



Home Control

Final Engineering Project

Title: Home Control

Number: ITK03

Class: 8A

Year: 2018/19

Author: Elias KARRÉ

Supervisor: Martin BERGER

Table of Contents

1	OBJECTIVE	4
2	FOREWORD	5
2.1	Motivation	5
2.2	Acknowledgments.....	5
3	INTRODUCTION	6
4	TECHNICAL BASICS	8
4.1.1	Hardware	8
4.1.2	Software	9
4.2	NodeMCU ESP8266	9
4.2.1	Hardware	10
4.2.2	Software	10
4.3	433 mHz radio modules.....	12
4.4	Lampholders and sockets.....	13
5	PROTOCOL	14
5.1	NodeMCU ESP8266 configuration	14
5.1.1	Arduino IDE installation.....	14
5.1.2	Radio Recevier.....	15
5.1.3	Webserver	17
5.2	Raspberry Pi 3 configuration.....	18
5.2.1	Raspbian installation	18
5.2.2	Webserver	19
5.3	MySQL database.....	21
5.3.1	Installation	21
5.3.2	Database structure	22
5.3.3	Insert Data Records	23
5.4	Webapplication.....	24
5.4.1	HTML structure	25

5.4.2	Responsive CSS	25
5.4.3	Dynamic Structure PHP	26
5.4.4	JavaScript	27
5.4.5	request.php	29
5.5	Potential Problems	30
5.5.1	Network problems	30
5.5.2	Radio range	30
5.5.3	Linux operating system	30
5.5.4	Hardware damage	30
5.6	Time table	31
5.7	Additional work	32
5.7.1	State of lamps	32
5.7.2	User Profile	32
6	COST CALCULATION	33
7	CONCLUSION	34

1 Objective

In a household, there are many small handles, such as switching on a lamp or a power distribution, which are usually dispersed throughout the apartment. This project centralizes all these physical switches in one web application.

The web application should be compatible with various smartphones as a host device. The graphical layout and position of the menu bar and buttons of the website adapts to the screen resolution of the host device. Thus, the user interface looks the same for all application devices and there are no compatibility issues due to a low / high display resolution of a smartphone or tablet.

The user interface provides the user with an area for each switching option of a device registered in the system, in which he can assign the device the desired status. There are two actions available for each item; the switch-on and switch-off command. Tapping on one of the action buttons a 433-mHz-radio signal is transmitted to the corresponding device by a microcontroller equipped with a radio module. Depending on the radio signal, the radio devices can react with two actions: the device is switched off or switched on.

A new lamp or socket can be added through a database entry. For each device name, there are two entries in the database table; the signal for a switch-on and switch-off action.

The web application differentiates in the menu navigation between the control of a light and that of a device. The individual menu tabs can be scrolled through on a smartphone with finger wipes or by tapping on a menu item.

To save several operations, several devices, such as all garden or dining table lights, can be assigned to a group. Thus, in addition to the individual control options, the user can operate the devices of an entire logical area of an apartment with a single touch.

The connection to the interface can be established at any time through a browser window of a smartphone, which is located in the same network. For access to the control of the lights and sockets of the apartment a network access in the form of a Wi-Fi password or a physical connection to the network is needed.

2 Foreword

2.1 Motivation

To find my engineering project, I asked myself how I could combine everyday things with the help of data processing. I quickly thought of the operation of light switches. Sometimes I wish there was a universal remote control so that I could switch lights and devices on and off without having to leave the couch. Or when you've just left the house and you're not sure if all the lights are switched off to save electricity.

Once I knew what I wanted to achieve with the project, I asked myself how I could implement it. I realized that there are an infinite number of ways to realize such a project. For the realization of the project, I decided on those technologies in which I was personally interested and already had a certain amount of prior knowledge. As we only studied programming languages such as PHP, JavaScript or SQL superficially in class, I first had to acquire a lot of knowledge about them myself.

2.2 Acknowledgments

My very special thanks go to:

- Prof. Berger, who supported me with help and advice during the development and completion of the project.
- Prof. Niederle, who taught me the ways of the terminal through his enthusiastic Linux lessons and guided me safely through one of the most popular operating systems of a computer.
- My parents, without whom my education would not have been possible.

3 Introduction

The project can be divided into 4 main chapters: Programming the web application, setting up the web server, preparing the database and configuring the microcontrollers.

I decided to start the project with the configuration of the NodeMCU ESP8266 microcontroller from AZDelivery. However, before this can be connected to a computer for transferring program code, the Arduino development environment must be installed and configured. After connecting a 433 MHz receiver module to the GPIO ports of the microcontroller, the radio code parameters of the individual radio-controlled lamp holders and radio-controlled sockets can be determined using the remote controls supplied and recorded in an Excel spreadsheet in the meantime.

Once the radio codes of each radio device have been determined, the radio receiver module can be removed again and replaced by a radio transmitter. To be able to respond to incoming http requests, a small web server must be installed on the microcontroller. This web server can extract radio parameters from an http request tailored to the project, which it can then send out via the radio transmitter. In this way, it is possible for every network-compatible device in the same network to command the microcontroller to send out a certain radio signal using a curl function.

The second main chapter of the project will deal with the configuration of the Raspberry Pi server. First, the Raspbian operating system optimized for the Pi must be installed on its MicroSDHC card. An Apache2 server is then installed as a web server, MySQL as a database system and a Php interpreter on the Linux distribution using a terminal command.

Once all the settings have been made in the Raspberry Pi operating system, the database structure for the transmission codes can be created using the phpMyAdmin tool. The project's database only requires one table called "codes". One row of the table contains information about the name of the device, the radio parameters (protocol number, pulse length and pulse value), the action that triggers the signal (on or off) and the category of the device (either a lamp or socket). The signal codes from the previously created Excel table can now be entered into the database. There are two entries for each device name: one with the "on" action and one with the "off" action.

When the database is ready for retrieval, the last main chapter of the project can begin: The programming of the web application. As the website should be responsive, the CSS file does not contain absolute values, but values relative to the viewport. This means that the menu is displayed in the same proportions on smartphones with different screen resolutions.

The menu is programmed using JavaScript. With normal JavaScript code and jQuery, the website is divided into menu tabs. These tabs can be scrolled through on a smartphone with the help of a script using a swipe gesture. A tap on the respective menu item is also sufficient to switch to the content of this tab.

As the web application is to be built dynamically depending on the database entries, the code is in a php file. Using Php as a scripting language, database entries can be entered and displayed as an element on

the website. The term element refers to the button of a radio device. The button of a radio offers the user two actions: Switching the device on and off.

Abbreviations used

curl	Client for URLs
http	Hypertext Transfer Protocol Transmission
mHz	Megahertz
NOOBS	New Out Of the Box Software
SQL	Structured Query Language
SDRAM	Synchronous Dynamic Random Access Memory
GPIO	General Purpose Input Output
GB	Gigabyte
GND	Grounding
HTML	Hyper Text Markup Language
LAN	Local Area Network
USB	Universal Serial Bus
HDMI	High Definition Multimedia Interface
CSS	cascading style sheet

4 Technical Basics

For the practical implementation of the project, a server with a Linux distribution as operating system and a microcontroller that is compatible with a 433MHz transceiver module are required as hardware. Furthermore, lamp sockets and socket plugs, which can be controlled by a 433 MHz radio signal, are used to wirelessly manage the power supply of light bulbs and devices.

The choice of Linux-compatible server for the “Home Control” project fell on the Raspberry Pi 3 B+. This single-board computer is very inexpensive at just under €35 and is also very small and portable. Nevertheless, with a 1.4 GHz processor, 1 GB DDR2 SDRAM and a Linux distribution as a platform, the device fulfills the minimum requirements of a server for this project.

4.1.1 Hardware

The Raspberry Pi 3 offers many hardware interfaces for an inexpensive single-board computer. For my project, I need all the connections for hardware components shown in the colorful frames in Figure 1.

The GPIO connections allow the mini fan supplied with the Raspberry Pi 3 housing to be operated with a voltage of 3.3 volts. The MicroUSB port operates the entire current flow of the Raspberry Pi with a voltage of 5 volts. There is also an HDMI port for image output and a LAN port to connect the Pi to the network. The manufacturer has also installed four USB-A 2.0 ports to enable operation with a keyboard and mouse.

There is also a microSDHC card reader on the underside of the device, in which a SanDisk Ultra 32GB microSDHC card is installed. 32 gigabytes are more than enough for this project to install all the necessary programs and the Raspbian operating system.

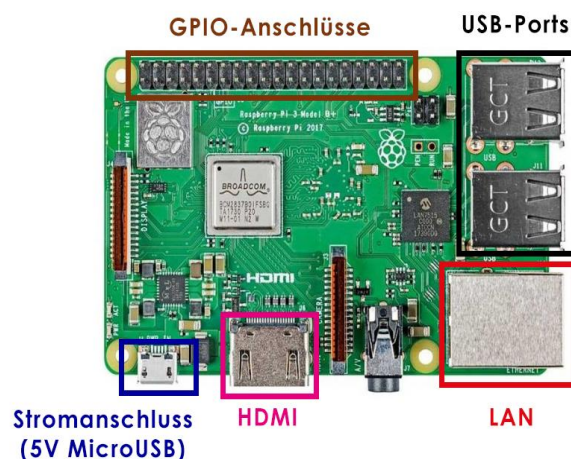


Figure 1: Raspberry Pi 3 Interfaces

4.1.2 Software

4.1.2.1 Win32-Disk-Imager

Win32 Disk Imager is a free software program that can be used to write an image file to an SD card. This program is required in this project as the operating system required for the Raspberry Pi 3 is installed using an image file.

4.1.2.2 Raspbian

Raspbian is a Linux Debian distribution specially optimized for the hardware of the Raspberry Pi. This operating system comes pre-installed with many programs and services relevant for programming a Pi, such as Python or drivers for the Bluetooth and WLAN module installed on the single-board computer. Raspbian is therefore the only logical choice for the server's operating system. Raspbian is free software and was not programmed by the official Raspberry Pi Foundation, but is offered for download on raspberrypi.org as an "officially supported operating system". The operating system can be installed either by NOOBS, a Raspbian installer, or by an image file on the Pi's SD card. (Source 5, 2019)

4.1.2.3 Apache2 Webserver

Apache is a free HTTP server software for Windows and UNIX. The software offers several http services with which the web server of the project is realized. The service can be installed with a simple console command and also runs automatically in the background each time the system is restarted. Apache creates the directory /var/www/http/ during installation. The files stored in the http folder, such as a php script, are located in the root directory of the web server. The content of the server's root directory can be displayed in a web browser using the http protocol. The IP address of the web server is that of the Raspberry Pi's network card and runs on well-known port 80 by default. (Source 6, 2019)

4.1.2.4 MySQL / phpMyAdmin

MySQL is one of the most widely used database management systems, which is based on the SQL (Structured Query Language) programming language. The system is subject to an open source license and runs on most major platforms such as Linux and Windows. The databases created are stored on an Apache web server and can be manipulated using a php script. The "Home Control" project requires such a form of data storage in order to assign all the various 433 MHz signal codes and properties to a radio device in tabular form. As the web application is built up dynamically using a php script based on the database entries, a new device can be added very easily by means of a new database entry. The database is administered using the phpMyAdmin tool. With this tool, the entire database structure can be created and data records added using a very clear GUI. (Source 7, 2019)

4.2 NodeMCU ESP8266

The NodeMCU ESP8266 is responsible for radio communication with the lamp holders and sockets. A radio module connected to the GPIO connections of the ESP8266 enables it to emit a radio signal in the

433MHz range, to which the radio receivers installed in the radio devices respond. The microcontroller is programmed via the Arduino IDE using the C++ programming language and an imported library (rcswitch), which provides a tool set for controlling the radio module.

The choice for the Mico controller for this project fell on the NodeMCU ESP8266 Lua Amica from AZDelivery, as it is much cheaper than its competitor the Raspberry Pi Zero at just under 5€ and is compatible with the rcswitch library. Furthermore, it even has wireless network capability thanks to a WLAN network card.

4.2.1 Hardware

The ESP8266 offers 30 GPIO connections as physical interfaces and a MicroUSB for the power connection and firmware data transfer. The manufacturer also includes a flash and reset button for complete configuration of the microcontroller.

The radio transceivers can be physically connected to the ESP8266 via the power and data transfer pins of the GPIOs. The type of interface is labeled on the side of the pins on the board. The “D” pins are for data transmission to the transmitter module, 3v3 for the 3.3 volt power supply and GND for the ground pole.

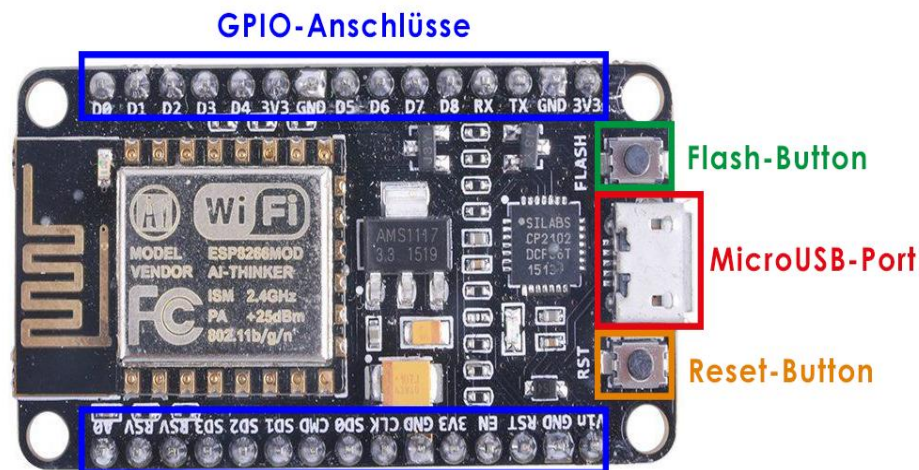


Figure 2: ESP8266 Interfaces

4.2.2 Software

4.2.2.1 Arduino IDE

The free Arduino IDE software enables the ESP8266 to be programmed using the C++ programming language. The program supports a variety of microcontrollers and offers the option of integrating some of Arduino's pre-installed libraries.

The operation of the ESP8266 can be configured in the “Tools” tab of the software. First, it must be connected to a computer running the program via the MicroUSB interface built into the board. Figure 3 highlights the most important setting options for the project.

The serial monitor is practically a debug output console for programming the Arduino, which provides an output via the Serial.print() function. In this way, the value of a variable, for example, can be determined without further effort.

To be able to upload program code to the connected microcontroller, the correct board, upload speed and COM port must be selected.

In the library management there are many program libraries of sensors and modules that can be integrated. These libraries provide tools in the form of functions that can be used to program transceivers and sensors.

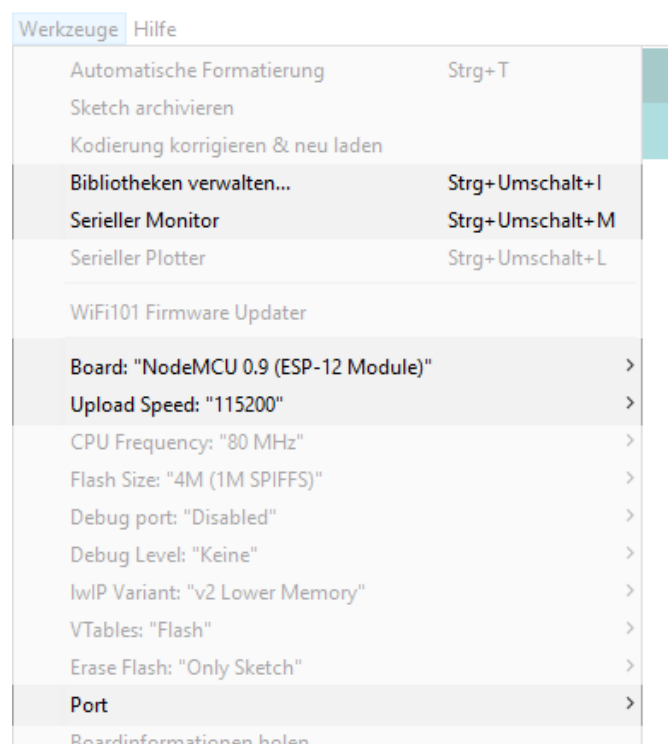


Figure 3: Arduino IDE Tools

4.2.2.2 rcswitch Library

In order for the ESP8266 to work with the 433 MHz radio modules, the rcswitch library must be integrated into the Arduino IDE. It provides the programmer with a set of functions for controlling the radio transmission of the module.

A radio signal contains 3 different parameters: the number of the protocol, the pulse length and the pulse value. With the following functions contained in the rcswitch library, the ESP8266 can emit a valid radio signal if it is connected to a radio transmitter:

```
mySwitch.setProtocol(<protocol_number>);
mySwitch.setPulseLength(<pulse_length>);
mySwitch.send(<pulse_value>, 24);
```

The mySwitch variable with the RCSwitch class is first configured with the correct protocol number and pulse length and then transmits a 24-bit radio value. If the microcontroller is connected to a radio receiver, such a signal can be recognized by the following command lines:

```
Serial.print( mySwitch.getReceivedProtocol() );
Serial.print( mySwitch.getReceivedBitlength() );
Serial.print( mySwitch.getReceivedValue() );
```

This command block recognizes the parameters of an incoming radio signal from a radio receiver and outputs them in the serial monitor of the Arduino IDE, as shown in Figure 4. The radio value is output as a decimal and binary value, the pulse length in microseconds and the protocol as a number. Receiving signals is just as important for the project as sending them, as it allows you to test which radio device reacts to which radio signal value, pulse length and protocol number.

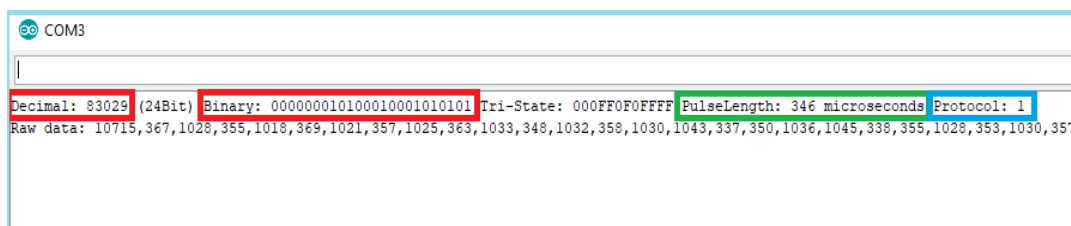


Figure 4: serial monitor output for radio signal receiver

4.3 433 mHz radio modules

The radio transmitters and receivers used for the “Home Control” project, manufactured by Anpro, work with signals in the 433 MHz frequency range. They are connected to an ESP8266 via its GPIO connections, namely a 3.3 volt voltage pin, a ground pin and a data transmission pin, and can thus communicate with it via the rcswitch library. The manufacturer has also installed an extendable wire antenna to increase the signal range. The manufacturer promises a radio range of at least 30 meters at an operating voltage of 3.3 volts.

Figure 5 shows the receiver module, which reacts to incoming radio radiation in the 433 MHz range. This module is only needed at the beginning of the project to determine the radio codes of the lamp holders and sockets, but also to add further devices to the system later.

The radio transmitter shown in Figure 6 is probably the most important part of the whole project; it is responsible for emitting the parameters set by the ESP8266 as a radio signal to cause a switching action on a radio lamp socket or socket outlet.



Figure 5: receiver module



Figure 6: transmitter module

4.4 Lampholders and sockets

Sockets and plugs are connected between the power source and the appliance in order to be able to manipulate the current flow of each light bulb or appliance in a home without having to intervene in the 230 volt electrical circuit. In the case of light bulbs, the lamp sockets are screwed between the power supply and the lamp and in the case of sockets, the appliance to be controlled is not plugged directly into the socket but into an intermediate socket.

As shown in Figures 7 and 8, these control units are supplied with a remote control. With this remote control and a radio receiver, it is possible to find out which radio signal parameters which radio device reacts to, so that these can then be registered in a database together with their properties. The different transmission codes are randomly assigned to the products by the manufacturer and do not overlap even if several identical sets are purchased. Each device has its own individual frequencies. The maximum

operating power recommended by the manufacturer for the radio-controlled sockets is 1100 watts and for the radio-controlled lamp holders 60 watts, depending on the holder.



Figure 7: lampholders and remote control



Figure 8: radio sockets and remote control

5 Protocol

5.1 NodeMCU ESP8266 configuration

5.1.1 Arduino IDE installation

In order to be able to load the NodeMCU ESP8266, the Arduino development environment must first be installed on a computer. This is done using an installer, the download link for which can be found on the official Arduino website. The installer installs the development program and some USB drivers in order to be able to communicate with the Arduino via a serial interface.

Once the installation is complete, a board URL for the ESP8266 must be specified in the preferences, as shown in Figure 9, as the program does not support this Arduino model by default.

`http://arduino.esp8266.com/stable/package_esp8266com_index.json`

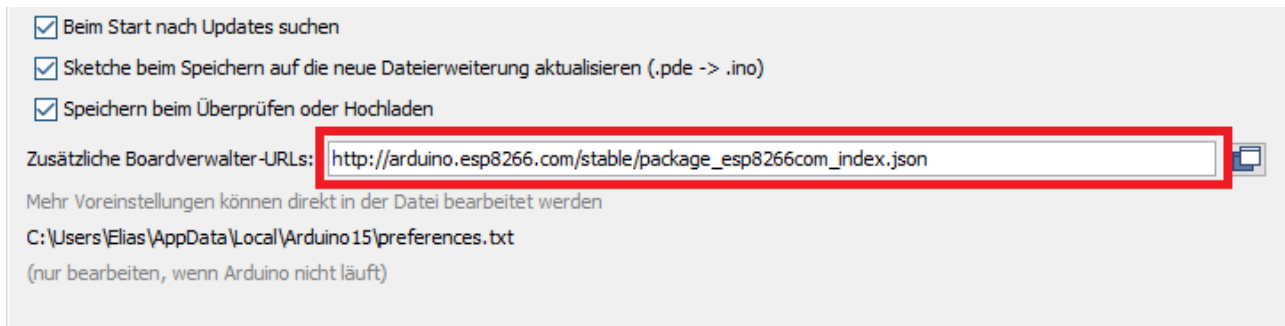


Figure 9: Arduino IDE preferences

After restarting the program, the board management can be opened in the “Tools” -> “Board: Arduino/Genuino Uno” tab. The ESP8266 software can be found and installed using the search field. After installation, “NodeMCU 0.9” must be selected in the ‘Tools’ tab under “Board:”.

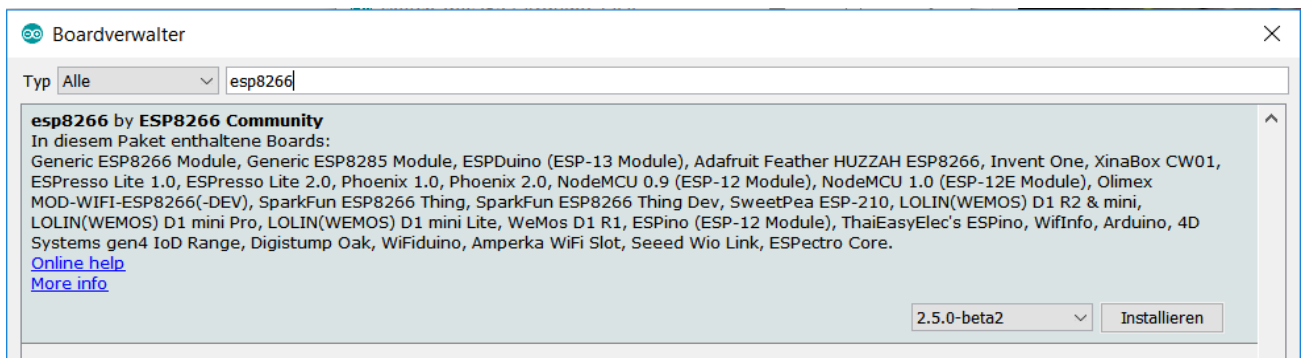


Figure 10: board management

The latest version of the rcswitch library must now be integrated into the program. The library management can be called up in the “Tools” tab. A window similar to that shown in Figure 10 opens in which the rcswitch library can be searched for and installed.

The NodeMCU ESP8266 can now be installed on a breadboard and connected to the computer via the MicroUSB port on the board. A COM port should now be automatically assigned to the device in the Tools tab. This completes the installation and configuration of the Arduino IDE. (Source 3, 2019)

5.1.2 Radio Receiver

To find out the radio codes of the individual lamp holders and sockets, a 433 MHz radio receiver module must first be connected to the microcontroller. This is done using jumper cables and the breadboard. The module and the ESP8266 are wired together as shown in Figure 11.

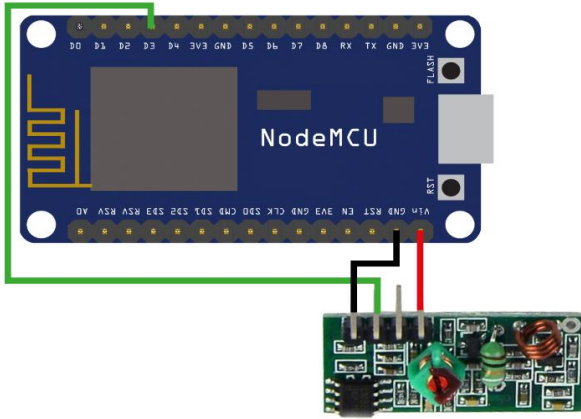


Figure 11: ESP8266 connected to radio receiver

To test whether the wireless module communicates correctly with the microcontroller, a program code can be loaded in the “File” -> ‘Examples’ -> “rc-switch” -> “ReceiveDemo_Advanced” tab. This small program detects incoming radio signals from the radio module, determines their parameters and outputs them in the serial monitor of the Arduino IDE.

The output of the serial monitor in Figure 4 shows a signal received by the radio receiver. The pulse value is marked in red in decimal and binary notation. Each radio device has two pulse values to which it reacts: one switches the device on, one switches it off. The pulse length for optimum transmission quality is marked in green and the number of the transmission protocol, which is usually 1, is marked in blue.

Using this script and the remote control supplied with the radio sockets and lamp sockets, the individual pulse values of each radio device can now be determined and what action they perform on it (on or off). As long as no database is available, these data records are noted in an Excel spreadsheet, as shown in Figure 12.

	A	B	C	D	E	F
1	device	protocol	pulse length	value (dec)	action	type
2						
3	505-1	1	345	83029	on	power
4	505-2	1	345	86101	on	power
5	505-3	1	345	70741	on	power
6	505-4	1	345	21589	on	power
7						
8	505-1	1	345	83028	off	power
9	505-2	1	345	86100	off	power
10	505-3	1	345	70740	off	power
11	505-4	1	345	21588	off	power

Figure 12: excel table radio codes

5.1.3 Webserver

A small web server is installed on the microcontroller to enable it to issue commands from outside. First, the ESP8266 must be connected to a network via its WLAN module. To do this, the following library is included and variables are filled:

```
#include <ESP8266WiFi.h>

#include <ESP8266WebServer.h>

ESP8266WebServer server(80);

const char* ssid = "<WLAN SSID>";

const char* password = "<WLAN Password>";
```

The libraries required for the network connection via WLAN are integrated and the port of the web server is set to 80. In the next step, a WLAN connection is established in the setup function of the program, which only runs automatically once when the program is started, the IP address of the microcontroller is displayed in the serial monitor, the “sendcode” function is defined for the server root directory “/send” and the web server is started:

```
WiFi.begin(ssid, password);

Serial.println(WiFi.localIP());

server.on("/send", sendcode);

server.begin();
```

As soon as an http request is sent to “192.168.0.20/send”, the NodeMCU ESP8266 executes the “sendcode()” function. A valid http link for this project looks like this:

```
192.168.0.20/send?protocol=1 & pulselength=163 & value=349955
```

The arguments “Protocol”, ‘Pulselength’ and “Value” are communicated to the microcontroller, which saves them in variables as follows:

```
String protocol = server.arg(0);

String pulselength = server.arg(1);

String value = server.arg(2);
```

Finally, the string variables are converted into an integer by the “toInt function” and used as parameters for radio transmission.

```
mySwitch.setProtocol(protocol.toInt());

mySwitch.setPulseLength(pulselength.toInt());

mySwitch.send(value.toInt(), 24);
```

A radio transmitter installed on the microcontroller as shown in Figure 13 sends the values received via the http request to all radio devices in the vicinity. The ESP8266 can now use its IP address to receive radio parameters from every device in the same network, which it then transmits via its 433 mHz radio transmitter module.

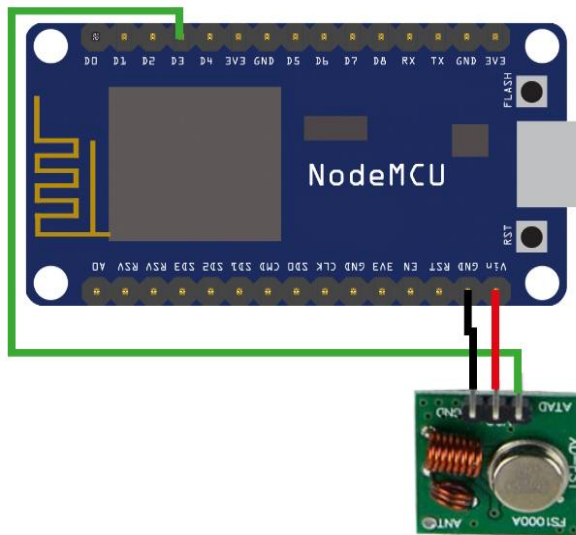


Figure 13: ESP8266 connected with radio transmitter

5.2 Raspberry Pi 3 configuration

5.2.1 Raspbian installation

The first step in configuring the Raspberry Pi 3 for this project is to install the Raspbian operating system on a MicroSDHC card. The card can be connected to a computer using a MicroSDHC to USB adapter. On the official Raspberry Pi website (raspberrypi.org), you can find the latest version of the Raspbian image with the pre-installed software packages recommended by the manufacturer. The downloaded zip file must now be unpacked using an archive management program such as 7-Zip or WinRAR.

Next, the Win32 Disk Imager program is installed and executed on the work computer. As shown in Figure 14, the location of the Raspbian image must be specified in the image file input field and the correct data carrier selected. By clicking on the Write button of the program, the MicroSDHC card is formatted and the content of the image file is saved to it. (Source 8, 2019)

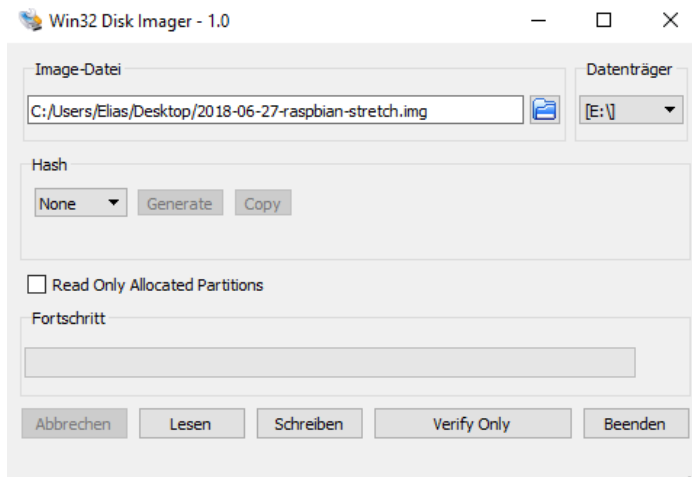


Figure 14: Win32-Disk-Imager

Once the process is complete, the SD card can be disconnected from the adapter and inserted into the slot provided on the Raspberry Pi. Now the Pi can be connected to the keyboard, mouse, power cable, LAN and HDMI cable and started. Once the boot process is complete, the Raspbian desktop appears.

The system must now be configured. To do this, open the “Raspberry Pi configuration” under Start -> Settings. In this window, the system language, the time zone and the keyboard layout can be changed in the “Localization” tab. In the “Interfaces” tab, the SSH function can be activated to control the Pi from another computer via the network. The host name, the system password and the screen resolution of the Raspberry Pi can also be changed in the system tab. The new configuration takes effect after a reboot. The software required for the project can now be installed on the single-board computer. (Source 9, 2019)

5.2.2 Webserver

Before a program is installed on the Raspberry Pi using the console command `apt-get install`, the package lists should be reloaded using the command: `sudo apt-get update`. After the package lists have been updated to prevent outdated program versions, the command `sudo apt-get upgrade` can be executed. This command updates already installed packages to the latest version. This process may take a few minutes.

As soon as the programs and package lists have been updated, the first program essential for the project can be installed: Apache Webserver. This is done with a simple console command:

```
sudo apt-get install apache2
```

The http web server service should now be installed and running in the background. To check this, open the pre-installed web browser of the Raspberry Pi and call up the localhost in the address bar. As shown in Figure 15, a website generated by Apache should appear, confirming that the web server is working.

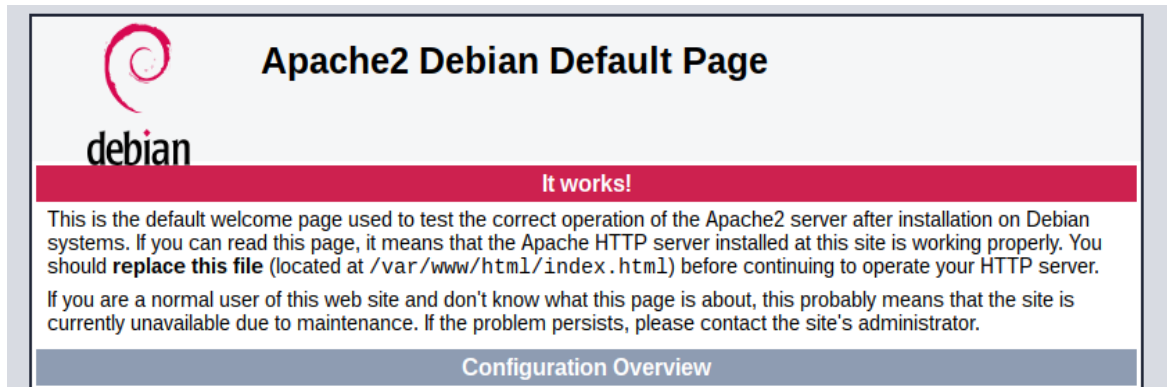


Figure 15: Apache default page

The latest version of Apache's Php module must be installed so that Apache can also process Php files:

```
sudo apt-get install php libapache2-mod-php
```

To check whether Php has been installed correctly, the root directory of the web server is called up in the Linux terminal with the following command:

```
cd /var/www/html
```

The index.html generated by Apache is located in this directory. This is now deleted and a new file called index.php is created using a Linux text editor such as "Geany". The following code is now saved in this file to check the function of php:

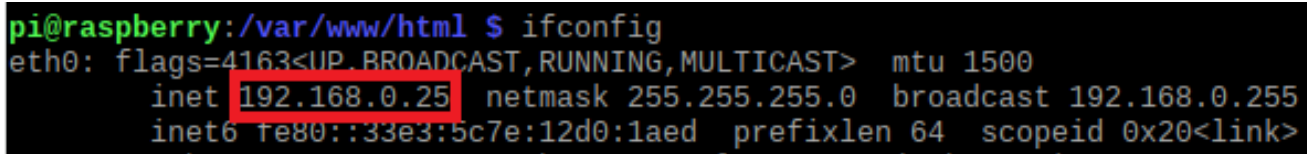
```
<?php phpinfo() ; ?>
```

If the php processing of the web server works, the website should now display information about the installed php version, similar to Figure 15 (source 10, 2019):

localhost	
<div> <div>PHP Version 7.0.33-0+deb9u1</div> <div>php</div> </div>	
System	Linux raspberry 4.9.0-8-686 #1 SMP Debian 4.9.130-2 (2018-10-27) i686
Build Date	Dec 7 2018 11:36:49
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.0/apache2
Loaded Configuration File	/etc/php/7.0/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/7.0/apache2/conf.d
Additional .ini files parsed	/etc/php/7.0/apache2/conf.d/10-opcache.ini, /etc/php/7.0/apache2/conf.d/10-pdo.ini, /etc/php/7.0/apache2/conf.d/20-calendar.ini, /etc/php/7.0/apache2/conf.d/20-ctype.ini, /etc/php/7.0/apache2/conf.d/20-exif.ini, /etc/php/7.0/apache2/conf.d/20-fileinfo.ini, /etc/php/7.0/apache2/conf.d/20-ftp.ini, /etc/php/7.0/apache2/conf.d/20-gettext.ini, /etc/php/7.0/apache2/conf.d/20-iconv.ini, /etc/php/7.0/apache2/conf.d/20-json.ini, /etc/php/7.0/apache2/conf.d/20-phar.ini, /etc/php/7.0/apache2/conf.d/20-posix.ini, /etc/php/7.0/apache2/conf.d/20-readline.ini, /etc/php/7.0/apache2/conf.d/20-shmop.ini, /etc/php/7.0/apache2/conf.d/20-sockets.ini, /etc/php/7.0/apache2/conf.d/20-sysvmsg.ini, /etc/php/7.0/apache2/conf.d/20-sysvsem.ini, /etc/php/7.0/apache2/conf.d/20-sysvshm.ini, /etc/php/7.0/apache2/conf.d/20-tokenizer.ini
PHP API	20151012
PHP Extension	20151012

Figure 16: Phpinfo

The website of the web server can be accessed from a smartphone in the same network using the IP address of the Raspberry Pi. To find this out, the `ifconfig` command is executed in the Pi's Linux terminal. Figure 16 shows the IP address of a Raspberry Pi Apache server.



```
pi@raspberrypi:/var/www/html $ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.0.25 netmask 255.255.255.0 broadcast 192.168.0.255
    inet6 fe80::33e3:5c7e:12d0:1aed prefixlen 64 scopeid 0x20<link>
```

Figure 17: Raspberry Pi `ifconfig`

5.3 MySQL database

The Raspberry Pi needs information about the radio parameters of the individual radio devices so that it can issue a send command to the ESP8266 via the http protocol. A database is now created for the radio codes already stored in an Excel spreadsheet, which the microcontroller can call up at any time.

5.3.1 Installation

The packages required for a MySQL database are loaded with the following console command:

```
sudo apt-get install mysql-server php-mysql mysql-client
```

Once the installation process is complete, the phpMyAdmin tool is installed with the following command:

```
apt-get install phpmyadmin
```

During the installation of phpMyAdmin, a package configuration opens, which requires you to select the web server for which phpMyAdmin is to be configured and to set a MySQL application password. After confirming the entered password, phpMyAdmin will be installed.

To be able to access phpMyAdmin in a browser, the Apache configuration file must be modified. To do this, open the file with the path `/etc/apache2/apache2.conf` in a text editor and add the following line at the bottom of the code (source 11, 2019):

```
Include /etc/phpmyadmin/apache.conf
```

After saving the file and restarting the Apache service using the console command

```
/etc/init.d/apache2 restart
```

, phpMyAdmin can be accessed via the Raspberry Pi's web browser under `localhost/phpmyadmin`. The login form that appears requires a user name and password to be entered. However, before logging in, the user "phpmyadmin" must be made a "privileged user" in order to be able to manage the database as an administrator. To do this, the MySQL program is started on the Raspberry Pi and the following MySQL query is executed in it:

```
sudo mysql
```

```
GRANT ALL PRIVILEGES ON *.* TO 'phpmyadmin'@'localhost' WITH GRANT
OPTION;
```

```
FLUSH PRIVILEGES;
```

The user “phpmyadmin” and the MySQL user password selected during installation can now be used to log into the phpmyadmin administration system. (Source 12, 2019)

5.3.2 Database structure

Before the database structure can be created, a database must first be created. This is done in phpmyadmin by clicking on “New” in the column in which all existing databases in the system are listed. The database can be named and created using the form that appears.

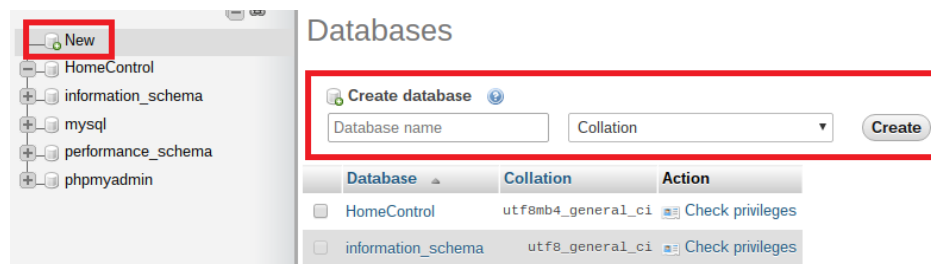


Figure 18: create database

A table structure can now be created for the created database. For this project, two tables with the names “codes”, in which the transmission codes of the radio devices are stored, and “groups”, in which the groups are defined, are required.

To create a table, simply click on the newly created database. The new table can be named and the number of columns defined in the form that appears. The “codes” table with 7 columns and the following attributes is created first:

#	Name	Typ	Kollation	Attribute	Null	Standard	Kommentare	Extra
1	id	mediumint(8)		UNSIGNED	Nein	kein(e)		AUTO_INCREMENT
2	name	char(10)	utf8mb4_general_ci		Nein	kein(e)		
3	protocol	int(2)			Nein	kein(e)		
4	pulselength	int(4)			Nein	kein(e)		
5	value	int(20)			Nein	kein(e)		
6	action	tinyint(1)			Nein	kein(e)		
7	type	char(5)	utf8mb4_general_ci		Nein	kein(e)		
8	group_id	mediumint(8)			Nein	kein(e)		

Figure 19: codes table

The “id” column is assigned the ‘autoincrement’ function, is the “primary key” of this table and ensures that each row can be called up individually. The data in the “name” and “type” columns is stored as a character string; “protocol”, “pulselength” and ‘value’ as an integer and “action” as a Boolean, which can be either 1 (on) or 0 (off). The “type” column identifies the radio devices created either as ‘light’ (light source) or as “power” (socket of a device).

#	Name	Typ	Kollation	Attribute	Null	Standard	Kommentare	Extra
1	group_id 	mediumint(8)			Nein	kein(e)		AUTO_INCREMENT
2	name	char(20)	utf8mb4_general_ci		Nein	kein(e)		

Figure 20: groups table

Now the table for grouping the radios is created. Only two columns are required for this: the ID and the name of the group. The ID is given the primary key, is of the integer data type and counts the groups entered using the auto-increment function. The “name” column contains the names of the groups as a character string.

5.3.3 Insert Data Records

The data records saved in the Excel table can now be entered into the “codes” table in the database. To do this, click on the table and select the “Insert” tab. As shown in Figure 20, a form appears which can be used to create a new data row.

Column	Type	Function	Null	Value
id	int(30)	<input type="text"/>		<input type="text"/>
name	char(30)	<input type="text"/>		303-1
protocol	int(30)	<input type="text"/>		1
pulselength	int(30)	<input type="text"/>		163
value	int(30)	<input type="text"/>		349491
action	char(30)	<input type="text"/>		on
type	char(30)	<input type="text"/>		light

Figure 21: enter data records

All data rows from the Excel table are now entered into the database to register each radio in the system. The individual groups of radios can then be entered in the “groups” table using the same procedure.

5.4 Webapplication

Now that the ESP8266 is ready for use, the Raspberry Pi web server is set up and the database is ready for retrieval, these parts can be connected to each other in the last main part of the project using a web application. To do this, the following files are first created in the root directory of the web server using the Linux terminal and opened with the Linux text editor Geany:

```
cd /var/www/html  
  
sudo geany index.php  
  
sudo geany styles.css  
  
sudo geany script.js  
  
sudo geany request.php  
  
cd /var/www/html/includes
```

In the following steps, the website is built using HTML, CSS, JavaScript and a php script, as outlined in Figure 22. The layout is separated by two sections; the menu and the body of the website. The menu section contains the tabs that determine the content of the body. The controls for the individual menu tabs are located in the body. A control element stands for a radio device registered in the database and can either switch it on or off using two action buttons.

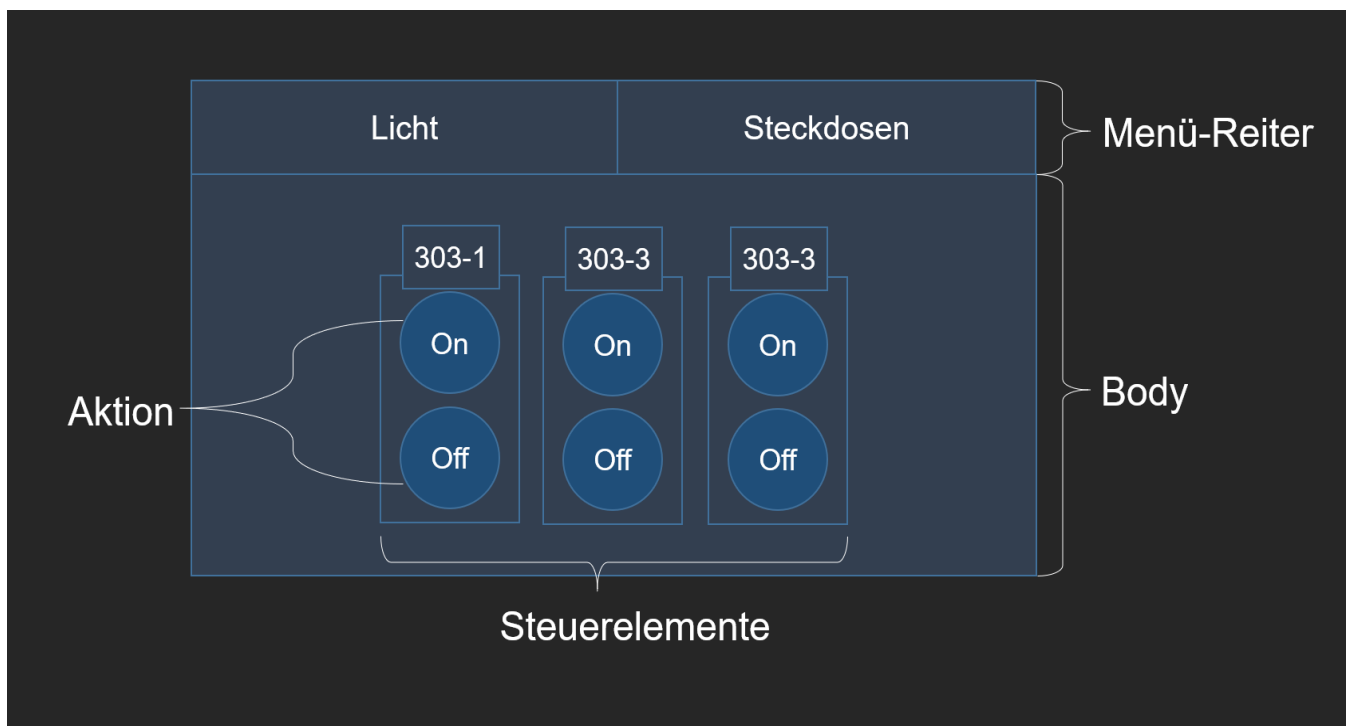


Figure 22: website sketch

5.4.1 HTML structure

You can now start to define the skeleton of the website in the index.php file using html tags. First, the main body of the website is divided into the menu and the tab body. The menu is given the tab elements “Light” and “Sockets”.

The tab body is again divided into a container and the control elements it contains using div tags. The control element area contains the name of the radio device and both an on and an off button.

5.4.2 Responsive CSS

To add a graphically appealing design to the layout, the stylesheet is integrated into the website as a CSS file with the following line in the head tag:

```
<link rel="stylesheet" type="text/css" href="styles.css">
```

Since the website should react to different screen resolutions and adapt accordingly, the viewport is defined by a meta tag.

```
<meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0, user-scalable=no"/>
```

The viewport is the area visible to the device accessing the website and its size can vary on smartphones. The parameter content="width=device-width sets the width of the page to the screen width of the host device and user-scalable=no prevents smartphone users from inadvertently zooming into the website. (Source 13, 2019)

Styles can now be added to the existing classes and tags in the CSS file. As CSS stands for “cascading style sheet”, which means that the file is read from top to bottom, there are lines of code for a CSS reset at the top. The CSS reset resets possible default settings of a browser and ensures that the web application looks the same in every browser.

After the CSS reset, areas such as the menu, the container and the control elements are styled and positioned. However, the position and size of an element are not determined by a static value, but by a value relative to the viewport or another element. This means that the relations of the layout remain the same for different viewport sizes and the website is responsive.

The layout of the controls should also adapt to the screen size. The controls are automatically arranged in rows and columns using flexboxes. The CSS flexbox layout arranges the individual controls in a container and determines their position in the body of a tab based on the display size. (Source 14, 2019)

5.4.3 Dynamic Structure PHP

5.4.3.1 Control elements

The individual control elements of the body of the web application are generated dynamically by a php script using the registered database entries. As this is a php file, the html code can be mixed with the php code. The php code only needs to be introduced by a “<?php” and closed by a “>”.

First, the script connects to the created database. This script is created as a php file in the “Includes” folder in the root directory of the web server. Before the database query process, this script is integrated using an include function.

As the database is installed on the same system, the script connects to it using PDO and “localhost”:

```
$servername = "localhost";

$user = "phpmyadmin";

$password = "evgym1!";

$dbname = "HomeControl";

$conn = new PDO("mysql:host=$servername;dbname=$dbname",$user, $password);
```

The access data for the database is identical to the login for phpMyAdmin. As soon as the connection to the database has been established, the following SQL query is used to load data records into a Php array:

```
$sql = "SELECT * FROM codes WHERE action = '1' AND type = 'light'";

$stmt = $conn->query($sql);

$result = $stmt->fetchAll();
```

The table rows from codes are loaded into the array \$result, in which the radio device is noted as the light source and the action is “on”. The condition for the on action is there to ensure that each radio is not loaded into the array twice, as each device is created once for each action (on and off).

The content of the array can now be projected onto the website as a control element using the echo function, a while loop and the condition that the array is not empty. The end result is the following html code for each radio entered in the database:

```
<div class='control_element'>

„Device name“

<a href='#' class='button'>On</a>

<a href='#' class='button'>Off</a>

</div>
```

The SQL query and while loop is used once in the div tag of the light tab and once in the socket tab. The condition for the socket tab is adjusted to type = 'light'.

5.4.3.2 Groups

After generating the individual control elements, the groups are projected onto the website using the Groups table. To do this, the existing connection to the database is used and the database entries of the groups table are loaded into an array. In a foreach loop, a database query is executed for each database entry of the groups table, which is selected for all data records of the codes table for the respective group.

```
SELECT * FROM codes WHERE action = '1' AND type = 'light' AND group_id  
=<Index of foreach-loop>
```

The array with the device names of the relevant data records is then read out in another foreach loop and the parameter adapted for the HTML attribute "onclick" is saved in a variable.

```
foreach($groups as $row) {  
  
    $functionson = $functionson.'turnOn("'.$row_groups["name"].'");';  
  
    $functionsoff = $functionsoff.'turnOff("'.$row_groups["code_device-  
name"].'");';  
  
}
```

This loop saves the values, which are then used as parameters in the button generation process, similar to the control elements. This means that a new group can be created in the system with a simple database entry and the assignment of a "group_id", which is then automatically generated by the php script.

5.4.4 JavaScript

The programming of the interactive menu is realized by a JavaScript and the jQuery library. To integrate JavaScript and jQuery into the project, the following script tags are written in the head of the source code:

```
<script src="script.js"></script>  
  
<script src="https://ajax.aspnetcdn.com/ajax/jquery/jquery-  
3.3.1.min.js"></script>
```

JavaScript code can now be written to the script.js file to program the content of the web application. (Source 15, 2019)

5.4.4.1 Menu

The menu can now be programmed. As the web application will mainly run on smartphones, it should also be possible to navigate the website using a finger swipe gesture. For this purpose, a small script is

added to the script.js file, which recognizes such a finger swipe gesture and increases or decreases the value of an integer variable depending on the swipe direction.

As soon as a left or right swipe is recognized, an update function is also called, which shows or hides the content of a div tag depending on the value of the changed variable. The variable therefore determines which tab is currently selected and only shows the content in the body of the website that is bound to the current variable value by a class.

Tapping/clicking on the respective menu item also changes the value of the variable so that the menu navigation is also functional for a desktop as a host device. (Source 16, 2019)

5.4.4.2 Ajax-Request

As pressing the operating buttons of the control elements does not yet operate a radio device, these are connected to the database and the ESP8266 microcontroller in the following steps. For this purpose, the “onclick” attribute with the “turnOn()” or “turnOff()” function is added to the control elements by the php script as follows:

```
<a href='#' class='button' onclick='turnOn("radioDeviceName")'>On</a>
```

```
<a href='#' class='button' onclick='turnOff("radioDeviceName")'>On</a>
```

The parameter of the function is always the name entered in the database for the radio device. This parameter is automatically generated by the dynamic php loop.

The attribute causes the specified JavaScript function to be executed when the button is clicked. These functions must now be defined in the script.js file. As JavaScript cannot establish a MySQL connection to a database, the two functions are used to forward the device name and the action to be performed on the device to a php script using an Ajax request.

If an On or Off button is pressed, the device name of the activated control element and the action are forwarded to the Request.php script as parameters of a request URL.

```
194
195     function turnon(device) {
196
197         var requesturl = "request.php?device=" + device + "&action=on"
198         $.ajax({
199             type: "GET",
200             url: requesturl
201         });
202     }
203
204     function turnoff(device) {
205
206         var requesturl = "request.php?device=" + device + "&action=off"
207         $.ajax({
208             type: "GET",
209             url: requesturl
210         });
211     }
212
213 }
214
```

Figure 23: turnOn and turnOff functions JavaScript

Figure 23 shows the two functions of the script.js file in a Sublime text editor. First, the request url to the php script is defined. To do this, the GET array of the php file is filled at the “device” position with the device name received as a parameter. jQuery uses the generated request url and the GET method to send an Ajax request to the corresponding php script. (Source 17, 2019)

5.4.5 request.php

The php script request.php transmits the parameters for a radio signal to be executed to the ESP8266 microcontroller. For this purpose, a connection to the radio code database is established using the include function and the values obtained using the GET method are stored in variables. An Isset function is now used as a condition to check whether the two variables \$device and \$action have content. If this condition is fulfilled, the action passed decides which database query must now be made.

```
$sql = "SELECT * FROM codes WHERE action = '1' AND name = '". $device. "'";  
  
or
```

```
$sql = "SELECT * FROM codes WHERE action = '0' AND name = '". $device. "'";
```

The database entry is selected by the device name and the action and its content is saved in an array. Now the link can be generated with which the server accesses the ESP8266 using a curl command.

```
$link = "http://192.168.0.20/send?protocol=".$row["protocol"]."&pulse-  
length=".$row["pulselength"]."&value=".$row["value"]  
  
curl_setopt($ch, CURLOPT_URL, $link);  
  
$content = curl_exec($ch);
```

By using dots, strings and variables can be linked together to form a string. The \$link variable generates an http request with the radio parameters that is understandable for programming the ESP8266. The microcontroller transmits these radio parameters via its 433 MHz transmitter module and the respective radio device then reacts to the signal with a corresponding action.

5.5 Potential Problems

During the practical execution of the work steps, some problems can occur in the “Home Control” project, which I will now discuss.

5.5.1 Network problems

There could be problems connecting the microcontroller to the school network as, unlike the Raspberry Pi 3, it does not have a LAN socket. The school protects the network through certification processes, which have not yet been tested for the ESP8266.

5.5.2 Radio range

Due to the 433 MHz radio method selected, the radio signal may be too weak to reach a socket or lamp socket. The reason for this could be walls that are too thick or distances that are too far.

For this reason, transmitter modules with antennas were chosen for the project in order to amplify the signal in this case. In addition, several microcontrollers could be set up at the same time to cover a larger area.

5.5.3 Linux operating system

Because I haven't yet gained much practical experience with the Linux operating system, unknown errors could pose a problem. I wouldn't be able to solve these unexpected blockages intuitively, but would first have to research them on the Internet, which takes time.

5.5.4 Hardware damage

It is possible that a component of the radio transmission may lose its function during the implementation of the project. This would delay the completion of the project, as in this case it must first be tested whether it is due to the software or hardware. Then it has to be evaluated whether only a radio device has broken down or perhaps the transmitter module is defective. Due to complex debugging processes, hardware damage can therefore delay the implementation time of the project.

5.6 Time table

My schedule is tabulated by the columns as main chapters in the work steps required for the sections.

The percentages represent the estimated working time required for the work steps of a chapter.

ESP8266 Configuration (15%)	Raspberry Pi 3 (25%)	Database (20%)	Webapplication (40%)
Arduino IDE Installation	Raspbian Installation	MySQL Installation	HTML base structure
Radio receiver Installation	Raspbian Configuration	phpMyAdmin Installation	CSS
Determine radio codes	Apache2 Webserver	phpMyAdmin Configuration	Control elements, Groups
Radio transmitter installation	Php-Service Installation	Create codes table	JavaScript
Test of radio codes	Read out the IP address	Create Groups Table	request.php File
Webserver	Test of Webserver	Insert Data Records	Test of whole system
Test of Webserver		Test of database	

5.7 Additional work

If I need less time than expected to complete my project, I have thought of two additional tasks to fill the remaining working time.

5.7.1 State of lamps

As the radio devices are currently operated using two buttons, these could be combined into a single button. This would save space in the HTML body and also display the status of the lamp or socket directly in the web application.

As it is not possible to record the status of the device due to the wireless lamp holders and sockets used in the project, the value of a database column of a device should change after each transmission signal initiated by the web application in order to be able to determine the status of the device. The website does not need to be reloaded either, as JavaScript ensures that all statuses are updated live.

The implementation of this addition requires that the radio transmission works perfectly. To guarantee this security, more ESP8266 microcontrollers are used. The transmission command is sent to all microcontrollers simultaneously in order to increase the range in which the radio devices can be located.

5.7.2 User Profile

User profiles are also being considered as an addition. As a household usually has several residents, it should be possible to assign certain devices to specific users. Not all devices need to be affected by such an assignment, but private devices, for example, to which not all residents of an apartment should have access.

User profiles should be implemented using php sessions and cookies. In this way, a login system can be created into which the residents can log in with their profile. It will also be necessary to create an additional database table for the user groups in order to store passwords and rights. These rights are then assigned to the individual radio devices in the codes table in the form of an ID.

6 Cost calculation

The hardware costs of the project are quite high because of the price-intensive radio-compatible interfaces. Registering a single light bulb or socket in the system already costs €6, which is why the project is easy to expand in terms of hardware, but very costly. Another major cost factor is the purchase of the single-board computer, which is around €60 for a project of this size. The micro controllers can also be expanded very easily with a broader project scope and have a unit price of around €5.

Product	Quantity	Price
Raspberry Pi 3 B+	1	35,90 €
Smraza Raspberry Pi Case	1	15,99 €
UGREEN USB 3.0 card reader	1	8,99 €
SanDisk Ultra 32GB microSDHC	1	10,98 €
MANAX radio sockets	4	24,99 €
KW-Commerce Radio lamp socket	6	36,80 €
Anpro 433-mHz radio transmitter	8	7,99 €
Anpro 433-mHz radio receiver	8	7,99 €
AZDelivery NodeMCU ESP8266 Lua Amica Micro-controller	3	15,12 €
ALLEU Jumper cable + Breadboards Set	1	10,99 €
		Total sum: 175,74 €

7 Conclusion

In conclusion, it can be said that the development of the “Home Control” project challenged me in many ways and I encountered some problems during the written formulation. I really enjoyed the process of solving these unexpected problems, knowing that the project would become better and more perfect with every problem solved or work step simplified. I can proudly say that I worked out and developed the theoretical and practical part of the engineering project completely independently with the help of the sources provided.

My project supervisor, Professor Berger, drew my attention to certain standards that I was not familiar with in the course of the written elaboration. Solving these problems showed me that there is usually not just one way to implement a new function or query in a system or to connect two systems with each other.

Furthermore, the technician project not only gave me valuable experience, but also a product that I can use in my everyday life. Thanks to the written elaboration, I will probably never forget how “Home Control” works and is structured.

Planning the whole project from scratch and finding ways to connect different technologies that were initially incompatible also taught me a lot. It has shown me that it is better to build bridges between existing technologies than to have to reinvent the wheel in complete isolation from everything else. Many processes in one system can be transferred to another, where they can then be processed further.

Declaration

I, Elias Karré, declare that the present technician project was written independently on the basis of theoretical and practical work carried out exclusively by myself and that no unauthorized aids were used.

Date: 14. 1. 2019

A handwritten signature in dark ink, appearing to read 'Karré', is positioned above a horizontal line.

Elias KARRÉ

References

Source 1: <https://tutorials-raspberrypi.de/nodemcu-esp8266-433mhz-funksteckdosen-steuern/>

Source 2: <https://www.instructables.com/id/Using-an-ESP8266-to-Control-Mains-Sockets-Using-43/>

Source 3: <https://arduino-hannover.de/2015/04/08/arduino-ide-mit-dem-esp8266/>

Source 4: <https://www.raspberrypi.org/>

Source 5: <https://www.raspbian.org/>

Source 6: <https://httpd.apache.org/>

Source 7: <https://searchoracle.techtarget.com/definition/MySQL>

Source 8: <https://www.heise.de/download/product/win32-disk-imager-92033>

Source 9: <https://jankarres.de/2012/08/raspberry-pi-raspbian-installieren/>

Source 10: <https://www.raspberrypi.org/documentation/remote-access/web-server/apache.md>

Source 11: <https://www.stewright.me/2012/09/tutorial-install-phpmyadmin-on-your-raspberry-pi/>

Source 12: <https://askubuntu.com/questions/763336/cannot-enter-phpmyadmin-as-root-mysql-5-7>

Source 13: https://www.w3schools.com/css/css_rwd_viewport.asp

Source 14: <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>

Source 15: https://www.w3schools.com/jquery/jquery_get_started.asp

Source 16: <https://stackoverflow.com/questions/2264072/detect-a-finger-swipe-through-javascript-on-the-iphone-and-android>

Source 17: <https://stackoverflow.com/questions/10734396/how-to-launch-php-file-from-javascript-without-window-open>

Figures 1-6 and 9-23: own illustration

Figure 7: https://www.amazon.de/gp/product/B0798SWCBC/ref=oh_aui_detail-page_o02_s00?ie=UTF8&psc=1 [Zugriff: 24.12.2018]

Figure 8: https://www.amazon.de/gp/product/B009VCZUQC/ref=oh_aui_detail-page_o00_s00?ie=UTF8&psc=1 [Zugriff: 24.12.2018]

List of Figures

Figure 1: Raspberry Pi 3 interface	9
Figure 2: ESP8266 interface.....	11
Figure 3: Arduino IDE tools	12
Figure 4: serial monitor output for radio Receiver	13
Figure 5: receiver module	14
Figure 6: Transmitter module	14
Figure 7: lamp sockets and remote.....	15
Figure 8: radio sockets and remote	15
Figure 9: Arduino IDE preferences	16
Figure 10: Board management	16
Figure 11: ESP8266 connected with radio receiver	17
Figure 12: excel table radio codes.....	18
Figure 13: ESP8266 connected with radio transmitter	19
Figure 14: Win32-Disk-Imager	20
Figure 15: Apache default page	21
Figure 16: Phpinfo	22
Figure 17: Raspberry Pi ifconfig	22
Figure 18: create database.....	24
Figure 19: codes table	24
Figure 20: groups table	25
Figure 21: insert data records.....	25
Figure 22: website sketch.....	26
Figure 23: turnOn and turnOff functions JavaScript	31