|  |
| --- |
| Benjamin Larsen  24-06-2022 |

|  |
| --- |
| **Senia** |
| Temperatur Styring |
|  |



|  |
| --- |
| **Elev:**  Benjamin Elif Larsen |
| **Projekt:**  Temperatur Styring |
| **Uddannelse:**  Datatekniker med speciale i programmering |
| **Projektperiode:**  21/06/2022 – 24/06/2022 |
| **Afleveringsdato:**  24/06/2022 |

# Titelblad



Techcollege Aalborg,

Struervej 70,

9220 Aalborg

# Forord

Hvad skal man vide om rapporterne og projektet før man går i gang med at læse?

Det er en god ide at læse ’Case Beskrivelsen’ først, da denne forklarer hvad der skulle udvikles og hvorfor.

Hvis man vil benytte projektets kode, skal man bruge Visual Micro for Visual Studio 2022 og have Optiboot på sin Arduino. Grunden til dette er pga. debugging og kodefilerne er ikke bare Arduinos Ino, men også C, CPP og Header filer. Case.py filen skal køres på en Raspberry Pi som også har docker-compose filen kørende.

# Case beskrivelse

Casens udfordring er at udvikle et system, på en Arduino, der kan sende data til et andet system, en Raspberry Pi, som sender data’en videre til en MQTT, som ligger på Raspberry Pi’en. Dette betyder at der er flere steder imellem systemer der kan fejle, hvor systemerne skal håndtere disse fejl. ...

Problemet dette case skal læse er temperatur styring i et område som et rum eller bygning. Normalt har folk eller virksomheder temperatur regulering aktiv hele tiden, selv mens folk ikke er i området. Dette kan være udluftning eller radiator som styrer varmen i et område.

Denne case er omkring at udvikle det del af systemet som registre temperaturen og sende den til en MQTT.

# Kravspecifikation

Kravspecifikationen, givet nedenunder, er for de vigtigste dele af systemet. Disse er kommunikation mellem Arduino og Raspberry, LED’er der benyttes til at vise tilstanden på Arduino’en. Sikkerheden er også vigtig, da systemet skal benyttes til at styre temperatur og dermed skal det undgås at andre kan påvirke temperaturstyringen.

Kravspecifikation – Kravsoversigt

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| KravId | Kategori | Krav | Prioritering | Kilde | Type |
| K-1 | Kommunikation | Data kan sendes via ZigBee | 2 | Protocol | Brainstorm |
| R-2 | Kommunikation | Raspberry skal kun modtage data fra bestemte, kendte enheder | 3 | Sikkerhed | Brainstorm |
| A-1 | Software | Arduino kan læse fra HC-SR04 sensor. | 1 | Hardware | Brainstorm |
| A-1-1 | Software | Arduino læser en enkel gang og venter indtil HC-SR04 ikke længer sender at noget er foran den før arduino læser igen | 1 | Funktion | Brainstorm |
| A-2 | Software | Arduino kan tænde og slukke LEDerne som benyttes til at vise state | 1 | Hardware | Brainstorm |
| A-2-1 | Software | Slukning og tænding sker ud fra bestemte states i koden | 1 | Funktion | Brainstorm |
| A-3 | Software | Den skal benytte state machine til at styre hvad kode der skal køres | 1 | Arkitektur | Brainstorm |
| A-4 | Kommunikation | Arduino kan kun sende til en enkel, bestemt, Xbee | 3 | Sikkerhed | Brainstorm |
| A-5 | Fejlhåndtering | Arduino skal kunne håndtere hvis ZigBee fejler | 2 | Klient | Brainstorm |
| R-1 | Kommunikation | Raspberry skal kunne sende data til MQTT | 1 | Protocol | Brainstorm |
| R-1-1 | Fejlhåndtering | Raspbery skal kunne håndtere hvis den ikke kan kontakte MQTT | 2 | Server | Brainstorm |
| R-3 | Fejlhåndtering | Raspberry skal kunne håndtere hvis den ikke kan åbne Xbee porten | 2 | Server | Brainstorm |
| A-6 | Software | Arduino skal kunne læse fra en temperatur sensor | 1 | Funktion | Brainstorm |
| A-6-1 | Software | Arduino skal have en timer, 10 sekunder, mellem temperaturlæsningerne | 1 | Funktion | Brainstorm |
| R-1-2 | Kommunikation | Raspbery skal kunne sende data til det rigtige topic | 1 | Funktion | Brainstorm |

Sikkerheden i system skal teste for at sikre at andre ikke kan udnytte det. På sammen tid skal der undersøges om både Arduino’en og Raspberry Pi’en kan håndtere at sende og modtage data, da dette er en vigtig del af systemet. Arduion’en skal kunne benytte LED’erne for at vise sin tilstand, da det ikke vil være muligt at debugge den i det færdige produkt, samt at temperatur delen virker korrekt.

Det meste af systemet skal teste via debugging. Den kode der kører på Raspberry Pi’en har ikke de samme debugging muligheder, men der kan, via prints og if-statments, tjekke hvad kode der køre og hvad data’en er. Der er muligt at ’debugge’ MQTT via en subscription i en konsol på det topic der sendes til.

Accepttestoversigt

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| KravId | Testbeskrivelse | Testkriterier | Bemærkninger | Status |
| K-1 | Der undersøges om Xbee'en på Arduino'en kan sende til raspberry pi'ens | Arduino kan sende en pakke som modtages korrekt af Raspberry Pi |  | K-1 |
| R-2 | Der undersøges om Raspberry Pi kan droppe det data der sendes af en ukendt Xbee | Data skal ikke analysers og sendes til MQTT, hvis sender addressen ikke er i bestemt liste | Kræver en 3rd Xbee | R-2 |
| A-1 | Der undersøges om Arduino kan læse fra HC-SRO4 sensor | Data skal være læst korrekt og uden forsinkelser | Benyt en debugger | A-1 |
| A-1-1 | Der undersøges om Arduino kan registrere når der ikke er noget foran sensoren | Data fra sensoren skal føst læse og derefter skal den vente til der ikke er noget foran den |  | A-1-1 |
| A-2 | Der undersøges om Arduino'en kan starte og slukke for dens state LEDer | Dette skal gøres via en enkel metoder kald der kan bruges for alle LED'er |  | A-2 |
| A-2-1 | Der undersøges om Arduino kan benytte statemachine til at tænde og slukke for de rigtige LED'er | Kun en enkel LED skal være tændt ad gangen. Hver state skal tænde og slukke for sin egen LED | LED'en position og port skal være en del af et array | A-2-1 |
| A-3 | Der undersøges om den rigtige kode er kaldt I Arduino'en ud fra hvad state machine siger | Den rigtige kode kaldes ud fra state machinen |  | A-3 |
| A-4 | Der undersøges om Arduino kun sender data til en bestemt XBEE | Arduino sender til en MAC addresse og ikke broadcast |  | A-4 |
| A-5 | Der undersøges om Arduino kan håndtere fejl i ZigBee protokollen | Arduino er ikke koordinator og skal ikke bryde ned hvis den fejler at sende |  | A-5 |
| R-1 | Der undersøges om Raspbery Pi kan sende data til MQTT | Det sendte data skal kunne læses fra en subscription og passe overens med det sendte data |  | R-1 |
| R-1-1 | Der undersøges om Raspbery Pi håndterer hvis den fejler i at sende til MQTT | Programmet skal kunne forsætte hvis den fejler |  | R-1-1 |
| R-3 | Der undersøges om Raspberry Pi kan håndtere hvis den ikke kan åbne XBEE Port | Programmet kan give en brugbar fejlbesked |  | R-3 |
| A-6 | Der undersøges om Arduino'en kan læse korrekt fra en temperatur sensor | Den læste temperatur skal passe overens med rummets |  | A-6 |
| A-6-1 | Der undersøges om Arduino'en kan læse på de korrekte tidspunkter | Der skal gå 10 sekunder + kode køre tid mellem hver læsning |  | A-6-1 |
| R-1-2 | Der undersøges om Raspberry Pi'en kan sende til MQTT på de rigtige topics | Raspberry Pi skal kunne ende på 'Temperatur' og en subscription læser fra dette topic |  | R-1-2 |

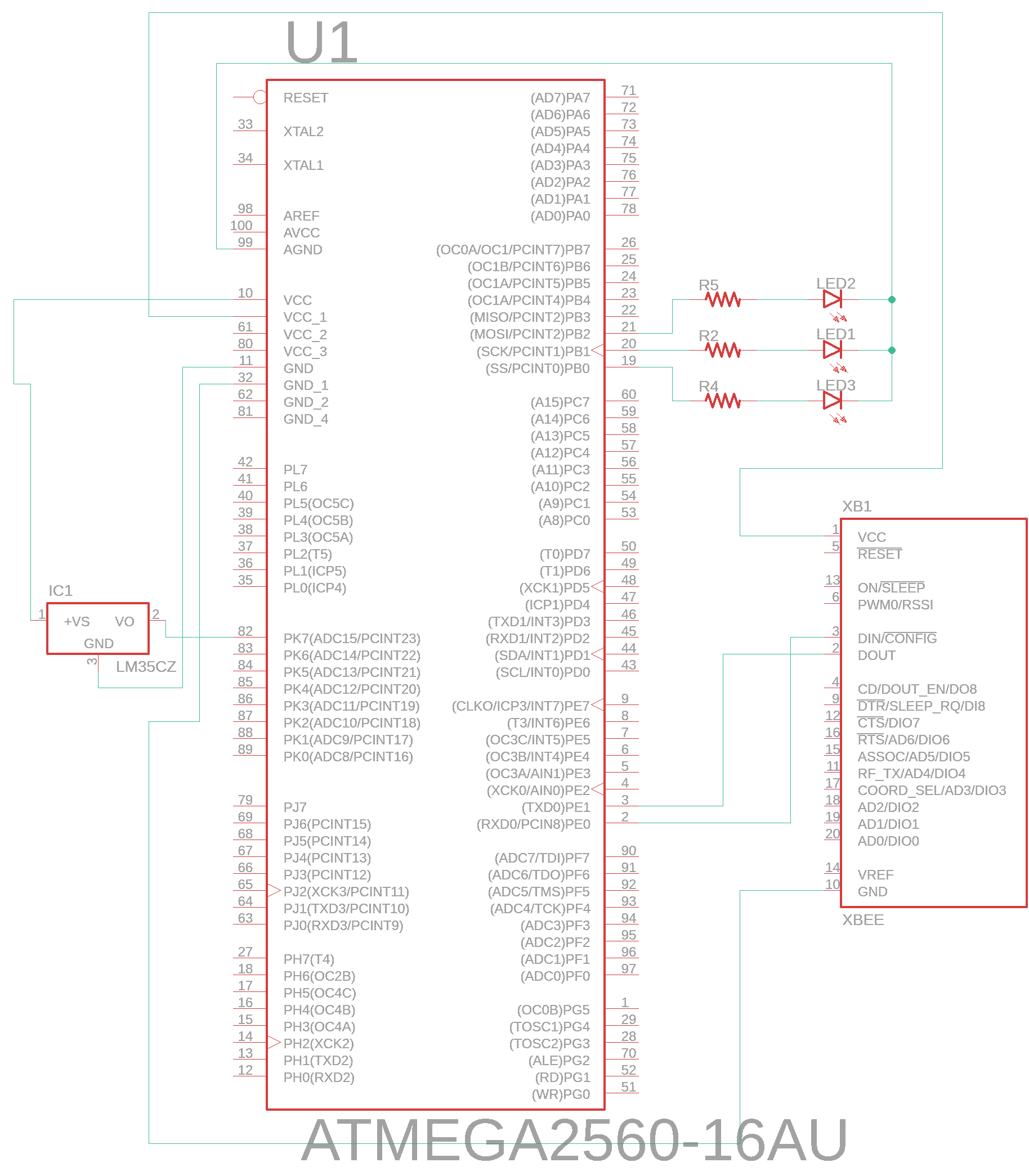
# Hardware opstilling

I forhold til hardware er der to mikrocontrollers, en Arduino Mega 2560 og en Raspberry Pi. Alle hardware komponenter var tilsluttet Arduinon’en medundtagen af en enkel XBEE S2C som var tilsluttet Raspberry Pi’en.

LED’erne var placeret på sit eget breadboard og LM35’en var placeret på et andet breadboard.

Komponent liste

* Raspberry Pi
* Arduino Mega 2560
* LED x 3 (Grøn, gul og rød)
* LM35
* HC-SR04
* XBEE S2C x 2
* Modstande x 3



Figur 1 Eagle Diagram

# Program dokumentation

Casen indeholder to programmer, et Python program, kørt på en Raspberry Pi, og en Arduino program, kørt på en Arduino Mega 2560.

## Arduino

Dette program benytter en statemachine til at styre hvad kode der skal køres, samt to timer til at styre hvornår kode køres. Der er tre hoved states, IDLE, READING\_TEMPERATURE og TRANSMITTING\_DATA. IDLE staten benyttes til at læse fra HC-SR04’en og hvis noget er tæt nok på vil den aktivere de to timer. Den første timer styre hvornår IDLE kan læse fra HC-SR04’en igen, da den kan afslutte den sidste timer. Den sidste timer vil, når den trigger, skifte state til READING\_TEMPERATURE.

Se tekstfeltet nedenunder for det forklarede kodeblok

READING\_TEMPERATURE staten læser fra temperaturmåleren, LM35, og skifte state til TRANSMITTING\_DATA. TRANSMITTING\_DATA staten vil sende data’en ud via den tilsluttede XBee og skifte tilbage til staten IDLE.

void DistanceHandling(void)

{

while (IDLE == State && true == CanReadHC)

{

TimerUpdates();

int distance = ReadHCSR04();

if (10 <= distance && 100 >= distance)

{

CanReadHC = false;

timerRead.start();

if (!TimerTempActive) {

timerTemp.start();

TimerTempActive = true;

}

else {

timerTemp.stop();

TimerTempActive = false;

}

}

}

}

Pga. hvordan koden er sat op, er koden nedenunder det eneste der skal kaldes via loop’en til at køre alle states i programmet, hvilket gør det let at udvide.

void loop()

{

TimerUpdates();

MachineStateInRAM\_Pointer = StateAllocateMemoryInRamAndGetCopyFromFlashProm();

MachineStateInRAM\_Pointer->FunctionPointer();

}

## Python

# Konklusion

Skal hænge sammen med case beskrivelsen og problemformuleringen.

Svar på om problemet er løst, brug Kravspecifikationen til at se hvad der mangler eller at alt opfylder kravet.

Hvad var udfordrende at få til at virke?

Der var en udfording i udviklingen af produktet, da den kode der skulle styre DDR og PORT ikke fungeret korrekt. Den blev ved med at vælge en andre adresse, så det var nødsaget til at benytte pinMode og digitalWrite.

Det meste af problemet er blevet løst, dog blev kravet A-1-1 ændret lidt, da det blev valgt at udvikle koden sådan at Arduion’en ikke kan læse for 2 sekunder efter den læste at noget var tæt nok på.

## Diskussion

Diskutere forskellige side af løsningen, fordele/ulemper.

Ulemper ved løsningen er at koden i Raspberry Pi’en skal ændres når der skal tilhøre en ny XBEE S2C, da den ikke tillader alle at snakke med den via ZigBee, dette blev dog gjort pga. sikkerhed.

Hvis der var mere tid, hvor kunne produktet udvides eller forbedres.

Hvis der havde være mere tid, kunne flere Arduino’er været sat op eller havde sat den nuværende op sådan at den ville tjekke om en person var på vej ind eller ud af området via flere HC-SR04’er. Lige nu tjekke den bare om nogen kommer forbi den. På sammen tid kunne man have sat noget kode op til at benytte det data der blev sendt hen til MQTT’en.

Reflekter over projektet

Hvad har du lært?

Valgte du de rigtige teknologier?

# (Referencer)

Denning, T. (2018, July 17). 15 Typical Life Problems And How To Solve Them. *Mission*, 4. Retrieved Januar 6, 2021, from https://medium.com/the-mission/15-typical-life-problems-and-how-to-solve-them-c56838f49738

# (Bilag)

Indholdsfortegnelse

Bilag A - Arduino Mega Pinout 4

Bilag B - Dagbog/Logbog 5

#### Arduino Mega Pinout

#### placeholder (1) - Nomad Foods

Figur 2 Biag A: tekst

#### Bilag