

Databases Coursework

"Angel Kanchev" University of Ruse

Erasmus+ Romanian students (Pitești)

Nicolescu Mihai-Robert

Marin Marian Puiu

Ene Remus Ionut

Nedelea Eugen Cristian

Oprea Alexandra Catalina

Year: 2

Lecturer: Irena Valova

[Github repository of the project](#)



What is the Raspberry PI ?

Raspberry Pi is the name of a series of single-board computers made by the Raspberry Pi Foundation, a UK charity that aims to educate people in computing and create easier access to computing education.

The Raspberry Pi launched in 2012, and there have been several iterations and variations released since then. The original Pi had a single-core 700MHz CPU and 256MB RAM, and the latest model has a quad-core 1.4GHz CPU with 1GB RAM. The main price point for Raspberry Pi has always been \$35 and all models have been \$35 or less, including the Pi Zero, which costs just \$5.

The model we have, the Raspberry PI 3 model B+ is the final revision of the third-generation single-board computer. It has a 1.4GHz 64-bit quad-core processor, dual-band wireless LAN, Bluetooth 4.2/BLE, faster Ethernet, and Power-over-Ethernet support (with separate PoE HAT).

All over the world, people use Raspberry Pis to learn programming skills, build hardware projects, do home automation, and even use them in industrial applications.

The Raspberry Pi is a very cheap computer that runs Linux, but it also provides a set of GPIO (general purpose input/output) pins that allow you to control electronic components for physical computing and explore the Internet of Things (IoT).

Setting it up



The USB Micro power supply goes into port #1 (in the image),
The HDMI cable goes into port #2,
The mouse/keyboard go into ports #3 and #4

Installing an OS

As for software, I formatted the SD Card using [SD Card Formatter from Canakit](#) into the FAT32 file system format.

Then I downloaded [Raspbian OS from the RaspberryPi site](#). As of the making of this project, the OS version was September 2019.



Raspberry Pi Desktop

After this, I installed the OS onto the SD card using [Flash Image Writer from Etcher](#).

Most of the work on the RPI was done through using a [Remote Desktop Connection](#) between our PCs and the RPI.

Setting up the sensors and leds

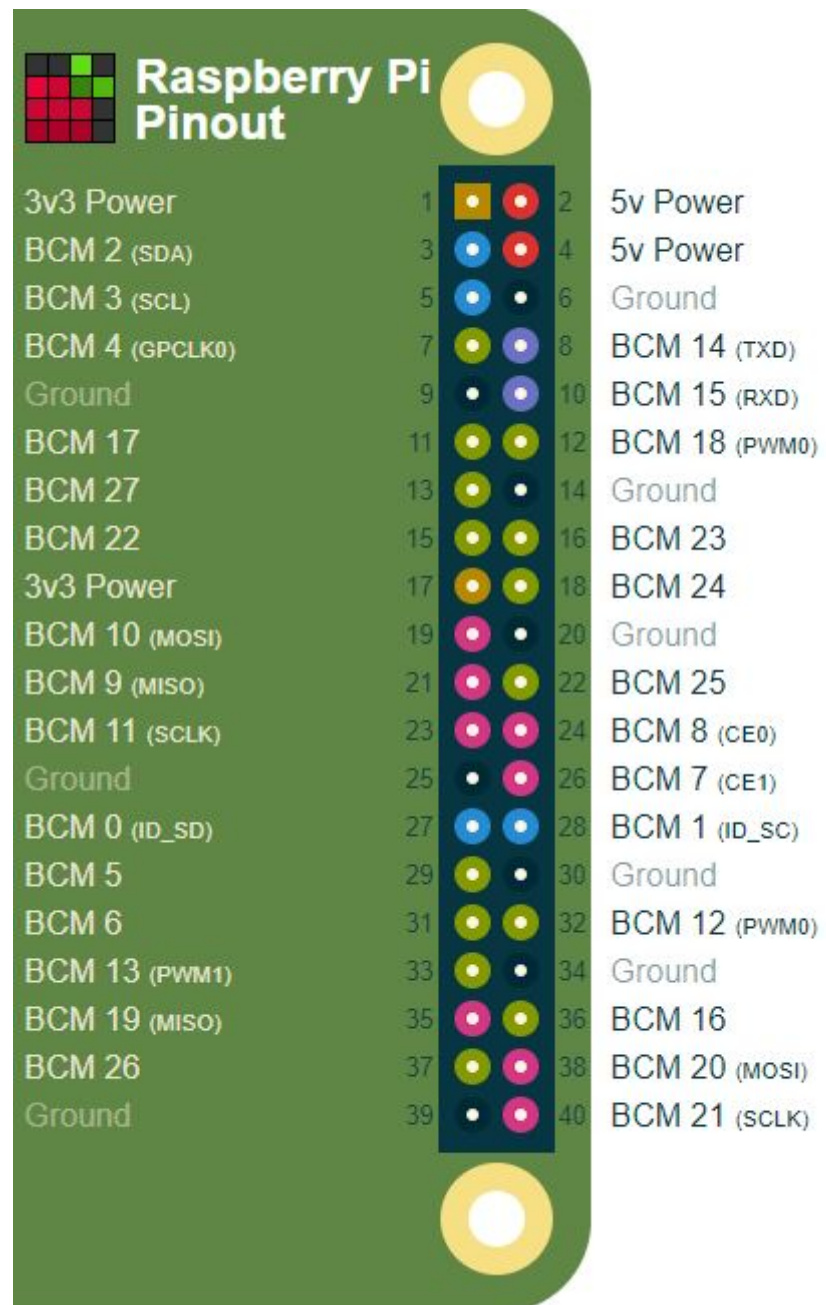
After installing Raspbian, we had to connect the sensors/leds by using the GPIO/I2C [pins](#) on the raspberry pi.

Sensors:

- Pins 4 and 6 were used to connect the cooler our RPI had.
- Pin 2 was used as power (+Vcc) for the pair of sensors (Red wire)
- Pin 3 was used to connect the SDA wire (Orange wire)
- Pin 5 was used to connect the SCL wire (Yellow wire)
- Pin 3 was used to connect the SDA wire
- Pin 9 was used to connect the GND(ground) (Black wire)

LEDs:

- Pin 34 was used as GND(ground) for the LEDs (Black wire)
- Pin 39 was also used as GND(ground) for the LEDs (Purple wire)
- Pin 36 is where the Blue wire goes
- Pin 12 is where the other Blue wire goes (the one next to the green LED)
- Pin 10 is where the other Purple wire goes
- Finally, the White wire goes into pin #15



Information about the sensors

BH1750

With the BH1750 Light Sensor, intensity can be directly measured by the lux meter without needing to make calculations. The data which is output by this sensor is directly output in Lux (Lx). When objects which are lighted in homogeneous get the 1 lx luminous flux in one square meter, their light intensity is 1lx. Sometimes to take good advantage of the illuminant, you can add a reflector to the illuminant. So that there will be more luminous flux in some directions and it can increase the illumination of the target surface.

For example:

- night: 0.001-0.02;
- moonlight night: 0.02-0.3;
- cloudy indoor: 5-50;
- cloudy outdoor: 50-500;
- sunny indoor: 100-1000;
- under the sunlight in summer afternoon: about 10^6 power;
- reading books for intensity of illumination: 50-60;
- home video standard intensity of illumination: 1400.

BME280

The BME280 is as combined digital humidity, pressure and temperature sensor based on proven sensing principles. This precision sensor from Bosch is the best low-cost sensing solution for measuring humidity with $\pm 3\%$ accuracy, barometric pressure with ± 1 hPa absolute accuracy, and temperature with $\pm 1.0^\circ\text{C}$ accuracy. Because pressure changes with altitude, and the pressure measurements are so good, you can also use it as an altimeter with ± 1 meter accuracy.

The most important part of this project was reading and displaying data from these sensors. We did this using python3, the code is in smartmod.py.

Reading the data from BME280 was pretty easy (with the help of

`import adafruit_bme280`), needing only 3 lines of code:

```
i2c = busio.I2C(board.SCL, board.SDA)
bme280 =
adafruit_bme280.Adafruit_BME280_I2C(i2c)
list = [bme280.temperature, bme280.humidity,
bme280.pressure]
```

Reading data from BH1750 was also easy since we found [this GitHub repo showing how to connect to it](#).

In the first instance of the project, the script also handled the LEDs, that's why the code is there.

This is the main function, which "prepares" a GPIO pin to output information. `GPIO.HIGH` is used to output full light, `GPIO.LOW` is used to stop outputting the light. In this case, the LED only stays "awake" for a half of a second

```
def turnLEDOn(GPno):
    GPIO.setup(GPno, GPIO.OUT)
    #print("LED on")
    GPIO.output(GPno, GPIO.HIGH)
    time.sleep(0.5)
    #print("LED off")
    GPIO.output(GPno, GPIO.LOW)
```

Finally, `solution()` puts the list of data into a string where the numbers are separated by a comma (e.g: 11.67,23.6,62.9,1010.6) and the lines of code afterwards put this string into a file every 3 seconds.

- 11.67 is the light level
- 23.6 is the temperature
- 62.9 is the humidity
- 1010.6 is the pressure

```
def solution():
    list = SensCheck()
    my_string = ','.join(map(str, list))
    return(my_string)

1.def fisier():
2.    f=open('date.txt','w+')
3.    lista = solution()
4.    #print(lista)
5.    f.write(lista)
6.    f.close()
7.
8.while (True):
9.    fisier()
10.    time.sleep(3)
```


The logic

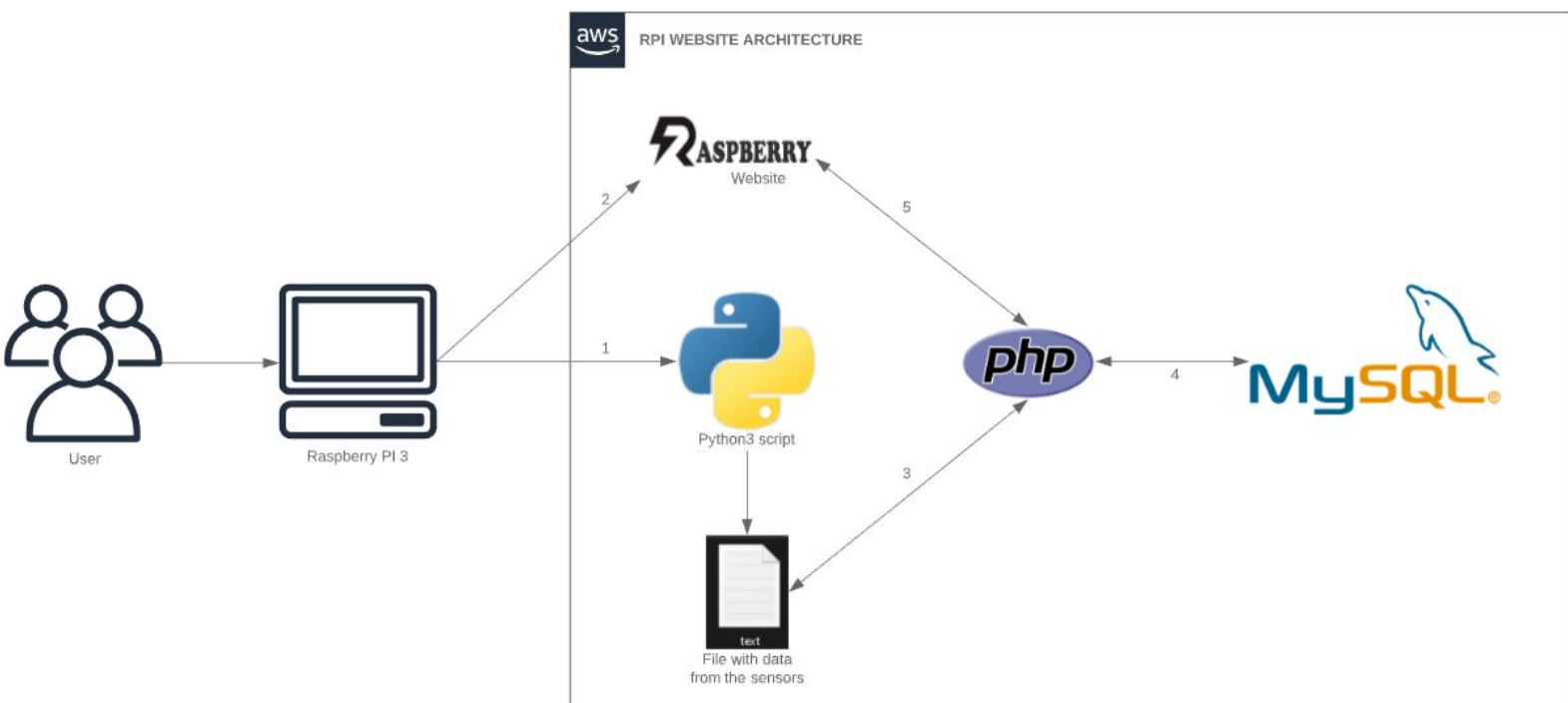
The python script outputs the data we need every 3 seconds in a file(to avoid data repetition in the datasets table when using the site rapidly).

The reason we had to use this file was because we've encountered some problems with the permissions when trying to run the python script right from the website.

Then, using PHP we put that data in a table in the local database, and then we show the table on the page.

The website compares the most recently read data from the table with the bounds set by admin and decides if the LEDs should turn on or off. Data is inserted into the table every time the page refreshes, so we told the page to refresh every 300 seconds (5 minutes).

I tried to explain the logic using this UML diagram:



Turning the leds on/off is simple, here's a snippet of the php code where the red led turns on if the temperature reaches a certain threshold:

```
$sql_temp = "select * from ledstates";
$result = mysqli_query($connectare,$sql_temp);
$row = $result -> fetch_assoc();
```

```
if($temperature >= $row['redLed'])
{
    system("gpio -g mode 15 out");
    system("gpio -g write 15 1"); //red led on
}
else
{
    system("gpio -g write 15 0"); //red led off
}
```

Just like in the python example, `system("gpio -g mode 15 out");` uses the terminal to tell the led with GPIO#15 to prepare for outputting light, and then `system("gpio -g write 15 1");` writes 1 to its memory to turn it on.

Further on, `system("gpio -g write 15 0");` writes 0 to the GPIO pin and turns off the light.

The database diagram

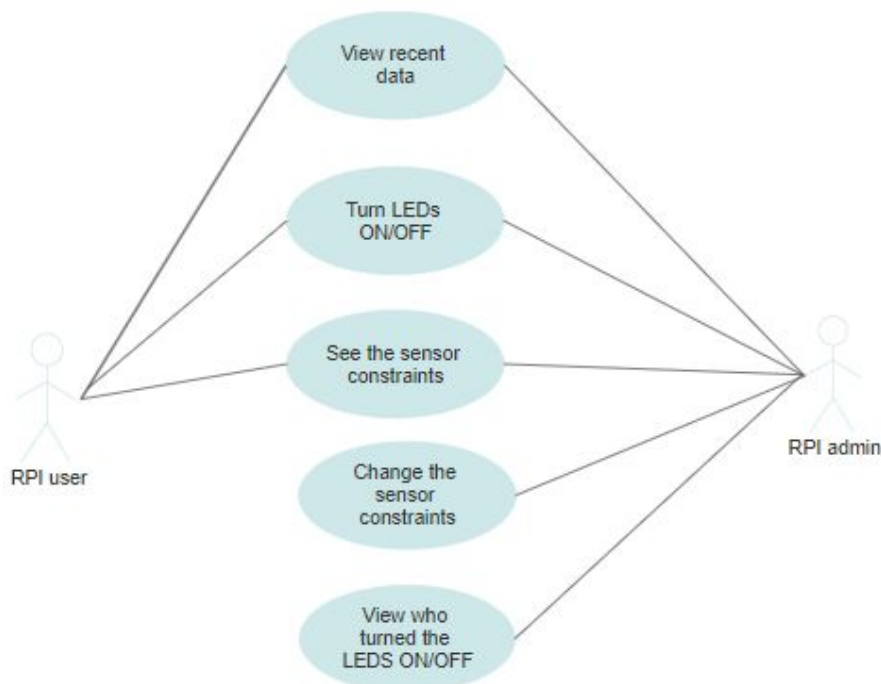
The database is pretty simple with only four tables.

bazabazei leds	
id : int(255)	
Username : varchar(50)	
LedName : varchar(50)	
Date : datetime	

bazabazei ledstates	
redLed : float	
greenLed : float	
yellowLed : float	
whiteLed : float	

bazabazei users	
userid : int(10)	
Username : varchar(266)	
Password : varchar(266)	
Name : varchar(266)	
Email : varchar(266)	

bazabazei datasets	
id : int(11)	
data : datetime	
temperature : float	
humidity : float	
pressure : float	
light : float	



The use case diagram

As seen here, there are some differences between what the normal user and the admin can do.

Mainly, the admin can change when the sensors turn on and see who turned the sensors on/off

How to set up the application

Setting up the application is pretty easy, you only need to do two things:

1. cd to where the script is and run it in the terminal using

```
cd /var/www/html/r3f
python3 smartmod.py
```

2. Go to the website located at

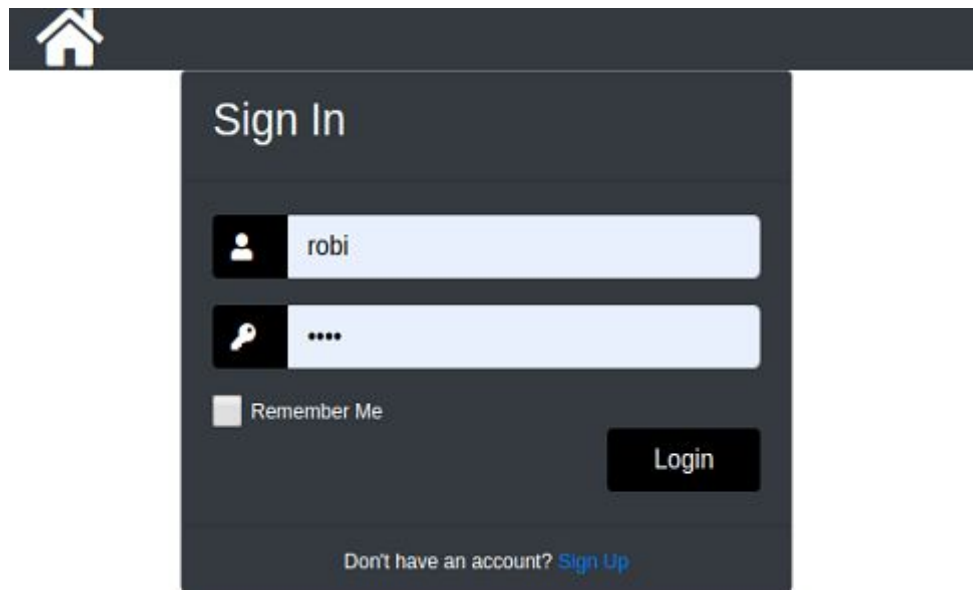
<http://localhost/r3f/index.php>
and log-in


That's it ! Just let the terminal and the website page run in the background and your smart home device is ready to go !

The front-end

The front-end is pretty straight-forward. Thanks to bootstrap the website looks really good and is easy to use.

Here are some screenshots from the site (Login page, main user page, admin page)
User page:



Welcome, robi !

Log Out!

Temperature > 20°C

Most recent reading:
24°C

Turn on! Turn off!

Humidity > 90%

Most recent reading:
85.4%

Turn on! Turn off!

Pressure > 1000hPa

Most recent reading:
1016.2 hPa

Turn on! Turn off!

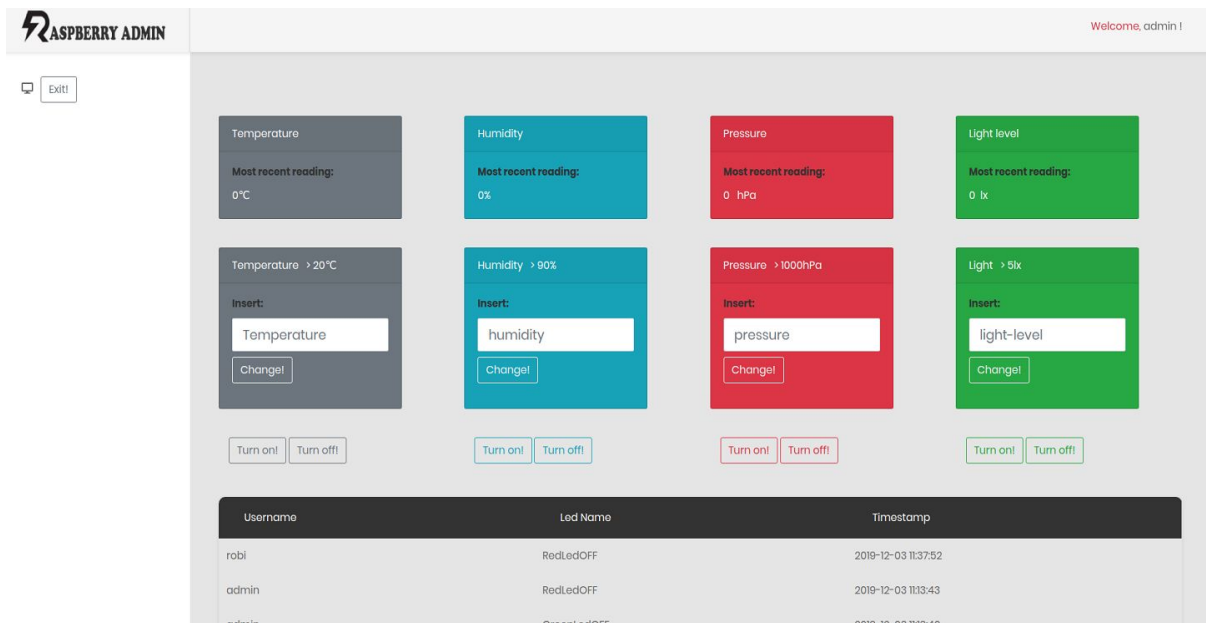
Light > 5lx

Most recent reading:
0 lx

Turn on! Turn off!

Temperature	Humidity	Pressure	Light Level	Timestamp
24	85.4	1016.2	0	2019-12-03 11:50:21
24	85.3	1016.1	0	2019-12-03 11:49:19
24	85.3	1016.1	0	2019-12-03 11:44:17
24	85.1	1016.1	0	2019-12-03 11:39:15
24	85.1	1016.1	0	2019-12-03 11:37:52
24	84.8	1016.1	0	2019-12-03 11:34:47
24	84.3	1016.1	0	2019-12-03 11:29:46
24	84.5	1016.1	0	2019-12-03 11:29:35

Admin page:



The back-end

For the back-end we used an apache2 local server and PHP/MYSQL for the database management system.

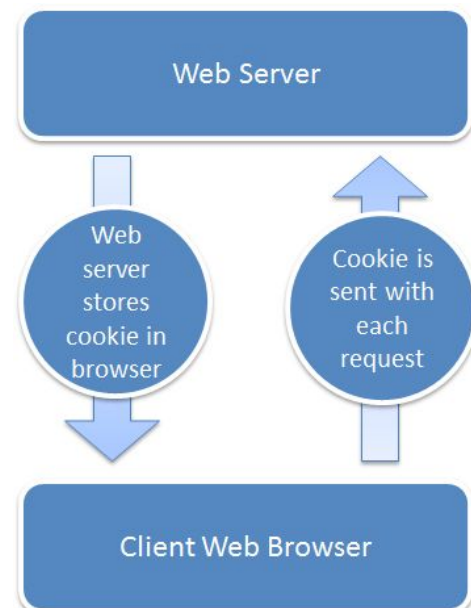
Information about how to do this is detailed [here](#).

The most interesting thing about the back-end is that we've used cookies to be able to turn multiple leds on/off.

The process isn't complicated at all, we basically have a cookie set for every led (it keeps track of its current state).

When you login, all the cookies are "unset". You can "set" them by pressing the turn on/off buttons. While the cookie is set, the specific led doesn't turn on/off until you press the reset button.

Also, we've made the site in such a way that when you log out of the site, the leds are automatically unset.



Features

The possibility of having separate admin and user account with hashed passwords

+ Options

	userid	Username	Password	Name	Email
<input type="checkbox"/> Edit Copy Delete	1	marian	\$2y\$10\$rlw9/Ev5XVcDptw0T2zu6OLSkUX9IW50GULjCoGJzmv...	marian	marian@yahoo.com
<input type="checkbox"/> Edit Copy Delete	2	admin	\$2y\$10\$2Pljw3xElQ1FQjUihWB7aefaN5wAh8pypuuCzLuh26y...	admin	admin
<input type="checkbox"/> Edit Copy Delete	3	robi	\$2y\$10\$6q6Tu57alaeIMVTyPqhE/.il3tJ/wdJ8PnlKjDuUEYq...	robi	robi
<input type="checkbox"/> Edit Copy Delete	4	r3nudinn0rd	\$2y\$10\$wTaD.loYMJMucyOh0SBZe1Y/#Pn/K3okUzqugtAlz...	R3nu	remusene69@gmail.com

Change the bounds at which the sensors turn on (as an admin)

<p>Temperature > 20°C</p> <p>Insert:</p> <input type="text" value="Temperature"/> <input type="button" value="Change!"/>	<p>Humidity > 90%</p> <p>Insert:</p> <input type="text" value="humidity"/> <input type="button" value="Change!"/>	<p>Pressure > 1000hPa</p> <p>Insert:</p> <input type="text" value="pressure"/> <input type="button" value="Change!"/>	<p>Light > 5lx</p> <p>Insert:</p> <input type="text" value="light-level"/> <input type="button" value="Change!"/>
--	---	---	---

Turn the leds on/off until you reset them

<p>Temperature > 20°C</p> <p>Most recent reading:</p> <p>23.9°C</p> <p><input type="button" value="Turn on!"/> <input <="" p="" type="button" value="Turn off!"/></p>	<p>Humidity > 90%</p> <p>Most recent reading:</p> <p>84.4%</p> <p><input type="button" value="Turn on!"/> <input <="" p="" type="button" value="Turn off!"/></p>	<p>Pressure > 1000hPa</p> <p>Most recent reading:</p> <p>1010.1 hPa</p> <p><input type="button" value="Turn on!"/> <input <="" p="" type="button" value="Turn off!"/></p>	<p>Light > 5lx</p> <p>Most recent reading:</p> <p>0 lx</p> <p><input type="button" value="Turn on!"/> <input <="" p="" type="button" value="Turn off!"/></p>
--	---	--	---

Know who turned leds on/off using PHP sessions

Username	Led Name	Timestamp
admin	RedLedOFF	2019-12-03 11:13:43
admin	GreenLedOFF	2019-12-03 11:13:40
robi	GreenLedOFF	2019-12-03 11:13:06

Mobile ready

The interface displays four sensor cards in a 2x2 grid. Each card has a header with a unit and a threshold, a 'Most recent reading' section, and two 'Turn on!'/'Turn off!' buttons below. The sensors are Temperature (grey), Humidity (blue), Pressure (red), and Light (green). Below the cards is a table with five columns: Temperature, Humidity, Pressure, Light Level, and Timestamp. The table contains five rows of data.

Temperature	Humidity	Pressure	Light Level	Timestamp
24	84.3	1016.1	0	2019-12-03 11:29:46
24	84.5	1016.1	0	2019-12-03 11:29:35
24	84.5	1015.7	0	2019-12-03 11:29:18
23.9	84.4	1016.1	0	2019-12-03 11:28:41
23.9	84.3	1016.1	0	2019-12-03 11:26:24

See what user you're logged in as using PHP sessions

Welcome, robi !

See the current set bounds as any user (e.g. here the temperature has to be higher than 20°C to turn on).

The Temperature sensor card shows the header 'Temperature > 20°C', the 'Most recent reading:' section, and the value '24°C'.

See the most recent 10 readings of the sensors

Temperature	Humidity	Pressure	Light Level	Timestamp
24	84.3	1016.1	0	2019-12-03 11:29:46
24	84.5	1016.1	0	2019-12-03 11:29:35
24	84.5	1015.7	0	2019-12-03 11:29:18
23.9	84.4	1016.1	0	2019-12-03 11:28:41
23.9	84.3	1016.1	0	2019-12-03 11:26:24
23.9	84.4	1016.1	0	2019-12-03 11:26:18
23.9	84.7	1016.1	0	2019-12-03 11:23:17
23.8	84.6	1015.8	0	2019-12-03 11:13:43
23.8	84.5	1016.1	0	2019-12-03 11:13:40
23.8	84.6	1016.1	0	2019-12-03 11:13:29

Other features:

- The ability to turn multiple leds on/off for an undetermined amount of time (using cookies)