For Attention of Examiner

Outline of notable files and their role within the artefact

"./Embedded Software/"

The two subdirectories contained within pertain to the software present on the embedded system.

"Basic Requirements.ino" is the embedded software which conforms to the basic requirements of the brief. "Advanced Requirements.ino" is the embedded software which conforms to the Advanced requirements of the brief.

The ".ino" files are stored in plain text and can be accessed using a text editor for review by the examiner. They are written in the Arduino Programming language, which has a similar syntax to c++.

These source files are commented throughout, and the functionality of different parts are explained in the coursework report.

For more information, see https://www.arduino.cc/

"./ClisTeach/"

This is the root directory of a Laravel application, which serves the website for the fictional home automation company.
For simplicity, file paths from this point forward will refer to "./Clisteach/" as the root ("/")

For dependencies, see /composer.lock and /package.json

Routing

When a user sends a HTTP GET request, it is managed by /routes/web.php.
In this file you can see routes being defined along with the respective actions taken by them.
Routes in this file only pertain to web users, as opposed to possible API requests, etc.

For instance, suppose a user were to navigate to the about page by searching "http://Fake_Clisteach_Domain.com/about":

The route is defined on lines 26-28 of /routes/web.php:

```
 Route::get("/about", function () {
        return view("about");
})->name("about");
```

Here we can see "/about" is its location. (1st parameter of get())

For simple pages, a callback function returns a "view" to the visitor. This is the case for /about.
Therefore, the second parameter in the route definition is a simple callback function with no inputs/parameters.
It merely returns view("about").

In this example, the "about" view is stored in /resources/views/about.blade.php

The ->name() part is used to refer to this route elsewhere but isn't relevant to this explanation.

To build a response with the appropriate html page, this view file is run through Blade, and PHP itself.
More on this can be found under the "Views" heading.

Controllers

Some routes call methods on controllers. These can be found at /app/Http/Controllers.
For instance,

```
[ProductController::class, "show"]
```

in the products route definition calls the show() method on the ProductController.
This is defined in /app/Http/Controllers/ProductController.php along with the other controllers.

Controllers are used for database exchanges, input validation, and passing data to views among a plethora of other uses.

At the very least, they decompose response functionality out into specific files as opposed to everything being handled by callback functions in /routes/web.php.

Views

These files are returned to the client by either route callback functions or controllers.

Blade, Laravel's templating engine, along with PHP itself process these files to construct a html page, which is sent in the response.

Blade can be identified in these files by the "@keyword" syntax.

This also allows components which can be inserted into other files using the "@include()" statement. This is how the navbar is inserted into every page.

For more information, see https://laravel.com/

"switch.py"

This python script allows the embedded system to be controlled by the website.

This functionality is outlined under the "Advanced Requirements" heading of the coursework report.

The file is executed by SwitchController, using the built-in php function "shell_exec()"

This execution includes an argument which states whether to turn the switch on ("1") or off ("0").
Shell arguments are accessed by python using sys.argv

Depending on what the argument is, the message ("on\n" or "off\n") is written to the device over a serial connection (device is at COM3)

Print statement at the end is necessary for debugging.