

Examensarbete

Dataingenjör 180hp



433 MHz mesh protokoll för smarta hem

Datateknik 15 hp

Halmstad 2020-08-30

Joacim Wärn

Abstract

In today's smart home systems, the sensor does not send back a response to the master computer so that the master computer has acknowledged that the packet has been received by the sensor and then that the sensor has carried out the command that was requested by the master.

This report focuses on implementing a response from the sensors back to the master. The master in this report is the RaspberryPi which acts as the controller for all the sensors in the network, and the nodes are Arduino Uno's.

The resulting system shows that it is possible to implement techniques such as CSMA/CA and a TCP like protocol in order to achieve a system with response.

Sammanfattning

I dagens smart hem skickas paket från en master som plockas upp av sensorer runt om i huset. Sensorerna har ingen dubbelriktad radiokommunikation för att verifiera att de har mottagit paket från mastern.

Denna rapport är fokuserad på implementering av svar från sensorer till mastern, som i sig själv är en RaspberryPi. Mastern agerar som kontrollern för alla sensorer i nätverket, noderna i nätverket är Arduino Uno.

Det resulterade systemet visar att det är möjligt att implementera tekniker som CSMA/CA och ett TCP liknande protocol för att få ett system med respons.

Innehåll

1 Inledning	6
1.1 Syfte och mål	6
1.2 Avgränsningar	6
1.3 Kravspecifikation	7
1.4 Frågor	7
2 Bakgrund	8
2.1 Trådlös kommunikation	8
2.1.1 ASK	8
2.1.2 Tvåvägskommunikation	9
2.1.3 Hidden node problem	9
2.1.4 Exposed node problem	10
2.2 Mesh-nätverk	11
2.2.1 Nod	11
2.2.2 Server	11
2.2.3 Zigbee	11
2.2.4 Z-Wave	11
2.3 Teori	12
2.3.1 Protokoll	12
2.3.2 Exempel på två välkända protokoll	12
2.3.3 OSI-modellen och TCP/IP stacken	13
2.3.4 Skapa ett eget protokoll	14
3 Metod	15
3.1 Projekt faser	15
3.1.1 Bakgrunds fas	15
3.1.2 Val av hårdvara	15
3.1.2.1 Arduino Uno	15
3.1.2.2 Raspberry Pi	15
3.1.2.3 PT2272	16
3.1.2.4 PT2262	16
3.1.3 Implementation	16
3.2 Verktyg	16
3.2.1 VNC viewer	16
3.2.2 C/C++	16
3.2.3 Arduino	17
3.2.4 rc-switch	17

3.2.5 WiringPi	17
3.2.6 Gnu Compiler Collection	17
3.2.7 Raspberry Pi OS	17
3.3 Implementering av protokoll	18
3.3.1 Fysiska skiktet	18
3.3.2 Datalänk och nätverksskiktet	19
3.3.3 Transportskiktet	20
4 Resultat	21
4.1 Test med server och nod, tilldelning av adress	21
4.2 Test med server och nod, tända och släcka lampa	22
4.3 Test med server och flera noder	23
4.4 Slutresultat, illustration av färdig produkt	25
5 Diskussion	27
5.1 Uppfyllda samt ej uppfyllda mål	27
5.2 Styrkor i protokollet	27
5.3 Brister i protokollet	28
5.4 Tester som utförts	28
5.4.1 Tilldelning av adress med en nod och en server	28
5.4.2 Test med server och nod, tända och släcka lampa	29
5.4.3 Test med server och flera noder	29
5.5 Slutresultat av produkt	29
5.5.1 IP	29
5.5.2 TCP	30
5.5.3 Protokollstack	30
6 Slutsats	31
6.1 Erfarenheter	31
6.2 Fortsatt arbete	31
6.3 Resultat jämfört mot mål	31
7 Referenser	32

1 Inledning

Hemautomation också kallat smarta hem är system där användaren kan hålla koll på exempelvis energiförbrukning av kyl och frys, klimatkontroll, regnmätare, tända och släcka lampor, med så kallade adaptrar som kan fungera som strömbrytare(exempelvis tända, släcka lampa) eller som en sensor(exempelvis mäta energiförbrukning hos en kyl).

Sensorerna och adaptrar ansluts oftast via en central server, och användaren kan därifrån styra dem olika adaptrarna via ett användargränssnitt som finns i en applikation till en dator eller en mobil.

Sensorerna och adaptrarna kommunicerar i frekvenser ända från 433 MHz ända upp till 2.4 GHz.

Om användaren väljer att tända eller släcka en lampa som är ansluten med en adapter så skickar den tillbaka ett svar till servern och via ett användargränssnitt hos servern så går det att se att det har hänt.

I ett 433 MHz system saknas det status på noder på grund av att det inte finns tvåvägskommunikation.

Huvudfokus i detta projekt är att utveckla ett system som använder 433MHz där noderna ska kunna skicka en respons till servern.

1.1 Syfte och mål

Utveckla ett trådlöst mesh-nätverk med noder som kan kommunicera med varandra genom att använda halv-duplex över 433 MHz.

1.2 Avgränsningar

- Tester kommer att utföras vid 433 MHz.
- Tester kommer att utföras i en miljö där vi har minimala störningar.

1.3 Kravspecifikation

- Noden ska både kunna skicka och ta emot data.
- Servern ska inte behöva vara inom räckvidd för att nå en viss nod, datan ska kunna vidarebefordra från nod till nod tills den når sin slutdestination.

1.4 Frågor

- Är det möjligt att skapa ett trådlöst mesh nätverk med 433MHz och filtrera bort oönskad data, om data som inte var ämnat till noden så förkastas datan.
- Är det möjligt att implementera ett nätverksprotokoll som stödjer respons som till exempel TCP genom att ta konceptet av TCP protokollet och implementera dem funktioner som behövs för att uppnå respons.

2 Bakgrund

Ett trådlöst mesh-nätverk [1] är ett system som är uppbyggt av två enheter, en server och en eller flera noder. I ett mesh-nätverk kan noderna skicka data mellan varandra och tillåter då noderna att fungera som en brygga för kommunikation mellan noder. Resultatet blir att servern inte behöver vara inom räckvidd för att nå en viss nod, datan kan då istället hoppa från nod till nod tills den når sin slutdestinationen.

Det finns redan protokoll som använder mesh-nätverk topologin som vi kommer gå in i mer djupare detalj hur dem fungerar.

2.1 Trådlös kommunikation

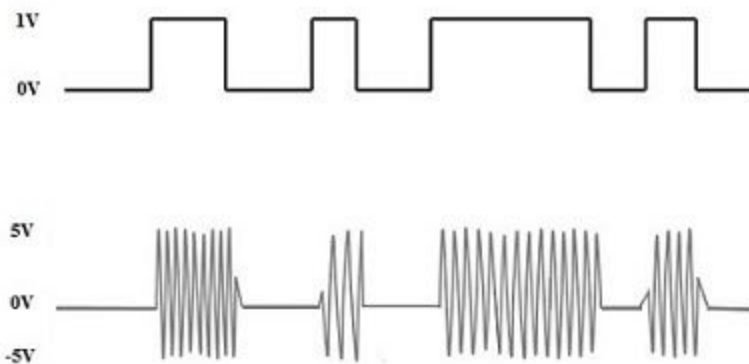
I denna sektion kommer vi att gå igenom den teknik som används i vårt mesh-nätverk för att skicka data och ta emot data. Vi kommer bland annat nämna ASK och tvåvägs kommunikation med eller utan respons, vi kommer även att gå in på två kända problem som finns inom trådlös kommunikation, "Hidden node problem" och "Exposed node problem".

2.1.1 ASK

Amplitude-shift keying(ASK) [2] också kallad amplitudmodulering, är den teknik som detta projekt kommer att använda för att skicka signaler.

Anledning till detta är för att den hårdvara som valts till systemet använder denna teknik för att skicka signaler.

När datan ska övergå till en analog signal måste först den först gå igenom en process som kallas digital analog konvertering(DAC). I denna process representeras en digital "0" bit som en bärvåg utan amplitud, och en digital "1" bit skapar en amplitud i bärvågen, se figur 1.



Figur 1: Digital '0' respektive digital '1' överst i bilden, nederst motsvarig modulation av bärervågen från radiosändaren

2.1.2 Tvåvägskommunikation

Ett system med dubbelriktad kommunikation [3] möjliggör att noden kan skicka tillbaka ett svar. I jämförelse kan noden i envägskommunikation bara ta emot eller skicka.

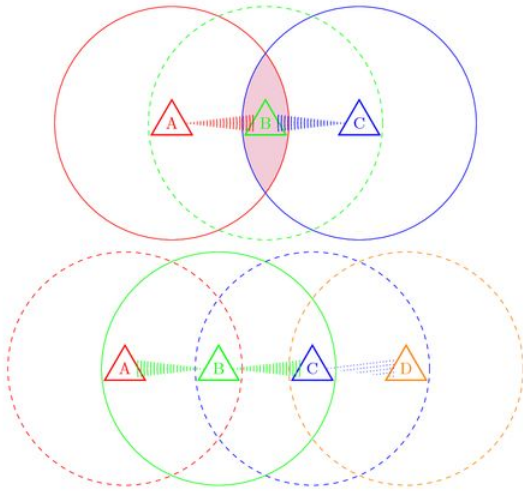
I system som använder respons förekommer det fördröjningar på grund av att sändaren antingen måste vänta tills den får tillbaka ett svar eller välja att skicka ett nytt paket på grund av för lång väntetid.

Skillnaden med ett system som inte använder respons är att sändaren inte behöver vänta på ett svar, men kan då inte garantera att paketet har nått sin slutdestination.

2.1.3 Hidden node problem

“Hidden node problem” är ett problem som uppstår i trådlösa nätverk när en nod vill kommunicera med en annan nod som är utom räckhåll, se figur 2, problemet som uppstår är att om nod A kollar om data skickas över kanalen och det råkar hända att nod C skickar data till nod B, så kan ej nod A se detta och så börjar nod A att skicka data till nod B och så slutar det med att ingen data kommer fram utan det blir bara en kollision av paket.

Lösningar till problemet är att exempelvis öka räckvidden för en nod så att den kan lyssna på en större räckvidd, eller att helt enkelt flytta noderna närmare varandra så att de kan se varandra.



Figur 2: Hidden node problem överst, Exposed node problem underst

2.1.4 Exposed node problem

Exposed node problem är ett problem som uppstår i trådlösa nätverk om två stycken sändare i detta fall nod B och nod C vill skicka data till respektive nod A och nod D, se figur 2, sändare B och C skickar data samtidigt så kommer det bli en kollision och data som skickats kommer inte att komma fram.

Lösningen till detta problem var att varje sändare måste först lyssna så att ingen data håller på att överföras på kanalen, om noden märker att data skickas över kanalen så får noden vänta och sedan lyssna igen om det går att skicka över kanalen, kallas för CSMA/CA(Carrier Sense Multiple Access / Collision Avoidance).

2.2 Mesh-nätverk

Mesh-nätverk är en typ av nätverkstopologi där noderna är direkt anslutna till varandra så att data kan vidarebefordras från nod till nod utan att behöva ha någon switch eller router emellan dem som dirigerar vart datan ska gå till.

2.2.1 Nod

En nod i ett mesh-nätverk kan agera som en router på så vis att när den får ett paket som inte var ämnat till just den noden kan den skicka paketet vidare till noder som finns inom räckhåll.

2.2.2 Server

En server i ett mesh-nätverk fungerar som instruktions givare till noder, härifrån kontrolleras vilken typ av instruktion som ska skickas till vilken nod.

2.2.3 Zigbee

Zigbee [4] är ett nätverksprotokoll som använder mesh teknologi. Det används främst inom hemautomation för att trådlöst styra elektronikprodukter i hemmet och även övervaka energiförbrukning på t.ex kyl och frys.

2.2.4 Z-Wave

Z-wave [5] är ett nätverksprotokoll som också använder mesh teknologi precis som Zigbee, den stora skillnaden mellan protokollen är att Z-wave använder 868MHz för att skicka data medans Zigbee kan användas från 868MHz ända upp till 2.4 GHz.

2.3 Teori

Vid detta kapitel ges en förståelse för vad ett protokoll är för något samt TCP/IP stacken och OSI modellen.

2.3.1 Protokoll

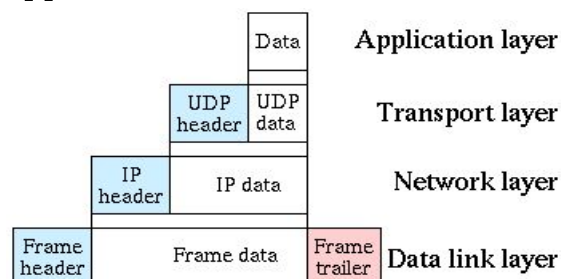
Ett protokoll är en överenskommelse av enheter i ett nätverk för att ha kännedom hur data ska skickas, behandlas och bearbetas. Det är bland annat hur stort ett paket får vara för att skickas från ett nätverk till ett annat (fragmentering), och hur många noder ett paket kan vidarebefordras innan paketet dör ut som kallas för TTL(Time To Live), varje gång ett paket vidarebefordras från en nod till en annan så sänks värdet på TTL med 1, som exempelvis löser problem så att ett paket inte vidarebefordras mellan noder för evigt.

2.3.2 Exempel på två välkända protokoll

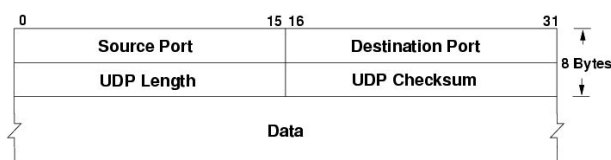
Här utforskar vi hur två välanvända protokoll som UDP och TCP fungerar och hur det packas ihop från det fjärde lagret i OSI-modellen som heter transportskiktet(se figur 3). Vi visar nedbrytningen till 1:or och 0:or i det fysiska lagret där de sedan skickas via kabelanslutning eller trådlös radiokommunikation som WiFi exempelvis.

UDP protokollet vilket jobbar vid det fjärde lagret i OSI-modellen hanterar data som ska skickas (se figur 4), vilken port det ska skickas ifrån och vilken port det ska skickas till på mottagarens sida. Det innehåller också storleken på paketet och även en checksumma.

TCP protokollet har istället flera fält i sin header på grund av att TCP använder respons när det skickar data. Då mottagaren behöver skicka tillbaka ett svar och säga att det antingen har tagit emot datan, eller om blev problem och det behöver skickas igen. UDP använder inte respons utan används mer i tillfällen då det inte är jätteviktigt att varje datapaket har anlänt eller om fel har uppstått.



Figur 3: UDP uppbyggnad

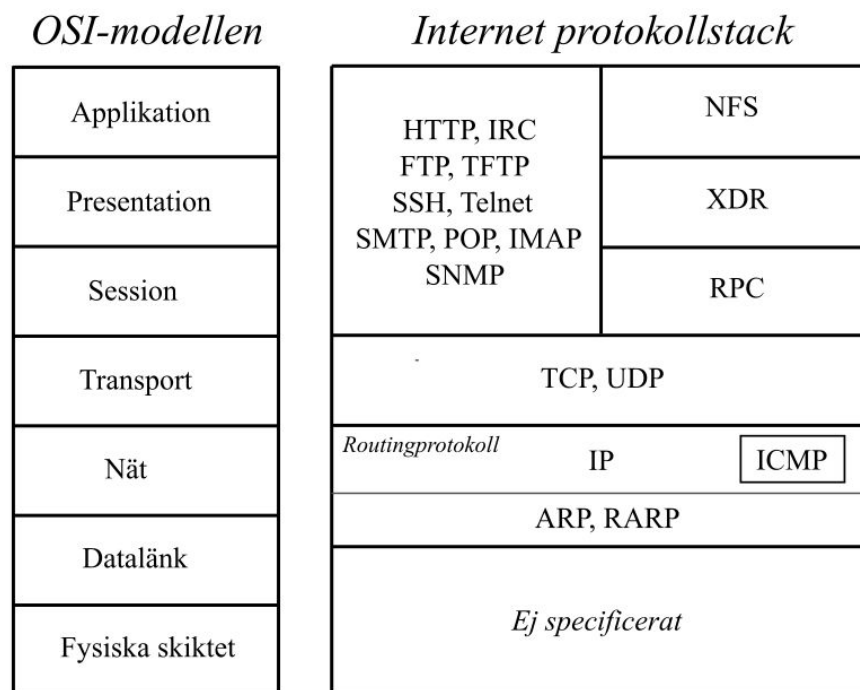


Figur 4: UDP header

2.3.3 OSI-modellen och TCP/IP stacken

TCP/IP stacken och OSI modellen är båda samlingar av protokoll i form av stackar, OSI modellen används inte i praktiken utan används istället som en referensmodell. I figur 2 kan man se både TCP/IP stacken och OSI modellen. Stacken är skapad så att varje lager är inkapslat, exempelvis om det fysiska lagret ändrar sin dataöverföring från WiFi till kabelanslutning påverkar det inte dem andra lagren.

Protokollstacken är likt den process som när ett brev skickas, innehållet i brevet kommer att förbli oförändrat och destination samt sändarens adress kommer vara densamma. Det spelar ingen roll hur hanteringen av brevet sker, om det skickas via flyg, bil eller tåg, innehållet i brevet är detsamma.

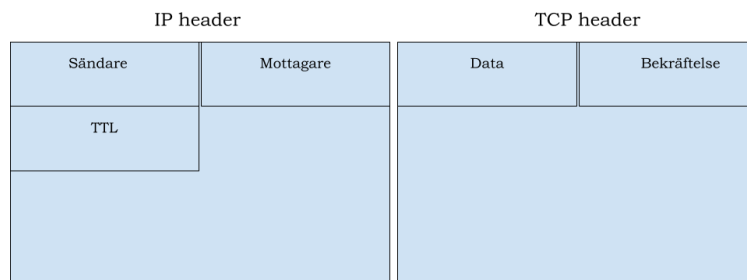


Figur 4: OSI-modell till vänster, TCP/IP stack till höger

2.3.4 Skapa ett eget protokoll

För att kunna skapa ett eget protokoll som ska fungera i ett system vid trådlös kommunikation krävs det att enheterna som finns i systemet har tillräckligt bra prestanda för att ta emot, skicka och bearbeta den data som kommer. Det ska även att kunna hantera fördröjningar eller att enheterna är för långt ifrån varandra vilket resulterar i att data inte kan komma fram.

Se figur 5 för planerat ip och tcp protokoll för dem två protokoll som kommer att byggas.



Figur 5: IP header till vänster, TCP header till höger

3 Metod

3.1 Projektfaser

Projektet är indelat i tre faser, denna sektion täcker vad dem är och går in i djupare detalj.

3.1.1 Bakgrunds fas

I den första fasen samlas information om tidigare relaterade arbeten för att se vilka problem som då uppstått och hur dessa har lösts. Exempelvis “Hidden node problem” (se kapitel 2.1.3) och “Exposed node problem” (se kapitel 2.1.4).

3.1.2 Val av hårdvara

Första fasen gav kunskap så att det gick att börja se efter vad som behövs för att bygga ihop systemet. Det som behövs monteras på noden är en mottagare och en sändare så att det blir tvåvägskommunikation, och samma saker behövs monteras på servern, så att när noden har fått ett paket som har skickats från servern så kan den skicka tillbaka ett svar till servern att den har fått paketet.

Anledning till varför Raspberry Pi valdes som server istället för en Arduino Uno är för att Raspberry Pi har mer processorkraft och mer minne som behövs för att lagra ett grafiskt användargränssnitt.

3.1.2.1 Arduino Uno

Arduino Uno kommer att användas som noder i mesh nätverket.

3.1.2.2 Raspberry Pi

Raspberry Pi används som server i systemet där användaren kan skicka kommandon till noderna.

3.1.2.3 PT2272

Är en mottagare [6] som avkodar signaler med hjälp av amplitudmodulering, och arbetar vid frekvensen 433 MHz.

3.1.2.4 PT2262

Är en sändare [7] som kodar signaler med hjälp av amplitudmodulering, och arbetar vid frekvensen 433 MHz.

3.1.3 Implementation

Sista fasen handlar om implementation av nätverksprotokollet. Se till så att kommunikationen fungerar mellan nod till nod, och nod till server.

Med hjälp av informationen som hämtats ut från bakgrunds fasen så kan systemet börja konstrueras.

3.2 Verktyg

I denna sektion så kommer vi att täcka dem bibliotek och utvecklingsverktyg som används i projektet.

3.2.1 VNC viewer

VNC viewer [8] är ett program som gör det möjligt att fjärrstyra enheter över ett nätverk, VNC står för Virtual Network Computing, och det tillåter användaren att få tillgång till enheter genom fjärrskrivbord, så användaren kan alltså redigera och ändra saker på enheten utan att faktiskt fysiskt behöva vara vid enheten.

3.2.2 C/C++

C++ är ett programspråk som är en förlängning av programspråket C, vid utveckling av inbäddade system så används oftast C eller C++ just förrå bådas snabba exekveringstid, samt att båda har tillgång till lågnivå funktioner som minnesallokering, kernel anrop, lågnivå filhantering, medmera.

3.2.3 Arduino

Arduino [9] är ett utvecklingsverktyg specifikt för mikrokontrollers av märket Arduino. Det är en textredigerare med en integrerad utvecklingsmiljö som tillåter användaren att kompilera kod och överför sedan koden till Arduino enhetens minne via USB.

3.2.4 rc-switch

Rc-switch [10] är ett bibliotek med öppen källkod som innehåller färdiga funktioner för att sända och ta emot data trådlöst över 433 MHz, det stödjer både Raspberry Pi och Arduino plattformar.

3.2.5 WiringPi

WiringPi [11] är ett bibliotek med öppen källkod som underlättar utveckling av applikationer till en Raspberry Pi, det ger tillgång till färdiga kommandon för att kommunicera med GPIO enheterna bland annat.

3.2.6 Gnu Compiler Collection

GCC [12], kort för (Gnu Compiler Collection) används för att kompilera C och C++ kod bland annat. Verket används i detta fall på Raspberry Pi för att kompilera kod.

3.2.7 Raspberry Pi OS

Det operativsystem som körs på Raspberry heter Raspberry Pi OS [13], kallades förut Raspbian OS. Det ger användaren tillgång till en skrivbordsmiljö och det kommer med färdiga verktyg som exempelvis en terminal för att köra kommandon, och webbläsaren chromium.

3.3 Implementering av protokoll

Tillvägagångssätt för att implementera ett nätverksprotokoll med respons i ett trådlöst 433 MHz system kommer att täckas i detta avsnitt.

Liknande som TCP protokollet så använder detta nätverksprotokoll bekräftnings datapaket som skickas från mottagaren till sändaren för att konfirmera att sändningen har lyckats eller misslyckats.

För att se till så att två stycken enheter inte försöker skicka data samtidigt så har en lösning av CSMA/CA implementerats, så att två eller fler enheter inte skickar data samtidigt och orsakar kollisioner i systemet.

3.3.1 Fysiska skiktet

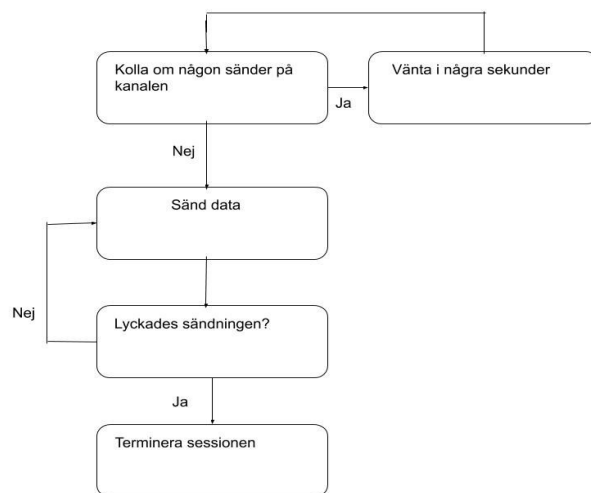
Det som behövs först är att se till så att kommunikationen på grundnivån fungerar, ett färdigt bibliotek som heter rc-switch som stödjer kommunikation över 433 MHz implementerades. Det är ett bibliotek som stödjer dem sändare och mottagare som används i projektet, (PT2262) och (PT2272).

Biblioteket skickar 1:or och 0:or via AM (Amplitudmodulering, se kapitel 2.1.1), och det har ingen högre form av abstraktion utan det är helt inkapslat(mjukvarumässigt), och därför valdes det biblioteket att användas i det fysiska lagret och sedan byggs nätverksprotokollet ovanpå.

3.3.2 Datalänk och nätverksskiktet

Här börjar byggnaden av protokollet, det som behövs vid det här skiktet är att varje enhet har en unik adress, noderna är konfigurerade så att när den startas upp första gången så väntar den på en adress från servern, när en adress har blivit allokerad så försvinner den från serverns tabell av tillgängliga adresser, likaså när en nod stängs av eller inte längre svarar så har servern en "timeout", så att efter x - antal minuter så avallokeras den adress som inte svarar och hamnar tillbaka i tabellen för lediga adresser.

Här implementeras också CSMA/CA, för att få en bättre inblick i hur systemet fungerar så finns det ett schema, se figur 6, och implementationen går att se vid figur 7.



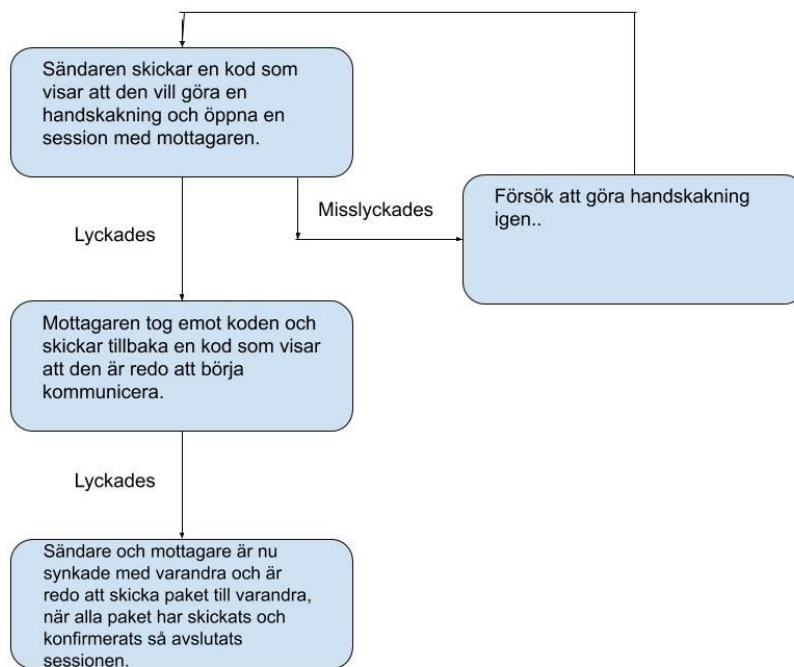
Figur 6: CSMA/CA schema

```
51 /*  
52 CSMA/CA, lyssnar på kanalen och tittar ifall någon sänder.  
53 Om det inte är någon som sänder, börja skicka.  
54 Om någon sänder, wait ett slumpmässigt intervall på 1 - 3 sekunder, och försök igen.  
55 */  
56 void ipv4::csmaca(bool listen){  
57  
58     if(listen){  
59  
60         srand(time(NULL));  
61  
62         //slumpmässig väntetid 1 - 3 sekunder  
63         int randomNumber = (rand() % 3 + 1);  
64  
65         isTransmitting = true;  
66         delay(randomNumber);  
67  
68         return;  
69     }  
70  
71 }  
72  
73 isTransmitting = false;  
74  
75 }
```

Figur 7: CSMA/CA implementation

3.3.3 Transportskiktet

I denna del av protokollet så implementeras en lösning likt det som redan existerar i TCP protokollet, sändaren börjar först med att göra ett “TCP-handshake” genom att skicka en kod till mottagaren, och mottagaren skickar sedan tillbaka en kod och då är både server och noden redo att ta emot eller skicka data till varandra. När sessionen är slutförd och kommandot har utförts så termineras sessionen och kanalen blir nu öppen att sända över. mottagaren skickar tillbaka en konfirmation till sändaren, så att sändaren vet att sändningen lyckades och att datan kom fram, läsaren kan följa schemat nedan för att få en bättre inblick i hur det fungerar(se figur 8), och implementationen av algoritmen kan ses i figur 9.



Figur 8: “TCP-handshake” schema

```
40  /*  
41  Ser till så att server och nod är synkade med varandra och redo att utbyta data.  
42  */  
43  bool ipv4::transmissionSync(int received){  
44      if(received == 777) return true;  
45      return false;  
46  }  
47  
48  
49  }
```

Figur 9: “TCP-handshake” implementation

4 Resultat

Testerna har utförts enligt följande, när en nod startas upp första gången så börjar den att fråga efter en tillgänglig adress från servern. När en tillgänglig adress har delats ut till noden så är den redo att styras från servern.

Kommandon som finns i nuläget är att antingen tända eller släcka en lampa som är ansluten till noden.

Tester som utförts är:

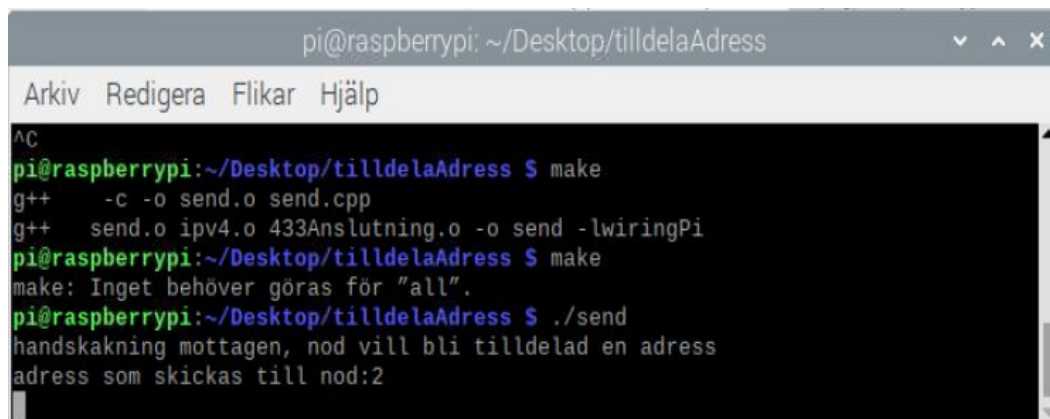
1. Tilldela adress till en nod.
2. Tända och släcka en lampa som är uppkopplad till en nod.
3. Tilldela adress till mer än en nod.

4.1 Test med server och nod, tilldelning av adress

Figur 10 är en skärmdump av servern som visar att den har en session med en nod och att noden vill bli tilldelad en adress som servern ger till den från en lista av lediga adresser.

Figur 11 är från en skärmdump av en nod som har gått igenom processen av att få en adress av servern.

Figur 12 visar ett flödesschema över testet.



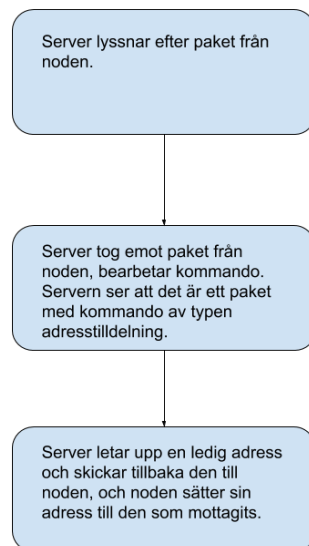
```
pi@raspberrypi: ~/Desktop/tilldelaAdress
Arkiv Redigera Flikar Hjälp
^C
pi@raspberrypi:~/Desktop/tilldelaAdress $ make
g++ -c -o send.o send.cpp
g++ send.o ipv4.o 433Anslutning.o -o send -lwiringPi
pi@raspberrypi:~/Desktop/tilldelaAdress $ make
make: Inget behöver göras för "all".
pi@raspberrypi:~/Desktop/tilldelaAdress $ ./send
handskakning mottagen, nod vill bli tilldelad en adress
adress som skickas till nod:2
```

Figur 10: Raspberry Pi, skärmdump



```
COM4
handskakning mottagen från server
17:51:03.190 -> adress är satt till:
17:51:03.224 -> 192.168.1.2.
```

Figur 11: Arduino Uno, skärmdump



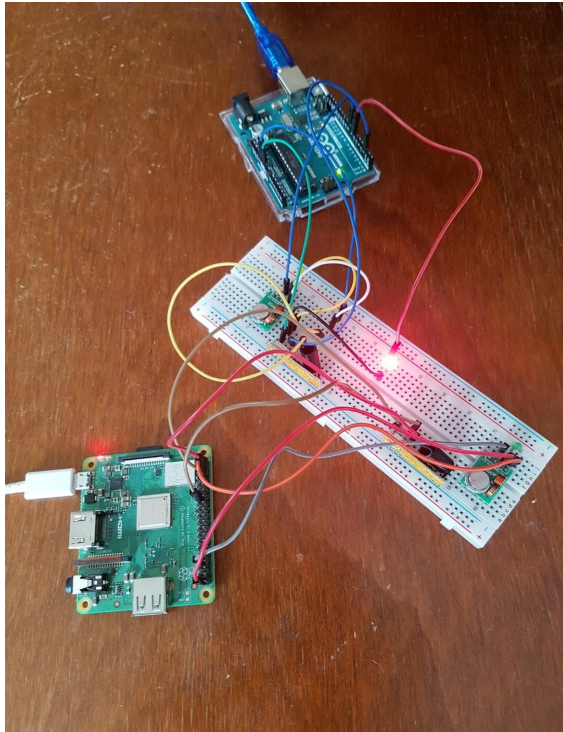
Figur 12: Flödesschema för test 1, tilldelning av adress

4.2 Test med server och nod, tända och släcka lampa

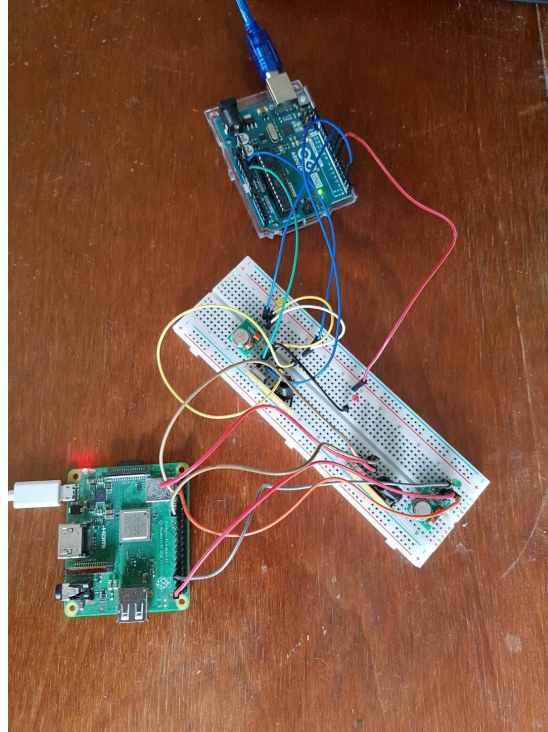
Figur 13 visar när servern har skickat en instruktion till en nod att tända en lampa.

Figur 14 visar när har servern skickat en instruktion till samma nod fast istället släcka lampan denna gång.

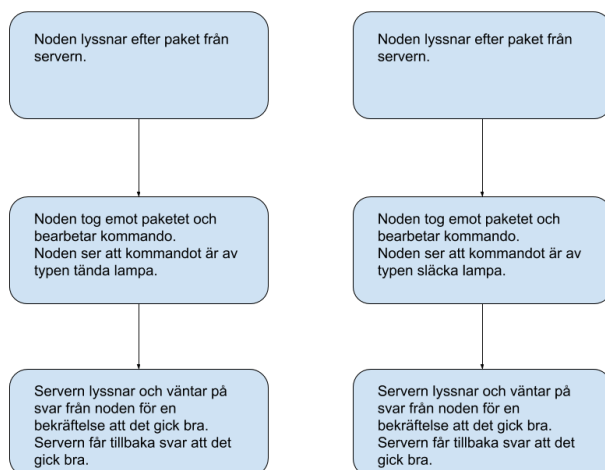
Figur 15 visar flödesschema.



Figur 13: Lampan tänds



Figur 14: Lampan släcks



Figur 15: Flödesschema för test 2, tända lampa till vänster, släcka till höger.

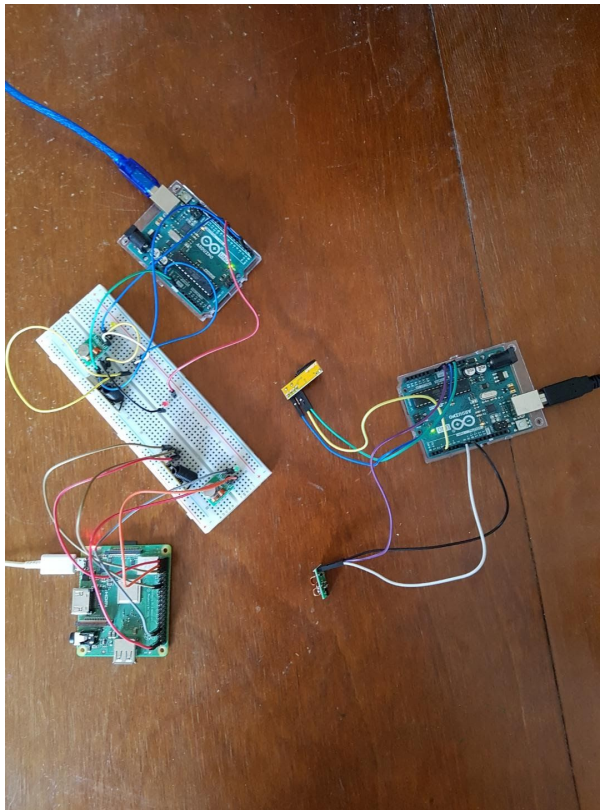
4.3 Test med server och flera noder

Vid detta test så är det totalt tre stycken enheter som kan kommunicera med varandra, det är en Raspberry pi som agerar som server och två stycken Arduino Uno enheter som agerar som noder.

Detta test var gjort för att se så att inte enheterna skickar data samtidigt och resulterar i en kollision vilket medför att ingen data kommer fram.

Testmiljön går att se vid figur 16, båda nodernas utskrift går att se vid figur 17 och 18, och serverns utskrift går att se vid figur 19.

Figur 20 visar flödesschema.



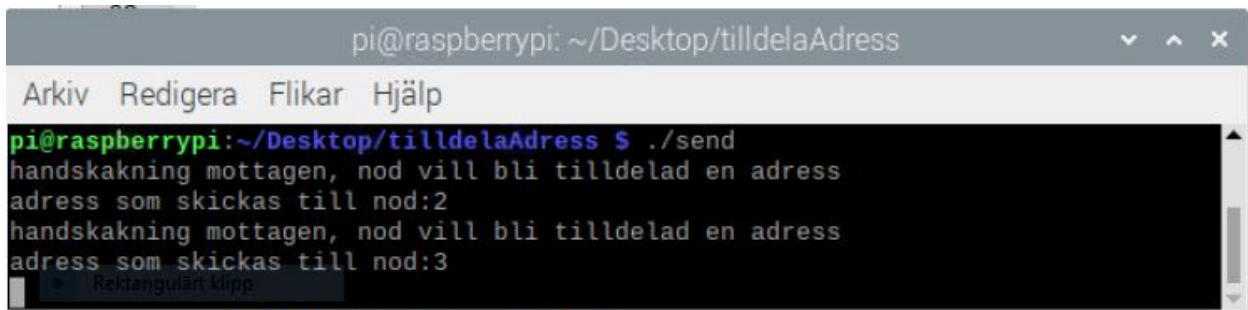
Figur 16: Två stycken noder och en Raspberry Pi



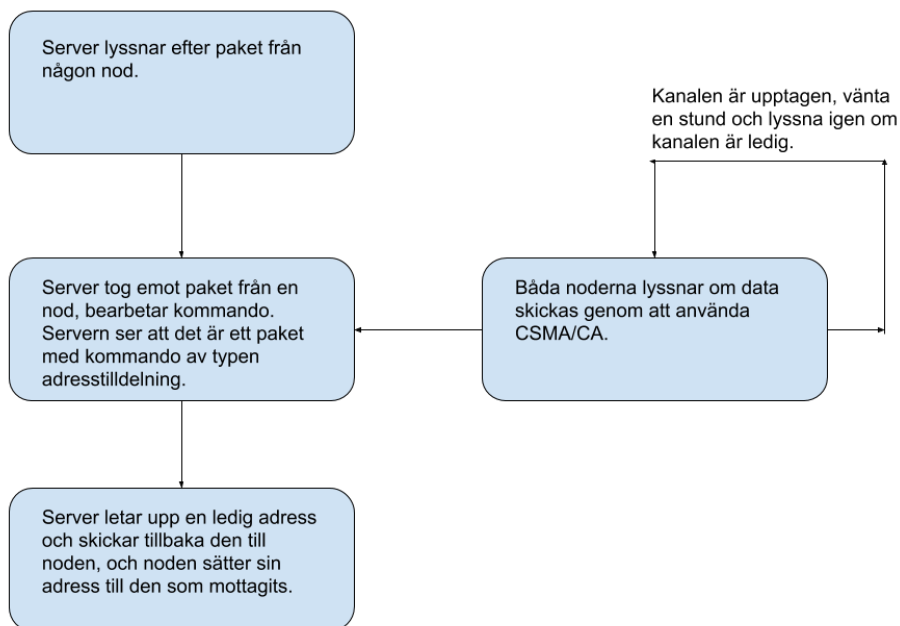
Figur 17: Skärmdump från en nod från comport 3



Figur 18: Skärmdump från en nod från comport 4



Figur 19: Skärmdump från Raspberry Pi

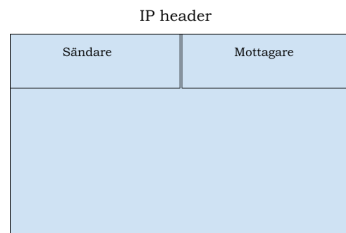


Figur 20: Flödesschema för test 3, tilldelning av adress med flera noder.

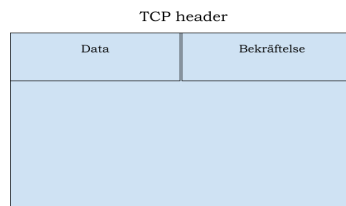
4.4 Slutresultat, illustration av färdig produkt

Figur 21 visar det färdiga IP liknande protokollet som skapades, figur 22 visar det färdiga TCP liknande protokollet som skapades.

Figur 23 visar en jämförelse mellan OSI-modellen och protokollstacken som används i systemet.

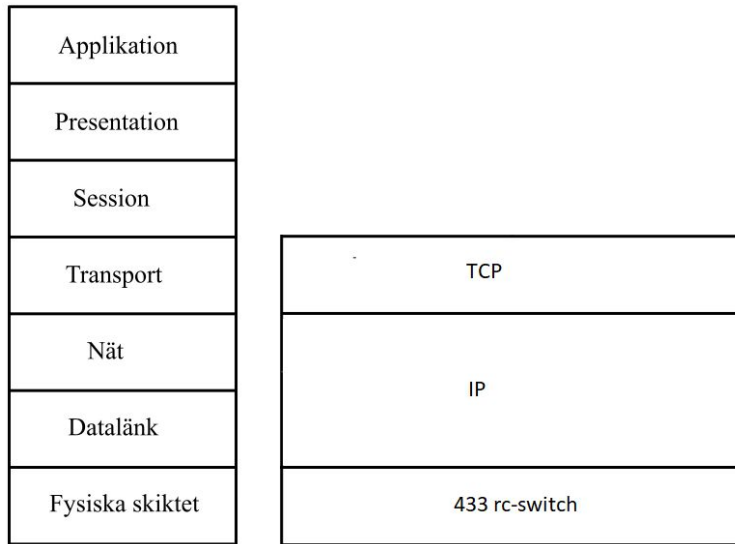


Figur 21: IP header



Figur 22: TCP header

OSI-modellen



Figur 23: OSI-modellen till vänster, projektets protokollstack till höger

5 Diskussion

I detta avsnitt täcks styrkor och brister i det protokoll som skapades, en avstämning mot målen görs också för att se om dem mål och krav uppfylls som ställs vid kapitel 1.1 samt 1.3.

5.1 Uppfyllda samt ej uppfyllda mål

I detta avsnitt så går dem krav och mål igenom som har uppfyllts och även dem krav och mål som ej har uppfyllts.

Om kraven jämförs med slutprodukten så ser det ut enligt följande:

- “Noden ska både kunna skicka och ta emot data”, detta är ett viktigt krav som ställdes, för att kunna utveckla ett system som använder respons så behöver noden kunna skicka tillbaka ett svar till servern, så detta krav har uppfyllts.
- “Servern ska inte behöva vara inom räckvidd för att nå en viss nod, datan ska kunna hoppa från nod till nod tills den når sin slutdestination”, detta krav har ej uppfyllts.

Om målet jämförs med slutprodukten så ser det ut enligt följande:

- “Utveckla ett trådlöst mesh-nätverk med noder som kan kommunicera med varandra med halv-duplex över 433 MHz”, i nuläget så kommunicerar noderna med halv-duplex över ett 433 MHz nätverk, det som inte är uppfyllt i nuläget är att noderna inte använder sig av mesh teknologi, så en nod kan ej vidarebefordra ett paket till en annan nod. Detta var på grund av tidsbrist, det tog mer tid än förväntat att få systemet att kunna skicka dem båda protokollen.

5.2 Styrkor i protokollet

Om det nuvarande fysiska lagret byts bort och istället ersätts med en annan teknik som exempelvis kabelanslutning till en router eller WiFi till en router så kommer protokollet fortfarande fungera genom att göra några små förändringar.

Protokollet använder sig av CSMA/CA, på detta sätt minimeras "Exposed node problem", beskrivet i 2.1.4.

5.3 Brister i protokollet

En brist i protokollet är att det inte blev ett protokoll som har möjligheten att användas som i ett mesh-nätverk, exempel, om vid kapitel 2.1.4 vid figur 2, nod A vill skicka data till nod D, så hade det fungerat om protokollet kunde vidarebefordra, om data kunde hoppa från nod till nod tills den når sin slutdestination.

En förbättring som skulle kunna gjorts om det funnits mer tid skulle vara att istället för systemet skickar datan i klartext i nuläget, så skulle en kodare användas när en enhet sänder, och en avkodare används när en enhet tar emot data så att det skulle försvåra att avlyssna trafik som skickas.

5.4 Tester som utförts

I detta avsnitt diskuteras dem resultat som publicerats i kapitel 4 i djupare detalj.

5.4.1 Tilldelning av adress med en nod och en server

Servern använder operativsystemet Raspberry Pi OS och programmet som har skrivits fungerar så att servern lyssnar efter paket från en nod, när ett paket har tagits emot på servern så tittar servern vad det är för typ av paket, och i detta fall så vill noden bli tilldelad en adress, och servern letar då upp en ledig adress från sin tabell av lediga adresser och skickar tillbaka den till noden, och när adressen har skickats till noden så termineras sessionen mellan nod och server och kanalen blir tillgänglig för att skicka data.

5.4.2 Test med server och nod, tända och släcka lampa

På servern körs operativsystemet Raspberry Pi OS och programmet som har skrivits fungerar så att den styr noden från genom att den kan skicka ett kommando att antingen tända eller släcka lampa på noden, och noden skickar tillbaka en bekräftelse om lampan har tänts eller släckts.

5.4.3 Test med server och flera noder

Likt som test vid kapitel 4.1 så körs operativsystemet Raspberry Pi OS på servern och programmet som har skrivits fungerar så att servern lyssnar hela tiden efter en signal från en nod, om servern tar emot en signal så bearbetar den datan och ser vad för typ av paket det är, om det är tilldelning av adress eller något annat kommando. Om det är en adress som noden önskar att bli tilldelad så gör servern samma sak som nämnt ovan i kapitel 5.4.2, att det tar en adress från tabellen av lediga adresser.

Detta test var gjort för att testa så att vi kunde se vad som skulle hända om flera enheter var uppkopplade samtidigt och se så att inte paket kollisioner uppstod så att ingen data kom fram överhuvudtaget. Vår implementation av CSMA/CA fungerade, en nod lyssnar om det är trafik som skickas över kanalen, och om det är trafik som skickas så vilar den en stund och lyssnar sedan på kanalen igen för att se om det är ledigt att börja skicka data.

5.5 Slutresultat av projektet

5.5.1 IP

Den slutliga produkten som tillverkades var två stycken protokoll, det första liknar då ett IP protokoll med sändarens adressfält och mottagarens adressfält. Tanken var att vi skulle ha ett till fält i IP protokollet som heter TTL(time to live), och det var menat att utnyttjas så att ifall ett paket blev vidarebefordrat av en nod till en annan nod som i ett mesh-nätverk så skulle inte paketet studsas i all oändlighet, ttl fältet hade bestått av ett heltal från exempelvis 1 - 255, och varje enhet som den hoppar förbi så skulle ttl fältet minskat med 1, och den hade fortsatt så tills ttl fältet blivit noll och paketet hade då förkastats, vilket hade gjort så att ett paket inte studsar för evigt och saboterar kommunikationen mellan enheterna, men eftersom vi inte lyckades utveckla ett mesh-nätverk så kändes det onödigt att ha ett ttl fält.

5.5.2 TCP

Det andra protokoll som tillverkades liknar ett TCP protokoll, och det tillåter systemet då att veta om servern har kontakt med en nod med hjälp av en handskakning och sedan skicka datan och slutligen terminera sessionen.

5.5.3 Protokollstack

Det som saknas hos stacken är ett protokoll som jobbar ett lager längre upp, så att istället för att styra noder och få information om noderna i en terminal, så hade det istället gå att göra ett program som ser ut att vara mer användarvänligt än en terminal på Raspberry Pi som det är i nuläget.

6 Slutsats

6.1 Resultat jämfört mot mål

Målet med projektet från kapitel 1.1 var: “Utveckla ett trådlöst mesh-nätverk med noder som kan kommunicera med varandra med halv-duplex över 433 MHz.”.

I det resulterande systemet så kan server kommunicera med noder med halv-duplex över 433 MHz, systemet är inte ett mesh-nätverk så en nod kan inte vidarebefordra data som var ämnat till någon annan nod än sig själv, det är en liten förändring som behöver göras på noderna så att när ett meddelande som inte var ämnat till den noden så istället vidarebefordrar den paketet till nästa nod som är inom räckvidd. Och då behövs det även läggas till ett ttl fält i IP protokollet som nämnt ovan i kapitel 5.5.1.

6.2 Fortsatt arbete

Det fortsatta arbetet skulle kunna vara att istället för att bara tända och släcka lampor som det används till i nuläget, så skulle noden kunna utföra något annat typ av arbete, t.ex ansluta en sensor på noden så att den skickar tillbaka data som temperatur, tryck, eller något annat, eller att ansluta en servomotor på noden som ska öppna gardinerna vid ett visst klockslag. Och att få systemet att fungera som ett mesh-nätverk genom att noderna vidarebefordrar paket.

6.3 Erfarenheter

Designa egna nätverksprotokoll har varit en väldigt lärorik upplevelse, och att få nätverksprotokollen att kommunicera från server till nod och nod till server över 433 MHz.

7 Referenser

- [1] “Reliable Low Latency Wireless Mesh Networks - From Myth to Reality”. *2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC)*.
- [2] Richard W. Middlestead, “Digital Communications with Emphasis on Data Modems: Theory, Analysis, Design, Simulation, Testing, and Applications”. *2017 IEEE*, sidnummer 227 - 250.
- [3] Amir K. Khandani, “Two-Way (True Full-duplex) Wireless”. *2013 IEEE*.
- [4] <https://zigbeealliance.org/solution/zigbee>
- [5] <https://www.zwavesverige.se/vad-ar-z-wave>
- [6] <https://cdn-shop.adafruit.com/datasheets/PT2272.pdf>
- [7] <https://cdn-shop.adafruit.com/datasheets/PT2262.pdf>
- [8] https://play.google.com/store/apps/details?id=com.realvnc.viewer.android&hl=en_US
- [9] <https://www.arduino.cc>
- [10] <https://github.com/sui77/rc-switch>
- [11] <https://wiringpi.com/>
- [12] <https://gcc.gnu.org/>
- [13] <https://www.raspberrypi.org/downloads/raspberry-pi-os/>

Joacim Wörn



Besöksadress: Kristian IV:s väg 3
Postadress: Box 823, 301 18 Halmstad
Telefon: 035-16 71 00
E-mail: registrator@hh.se
www.hh.se