

# Chapter 1

## Background Works

The background works involved in this thesis in order to create the Smart Home Lab website divided into two parts. The first part is the analysis of selected Smart Home Lab websites of other institutions. The second part is the analysis of the WordPress plugin in order to create the Info-Terminal.

### 1.1 Website Analysis

In this work, altogether six websites have been analyzed. These include FZI Forschungszentrum, KIT iZEUS, IoTLab Reutlingen, Duke Smart Home Program, MIT Mobile, and MIT Smart Living. In general, criteria such as the layout, contents, main navigation menu, media supported, responsiveness and languages supported have been analyzed.

#### 1.1.1 FZI Forschungszentrum

##### Website Information

- URL: <https://www.fzi.de/en/home/>
- Web server: Apache
- Application server: PHP 5.4.45
- CMS: Typo3

##### Web Layout

The website uses normal layout, not screen-wide. The theme look modern, but the normally used one. The Header contains institution logo on the

right, and the top part has option to change language, link to contact page and search bar. Lower part of the header has main navigation menu (Home, News, Research, Our Offer, Work For Us, and About Us). The header is fixed, but collapse when user scroll down.

Body is two column sized. In all pages except home page, the small left column has sub menu respective to that individual page and large right column has the content. On the main page, the the left column is relatively bigger which has slideshow and latest news feed. Right column is relatively small and has list of upcoming events and quicklinks.

Footer is small sized. It has sub menu on the left (Home, Privacy, Legal Notice, Sitemap, Search) It also has social media plugins on the (Xing), RSS feeds and contact links.

## **Contents**

Contents presented on the website are the home page (featured article/news, upcoming events, latest news, quicklinks), newsfeed, research (research sector, research focuses, projects etc), offers, work for us (carrier page), about us, privacy, legal notice, and sitemap. As analyzed, the website has a lot of contents, perhaps it could have been organized in a better way.

## **Menu**

There are 3 types of menu:

- Main navigation menu situated at the header (Home, News, Research, Our Offer, Work For Us, and About Us)
- Sub menu respective to individual pages which is different from one another located at right column of the body part
- Footer menu located the footer (Home, Privacy, Legal Notice, Sitemap, Search).

## **Media**

The website supports the following medias, such as images, slide show, videos, contact form, search bar, social media button.

## **Responsiveness**

The webpage is responsive. The header is compressed. The main navigation menu is shifted to the bottom of the page with a link to the menu located

at the header. The two column body part is changed to single column on window resize. The sub menu on the individual pages are merged with main navigation menu in hierarchical order, which is situated at the bottom of the page.

## **Languages**

The website supports German and English languages.

### **1.1.2 KIT iZEUS**

#### **Website Information**

- Url: <http://www.izeus.kit.edu/62.php>
- Web server: Apache 2.4.10
- Application server: PHP
- Server: Debian

#### **Web Layout**

The web layout is classic normal one. It is not wide and not stretched. The institution logo is located in the header at the top. The sub-menu can be spotted small in the upper right part of the header. The main frame is divided into three columns. The left column contain the main navigation menu. The center column serves as the main content. Lastly, the right column fixed banner. The footer is small and has copyright statement.

The web layout is classic without fancy popup animations, graphics, slider etc. It has to also noted that it is most common and user friendly where most users know how to navigate.

#### **Contents**

Contents presented on the website are the home page, energy smart home lab, information materials, project consortium (partners and KIT chairs), publications, press review, links, contact, legals and sitemap. The content look organized, clear to be viewed and read. Navigating through the contents is also easy, but it appears not to be updated anymore.

## **Menu**

The website contains two menus:

- Small top right menu (Home, Lang Pref, Legals, Sitemap, Link to KIT)
- Main Menu on Left Column (Home, Energy Smart Home Lab, Information Material, Project Consortium, Publications, Press Review, Links, Contact)

Analysis: Both navigation are visible on all pages, static, easy to use, simple hovering effect, highlight on active link.

## **Media**

The website supports the following media, video, images, banners, PDFs, and external links. However, no animation, picture gallery and image slider can be spotted.

## **Responsiveness**

The website is not responsive. It is best viewed with desktop and laptop. Having said that, it is not mobile- or tablet-friendly and not recommended for Microsoft Surface Hub.

## **Languages**

The website has German as the primary language and supports English as an alternative language. The website is designed will on both the languages.

### **1.1.3 IoTLab Reutlingen**

#### **Website Information**

- URL: <http://iotlab.reutlingen-university.de>
- Web server: Apache
- Application server: PHP 5.5.36
- CMS: Joomla!

#### **Web Layout**

The website uses a modern web layout and the web page is full-width stretched.

# Chapter 2

## Device

Even though, the website designed for this thesis encompasses all range of devices from desktop to mobile devices, the main focus has been the Microsoft Surface Hub.

### 2.1 Microsoft Surface Hub

Surface Hub is a collaborative device from Microsoft. It has a 55-inch touch display with 1920 times 1080 pixel resolution. The main aim of the device is to encourage team work between people and increase their productivity. It has been adapted with various technologies such as wide angle view display, camera and responsive pen.

# Chapter 3

## Technologies

### 3.1 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](http://wordpress.org). The current stable version of WordPress is 4.8, 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.

## 3.2 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 website<sup>1</sup>.

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.

## 3.3 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 website<sup>2</sup>. MySQL uses structured query lan-

---

<sup>1</sup><http://php.net>

<sup>2</sup><https://dev.mysql.com/downloads/>

guage (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.

### **3.4 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.



# Chapter 4

## Server Setup and WordPress Installation

In this project, WordPress is setup on Nginx HTTP web server running PHP preprocessor and MySQL database server. This chapter will explain on how to configure Ubuntu server, setup Nginx, PHP, MySQL database as well as WordPress.

### 4.1 Ubuntu Server Configuration

The Smart Home Lab has its own server running Ubuntu Server 16.04.2 LTS. On this server, a virtual machine (VM) application has been installed. This VM runs a separate virtual Ubuntu Server on the host. This VM has been allocated with its own processing power, memory and disk space. Local machine can connect to the virtual Ubuntu Server using the Wi-Fi network, SHLAB02 as shown in Figure 4.1.

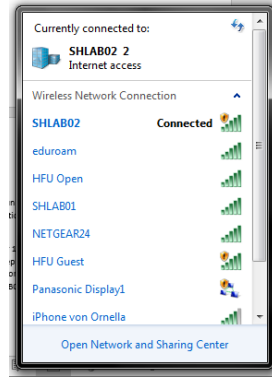
On the local machine, first, a SSH key or commonly known as public/private key should be generated. This can be done using the PuTTY Key Generator application as shown in Figure 4.2a. This application can be downloaded at the URL given in the footnote<sup>1</sup>. The current stable version of Putty Key Generator is 0.68, as at time of writing.

The keys can be generated by clicking generate button and by moving the mouse cursor randomly at the blank area as shown in Figure 4.2b. After that, the Key Passphrase and Confirm Passphrase must be given. This Passphrase must be noted down or remembered for future usage when connecting to the server every time. The Passphrase is **BSY2qjtu\$#**

---

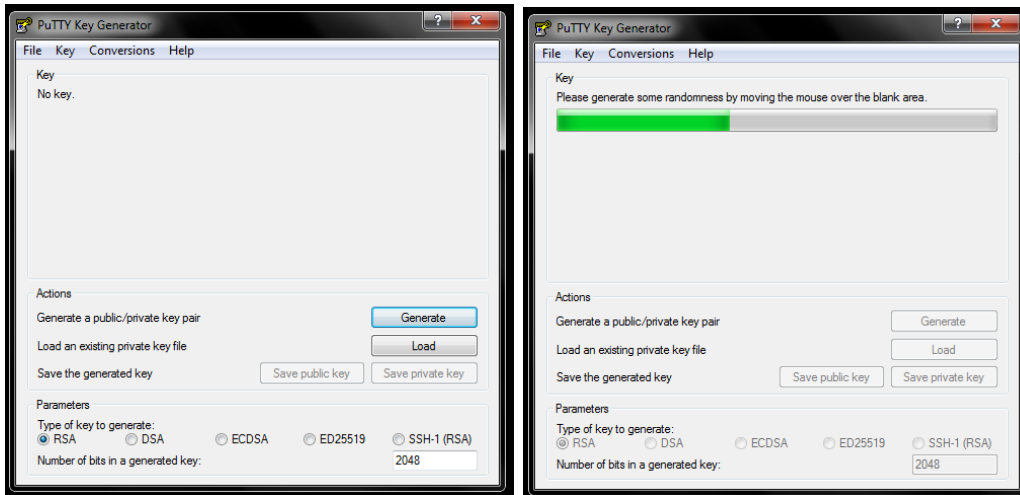
<sup>1</sup><http://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>

Figure 4.1: SHLAB02 WLAN network to connect to Ubuntu Server



(a) PuTTY SSH Key Generator

(b) Generating random key in PuTTY

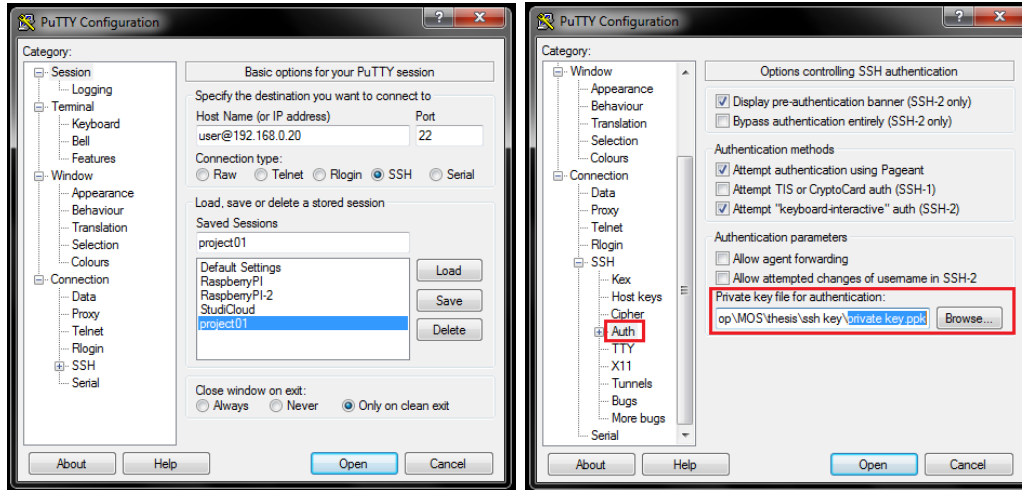


The generated Public Key and Private Key must be saved. The generated Public Key will be provided to the virtual Ubuntu Server running on the server. Where else the generated Private Key must be saved in the local machine that will be used to connect to the server.

To connect to the server, the PuTTY application will be used. This application can be downloaded at the same link as the PuTTY Key Generator link given above. The PuTTY application will be provided with Host Name (or IP Address) of the server, which is `user@192.168.0.20` and Port number, which is 22 as shown in the Figure 4.3a. After that the generated Private Key from previous step must be given in the Private key file for authentication field as shown in Figure 4.3b. This field can be found under Connection  $\rightarrow$  SSH  $\rightarrow$  Authentication tab. Lastly X11 forwarding must be enabled. Connection to the server can be established by pressing Open button.

(a) Host Name and Port

(b) Attaching Private Key



After successfully connecting, the passphrase, BSY2qjtu\$# that have been entered while generating the Public/Private keys must be entered in the terminal to proceed.

## 4.2 Nginx HTTP Server Installation

Nginx is a powerful, versatile and efficient HTTP or web server. Nginx web server can be installed in Ubuntu using the `apt` package.

### 4.2.1 Step 1: Installing Nginx

The following has been used to install Nginx.

```
$ sudo apt-get update
$ sudo apt-get install nginx
```

### 4.2.2 Step 2: Enabling Nginx

The Nginx, by default, will be configured and started running automatically after the installation. If the server runs firewall, the Nginx should be enabled manually. The Nginx will however register itself with the server firewall, which is `ufw` upon installation. The following command has to be ran in the terminal to start the Nginx manually.

```
$ sudo ufw allow 'Nginx HTTP'
```

### 4.2.3 Step 3: Checking Nginx Status

The status of Nginx HTTP server can be checked by using the following command.

```
|$ sudo ufw status
```

Running this command will output the following.

To	Action	From
—	—	—
OpenSSH	ALLOW	Anywhere
Nginx HTTP	ALLOW	Anywhere
OpenSSH (v6)	ALLOW	Anywhere
Nginx HTTP (v6)	ALLOW	Anywhere

The output on the second and third column which are 'ALLOW' and 'Anywhere' respectively indicates that the Nginx is running. The successful installation of Nginx can be further tested using any web browser by entering the host's IP address, which in this case `http://192.168.0.20`. The following page will be displayed as shown in the Figure 4.4.

Figure 4.4: Nginx Default Start Screen



## 4.3 MySQL Database Installation

After installing Nginx HTTP server, database server has been installed to manage the storing of the website content. In this project MySQL database server has been used.

### 4.3.1 Step 1: Installing MySQL

The following command has to be entered to install MySQL database on the Ubuntu server.

```
|$ sudo apt-get install mysql-server
```

During the installation, root password will be prompted or asked. Here, the password `d_XKdEoCwX,4EnO` has been given. Here are the MySQL database credentials for future reference:

- username: root
- password: `d_XKdEoCwX,4EnO`

### 4.3.2 Step 2: Securing MySQL Database

This section will discuss on how to secure the MySQL database server. To get started, the following command has to be entered in the terminal.

```
|$ sudo mysql_secure_installation
```

Running this command will take us through security setup of MySQL database server which consist of a series of questions. Here are the three questions and the settings that have been given for the future reference. The rest of settings have been given 'yes' as the answers.

```
|VALIDATE PASSWORD PLUGIN can be used to test passwords
|and improve security. It checks the strength of password
|and allows the users to set only those passwords which are
|secure enough. Would you like to setup VALIDATE PASSWORD plugin?
```

```
|Press y|Y for Yes, any other key for No: y
```

```
|There are three levels of password validation policy:
|LOW Length >= 8
|MEDIUM Length >= 8, numeric, mixed case, and special characters
|STRONG Length >= 8, numeric, mixed case, special characters and
|dictionary file
```

```
|Please enter 0 = LOW, 1 = MEDIUM and 2 = STRONG: 1
```

```
|Using existing password for root.
```

```
|Estimated strength of the password: 100
```

```
| Change the password for root ? ((Press y|Y for Yes, any other  
| key for No) : n
```

## 4.4 PHP Preprocessor Installation

Nginx web server does not contain native PHP preprocessor. A package called **php-fpm**, which stand for "fastCGI process manager" must be installed on the Ubuntu server. This package tells Nginx to pass the PHP request to **php-fpm** for processing. Second package that has to be installed is **php-mysql**, which allows PHP to communicate with the database server.

### 4.4.1 Installing php-fpm and php-mysql

The following command has been entered to install **php-fpm** and **php-mysql**.

```
| $ sudo apt-get install php-fpm php-mysql
```

### 4.4.2 Securing PHP configuration

Initial installation of PHP has to be secured by commenting out the `cgi_fix_pathinfo` line and setting its value to 0 in the PHP configuration file, `php.ini`.

```
| $ sudo nano /etc/php/7.0/fpm/php.ini  
| /etc/php/7.0/fpm/php.ini  
| ...  
| cgi.fix_pathinfo=0  
| ...
```

Save and close the nano editor. Next, the PHP preprocessor has to be restarted so that the changes will take effect by entering the following command.

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

### 4.4.3 Configuring Nginx to use PHP Preprocessor

In this section, a few settings changes on the Nginx will be shown in order to ask Nginx to use PHP preprocessor. The server settings of Nginx can be opened using the following command.

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

After opening the server settings, `index.php` has to be added to `index` line in the server block. In addition to that, two new block `location ~\php$` and `location ~/\.ht {` has to be introduced.

```
server {  
    ...  
    index index.php  
  
    location ~ \.php$ {  
        include snippets/fastcgi-php.conf;  
        fastcgi_pass unix:/run/php/php7.0-fpm.sock;  
    }  
  
    location ~ /\.ht {  
        deny all;  
    }  
}
```

After adding those lines to the Nginx configuration files, the configuration file has to be checked for error. If no error is reported, the web server can be restarted safely as shown below.

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

#### 4.4.4 Testing PHP Preprocessor

The installation and configuration of PHP preprocessor can be tested by introducing `info.php` to the server root directory. A PHP information file can be created by using the following command.

```
$ sudo nano /var/www/html/info.php
```

In the PHP file, following lines have to be added. After adding those lines, save and close the file.

```
<?php  
phpinfo();
```

Now the PHP preprocessor can be tested using web browser by entering the URL, `http://192.168.0.20/info.php`. After testing the PHP preprocessor, it is important that `info.php` file has to be deleted to prevent anyone from learning the PHP configurations.

## 4.5 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('AUTHKEY', 'put your unique phrase here');  
define('SECUREAUTHKEY', '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('SECUREAUTH_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:

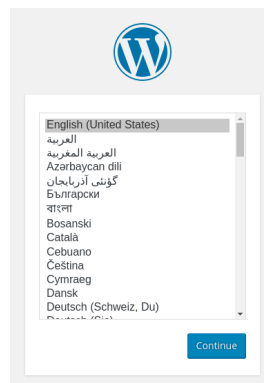
```
| define( 'FS_METHOD' , 'direct' );
```

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 4.5: WordPress language selection



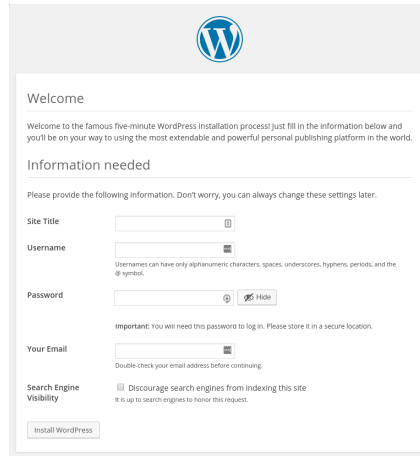
After that, we need to provide a few information on the main setup page such as *Site Title*, *Username*, *Password* and *Email*.

These are information given on the page shown above:

- Site Title: Smart Home Lab
- Username: ashiqmoh
- Password: oSm0cCCAgulMsalrZd(#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:

Figure 4.6: Main WordPress setup page

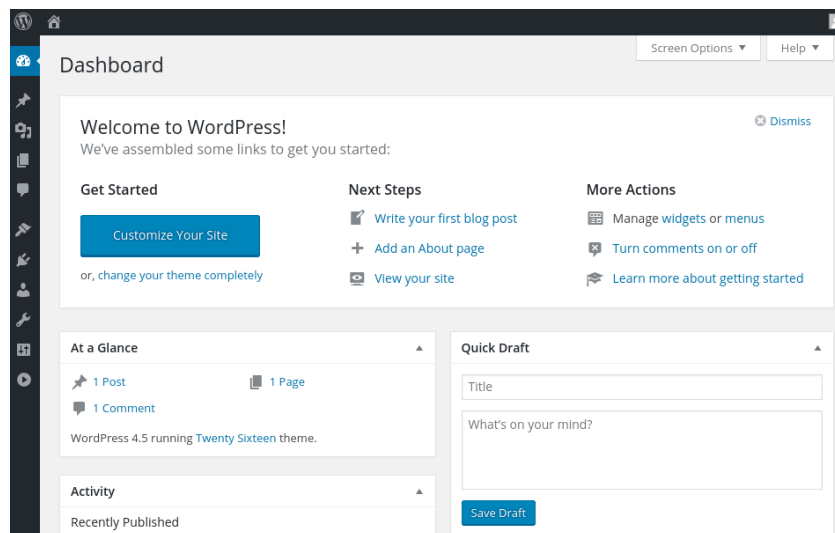


The image shows the WordPress installation setup page. At the top is the WordPress logo. Below it is a 'Welcome' section with a brief introduction. The main section is 'Information needed', which contains several input fields: 'Site Title', 'Username', 'Password' (with a 'Hide' button), and 'Your Email'.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

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

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

Figure 4.7: Administration page of WordPress



# Chapter 5

## Website Designing

The Smart Home Lab website consists of 5 web pages. Those are:

- Home
- The Lab
- Components
- Panels
- Use Cases

These web pages have been created and designed within the WordPress by using a WordPress plugin called 'Page Builder By SiteOrigin'. This chapter will discuss on how these web pages has been created and designed, as well as the important steps and settings involved in it.

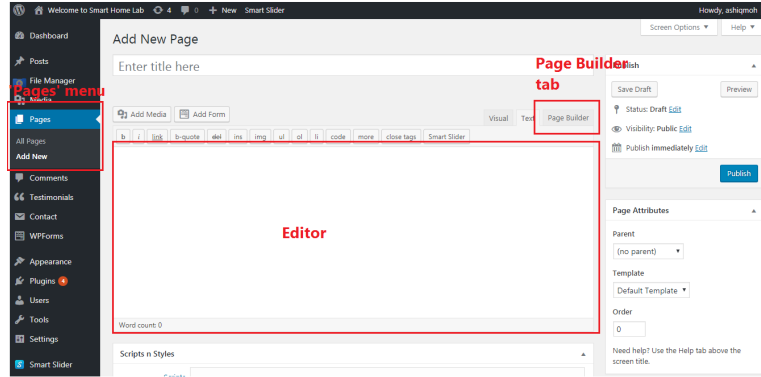
### 5.1 Enabling Page Builder Plugin

Firstly, the 'Page Builder by SiteOrigin' has to be downloaded and activated in the WordPress. This can be done through the 'Plugins' section in the admin side of the WordPress. After downloading and activating the plugin, the Page Builder plugin will automatically add a tab on the top right of the page editor (refer Figure 5.1), when a new page has been created. The usage of Page Builder can be enabled simply by clicking on this tab.

### 5.2 Adding row, column and widget

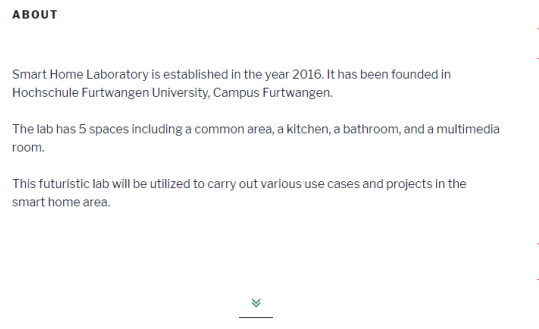
When Page Builder plugin has been enabled, the editor will display a series of buttons on the top of the editor. One of the important thing here is the

Figure 5.1: Enabling Page Builder Plugin



'Add Row' option. A row in the page builder signifies a horizontal section in a web page as shown in the Figure 5.2. Each of this section is a row.

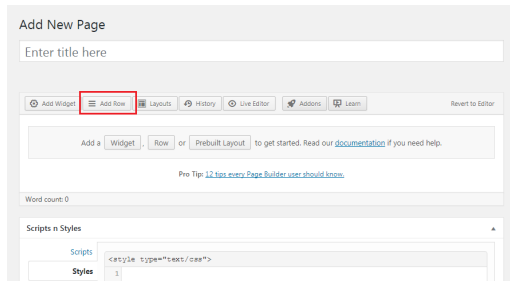
Figure 5.2: A row representing a horizontal column on the website



When the 'Add Row' button (see Figure 5.3a) is clicked, a new window (see Figure 5.3b) will be opened asking to set the column. Here, the number of columns as well as the individual size of the column has to be given in.

After a row and a column (optionally multiple columns to a row) have been added, the editor will display a box within it. Within this box, widgets can be added by clicking the 'Add Widget' button next to the 'Add Row' button. When this button is clicked, a new window (see Figure 5.4) will pop up prompting which widget to be added into the created row i.e. column. The list of widgets consists of, for an example, 'SiteOrigin Editor', 'SiteOrigin Button', 'SiteOrigin Button' etc. To add normal text into the web page, one needs to insert the 'SiteOrigin Editor' widget into the row. For a short note, multiple widgets can be inserted into a single row i.e. column.

(a) Adding Row



(b) Adding Column

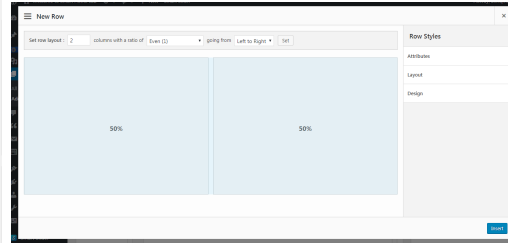
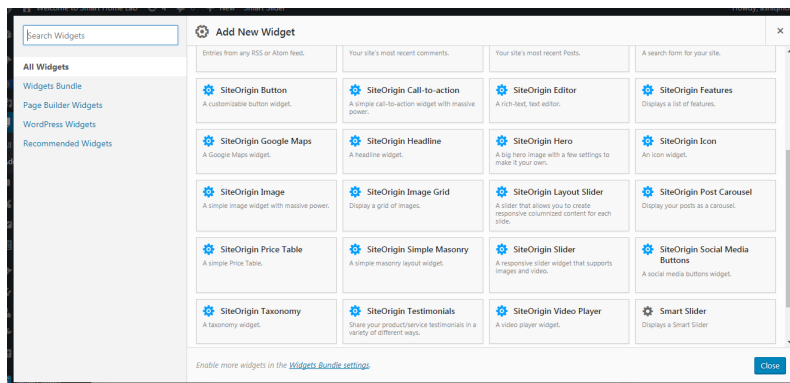
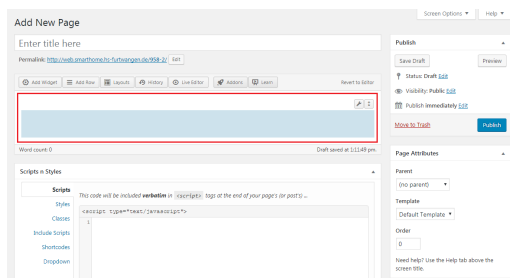


Figure 5.4: Window showing a list of widgets to be inserted in to row i.e. column

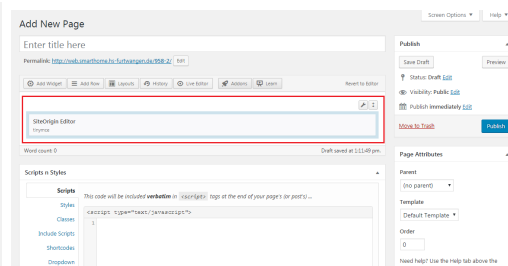


Here is an example of screenshot of page builder row before and after adding a widget. In this example, the SiteOrigin Editor widget has been added.

(a) Row before adding widget



(b) Row after adding widget





## 5.3 Adding text, image and button

Text can be added into the web page through the SiteOrigin Editor plugin. After the SiteOrigin Editor plugin has been inserted into the row/column, a menu list will appear when one hover the mouse over it. In the menu list, the 'edit' option has to be clicked to open the editor. In the editor, the text that has been intended to be added to web page can be typed. Text here can represent the header text and the normal paragraph text.

Next, in order to add an image, the SiteOrigin Image plugin can be used. After inserting the plugin, through the edit option, a new window will be opened. Here, the URL of image as well as other settings such as image size, image alignment, image title etc. can be entered.

The button which are found on the web pages are added through the SiteOrigin Button plugin. Same as step above, after adding the plugin, one has to click on the 'edit' option on mouse over. Here, one has to define the destination URL which will be opened when the button is clicked. Optionally, the text or icon that should appear on the button can be added. There are also other options for alignment layout designing of the button.

## 5.4 Adding Scroll Effect

As can be noticed, clicking on the button scroll the content to a specific part of the web page. To enable the scroll effect, a few steps have to be taken.

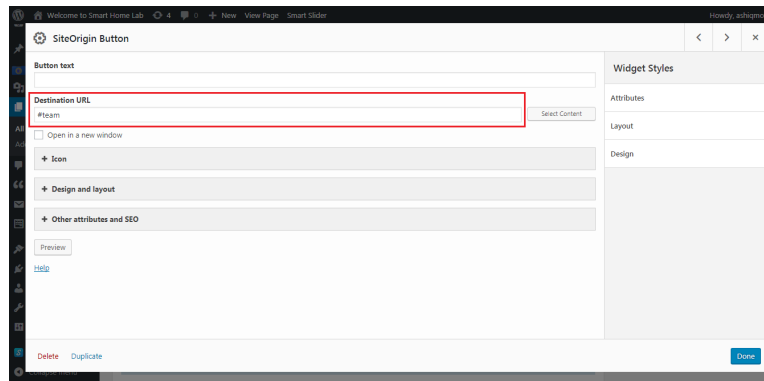
First, a plugin called 'Page Scroll to Id' has to be added and activated in the WordPress. After adding and activating, buttons that have been added has to be given the class name 'ps2id'. The class name of a button can be given by the 'edit' option mentioned in the previous section. In the edit section, under the 'Other attributes and SEO', the class name can be given in the 'Button Classes' field as can be seen in the Figure 5.6.

Lastly, the destination URL has to be set. The destination URL has to be entered in two places. First in button edit section, under the 'Destination URL' field as can be seen in the Figure 5.7. Secondly, the same destination URL has to be given to 'Row ID' field of the target element. The step to provide the target element an id is shown in the Figure 5.8. By assigning the button and the target element the id, the button will scroll the web page to target element.

Figure 5.6: Button Classes field where 'ps2id' has to be entered



Figure 5.7: Adding the destination URL for a button



## 5.5 Adding ScrollReveal Effect

ScrollReveal effect is an effect that can be added to the content of a web page, where the content will get an animation or effect that it is being revealed to the users as they scroll down or up through the page. This effect can be added to any website through importing a JavaScript library, ScrollReveal written by Julian Lloyd<sup>1</sup>.

This section will explain how to add this effect to a WordPress page. First, a plugin called 'Scripts n Styles'<sup>2</sup> has to be added and activated in the WordPress. After activating the plugin, the 'Page' section in the WordPress will receive a new screen option under the editor, where JavaScript and CSS code can be added as shown in the Figure 5.9.

Here, the script tab has to be clicked so that our own JavaScript can be

---

<sup>1</sup><https://scrollrevealjs.org/>

<sup>2</sup><https://wordpress.org/plugins/scripts-n-styles/>

Figure 5.8: Adding Row Id to the Target Element

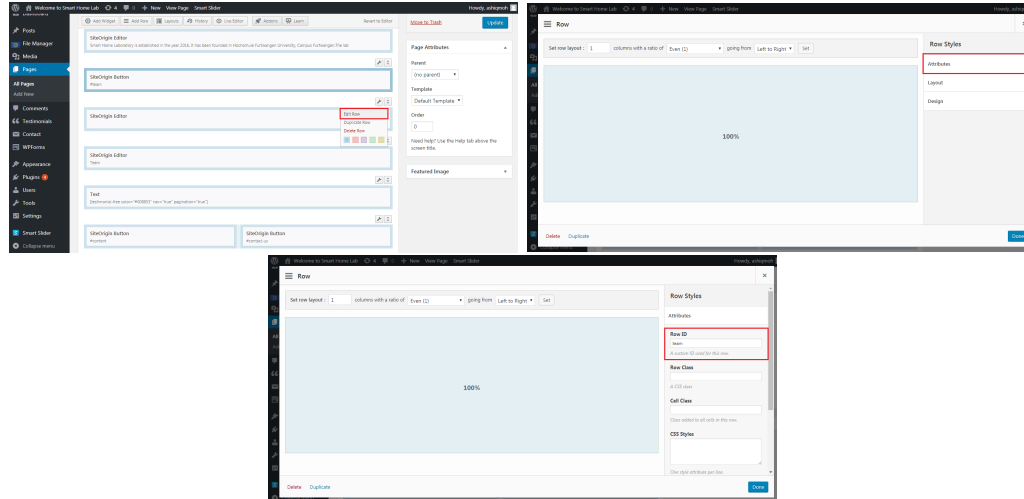
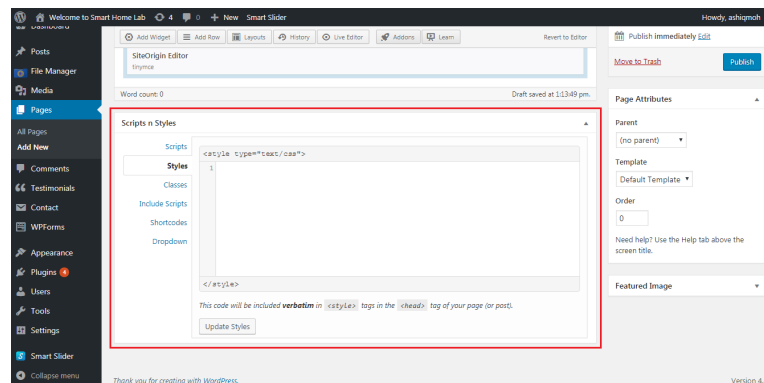


Figure 5.9: Scripts n Styles screen option

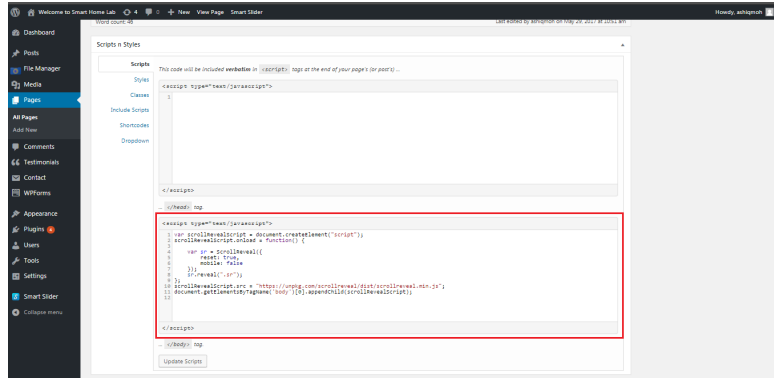


added to the current page. There, the plugin offers two option of adding JavaScript codes to current web page. Either at the header part of web page at the top, or at body part of the web page at the bottom. Here, the codes required to enable ScrollReveal effect will be added at the body of the web page at the bottom.

The JavaScript codes that has been added will do two tasks. First, it will import the ScrollReveal library script from Content Delivery Network (CDN) and insert it to current web page.

Secondly, the JavaScript code will be written to declare and initiate the ScrollReveal object, so that any HTML element assigned will receive the effect. Here, the elements with class name 'sr' has been assigned to get the ScrollReveal effect. The class name in JavaScript is identified with the

Figure 5.10: Adding JavaScript to bottom of page



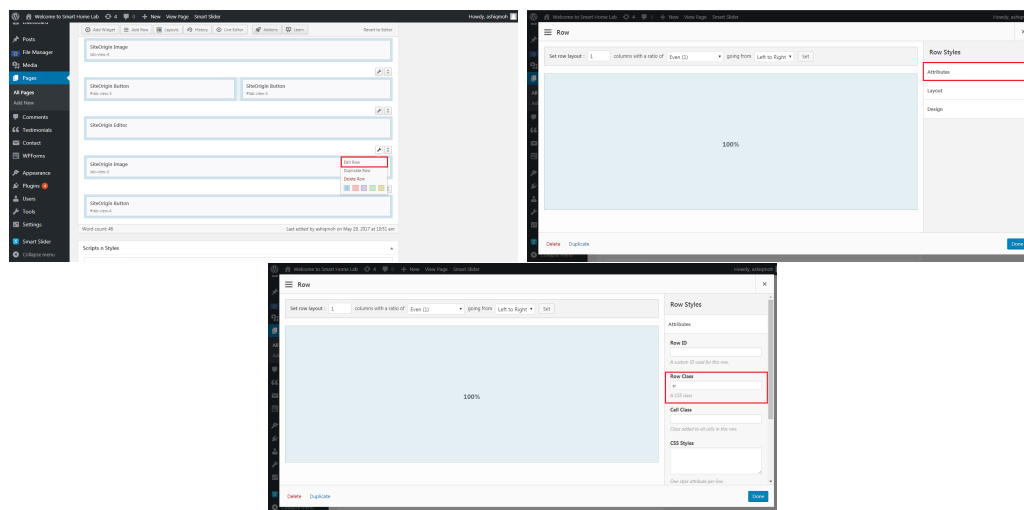
punctuation mark `'.'` in the beginning. This has been done using the method `.reveal('.sr');`.

```
var scrollRevealScript = document.createElement("script");
scrollRevealScript.onload = function() {

    // declare and initiate ScrollReveal object
    var sr = ScrollReveal({
        reset: true,
        mobile: false
    });
    // assign element that should receive the effect
    sr.reveal(".sr");
};
// import ScrollReveal library from CDN
scrollRevealScript.src =
    "https://unpkg.com/scrollreveal/dist/scrollreveal.min.js";
// insert the imported library into the web page
document.getElementsByTagName('body')[0].appendChild(scrollRevealScript);
```

The final step that has to be taken in order to enable the effect is to add the class name, in this case `'sr'` to the HTML elements. This class name will be entered in the `'Row Class'` field as shown in the Figure 5.11.

Figure 5.11: Adding Class Name to HTML Element



## Chapter 6

# Implementation of the Info-Terminal

The info-terminal has been created by using a WordPress plugin called SmartSlider3<sup>1</sup>. SmartSlider3 is a free plugin available in the WordPress plugin repository. It enables the creation and designation of slider easily with a lot of features.

### 6.1 Getting Started

First, the SmartSlider3 plugin has to be added and activated in the WordPress. After the activation, the left menu list will receive a new entry 'Smart Slider'. Click it will bring the Smart Slider dashboard interface, where slider can created, modified and deleted. A new slider can be created by clicking the 'New Slider' tile on the dashboard. Here the following details have been given:

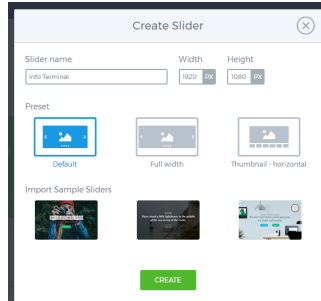
- Slide name: Info Terminal
- Width: 1920 px
- Height: 1080 px
- Preset: Default
- Import Sample Sliders: none

After the above mentioned details have been given, the creation of new slider have been proceed by clicking the 'Create' button.

---

<sup>1</sup><https://wordpress.org/plugins/smart-slider-3/>

Figure 6.1: Creating a new slider



## 6.2 Slider Settings

There are a few general settings that has to be configured after creating a new slider. The slider settings options can be found immediately after getting into the 'Info Terminal' slider.

subsection\*General Settings Under 'General' tab, two settings has be changed, which are:

- Align: Center
- Main animation properties (Duration): 600 ms

### Size Settings

Here, the following attributes has to be changed:

- Slide size (Width): 1920 px
- Slide size (Height): 968 px
- Slider height (Min): 300 px
- Slider height (Max): 968 px
- Slider width (max): 1920 px

### Autoplay Settings

Here, the following attributes has to be changed:

- Autoplay (Enabled): True
- Autoplay (Interval): 5000 ms

## Arrow Settings

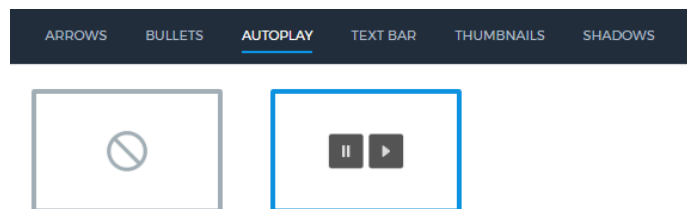
Here, the following attributes has to be set:

- Previous (Color): 999999FF
- Style: Static

## Autoplay Icon Settings

By default, the autoplay mode in the slider is turned off. This mode has to be turned on as shown in Figure 6.2 and the rest the of settings are left to default settings.

Figure 6.2: Autoplay Icon Settings



## Thumbnails Setting

After enabling the Thumbnails, which by default is turned off, the following attributes have to be set.

- Thumbnail size (Width): 150 px
- Thumbnail size (Height): 100 px
- Position: Outer, bottom

## Publish Setting

Under the 'Publish' tab, the shortcode of the created slider has to be taken note (refer Figure ??). In this case, the short code is

$$smartslider3slider = 1$$

. This code will be used to insert the slider into the WordPress web page as discussed in the Section 6.3.



## 6.3 Inserting Slider Into Web Page

A new WordPress page has to be created through the **Add New** option under the **Pages** menu.

1. The title of page has been given as 'Info Terminal' in the 'Title' field.
2. The permalink or slug has been set to `info/`.
3. The shortcode, which have been noted down from previous section has to be pasted or typed in the editor.
4. Lastly, the template of this web page has been changed from 'Default Template' to 'Blank Slate'.

## 6.4 Creating and Setting Slide

After setting the slider setting, slide can be created and set. The slider can be created by clicking the 'NEW SLIDE' green icon tile as shown in the Figure 6.3. After clicking it, we will be prompted to select an image. Here, any dummy image has to be chosen for time being until the slide background is set in the next step.

Figure 6.3: New slide tile icon



Next, after the slide has been created, the following Background and Settings has to be given as shown in the Figure 6.4. Through the 'Background' tab as shown in the Figure 6.4a, these attributes has to be set to the following:

- Background: Color
- Color: FAFABFFF

Where else, through the 'Settings' tab, the name of the current slide along with the respective thumbnail can be inserted.

Figure 6.4: Slide settings



## 6.5 Adding Home Button


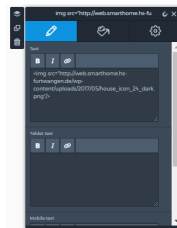
Home button has been added to all slides manually. The following steps explain how it is done in detail. First the 'Text layer' has to be added to the slide by clicking the  icon that can be found to the right of the slide view. This will bring up a widget (refer Figure 6.5, where the following has to be set).

Figure 6.5: Text layer widget

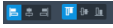


Through the first tab with icon  in the 'Text' field, the following HTML codes has to be entered:

```

```

Through the third tab with icon , following has to be set:

- Align: 
- Position X: 25 px
- Position Y: 25 px
- Width: Auto
- CSS class: home-button cursor-pointer

Next, through the **Pages** menu, 'Edit' option has to be clicked to open the 'Info Terminal' page, which has been created in the Section 6.3.

Under the 'Scripts n Styles' section, the following CSS codes has to be added under the 'Styles' tab:

```
.cursor-pointer {
  cursor: pointer;
  text-decoration: none;
}
```

After that, under the 'Scripts' tab, the following JavaScript codes has been added:

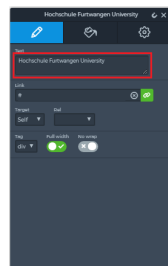
```
window.n2ss.ready(1, function(slider){
  jQuery( '.home-button' ).click(function(){
    window.location.href="/";
  });
});
```

These JavaScript codes give the callback functionality to the home button found on every slides. When, it is clicked, the browser takes the user to the website's home page.

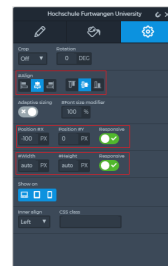
## 6.6 Adding Header

Header elements can be added into the slide by clicking on **H** icon which can be found on the left side of the slide view. This will bring up a widget where the header text can be entered as shown in Figure 6.6a. By click the **⚙** on the widget, the alignment, position and size (width, height) of the header element can be set, shown in Figure 6.6b.

(a) Adding header



(b) Header element settings



## 6.7 Adding Images


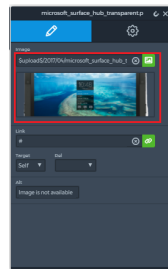
Images can be added to slide by clicking the  icon on left of the slide viewer. Clicking this icon will bring up the media selector window where an image has to be selected. After an image had been selected, the image widget can be seen with the selected image as shown in Figure 6.7. Here also the gear icon can be selected to apply further settings to the image such as positioning, alignment and sizing.

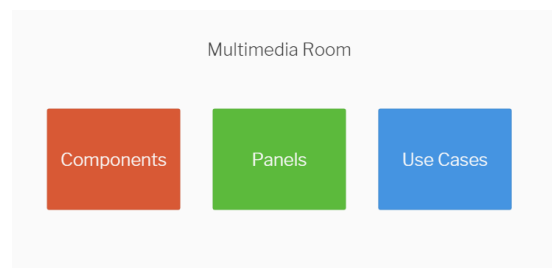
Figure 6.7: Image widget



## 6.8 Creating Overview Buttons

Each spaces such as (living room, kitchen, washroom, workspace and multimedia room) will be presented with overview buttons as shown in Figure ???. This section will explain on how to create an overview button.


Figure 6.8: Overview buttons



### Step 1: Adding Header Element

An header element has to be added to slide as discussed in the Section 6.6. Here the text should be given either 'Components', 'Panels' or 'Use Cases'.

## Step 2: Header Designing

After adding a appropriate text, the header element has to be designed clicking the . Clicking this, will show the CSS designing panel as shown in the Figure 6.9. Here, the following design settings have to be changed:


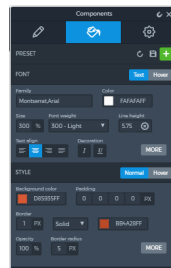

- Color: FAFABAFF
- Line height: 5.75
- Text align: 
- Background color: D85935FF (eg. for 'Components' button)
- Border: 1 px Solid BB4A28FF (eg. for 'Components' button)
- Border radius: 5 px

Figure 6.9: Designing header element



subsection\*Step 3: Header Settings Next, the header element need to be set. The setting screen can be opened by clicking the . Here besides alignment, positioning and sizing, the CSS class has to be set, which is important. In the CSS class text field as shown in Figure 6.10, the following text has to be entered:

```
| card-4 living-room-components
```

The `card-4` is constant for all overview button. The `living-room-` is depends on for which space the button are being added. For an example, for *kitchen*, this part has to be replaced with `kitchen-`. The last part of is `components`. This part has to replaced accordingly. For an example, for *panels*, this part should be replaced with `panels` and for *use cases* with `use-cases`.


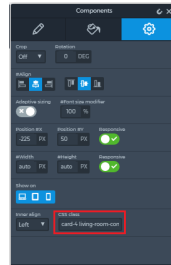
Next by toggling to *Hover* section by clicking  icon, the following entries have to be given:

Figure 6.10: Adding CSS Class



- Background color: BB4A28FF
- Padding: 0 0 0 0 px
- Border: 1 px Solid BB4A28FF
- Border radius: 5 px

### Step 3: Adding CSS

Next, click the **Pages** tab. Then edit the 'Info-Terminal' page. Under the 'Style' tab in the 'Scripts n Styles' section, few lines of CSS codes have to be added for additional designing of the overview button as listed below:

```
.card-4 {  
    min-width: 19\%;  
    min-height: 30\%;  
    border-radius: 5px;  
    box-shadow: 0 14px 28px rgba(0,0,0,0.25), 0 10px 10px  
        rgba(0,0,0,0.22);  
    cursor: pointer;  
}  
  
.card-4 div {  
    position: absolute;  
    min-width: 100\%;  
    min-height: 100\%;  
}
```

### Step 4: Adding JavaScript Callback

Lastly, a JavaScript callback has to be added to button, so that when the button is clicked, the slider takes the user to to respective slide. This can be

done by adding the following code under the *Script* tab under the *Scripts n Styles* section.


```
window.n2ss.ready(1, function(slider){
    jQuery('.living-room-components').click(function() {
        slider.slide(10);
    });
});
```

Above is an example of JavaScript callback implemented by using jQuery `.click` function. Here, the `'.living-room-components'` is the text entered in the CSS Class text field as discussed in the Step 3. The line `slider.slide(10)` is the execution telling the slider to move to slide number 10. Here, the destination slide is at index 10 starting with the first slide at index 0.


## 6.9 Creating Component Buttons

This section will discuss how to create the component buttons as shown in the Figure ???. The component buttons shown in figure belongs to the one that can be found in the living room. In this case, there are only two components in the living room, namely *Surface Hub* and *BenQ Smart Projector*. In order to create this buttons, there are few steps to be taken which will be discussed here.

### Step 1: Adding header element


Header element can be added to the slider by clicking the . Then, the corresponding text has to be added to the *Text* field which can be found on the widget first tab.

### Step 2: Designing the header element

Next, the header element has to be designed. To proceed with designing, click on the . In this section of the widget as shown in the Figure 6.11, following entries have to be changed:

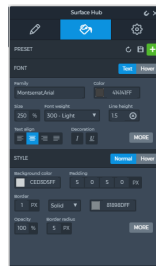
- Color: 414141FF
- Line height: 1.5
- Background color: CED3D5FF

- Padding: 5 0 5 0 px
- Border: 1 px solid 81898DFF
- Border radius: 5px


Next by toggling to *Hover* section by clicking  icon, the following entries have to be given:

- Background color: BDC1C3FF
- Padding: 5 0 5 0 px
- Border: 1 px Solid 81898DFF
- Border radius: 5 px

Figure 6.11: Designing component buttons



### Step 3: Setting Header Element


The header element has to be set with following settings. The setting panel of the header element can be opened by clicking the . Here, the following entries has to be set:

- Align:
- Width: 1000 px
- Height: auto
- CSS claa: card-1 components-surface-hub

CSS Class with text `card-1 components-surface-hub` is very important. The `card-1` is the reference to CSS styling which will be discussed in the next section. Whereby `components-surface-hub` is the reference for the JavaScript callback function. In this case, it refers to the destination slide which contains the component Surface Hub.



## Adding CSS Styling

To add the CSS styling for the component buttons, navigate to . Edit the *Info Terminal*. Under the *Styles* tab under *Scripts n Styles* section, the following CSS codes have to be added:

```
.card-1 {  
  border-radius: 5px;  
  box-shadow: 0 5px 10px rgba(0,0,0,0.19) , 0 6px 6px  
    rgba(0,0,0,0.23);  
  cursor: pointer;  
}
```

## Adding JavaScript Callback

The JavaScript callback is a small codes that will get executed when the user click on the component buttons. This codes will be added to the *Scripts* tab under the *Scripts n Styles* section. Here, the following codes have to be added:

```
window.n2ss.ready(1, function(slider){  
  jQuery( '.components-surface-hub' ).click(function(){  
    slider.slide(11);  
  });  
});
```

The example callback function above contains the reference '`.components-surface-hub`'. When the button with this reference get clicked, the `.click()` function will be executed which will slide the slider to to slide with index 10. The slide number 10 should contain, in this case, the *Surface Hub*. The index counting starts from 0.

## 6.10 Creating Panel Buttons

This section will explain on how to create a panel button as shown in the Figure 6.12. Panel buttons are buttons that have been used as navigation to items that can be found on the panels in the lab. These buttons have been created with images of item itself.

### Step 1: Adding Text Layer


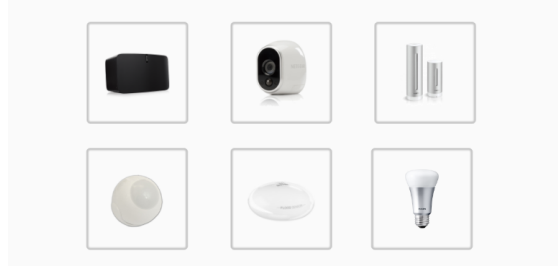

In order to add or create a panel button, a text layer has to be added by clicking  icon that can be found on left of the slide viewer. This is bring


Figure 6.12: Panel buttons



up the text layer widget as shown in the Figure ???. In the *Text* field which can be found under , the following HTML codes have to be added. This code will result in the following button containing the Sonos Play:5 speaker as the button image.

```
<div class="panel-nav-container panels-sonos-play5">
  <span class="span-helper"></span>
  
</div>
```


## Step 2: Setting of Text Layer

Setting of the text layer can be accessed by clicking the . Here, only two settings has to be changed as listed below:

- Width: 200 px
- Height: 200 px

Apart from the width and height, the positioning of the button in X and Y direction has to be set accordingly as can be seen in Figure 6.12.

## Step 3: Adding CSS Styling

To style the button, CSS styling has to applied. Edit the *Info Terminal* page which can be found under . Under the *Styles* tab under *Scripts n Styles* section, the following CSS code has to be added:

```
.panel-nav-container {
  background-color: #fff;
  text-align: center;
  width: 100%;
```

```

    height: 100%;
    border-radius: 10px;
    border: 1px solid #aaa;
    box-shadow: 0 0 5px 2px rgba(0, 0, 0, 0.2) inset;
    cursor: pointer;
    position: absolute;
}

.panel-nav-container:hover {
    border: 1px solid #008855;
    box-shadow: 0 0 5px 2px rgba(0, 136, 85, 0.2) inset;
}

.span-helper {
    display: inline-block;
    height: 100%;
    vertical-align: middle;
}

.panel-nav-container img {
    transform: scale(1);
    transition: transform 0.5s;
    transition-timing-function: ease;
}

.panel-nav-container:hover img {
    transform: scale(1.1);
}

```

This CSS code will style the panel button and the image of button as well. The CSS with *:hover* tags add the hover effect to buttons.

## Step 4: Adding JavaScript Callback

Lastly, the JavaScript callback function has to be added to the button when the user click on the buttons. The JavaScript callback has to be added under the *Scripts* tab under the *Scripts n Styles* section of the *Info Terminal* page.

```

window.n2ss.ready(1, function(slider){
    jQuery('.panels-sonos-play5-2').click(function(){
        slider.slide(55);
    });
});

```



## 6.11 Creating Use-Cases Buttons

Use-cases buttons are the buttons used to navigate user to the use-cases that can be found in the lab. A use-case button contains an image and a short descriptive text below the text as shown in Figure 6.13

Figure 6.13: Use-cases buttons



### Step 1: Adding Text Layer


Begin by adding the text layer to the slide by clicking the . In the *Text* field under the  the following HTML codes have to be entered:

```

<div class="use-cases-desc">
  <p>Window Ligthing</p>
</div>
```

The above code is an example of use-cases button for *Window Lighting*.

### Step 2: Adding CSS Class

After adding the HTML codes to the *Text* field, class text has to be added to the *CSS class* field which can be found under the . Here the following CSS class has to be entered:

```
use-cases-container use-cases-window-lighting
```

The `use-cases-container` will be used as a reference for CSS styling. And the `use-cases-window-lighting`, in this case as an example for window lighting use-cases button, will be used as CSS class reference for the JavaScript callback.

### Step 3: Adding CSS Styling

The use-cases buttons have been styled by using custom CSS code. The CSS code has been entered under the *Styles* tab under the *Scripts n Styles* section of the *Info Terminal* page. The following CSS code has been added for styling:

```
div.use-cases-container {
  max-width: 20%;
  background-color: white;
  box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2);
  cursor: pointer;
  transition: box-shadow 0.5s;
  transition-timing-function: ease;
}

div.use-cases-desc {
  border-top: 1px solid #aaa;
  text-align: center;
  padding: 3.5% 5.5%;
}

div.use-cases-container:hover {
  box-shadow: 0 10px 20px 5px rgba(0, 0, 0, 0.2);
}
```

### Step 4: Adding JavaScript Callback

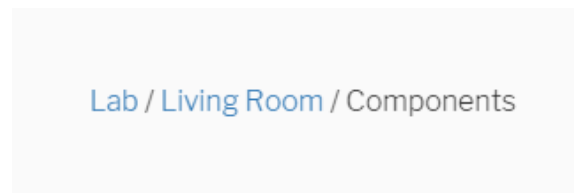
Lastly, JavaScript callback has to be programmed so that the slider will navigate to the destination slide when the user click on the use-cases buttons. The JavaScript callback script has to be added under the *Scripts* tab under the *Scripts n Styles*.

```
window.n2ss.ready(1, function(slider){
  jQuery('.use-cases-window-ligthing').click(function(){
    slider.slide(62);
  });
  jQuery('.use-cases-rainbow-ligthing').click(function(){
    slider.slide(63);
  });
});
```


## 6.12 Creating Breadcrumb

Breadcrumb, as shown in the Figure 6.14, is the navigation bar that can be found on the slides, where user can navigate from the current level to the upper hierarchy level.

Figure 6.14: Breadcrumb



### Step 1: Adding Text Layer

Start creating a breadcrumb by adding a text layer to the slide by clicking  icon

```
<span class="breadcrumb_link breadcrumb-lab">Lab</span> / <span  
  class="breadcrumb_link lab-multimedia-room">Multimedia  
  Room</span> / Use Cases
```

### Step 2: Adding CSS Styling

Next, CSS styling has to be applied. The CSS code has to be entered under the *Styles* tab under the *Scripts n Styles* section in the Info Terminal page.

```
.breadcrumb_link {  
  color: #337ab7;  
  cursor: pointer;  
}  
  
.breadcrumb_link:hover {  
  filter: brightness(70%);  
  text-decoration: underline;  
}
```

### Step 3: Adding JavaScript Callback

Lastly, JavaScript callback script has to be added so that when the user clicks on the breadcrumb link, the slider will take user to the destination slide.

```
window.n2ss.ready(1, function(slider){  
  jQuery( '.breadcrumb-lab ' ).click(function(){  
    slider.slide(4);  
  });  
});
```

# Chapter 7

## CCTV Live Stream

### 7.1 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 as shown in the Figure 7.1. 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 7.1: 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.



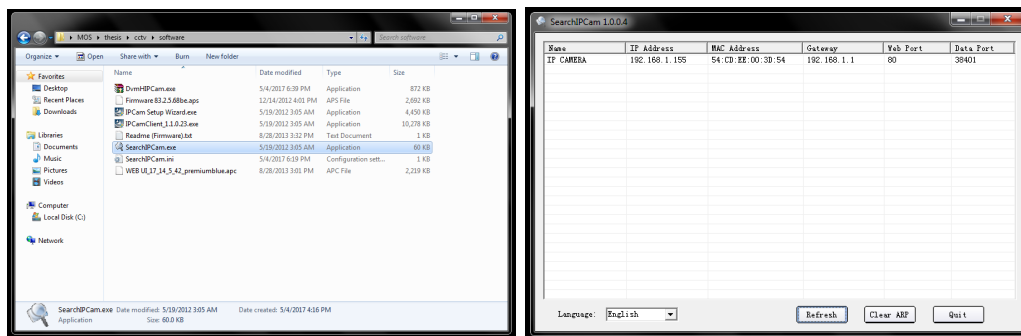
## 7.2 Pre-Configuration

### 7.2.1 Identifying IP Address

*The steps discussed here is part of configuration steps from the camera vendor, which can be found in the following documentation<sup>1</sup>.*

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 7.2). This application can also be downloaded from the vendor's website<sup>2</sup>. Run the program and hit the 'Refresh' button to obtain the IP address of the camera.

Figure 7.2: SearchIPCam.exe



### 7.2.2 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.

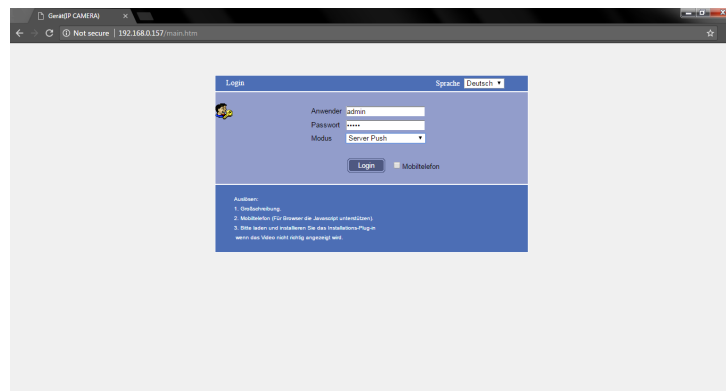
- Anwender: admin
- Passwort: admin
- Modus: Server Push

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

<sup>2</sup><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 7.3: Login in into IP-Camera web interface



### 7.2.3 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 7.4: IP-Camera WLAN settings

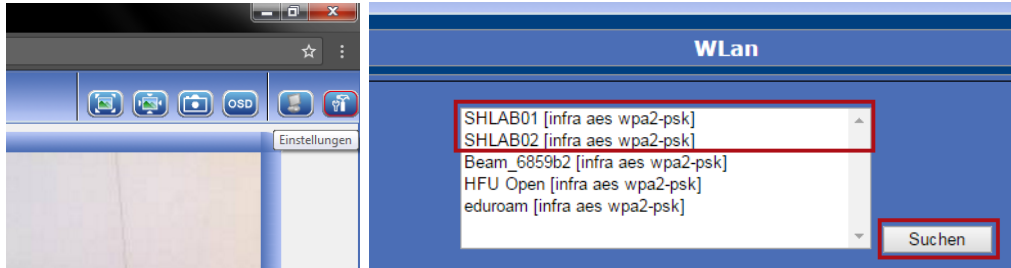


Figure 7.5: WLAN Credentials

WLAN wird verwendet	<input checked="" type="checkbox"/>
SSID	SHLAB01
Netzwerk Typ	Infra
Sicherer Modus	AES
Verschlüsselung	WPA2-PSK
Key	*****

Update Speichern

## 7.2.4 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 7.2.1 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.

## 7.3 Identifying the cgi-bin Module

### 7.3.1 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*.

### 7.3.2 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).

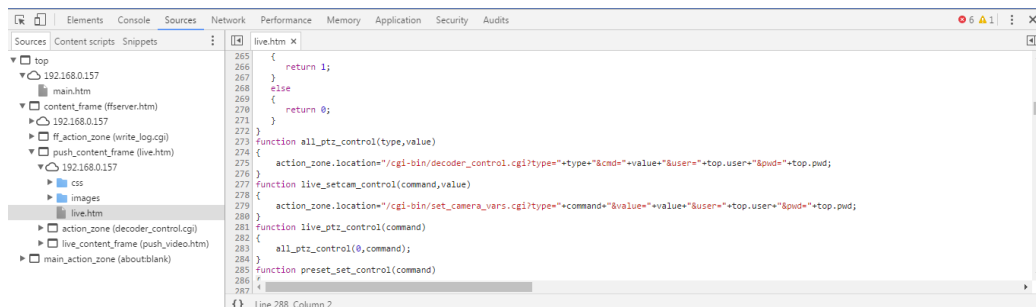
The the HTML and JavaScript codes of the IP-Camera web interface, however, can't be studied easily because there are a lot of files which are embedded upon one another and referred to and from one file to another. The method used to extract the required URI and query parameters are through backtracking 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 7.6: 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 7.6).

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 7.1*
- username: admin
- password: admin

Table 7.1: 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

## 7.4 Designing the Web Page

### 7.4.1 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, Word-Press, 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.

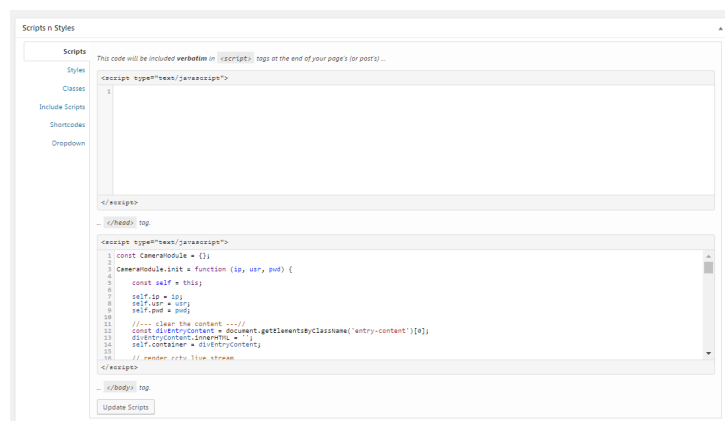
## 7.4.2 Creating the Web Page

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.

## 7.4.3 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 7.7). 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 footnote<sup>3</sup>. 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 7.7: Scripts n Styles Column



<sup>3</sup><https://de.wordpress.org/plugins/scripts-n-styles/>

### 7.4.4 Self-Invoked JavaScript Function

First, a self-invoked JavaScript function will be added into 'Scripts' lower column as shown in the Figure 7.7. 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 7.8: 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 page, there is 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
    ...
    ...
})();
```

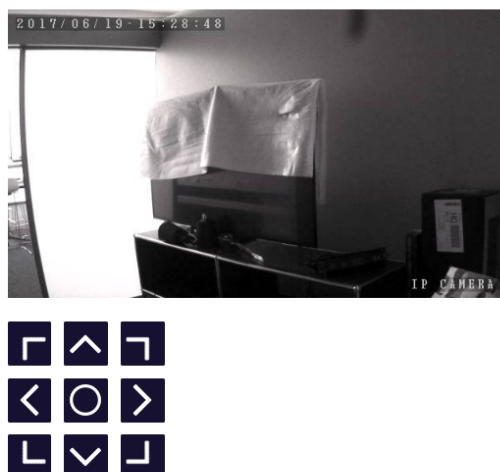
### 7.4.5 Rendering CCTV Footage and Controller

After the form as shown in Figure 7.8 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 7.9: 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 7.9.

### 7.4.6 Styling and Positioning the Buttons

The controller buttons that are rendered through JavaScript alone do not appear as shown in Figure 7.9. 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 CSS<sup>4</sup>. 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 */
```

---

<sup>4</sup><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 */

```

### 7.4.7 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 7.4.5. 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 7.3.2.

## Chapter 8

# 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.

### 8.1 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 8.1. 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]

```

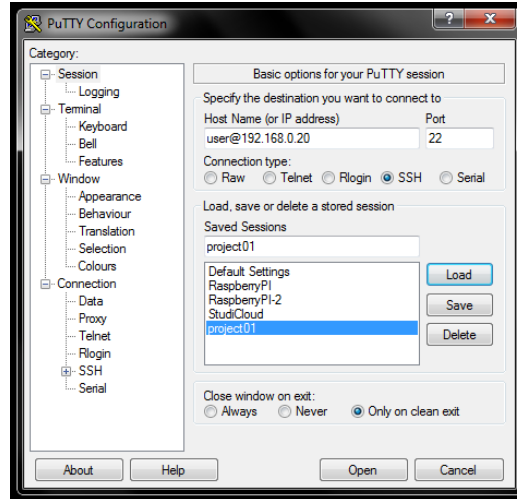
Since the directories and files are uploaded through the system user of Ubuntu Server, by default WordPress don't have permission to access them. File ownership and file permission for those directories and files has to set using `chown` and `chmod`.

```

| 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 8.1: 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.

## 8.2 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 8.1.

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;
}

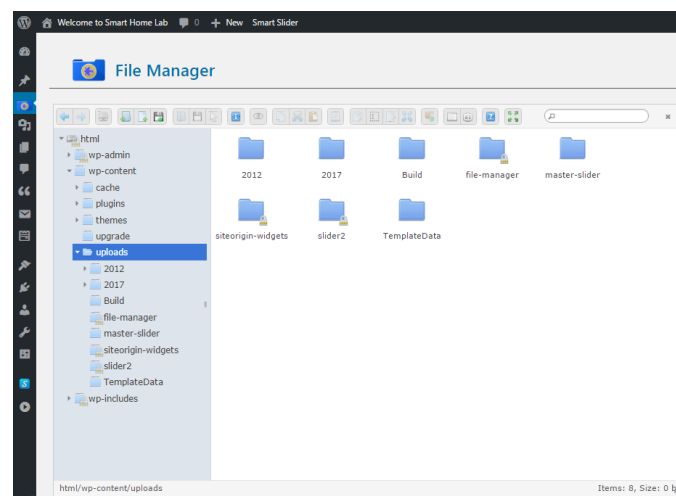
```



## 8.3 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' plugin<sup>1</sup>. This plugin is available free of cost in the WordPress plugins directory<sup>2</sup>. This plugin can be added to the website and activate through the web browser.

Figure 8.2: 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 8.2).

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 8.1, a problem has occurred. There is a file with size of almost 12

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

<sup>2</sup><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.