

Chapter 1

Technologies

1.1 WordPress

WordPress is one of the famous Content Management System (CMS) that is powering almost 25% of the websites globally. It is an Open Source project created, developed and maintained by the community. Hence, it can be downloaded, used, modified and distributed freely without any license fees.

The WordPress can be downloaded from their official website, wordpress.org. The current stable version of WordPress is 4.7, as at time of writing. Initially, it started as blogging system and became a full content management system with thousands of plugins, widgets and themes.

To run WordPress, the server should be equipped with PHP version 7 or greater, MySQL database server version 5.6 or greater or MariaDB version 10.0 or greater, and HTTPs support. As for web server, it is recommended to use either Apache or Nginx. They are reliable, have a lot of features, and works well with PHP and MySQL. For Smart Home Lab website, WordPress will be ran using Nginx web server with PHP and MySQL. The Installation and configuration of them before installing WordPress will be discussed in the next chapter.

With WordPress, various website can be developed right from simple blogs to major websites comprising custom made plugins and widgets. This is one of the advantage of using WordPress is handling the flexibility and complexity while being easy to use.

Using WordPress enables the web developer to concentrate on the contents and structuring the websites, rather than developing the backend and frontend from scratch, which is time consuming and increased time to market. Other than that, WordPress also provides multilingual pack consisting of 70 languages. This enable the website to developed in multiple languages,

primarily in English and German languages. This not only changes the contents of the website as viewed by the web user, but also the administrator dashboard.

WordPress also provides excellent security for its users. Their security implementation consists of protection against Injection, Broken Authentication, Cross Site Scripting (XSS), Insecure Direct Object Reference, Security Misconfiguration etc. Their themes, plugins as well as hosting providers are tested and secured.

Using WordPress also make it easy for others to develop and continue the development of the website in the future, as everything can be managed in an easy web interface. And through this documentation, most of the essential information regarding the installation, configuration and contents development will be provided.

1.2 PHP

PHP stands for Hypertext Preprocessor. It is a general-purpose scripting language for generating dynamic web contents. It is commonly used for web development and can be embedded into HTML. The PHP scripts are executed on the server-side and the clients are unaware of the script and its execution. The result of the executed PHP will be sent to client in HTML format.

PHP scripts are processed by PHP parser or PHP interpreter. This parser will be installed in the web server such as Apache, IIC, lighttpd or Nginx as a Common Gateway Interface (CGI). The PHP parser is open source software, which can be downloaded from their official website¹.

PHP is not only limited to deliver web contents like HTML, but can be used to output text, images, PDFs and flash-videos. One of the PHP advantages is it supports and works well with various databases. For this website, the database called MySQL is used and will be discussed in the next section.

1.3 MySQL

MySQL is relational database management system (RDBMS). The Community Edition of MySQL is open source database system which can be downloaded from the official website². MySQL uses structured query language

¹<http://php.net>

²<https://dev.mysql.com/downloads/>

(SQL) to query, filter, add, modify and delete data in the database. The data in the database will be arranged in table form. MySQL is used in majority of the websites and content management system such as WordPress.

1.4 Nginx

Nginx is a high-performance web server. It is also act as reverse proxy, IMAP/POP3 proxy server, load balancer and accelerator. It works with HTTP-, TCP- as well as UDP-based services. The main advantages of Nginx are its high performance, stability, rich feature set, simple configuration and low resource consumption.

The architecture of Nginx uses scalable event-drive asynchronous architecture. Not only it uses small amount of memory, but the memory usage is predictable under load. It success relies in the ability to handle thousands of requests at the same time. For this project, Nginx web server will be used, where it communicates with PHP parser to run WordPress and MySQL database.

1.4.1 WordPress Installation

After installing Nginx, PHP and MySQL, we can begin with WordPress installation. Please make sure to log in into server-side using PuTTY. Before we can install the WordPress, there are few settings / configurations that have to be done, which includes:

- Setting up database table and database user
- Configuring Nginx server
- Installing additional PHP modules to handle WordPress
- Generating secret keys
- Downloading latest stable version of WordPress
- Configuring WordPress

Step 1: Creating MySQL Database and User

First, we need to prepare a database table and user in the installed MySQL. The database table and user will then be used by WordPress to store data and access them.

Start MySQL database by issuing the following command:

```
| $ mysql -u root -p
```

Enter the password `d_XKdEoCwX,4En0` for the user `root` when prompted.

Create a new database table **wordpress** by issuing following command. And don't forget the semicolon at the end of the line.

```
| mysql > CREATE DATABASE wordpress DEFAULT CHARACTER SET utf8  
| COLLATE utf8_unicode_ci;
```

Next, create a new user to operate the above created **wordpress** database. The following command will create a new user **wordpressuser** with password **ABcd1234#**.

```
| mysql > GRANT ALL ON wordpress.* TO 'wordpressuser'@'localhost '  
| IDENTIFIED BY 'ABcd1234#';
```

This will create the new user with given password, as well as granting the user the all accesses to the database **wordpress**. Finally, flush the privileges so that the MySQL aware of the changes we have made and exit MySQL.

```
| mysql > FLUSH PRIVILEGES;  
| mysql > EXIT;
```

Step 2: Adjust Nginx's Configuration

There are two modifications that have to be made to the Nginx web server to handle WordPress correctly:

1. Instruct server not to log requests for the static files with extensions such as .css, .gif etc.
2. Insert `index.php` into `try_files`

Open the Nginx configuration file using `sudo` privileges:

```
|$ sudo nano /etc/nginx/sites-available/default
```

We can request server not to log requests to the static files by issuing command `log_not_found off`. It is a best practice not to log requests to static files.

```
|server {
|    ...
|    location = /favicon.ico {
|        log_not_found off;
|        access_log off;
|    }
|    location = /robot.txt {
|        log_not_found off;
|        access_log off;
|        allow all;
|    }
|    location ~* \.(css|gif|ico|jpeg|jpg|js|png)$ {
|        expires max;
|        log_not_found off;
|    }
|    ...
|}
```

For the second task, passing `index.php` as `try_files` is done so that the server will return home page instead of 404 error as a default option. To do this, following line has to put inside `location /` block.

```
|server {
|    ...
|    location / {
|        #try_files $uri $uri/ =404;
|        try_files $uri $uri/ /index.php$is_args$args;
|    }
|    ...
|}
```

When these changes have been made, save the configuration file and exit **nano** editor by pressing **Ctrl-x**. Check for the syntax errors of the new configurations. If no error is been reported, restart the Nginx.

```
$ sudo nginx -t
$ sudo systemctl reload nginx
```

Step 3: Installing additional PHP extensions

PHP preprocessor has been installed and setup with minimal setting in the previous section. Here, additional modules or extensions will be installed, which is required by WordPress. Use to following command to install the required additional extensions:

```
$ sudo apt-get update
$ sudo apt-get install php-curl php-gd php-mbstring php-mcrypt
  php-xml php-xmlrpc
```

After finish installing the extensions, restart the **php-fpm** process.

```
$ sudo systemctl restart php7.0-fpm
```

Step 4: Generating secret keys

Secret keys have to be generated to provide security for the WordPress. The secret keys are provided by WordPress through their secure key generator. To get secret keys from WordPress key generator, issue the following command on the terminal:

```
$ curl -s https://api.wordpress.org/secret-key/1.1/salt
```

Or another workaround is to use the web browser. Enter the URL <https://api.wordpress.org/secret-key/1.1/salt> and press enter. Save the generated secret keys, which must be used in the next step.

Step 5: Downloading WordPress

Download the latest version of stable WordPress from <https://wordpress.org> website. After downloading, extract the downloaded content.

```
$ cd /tmp
$ curl -o https://wordpress.org/latest.tar.gz
$ tar xzvf latest.tar.gz
```

By default, WordPress has provided a sample copy of the WordPress configuration file. This file can be found under directory and file name `tmp/wordpress/wp-config-sample.php`.

Copy the provided configuration file to used in our website. And Create a new directory **upgrade**, so that WordPress can upgrade its software in the future without any permission issue.

```
$ cp /tmp/wordpress/wp-config-sample.php
   /tmp/wordpress/wp-config.php
$ mkdir /tmp/wordpress/wp-content/upgrade
```

Finally, copy the contents into root server directory. This are the contents used by the Nginx web server to serve when request is made to the website.

```
$ sudo cp -a /tmp/wordpress/. /var/www/html
```

Step 6: Configuring WordPress

Open the WordPress configuration file:

```
$ nano /var/www/html/wp-config.php
```

In the file, find the 'secret key' section, which appears as follow and paste the generated secret keys from Step 4.

```
...
define('AUTH_KEY', 'put your unique phrase here');
define('SECURE_AUTH_KEY', 'put your unique phrase here');
define('LOGGED_IN_KEY', 'put your unique phrase here');
define('NONCE_KEY', 'put your unique phrase here');
define('AUTH_SALT', 'put your unique phrase here');
define('SECURE_AUTH_SALT', 'put your unique phrase here');
define('LOGGED_IN_SALT', 'put your unique phrase here');
define('NONCE_SALT', 'put your unique phrase here');
...
```

After pasting the generated secret keys, the MySQL database credentials have to be given. These credentials have been created in the Step 1.

```
...
define('DB_NAME', 'wordpress');

/** MySQL database username */
define('DB_USER', 'wordpressuser');

/** MySQL database password */
define('DB_PASSWORD', 'ABcd1234#');
```

Lastly, we need to give permission to Nginx web server to write where it needs to. Otherwise, WordPress will be asking FTP credentials when any actions need to be performed. This can be done through following setting:

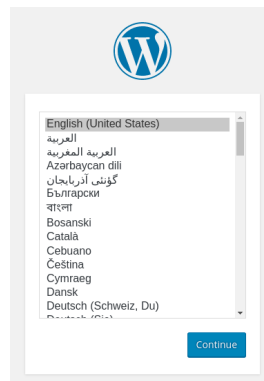
```
| define( 'FS_METHOD', 'direct' );
```

Save the `wp-config.php` file and exit the nano editor.

Step 7: Complete WordPress Installation

We need to complete the WordPress installation now through the web browser. Open the web browser and enter the URL `192.168.0.20`. First, we need to select the language.

Figure 1.1: WordPress language selection



After that, we need to provide a few information on the main setup page such as *Site Title*, *Username*, *Password* and *Email*.

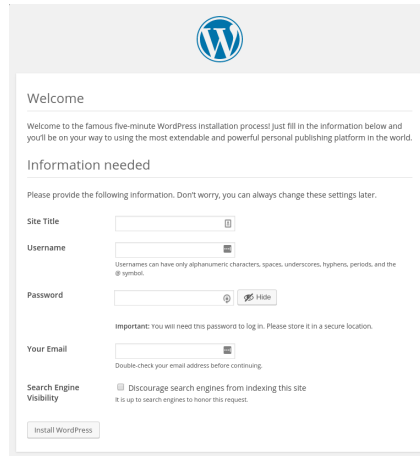
These are information given on the page shown above:

- Site Title: Smart Home Lab
- Username: ashiqmoh
- Password: oSm0cCCAguIMsalrZd(#2i5(
- Email: ashiqmoh@192.168.0.20

The installation can be completed by clicking 'Install WordPress' button. Upon installation, the administration side of the website can be accessed by entering following URL:

```
| http://192.168.0.20/wp-admin
- or -
| http://web.smarthome.hs-furtwangen.de/wp-admin
```


Figure 1.2: Main WordPress setup page



The image shows the main WordPress setup page. At the top is the WordPress logo. Below it is a 'Welcome' section with a brief introduction. The 'Information needed' section contains several input fields: 'Site Title', 'Username', 'Password' (with a 'Hide' button), and 'Your Email'. There are also checkboxes for 'Search Engine Visibility' and an 'Install WordPress' button at the bottom.

Welcome

Welcome to the famous five-minute WordPress installation process! Just fill in the information below and you'll be on your way to using the most extendable and powerful personal publishing platform in the world.

Information needed

Please provide the following information. Don't worry, you can always change these settings later.

Site Title

Username

Username can have only alphanumeric characters, spaces, underscores, hyphens, periods, and the @ symbol.

Password [Hide](#)

Important: You will need this password to log in. Please store it in a secure location.

Your Email

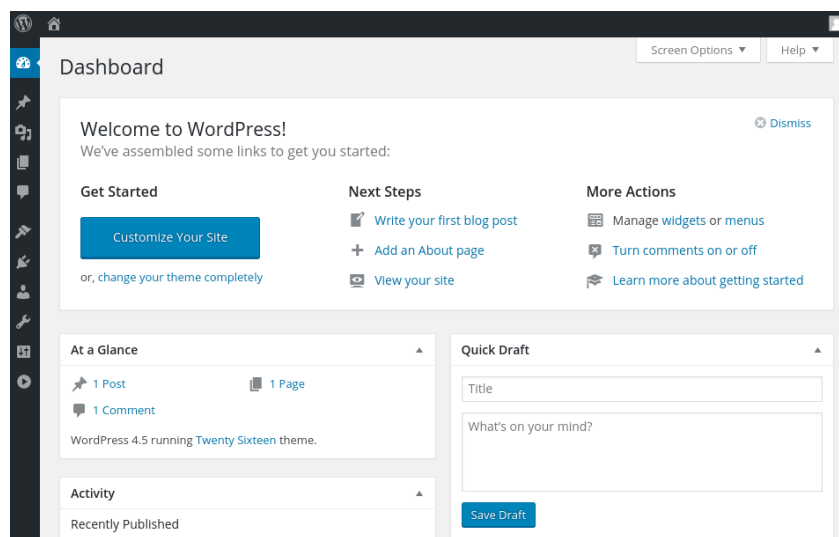
Double-check your email address before continuing.

Search Engine Visibility ☐ Discourage search engines from indexing this site
It is up to search engines to honor this request.

[Install WordPress](#)

Provide the username and password stated above and login. The administration side of the website appears as shown in the figure below.

Figure 1.3: Administration page of WordPress



Chapter 2

CCTV Live Stream

2.1 Introduction

Another task in this thesis work is to stream live CCTV footage on the website. The CCTV used in thesis work is called PremiumBlue IP Camera (*see* Figure 2.1). This camera is built with 720p resolutions, can be rotated and tilted remotely and monitored wirelessly. It is suitable to be used in homes, offices and labs. The user interface of the camera, which is used to view the footage and control the camera itself, can be accessed through the web browser. Additionally, it has a SD-card slot, which enables the storing of the footages and replay of them in the later time.

Figure 2.1: PremiumBlue IP-Camera



This chapter will discuss on how to configure the above mentioned IP-Camera; extracting the information to get the camera footage and controlling the camera; and designing the web page in the WordPress.

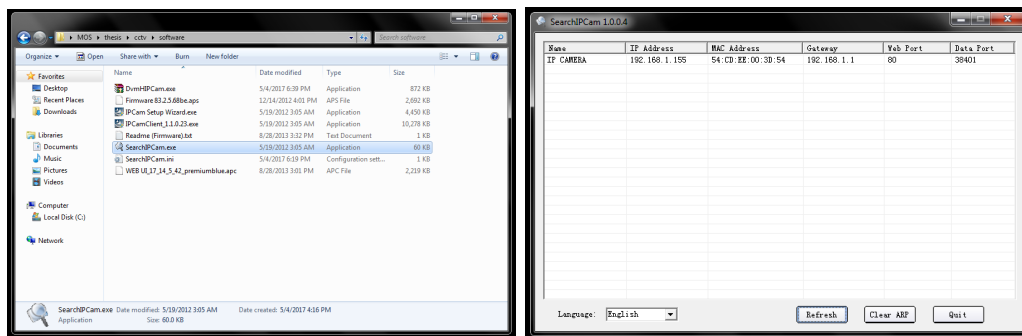
2.2 Pre-Configuration

2.2.1 Identifying IP Address

The steps discussed here is part of configuration steps from the camera vendor, which can be found in the following documentation¹.

The camera came with built-in web application and a bundle of softwares. To get started, the IP-Camera need to be connected to power supply and LAN-network. Next, we need to identify the IP of the camera by using the software came bundled with the camera called *SearchIPCam.exe* (see Figure 2.2). This application can also be downloaded from the vendor's website². Run the program and hit the 'Refresh' button to obtain the IP address of the camera.

Figure 2.2: SearchIPCam.exe



2.2.2 Login into Web Interface

After identifying the IP address of the camera, the web interface of the camera can be accessed by giving in the identified IP address (e.g. 192.168.0.157) from previous step in the web browser's address bar. Login by using the following credentials.

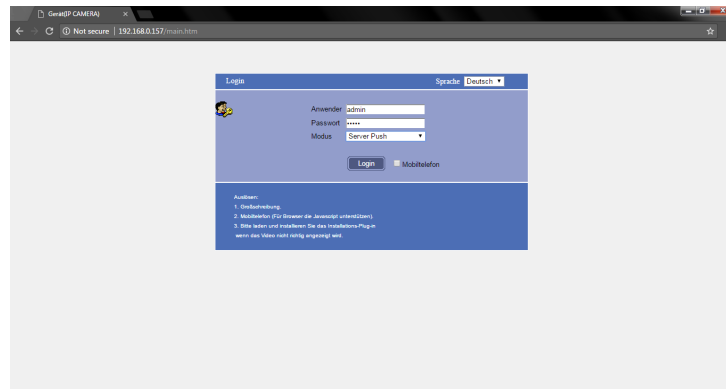
- Anwender: admin
- Passwort: admin
- Modus: Server Push

¹<https://www.pollin.de/shop/downloads/D722622B.PDF>

²<https://www.pollin.de/shop/downloads/D722622S.ZIP>

(Note: 'Anwender' and 'Passwort' were not changed as at time of writing this. It may be changed in the future.)

Figure 2.3: Login in into IP-Camera web interface



2.2.3 WLAN Setup

After successful login, the settings of the IP Camera can be accessed by clicking the tool button 'Einstellung' on top right of the page. In the menu list on the left panel, click on the 'WLAN'. In there search for the available WLAN network by clicking the 'Suchen' button. After the program finish the search, choose the lab network identified by SSID name 'SHLAB01' or 'SHLAB02'. Enter the following details:

- WLAN wird verwendet: tick
- SSID: 'SHLAB01' or 'SHLAB02'
- Netzwerk Type: Infra
- Sicherer Modus: AEP
- Verschlüsselung: WPA2-PSK
- Key: 91054319

Hit the 'Update' and 'Speichern' buttons and close the web browser or the tab. (Note: The SSID and password may be changed in the future time.)

Figure 2.4: IP-Camera WLAN settings

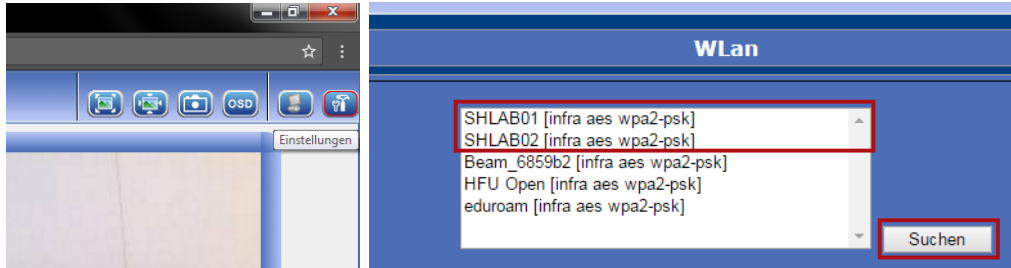
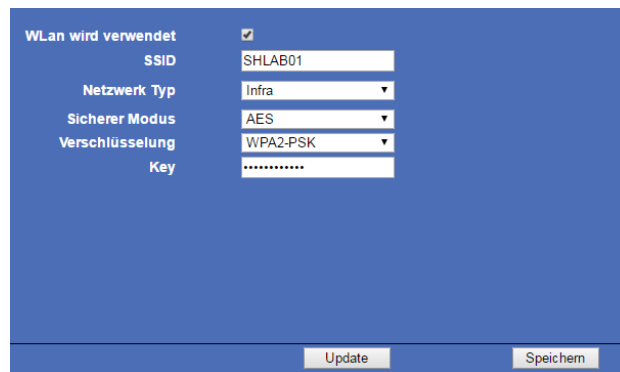


Figure 2.5: WLAN Credentials



2.2.4 Restart

Now, the IP-Camera has to be restarted without connecting it to network via LAN cable. Since the wireless network credentials have been entered in the previous step, it will connect to lab network, 'SHLAB02' automatically upon start-up and will have a different IP address. Run the *SearchIPCam.exe* program again as discussed in the Section 2.2.1 and login into the web interface using the new IP address. The IP address will be renewed when the IP-Camera is connected through wireless LAN.

2.3 Identifying the cgi-bin Module

2.3.1 Introduction

In order to stream the live footage or recording of the IP-Camera in the Smart Home Lab website, the URI address and additional query parameters are required in order to access the IP-Camera recording. These parameters are not available openly but can be obtained by studying the HTML and JavaScript codes of the web interface of the camera. The studying of the

web interface codes may take time up to 1 to 2 days depending on the level of web programming knowledge. The URI where the footage of the camera can be obtained is identified with relative path *cgi-bin*.

2.3.2 Studying the Codes

The codes can be studied by using the *Developer Mode* of any web browser. In this thesis, the Chrome web browser has been used. The developer mode of Chrome web browser can be opened either through (Right click > Inspect) option or keyboard shortcut key (Ctrl-Shift-I).

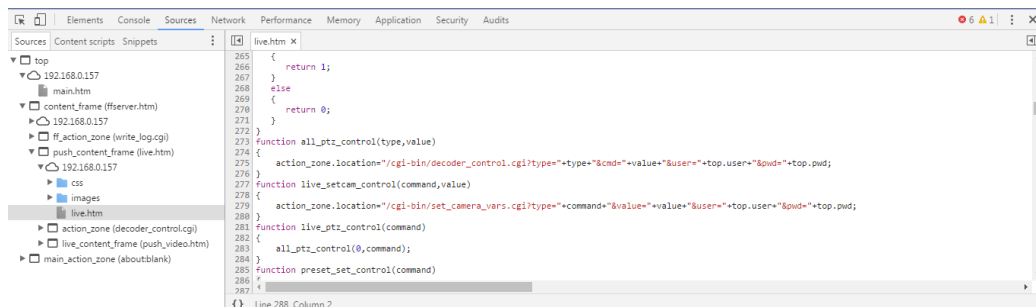
The the HTML and JavaScript codes of the IP-Camera web interface, however, can't be studied easily because there are a lot of files which are embedded upon one another and referred to and from one file to another. The method used to extract the required URI and query parameters are through backtracking the video footage and the callback functions of the controller buttons.

Upon successful backtracking, the video footage can be found out to be delivered through the following URI:

`http://<base_uri>/cgi-bin/videostream.cgi?user=<username>&pwd=<password>`
Where,

- base_url: 192.168.0.157 (*as per SearchIPCam.exe search result*)
- username: admin
- password: admin

Figure 2.6: Developer Mode panel Sources



The URI required to control the camera (panning and tilting), can be backtracked through the controller buttons. The backtrack will lead to a function called `all_ptz_control(type, value)` in the `live.htm` file. This

file can be found through the developer mode panel *Sources*. Expand content_frame (ffserver.htm) >push_content_frame (live.htm) >192.168.0.157 nodes to find and open that file (see Figure 2.6).

After opening the file, the function can be found either through manual scrolling and searching or through search functionality that can be invoked through keyboard shortcut key (Ctrl-F). Through this function, the required URI to control the camera can be found out, which is:

`http://<base_uri>/cgi-bin/decoder_control.cgi?type=<type>&cmd=<command>user=<username>&pwd=<password>`

Where,

- base_uri: 192.168.0.157 (*as per SearchIPCam.exe search result*)
- type: 0
- command: *see Table 2.1*
- username: admin
- password: admin

Table 2.1: Camera Control Commands

Command	Camera Movement
0	Up
1	Down
2	Right
3	Left
10	Stop
11	Centralize
13	Up-Left
14	Down-Left
15	Up-Right
16	Down-Right

2.4 Designing the Web Page

2.4.1 Introduction

In this section, the process of designing the web page to stream live footage of CCTV or IP-Camera will be discussed. In order to design this page, WordPress, CSS, JavaScript and jQuery knowledge will be required. In general,

the development involves styling with CSS. Additionally, JavaScript will be used to render the HTML view, handle user actions and execute AJAX request.

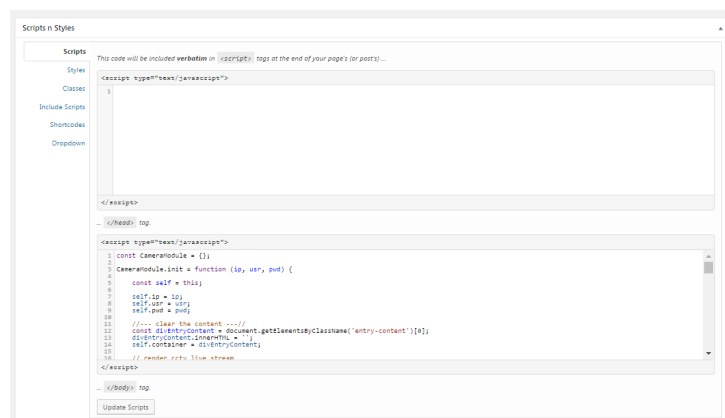
2.4.2 Creating the Web Page

The web page is created by using WordPress built-in functionality. To create a new page, go to 'Page' section from the left side menu and click on 'Add New' sub-menu. Here, the page name is given as 'Live Stream' and then the page is published by clicking 'Publish' button. All the other settings are left in the default mode.

2.4.3 Scripts n Styles Plugin

Next, JavaScript code will be added to render the HTML view. Below the WordPress text editor column, the 'Scripts n Styles' column can be found (*refer* Figure 2.7). This column will be added to every created page because of the installation of a WordPress plugin by the same name, 'Scripts n Styles'. To find out about the installed and activated plugins, explore the Plugins page. To learn more about this plugin, refer to URL linked in given footnote³. Basically, this plugin enable the WordPress user to add custom CSS and JavaScript codes globally or to individual pages. Custom CSS and JavaScript is added into the page created to render the IP-Camera recording and the controller buttons.

Figure 2.7: Scripts n Styles Column



2.4.4 Self-Invoked JavaScript Function

First, a self-invoked JavaScript function will be added into 'Scripts' lower column as shown in the Figure 2.7. The self-invoked JavaScript will be executed by the browser as soon as the script has been loaded without being called. This function contains codes to render the HTML view, which will render a HTML form with 3 input text fields:

- IP
- Username
- Password

and a submit button. The rendered view will then be inserted into HTML page under the `div` element with class name `'entry-content'`.

Figure 2.8: HTML form with 3 input text fields and a submit button



The image shows a web form with three input fields stacked vertically. The first field contains the text '192.168.0.'. The second field is labeled 'Username'. The third field is labeled 'Password'. Below these fields is a dark blue button with the text 'Submit' in white. At the bottom of the form, there is a thin horizontal line and a small copyright notice: '© 2017 Smart Home Lab, Hochschule Furtwangen University.'

In addition to that, the function has the callback method to handle when the form is submitted by the user. The codes can be found within the function block as shown below.

```
// self-invoked function to run when page is loaded
(function () {
    // codes to render view and form submission
    ...
    ...
})();
```

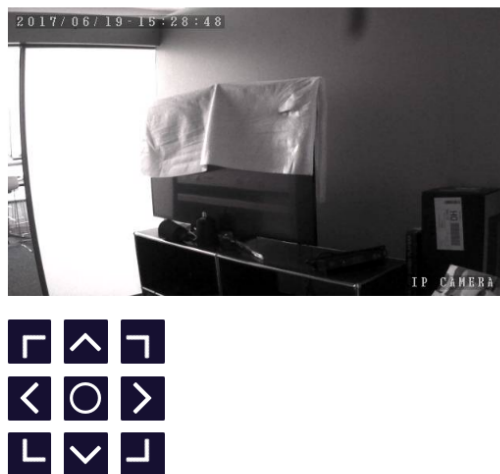
2.4.5 Rendering CCTV Footage and Controller

After the form as shown in Figure 2.8 is submitted by user, the JavaScript will execute the codes to render the CCTV footage view and controller buttons. The rendering of these elements is handled by a function with the name `CameraModule`. This function is divided into 3 sub-functions:

- `init()`
- `renderStream()`
- `renderController()`

```
const CameraModule = {};  
  
CameraModule.init = function () {  
  // codes  
  ...  
}  
CameraModule.renderStream = function () {  
  // codes  
  ...  
}  
CameraModule.renderController = function () {  
  // codes  
  ...  
}
```

Figure 2.9: Camera Module HTML View



These sub-functions contain codes to execute what their name suggest. The `init()` has codes to initialize the HTML view rendering process by clearing the current view and setting the text fields input values from the HTML form in the functional scope variables. The `renderStream()` function render the camera footage stream, and the `renderController()` renders the controller buttons into the view. The rendered HTML view will appears as shown in the Figure 2.9.

2.4.6 Styling and Positioning the Buttons

The controller buttons that are rendered through JavaScript alone do not appear as shown in Figure 2.9. In order to make them appear good and positioned in a group as such, some CSS styling and positioning have been done. The CSS codes used for this purpose can be viewed from 'Styles' section of the 'Scripts n Styles' column.

The positioning of the buttons are done through relative and absolute positioning mechanism of the CSS⁴. In addition to that, the buttons has to be given the position as where it should find itself within the button container. The direction of arrow, as to which direction it should be pointing is done through CSS-Transform rotate effect.

```
/* relatively positioned button container */
.btn-container {
    position: relative;
    width: 200px;
    height: 200px;
}

/* absolutely positioned buttons */
.navi-btn {
    line-height: 0;
    padding: 8px;
    position: absolute;
}

/* position of button in the button container */
.btn-top {
    top: 0;
    left: 72px;
}

... /* positioning of other buttons */

/* arrow direction of the button */
```

⁴<https://www.mediaevent.de/tutorial/css-position-absolute-relative.html>

```

| .arrow-down {
|   -webkit-transform: rotate(180deg);
|   -moz-transform: rotate(180deg);
|   -ms-transform: rotate(180deg);
|   -o-transform: rotate(180deg);
|   transform: rotate(180deg);
| }
|
| ... /* arrow positioning of other buttons */

```

2.4.7 Callback Functions

The last thing that needed to be added to this web page is the callback functions. Callback functions here refers to what should be done when the controller buttons are clicked using mouse or pressed using touch display such as on smartphone. The controller buttons remotely control the movement of the camera. All the buttons are registered with 4 types of event handlers except for the middle button, which has only 2 event handlers.

The 4 event handlers registered are as follow:

- onmousedown - when user click the mouse left button
- onmouseup - when user lift the clicking of mouse left button
- ontouchstart - when user start the touch of button (touch display)
- ontouchend - when user end the touch of button (touch display)

The 2 event handlers registered for center button are as follow:

- onclick - when user click on the button
- ontouch - when user touch on the button

All these event handlers are registered on the respective buttons during the rendering process as discussed in the Section 2.4.5. The registering is done through the JavaScript method `setAttribute()`. This method register the event handler and the callback function to be invoked as shown in the code example below.

```

| // .setAttribute(event_handler , callback_fn);
| btn.setAttribute('onmouseup', 'navi_stop(event)');

```

For these buttons, only 2 callback functions are required. First is the callback function when the events 'onmousedown', 'ontouchstart', 'onclick',

and 'ontouch' are detected. Second callback function is when the event 'onmouseup' and 'ontouchend' are detected. The first callback function, `navi_start(event, command)` contain codes to make AJAX request to the IP-Camera for starting the navigation as per user command. The second callback function, `navi_stop(event)` contain codes to make AJAX request to the IP-Camera to stop the navigation.

Each of the AJAX request is done through jQuery's AJAX method and by setting 'cross-domain' flag to true. The codes for both callback functions are as follow.

```
//--- Button Callback Functions ---//
function navi_start (event, cmd) {
    event.preventDefault();
    const ip = CameraModule.ip;
    const usr = CameraModule.usr;
    const pwd = CameraModule.pwd;

    jQuery.ajax({
        crossDomain: true,
        url: 'http://' + ip + '/cgi-bin/decoder_control.cgi?type=0&
            cmd='+cmd+'&user='+usr+'&pwd='+pwd,
    });
}

function navi_stop (event) {
    event.preventDefault();
    const ip = CameraModule.ip;
    const usr = CameraModule.usr;
    const pwd = CameraModule.pwd;

    jQuery.ajax({
        crossDomain: true,
        url: 'http://' + ip + '/cgi-bin/decoder_control.cgi?type=0&
            cmd=10&user='+usr+'&pwd='+pwd,
    });
}
```

For both AJAX request, 5 parameters are required, which includes

- URL
- type
- cmd
- user
- pwd

These parameters are the same as been discussed before in the Section 2.3.2.

Chapter 3

Integrating Unity 3D-Model

The Smart Home Lab website has been integrated with Unity 3D-Model. Unity is a game engine used to develop computer games and console games for cross-platform devices. With Unity game engine, 2D- and 3D-Model can be built through the provided APIs. The created 3D-Model can be viewed on web browsers through Unity Web Player plugins.

By default, WordPress doesn't have built-in Unity Web Player nor provide any support to view the Unity 3D-Models. Hence, integrating the 3D-Model has to be done manually. This chapter will outline and explain on how to integrate the 3D-Model into a WordPress powered website. Integrating Unity 3D-Model involves 2 steps. First, uploading the Unity files to resource folder. Second, developing a web page to render the 3D-Model.

3.1 Uploading Unity Files

This section will explain how to upload Unity files into WordPress resource folder. The Unity 3D-Model consists of following directories and files as shown below.

```
/
├── Build
│   ├── Prototyp_V2_Web.asm.code.unityweb
│   ├── Prototyp_V2_Web.asm.framework.unityweb
│   ├── Prototyp_V2_Web.asm.memory.unityweb
│   ├── Prototyp_V2_Web.data.unityweb
│   ├── Prototyp_V2_Web.json
│   └── UnityLoader.js
└── TemplateData
```

```

├─ favicon.ico
├─ fullscreen.png
├─ progressEmpty.Dark.png
├─ progressEmpty.Light.png
├─ progressFull.Dark.png
├─ progressFull.Light.png
├─ progressLogo.Dark.png
├─ progressLogo.Light.png
├─ style.css
├─ UnityProgress.js
└─ webgl-logo.png

```

These files has to be uploaded to the directory

```
/var/www/html/wp-content/uploads/
```

in the server. Uploading those 2 directories together with the files is done through `git clone` method. Fisrt, these files have been uploaded to a git repository. There is no any specific git account or git repository has been created for the Smart Home Lab website purpose. Please use your own or create one for the website if it is necessary.

After uploading 3D-Model files into a git repository, login into the backend of the server using PuTTY with hostname and port number as shown in the Figure 3.1. Don't forget to include the `private_key.ppk` for authentication under **Auth** tab in the PuTTY. Once the session is opened, enter the SSH passphrase `BSY2qjtu$#`

Navigate to the directory `/var/www/html/wp-content/uploads`. And use `git clone` method to clone i.e. download the required Unity 3D-Model files to the server. Replace the `[url]` with the URL address of the git repository.

```

| cd /var/www/html/wp-content/uploads
| git clone [url]

```

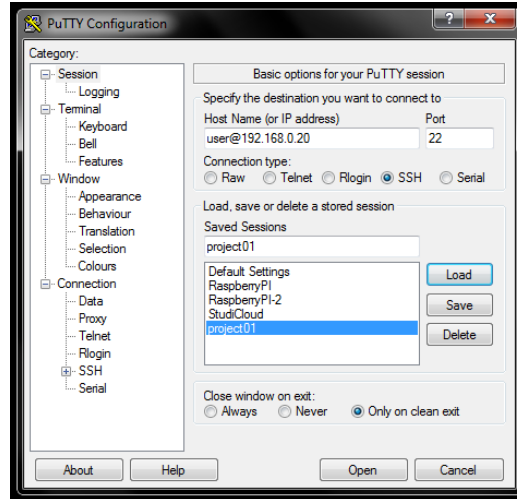
Since the directories and files are uploaded through the system user of Ubuntu Server, by default WordPress don't have permission to access them. File ownership and file permission for those directories and files has to set using `chown` and `chmod`.

```

| chown -R user:www-data /var/www/html/wp-content/uploads/Build
| chown -R user:www-data
|   /var/www/html/wp-content/uploads/TemplateData
| chmod -R g+w /var/www/html/wp-content/Build
| chmod -R g+w /var/www/html/wp-content/TemplateData

```


Figure 3.1: Login into backend using PuTTY



Uploading the required Unity 3D-Model is done with setting of file ownership and file permission. In the next section, the rendering of HTML page will be discussed and explained.

3.2 Rendering HTML for Unity Web Player

This section will discuss on adding the HTML tag to render the Unity Web Player as well as linking the required Unity 3D-Model files, that have been uploaded to web server directory as discussed in the Section 3.1.

First, a new web page has to be added by using the WordPress built-in Page function. Click on 'Pages' option from the side menu and 'Add New' sub option. Here, '3D Model' has been given as the title of the page with /unity as the relative web page path.

In the editor column, HTML codes have been added to render the Unity Web Player. The web player will be rendered inside 2 div containers with class names `webgl-content` and `gameContainer` respectively. However, the web player is fixed to size of 960 pixel width and 600 pixel height as set by the author of the 3D-Model. This attributes have not been changed. Fixing the width and height will cause the page not to act responsively across various screen sizes. Codes snippet below shows the rendering of Unity Web Player.

```
<div class="webgl-content" style="margin-top: 17%;  
margin-bottom: 10%">  
  <div id="gameContainer" style="width: 960px; height:  
    600px"></div>
```

```

    <div class="footer">
        <div class="webgl-logo"></div>
        <div class="fullscreen"
            onclick="gameInstance.SetFullscreen(1)"></div>
        <div class="title">Smart-Home-Labor</div>
    </div>
</div>

```

Next, required files has to be linked to web page. These files include **style.css**, **UnityProgress.js**, and **UnityLoader.js**. These files are included onto the web page as shown in code snippet below.

```

// linking required files
<link rel="stylesheet"
    href="../../wp-content/uploads/TemplateData/style.css">
<script
    src="../../wp-content/uploads/TemplateData/UnityProgress.js"></script>
<script
    src="../../wp-content/uploads/Build/UnityLoader.js"></script>

```

After that, the web player has to instantiated using the **UnityLoader.instantiate()** function.

```

<script>
    const gameInstance = UnityLoader.instantiate(
        "gameContainer",
        "../../wp-content/uploads/Build/Web.V6.json",
        {
            onProgress: UnityProgress
        }
    );
</script>

```

Lastly, in order to enhance the user interface level, the title of the web page has to be removed. This ensures that the Unity Web Player uses the maximum space in the view. The default title of the web page is removed by using CSS code as shown below. This code has be inserted into 'Scripts n Styles' column under 'Styles' tab.

```

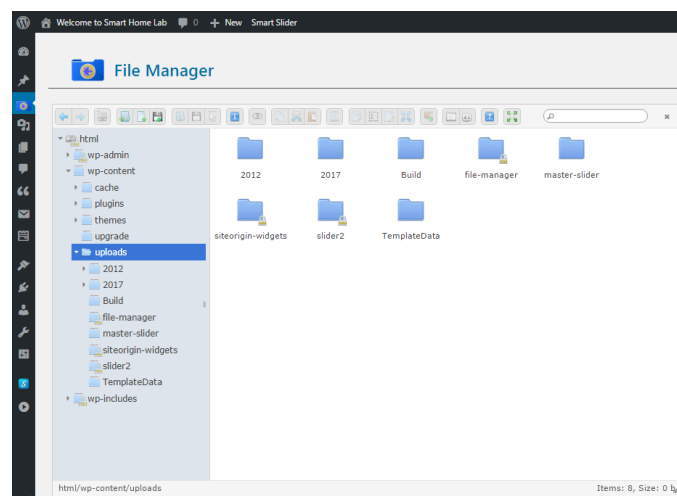
.entry-header {
    display: none;
}

```

3.3 Alternative way to upload Unity 3D-Model Files

Another alternative way to upload the required Unity 3D-Model files is by using the 'File Manager' plugin¹. This plugin is available free of cost in the WordPress plugins directory². This plugin can be added to the website and activate through the web browser.

Figure 3.2: File Manager Plugin



After installation and activation of this plugin, a new 'File Manager' option will appear on the left menu bar. This plugin enables the directories and files on the backend server to be viewed on the front-end web interface (see Figure 3.2).

Through this plugin, directories and files can be added, modified and deleted. Here, the root directory starts with `/html/.`, instead of `/var/www/html/.`. The `html` is the root directory of WordPress where else `/var/www` is the root directory of the Nginx web server.

Uploading file through 'File Manager' plugin can be done simply by drag-n-drop method. The files, however, have to be made sure uploaded to the correct directory, which is `/html/wp-content/uploads`. Since we used WordPress frontend to upload the files, no file ownership and file permission has to be set. Here the files and directories are added with WordPress as the owner.

Even though this method is much simpler than the method discussed in the Section 3.1, a problem has occurred. There is a file with size of almost 12

¹<https://wordpress.org/plugins/file-manager/>

²<https://wordpress.org/plugins/>

MB in the **TemplateData** directory. This file are not uploading to the server by using the 'File Manager' plugin. The WordPress deliver a **http-error**. After research and studies, no working solution has been found on how to enable large file upload of more than 10 MB.

This is taking into consideration after increasing the file upload size limit both in the Nginx web server configuration and PHP application server configuration. Solving the **http-error** will enable the file to uploaded or upgraded easily in the future by using this plugin, without having to login in to server backend and uploading file using PuTTY or SFTP.