Besturen van een auto met behulp van zichtbaar licht communicatie

Klik hier als u tekst wilt invoeren.

Academiejaar 2016-2017

Projectlab bachelor electronica-ICT

Pieter BERTELOOT

Pascal BARBARY

Pieter DEWACHTER

Kristof T’JONCK

Inhoud

[Inhoud ii](#_Toc482883952)

[1 Doel van de opdracht 1](#_Toc482883953)

[2 Algemeen overzicht 2](#_Toc482883954)

[2.1 Algemene richtlijnen 2](#_Toc482883955)

[3 Android applicatie 3](#_Toc482883956)

[3.1 Screenshots, uitleg, verbeteringen mogelijk?, Custom slider klasse & custom bullet voor size 3](#_Toc482883957)

[5 MBED (server) 4](#_Toc482883958)

[6 Lichtcommunicatie (zender) 5](#_Toc482883959)

[6.1 Keuzes componenten, schema’s, scoopbeelden, problemen & eventuele verbeteringen 5](#_Toc482883960)

[7 Lichtcommunicatie (Ontvanger) 6](#_Toc482883961)

[7.1 Keuzes componenten, schema’s, scoopbeelden, problemen & eventuele verbeteringen 6](#_Toc482883962)

[8 MBED Polulu M3PI 7](#_Toc482883963)

[8.1 Keuzes componenten 7](#_Toc482883964)

[9 Algemeen besluit 8](#_Toc482883965)

[9.1 8](#_Toc482883966)

[Referenties 9](#_Toc482883967)

[Bijlagen 10](#_Toc482883968)

[Bijlage A Detailtekeningen van de proefopstelling 1](#_Toc482883969)

# Doel van de opdracht

# Algemeen overzicht

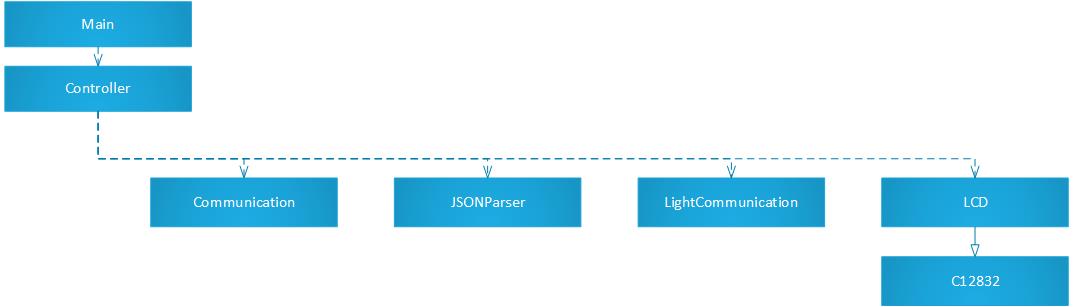
## Algemene richtlijnen

# Android applicatie

## Screenshots, uitleg, verbeteringen mogelijk?, Custom slider klasse & custom bullet voor size

# MBED (server)

Een programma op de MBED zorgt er voor dat de data van de android applicatie via een TCP Server ontvangen kan worden, deze data wordt vervolgens omgezet om te versturen via lichtcommunicatie naar de polulu M3PI. Op figuur 4-1 zien we een algemeen overzicht van hoe dit er uit ziet.



Figuur 4‑1 Klassendiagram van de MBED server

## Controller

De controller klasse zorgt voor de samenhang van het gehele programma. De run functie van deze klasse zal eigenlijk de hoofdmethode zijn die het gehele programma bepaalt in de “main”.

Via de “Communication” klasse wordt data ontvangen die vervolgens verwerkt wordt in de “JSONParser”, dat op zijn beurt dan verzonden wordt via de “LightCommunication” klasse.

## Communication

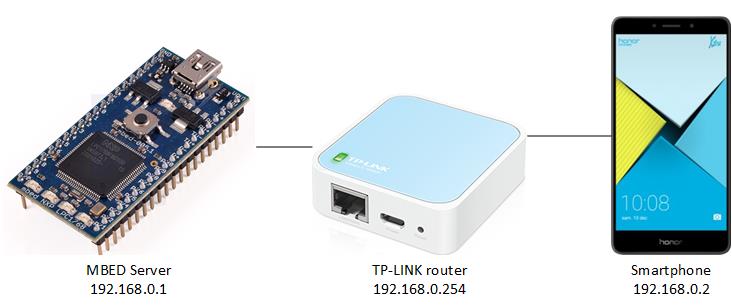
De communication klasse zorgt voor een ethernetverbinding via de “EthernetInterface” package. Om zo dan via een socket informatie van een client kan verkrijgen.

### Connectie met ethernet

Deze verbinding wordt gemaakt om zo via TCP/IP data te krijgen van de Android-applicatie. De ethernetinterface zal als volgt geconfigureerd worden:

* IP address: 192.168.0.1
* Subnet mask: 255.255.255.0
* Default gateway 192.168.0.1

Dit wil zeggen dat de Android-applicatie ook een ip address moet hebben in dezelfde range. Dus een IP adres dat tussen 192.168.0.2 en 192.168.0.255 gelegen ligt. Aangezien we gebruik maken van een eigen router die niet aan het net hangt moeten we deze statisch instellen. De router dat we eerst gebruikten maakte gebruik van 192.168.0.x adressen. Deze router was de TP-LINK WR702N, deze maakte gebruik van het IP adres 192.168.0.254. Deze gaf dus automatisch IP adressen in het juiste bereik. Een overzicht van hoe de IP’s verdeeld zijn is te zien op figuur 4-2.

Aangezien de verbinding met deze router wegviel hebben we een andere router gebruikt. Deze gebruikte 192.168.1.x als lokale adressen. Door op de Smartphone een statisch IP adres in te stellen, alsook op de mbed, kan er dus een communicatie voorzien worden. 

Figuur 4‑2 Overzicht IP adressen van de verschillende apparaten

### Verbinden met een socket

De “EthernetInterface” library bevat tevens ook methodes waarmee een socket server opgezet kan worden. De server zal luisteren op poort 4000 om zo inkomende data van de Android applicatie op te vangen. Dit door een client via een TCPSocketConnection te ontvangen als er data komt. Het ontvangen is volledig “blocking”, de MBED zal dus wachten tot er data binnen gekomen is vooraleer er ontvangen wordt.

## JSONParser

De binnengekomen data zal in de vorm van een JSON (JavaScript Object Notation) String zijn. De data van de client, left en right worden uit de JSON string gehaald via een parser en vervolgens in een char array geplaatst met een check byte.

### Parsen van de data

De data moet uit de json string gehaald worden, hiervoor gebruiken we de “Picojson” library. Deze zal een error geven als geen correcte string ingelezen is om zo fouten te voorkomen. Als een foute string doorgekomen is zal dit pakket simpelweg weggegooid worden.

Deze data moet worden omgezet naar een array van int8\_t waarbij we een zo’n klein mogelijk aantal bytes willen doorsturen. De data moet namelijk via het licht doorgestuurd worden, hoe langer de data hoe langer het zal duren om door te sturen (zie 4.5.2). De data zal dus 4 bytes lang zijn. Éen byte voor de client, één voor left, één voor right en één voor de checkbyte. Left en right zijn waarden van -100 tot 100, aangezien een int8\_t alle waardes kan bevatten tussen -128 tot 127, is dit ideaal om deze data in op te slaan met zo weinig mogelijk bits. In tabel 4-1 staat een oversicht over hoe de byte array er uit ziet.

|  |  |  |  |
| --- | --- | --- | --- |
| **int8\_t[0]** | **int8\_t[1]** | **int8\_t[2]** | **int8\_t[3]** |
| client | left | right | check byte |

Tabel 4‑1 Overzicht van de byte array

### Berekenen checkbyte

De checkbyte wordt berekend met de volgende formule:

Dit zorgt er voor dat er kan gechecked worden als de left en right correct zijn. 251 is het grootste priemgetal onder de 255, dus onder de maximale waarde van 8 bits. De 127 is het priemgetal in de helft hiervan. Priemgetallen worden vaak gebruikt in cryptografie om een betere security te garanderen. We hadden voor deze simpele methode gebruikt aangezien er maar 2 bytes in rekening gebracht moesten worden. Echter bij nader inzien was deze methode niet zo goed. (left + right) kan namelijk soms gelijk uitkomen bij verschillende getallen waardoor de checkbyte dan ook gelijk kan zijn. Dit komt niet veel voor maar er zijn betere alternatieven.

Omdat de checkbyte methode niet goed was, is de checkbyte veranderd in een CRC-6 checksum, deze zal een betere error detectie geven. Deze methode wordt namelijk vaak gebruikt voor error detectie in digitale netwerken. In ons voorbeeld zal de checksum 0x02 geven.

|  |  |  |  |
| --- | --- | --- | --- |
| **0x01** | **0x37** | **0xF1** | **0x95** |
| 1 | 37 | -15 | 149 |

## LCD

## Lightcommunication

# Lichtcommunicatie (zender)

## Keuzes componenten, schema’s, scoopbeelden, problemen & eventuele verbeteringen

# Lichtcommunicatie (Ontvanger)

## Keuzes componenten, schema’s, scoopbeelden, problemen & eventuele verbeteringen

# MBED Polulu M3PI

## Keuzes componenten

# Algemeen besluit

## 

Referenties

*Hier komt de volledige referentielijst in de gekozen stijl APA of IEEE.*

Bijlagen

Bijlagen worden bij voorkeur enkel elektronisch ter beschikking gesteld. Indien essentieel kunnen in overleg met de promotor bijlagen in de scriptie opgenomen worden of als apart boekdeel voorzien worden.

Er wordt wel steeds een lijst met vermelding van alle bijlagen opgenomen in de scriptie. Bijlagen worden genummerd het een drukletter A, B, C, …

Bijlage A Detailtekeningen van de proefopstelling

Bijlage B Meetgegevens (op USB)

1. Detailtekeningen van de proefopstelling