Verslag

IOT LED / TEMPRATUUR – PHP REST API

Laravel – php

Jesse Helmes

2IAO3B

Inhoud

Beschrijving opstelling

Mijn project is het lezen van tempratuur intern en extern. leds die je uit en aan kunt zetten en ook in morse code kunt laten knipperen.

De tempraturen worden elke 30 seconde refreshed en je kunt de geschiedenis van de cpu tempratuur ophalen.

Ik heb de api in de framework laravel gedaan en alleen daarvoor heb ik alleen php voor gebruikt.

de grafische kant heb ik met HTML, CSS en JQUERY gedaan. Dit draait op mijn eigen laptop hierin kun je de leds los van elkaar uit en aan zetten, morse code versturen, de tempraturen zien en de geschiedenis van de CPU temratuur ophalen.

Gebruikte hardware

in dit project heb ik de volgende hardware onderdelen gebruit:

- Raspberry Pi 3 Model B+
- WiringPi prototype board
- breadboard
- DS18B20 temperatuur sensor
- 1x rode LED
- 1x groene LED
- Aantal weerstanden (2x 220, 4k7 Ohm)

Hoe te gebruiken

Je moet eerst met de raspberry pi verbinden en inloggen dan ga je naar de map api met de command 'cd /var/www/api' dan doe je sudo php artisan serve --host=<ip adres> voor mij was dit 192.168.178.86 vergeet niet om ‘sudo’ te doen want anders heb je geen permissie voor de leds.

Je kunt de api afsluiten door de ‘Ctrl + c’ te doen in de terminal.

Uitleg code met snippets

Api routings

Dit verbind de url met de juist class en functie

```
Route::get('/temperature', 'TemperatureController@temperature');
Route::get('/temperature/cpu', 'TemperatureController@cpu');
Route::get('/temperatures', 'TemperatureController@temperatures');

Route::get('/leds', 'LedController@leds');
Route::post('/leds', 'LedController@led');
Route::post('/leds/morse', 'LedController@morse');
```

Omgeving tempratuur ophalen

In deze functie werd de tempratuur van de omgeving op gevraagt (DS18B20 temperatuur sensor)

```php
public function temperature(){

    //activeerd gpio and therm
    exec('modprobe w1-gpio');
    exec('modprobe w1-therm');

    //leest de gegevens van de sensor die aangesloten is
    $base_dir = '/sys/bus/w1/devices/';
    $device_folder = glob($base_dir . '28*')[0];
    $device_file = $device_folder . '/w1_slave';
    $data = file($device_file, FILE_IGNORE_NEW_LINES);

    $temp = null;
    $zin;
    if (preg_match('/YES$/', $data[0])) {

        if (preg_match('/t=(\d+)$/', $data[1], $matches,
            PREG_OFFSET_CAPTURE)) {
            $temp = round($matches[1][0] / 1000, 1);
        }

    }

    if ($temp) {
        $zin = "Temperature is " . $temp . "C\n";
    } else {
        $zin = "Unable to get temperature\n";
    }

    $json = '{"celsius":'.$temp.'}';
    //echo $zin;
    return json_decode($json, true);
}
```

Cpu tempratuur ophalen

Hier word te tempratuur van de cpu opgehaald en in de database gestopt

```php
public function cpu(){
    //leest de intern tempratuur van de cpu bestand
    $f = fopen("/sys/class/thermal/thermal_zone0/temp","r");
    $temp = (float)fgets($f);
    $temp = round($temp/1000, 1);
    //echo 'CPU temperature is ' . $temp . "°C";
    fclose($f);

    $json = '{"celsius":'.$temp.'}';

    //hier voeg ik de gegevens toe aan de database
    $arr = array('temperature' => $temp, 'date' => date("Y-m-d"));
    $this->store($arr);

    return json_decode($json, true);
}
```

Tempratuur model

Dit is de tempratuur model die gebruikt werd voor het opslaan en ophalen van de geschiedenis.

```php
class Temperature extends Model
{
    use HasFactory;

    //protected $table = 'temperatures';

    protected $fillable = [
        'temperature',
        'date'
    ];
}
```

Cpu tempratuur geschiedenis

Deze functie haalt de tempratuur geschiedenis op van de cpu

```php
public function temperatures(){
    //pakt de laaste temperatuur die opgeslagen is als eerst op
    $temps = Temperature::latest()->get();
    return response()->json($temps);
}

//persist the new resource
public function store($temp){
    $temperature = Temperature::create($temp);
}

//validation
protected function validateTemperature(){
    return request()->validate([
        'temperature' => 'required'
    ]);
}
```

Leds ophalen

Deze functie haalde alle pins op die je kon besturen wat bij deze project leds waren en boven in is de array met pins van de raspberry pi zelf die je nodig hebt om de goede pin aan te sturen. In deze geval nummer 0 en 1 op de WiringPi prototype board.

```
////Module 5 - Solderen Prototype board - KBE2020.pdf
public $leds = array(
            17, 18
        );

public function leds(){
    exec('modprobe w1-gpio');

    $gpio = new GPIO();

    $json = array();

    for($i = 0; $i < count($this->leds); $i++){
        $pin = $this->leds[$i];

        $gpio->setup($pin, "out");
        $gpio->output($pin, 1);

        //ik gebruik $i nummer wegens de nummers op de pi
        $json[] = ["id" => $i, "pin" => $pin, "status" => 1];
    }

    return response()->json($json);
}
```

Leds aansturen

In deze functie kun je met de in komende json post de leds aansturen let wel op 1 is uit en 0 is aan. Het pakt namelijk de id van de json en de status en dan gevolgens kijkt die bij welke pin de id hoort en gebruikt dat om de pin aan te sturen.

```
public function led(Request $request){
    exec('modprobe w1-gpio');

    //zet inkomende json om naar parameters
    $id = $request->input('id');
    $status = $request->input('status');

    $gpio = new GPIO();
    $pin = $this->leds[$id];

    $gpio->setup($pin, "out");
    $gpio->output($pin, $status);

}
```

Led morse code.

Dit was een extra idee waar bij de tekst die je mee geeft word omgezet naar morse code en dan daar heen werd geloopt om alle leds aan en uit te zetten, de tijd dat ze aan bleven hing af van de lengte van de streepje.

```
public function morse(Request $request){
    exec('modprobe w1-gpio');
    $gpio = new GPIO();

    $message = strtolower($request->input('message'));
    $message = str_split($message);
    $ledCount = count($this->leds);

    //zet eerst alles uit
    for($i = 0; $i < $ledCount; $i++){
        $pin = $this->leds[$i];

        $gpio->setup($pin, "out");
        $gpio->output($pin, 1);
    }

    //https://morsecode.world/international/translator.html
    //morse code tabel
    $morseTable = ["a" => ".-", "b" => "-...", "c" => "-.-.", "d" => "
        -..",
            "e" => ".", "f" => "..-.", "g" => "--.", "h" => "....",
            "i" => "..", "j" => ".---", "k" => "-.-", "l" => ".-..",
            "m" => "--", "n" => "-.", "o" => "---", "p" => ".--.",
            "q" => "--.-", "r" => ".-.", "s" => "...", "t" => "-",
            "u" => "..-", "v" => "...-", "w" => ".--", "x" => "-..-",
            "y" => "-.--", "z" => "--..", "0" => "-----", "1" => "
                .----",
            "2" => "..---", "3" => "...--", "4" => "....-", "5" => "
                .....",
            "6" => "-....", "7" => "--...", "8" => "---..", "9" => "
                ----.",
            " " => " ", "-" => "-", "." => ".", "?" => "..--..",
            "!" => "-.-.--", "," => "--..--"
    ];
```

```
    ////convert text naar morse want dat is minder internet verkeer
    $morse = "";
    for($i = 0; $i < count($message); $i++){
        //if(in_array($message[$i], $morseTable)){
            $morse .= $morseTable[$message[$i]];
            //echo $message[$i];
        //}
    }
    $morse = str_split($morse);

    //voert code uit
    $unit = 1024;

    for($x = 0; $x < count($morse); $x++){
        $dot = $morse[$x];
        if($dot == "-"){
            for($i = 0; $i < $ledCount; $i++){
                $gpio->output($this->leds[$i], 0);
            }
            usleep($unit * 300);//$unit * 300
        }
        else if($dot == "."){
            for($i = 0; $i < $ledCount; $i++){
                $gpio->output($this->leds[$i], 0);
            }
            usleep($unit * 20);//$unit * 10
        }else{
            for($i = 0; $i < $ledCount; $i++){
                $gpio->output($this->leds[$i], 1);
            }
            usleep($unit * 200);
        }
        for($i = 0; $i < $ledCount; $i++){
            $gpio->output($this->leds[$i], 1);
        }
        usleep(1500000);
    }

}
```

Front end

Gegevens ophalen op load

Hier werden alle gegevens opgehaald op load en werden de hoofd html items die aangepast werden aangemaakt voor een const, om de 30 seconde werd de tempraturen vernieuwd. Ook heb ik hier een dark mode aan toegevoegd met een tijd en mijn naam omdat ik van dark mode hou.

```
    //school
    //const baseApi = "http://10.9.6.104:8000/api/";

    //thuis frits box
    const baseApi = "http://192.168.178.86:8000/api/";

    //belangerijk html classen die aangepast worden
    const cpu = document.querySelector('.cpu');
    const omgeving = document.querySelector('.omgeving');
    const leds = document.querySelector('.leds');
    const temperatures = document.querySelector('.temperatures');

    const morseCode = document.querySelector('.morse-code');


    $(document).ready(function(){
        url = window.location.href;

        //auto naar dark mode
        time = (new Date()).getHours();
        if(time >= 18 || url.search("jesse")){
            darkMode();
        }

        //leds knoppen
        getData('leds', setLeds);

        getData('temperature/cpu', setCPU);
        getData('temperature', setTemp);

        //list
        getData('temperatures', setTemps);

        //refresh elke 30 secs
        setInterval(function(){
            getData('temperature/cpu', setCPU);
            getData('temperature', setTemp);

        }, 30000);
    });
```

Json short get method

Zoals de comments al zegt het verkort de code gebruik in de project

```
//verkort ajax code herhalen
function getData(api, func){
    $.ajax({
        url: baseApi + api
    }).done(func);
}
```

Tempratuur geschiedenis lijst

Deze functie maakt een tabel aan op basis van de inkomende gegevens van de history van de cpu en weergeeft het op de scherm.

```
function setTemps(data){

    var table = "<tr> <th>cpu temperature</th> <th>datum + tijd</th> <th>id</th> </tr>";
    var max = 30;//data.length

    for(var i = 0; i < 30; i++){
            table += "<tr>" +
                "<td>" + data[i].temperature + " °C</td>" +
                "<td>" + data[i].time + "</td>" +
                "<td>" + data[i].id + "</td>" +
            "</tr>";
    }

    temperatures.innerHTML = table;
}
```

Setleds functie

Deze functie maakt de knoppen aan voor de leds op basis van de leds de het binnen krijgt, het gebruikt de status om te bepalen of het een uit of aan knop is en voegt de id aan de knop toe zodat het weet over welke knop het ging die je later gebruik voor de post functie. De knop veranderd zelfs van kleur als die aan of uit staat.

```
function setLeds(data){
    console.log(data);

    var led = "";
    var ledStatus = "", id = 0;

    for(var i = 0; i < data.length; i++){
        ledStatus = (data[i].status == 0) ? 'off' : 'on';
        id = data[i].id;

        //background on / off color
        var color = (ledStatus == "off") ? "green" : "red";
        $(this).css("background", color);

        led += '<button id="led' + id + '" ledStatus="' + ledStatus + '" style="background:'+ color + '"> turn led ' + id + " " + ledStatus + '</button>';
    }
    //console.log(leds);
    leds.innerHTML = led;
}
```

Led action

In deze functie word de led aan en uit gezet door de status te versturen en de id.

```
//ledAction
$(leds).on('click', 'button', function(){
    var id = parseInt($(this).attr("id").replace("led", ""));
    var ledStatus = $(this).attr("ledStatus");
    ledStatus = (ledStatus == 'off') ? 1 : 0;

    var json = {"id": id, "status": ledStatus };
    json = JSON.stringify(json);
    console.log(json);

    $.ajax({
        type: "POST",
        url: baseApi + 'leds',
        dataType: 'JSON',
        data: json,
        headers: {
            "Content-Type": "application/json"
        },
    }).done(function(){
        console.log("post led");
    });

    updateLed($(this));
});
```

Updaten van de led status kleur

Hier word de led in de pagina die je aan of uit gezet hebt up-ge-date met de kleur groen of rood

```
function updateLed(led){
    var id = led.attr("id").replace("led", "");
    var ledStatus = (led.attr("ledStatus") == 'on') ? 'off' : 'on';

    //background on / off color
    var color = (ledStatus == "off") ? "green" : "red";
    $(led).css("background", color);

    led.attr("ledStatus", ledStatus);
    led.html("turn led " + id + " " + ledStatus);
}
```

Cpu tempratuur

Hier word de tempratuur op de pagina gezet en krijgt het een kleur op basis van de warmte blue is koud, orange is normaal en red is heet.

```
function setCPU(data){
    //console.log(data);
    var celsius = data.celsius;
    var color = "";

    cpu.innerHTML = data.celsius;

    //kleur text met temperature
    if(celsius < 40)
        color = "blue";
    else if(celsius >= 40 && celsius < 60){
        color = "orange";
    }else{
        color = "red";
    }

    $(cpu).css("color", color);

}
```

Omgeving tempratuur

Zelfde als de cpu tempratuur

```
function setTemp(data){
    //console.log(data);
    var celsius = data.celsius;
    var color = "";

    omgeving.innerHTML = celsius;

    //kleur text met temperature
    if(celsius < 20)
        color = "blue";
    else if(celsius >= 20 && celsius < 30){
        color = "orange";
    }else{
        color = "red";
    }

    $(omgeving).css("color", color);
}
```

Morse code

In deze functie word de bericht die je in de textarea voor de morse code getypt hebt gebruikt en verstuurt naar de leds. De leds worden ook verborgen om aan te geven dat ze bezig zijn.

```
$(".sendMorse").on("click", function(){
    morse();
});

function morse(){
    //verkomt dat je op knop kunt blijven drukken door het te verbergen
    $(leds).css( "visibility", "hidden" );

    message = $(morseCode).val();

    var json = {"message": message};
    json = JSON.stringify(json);
    console.log(json);

    $.ajax({
        type: "POST",
        url: baseApi + 'leds/morse',
        dataType: 'JSON',
        data: json,
        headers: {
            "Content-Type": "application/json"
        }
    }).done(function(){
        console.log("post morse code");

        //simple leds reset
        //getData('leds', setLeds);
    }).always(function(){
        $(leds).css( "visibility", "visible" );
    });
}
```

Dark mode

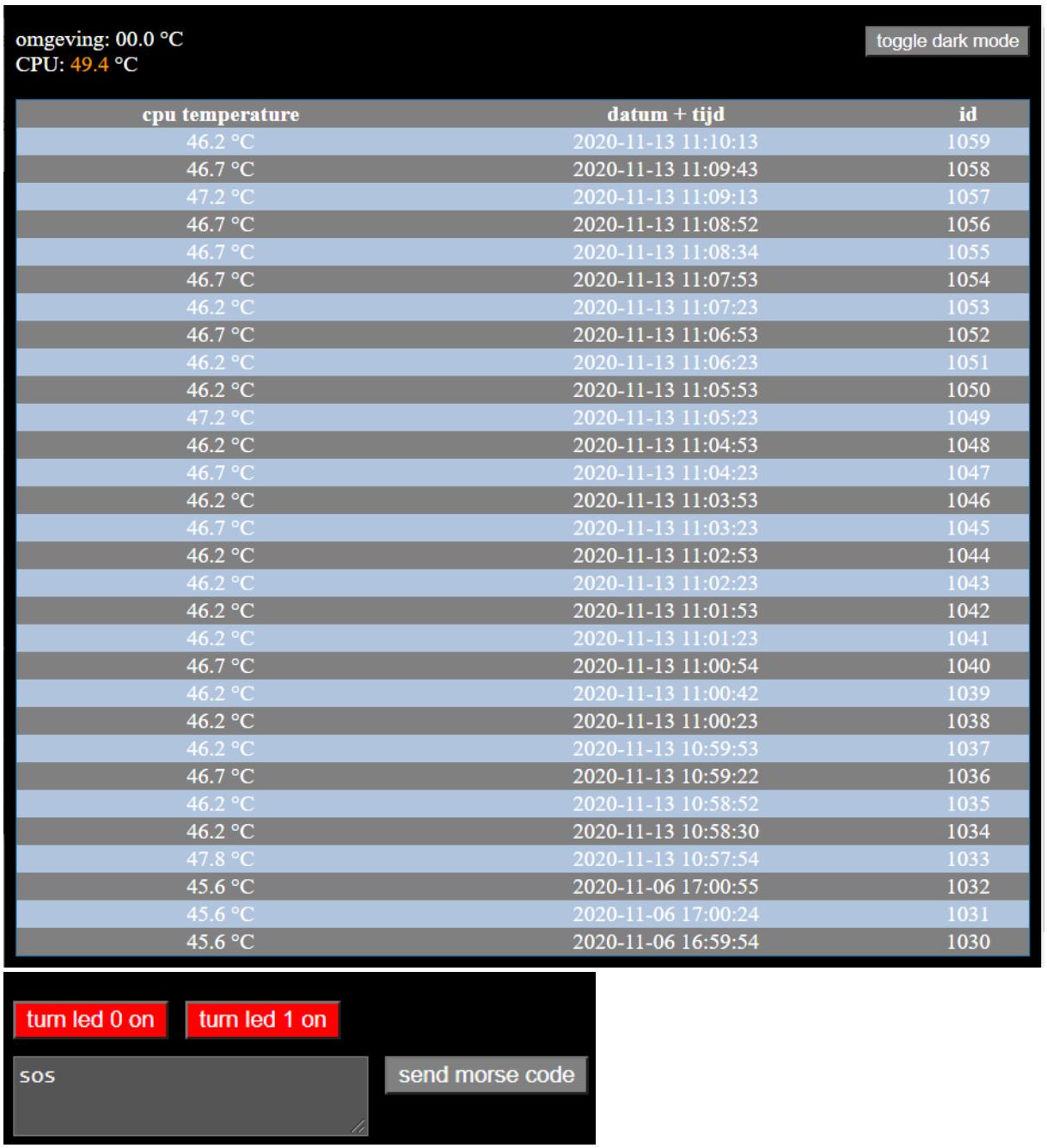
Dit heb ik voor mezelf ingevoerd omdat ik dark mode fijner vond. Ook kwam ik achter dat je niet elke class hoeft te veranderen door simple ‘body.dark-mode’ en dan class of id kun je alles veranderen zonder extra code in de javascript te schrijven.

```
$('.toggle-dark-mode').on('click', function(){
    darkMode();
});

//DARK MODE fijner voor je ogen
function darkMode(){
    document.body.classList.toggle("dark-mode");
}
```

```
 /* DARK MODE SECTION */
.toggle-dark-mode{
    float: right;
}

body.dark-mode{
    background: black;
    color: white;
}

body.dark-mode .temperatures{
    background: grey;
}

body.dark-mode .temperatures tr:nth-child(even){
    background: lightsteelblue;
}

body.dark-mode .morse-code{
    background: #555;
    color: white;
}

body.dark-mode button{
    background: grey;
    color: white;
}
```

Interface

Dit is de interface die ik gemaakt had om de leds en tempratuur te beheren.



cpu temperature	datum + tijd	id
46.2 °C	2020-11-13 11:10:13	1059
46.7 °C	2020-11-13 11:09:43	1058
47.2 °C	2020-11-13 11:09:13	1057
46.7 °C	2020-11-13 11:08:52	1056
46.7 °C	2020-11-13 11:08:34	1055
46.7 °C	2020-11-13 11:07:53	1054
46.2 °C	2020-11-13 11:07:23	1053
46.7 °C	2020-11-13 11:06:53	1052
46.2 °C	2020-11-13 11:06:23	1051
46.2 °C	2020-11-13 11:05:53	1050
47.2 °C	2020-11-13 11:05:23	1049
46.2 °C	2020-11-13 11:04:53	1048
46.7 °C	2020-11-13 11:04:23	1047
46.2 °C	2020-11-13 11:03:53	1046
46.7 °C	2020-11-13 11:03:23	1045
46.2 °C	2020-11-13 11:02:53	1044
46.2 °C	2020-11-13 11:02:23	1043
46.2 °C	2020-11-13 11:01:53	1042
46.2 °C	2020-11-13 11:01:23	1041
46.7 °C	2020-11-13 11:00:54	1040
46.2 °C	2020-11-13 11:00:42	1039
46.2 °C	2020-11-13 11:00:23	1038
46.2 °C	2020-11-13 10:59:53	1037
46.7 °C	2020-11-13 10:59:22	1036
46.2 °C	2020-11-13 10:58:52	1035
46.2 °C	2020-11-13 10:58:30	1034
47.8 °C	2020-11-13 10:57:54	1033
45.6 °C	2020-11-06 17:00:55	1032
45.6 °C	2020-11-06 17:00:24	1031
45.6 °C	2020-11-06 16:59:54	1030

Reflectie

Ik had best veel problemen met het maken van deze project.

in de begin wal ik het in java met maven doen maar dat niet wal werken dus ging ik over na Laravel wat makkelijker ging om voor mij een api te maken.

het lezen van de temperaturen was best te doen zodra ik gevonden had hoe het moest ik liep wel tegen dat ik het eest moest casten naar float.

de leds daar ging het meeste tijd aan uit waardoor ik niet wist of ik het wel af zou. ik liep ook tegen de database waar ik niet mee kon verbinden voor deze dingen zou ik eerst hulp krijgen maar kwam er later toch uit.

voor de leds voerde ik steeds de opstart commando uit zonder sudo waardoor het geen permissies had om de gpio pins aan te passen, daarna liep ik tegen aan dat ik naar de verkeerde nummers keek ik keek namelijk naar de WiringPi prototype board dus 0, 1, 2 in plaats van de nummer waarmee die verbonden waren op de respberry pi zelf. ik liep ook nog eens tegen ajax post api maar dankzij Postman -> code kon ik zien wat ik nodig had.

er waren veel dingen waar ik tegen aan liep niet alleen in dit project maar ook daar buiten, maar toen het werkte was ik best opgelucht, ik vond het project in de begin niet echt leuk maar dat kwam door de dingen waar ik tegen aan liep toen het beter ging begon ik het ook leuker te vinden.

Links en commands die me hielpen tijdens dit project.

https://tecadmin.net/install-laravel-on-debian-9-stretch/

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

mariadb-server doen want mysql-server kan de pi niet vinden.

Voor de leds gebruikte ik php-gpio

https://github.com/ronanguilloux/php-gpio

toen ik route 404 error terug gaf gebruikte ik "php artisan route:clear"

voor de database problemen Laravel SQLSTATE[HY000] [1698] Access denied for user 'root'@'localhost' (1698)

"php artisan config:cache"

"php artisan config:clear"

"php artisan cache:clear"