry Pi boards (unpopulated on Pi Zero and Pi Zero W).

This package provides a class to control the GPIO on a Raspberry Pi.

Page: <https://pypi.org/project/RPi.GPIO/>

Installing:

”$ sudo apt-get update”

“$ sudo apt-get install rpi.gpio”

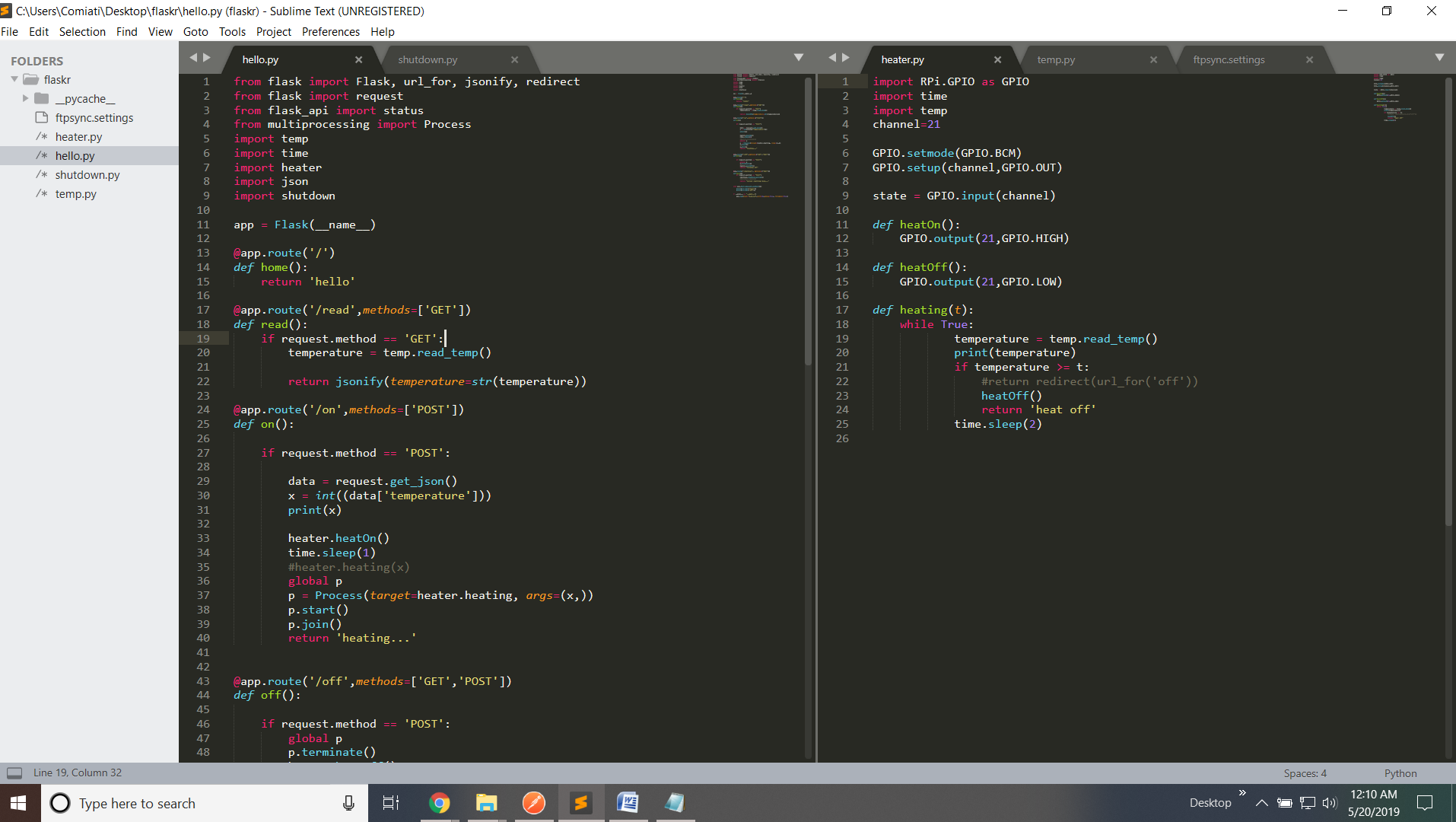
**Multiprocessing**

[**Multiprocessing**](https://docs.python.org/2/library/multiprocessing.html#module-multiprocessing) is a package that supports spawning processes using an API similar to the [**threading**](https://docs.python.org/2/library/threading.html#module-threading) module. The [**multiprocessing**](https://docs.python.org/2/library/multiprocessing.html#module-multiprocessing) package offers both local and remote concurrency , effectively side-stepping the [Global Interpreter Lock](https://docs.python.org/2/glossary.html#term-global-interpreter-lock) by using subprocesses instead of threads. Due to this, the [**multiprocessing**](https://docs.python.org/2/library/multiprocessing.html#module-multiprocessing) module allows the programmer to fully leverage multiple processors on a given machine.

The [**multiprocessing**](https://docs.python.org/2/library/multiprocessing.html#module-multiprocessing) module also introduces APIs which do not have analogs in the [**threading**](https://docs.python.org/2/library/threading.html#module-threading) module. A prime example of this is the **Pool** object which offers a convenient means of parallelizing the execution of a function across multiple input values, distributing the input data across processes (data parallelism).

Page: <https://docs.python.org/2/library/multiprocessing.html>

Installing:

Shipped with python, no need to install

Web Server

.NET Core was the language that we opted for in trying to create a web server that would make it easier to interact with the web UI.

.NET Core supports four [cross-platform](https://en.wikipedia.org/wiki/Cross-platform) scenarios: [ASP.NET Core](https://en.wikipedia.org/wiki/ASP.NET_Core) [web apps](https://en.wikipedia.org/wiki/Web_app), command-line apps, libraries, and [Universal Windows Platform apps](https://en.wikipedia.org/wiki/Universal_Windows_Platform_apps).

Of those we used ASP.NET Core. **ASP.NET Core** is a [free and open-source](https://en.wikipedia.org/wiki/Free_and_open-source) [web framework](https://en.wikipedia.org/wiki/Web_framework), and higher performance than [ASP.NET](https://en.wikipedia.org/wiki/ASP.NET), developed by [Microsoft](https://en.wikipedia.org/wiki/Microsoft) and the community.

**ASP.NET Core** applications supports side by side versioning in which different applications, running on the same machine, can target different versions of ASP.NET Core. This is not possible with previous versions of ASP.NET.

The most important component of the .NET part of the project is the Controller class object usage.

Controllers are responsible for responding to requests made against an ASP.NET MVC website. Each browser request is mapped to a particular controller. For example, imagine that you enter the following URL into the address bar of your browser:

http://localhost/Product/Index/3

In this case, a controller named ProductController is invoked. The ProductController is responsible for generating the response to the browser request. For example, the controller might return a particular view back to the browser or the controller might redirect the user to another controller.

User Interface

The user interface is done with Angular.

