

KotiPi: Developing a smart home application with Raspberry Pi

Mirya Nezvitskaya

Bachelor's Thesis in
Business Information
Technology

August 2015



Author Mirya Nezvitskaya	
Degree programme Business Information Technology	
Report/thesis title KotiPi: Developing a smart home application with Raspberry Pi	Number of pages and appendix pages 37 + 21
<p>Nowadays smart home is an important segment of Internet of Things. This research is created for enthusiasts who are interested in IoT and home automation and would like to engineer it themselves.</p> <p>Raspberry Pi is an excellent affordable computer that can be used for practical projects; however there are no turnkey full home automation systems that would be available to a user, though many parts of automation tasks in Raspberry Pi are available via tutorials, blogs and forums. This project is a basic home automation system that is open-source, so it can be used by anyone, and improved and developed further.</p> <p>The idea of this project is to make functions usable via a web application that can be accessed via a smart phone, a tablet or a laptop. The project also introduces areas for improvement and ideas for future development that can bring this prototype to a fully customized home automation system and even for commercial areas.</p> <p>Overall, the project explains theoretical background of technologies and concepts used, how to set up necessary environment, how to develop its features, and how to create a web application. The evaluation and analysis have also been made in order to assess future probability that such a project can be used commercially and privately.</p>	
Keywords Automation, Embedded Systems, Control Engineering, Raspberry Pi, Smart home, Internet of Things, PHP, JavaScript, Python, Bash	

Abbreviations and Terms

App	Application
ARM	Advanced RISC Machines processor
Bash	UNIX shell and command language
CSS	Style sheet language
CSI	Camera Serial Interface
DDNS	Dynamic Domain Name System
DIY	Do it yourself
Ethernet	Family of computer networking technologies
GitHub	Platform for sharing code
GPIO	General-purpose Input/Output
Ground (pins)	Integrated circuit power-supply
GUI	Graphical User Interface
HTML	Hypertext markup language
Hz	Hertz, the unit of frequency
IoT	Internet of Things
IP	Internet Protocol
IT	Information Technology
JavaScript	Dynamic programming language

kΩ	kilo ohm, SI unit of electrical resistance
LED	Light-emitting diode
OS	Operating System
Open Source	Project developed with a free license
PHP	Server-side scripting language
Pi	Raspberry Pi
Python	High-level programming language
SD	Secure Digital
SSH	Secure shell, cryptographic network protocol
SQLite	Relational database management system
UI	User Interface
URL	Uniform Resource Locator
V	Volt
VLC	videoLAN cross-platform multimedia player

Table of Contents

1	Introduction.....	1
1.1	Need for the application.....	1
1.2	Thesis objectives and deliverables	2
1.3	Delimitation.....	3
2	Theoretical background	4
2.1	Home automation	4
2.2	Embedded systems	4
2.3	Raspberry Pi.....	5
2.4	Relays and sensors	6
2.5	Web application	6
3	Research plan	8
3.1	Traditional system development	8
3.2	Project plan	9
3.3	Software requirements.....	10
4	Necessary environment	11
4.1	Setting up Raspberry Pi's native OS, Raspbian	11
4.2	Setting up SSH for remote login	11
4.3	Setting up Nginx web server for a web app.....	13
4.4	Creation of database	13
5	Architecture and design of the system	15
5.1	Architecture of KotiPi	15
5.2	Design of KotiPi	16
5.3	Testing of the architecture's system.....	16
6	Developing the features	20
6.1	Connection of sensors to Pi.....	20
6.2	Controlling lights with Raspberry Pi with remote control outlets	21
6.3	Setting up music for alarm and web app to play.....	25
6.4	Setting up Raspberry Pi's camera module for sensing movement	26
6.5	Creating a web app and controlling all the features with it	27
6.6	Testing	33
7	Evaluation.....	35
7.1	Evaluation of results	35

7.2	Project evaluation	35
7.3	Future development.....	36
8	Summary	37
	References	38
	Appendices.....	40

1 Introduction

It is paradoxical, yet true, to say, that the more we know, the more ignorant we become in the absolute sense, for it is only through enlightenment that we become conscious of our limitations. Precisely one of the most gratifying results of intellectual evolution is the continuous opening up of new and greater prospects.

Nikola Tesla

Nowadays, simplifying life tasks plays an important role of IT development. It is with automation and IoT that developers and researchers are being fascinated and determined that it is the future where all IT development would go. Without any doubt, automating everyday tasks can save a lot of time and effort. However, still these days home automation products are not widely available and are quite expensive, so not everyone can afford them. But a new trend in technology is DIY automation systems.

This project follows the trend of DIY philosophy, meaning it is not a ready-made system, but combined by oneself, and is dedicated to creating a fully functional prototype of home automation application system management with Raspberry Pi that can be used by anyone. Since all of the software that is used in this project is open source and the hardware is inexpensive, anyone can create and enjoy their own “smart home” application without big expenses. Moreover, this project is licensed under Apache license, which is an open source license, so anyone can use KotiPi for studying, using or modifying.

1.1 Need for the application

Currently there are no full smart home Raspberry Pi systems available, only separate automation functions. However, Raspberry Pi is used for many practical projects. It is an excellent inexpensive tool for learning and creating new solutions. Raspberry Pi has a rather large community built around it with the purpose of sharing code, solutions and techniques associated with the platform. Since the source code is open source, anyone can use this prototype for developing their own smart home application. Also once the project is complete and the results are satisfying, it can be published in Raspberry Pi Foundation page or Raspberry Pi Foundation Blog as a part of open source projects written on Raspberry Pi. Furthermore, a private stakeholder for this project, Matias Henri Rönnerberg, who is also acting as an active supporter and collaborator, plans on to continue improving and developing this project from the prototype that is created.

1.2 Thesis objectives and deliverables

The thesis objectives are as follows:

1. to use Raspberry Pi for engineering a smart home application

Several solutions for developing a smart home application can be deployed. However, as stated above, Raspberry Pi is used for its practicality. In this project there is a working Raspberry Pi server with functional sensors and lights attached to it through GPIO pins and a relay board; a working web application that can interact with Raspberry Pi.

2. to develop a smart home application with Bash, Python, JavaScript and PHP

The target is to develop a working prototype of a smart home application. The concrete and measurable result is a smart home application and an analysis written about it. The project is concentrated on the back-end programming (reading output from sensors, turning on/off relays on the relay board) as well as front-end programming (web application for the user). The application has 3 main modules, which are: wake up call, lock house and unlock house. The lock house turns off all the lights automatically, music and activates camera module that detects movement. If the camera would detect movement an email is sent to a user about it; once the user unlocks the house, the camera module is turned off. The wakeup call is a smoother version of an alarm clock. Instead of an alarm tone, the application wakes you up with music and lights turning on. Also the app controls outside lights, shows inside temperature, and can play music.

3. to make the end product open-source, so anyone can learn how to use it

One critical point of this project is to make the prototype of a smart home application available for everyone, so it can be used for studying, using and modifying the solution. The idea is to follow the open source philosophy and to contribute to the open source. Thus, this project is registered under Apache license and is available on GitHub.

1.3 Delimitation

For such a project with a narrow scope it is important to mention what does not belong to it and should be delimited. Since the research is not based on theoretical or literature analysis, information about Raspberry Pi in general is omitted (how Raspberry works or how GPIO pins work). However, Chapter 2, Theoretical Background, covers Raspberry Pi basics. Also, this project does not cover Raspberry Pi Model A, Model B+ and 2 Model B. Moreover, there are no other possible solutions or features of a web application other than those that are mentioned in software requirements. Last, but not least, this project does not cover the discussion of whether Raspberry Pi is the best feasible tool for home automation application.

2 Theoretical background

Theoretical background covers general information and tools used for developing the smart home application. Furthermore, it explains some concepts that are used for creating this project.

2.1 Home automation

Home automation goes back to the beginning of the 20th century. Home appliances are considered to be the first step into automation of home tasks. The first to ever be born was the engine-powered vacuum cleaner in 1901. In the next 40 years other appliances were invented, for example irons, washing machines, toasters, etc.

During 60s the first smart device was born, though a commercial failure, it was a device for creating shopping lists, controlling temperature and turning different appliances off and on. During 90s smart home became a very popular concept, and home automation became a new multi-billion industry from affordable options to expensive unique automotive tasks. Nowadays, home automation is mainly about security and energy-efficiency. (Hendricks, D. The History of Smart Homes, 2014) Current home automation trends are DIY systems, appliances, security, smart locks and IoT. (Clauser, G. New Trends in Home Automation, 2014)

2.2 Embedded systems

Embedded system is a device that is used to control, monitor, and assist the operation of equipment, machinery or plant. (Jain, P. Embedded System, Engineer's Garage, 2015) Embedded systems are interacting with the outside world with the help of sensors, relays and other systems, for example mechanical or electrical. Most commonly used examples of embedded systems are navigation systems, mobile phones, internet servers, etc.

“Today, embedded systems are found in cell phones, digital cameras, camcorders, portable video games, calculators, and personal digital assistants, microwave ovens, answering machines, home security systems, washing machines, lighting systems, fax machines, copiers, printers, and scanners, cash registers, alarm systems, automated teller machines, transmission control, cruise control,

fuel injection, anti-lock brakes, active suspension and many other devices/gadgets.” (Jain, P. Embedded System, Engineer’s Garage, 2015)

2.3 Raspberry Pi

Raspberry Pi is a small inexpensive single-board computer that was created in 2006 in University of Cambridge as a learning tool for students. (Raspberry Pi Foundation, 2015)

Nowadays it is used both by professionals and amateurs. The power of Raspberry Pi is in GPIO pins that it has.

“These pins are a physical interface between the Pi and the outside world. They are switches that you can turn on or off (input) or that the Pi can turn on or off (output). Seventeen of the 26 pins are GPIO (system) pins; the others are power (3.3V or 5V) and ground.” (Raspberry Pi Foundation, 2015)

Raspberry Pi has several models; the model used for this project is Model B. Figure 1 shows the schema of Pi Model B.

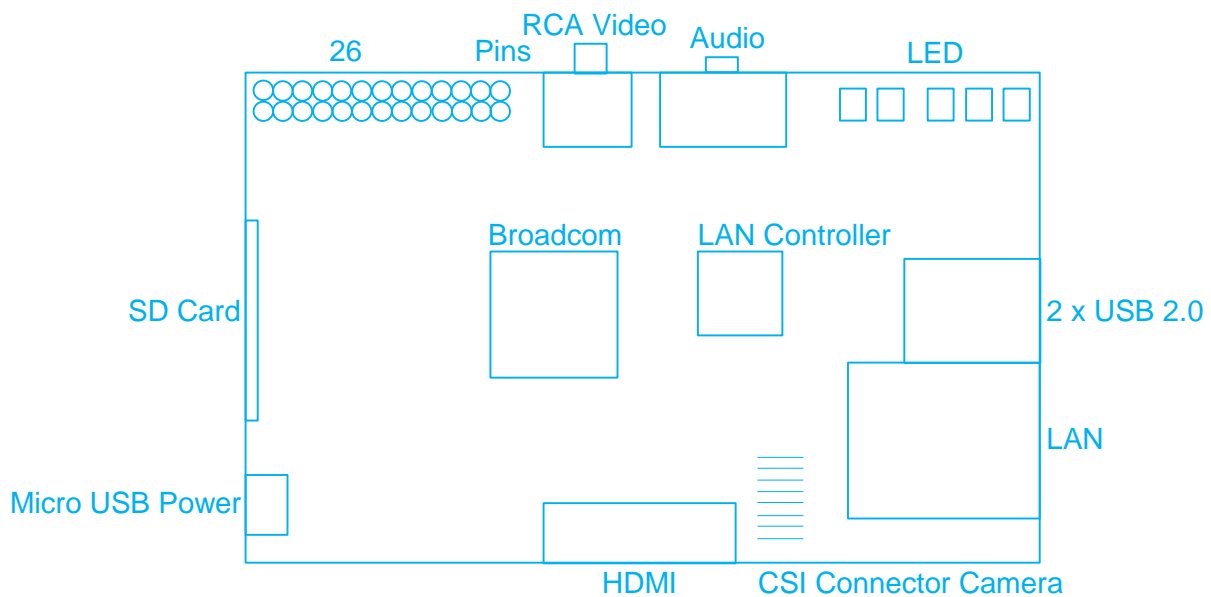


Figure 1. Schema of Raspberry Pi Board B (Source: made by author)

2.3.1 Raspberry Pi as an embedded system

Since Raspberry Pi can communicate with an outside world, the most popular usage for Raspberry Pi is as a part of an embedded system, where other boards, sensors, etc. would communicate with each other. Pi does that with the help of GPIO pins. Figure 2 provides a detail schema of pins.

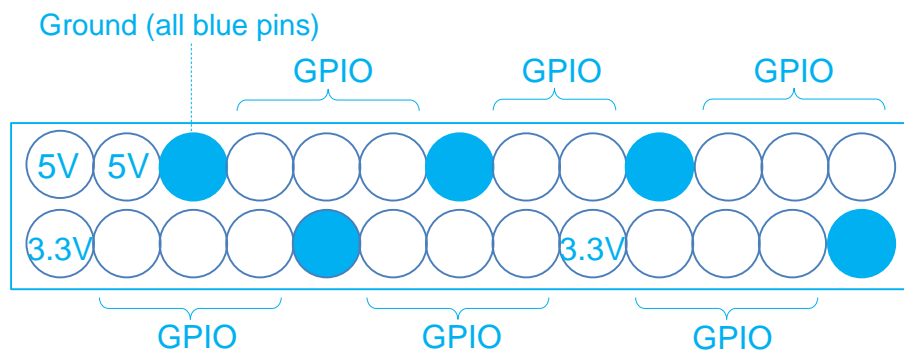


Figure 2. Schema of Pi pins on Model B (Source: made by author)

2.4 Relays and sensors

A relay is an electromechanical switch made up of an electromagnet and a set of contacts. A relay consists of 2 different circuits that are independent. In the first circuit a switch is controlling power to electromagnet. The second circuit is being operated by an armature, which is acting like a switch. (Bullock, M. How Relays Work, Electronics How Stuff Works, 2015) Relays are used to control a circuit with a low-power signal, for example, radio or telephone. A sensor is a device that senses or detects some characteristics of the environment, for example temperature or humidity. There are two types of sensors: analog and digital. Raspberry Pi uses digital sensors. However, it is possible to attach analog sensors to Raspberry Pi via a breadboard or an analog-to-digital converter.

2.5 Web application

Web applications are involved with lots of moving parts and interacting components. (Purewal, S. Learning Web App Development, 2014) A web application simply put is a program that runs in a web browser. It is created by a browser supported language. In this project the front end part of the application is created using HTML, CSS, and JavaScript, while back-end of this project is implemented with Bash, Python, PHP and JavaScript. There are different technologies that can be used for web development. However, these

days the most crucial part is the responsive web style. Lots of devices are used that are different in size, thus adjustable screen size is important for any web application. For this application, also a responsive web style is used, so anyone from any device can access without depending on the size of the application. Bootstrap CCS is an open source project, it has many responsive web styles that can be used by anyone. One of the styles, Jumbotron, is used in this project.

3 Research plan

Research plan covers the methodology of this project, project plan, as well as the intended results of the project. It is particularly important in this project, since here is specified how the work is conducted, how a system should work, and what should be the deliverables.

3.1 Traditional system development

Since this project has a narrow scope and only a prototype of a system is introduced, a traditional system development cycle was chosen, as represented on figure 3. First, the project's priorities and opportunities are selected, where it is determined that a home automation application with Raspberry Pi is needed, since there are no full turnkey solutions, only parts of the system. Second, basic requirements are made and the intended results are introduced. Third, the specifications of architecture and design are introduced, where also the architecture is being tested and confirmed that it can be used as a feasible solution. Fourth, the features are being developed and tested. And last, the system is being evaluated, and the future development for improved system is being introduced in order to continue development of this system.

Only basic requirements are being introduced, basic architecture and design, because requirements engineering does not belong to the scope of this project, and the project's main concentration is on the development process, other than requirements engineering process. However, it would be recommended, if such a project would be picked up as a commercial project, a traditional IS Development cycle could be used, since this method's every single step is logical and rational.

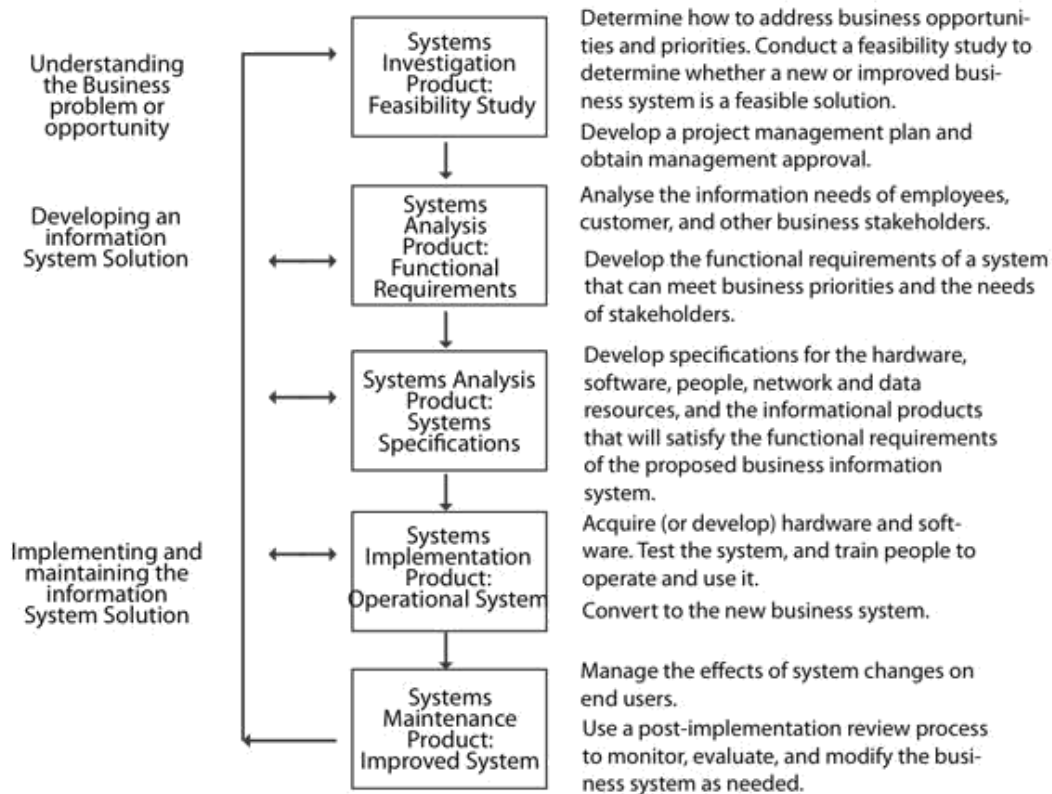


Figure 3. Traditional System Development Cycle (Source: O'Brien, J.A. and Marakas, G. Introduction to Information Systems, 2005)

3.2 Project plan

The development part of the project can be divided into 4 parts:

1. **Setting up the necessary environment.** This part includes setting up Raspbian OS, setting up a cryptographic network protocol (SSH) to allow remote login, setting up a web server for a web application and creation of a database with SQLite.
2. **Architecture and design of the system and testing the architecture of a smart home application.** This covers an architecture design and a preliminary system design, as well as the connection of a relay board to Raspberry Pi, turn it on/off with Python and Bash scripts and lastly, turn it on/off with PHP through a web server to test the architecture.

3. **Developing features.** After necessary environment is set up, all the features that are mentioned in Chapter 3.3., Software requirements, are developed and explained.
4. **Testing.** The final part compares Chapter 3.3, Software requirements, features of the project with the actual results, after which the testing is done. Since the scope of this project is narrow, automated testing is omitted.

3.3 Software requirements

Since this project has a narrow scope, it is necessary to also point out the requirements of the project and intended results.

- a web app should be working in the local area network
- an app can turn on/off outside and inside lights
- an app shows temperature in live time (being updated every second)
- a user can set up an alarm clock that will be remembered by the app
- a user can play music with the app
- an app, when house is locked, detects movement with the help of a camera by calculating pixels, and sends an email to a user
- a history can be accessible via an app

These requirements are the foundation of the development of the features, and are precisely followed during the system development, as well as used during final part of development, which is testing.

4 Necessary environment

In this part, how to set up the necessary environment for this project is explained.

4.1 Setting up Raspberry Pi's native OS, Raspbian

Any Linux OS that has been ported for the ARM processor of the Raspberry Pi can be used with Raspberry Pi. The list of these can be found on the official Raspberry Pi site (raspberrypi.org). For this project, Raspberry Pi native OS, Raspbian, is used. The official image can be downloaded from the official Raspbian page (raspbian.org). After downloading the OS image, it can be written onto an SD card using the `dd` utility found in most Linux distributions with the following command as a root user:

```
dd bs=4M if=nameofimage.img of=/path/to/SDcard
```

The `dd` utility is a Linux utility for converting and copying files; the arguments used are block size (`bs=`), source (`if=`) and destination (`of=`). Once OS is written onto an SD card, the card needs to be put into the SD card slot of the Raspberry Pi, and the machine needs to be turned on, by plugging it in. Configuration settings need to be set up by localizing keyboard, time, etc. In this step, an important parts in configuration are to enable camera and to enable SSH. Raspbian has a set-up utility called `raspi-config` through which all of these settings can be done.

4.2 Setting up SSH for remote login

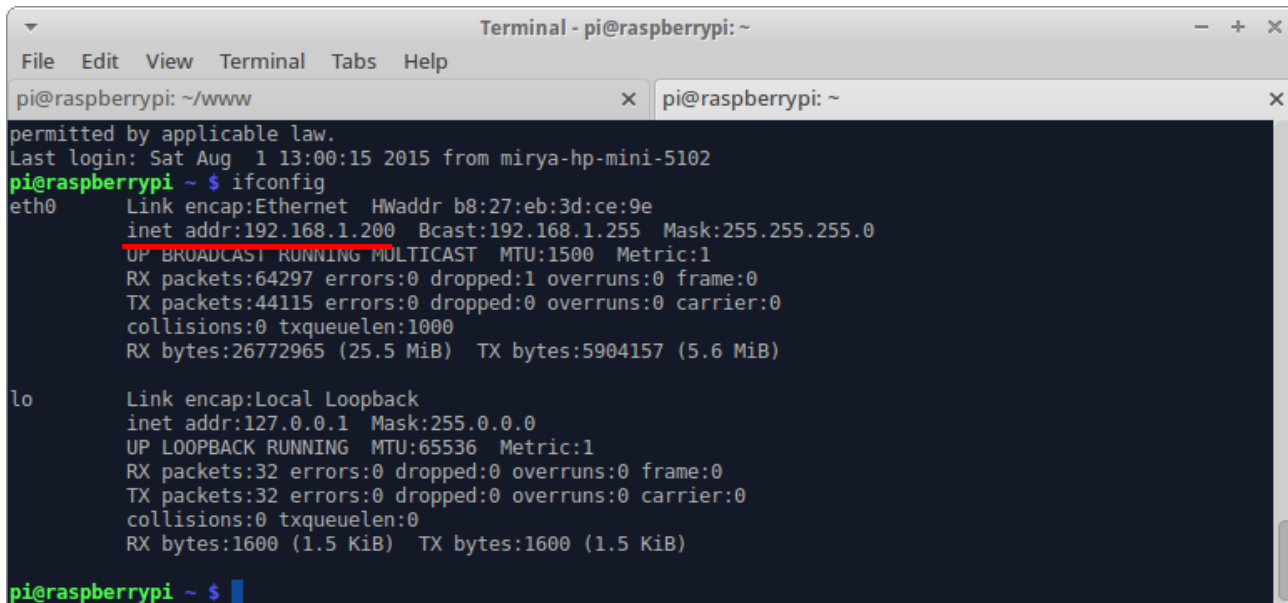
Raspberry Pi can be accessed via SSH, for this a local IP of Pi should be known. In order to learn machine's IP address a simple Bash command can be typed:

```
ifconfig
```

Once the command is being executed, network interface configuration is shown. The Pi's local IP address can be found under `eth0 inet addr` as shown on figure 4. Once machine's IP address is known, Pi can now be accessible by any machine in local network. In order to access it on a Linux machine (or any machine that supports Bash interpreter) a simple SSH command is used:

```
ssh user@youripaddress
```

Where a default user in Raspberry Pi is pi, and your IP address is the address of Pi machine. After executing the command, the program asks for the password; the default password for pi is raspberry. Once the password is entered successfully it is important to change your password. A secure connection with Raspberry Pi via SSH has been set. An example is shown on figure 5.

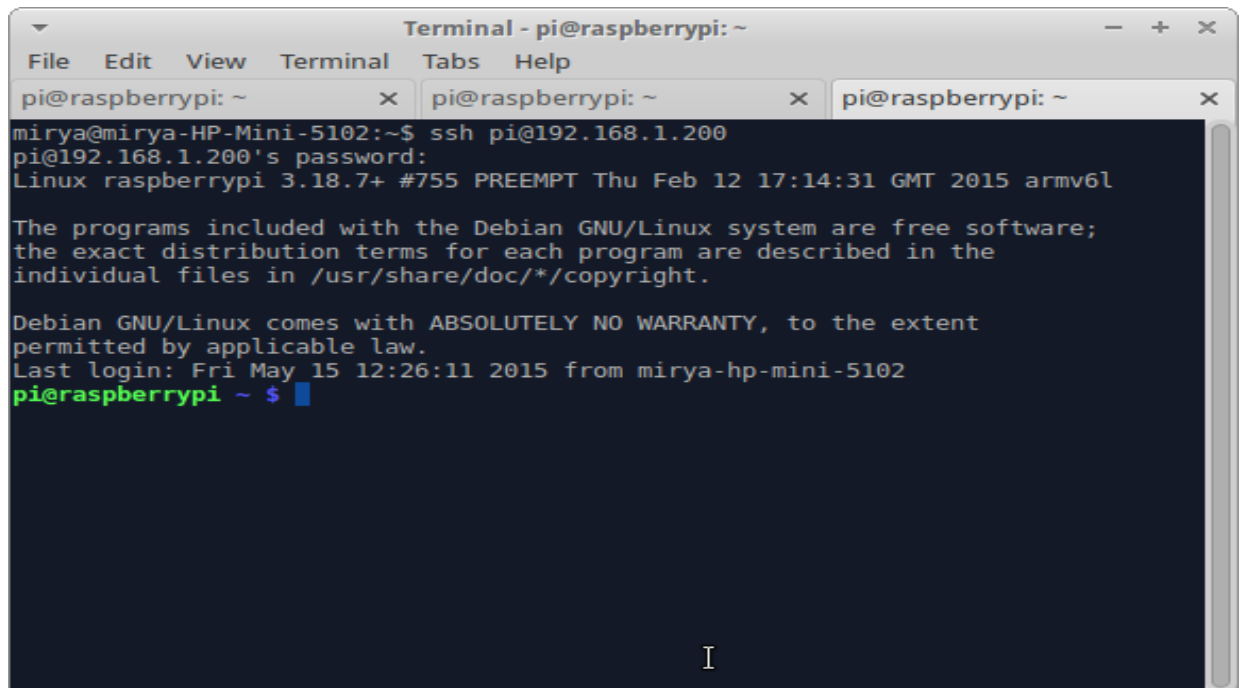


```
Terminal - pi@raspberrypi: ~
File Edit View Terminal Tabs Help
pi@raspberrypi: ~/www x pi@raspberrypi: ~
permitted by applicable law.
Last login: Sat Aug 1 13:00:15 2015 from mirya-hp-mini-5102
pi@raspberrypi ~ $ ifconfig
eth0      Link encap:Ethernet  HWaddr b8:27:eb:3d:ce:9e
          inet addr:192.168.1.200  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:64297 errors:0 dropped:1 overruns:0 frame:0
          TX packets:44115 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:26772965 (25.5 MiB)  TX bytes:5904157 (5.6 MiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:32 errors:0 dropped:0 overruns:0 frame:0
          TX packets:32 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1600 (1.5 KiB)  TX bytes:1600 (1.5 KiB)

pi@raspberrypi ~ $
```

Figure 4. Command ifconfig being executed (Source: author's screenshot)



```
Terminal - pi@raspberrypi: ~
File Edit View Terminal Tabs Help
pi@raspberrypi: ~ x pi@raspberrypi: ~ x pi@raspberrypi: ~
mirya@mirya-HP-Mini-5102:~$ ssh pi@192.168.1.200
pi@192.168.1.200's password:
Linux raspberrypi 3.18.7+ #755 PREEMPT Thu Feb 12 17:14:31 GMT 2015 armv6l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri May 15 12:26:11 2015 from mirya-hp-mini-5102
pi@raspberrypi ~ $
```

Figure 5. Once SSH is accessed, welcoming screen (Source: author's screenshot)

4.3 Setting up Nginx web server for a web app

Nginx is an open source HTTP server and proxy. In order to download it onto Raspberry Pi, the apt-get command, which is a command for the advanced packaging tool, needs to be used as a root user:

```
apt-get nginx
```

After Nginx has been downloaded, it is important to check if PHP is available on a machine, if not it is necessary to install PHP by executing next command as a root user:

```
apt-get install php5-fpm php-apc
```

Now that Nginx and PHP are available on Raspberry Pi machine, several modifications are needed to be made onto Nginx configuration to add PHP to Nginx, so that instead of basic HTML files, Nginx can also read PHP files; as a root user it is needed to go into Nginx configuration file, where nano is a text editor:

```
nano /etc/nginx/sites-available/default
```

In Appendix 1 the modified configuration is attached. Once the configuration is set, Nginx has added PHP to be used by the web server.

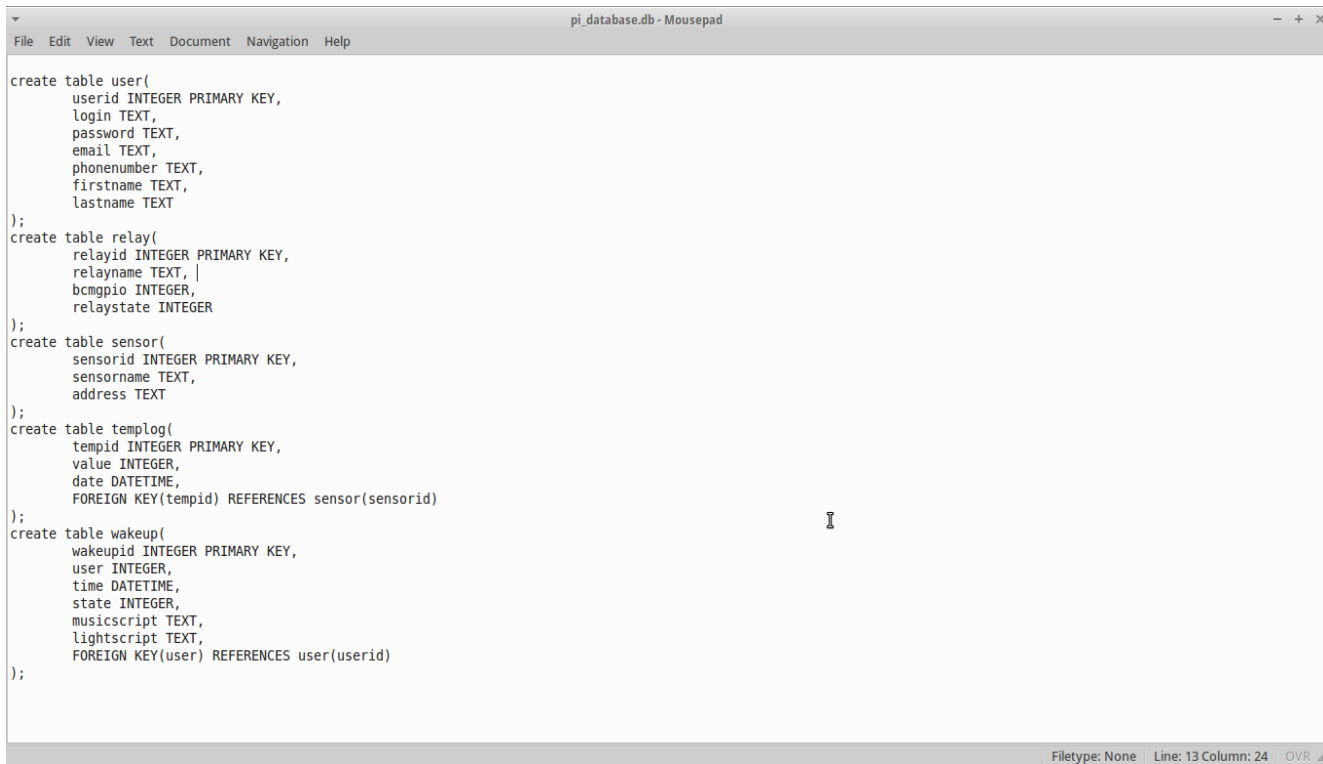
4.4 Creation of database

Database is needed only if some data needs to be stored. In the case of this project, data is needed for temperature data, and also to store users and their passwords. For this project a simple SQLite database is created. In order to create a database, SQLite needs to be installed on Raspberry Pi, for this the following command is executed as a root user:

```
apt-get install sqlite3
```

For this project SQLite3 is used, other than SQLite, because SQLite3 has more functions that can be used for the Pi's database. After SQLite has been installed, the project's database can be created. In order to create a database `sqlite3 nameofdatabase`

needs to be executed in bash terminal. Next the tables are created with their values, and the tables can also be filled with test data, for now. Figure 6 shows the tables that are used for the database for this project.



```
pi_database.db - Mousepad
File Edit View Text Document Navigation Help

create table user(
  userid INTEGER PRIMARY KEY,
  login TEXT,
  password TEXT,
  email TEXT,
  phonenumber TEXT,
  firstname TEXT,
  lastname TEXT
);
create table relay(
  relayid INTEGER PRIMARY KEY,
  relayname TEXT, |
  bcmgpio INTEGER,
  relaystate INTEGER
);
create table sensor(
  sensorid INTEGER PRIMARY KEY,
  sensorname TEXT,
  address TEXT
);
create table templog(
  tempid INTEGER PRIMARY KEY,
  value INTEGER,
  date DATETIME,
  FOREIGN KEY(tempid) REFERENCES sensor(sensorid)
);
create table wakeup(
  wakeupid INTEGER PRIMARY KEY,
  user INTEGER,
  time DATETIME,
  state INTEGER,
  musicscript TEXT,
  lightscript TEXT,
  FOREIGN KEY(user) REFERENCES user(userid)
);

Filetype: None Line: 13 Column: 24 OVR
```

Figure 6. Database for this project (Source: made by author)

5 Architecture and design of the system

This chapter explains the architecture and design of the system, as well as the architecture is being tested, by connecting relay board to Raspberry Pi, turning it on/off with Python and then with web app.

5.1 Architecture of KotiPi

The core architecture of this project is simple: Raspberry Pi interacts with sensors, a relay board and a camera module through Python or Bash; PHP then sends the scripts' data to a web server, where JavaScript together with HTML and CSS translate the data into human readable form, as well as inserting data into a web app. Figure 7 represents how the architecture of such project would work.

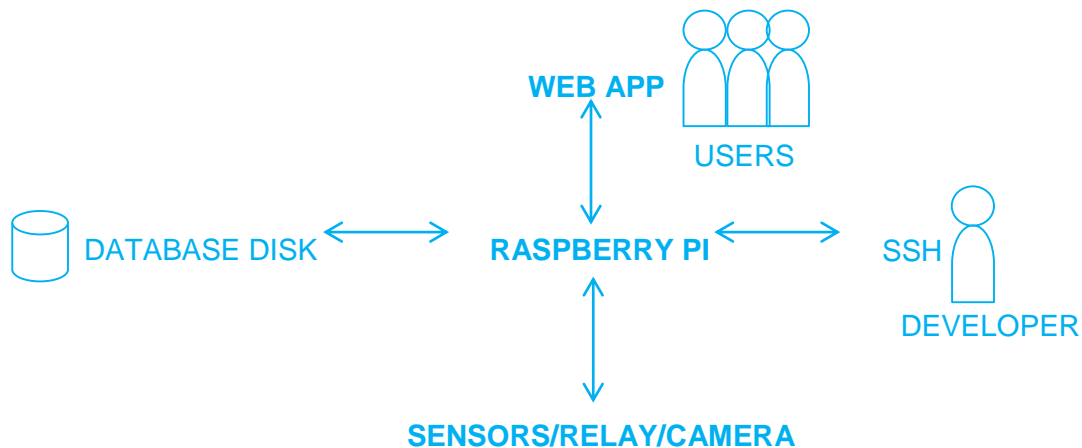


Figure 7. Architecture of KotiPi (Source: made by author)

5.2 Design of KotiPi

The main concentration of this project is to make a web application interact and get data from sensors, relays and a camera module; however, front-end is still important for this project. To make the application creation easier, some CSS has been used from Bootstrap's Narrow Jumbotron. Bootstrap is a framework to create responsive web sites. Since this app can be used by a smartphone, a laptop or a tablet, a responsive web page is an important aspect of the design for this project. Bootstrap's Narrow Jumbotron is a suitable library for such a project. It can be downloaded from the official Bootstrap page (getbootstrap.com). In this project, only Narrow Jumbotron's CSS files are used. Figure 8 represent a sketch of a web app

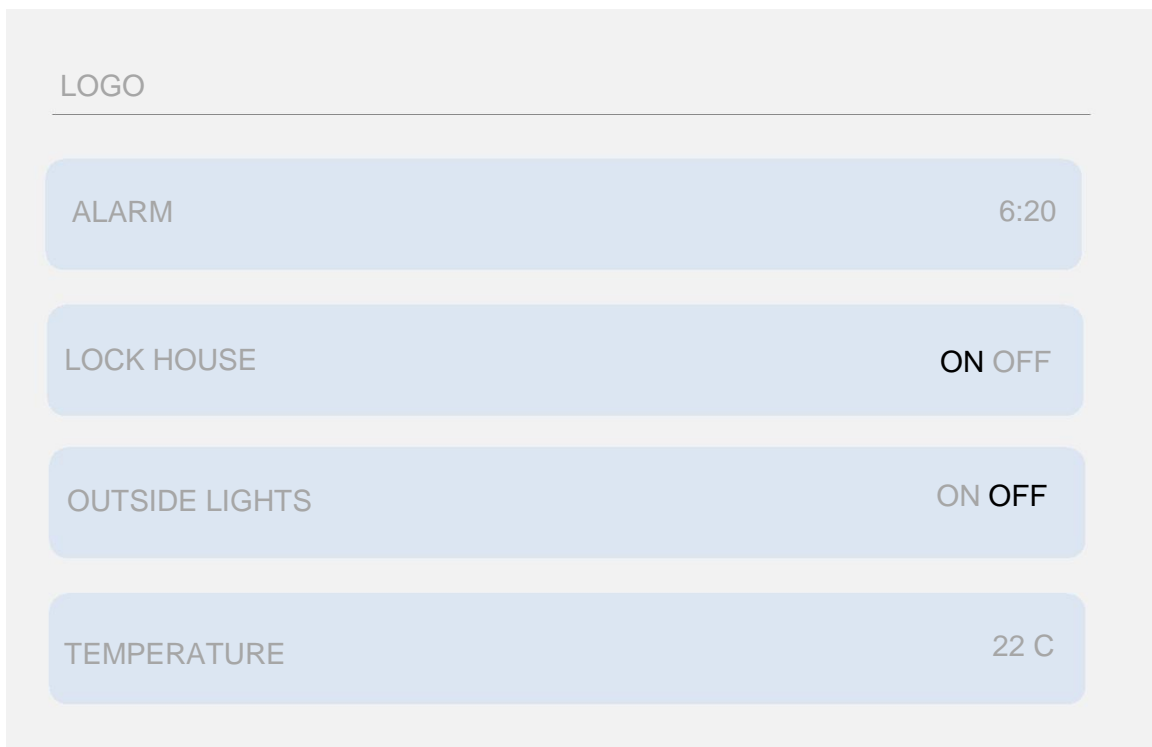


Figure 8. Design of KotiPi (Source: made by author)

5.3 Testing of the architecture's system

Relay board is a switch with a low-voltage signal, similar to GPIO pins that Raspberry Pi has. The relay board used for this project is Sainsmart 5V 8 channel relay board.

In order to connect a relay board to Pi, first Raspberry Pi needs to be shut down and unplugged. Once Pi is unplugged, the relay's female cables need to be connected into

Raspberry Pi's pins; the pins to connect them in should match, for example a ground female cable goes into a ground pin slot. In figure 2 a sketch of Raspberry Pi pins provides detailed information of pins' position on the board. (Pater, T. Connect a relay board to your Raspberry Pi. 2015)

5.3.1 Python script to turn on/off relays

In order to check if the relay board is connected correctly, a simple Python script can be created for turning on/off relay boards' LED lights. The script imports Raspberry Pi GPIO, sets the loop through pins and turns on lights corresponding to the pin. Once the script is run, the LED lights on relay board go on and a click can be heard:

```
#!/usr/bin/python
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
#pin numbers should be specified
pinNumbers = [17, 23]
#create a loop
for i in pinNumbers:
    GPIO.setup(i, GPIO.OUT)
    GPIO.output(i, GPIO.HIGH)
#time to sleep in between
sleepTime = 2
# main loop
try:
    GPIO.output(17, GPIO.LOW)
    print "1"
    time.sleep(sleepTime)
    GPIO.output(23, GPIO.LOW)
    print "2"
    time.sleep(sleepTime)
GPIO.cleanup()
print "Bye"
```

```
except KeyboardInterrupt:
    print "Quit"
GPIO.cleanup()
```

5.3.2 Accessing relay controls with web

In order to access a relay board with web, first of all a GPIO interface library needs to be installed, called wiringPi. In order to install wiringPi library, in terminal the following commands needs to be executed as a root, first git needs to be installed, and the GPIO interface library from git, which is an open-source control system, is installed:

```
apt-get install git-core
git clone git://git.drogon.net/wiringPi
```

Now it is needed to clone the library from git, and the library needs to be installed and built:

```
cd wiringPi
git pull origin
./build
```

After installing and building wiringPi library, it is much easier to access GPIO pins. With wiringPi library, it is possible to turn on and off relays via simple Bash command where 17 is the number of pin, where one of the relay boards is connected:

```
gpio mode 17 out
gpio write 17 1
gpio write 17 0
```

These commands first specify that pin 17 produces an output, next the relay board is switched on, and then switched off. Next, a simple webpage can be created to turn on and off a relay board. Nginx stores all the files in `/usr/share/nginx/www`; there 2 new files need to be created, one for turning on relay – `turnon.php`, where it is specified that the output is produced, and the relay goes on:


```
<?php
    system(gpio mode 17 out);
    system(gpio write 17 1);
?>
```

Another file needs to be created to turn off relay – turnoff.php; in which the program turns off the relay.

```
<?php
    system(gpio mode 17 out);
    system(gpio write 17 0);
?>
```

Now web access needs to be tested, in web browser the following is executed, where IP address is the Pi's IP address:

```
ipaddress/turnon.php
ipaddress/turnoff.php
```

When first page is executed, relay board is switched on, the light can be seen and the click can be heard, while the second page, once executed, turns relay off.

6 Developing the features

This chapter contains all of the developing features for KotiPi, as well as the creation of the web app.

6.1 Connection of sensors to Pi

There are different sensors that can be used for Raspberry Pi, for example, humidity or infrared motion sensor. In this project, only one sensor is used, temperature sensor that measures house's temperature and is updated every second.

6.1.1 Connecting temperature sensor to Pi

Digital temperature sensor that is used for Raspberry Pi for this project is Sainsmart. In order for a temperature sensor to work, a 4.7k Ω resistor needs to be soldered onto the sensor between pin 2 and 3. Then the temperature sensor is connected into Raspberry Pi with dupont wires to the corresponding pins, as shown on figure 2.

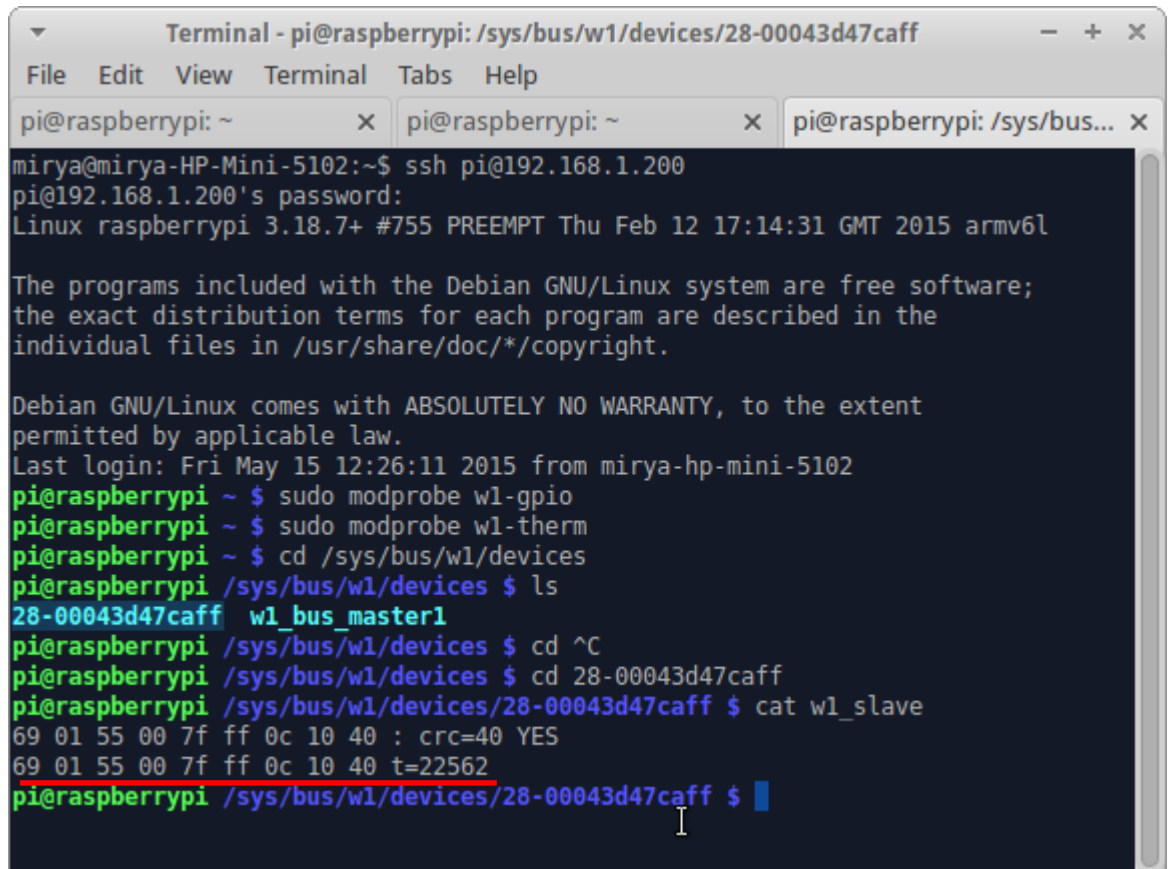
Once the sensor is connected to Raspberry Pi the kernel modules need to be enabled to support one wire sensors, which in this case measure temperature. The command for enabling and disabling kernel modules is modprobe, which can only be run as a root user:

```
modprobe w1-gpio modprobe w1-therm
```

Once the kernel module is loaded, the temperature values are located in `/sys/bus/w1/devices`. Once the folder is accessed via the terminal, in this folder there is a unique alphanumeric address for each sensor, which is connected to Raspberry Pi, next a sensor's folder is accessed, and inside the folder, the command for concatenating the file that houses the output of the sensor needs to be executed:

```
cat w1_slave
```

Two lines are printed, where on the second line the section that starts with "t=" is the temperature degrees in Celsius. In figure 9 an example is shown. (Kirk, M. Raspberry Pi Temperature Sensor. 2015)



```
Terminal - pi@raspberrypi: /sys/bus/w1/devices/28-00043d47caff
File Edit View Terminal Tabs Help
pi@raspberrypi: ~ x pi@raspberrypi: ~ x pi@raspberrypi: /sys/bus... x
mirya@mirya-HP-Mini-5102:~$ ssh pi@192.168.1.200
pi@192.168.1.200's password:
Linux raspberrypi 3.18.7+ #755 PREEMPT Thu Feb 12 17:14:31 GMT 2015 armv6l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri May 15 12:26:11 2015 from mirya-hp-mini-5102
pi@raspberrypi ~ $ sudo modprobe w1-gpio
pi@raspberrypi ~ $ sudo modprobe w1-therm
pi@raspberrypi ~ $ cd /sys/bus/w1/devices
pi@raspberrypi /sys/bus/w1/devices $ ls
28-00043d47caff w1_bus_master1
pi@raspberrypi /sys/bus/w1/devices $ cd ^C
pi@raspberrypi /sys/bus/w1/devices $ cd 28-00043d47caff
pi@raspberrypi /sys/bus/w1/devices/28-00043d47caff $ cat w1_slave
69 01 55 00 7f ff 0c 10 40 : crc=40 YES
69 01 55 00 7f ff 0c 10 40 t=22562
pi@raspberrypi /sys/bus/w1/devices/28-00043d47caff $
```

Figure 9. Concatenated file of the temperature sensor (Source: author's screenshot)

6.2 Controlling lights with Raspberry Pi with remote control outlets

There are several options to control lights with the Raspberry Pi. One is to connect the power wires from the light into the output slots of a relay board, so when the relay board is switched on, the light is switched on. However, this option is quite dangerous, since it is needed to work with open wires that are 230V, and knowledge of basic electronics is required. Much simpler and safer solution is to buy wireless remote control outlets. For this project a standard 230V 50Hz package of wireless remote control outlets from Clas Ohlson is used, as you can see on figure 10.

In order to make a light turn on and off with a relay board, a remote control needs to be disassembled and soldered to wires, that would be connected to a relay board. First of all, a plastic container needs to be removed from the remote control, as shown on figure 11. Once a plastic container is removed, a battery and buttons on/off can be seen. These buttons need to be connected to the pins on the relay board, so when Pi sends a signal to the relay board that it should go on, the button goes on, and a wireless outlet goes on also.

Any kind of wires can be used, for this project wires from an old Ethernet cable is used. As well as buttons, a battery needs to be connected to the power, the power of this remote control is 3V. Figure 12 shows how and where to solder a wire for the power. In order to solder: first, a soldering iron needs to be warmed up, once it is hot, a small part of a solder is used to heat up to hold together the wire and the power; once it is done, it needs to be cooled off.



Figure 10. Remote control with a wireless outlet (Source: photo by author)



Figure 11. Inside remote controller (Source: photo by author)

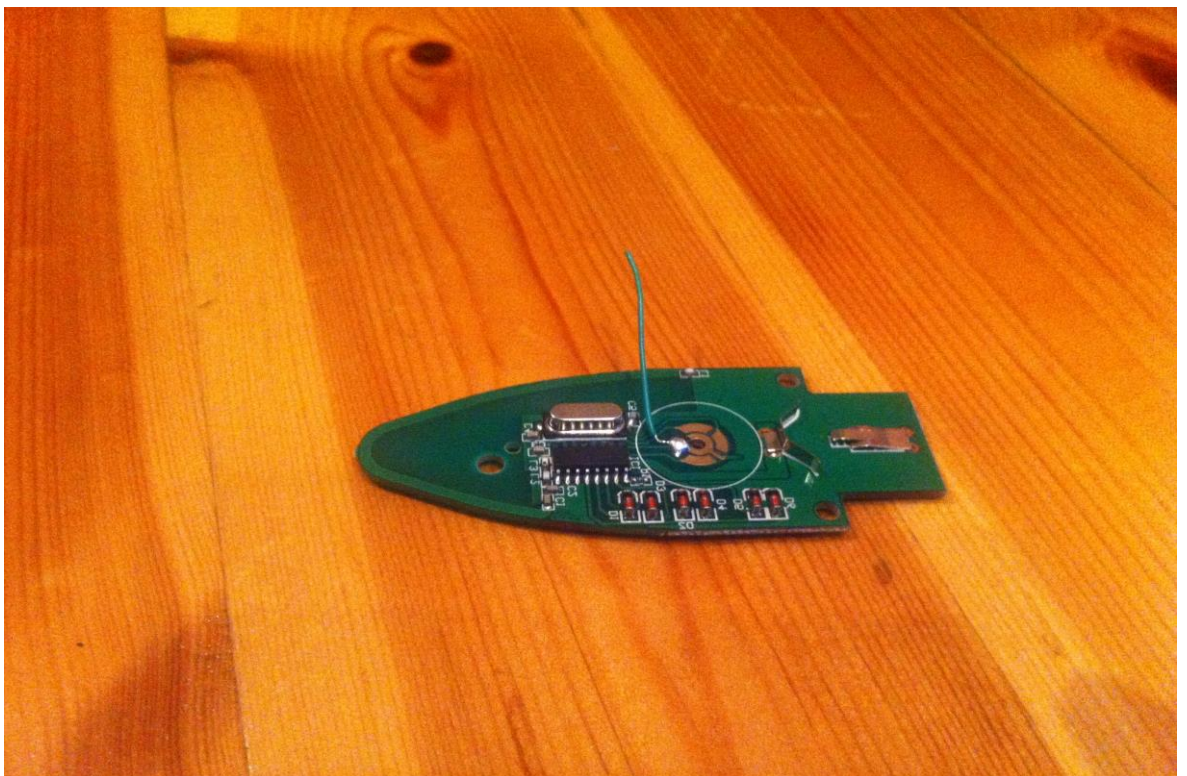


Figure 12. Solder connects power and a wire (Source: photo by author)

The buttons on the remote control need to be soldered to the wires as well. In order to solder them, it is important to understand how such buttons work. For each wireless outlet there are two buttons: on and off. Each button (on or off) turns on for a second and then goes off. Figure 13 shows where each wire needs to be soldered – one inside and one outside, which are then connected to a single relay, which can then make or break the connection between leads simulating the button being pressed. Once everything is soldered, the wires can be attached to the correct terminals on the relay board. To check if everything is set up correctly, two PHP scripts can be run, that were created in Chapter 5.3.2, Accessing relay controls with web, script `turnon.php` and script `turnoff.php`. However, since the structure of turning on and off is different with remote control buttons, it is needed to modify those scripts. In one script the pin needs to go on, then wait for a second, and go off. To turn the light on pin 17 is used:

```
<?php
    system(gpio mode 17 out);
    system(gpio write 17 1);
    sleep(1);
    system(gpio write 17 0);
?>
```

To turn the light off, pin 23 is used:

```
<?php
    system(gpio mode 23 out);
    system(gpio write 23 1);
    sleep(1);
    system(gpio write 23 0);
?>
```

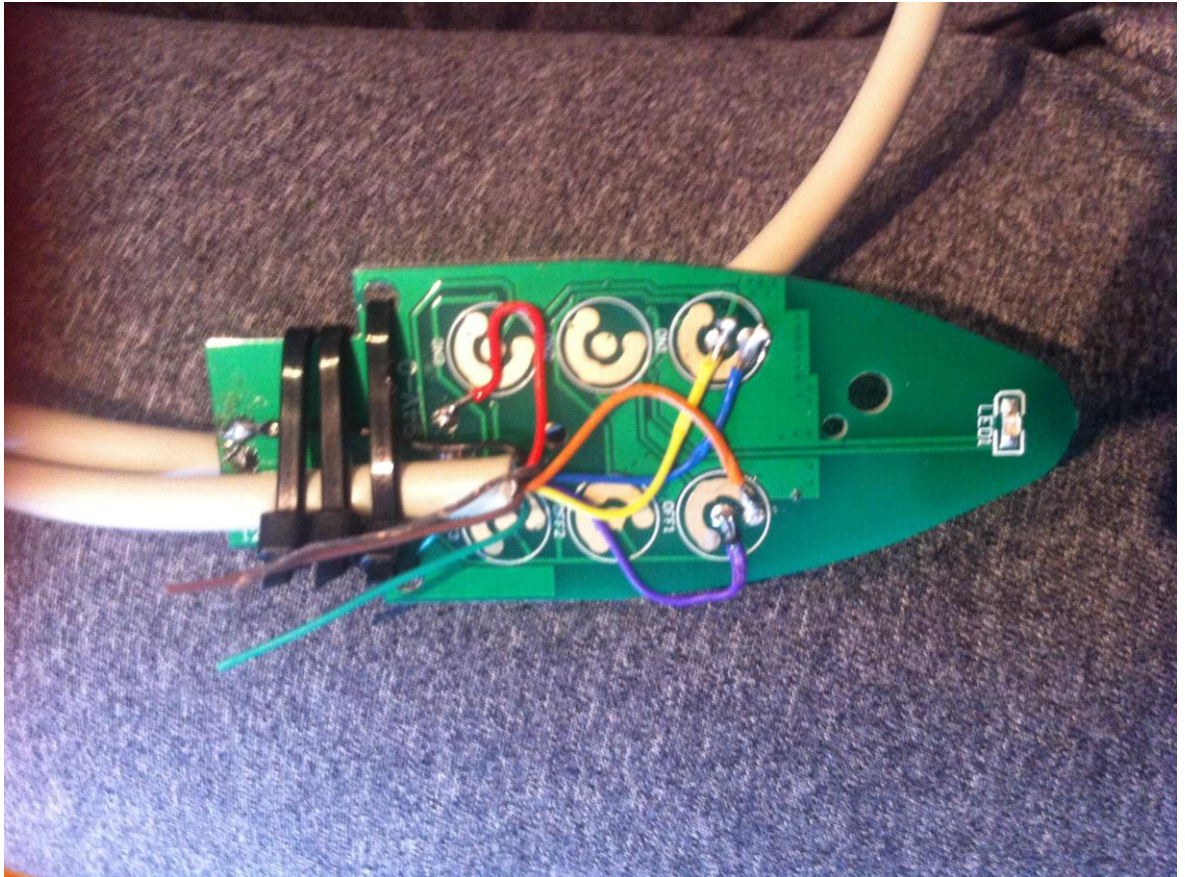


Figure 13. Where the wires are to the buttons (Source: photo by author)

6.3 Setting up music for alarm and web app to play

When the alarm starts, music is played. It is possible to create a playlist and store it locally; however, it would require capacity to be used on the SD card. Another way is to download an internet radio playlist, since the music is streamed online, very little capacity is needed to store a playlist file. For this project, a playlist is used from RadioTunes (radiotunes.com). RadioTunes is an online radio, where different playlists can be downloaded and listened through different music players. For this project, VLC player is used. In order to download VLC player an apt-get command needs to be executed from Terminal as a root user:

```
apt-get install vlc browser-plugin-vlc
```

Once VLC player is downloaded, a change to the binary file for the VLC player needs to be made, because VLC commands cannot be run with a root user privilege. The next command needs to be executed as a root user:

```
nano /usr/bin/vlc
```

There a word `geteuid`, a function that returns user ID, needs to be replaced with `getppid`, a function that returns the parent process ID of the process that is being called. Once the file is saved, VLC commands can now be accessible for a root user also. In the production environment a more elegant solution should be made as manually editing binary files may result in unwanted behaviour. (Alam, S. Hacker's Garage, VLC is not supposed to be run as root. Sorry – Solution, 2015)

In this project the playlist is also ran without a display, just through an SSH connection. In order to do this the following Bash command needs to be executed (once the playlist is downloaded onto the Pi machine):

```
cvlc --x11-display :0 RadioTunes-RootsReggae.pls
```

In this command `x11-display :0` is an X server for windowing GUI programs, while `display :0` tells GUI programs how to communicate with GUI, in this case that VLC is starting from display number 0 (the first display); `RadioTunes-RootsReggae.pls` is the name of the file of a streaming playlist.

6.4 Setting up Raspberry Pi's camera module for sensing movement

For this project an official Raspberry Pi camera module is used, which is needed to be connected into the CSI connector port, shown on figure 1. Once it is connected, Raspberry Pi needs to be updated, two commands need to be executed in Terminal as a root user:

```
apt-get update apt-get upgrade
```

Once all the necessary updates have been made to Raspberry Pi, a simple Bash command can be executed to take a photo from the camera and to check if it works:

```
raspistill -o image.jpg
```


There are two main commands used for Raspberry Pi camera module: `raspistill` for taking images, and `raspivid` for taking videos. These commands have some parameters that can be used with them:

- `-t 1000` for recording 1000 milliseconds (1 second) video, can be any amount
- `-vf` for flipping vertically
- `-hf` for flipping horizontally

(Raspberry Pi Foundation, Camera Module, 2015)

In this project, the camera is used for sensing motion by calculating pixels in the frame. In order to create such a script, first an image Python library needs to be installed as a root user:

```
apt-get install python-imaging-tk
```

Once the library is installed a Python script for sensing motion can be created. A full script is attached to Appendix 2. The script works so, that it watches for a motion, while piping an image from `raspistill` to analyse and process. Once the motion is detected it calls `raspistill` that takes a high-resolution image to the disk. The script then checks for the free space that is available, and if none is available it starts to erase old images. (Modified from Raspberry Pi Foundation, Lightweight python motion detection, 2013)

6.5 Creating a web app and controlling all the features with it

This chapter goes through basics of creating a web app for this project. The source code for this web app is available on GitHub (<https://github.com/miryanezvitskaya>) and in Appendix 3.

6.5.1 Creating home page

The main two pages for this project are `index.php` and `home.php`. These files need to be created in Nginx web server pages in `/usr/share/nginx/www`. There these two files need to be created, and also it is the location for downloading Bootstrap's Jumbotron. In this file, the previous two PHP files should be available as well: `turnon.php` and `turnoff.php`. These files are being used for turning on and off inside light with the

remote control attached to a relay board and a wireless outlet. All of the files created for the web app need to be kept in this folder.

6.5.2 Adding lights to be controlled by the web app

As stated previously, in Chapter 6.2, Lights, two PHP files were already modified for turning on and off inside lights. The same needs to be done for outside lights. In order to do this, it is needed to know in which pins the buttons on and off are connected from the remote control in the relay board. Once it is learnt, two new PHP files can be created, for example, `outsideon.php` and `outsideoff.php`. The syntax is the same as in turning inside lights on and off. In this project, the pin for turning on outside light is 24, while for turning off is 22:

```
<?php
    system(gpio mode 24 out);
    system(gpio write 24 1);
    sleep(1);
    system(gpio write 24 0);
?>
```

```
<?php
    system(gpio mode 22 out);
    system(gpio write 22 1);
    sleep(1);
    system(gpio write 22 0);
?>
```

Once it is tested that two PHP files work, a JavaScript function can be created in order to control these two files from the home page with buttons. (Tinkernut. Making Raspberry Pi Web Controls. 2015)

The syntax for turning on lights is as follows (outside or inside, depending on the PHP file that would be opened by JavaScript):

```

$(document).ready(function() {
    $('clickON').click(function() {
        var a = new XMLHttpRequest();
        a.open("GET", "turnon.php");
        a.onreadystatechange=function()
            if(a.readyState==function() {
                if (a.readyState==4) {
                    if(a.status == 200)
                }
            }
            else alert("HTTP error");
        }
    }
    a.send();
});
});

```

The same syntax goes for turning off the lights, though when JavaScript opens up a file, a PHP file needs to be addressed there for turning off the lights (a.open("GET", "turnoff.php")).

Once JavaScript functions are ready, two buttons can be created for turning on and off buttons in home page. In the body of a home file the following syntax needs to be put:

```

<button type="button" id="clickON">ON</button><br>
<button type="button" id="clickOFF">OFF</button><br>

```

Then the home page needs to be accessed by going into a web browser and typing IP address of Pi machine/home.php. Two buttons should be seen "ON" and "OFF", when clicked lights should go on and off. The same process is used for the outside lights, creating two JavaScript functions for reading PHP files for turning on and off files. Then the files are inserted into the button ID.

6.5.3 Temperature data to the web app

In order to read temperature data on the web app, first it needs to be collected and sent by a PHP file. (Modified script from Henrik, N. PHP Temperature Monitor. Raspberry Pi Foundation. 2013) A new PHP file needs to be created; there PHP opens the temperature

sensor, gets the data, transfers the data into human readable form (makes a decimal from the integer format as it is shown by the Terminal), and last, sends the data it got:

```
<?php
    $file = '/sys/devices/w1_bus_master1/28-
00043e0f55ff/w1_slave';
    $lines = file($file);
    $temp = explode('=', $lines[1]);
    $temp = number_format($temp[1] / 1000, 1, ',', '');
    echo $temp . " C";
?>
```

Then a JavaScript function needs to be created in home file in order to get the data from the PHP file, where first the interval is set as to how often the function gets the data from the PHP file (in this project every second):

```
var auto_refresh =
setInterval( function ()
{
    $('#load_tweets').load('readtempdata.php');
}, 1000);
```

In order to get this data to the home page, the ID of this JavaScript function needs to be inserted:

```
<label id="load_tweets"></label>
```

6.5.4 Controlling music through web app

In order to play music through the web app, the data needs to be sent into PHP file:

```
<?php
$outcome = shell_exec('cvlc --x11-display :0 RadioTunes-
RootsReggaps');
?>
```

6.5.5 Adding lock/unlock house to the web app

For locking and unlocking the house, a motion Python script is needed that was created in Chapter 6.4, Camera. (The full script is available in Appendix 2). For locking the house the JavaScript functions for turning off the lights are needed as well (in Chapter 6.5.4, Lights). In order for the Python script to be available on the web app, first a PHP file reads the data from the Python script, then a JavaScript function sets the interval to run the PHP file all the time, while the house is locked. In order to do so, first a PHP file needs to be created, where the data from a Python file is read:

```
<?php
    $result=system(python motion.py);
    exec($result);
?>
```

Now a JavaScript function needs to be created for getting data from a PHP file and also running the Python script constantly, while the lock house is on:

```
var auto_refresh = setInterval(
$(document).ready(function(){
$('motionON').click(function(){
var a = new XMLHttpRequest();
a.open("GET", "motionon.php");
a.onreadystatechange=function()
    if(a.readyState==function(){
        if (a.readyState==4){
            if(a.status == 200)
        }
        else alert("HTTP error");
    }
});
```

```
}  
}  
        a.send();  
});  
});1000);
```

6.5.6 Final configuration and hardware

The features can be expanded and improved, if needed. Once the project is complete, and the desired modifications are made, the system can go “online”. In the scope of this project, the system is only available on a local network. It is always available on a local network, as long as Raspberry Pi is on. Nginx server is known to sometimes cause problems and is needed to be restarted. In this case, `crontab` can be used to schedule nightly restart of Raspberry Pi and Nginx. Also it is recommended to not update and upgrade Raspberry Pi often, as new libraries and kernels can interact with the current ones that are used by the web app. If upgrade is required, it is recommended to back up all the files and spend some time carefully upgrading and updating the machine. If power outage is known to happen, a UPS (uninterruptible power supply) is recommended to get, so the web server would never go down. On Figure 14, a hardware of this embedded system can be seen. The web application’s full source code is available in Appendix 3. The screenshots of different pages of the web app on different devices are available in Appendix 4. The app is resizable and is in correspondence with the initial mock-up design of KotiPi in Chapter 5.2, Design of KotiPi. The logo of KotiPi has been added as well.

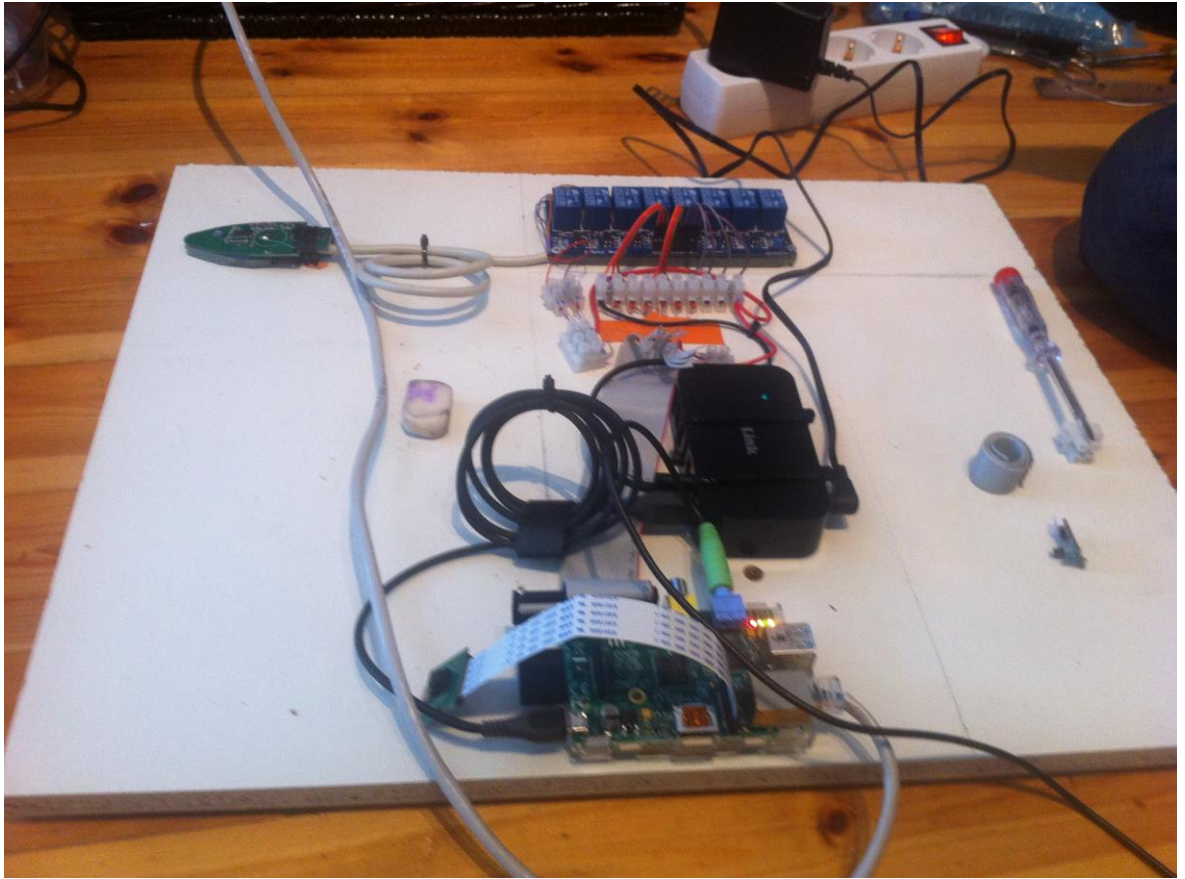


Figure 14. KotiPi hardware (Source: photo by author)

6.6 Testing

Since the project has quite a narrow scope, the automated debugging testing of the system is omitted this time, and the testing process compares only the intended results that were introduced in Chapter 3.3, Software requirements. During the creation of this project, each feature was tested individually to make sure it works, or which features are needed to be improved. The app is available on the local area network twenty-four seven. A user can easily turn on and off the outside and inside lights. The app shows temperature in live time, being updated every second. A user can set up an alarm, which once it's on, will turn on the inside light and start the playlist. A user can play a playlist anytime by turning the button on, or off if to turn off music. A user can lock or unlock the house.

6.6.1 Minor shortcomings

The sending of the email about the intruder does not work. In the first prototype, it is possible to send an email to an administrator, however, not to a user. In second prototype, this is the first part of the application that is prioritized. Second, because of such a short timeline, not all data is available in the history, only the temperature, name of sensors, and functions of relays. History tab in the app can be improved and developed further, for example, it can also contain all the photos that the web app takes.

7 Evaluation

This chapter describes the evaluation of results excluding testing that was done in Chapter 7, Testing. Furthermore, a full project evaluation is conducted, as well as ideas for future development are introduced.

7.1 Evaluation of results

For the home automation application, the testing can be found in previous Chapter. 7. However, that testing only covers the web app itself. It is also important to evaluate the results of the development as a whole. There were several minor shortcomings during the development, the main one being a complicated set up. It turned out to be much more challenging to write scripts in Bash and Python, then to make scripts accessible with PHP, and to create a user interface with JavaScript, HTML and CSS. For the next project, it would be recommended to use a Python framework, for example Django or Flask. However, such challenges can be considered an advantage if new things are being learnt constantly. Otherwise, the results of this project as a whole are quite satisfying; during the development new technologies have been learnt, and a prototype of a home automation application has been made.

7.2 Project evaluation

This project started as an idea to learn how to use basics of engineering embedded systems. It turned to be more time-consuming than originally planned. The work time was 400 hours. The main purpose of this project was to create a prototype of a home automation system from an idea. In order to realize such a project in real life and make it available to more users more time is needed, as well as knowledge. Only with thoroughness, forethought of logical actions, testing the system in practice in order to adapt it to real life such a system can become a turnkey solution.

During the process, a small survey was conducted (10 people, different age group) with the purpose to understand the chances of the prospect of realization of such a system. It is clearly quite common for people over 40 to be concerned with such a project, main reasons being safety and security of a home automation application. However, the younger age group agreed that home automation makes life easier and more enjoyable, since the daily tasks are outsourced to a machine.

7.3 Future development

Since the timeline of this project was tight, only a basic working prototype of Raspberry Pi home automation app was created. In future, more features and improvements can be made. First of all, it would be quite useful to set up a DDNS for this project. A DDNS is a dynamic domain name system that is updating a name server in DNS in real time automatically, allowing the user to type a URL. Currently, the system can be accessed only on a local network, but with DDNS it can be accessed anywhere. In particular, No-IP services can be used. They provide a free name or a custom name that can be bought for a small fee.

Currently, the system does not let the user to customize a lot of things. In particular, he/she can't change what music to play on the web app, how often a temperature sensor is updating its data, or what features are set for alarm or lock/unlock house. In future development, more freedom could be provided to a user, even with some administration tasks. The system becomes interesting for a user to use, when there are options to choose from.

Furthermore, better security can be introduced into such a system, particularly if the application would be accessible through public internet. For example, now the passwords are stored as plain text in the database; however, the passwords should be hashed and/or salted in the database, so they can't be stolen. Last, but not least, since Raspberry Pi acts as a web server in this project, it can also become a cloud, where a user can upload and store files.

8 Summary

The purpose of this project was to create a simple working open source Raspberry Pi home automation application, named KotiPi. During the process, some theoretical background as well as how to set up necessary environment, how to develop features of automation, and how to create a web app were presented. Anyone can create such a system from scratch following this project. The project also introduced the ideas for future development, and how to make such a project into a turnkey solution. Shortcomings of the project were also mentioned, so anyone can improve it, and develop it further.

Internet of Things is growing and expanding; it is important to follow new innovative technologies, however like Nikola Tesla said “the more we know, the more ignorant we become”, thus only by learning and continuing to grow “intellectual evolution is opening up of new and greater prospects”. (Nikola Tesla) Home automation simplifies everyday tasks and helps in freeing time for acquiring new knowledge and explore the world of unknown.

References

- Alam, S. (2015). *VLC is not supposed to run as root. Sorry. – Solution*. URL: www.hackersgarage.com/vlc-is-not-supposed-to-be-run-as-root-sorry-solution.html/
- Brainflakes. (2013). Raspberry Pi Foundation. *Lightweight Python Motion Detection*. URL: www.raspberrypi.org/forums/viewtopic.php?t=45235/
- Bullock, M. (2015). *How relays work*. URL: electronics.howstuffworks.com/relay.htm/
- Clauser, G. (2014) *New Trends in Home Automation*. URL: <http://www.electronichouse.com/daily/smart-home/new-trends-in-home-automation/>
- Hendricks, D. (2014). *The History of Smart Homes*. URL: www.iotevolutionworld.com/m2m/articles/376816-history-smart-homes.htm/
- Henrik, N. (2013) *PHP temperature monitor*. URL: www.raspberrypi.org/forums/viewtopic.php?t=64902&p=479148/
- Jain, P. (2015). *Embedded System*. URL: www.engineersgarage.com/articles/embedded-systems/
- Kirk, M. (2015). *Raspberry Pi Temperature Sensor*. URL: www.cl.cam.ac.uk/projects/raspberrypi/tutorials/temperature/
- O'Brien, J.A., and Marakas, G. (2005). *Introduction to Information Systems*. New York.
- Pater, T. (2015). *Connect your relay board to your Raspberry Pi*. URL: www.trafex.nl/2014/08/25/connect-a-relay-board-to-your-raspberry-pi/
- Purewal, S. (2014) *Learning Web App Development*. O'Reilly.
- Raspberry Pi Foundation. (2015). *GPIO: Raspberry Pi Models A and B*. URL: www.raspberrypi.org/documentation/usage/gpio/

Raspberry Pi Foundation. (2015). *Camera Configuration*. URL:
<https://www.raspberrypi.org/documentation/configuration/camera.md/>

Tinkernut. (2015). *Making Raspberry Pi Web Controls*. URL:
www.youtube.com/watch?v=EAMLwbShFFQ/

Appendices

Appendix 1. Nginx Configuration

```
1 # You may add here your
2 # server {
3 # ...
4 # }
5 # statements for each of your virtual hosts to this file
6 # You should look at the following URL's in order to grasp a solid understanding
7 # of Nginx configuration files in order to fully unleash the power of Nginx.
8 # http://wiki.nginx.org/Pitfalls
9 # http://wiki.nginx.org/QuickStart
10 # http://wiki.nginx.org/Configuration
11 #
12 # Generally, you will want to move this file somewhere, and start with a clean
13 # file but keep this around for reference. Or just disable in sites-enabled.
14 #
15 # Please see /usr/share/doc/nginx-doc/examples/ for more detailed examples.
16 ##
17 server {
18     #listen 80; ## listen for ipv4; this line is default and implied
19     #listen [::]:80 default_server ipv6only=on; ## listen for ipv6
20     #lisays php:ta varten
21     listen 80;
22     server_name $domain_name;
23     #root /var/www;
24     index index.html index.htm;
25     access_log /var/log/nginx/access.log;
26     error_log /var/log/nginx/error.log;
27     location ~\.php$ {
28         fastcgi_pass unix:/var/run/php5-fpm.sock;
29         fastcgi_split_path_info ^(.+\.(php|\.*)$);
30         fastcgi_index index.php;
31         fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
32         fastcgi_param HTTPS off;
33         try_files $uri =404;
34         include fastcgi_params;
35     }
```

```
36 root /usr/share/nginx/www;
37 index index.html index.htm index.php;
38 # Make site accessible from http://localhost/
39 server_name localhost;
40 location / {
41     # First attempt to serve request as file, then
42     # as directory, then fall back to displaying a 404.
43     try_files $uri $uri/ /index.html;
44     # Uncomment to enable naxsi on this location
45     # include /etc/nginx/naxsi.rules
46 }
47 location /doc/ {
48     alias /usr/share/doc/;
49     autoindex on;
50     allow 127.0.0.1;
51     allow ::1;
52     deny all;
53 }
54 # Only for nginx-naxsi used with nginx-naxsi-ui : process denied requests
55 #location /RequestDenied {
56 # proxy_pass http://127.0.0.1:8080;
57 #}
58 #error_page 404 /404.html;
59 # redirect server error pages to the static page /50x.html
60 #
61 #error_page 500 502 503 504 /50x.html;
62 #location = /50x.html {
63 # root /usr/share/nginx/www;
64 #}
65 # pass the PHP scripts to FastCGI server listening on 127.0.0.1:9000
66 #
67 #location ~ /\.php$ {
68 # fastcgi_split_path_info ^(.+\.(php|php5))(/.+)$;
69 # # NOTE: You should have "cgi.fix_pathinfo = 0;" in php.ini
70 #
```

```
71  # # With php5-cgi alone:
72  # fastcgi_pass 127.0.0.1:9000;
73  # # With php5-fpm:
74  # fastcgi_pass unix:/var/run/php5-fpm.sock;
75  # fastcgi_index index.php;
76  # include fastcgi_params;
77  #}
78  # deny access to .htaccess files, if Apache's document root
79  # concurs with nginx's one
80  #
81  #location ~ /\.ht {
82  # deny all;
83  #}
84  }
85  # another virtual host using mix of IP-, name-, and port-based configuration
86  #
87  #server {
88  # listen 8000;
89  # listen somename:8080;
90  # server_name somename alias another.alias;
91  # root html;
92  # index index.html index.htm;
93  #
94  # location / {
95  #   try_files $uri $uri/ =404;
96  # }
97  #}
98  # HTTPS server
99  #
100 #server {
101 # listen 443;
102 # server_name localhost;
103 #
104 # root html;
105 # index index.html index.htm;
```

```
105 # index index.html index.htm;
106 #
107 # ssl on;
108 # ssl_certificate cert.pem;
109 # ssl_certificate_key cert.key;
110 #
111 # ssl_session_timeout 5m;
112 #
113 # ssl_protocols SSLv3 TLSv1;
114 # ssl_ciphers ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv3:+EXP;
115 # ssl_prefer_server_ciphers on;
116 #
117 # location / {
118 #     try_files $uri $uri/ =404;
119 # }
120 #}
```

Appendix 2. Python Camera Motion

```
1  #!/usr/bin/python
2  import StringIO
3  import subprocess
4  import os
5  import time
6  from datetime import datetime
7  from PIL import Image
8  threshold = 10
9  sensitivity = 20
10 forceCapture = True
11 forceCaptureTime = 60 * 60 # Once an hour
12 filepath = "/home/pi/www/picam"
13 filenamePrefix = "capture"
14 diskSpaceToReserve = 40 * 1024 * 1024 # Keep 40 mb free on disk
15 cameraSettings = ""
16 # settings of the photos to save
17 saveWidth = 1296
18 saveHeight = 972
19 saveQuality = 15 # Set jpeg quality (0 to 100)
20 # Test-Image settings
21 testWidth = 100
22 testHeight = 75
23 # this is the default setting, if the whole image should be scanned for changed pixel
24 testAreaCount = 1
25 testBorders = [ [[1,testWidth],[1,testHeight]] ]
26 debugMode = False # False or True
27 # Capture a small test image (for motion detection)
28 def captureTestImage(settings, width, height):
29     command = "raspistill %s -w %s -h %s -t 200 -e bmp -n -o -" % (settings, width, height)
30     imageData = StringIO.StringIO()
31     imageData.write(subprocess.check_output(command, shell=True))
32     imageData.seek(0)
33     im = Image.open(imageData)
34     buffer = im.load()
35     imageData.close()
```

```

36     return im, buffer
37 # Save a full size image to disk
38 def saveImage(settings, width, height, quality, diskSpaceToReserve):
39     keepDiskSpaceFree(diskSpaceToReserve)
40     time = datetime.now()
41     filename = filepath + "/" + filenamePrefix + "-%04d%02d%02d-%02d%02d%02d.jpg" % (time.year, time.month, time.day, time.hour, time.minute, time.second)
42     subprocess.call("raspistill %s -w %s -h %s -t 200 -e jpg -q %s -n -o %s" % (settings, width, height, quality, filename), shell=True)
43     print "Captured %s" % filename
44 # Keep free space above given level
45 def keepDiskSpaceFree(bytesToReserve):
46     if (getFreeSpace() < bytesToReserve):
47         for filename in sorted(os.listdir(filepath + "/")):
48             if filename.startswith(filenamePrefix) and filename.endswith(".jpg"):
49                 os.remove(filepath + "/" + filename)
50                 print "Deleted %s/%s to avoid filling disk" % (filepath,filename)
51                 if (getFreeSpace() > bytesToReserve):
52                     return
53 # Get available disk space
54 def getFreeSpace():
55     st = os.statvfs(filepath + "/")
56     du = st.f_bavail * st.f_frsize
57     return du
58 # Get first image
59 image1, buffer1 = captureTestImage(cameraSettings, testWidth, testHeight)
60 # Reset last capture time
61 lastCapture = time.time()
62 while (True):
63     # Get comparison image
64     image2, buffer2 = captureTestImage(cameraSettings, testWidth, testHeight)
65     # Count changed pixels
66     changedPixels = 0
67     takePicture = False
68     if (debugMode): # in debug mode, save a bitmap-file with marked changed pixels and with visible testarea-borders
69         debugImage = Image.new("RGB", (testWidth, testHeight))
70         debugIm = debugImage.load()

```

```

71     for z in xrange(0, testAreaCount): # = xrange(0,1) with default-values = z will only have the value of 0 = only one scan-area = whole picture
72     for x in xrange(testBorders[z][0][0]-1, testBorders[z][0][1]): # = xrange(0,100) with default-values
73         for y in xrange(testBorders[z][1][0]-1, testBorders[z][1][1]):
74             if (debugMode):
75                 debugIm[x,y] = buffer2[x,y]
76                 if ((x == testBorders[z][0][0]-1) or (x == testBorders[z][0][1]-1) or (y == testBorders[z][1][0]-1) or (y == testBorders[z][1][1]-1)):
77                     debugIm[x,y] = (0, 0, 255) # in debug mode, mark all border pixel to blue
78                 # Just check green channel as it's the highest quality channel
79                 pixdiff = abs(buffer1[x,y][1] - buffer2[x,y][1])
80                 if pixdiff > threshold:
81                     changedPixels += 1
82                 if (debugMode):
83                     debugIm[x,y] = (0, 255, 0) # in debug mode, mark all changed pixel to green
84                 # Save an image if pixels changed
85                 if (changedPixels > sensitivity):
86                     takePicture = True # will shoot the photo later
87                 if ((debugMode == False) and (changedPixels > sensitivity)):
88                     break # break the y loop
89                 if ((debugMode == False) and (changedPixels > sensitivity)):
90                     break # break the x loop
91                 if ((debugMode == False) and (changedPixels > sensitivity)):
92                     break # break the z loop
93     if (debugMode):
94         debugImage.save(filepath + "/debug.bmp") # save debug image as bmp
95         print "debug.bmp saved, %s changed pixel" % changedPixels
96 # Check force capture
97 if forceCapture:
98     if time.time() - lastCapture > forceCaptureTime:
99         takePicture = True
100 if takePicture:
101     lastCapture = time.time()
102     saveImage(cameraSettings, saveWidth, saveHeight, saveQuality, diskSpaceToReserve)
103 # Swap comparison buffers
104 image1 = image2
105 buffer1 = buffer2#!/usr/bin/python

```

Appendix 3. Web App Source Code

```
,  
1  /* Space out content a bit */  
2  body {  
3    padding-top: 20px;  
4    padding-bottom: 20px;  
5    background-color: #e4ecf9;  
6  }  
7  
8  label {  
9    font-weight: normal;  
10 }  
11  
12 button {  
13   padding: none;  
14   border: none;  
15   background: none;  
16 }  
17 /* Everything but the jumbotron gets side spacing for mobile first views */  
18 .header,  
19 .marketing,  
20 .footer {  
21   padding-right: 15px;  
22   padding-left: 15px;  
23 }  
24  
25 /* Custom page header */  
26 .header {  
27   padding-bottom: 20px;  
28   border-bottom: 1px solid #5a6374;  
29 }  
30 /* Make the masthead heading the same height as the navigation */  
31 .header h3 {  
32   margin-top: 0;  
33   margin-bottom: 0;  
34   line-height: 40px;  
35 }
```

```

37  /* Custom page footer */
38  .footer {
39    padding-top: 19px;
40    color: #777;
41    border-top: 1px solid #5a6374;
42  }
43
44  /* Customize container */
45  @media (min-width: 768px) {
46    .container {
47      max-width: 730px;
48    }
49  }
50  .container-narrow > hr {
51    margin: 30px 0;
52  }
53
54  /* Main marketing message and sign up button */
55  .jumbotron {
56    text-align: left;
57    background-color: #fff;
58  }
59  .jumbotron .btn {
60    padding: 14px 24px;
61    font-size: 21px;
62  }
63
64  /* Supporting marketing content */
65  .marketing {
66    margin: 40px 0;
67  }
68  .marketing p + h4 {
69    margin-top: 28px;
70  }
71

```

```
72  /* Responsive: Portrait tablets and up */
73  @media screen and (min-width: 768px) {
74    /* Remove the padding we set earlier */
75    .header,
76    .marketing,
77    .footer {
78      padding-right: 0;
79      padding-left: 0;
80    }
81    /* Space out the masthead */
82    .header {
83      margin-bottom: 30px;
84    }
85    /* Remove the bottom border on the jumbotron for visual effect */
86    .jumbotron {
87      border-bottom: 0;
88    }
89  }
```

```
1 body {
2   padding-top: 40px;
3   padding-bottom: 40px;
4   background-color: #e4ecf9;
5 }
6
7 .form-signin {
8   max-width: 330px;
9   padding: 15px;
10  margin: 0 auto;
11 }
12 .form-signin .form-signin-heading,
13 .form-signin .checkbox {
14   margin-bottom: 10px;
15 }
16 .form-signin .checkbox {
17   font-weight: normal;
18 }
19 .form-signin .form-control {
20   position: relative;
21   height: auto;
22   -webkit-box-sizing: border-box;
23   -moz-box-sizing: border-box;
24   box-sizing: border-box;
25   padding: 10px;
26   font-size: 16px;
27 }
28 .form-signin .form-control:focus {
29   z-index: 2;
30 }
31 .form-signin input[type="login"] {
32   margin-bottom: -1px;
33   border-bottom-right-radius: 0;
34   border-bottom-left-radius: 0;
35 }
36 .form-signin input[type="password"] {
37   margin-bottom: 10px;
38   border-top-left-radius: 0;
39   border-top-right-radius: 0;
40 }
41
```

```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="utf-8">
5     <meta http-equiv="X-UA-Compatible" content="IE=edge">
6     <meta name="viewport" content="width=device-width, initial-scale=1">
7     <meta name="description" content="KotiPi web application">
8     <meta name="author" content="Mirya Nezvitskaya">
9     <link rel="icon" href="../../favicon.ico">
10
11     <title>KotiPi</title>
12
13     <!-- Bootstrap core CSS -->
14     <link href="css/bootstrap.min.css" rel="stylesheet">
15
16     <!-- Custom styles for this template -->
17     <link href="css/signin.css" rel="stylesheet">
18
19     <!-- sql.js for reading database -->
20     <script src="js/sql.js"></script>
21     <script>
22       var xhr = new XMLHttpRequest();
23       xhr.open('GET', '/home/pi/pi_database.sqlite', true);
24       xhr.responseType = 'arraybuffer';
25
26       xhr.onload = function(e) {
27         var uInt8Array = new Uint8Array(this.response);
28         var db = new SQL.Database(uInt8Array);
29         var contents = db.exec("SELECT * FROM user");
30       };
31     xhr.send();
32
33   </script>
34
35   <script language="javascript">
```



```

36     function pasuser(form) {
37         if (form.login.value=="admin") {
38             if (form.pswd.value=="penis") {
39                 location="home.php"
40             } else {
41                 alert("Invalid Password")
42             }
43             } else { alert("Invalid UserID")
44             }
45         }
46     </script>
47
48 </head>
49 <body>
50
51     <div class="container">
52
53         <form class="form-signin">
54             <h2 class="form-signin-heading">Please sign in</h2>
55             <label for="inputLogin" class="sr-only">Login</label>
56             <input name="login" type="login" id="inputLogin" class="form-control" placeholder="Login" required autofocus>
57             <label for="inputPassword" class="sr-only">Password</label>
58             <input name = "pswd" type="password" id="inputPassword" class="form-control" placeholder="Password" required>
59             <div class="checkbox">
60                 <label>
61                     <input type="checkbox" value="remember-me"> Remember me
62                 </label>
63             </div>
64             <button class="btn btn-lg btn-primary btn-block" type="submit" onClick="pasuser(this.form)">Sign in</button>
65         </form>
66     </div>
67     <!-- IE10 viewport hack for Surface/desktop Windows 8 bug -->
68     <script src="js/ie10-viewport-bug-workaround.js"></script>
69 </body>
70 </html>

```

```

1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="utf-8">
5     <meta http-equiv="X-UA-Compatible" content="IE=edge">
6     <meta name="viewport" content="width=device-width, initial-scale=1">
7     <meta name="description" content="KotiPi web application">
8     <meta name="author" content="Mirya Nezvitskaya">
9
10    <title>KotiPi</title>
11
12    <!-- Bootstrap core CSS -->
13    <link href="css/bootstrap.min.css" rel="stylesheet">
14
15    <!-- Custom styles for this template -->
16    <link href="css/home.css" rel="stylesheet">
17
18    <!-- Script for buttons-->
19    <script src="http://code.jquery.com/jquery-1.11.0.min.js"></script>
20    <!--script for inside light-->
21    <script type="text/javascript">
22      $(document).ready(function(){
23        $('#clickON1').click(function(){
24          var a = new XMLHttpRequest();
25          a.open("GET", "insidelighton.php");
26          a.onreadystatechange=function(){
27            if (a.readyState==4){
28              if(a.status == 200){
29                }
30              else alert("HTTP error");
31            }
32          }
33          a.send();
34        });
35

```

```
36 });
37 $(document).ready(function(){
38 $('#clickOFF1').click(function(){
39     var a = new XMLHttpRequest();
40     a.open("GET", "insidelightoff.php");
41     a.onreadystatechange=function(){
42         if (a.readyState==4){
43             if(a.status == 200){
44             }
45             else alert("HTTP error")
46         }
47     }
48     a.send();
49 });
50
51 });
52 <!--script for outside light-->
53     $(document).ready(function(){
54         $('#clickON2').click(function(){
55             var a = new XMLHttpRequest();
56             a.open("GET", "outsidelighton.php");
57             a.onreadystatechange=function(){
58                 if (a.readyState==4){
59                     if(a.status == 200){
60                     }
61                     else alert("HTTP error");
62                 }
63             }
64             a.send();
65         });
66
67     });
68
69     $(document).ready(function(){
70         $('#clickOFF2').click(function(){
```

```

71     a.open("GET", "outsidelightoff.php");
72     a.onreadystatechange=function(){
73         if (a.readyState==4){
74             if(a.status == 200){
75     }
76             else alert("HTTP error")
77     }
78     }
79     a.send();
80 });
81
82 });
83     <!--script for speakers-->
84     $(document).ready(function(){
85         $('#clickON3').click(function(){
86     var a = new XMLHttpRequest();
87     a.open("GET", "musicon.php");
88     a.onreadystatechange=function(){
89         if (a.readyState==4){
90             if(a.status == 200){
91     }
92             else alert("HTTP error");
93     }
94     }
95     a.send();
96 });
97
98 });
99     $(document).ready(function(){
100     $('#clickOFF3').click(function(){
101     var a = new XMLHttpRequest();
102     a.open("GET", "musicoff.php");
103     a.onreadystatechange=function(){
104         if (a.readyState==4){
105

```

```
106 }
107         else alert("HTTP error")
108 }
109 }
110         a.send();
111 });
112
113 });
114 </script>
115
116 <!--script for alarm clock-->
117 <script type="text/javascript">
118     var sound = new Audio("/home/pi/www/RadioTunes-RootsReggae.pls");
119     getID = function(value){
120         return document.getElementById(value);
121     }
122     var hour = getID("hour"),
123         minute = getID("minute"),
124         h = getID("h"),
125         m = getID("m"),
126         aswitch = getID("switch"),
127         off = getID("turnoff"),
128         refreshtime = 600,
129         alarmtimer = null;
130     aswitch.On = false;
131     aswitch.value = "OFF";
132     function alarmonoff(){
133         switch(aswitch.On)
134         {
135             case false:
136                 aswitch.On = true;
137                 aswtich.value = "ON";
138                 alarmset();
139                 break;
```

```

140     case true:
141         aswitch.On = false;
142         aswitch.value = "OFF";
143     }
144 }
145 function disablealarm(){
146     sound.pause();
147     off.style.display = "none";
148 }
149 function alarmplay(){
150     if (aswitch.On)
151     {
152         off.style.display = "block";
153         sound.play();
154     }
155     else
156         alert("There was an error.");
157 }
158 function alarmset(){
159     clearTimeout(alarmtimer);
160     var tomorrow = false;
161     if (hour.value < h.value)
162         {tomorrow = true;}
163     else if (hour.value == h.value && minute.value < m.value)
164         {tomorrow = true;}
165     var date = new Date(), year = date.getFullYear(), month = date.getMonth(), day = parseInt (date.getDate());
166     if (tomorrow){day += 1;}
167     time = new Date (year, month, day, hour.value, minute.value, second.value, date.getMilliseconds());
168     time = (time - (new Date()));
169     if (alarmswitch.On = false)
170         alarmswitch();
171     alarmtimer = setTimeout(function(){alarmplay();},parseInt(time));
172 }
173 timeRefresh = function(){
...

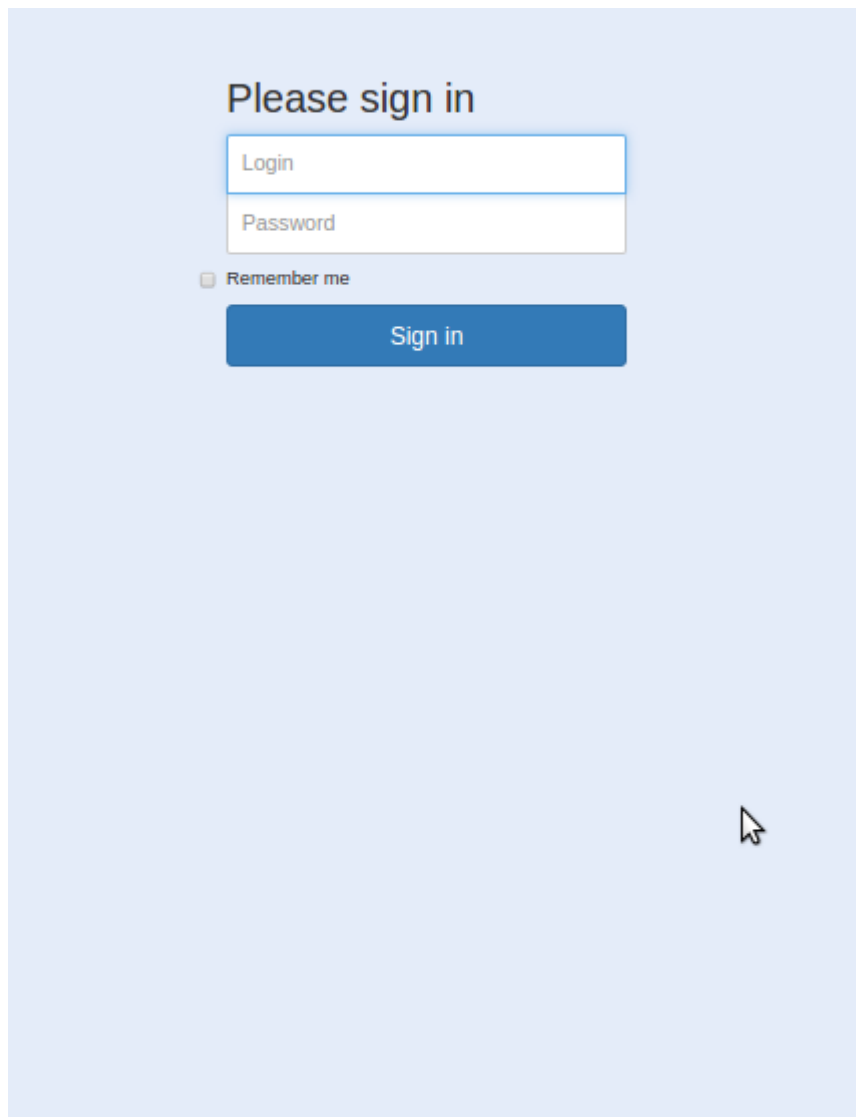
```

```

175     h.innerHTML = date.getHours();
176     h.value = h.innerHTML;
177     m.innerHTML = date.getMinutes();
178     m.value = m.innerHTML;
179     setTimeout("timeRefresh()", refreshTime);
180 };
181 numCap = function( obj, min, max){
182     obj.value = Math.max(obj.min, Math.min(obj.max, obj.value) );
183     alarmSet();// Starts up the alarm automatically when a value is changed.
184 };
185 timerefresh();
186
187     var a = getID("hour");
188     a.value = h.innerHTML;
189     a = getID("minute");
190     a.value = m.innerHTML;
191 </script>
192
193 <!--show temperature every second refreshes temperature-->
194 <script type="text/javascript" src="http://code.jquery.com/jquery-latest.js"></script>
195 <script type="text/javascript">
196     var auto_refresh = setInterval(function(){
197         $('#showtemp').load('tempshow.php');
198     }, 1000);
199 </script>
200
201 </head>
202 <body>
203
204 <div class="container">
205     <div class="header clearfix">
206         
207
208
209
210
211     <div class="jumbotron">
212         <h2><span class="glyphicon glyphicon-time" aria-hidden="true"></span> Wake Up <input id="hour" type="number" min="0" max="23" onChange="numCap(this)" />
213     </div>
214     <div class="jumbotron">
215         <h2><span class="glyphicon glyphicon-music" aria-hidden="true"></span> Music play</h2>
216     </div>
217     <div class="jumbotron">
218         <h2><span class="glyphicon glyphicon-lamp" aria-hidden="true"></span> Lights In<button type="button" id="clickON1"> ON </a>
219 <button type="button" id="clickOFF1"> OFF </button>Lights Out<button type="button" id="clickON2"> ON </a> <button type="button" id="clickOFF2"> OFF </button>
220     </div>
221     <div class="jumbotron">
222         <h2><span class="glyphicon glyphicon-home" aria-hidden="true"></span> Lock House <button type="button"> ON </a> <button type="button"> OFF </button><
223     </div>
224     <div class="jumbotron">
225         <h2><span class="glyphicon glyphicon-facetime-video" aria-hidden="true"></span> Video<button type="button"> take photo </a> </button></h2>
226     </div>
227     <div class="jumbotron">
228         <h2><span class="glyphicon glyphicon-cloud" aria-hidden="true"></span> Temperature <label id="showtemp"></label></h2>
229     </div>
230     <div class="jumbotron">
231         <h2><span class="glyphicon glyphicon-calendar" aria-hidden="true"></span> History</h2>
232     </div>
233     <footer class="footer">
234         <p>&copy; KotiPi 2015</p>
235     </footer>
236 </div> <!-- /container -->
237 <!-- IE10 viewport hack for Surface/desktop Windows 8 bug -->
238 <script src="../../assets/js/ie10-viewport-bug-workaround.js"></script>
239 </body>
240 </html>

```


Appendix 4. Screenshots of application on different devices





🕒 Wake Up OFF


🎵 Music play


💡 Lights In ON OFF Lights Out ON OFF 


🏠 Lock House ON OFF

Please sign in

 Remember me

 Video take photo

 Temperature 20,4 C

 History

© KotiPi 2015

