Rapport projectwerk 1
Biometrisch station

Jonas Meeuws Jonas Van Dycke

Academiejaar 2017-2018

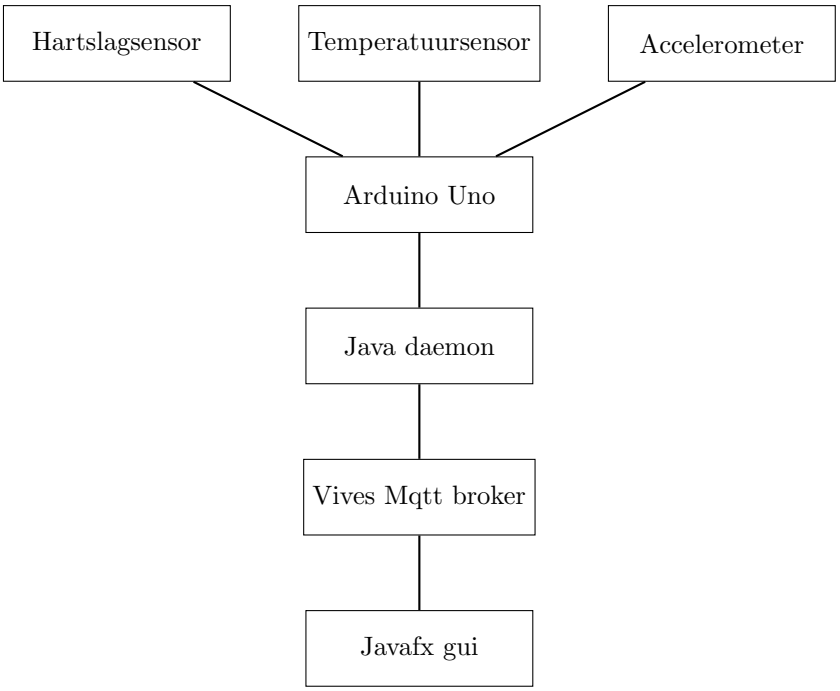
Inhoudsopgave

1 Omschrijving

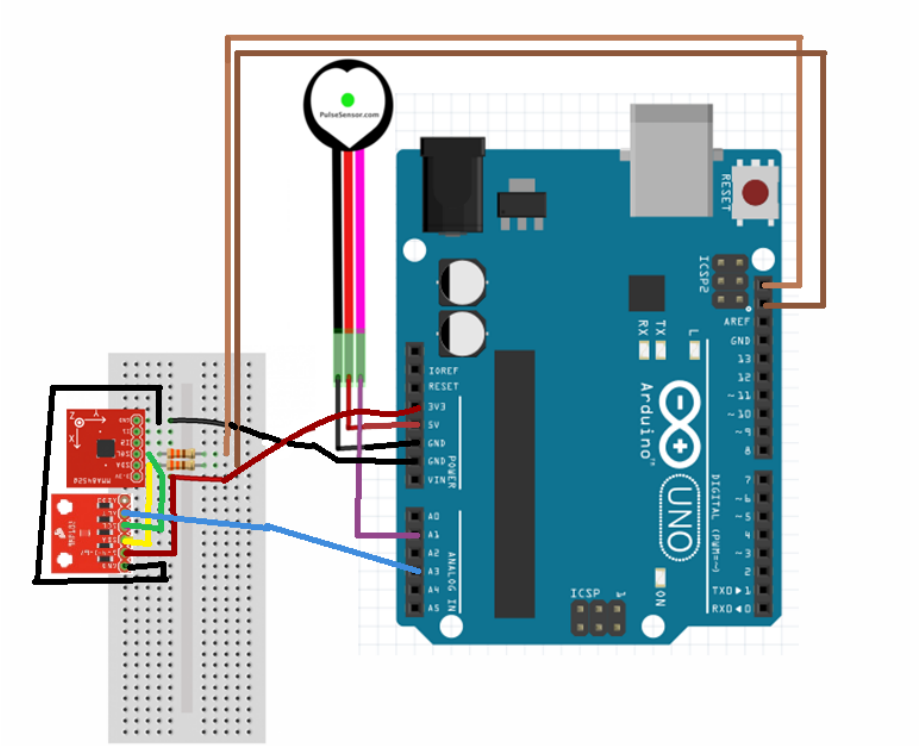
1.1 Doelstellingen

Het doel van dit project is om gegevens uit sensoren te halen en ze te visualiseren. Er worden gegevens gegenereerd door verschillende sensoren die verbonden zijn aan een Arduino. De Arduino hangt aan de USB-poort van een computer. Er draait een java daemon op die computer die communiceert met de Arduino. De daemon zendt de gegevens via Mqtt naar een andere computer waar een grafisch javafx-programma op draait. Het grafisch programma visualiseert de gegevens van meerdere stations aan de hand van grafieken.

1.2 Blokschema



1.3 Schakeling



Op de arduino zijn er 3 sensoren (Temperatuursensor, Accelerometer en Hartslagsensor) aan gekoppeld om de waarden van de sensoren te kunnen meten. Hierbij worden de waarden van de sensoren zichtbaar wanneer er op de knoppen van de “LCD BUTTON SHIELD V2” word gedrukt. Daarom bij het opstarten van de Arduino komt er op de LCD scherm te voorschijn welke knop ingedrukt moet worden voor de sensor waarden te kunnen zien. De gemeten waarden worden van de Arduino naar de “Vives Mqtt broker” gestuurd, om zo via “Javafx gui” een grafiek kunnen voorstellen van deze gemeten waarden.

2 Planning

2.1 Taakverdeling

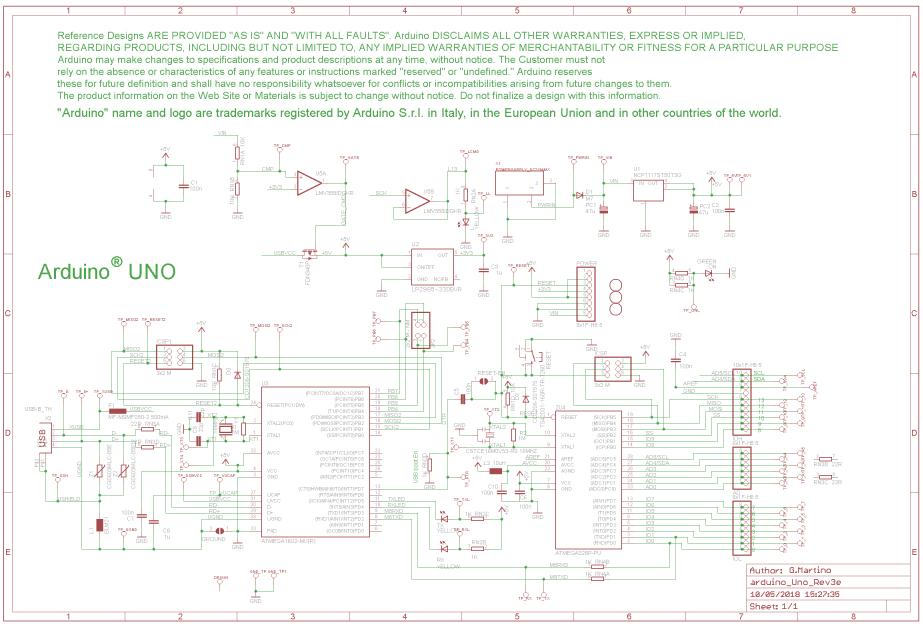
Jonas Meeuws heeft beide Java applicaties volledig geschreven en een deel van de Arduino software (serial). Jonas Van Dycke heeft de schakeling gemaakt en deels de Arduino software geschreven. Tijdens de labo's werkten we vooral samen aan de Arduino software.
De inhoud van dit verslag werd door ons beide geschreven. `%JVD` en `%JM` staan voor Jonas Van Dycke en Jonas Meeuws, respectievelijk. Het document werd opgesteld door Jonas Meeuws.

2.2 Git

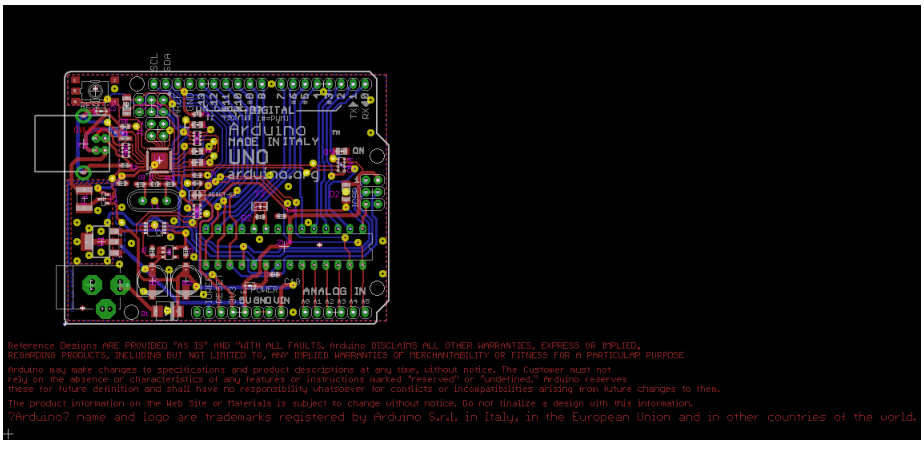
Voer `git blame` uit in de mappen van de gezamelijke delen om lijn per lijn te zien wie welk stuk geschreven heeft. De commits van de Arduino software en het verslag tijdens de lesuren zijn samen gemaakt. Jonas Van Dycke is "unknown". Gebruik `git log` om revisions te zien en `git checkout [revision id]` om naar een vorige versie te gaan. Gebruik `git checkout master` om naar de laatste versie te gaan.

3 Hardware

3.1 Eagle schema



3.2 Eagle board



3.3 Stuklijst (BOM = Bill of Materials)

Component	Prijs
Arduino UNO	€20.00
Pulse Sensor	€24.95
TEMPERATURE SENSOR	€4.95
LCD BUTTON SHIELD V2	€12.95
Triple Axis Accelerometer Breakout	€8.46

3.4 Kostprijsberekening

Totale kost prijs	€61.31

3.5 Overzicht connectoren

2xWeerstanden (330Ω)

3.6 Overzicht test-pinnen

3.7 Stuklijst (BOM = Bill of Materials)

3.7.1 Component: heart pulse sensor amped

Draden	Pin
rood	J1 Nr:3 (5V)
zwart	J1 Nr:5 (GND)
paars	J2 Nr:2 (A1)

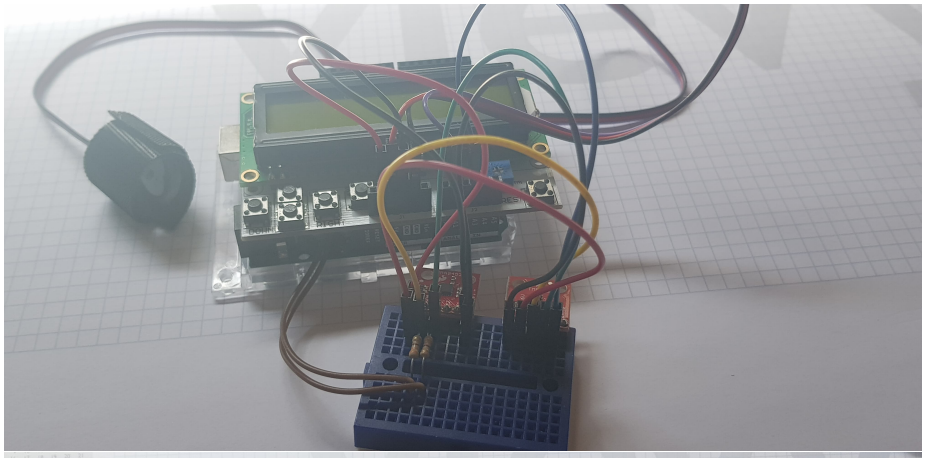
3.7.2 Component: Accelerometer (MMA8452)

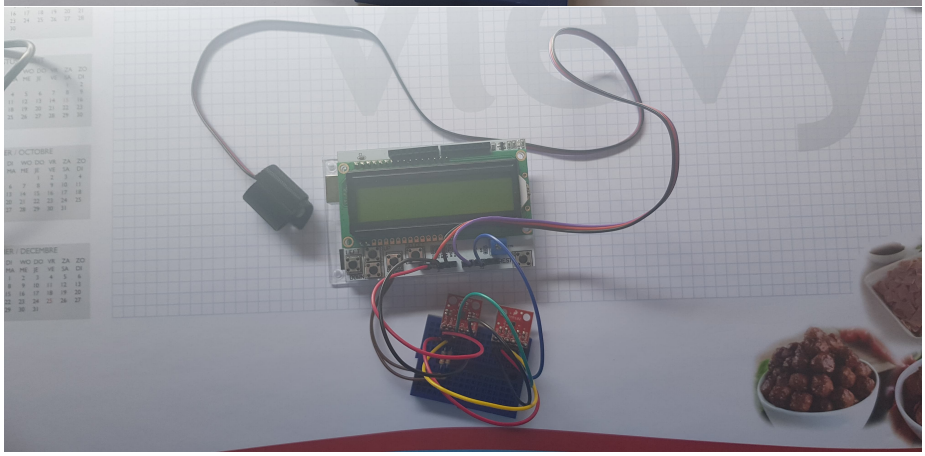
Pin component	Pin Arduino
3.3V	J1 Nr:2 (3.3V)
SDA	SDA
SCL	SCL
I2	/
I1	/
GND	J1 Nr:4 (GND)

3.7.3 Component: Temperature meter (TMP102)

Pin component	Pin Arduino
GND	J1 Nr:4 (GND)
3.3V	J1 Nr:2 (3.3V)
SDA	SDA
SCL	SCL
ALT	J2 Nr:4 (A3)
ADD0	/

3.8 Digitale fotos van opstellingen





4 Software

4.1 Arduino

4.1.1 Main

De code van de arduino bestaat uit 7 verschillende tabbladen, die zelf verschillende classes bevatten (`main` , `Accelerometer` , `Button` , `HeartRate` , `Lcd` , `SerialController` , `Temperature`).

4.1.2 De main

In de main komen alle codes samen, om zo de gewenste eind resultaat te bekomen.

4.1.3 De Accelerometer

Hierin worden de waarden die gemeten worden door de MMA8452 [5] via de poort SDA naar de arduino gebracht. De Arduino zet deze data om naar X, Y en Z waarvan de waarden visueel op de LCD [16] kunnen zien. Om dit te kunnen doen werken moeten de gevraagde 'library' toegevoegd worden.

4.1.4 De HeartRate

Hierbij komt er via Pin A1 een analoge spanning toe die door de pulssensor [12] gemeten word, om zo de hartslag te kunnen meten. Om de digitale waardes te kunnen krijgen voor de LCD [16] moet de gemeten waarden op `A1` laten lezen in `analogRead` .

4.1.5 De Temperature

We meten de waarden van de temperature [13] aan de hand van pin A3. Hier worden de gemeten waarden omgezet in C, om zo de waarde op de LCD [16] te kunnen plaatsen. Om dit te kunnen doen werken moeten de gevraagde 'library' toegevoegd worden.

4.1.6 De LCD

Hier in worden de teksten met de gemeten waarden samen gevoegd om zo duidelijk de waarden via de LCD [16] te kunnen lezen. Om de LCD [16] te kunnen doen werken moet er eerst de 'library' van de LCD [16] weergeven met de pinnen waar de LCD [16] verbonden mee is. Als de verbonden pinnen niet weergeven is, dan zal de LCD [16] ook niet werken. We beschrijven de in deze code de `HeartRate` , `Temperature` en de `Accelerometer` samen met de gemeten waarden.

4.1.7 Button

In deze code beschrijven we welke gemeten waarden te voorschijn komt bij het drukken van de knoppen [16] (`left` , `right` en `down`). We beschrijven welke knop [16] ingedrukt word door elke knop tussen een bepaalde waarde te steken. Deze waarden worden analoog gelezen en omgezet, om zo de drukknoppen [16] doen werken. (PS individueel werken de knoppen [16], maar bij het samen steken van de codes zijn de waarden verandert waardoor de knoppen niet meer werken).

4.1.8 SerialController

Bij deze code worden alle waarden die gemeten worden mooi bij elkaar geplaatst.

4.2 Java parser daemon

Het doel van de parser is om waarden uit de Arduino te krijgen over de seriële poort en ze via Mqtt te verzenden.

4.2.1 Main

De Main classe maakt een nieuw object van `BiometricsParser` aan en roept zijn constructor aan.

4.2.2 BiometricsParser

De constructor roept eerst `initialize()` aan. Het programma wacht tot er een Arduino aangesloten wordt en stelt de baud rate van de poort in op 115200 Bd. Hierna begint de "main loop". Het programma wacht tot de `SerialPort` data beschikbaar heeft. De data wordt uitgelezen door `readMessage()` en wordt opgeslagen in een variabele. Daarna wordt er een `BiometricsData` object samengesteld met behulp van de parser. In `SendMessage()` wordt er met Gson een json-string gemaakt van het `BiometricsData` object. Ten slotte wordt deze string met de MqttService verzonden naar de Mqtt broker, waarna de loop opnieuw begint. Wanneer de Arduino wordt losgekoppeld, wordt het programma afgesloten.

4.2.3 Parser

De statische methode `Parser.parse()` kan gebruikt worden om een `BiometricsData` object te maken van een string. Intern wordt er een methode `isValid()` gebruikt om te controleren of de string wel aan het protocol voldoet. Als dit niet het geval is, wordt het bericht weggesmeten. Het bericht wordt omgezet in een array van doubles door te zoeken naar de locaties van de separators. De methode `Double.parseDouble()` wordt gebruikt om strings in doubles om te zetten. De methode `parse()` maakt een nieuw `Biometricsdata` object met alle waarden en de naam van het station in. Dat object wordt meegegeven als de return value.

4.2.4 BiometricsData

`BiometricsData` is een classe die zowel bij de daemon als de grafische applicatie gelijk is. Dit is omdat ze aan de hand van Gson wordt gebruikt als communicatieprotocol. De classe heeft een naam en 3 waarden als attributen, een constructor en enkele getters en setters. De waarde `acceleration` is een object van de classe `Acceleration`, wat `BiometricsData` een compound maakt. De classe `Acceleration` bestaat uit een x-, een y- en een z-acceleration als attributen, met een paar getters en setters.

4.2.5 MqttService

Deze classe komt uit een andere opdracht, de "Chat client". In de `BiometricsParser` wordt er een object gemaakt van deze classe. Hierna kan `sendMessage()` gebruikt worden om een bericht te verzenden op de topic `biometrics/general`. Behalve de naam van de base topic is er niets veranderd aan de classe.

4.3 JavaFX visualisatie

4.3.1 BiometricsGuiController

Dit is de "Main" classe van de gui. De attributen bestaan uit 4 fxml-objecten, de grafieken. Een `ArrayList` houdt de stations bij en een index houdt de huidige x-waarde bij. Nadat de grafische omgeving is aangemaakt, wordt `initialize()` aangeroepen. De `ChatService` wordt ingesteld en de `SeriesManager` wordt aangemaakt. De `ChatService` zorgt ervoor dat we een methode `MessageArrived()` kunnen maken volgens een interface. Wanneer er een bericht binnenkomt, wordt `MessageArrived()` aangeroepen met als argumenten het channel en het bericht. Met behulp van de Gson-library kunnen we van het bericht een nieuw object van `BiometricData` maken. Als de naam van het station nog niet voorkwam wordt er een nieuw station gemaakt. Nadat het station bestaat, wordt zijn object opgezocht in de `ArrayList` door `getStation()`. Ten slotte wordt de data aan de grafieken toegevoegd.

4.3.2 Station

Elk object van `Station` definieert een apart station. Wanneer de contructor wordt aangeroepen, worden nieuwe series aangemaakt met behulp van de `SeriesManager` uit de controller. Verder kan er een object van `BiometricData` worden toegevoegd aan de series.

4.3.3 SeriesManager

De `SeriesManager` bevat alle grafieken als attributen. Er zijn "create" functies die series maken die in `Station` worden opgeslagen. We moeten `Platform.runLater()` gebruiken om "cross-thread" errors te voorkomen.

4.3.4 BiometricsData

Zie sectie 4.2.4. Belangrijk hier is de name attribuut om de stations te identificeren.

4.3.5 MqttService

Zie sectie 4.2.5. Hier implementeert de classe `BiometricsGuiController` de interface `MqttService.IMqttMessageHandler`. Dit zorgt ervoor dat we `messageArrived()` kunnen gebruiken. Er worden geen andere methoden gebruikt.

4.4 Protocollen

4.4.1 Serial

Voor de seriële communicatie heeft Jonas Meeuws een protocol gemaakt die vooral gemakkelijk te parsen is. De startkarakter is `{`, de separator is een komma en de stopkarakter is `}`. Merk op dat ook na de laatste waarde er een separator komt. Het volgende is een voorbeeld van een bericht.

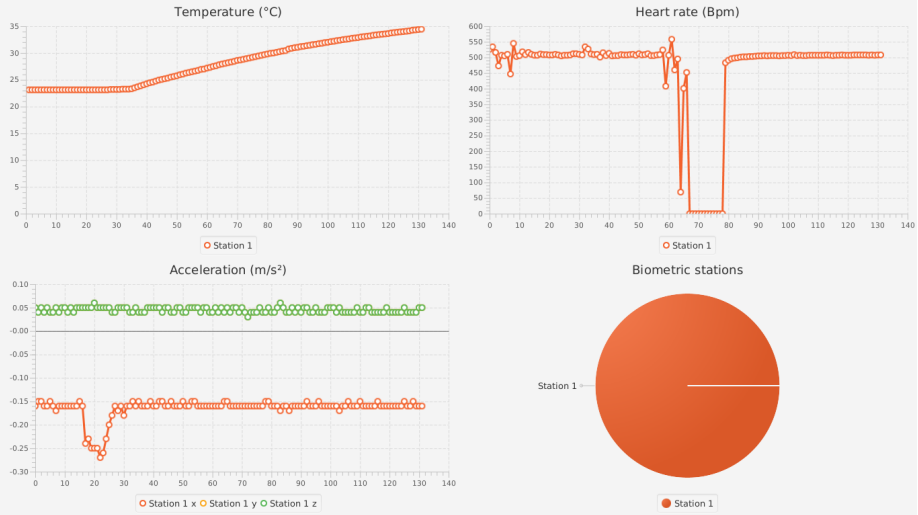
```
{25.00,0.00,2.00,-9.81,85.05,}
```

Eerst komt er een temperatuur, dan de x-versnelling, dan de y-versnelling, dan de z-versnelling en als laatste de hartslag. Alle waarden worden met 2 kommagetallen voorgesteld.

4.4.2 Mqtt

Op de topic `biometrics/general` wordt er elke seconde (bepaald door de Arduino) een Json-string verstuurd. Deze wordt gemaakt door de java library Gson. Er wordt een Json-string gemaakt van een `BiometricsData` object, waar de waarden en de naam van het station in zitten. Die string wordt bij de javafx applicatie terug omgezet in een object van de classe `BiometricsData`.

5 Testen



Dit is een screenshot van de temperatuursensor die opwarmt. Zie ook de videos `Java.mkv` en `Arduino.mp4` om een voorbeeld te zien van de volledige werking.

6 Besluiten

6.1 Java

Alles werkt in de 2 java-applicaties. Seriële communicatie, parsen, mqtt zenden, mqtt ontvangen, meerdere stations beheren en grafieken updaten werken allemaal.

6.2 Arduino

Bij de Arduino werkt de accelerometer, de temperatuurmeter en de hartslagmeter. De gemeten waarden komen op het LCD-scherm. De waarde van de hartslagmeter is de gemeten analoge spanning, niet de beats per minute. We waren van plan om de drukknoppen te gebruiken voor het tonen van de verschillende gemeten waarden. Dit is niet volledig gelukt, maar de code is nog aanwezig. Ze kunnen wel elk apart werken en er zijn meerdere menu's gemaakt voor elke sensor en een "help" menu.

7 Dankwoord

Wij bedanken Nico De Witte, Sille Van Landschoot, Franky Loret en Luc Demeersseman voor de goede begeleiding van dit project. Het was een leerrijke ervaring.

Bibliografie

[1] *Accelerometer*, SM, 2015/07/28, https://www.arduino.cc/en/Tutorial/ADXL3xx.

[2] *Accelerometer*, Jonathanrjpereira, 2016/02/10, http://www.instructables.com/id/Ultimate-Guide-to-Adruino-Serial-Plotter/.

[3] *Accelerometer*, JIMB0, 2014/06/11, https://learn.sparkfun.com/tutorials/mma8452q-accelerometer-breakout-hookup-guide/all.

[4] *Accelerometer*, Louis Moreau, 2018/02/23, https://github.com/sigfox-earthquake/Arduino-MKRFox-MMA8452/blob/master/Arduino-MKRFox-MMA8452.ino.

[5] *Accelerometer*, ToniCorinne, 2015/05/14, https://github.com/sparkfun/SparkFun_MMA8452Q_Arduino_Library/tree/V_1.1.0.

[6] *Hartslagsensor*, SM, 2015/07/29, https://www.arduino.cc/en/Tutorial/Graph.

[7] *Hartslagsensor*, MobiusHorizons, 2017/04/02, https://www.youtube.com/watch?v=BPHinoRZOK4.

[8] *Hartslagsensor*, Pulse Sensor, 2017/03/23, https://www.youtube.com/watch?v=82T_zBZQkOE.

[9] *Hartslagsensor*, VE7JRO, 2017/08/23, https://arduino.stackexchange.com/questions/43956/getting-bpm-from-the-given-code.

[10] *Hartslagsensor*, yury-g, 2017/01/26, https://github.com/WorldFamousElectronics/PulseSensor_Amped_Arduino/blob/master/PulseSensorAmped_Arduino_1.5.0/PulseSensorAmped_Arduino_1.5.0.ino.

[11] *Hartslagsensor*, biomurph, 2018/02/27, https://pulsesensor.com/pages/getting-advanced.

[12] *Hartslagsensor*, Joel Murphy, Yury Gitman, lente van 2013, http://www.theorycircuit.com/pulse-sensor-arduino/.

[13] *Temperatuursensor*, ALEX THE GIANT, 2017/02/02, https://learn.sparkfun.com/tutorials/tmp102-digital-temperature-sensor-hookup-guide.

[14] *Temperatuursensor*, Texas Instruments, 2017/10/31, https://www.youtube.com/watch?v=G0Gn3oQTLaI.

[15] *Temperatuursensor*, awende, 2016/08/15, https://github.com/sparkfun/SparkFun_TMP102_Arduino_Library/blob/master/examples/SparkFun_TMP102_Breakout_Example/SparkFun_TMP102_Breakout_Example.ino.

[16] *LCD, 2017/04/14*, http://linksprite.com/wiki/index.php5?title=16_X_2_LCD_Keypad_Shield_for_Arduino_V2.

[17] *LCD, David Riewe*, 2015/12/30, http://hackerspacetech.com/lcd-button-shield-v2-for-arduino-by-sparkfun.html#.WvRSWoiFNPZ.