NTNU i Gjøvik, ELE3391 Elektro prosjekt

IQRF Smartby

Manual for IQRF basert smartby-prosjekt

Simen Stensås, Kjell Kirkaune
26.11.2019 rev. 28.11.2019

Innholdsfortegnelse

1. Introduksjon

Kort om prosjektet og arbeidet som er gjort hittil.

- Prosjektet startet sen august/tidlig september 2019, og ble avsluttet 20. november 2019. Noe som tilsvarer rundt tre måneder, hvor av en og en halv til to måneder var kun jobb med prosjektet. Tidsforbruk blir antatt til fire-fem-seks timer hver dag i denne perioden hver for oss.
- Det består av seks deler:
    - Temperaturstyrt rom
    - Kodelåst dør
    - Gatelys
    - Trafikklys
    - Parkeringsplass m/ skjerm
    - RPi gateway koblet til IBM Cloud
- For øyeblikket (per 27. nov 2019) er nettsiden http://iqrf-smartcity.eu-gb.mybluemix.net/ui det som styrer alt i smartbyen.
- IQRF eller intelligent radio frequency er den trådløse standarden som er brukt i prosjektet. TR-72DA er transceiveren som ble brukt i prosjektet, den har egen antenne og en mikrokonroller PIC16LF1938-I/MV.

For å fortsette videre antas det at leser innehar noe kunnskap om IQRF, mikrokontrollere og generell elektronikk. Programmeringskunnskaper og forståelse av diverse dataprotokoller er en fordel. Under mappen «IQRF Datasheets and user guides» ligger det brukermanualer til IQRF og underkonsepter slik som DPA, CDC, UDP, OS, CC5X compiler etc., og datablad til DK-EVAL-04, PIC16LF1938-I/MV, TR-72DA etc.

2. Mappeinnhold

For at prosjektet skal leve og utvikles videre er alt av programmeringskode, kretstegninger, PCB-design etc. lagret i en og samme mappe slik at det skal være enklere å sette seg inn i prosessen.

Generelt oppsett i hver av mappene under er:

- «Circuit»-mappe som inneholder kretstegninger og PCB-design.
    - «pcb»-mappe.
        - En SVG-fil med PCB-design av aktuell modul (inkl. topp og bunn).
        - En PDF-fil med PCB-design av aktuell modul (inkl. topp og bunn).
        - En DXF-fil med PCB-design av aktuell modul.
        - En Altium prosjektfil med PCB-design av aktuell modul i «pcbdoc» format.
        - En EasyEDA prosjektfil med PCB-design av aktuell modul i «json» format.
    - «schematics»-mappe.
        - En SVG-fil med kretstegning av aktuell modul.
        - En PDF-fil med kretstegning av aktuell modul.
        - En Altium prosjektfil med kretstegning av aktuell modul i «schdoc» format.
        - En EasyEDA prosjektfil med kretstegning av aktuell modul i «json» format.
- «CustomDpaHandler»-mappe som inneholder koden til IQRF-brikkene.
    - «CustomDpaHandler- ...c»-fil som inneholder koden til aktuell modul.
    - «hex»-mappe
        - «CustomDpaHandler- ... hex»- fil som er ferdig bygget kode. Det er denne som overføres til transceiveren.

Alle andre filer utenom de nevnt over blir gått gjennom under.

Som nevnt tidligere har «IQRF Datasheets and user guides»-mappen diverse datablader og brukermanualer som er relevante for dette prosjektet.

2.1. «Streetlights»-mappen

For alt som har med gatelysene å gjøre.

- «macro.iqrfmcr» er en macro fil for å sende kommandoer i IQRF IDE. Ikke spesielt nødvendig fil. Bra for effektiv testing.

2.2. «Parking Lot»-mappen

For alt som har med parkeringsplassen å gjøre. Denne mappen er delt inn i to deler, «space» og «screen» som representerer selve parkeringsplassen med sensoren, og skjermen som viser hvor mange plasser det er ledig.

«space»-mappen er lik oppsettet i 2.

2.2.1. «screen»-mappen

Her er oppsettet kjent som i 2., bortsett fra:

- «Arduino»-mappe
  - «ArduinoNanoScreenController»-mappe
    - «ArduinoNanoScreenController.ino» er en Arduino-fil som lastes opp til Arduino Nanoen som er koblet til kretskortet. Det gjør det mulig å koble til en OLED-skjerm.

2.3. «Locked Door»-mappen

For alt som har med den kodelåste døren å gjøre.

Her er oppsettet kjent som i 2., bortsett fra:

- «Arduino»-mappe
  - «ArduinoNanoServoDoor»-mappe
    - «ArduinoNanoServoDoor.ino» er en Arduino-fil som lastes opp til Arduino Nanoen som er koblet til kretskortet. Det gjør det mulig å styre en servo som også er koblet til.

2.4. «Temperature Room»-mappen

For alt som har med det temperaturstyrte rommet å gjøre.

- «Datasheets»-mappen inneholder datablader
  - «Mosfet Driver SN75372P.pdf» er en PDF-fil med databladet til SN7532P.
  - «Mosfet NTD20N03L27.pdf» er en PDF-fil med databladet til NTD20N03L27.

2.5. «Trafficlights»-mappen

For alt som har med trafikklysene å gjøre. Følger oppsettet i 2. bortsett fra at det er delt inn i 1 SHR (shift register) og 2 SHR.

- «Datasheets»-mappen inneholder datablad og et dokument for å forstå det gamle systemet.
  - «LED Light Bars.pdf» er en PDF-fil med databladet til diodene som ble brukt i trafikklysene.
  - «Understanding the old system.pdf» er en PDF-fil med hvordan utregninger ble gjort etc.
- «Trafficlight combinations (cases).txt» er et raskt og grovt tekstdokument i enkle trekk om hvordan trafikklyset fungerer.

2.6. «Gateway and Cloud»-mappen

For alt som har med RPi gateway og IBM Cloud å gjøre.

3. Prosjektet oppdelt

Prosjektet er delt opp i seks deler, hvorav alle gjennomgås del for del. Rekkefølgen er helt tilfeldig. Eksempler i hex-formater blir gjerne sendt fra IQRF IDE el., mens eksempler i JSON-format bruker desimale tall (slik som fra Node Red på Raspberry Pi gateway).

I eksemplene under blir gatelys nummer én skrudd på med 100% lysstyrke. Mer informasjon finnes under gatelys-avsnittet.

Eksempel 1: HEX-strukturen til DPA Request:

	Node Address (NADR)	**Peripheral Address (PNUM)**	**Peripheral Command (PCMD)**	**Hardware Peripheral ID (HWPID)**	**Peripheral Data (PDATA[])**
Størrelse i bytes	2B	1B	1B	2B	0 til 56B
Eksempel 1 oppdelt	0x0001	0x20	0x00	0xFFFF	0x00, 0xFF
Eksempel 2 i string	**00.01.20.00.FF.FF.00.FF** (DETTE ER DET MAN SENDER)				

Eksempel 2: JSON-strukturen til DPA Request:

```
msg.payload = {
      "mType": "iqrfRawHdp",
      "data": {
            "msgId": "Streetlight",
            "req": {
                  "nAdr": 1,
                  "pNum": 32,
                  "pCmd": 0,
                  "hwpId": 65535,
                  "pData": [0, 255]
            },
      "returnVerbose": true
       }
};
```

I eksempelet over er alle verdier i det desimale tallsystemet. Dermed er de to eksemplarene over helt like, forskjellen ligger i hvor de blir sendt i fra. En Raspberry Pi gateway vil kun forstå eksempel 2, mens direkte kontakt fra coordinator til node via f.eks. IQRF IDE vil benytte seg av eksempel 1.

NB! Det er unntak for dette! Men i dette prosjektet brukes DPA-strukturen.

3.1. Temperaturstyrt rom

Det temperaturstyrte rommet består av tre forskjellige deler. Et kretskort koblet til varmeelement, et kretskort koblet til kjøleelement og et kretskort koblet til en temperatursensor med sensitivitet 0,5 grader C.

En typisk datapakke vil se slik ut:

- **NADR**:
  - **Varmeelement**: 0x08 (8)
  - **Kjøleelement**: 0x09 (9)
  - **Temp. sensor**: 0x0A (10)
- **PNUM**: 0x22 (34)
- **PCMD**: 0x00 (0)
- **HWPID**: ubetydelig dvs. 0xFFFF (65535)
- **PData**[0]:
  - **For å slå av:** 0x00 (0)
  - **For å slå på:** 0x01 (1)

Eksempel 1: skru på varmeelement (node 8) i HEX-format:

00.08.22.00.FF.FF.01

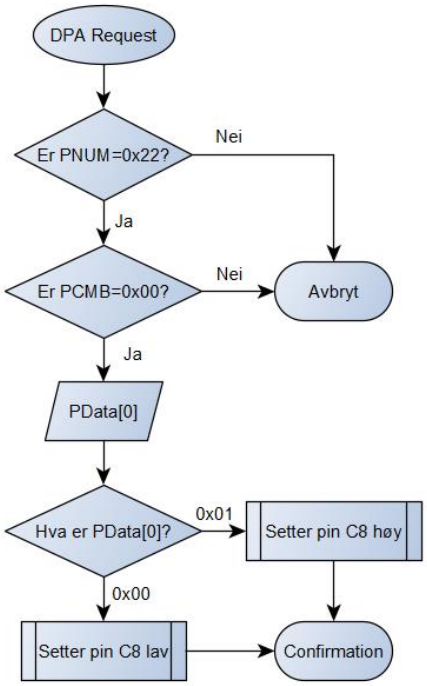
Eksempel 2: skru av kjøleelement (node 9) i JSON-format:

```
msg.payload = {
      "mType": "iqrfRawHdp",
      "data": {
            "msgId": "Temperature controlled room",
            "req": {
                  "nAdr": 9,
                  "pNum": 34,
                  "pCmd": 0,
                  "hwpId": 65535,
                  "pData": [0]
            },
      "returnVerbose": true
       }
};
```

3.1.1. Varmeelement

DPA:

- **NADR:** 0x08 (8)
- **PNUM:** 0x22 (34)
- **PCMD:** 0x00 (0)
- **HWPID:** 0xFFFF (65535)



Elektriske merknader:

- **V_{in}:** 12V og 5V
- **I_{typ}:** 1,65 – 1,75A (fra 12V)

I varmeelement-kretskortet er TR-72DA blitt programmert på en slik måte at pin C8 blir satt som output ved initialisering og reset.

For hver gang transceiveren mottar en DPA Request vil den sjekke om PNUM og PCMD er riktig, hvis ikke vil den avbryte.

Er PNUM og PCMD riktige vil pin C8 bli satt høy eller lav ettersom hva PData[0] er. Pin C8 er koblet til en MOSFET driver som er koblet til en MOSFET som igjen leder eller ikke leder strøm til motstanden (varmeelementet).

HEX-format

DPA Request	PData[0]
Skru på varmeelement:	00.08.22.00.FF.FF.**01**
Skru av varmeelement:	00.08.22.00.FF.FF.**00**

JSON-format

Venstre: hvordan skru på varmeelement

```
msg.payload = {
      "mType": "iqrfRawHdp",
      "data": {
            "msgId": "Heating",
            "req": {
                  "nAdr": 8,
                  "pNum": 34,
                  "pCmd": 0,
                  "hwpId": 65535,
                  "pData": [1]
            },
      "returnVerbose": true
       }
};
```

Høyre: hvordan skru av varmeelement

```
msg.payload = {
      "mType": "iqrfRawHdp",
      "data": {
            "msgId": "Heating",
            "req": {
                  "nAdr": 8,
                  "pNum": 34,
                  "pCmd": 0,
                  "hwpId": 65535,
                  "pData": [0]
            },
      "returnVerbose": true
       }
};
```

3.1.2. Kjøleelement

DPA:

- **NADR:** 0x09 (9)
- **PNUM:** 0x22 (34)
- **PCMD:** 0x00 (0)
- **HWPID:** 0xFFFF (65535)

Elektriske merknader:

- **V_{in}:** 12V og 5V
- **I_{typ}:** 150mA (fra 5V)

I varmeelement-kretskortet er TR-72DA blitt programmert på en slik måte at pin C8 blir satt som output ved initialisering og reset.

For hver gang transceiveren mottar en DPA Request vil den sjekke om PNUM og PCMD er riktig, hvis ikke vil den avbryte.

Er PNUM og PCMD riktige vil pin C8 bli satt høy eller lav ettersom hva PData[0] er. Pin C8 er koblet til en MOSFET driver som er koblet til en MOSFET som igjen leder eller ikke leder strøm til vifta (kjøleelementet).

Flow-diagrammet i 3.1.1. gjelder også for kjøleelementet.

HEX-format

DPA Request	PData[0]
Skru på kjøleelement:	00.09.22.00.FF.FF.**01**
Skru av kjøleelement:	00.09.22.00.FF.FF.**00**

JSON-format

Venstre: hvordan skru på kjøleelement

```
msg.payload = {
      "mType": "iqrfRawHdp",
      "data": {
            "msgId": "Cooling",
            "req": {
                  "nAdr": 9,
                  "pNum": 34,
                  "pCmd": 0,
                  "hwpId": 65535,
                  "pData": [1]
            },
      "returnVerbose": true
       }
};
```

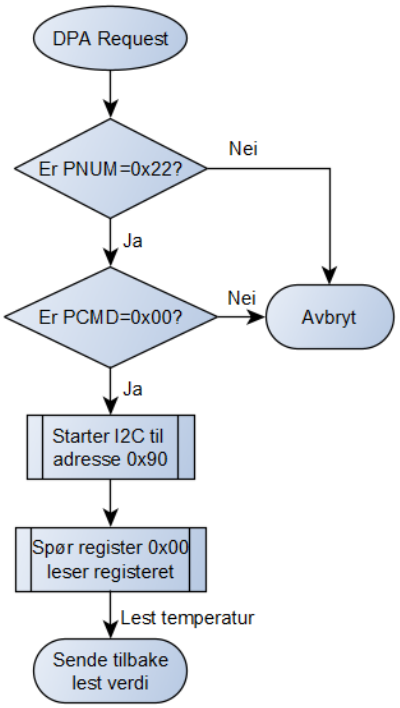
Høyre: hvordan skru av kjøleelement

```
msg.payload = {
      "mType": "iqrfRawHdp",
      "data": {
            "msgId": "Cooling",
            "req": {
                  "nAdr": 9,
                  "pNum": 34,
                  "pCmd": 0,
                  "hwpId": 65535,
                  "pData": [0]
            },
      "returnVerbose": true
       }
};
```

3.1.3. Temperatursensor

DPA:

- **NADR:** 0x0A (10)
- **PNUM:** 0x22 (34)
- **PCMD:** 0x00 (0)
- **HWPID:** 0xFFFF (65535)



Elektriske merknader:

- **V_{in}:** 5V
- **I_{typ}:** 10mA (fra 5V)
- **Sensor:** temperatursensor

Temperatursensor-kretskortet består av en TR-72DA og I2C tilkobling til temperatursensor. Skriver først I2C adresse og så register adresse, deretter skriver man I2C adressen på nytt og ber om å lese (i henhold til datablad).

Blir temperatur suksessfullt lest, blir temperaturen sendt tilbake som DPA Response i PData[0] og PData[1]. Grunnen til at den blir sendt oppdelt er fordi størrelsen er på 9 bit.

HEX-format

	DPA Request	DPA Response
Spørre etter temperatur	00.0A.22.00.FF.FF	00.0A.22.00.FF.FF.??.??.PData[0].PData[1]

MSB i PData[0] bestemmer fortegn, hvis 1 → minusgrader. De andre verdiene sier hva temperaturen er. MSB i PData[1] sier om det er en halv grad eller ikke, dvs. hvis 1 → x.5 grad.

JSON-format

Venstre: spørre etter temperatur

```
msg.payload = {
      "mType": "iqrfRawHdp",
      "data": {
            "msgId": "Temp. sensor",
            "req": {
                  "nAdr": 10,
                  "pNum": 34,
                  "pCmd": 0,
                  "hwpId": 65535,
                  "pData": []
            },
      "returnVerbose": true
       }
};
```

Høyre: hva man kan få tilbake

```
msg.payload = {
      "mType": "iqrfRawHdp",
      "data": {
            "msgId": "Cooling",
            "req": {
                  "nAdr": 9,
                  "pNum": 34,
                  "pCmd": 0,
                  "hwpId": 65535,
                  "pData": [16, ]
            },
      "returnVerbose": true
       }
};
```

3.2. Kodelåst dør

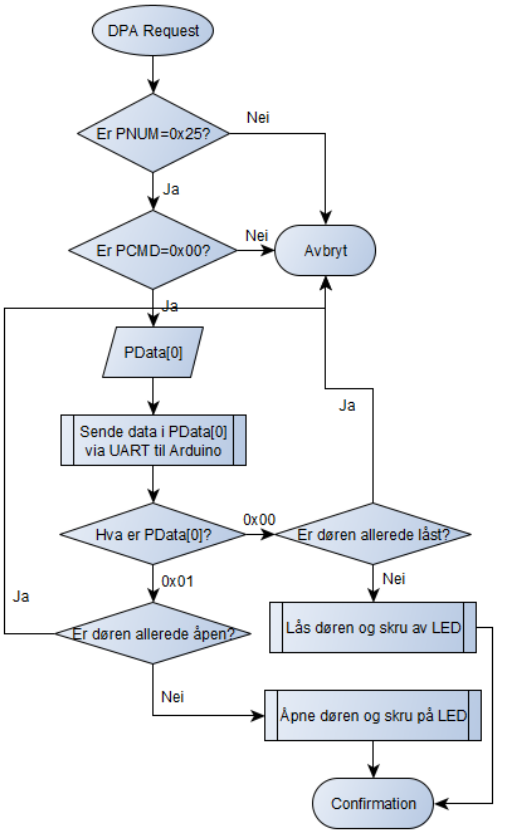
DPA:

- **NADR:** 0x15 (21)
- **PNUM:** 0x25 (37)
- **PCMD:** 0x00 (0)
- **HWPID:** 0xFFFF (65535)

Elektriske merknader:

- **V_{in}:** 12V, 5V
- **I_{max}:** 190mA (fra 5V)
- **Servo:** Parallax Standard Servo (#900-00005)
- **Arduino Nano**

I denne kretsen er det brukt en Arduino Nano for enklere kontrollering av servoen. TR-72DA mottar enten 0x00 (låse) eller 0x01 (åpne), som blir sendt til Arduinoen via UART. Det blir sjekket om det er en endring fra siste låsing/åpning.



HEX-format

DPA Request	PData[0]
Åpne døren:	00.15.25.00.FF.FF.**01**
Lukke døren:	00.15.25.00.FF.FF.**00**

JSON-format

Venstre: åpne døren

```
msg.payload = {
      "mType": "iqrfRawHdp",
      "data": {
            "msgId": "Open door",
            "req": {
                  "nAdr": 21,
                  "pNum": 37,
                  "pCmd": 0,
                  "hwpId": 65535,
                  "pData": [1]
            },
      "returnVerbose": true
       }
};
```

Høyre: lukke døren

```
msg.payload = {
      "mType": "iqrfRawHdp",
      "data": {
            "msgId": "Closing door",
            "req": {
                  "nAdr": 21,
                  "pNum": 37,
                  "pCmd": 0,
                  "hwpId": 65535,
                  "pData": [0]
            },
      "returnVerbose": true
       }
};
```

3.3. Gatelys (1 til 7)

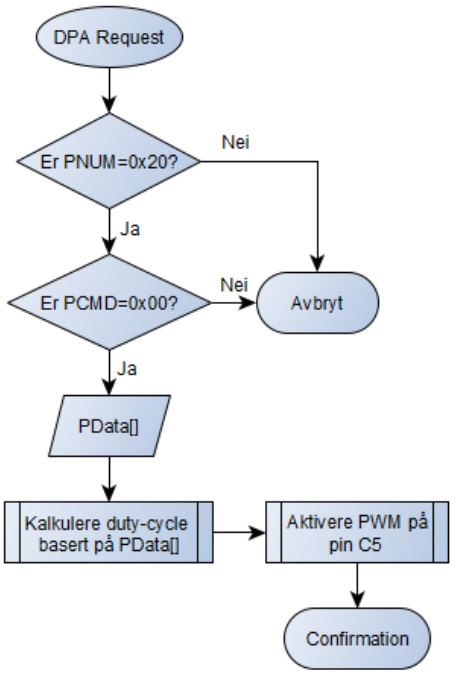
DPA:

- **NADR:** 0x01 (1), 0x02 (2), 0x03 (3), 0x04 (4), 0x05 (5), 0x06 (6), 0x07 (7)
- **PNUM:** 0x20 (32)
- **PCMD:** 0x00 (0)
- **HWPID:** 0xFFFF (65535)

Elektriske merknader:

- **V_{in}:** 5V
- **I_{max}:** 60mA (fra 5V)

På pin C5 på TR-72DA sendes det ut et bestemt PWM signal inn på basen til en BJT. På denne måten er det mulig å kontrollere lysstyrken til LED-en.



HEX-format (eksempel for gatelys 1, da alle er like)

DPA Request	PData[0], PData[1], PData[2], PData[3], PData[4]
0% lysstyrke	00.01.20.00.FF.FF.**00**.**00**
100% lysstyrke	00.01.20.00.FF.FF.**00**.**FF**
Dim fra 0% til 100%	00.01.20.00.FF.FF.**01**.**00**.**01**.**FF**.**00**
Dim fra 100% til 0%	00.01.20.00.FF.FF.**01**.**00**.**01**.**01**.**FF**

JSON-format

Venstre: 0% lysstyrke

```
msg.payload = {
      "mType": "iqrfRawHdp",
      "data": {
            "msgId": "0% light",
            "req": {
                  "nAdr": 1,
                  "pNum": 32,
                  "pCmd": 0,
                  "hwpId": 65535,
                  "pData": [0,0]
            },
      "returnVerbose": true
       }
};
```

Høyre: 100% lysstyrke

```
msg.payload = {
      "mType": "iqrfRawHdp",
      "data": {
            "msgId": "100% light",
            "req": {
                  "nAdr": 1,
                  "pNum": 32,
                  "pCmd": 0,
                  "hwpId": 65535,
                  "pData": [0,255]
            },
      "returnVerbose": true
       }
};
```

Venstre: Dim fra 0% til 100%

```
msg.payload = {
      "mType": "iqrfRawHdp",
      "data": {
            "msgId": "Dim up",
            "req": {
                  "nAdr": 1,
                  "pNum": 32,
                  "pCmd": 0,
                  "hwpId": 65535,
                  "pData": [1,1,0,255,0]
            },
      "returnVerbose": true
       }
};
```

Høyre: Dim fra 100% til 0%

```
msg.payload = {
      "mType": "iqrfRawHdp",
      "data": {
            "msgId": "Dim down",
            "req": {
                  "nAdr": 1,
                  "pNum": 32,
                  "pCmd": 0,
                  "hwpId": 65535,
                  "pData": [1,0,1,1,255]
            },
      "returnVerbose": true
       }
};
```

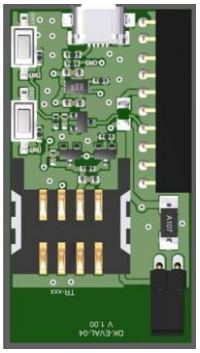
3.3.1. Lyssensor (enkel løsning)

DPA:

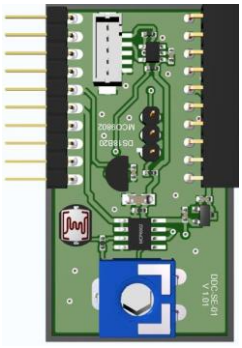
- **NADR:** 0x0F (15)
- **PNUM:** 0x20 (32)
- **PCMD:** 0x31 (49)
- **HWPID:** 0xFFFF (65535)

Elektriske merknader:

- **V_{in}:** 5V



DK-EVAL-04



DDC-SE-01

Her er det ikke et eget designet kretskort. Det er brukt et development-kit til å løse denne. Det er ferdig kode og krets, som gjør den enkel å bruke. TR-72DA sitter i DK-EVAL-04 som da festes som vist til DDC-SE-01. Det er et batteri på DK-EVAL-04 som er praktisk.

HEX-format og JSON-format (under)

	DPA Request	DPA Response
Spørre etter lysstyrke	00.0F.20.31.FF.FF	00.0F.20.31.FF.FF.??.??.??.PData[1]

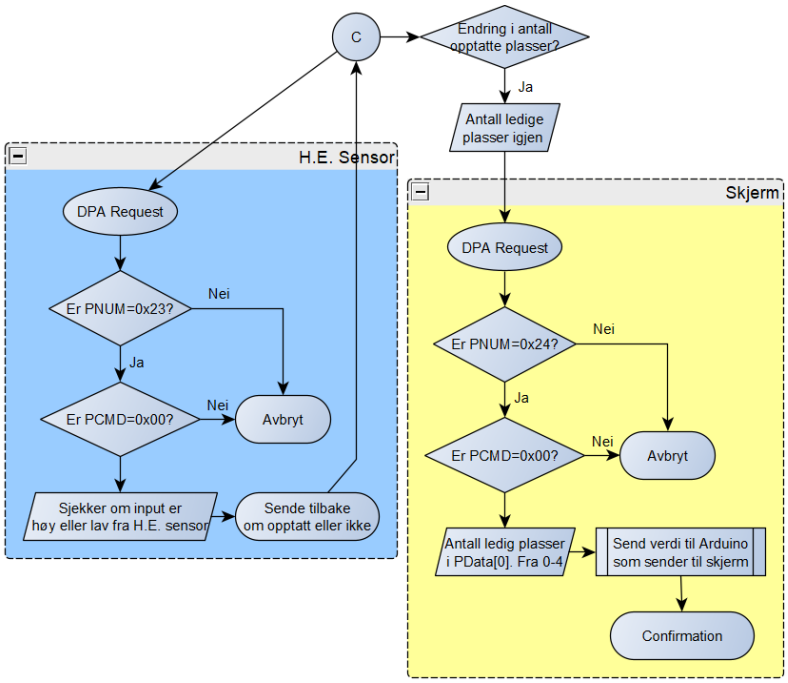
Venstre: spørre etter lysstyrkeverdi

Høyre: hva man får tilbake

```
msg.payload = {
      "mType": "iqrfRawHdp",
      "data": {
            "msgId": "Light",
            "req": {
                  "nAdr": 15,
                  "pNum": 32,
                  "pCmd": 49,
                  "hwpId": 65535,
                  "pData": []
            },
      "returnVerbose": true
       }
};
```

```
msg.payload = {
      "mType": "iqrfRawHdp",
      "data": {
            "msgId": "Light",
            "req": {
                  "nAdr": 15,
                  "pNum": 32,
                  "pCmd": 49,
                  "hwpId": 65535,
                  "pData": [??, ??]
            },
      "returnVerbose": true
       }
};
```

3.4. Parkeringsplass (sensor)



DPA:

- **NADR:** 0x10 (16), 0x11 (17), 0x12 (18), 0x13 (19)
- **PNUM:** 0x23 (35)
- **PCMD:** 0x00 (0)
- **HWPID:** 0xFFFF (65535)

Elektriske merknader:

- **V_{in}:** 5V, 3.3V
- **I_{max}:** 20mA (fra 5V), 10mA (fra 3.3V)
- **Sensor:** Hall effect

Parkeringsplass består av to deler.
Selveste sensorene for parkeringsplassen og skjermen som viser antall ledige plasser. Det er en Hall Effect switch sensor, dvs. den bryter ved stort nok magnetfelt. Under bilene er det magneter. Det er LED-lys som viser om plassen er ledig.

HEX-format (eksempel for parkeringsplass 1, da alle er like)

	DPA Request	**DPA Response –** **0x00 hvis ledig og 0xFF hvis opptatt**
Spørre etter ledig plass	00.10.23.00.FF.FF	00.10.23.00.FF.FF.??.??.**PData[0]**

JSON-format

Venstre: spørre etter ledig plass

```
msg.payload = {
      "mType": "iqrfRawHdp",
      "data": {
            "msgId": "Parking",
            "req": {
                  "nAdr": 16,
                  "pNum": 35,
                  "pCmd": 0,
                  "hwpId": 65535,
                  "pData": []
            },
      "returnVerbose": true
       }
};
```

Høyre: får tilbake at det er f.eks. opptatt

```
msg.payload = {
      "mType": "iqrfRawHdp",
      "data": {
            "msgId": "Parking",
            "req": {
                  "nAdr": 16,
                  "pNum": 35,
                  "pCmd": 0,
                  "hwpId": 65535,
                  "pData": [255]
            },
      "returnVerbose": true
       }
};
```

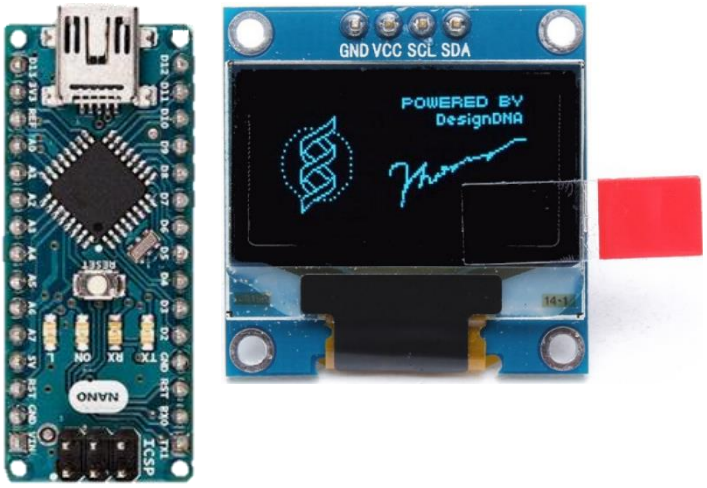
3.4.1. Skjerm

DPA:

- **NADR:** 0x14 (20)
- **PNUM:** 0x24 (36)
- **PCMD:** 0x00 (0)
- **HWPID:** 0xFFFF (65535)

Elektriske merknader:

- **V_{in}:** 5V, 3.3V
- **I_{max}:** 20mA (fra 5V), 10mA (fra 3.3V)
- **Arduino Nano**
- **0.96'' OLED I2C Skjerm**



TR-72DA modulen er koblet til en Arduino Nano som er koblet til en 0.96'' OLED skjerm via I2C. Etter at gateway har spurt alle fire parkeringsplasser om de har ledige og fått tilbake svar. Så kalkuleres det ut hvor mange ledige plasser det er igjen og sender det som en tallverdi til skjerm-kretskortet. Hvor det sendes videre til Arduino Nano via UART, som igjen har gjort klart skjermen og oppdaterer med tallet på ledige parkeringsplasser.

HEX-format

DPA Request	PData[0]
Sende X antall ledige plasser, f.eks. at det er 3 ledige plasser	00.14.24.00.FF.FF.**03**

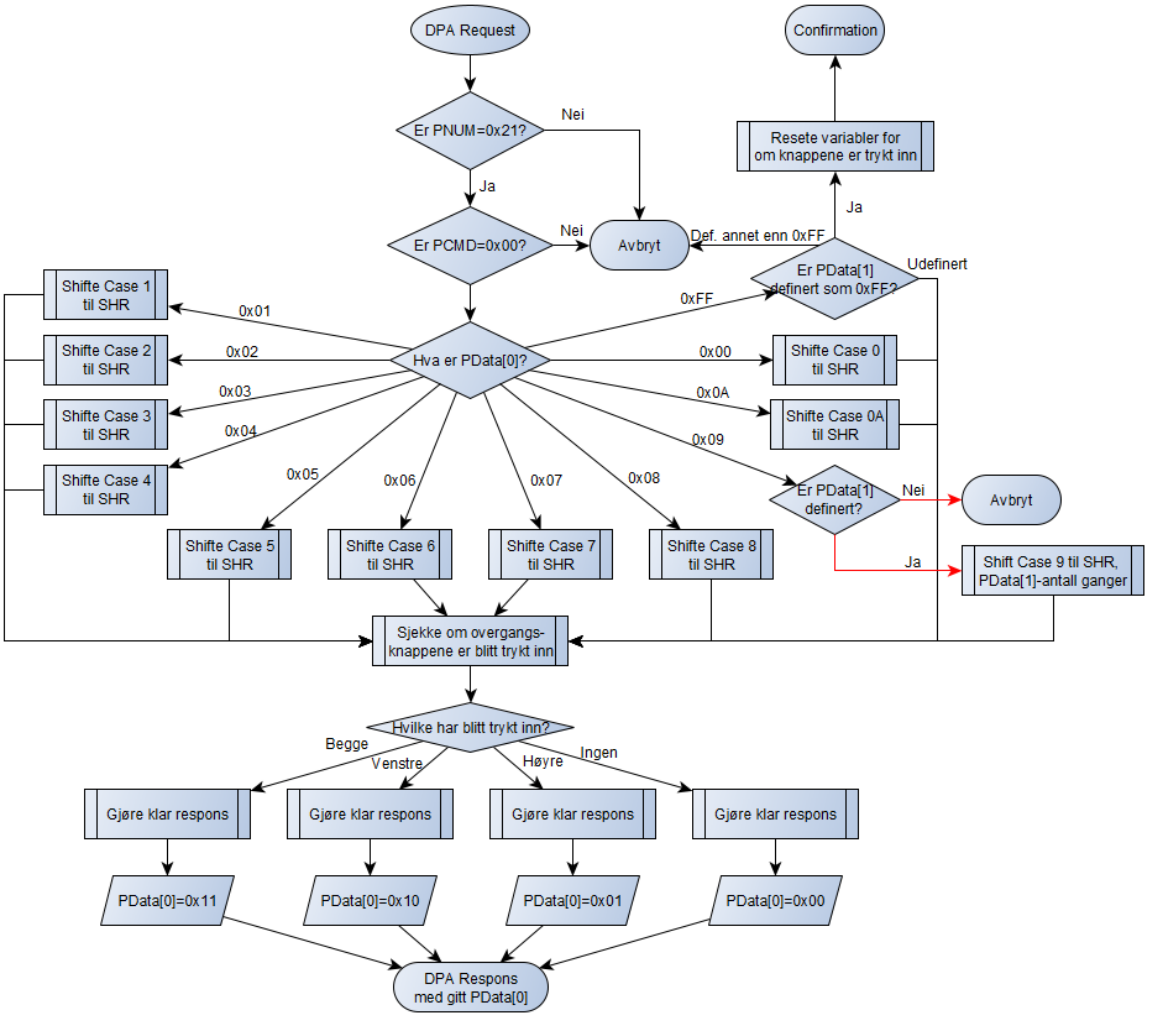
JSON-format

Sende X antall ledige plasser, f.eks. at det er 3 ledige plasser

```
msg.payload = {
      "mType": "iqrfRawHdp",
      "data": {
            "msgId": "Parking Screen",
            "req": {
                  "nAdr": 20,
                  "pNum": 36,
                  "pCmd": 0,
                  "hwpId": 65535,
                  "pData": [3]
            },
      "returnVerbose": true
       }
};
```

3.5. Trafikklys

LOL



3.6. Gateway og cloud

3.7. Oppsummering

Node Adresse	Tilhører Modul	Kommentar	V_{in}	I_{max}	I_{typ}	PNUM	DPA Request Pdata[x]

DEC	HEX			*alle har GND*		*ved typisk bruk*		0	1	2	3	4
0	0x00	Gateway	RPi	5V microUSB			0x00					
1	0x01	Gatelys	Nr. 1	5V	60mA[1]	30-50mA	0x20	**AV**: 0x00 **PÅ**: 0x00 **DIM PÅ**: 0x01 **DIM AV**: 0x01	0x00 0xFF 0x00 0x00	0x01	0xFF	0x00 0xFF
2	0x02	Gatelys	Nr. 2	5V	60mA[1]	30-50mA	0x20					
3	0x03	Gatelys	Nr. 3	5V	60mA[1]	30-50mA	0x20					
4	0x04	Gatelys	Nr. 4	5V	60mA[1]	30-50mA	0x20					
5	0x05	Gatelys	Nr. 5	5V	60mA[1]	30-50mA	0x20					
6	0x06	Gatelys	Nr. 6	5V	60mA[1]	30-50mA	0x20					
7	0x07	Gatelys	Nr. 7	5V	60mA[1]	30-50mA	0x20					
8	0x08	Temp. rom	Varme	12V, 5V	1.8A[2]	1.65-1.75A	0x22	**AV:** 0x00 **PÅ:** 0x01				
9	0x09	Temp. rom	Kjøle	12V, 5V	150mA	150 mA	0x22					
10	0x0A	Temp. rom	Sensor	5V			0x22	**INGEN PDATA FOR REQUEST.**				
11	0x0B	Trafikklys	Nord	12V, 5V	200mA[3]	70-80mA	0x21	**ALLE LYS AV/PÅ**: 0x00 (0) / 0x0A (10) **CASE 1-8**: 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08 **CASE 9/WORK M**: 0x09 (9)				**ANTALL BLINK**: 0x XX
12	0x0C	Trafikklys	Sør	12V, 5V	200mA[3]	70-80mA	0x21					
13	0x0D	Trafikklys	Øst	12V, 5V	140mA[3]	50-60mA	0x21					
14	0x0E	Trafikklys	Vest	12V, 5V	140mA[3]	50-60mA	0x21					
15	0x0F	Gatelys	Lyssensor	5V m/ batteri			0x20	**INGEN PDATA FOR REQUEST.** **MEN** PCMD er 0x31 eller 49				
16	0x10	Park. plass	H.E. sensor	5V, 3.3V			0x23	**INGEN PDATA FOR REQUEST.**				
17	0x11	Park. plass	H.E. sensor	5V, 3.3V			0x23					
18	0x12	Park. plass	H.E. sensor	5V, 3.3V			0x23					
19	0x13	Park. plass	H.E. sensor	5V, 3.3V			0x23					
20	0x14	Park. plass	Skjerm	12V, 5V			**0x24**	**SEND ANTALL LEDIGE PLASSER:** 0x XX				
21	0x15	Kodelåst dør	Servo	12V, 5V	190mA	15mA[4] 140mA	0x25	**LÅSE / ÅPNE DØR**: 0x00 (0) / 0x01 (1)				

[1]: Dette er ved maks lysstyrke på gatelysene.

[2]: Dette er fra 12V uttak. Rundt 10mA fra 5V.

[3]: Teoretisk 20mA per LED aktiv. Det er 10 LED på node 11 og 12, og 7 LED på node 13 og 14.
[4]: 15mA gjelder når den står stille dvs. statisk strømtrekk.

NB! PCMD er lik 0x00 eller 0 på **alle, bortsett fra** node 15 lyssensor som har PCMD lik 0x31 eller 49.

4. Forslag til oppgraderinger i fremtiden