BACHELOR'S THESIS 2021

Zigbee-brygga för Palcom - Kommunikation för IoT

Simon Berglund, Dag Isaksson

ISSN 1651-2197
LU-CS/HBG-EX: 2021-07

DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

Zigbee-brygga för Palcom

Kommunikation för IoT

LU-CS/HBG-EX 2021-07

LUNDS UNIVERSITET
Campus Helsingborg

LTH Ingenjörshögskolan vid Campus Helsingborg
Institutionen för datavetenskap

Examensarbete:
Simon Berglund
Dag Isaksson

LTH Ingenjörshögskolan vid Campus Helsingborg
Lunds universitet
Box 882
251 08 Helsingborg

LTH School of Engineering
Lund University
Box 882
SE-251 08 Helsingborg
Sweden

Tryckt i Sverige
Lunds universitet
Lund 2021

Sammanfattning

Att kunna kontrollera apparater trådlöst är något som blir en allt vanligare önskan hos användare. Möjligheten att sätta på och stänga av lampor från sina smartphones är något man kan vilja ha i sitt hem.

Palcom är en plattform som kan användas för att koppla ihop olika enheter så att de kan kommunicera med varandra, och Zigbee är ett vanligt protokoll för trådlösa lampor. I detta examensarbete vill vi skapa en brygga mellan Palcom och Zigbee, för att göra det möjligt att styra lampor byggda för Zigbee via Palcom.

Plattformen PalCom består av flera delar som används för att skapa kopplingar mellan olika trådlösa enheter och styra dessa via ett grafiskt gränssnitt. Centralt för Zigbee är att det är en specifikation för ett högnivåkommunikationsprotokoll. Den arbetar på 2,4 GHz-bandet vilket betyder att den kan användas mellan olika marknader då frekvensen inte är skyddad.

Examensarbetet gick ut på att koppla ihop Palcom och Zigbee, och använda kopplingen för att styra lampor. Resultatet blev en sammankoppling av flera olika program, och exempel som visar hur man kan använda denna kombinerade plattform för att styra lampor.

Nyckelord: IoT, Internet of Things, Palcom, Zigbee, Raspberry Pi, Kommunikation, Sammankoppling.

Abstract

Being able to wirelessly control devices is becoming an increasingly more common wish from consumers. The possibility to turn on and off lamps from their smartphones is something that a lot of people want.

PalCom is a platform that can be used in order to connect different devices so that they can communicate with each other, and Zigbee is a common protocol for wireless lamps. In this thesis we want to create a bridge Between Palcom and Zigbee, in order to make it possible to control lamps built for Zigbee through Palcom.

The platform PalCom consists of multiple parts used for creating connections between wireless devices, and controlling these through a graphical interface. Zigbee is a specification for a high-level communications protocol. It works on the 2,4 GHz band which means that it can be used between different markets as the frequency isn't protected.

The task for the degree project was to connect Palcom and Zigbee, and use the connection to control lamps. The result was an interconnection between multiple programs, and examples showing how this combined platform can be used to control lamps.

Keywords: IoT, Internet of Things, Palcom, Zigbee, Raspberry Pi, Communication, Interconnection

Förord

Vi vill tacka LTH:s institution för datavetenskap för möjligheten att göra vårt examensarbete där.

Stort tack till Alfred Åkesson som var handledare under arbetet, för all hjälp och vägledning. Tack även till Görel Hedin, som var examinator för examensarbetet.

Vi vill också tacka Mattias Nordahl och Björn Johnsson för deras hjälp, och för för lånet av en lampa av Mattias Nordahl.

Innehållsförteckning

1 Inledning

Detta examensarbete är den sista delen i utbildningen på högskoleingenjörsprogrammet i datateknik vid Lunds Tekniska högskola. Vi har valt att genomföra det på institutionen för datavetenskap.

1.1 Bakgrund

IoT(Internet of things) eller "sakernas internet" på svenska är en samlingsterm för elektroniska enheter som kan styras trådlöst i ett nätverk. Dess syfte är att möjliggöra kommunikation mellan enheter för att dela med sig av information till varandra och till oss människor. Det tillhör en väsentlig del av den digitala utvecklingen i samhället samt dess ekonomiska utveckling. Eftersom IoT sammankopplar den fysiska världen med den digitala, innebär detta ett stort steg för vår moderna samhällsutveckling. [1]

Det finns redan en uppsjö av smarta produkter på marknaden- lampor, sensorer, rullgardiner, kaffekokare, väckarklockor, osv. och det tillkommer ständigt nya i en aldrig sinande ström för att mätta vårt behov på nästa nya spännande produkt. Mobiltelefonen har kommit att ändra vårt sätt att kommunicera och har kommit att ha en vital funktion även inom IoT för att kommunicera med dess enheter.[2]

Med alla dessa enheter kommer rimligtvis en önskan om att kunna kombinera dem på nya och kreativa vis för att skräddarsy sina egna behov.

Ett bekymmer som uppstår med IoT är att få enheter som har olika protokoll att kommunicera med varandra och agera på ett visst kommando för att åstadkomma något värdefullt. Man kan till exempel vilja skicka data till en lampa, och få den att förstå vad datan betyder. Om man dock skickar datan från en annan typ av enhet, så kan man behöva översätta informationen så att mottagaren kan läsa av innehållet.

Detta kan göras med en så kallad "brygga". En brygga är en enhet som kopplar ihop 2 nätverk så att de kan fungera som 1 nätverk. Bryggan kan inspektera data från ena nätverket och sedan skicka vidare det till andra nätverket. Den kan då översätta informationen så att kommunikationen fungerar som den ska.[3]

Zigbee är ett trådlöst protokoll som möjliggör kommunikation mellan enheter. Zigbee används av flera miljoner olika produkter, och är väletablerat bland flera olika tillverkare.[4] Palcom är en arkitektur[5] som är skapad för att koppla samman enheter och tjänster. Vårt arbete har gått ut på att skapa en brygga till Zigbee-protokollet för att användas tillsammans med Palcom. På så vis möjliggör vi kommunikation mellan enheter som använder sig av olika protokoll. Bryggan ska ligga på en Raspberry Pi, där översättning mellan protokollen ska utföras.

För att skicka signaler till enheter i Zigbee-nätverket används REST-API som i sin helhet står för Representational State Transfer Application Programming Interface.

API (Application Programming Interface) är en mängd protokoll som används för att interagera med ett system. API:et gör det möjligt att kommunicera med systemet så att man kan använda sig av systemets funktioner. REST är en typ av API som kan skicka en representation av en enhets tillstånd. I vårt fall skickar den information om lampans tillstånd.[6] Ett REST-API använder sig av en nyckel för att identifiera användaren av API:et.[7]

Vår lösning går ut på att använda oss av en RaspBee2, vilket är en Zigbee-modul skapat av Dresden Elektronik. Denna modul installeras på en Raspberry Pi och har förmågan att skicka Zigbee-signaler och gör det därför möjligt att förvandla en Raspberry Pi till en Zigbee gateway. Detta fungerar genom att skicka REST-API-förfrågningar till denna modul som i sin tur skickar Zigbee-signaler till våra Zigbee-enheter. På så vis kan vi implementera REST-API-förfrågningar från Palcom som därefter kan kommunicera med Zigbee-enheter. Ett REST-API är oftast implementerat med HTTP-protokollet, men det är inte strängt tvunget. I Dresdens Elektroniks fall är det emellertid HTTP som används och således vad vi använt.

1.2 Mål

Målet är att skapa en lösning där PalCom och Zigbee har möjlighet att kommunicera med varandra och använda denna för att kontrollera lampor.

Vår lösning kommer att vara en brygga och denna ska bestå av en Raspberry Pi som ska kunna ta emot PalCom-signaler och skicka vidare dessa som Zigbee-signaler till Zigbee-enheter i vårt Zigbee-nätverk. Därefter ska vi skapa en applikation som använder denna brygga från en smartphone till en Zigbee-enhet. Med hjälp av denna applikation kan vi då demonstrera bryggan och testa hur den fungerar.

1.3 Problemformulering

Problemformuleringen är en mängd frågor, som vi har försökt svara på i detta examensarbetet. Ni kan hitta svar på frågorna i kapitel 9.1 (Återkoppling på frågor)

Hur kan man använda en Raspberry Pi för att styra Zigbee-enheter?

Kan man skapa en generell lösning för kommunikation med andra Zigbee-enheter än lampor? Såtillvida att varje typ av enhet går att ansluta och användas i systemet utan att först behöva implementera särskilt stöd för just den typen?

Hur tar man emot data från Palcom i en Raspberry Pi?

Vilka är skillnaderna mellan Palcom och Zigbee?

Hur kan man lättast översätta mellan signalerna inom Palcom och Zigbee?

1.4 Motivering

Eftersom allt mer hemelektronik börjar styras trådlöst, kommer man antagligen att vilja använda smartphones i större utsträckning för styrningen. Då kommer det att krävas sätt att kommunicera mella olika sorters protokoll.

En brygga skulle göra det möjligt att bygga applikationer i Palcom som kan kommunicera med Zigbee. Då kan man använda Palcoms program för uppbyggnad av användargränssnitt och sedan koppla en app med det gränssnittet till Zigbee. Detta är användbart eftersom att man inte kan bygga användargränssnitt direkt i Zigbee, men då kan använda Palcoms program istället.

Vi har valt att göra detta som vårt exjobb eftersom vi tycker hemautomation är ett intressant område som vi gärna vill lära oss mer om. Att skapa skräddarsydda lösningar för ens egna behov finner vi både engagerande och utmanande och hemautomation verkar ha framtiden med sig.

1.5 Angreppssätt och metoder

För att uppnå målen har vi använt oss av litteratursökning och programmering. Vi kommer att söka upp litteratur och använda informationen som vi hittar för att öka vår kunskap om Palcom och Zigbee. Därefter kan vi börja skriva kod till Palcom-applikationer, och försöka integrera dessa med Zigbee-kommandon.

När integrationen fungerar ska vi försöka skapa en brygga mellan Palcom och Zigbee, som vi sedan testar med hjälp av en applikation. På så sätt testar vi vilka möjligheter som finns, genom kodning.

1.6 Avgränsningar

- Bryggan är endast menad att fungera mellan Palcom och Zigbee, d.v.s. inga andra protokoll.
- Applikationen är ämnad att fungera för en specifik enhet (inte generellt för alla typer av enheter), i detta fall en lampa.
- Vi kommer inte att lägga fokus på designen av applikationen, utan kommer att fokusera på funktionen.

- Vi kommer inte att ta någon särskild hänsyn till applikationens säkerhet och integritet mot olika slags intrång(i form av hacking).

2 Bakgrund

Detta kapitel kommer att beskriva de de hård- och mjukvarusystem som vi har använt oss av under examensarbetet.

2.1 Palcom

Palcom är en arkitektur skapad för att koppla samman enheter och tjänster. Problemet med de olika protokoll som används för hemautomation idag är att de inte är kompatibla med varandra. Utan en mellanhand som kan översätta mellan dem går det inte att föra någon kommunikation mellan dem. Det är här Palcom kommer in i bilden.[5]

Palcom är som ett klister som binder ihop olika typer av enheter så att de kan kommunicera med varandra. Enheter som från början inte är designade för att passa ihop kan nu kombineras på nya, oplanerade sätt kallat "ad-hoc composition". Med ett enkelt skriptspråk är det nu möjligt att skapa nya sätt att använda enheterna på.

Palcom utvecklades ursprungligen som en del av ett EU IST-projekt kallat Palpable Computing, men har vidareutvecklats i EASE och andra ViNNOVA- och SSF-projekt. Den används för närvarande i sjukvårdssystem, i kombination med dlna-utrustning för att möjliggöra flexibel kombination mellan fjärranslutna och mobila enheter.[5]

2.1.1 Arkitektur

De centrala delar com utgör Palcom-arkitekturen är: enheter(devices), tjänster(services), anslutningar(connections) och sammansättningar(assemblies). Vidare används de engelska orden som de förekommer i mjukvaran.

Dessa delar har olika uppgifter i kommunikationen mellan enheter. Man kan till exempel ha en service med instruktioner för att skicka kommandon till Zigbee-enheter (som i vårt arbete). Denna kan sedan köras från olika enheter, och på så sätt brygga enheterna med Zigbee.

2.1.2 Services

Palcom använder sig av olika sorters services för att skapa funktioner. En service kan utföra beräkningar eller förse enheter och mjukvarusystem med ett interface. En service består av tre delar, identitet (namn, typ), beskrivning (vad servicen kan göra), och aktiviteter (mottagande och utsändande av kommandon).[5]

2.1.3 Assemblies

Assemblies är vad som möjliggör kommunikationen enheter emellan. En assembly skapas, i vilken regler sätts upp för hur och vad de ska åstadkomma tillsammans. Ett exempel skulle kunna vara att en smart-kaffekokare startas när smart-väckarklockan ringer.

2.1.4 theThing

TheThing är en applikation som används för att exekvera Palcom services, Palcom assemblies, och för att manövrera Palcom tunnlar och dess routing.[5]

2.1.5 Palcom GUI

Vägen för att skapa ett GUI för Palcom services är uppdelad i flera delar. Programmet som man använder för det grafiska i Palcom heter PML-editor. Detta programmet kan skapa filer med PUIML-kod, och fungerar också som interpreter för de filerna. PUIML står för "Palcom User Interface Markup Language" och är ett XML-baserat språk som används för att beskriva applikationerna. Interpretern är ett program som används för att skapa gränssnitt från en fil med PUIML-kod.[8]

Dessa delar gör att man kan skapa flera olika sorters services, utan större programmerings -kunskaper.

När man bygger en applikation i PML-editor skapas en fil som innehåller information om appens utseende och dess funktioner. Med hjälp av en android-applikation kan man ladda in filen i en smartphone, som kan köra filen.

I bild 2.1 visas en bild av PML-editor, med en enkel applikation:

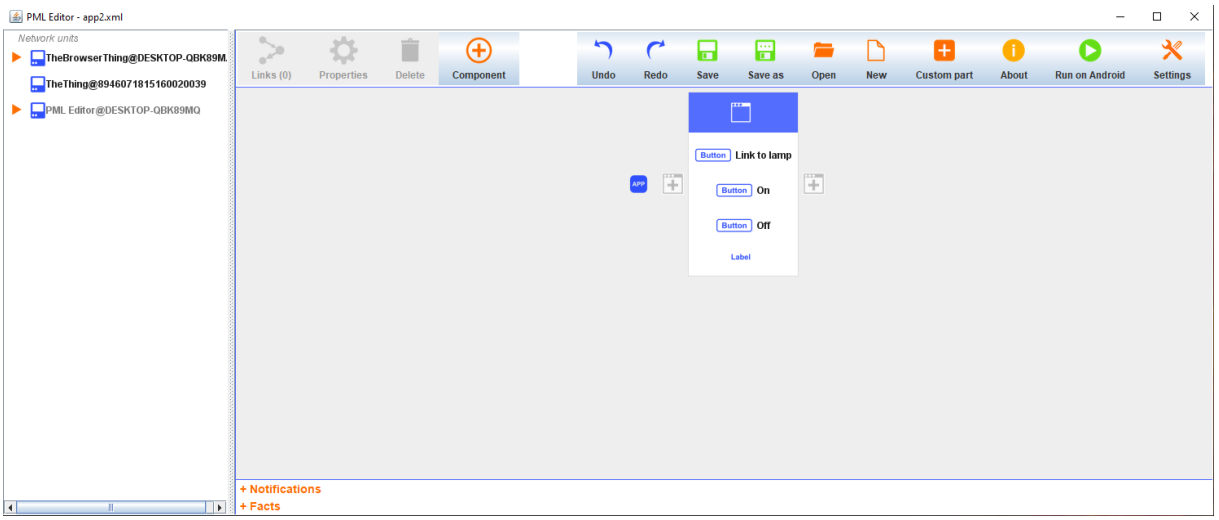

Bild 2.1: PML-editor med enkel applikation

2.1.5.1 Custom part

I PML-Editor finns flera olika komponenter som man enkelt kan skapa i programmet. Om man behöver en annan sorts komponent så måste man dock skapa en custom part. En custom part är en komponent som man själv programmerar, och därför enklare kan välja funktionalitet för. Detta är användbart ifall man ska göra mer komplicerade program.

Kodningen i custom parts skrivs i generell android-kod, vilket gör att det finns mycket dokumentation och information om hur man skapar viss funktionalitet.

2.2 Zigbee

2.2.1 Zigbee och IoT

Zigbee är ett protokoll för att styra smarta enheter i ett nätverk. Ett exempel på en smart enhet är en sådan som är utrustad med en radiosändare samt mottagare. I dagens värld kan detta vara en lampa(t.ex Philips hue), en brödrost, en kaffekokare eller varför inte en kamera, alla är de utrustade med detta protokoll för att kunna kommunicera med varandra inom samma nätverk. Det betyder att de förutom att vara en produkt med funktionalitet av något slag, så har de även förmågan att kommunicera med andra enheter inom ett Zigbee-nätverk.[9]

Zigbee påminner om andra protokoll för dataöverföring som Wi-Fi och Bluetooth. Det är dock utformat för andra ändamål och den stora skillnaden för en konsument är typen av enheter som den stöder och erbjuder. Zigbee erbjuder konsumenter mer i form av lampor och smarta uttag samt sensorer då andra protokoll främst stöder andra typer av smarta enheter. Det gör Zigbee-protokollet väldigt attraktivt för hemautomatisering vilket medför att Zigbee är den vanligaste standarden för IoT.

Zigbee Alliance är en förening av fler än 600 olika teknikföretag som inkluderar Siemens, Schneider Electric, Philips, Osram, Bosch, Busch-Jaeger, Samsung, Sony m.fl. Dessa har skapat samt underhåller Zigbee-protokollet som i skrivande stund är Zigbee 3.0, men benämns kort och gott Zigbee då det är helt bakåtkompatibelt med äldre enheter som är utvecklade för tidigare versioner av protokollet.[10]

Zigbee är den populäraste standarden för trådlösa nätverk när man önskar koppla ihop sensorer med kontrollsystem. Det är ett öppet, globalt, paketbaserat protokoll som är utformat att erbjuda en användarvänlig arkitektur för säkra, pålitliga samt strömsnåla trådlösa nätverk. Det är ett standardiserat trådlöst protokoll som tillåter enheter från olika tillverkare att kommunicera med varandra.[11]

Zigbee implementerar tvåvägskommunikation. Det betyder att en controller får kvittens på att datan som den skickat har kommit fram och medför att enheter kan skicka tillbaka information till controllern, till exempel sensorsignal och temperatur.
Till skillnad från andra protokoll finns det inget tak på antalet hopp som ett Zigbee-system tillåts göra.

2.2.2 Enhetstyper

En Zigbee-enhet kan vara av 3 olika typer: Coordinator, router eller end device. [Bild 2.2]
Coordinator: Är själva controllern och det finns bara en av denna i varje Zigbee-nätverk. De agerar gateway till externa system, sköter säkerheten inom Zigbee-nätverket och kopplar ihop routers med end devices inom Zigbee-nätverket.

Routers ansvarar för att routa paket och kommer med funktionalitet av varierande sort. Enheter som lampor, fläktar, rullgardiner är alltså också en router. De är ej batteridrivna utan kopplas in på elnätet.
End devices är batteridrivna enheter, såsom portabla strömbrytare och sensorer. De är designade att vara väldigt strömsnåla.[12]

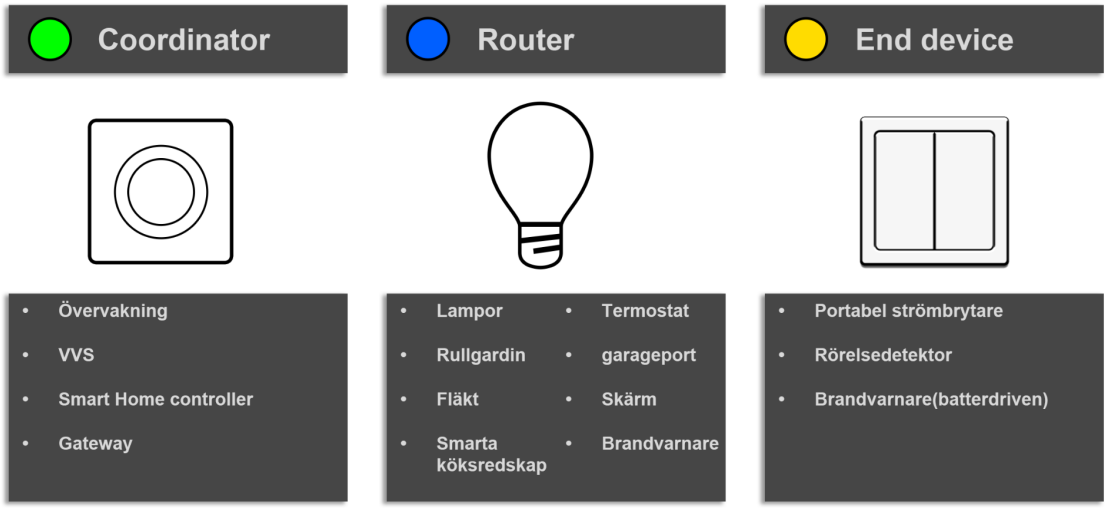

Bild 2.2. Översikt över ett axplock av Zigbee-produkter och av vilken enhetstyp de är.

Zigbee-nätverk med enheterna förklarade i bild 2.2 kan konfigureras enligt bild 2.3.

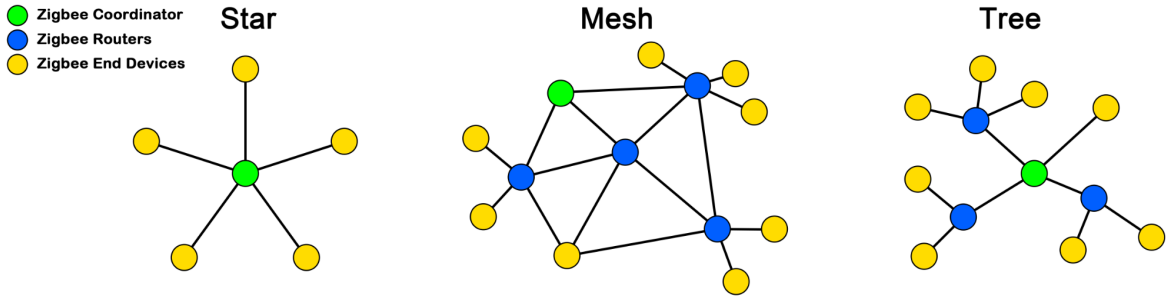

Bild 2.3. Exempel på olika sorters konfigurationer som stöds av Zigbee.

2.2.3 Profiler

Varje enhet i ett Zigbee-nätverk har en särskild applikationsprofil för sin typ. Applikationsprofilerna kan delas upp i publika, de som specificerats av Zigbee Alliance samt de som skapats av privata aktörer.
HA(home automation) är ett exempel på en publik profil som används för Zigbee-enheter avsedda för användning i hemmet. Detta är lampor och strömbrytare, vägguttag, fjärrkontroller, termostater, luftkonditioneringsapparater etc. Syftet med dessa profiler är att de ska möjliggöra kompabilitet mellan enheter av olika tillverkare.

Dessa profiler är i sin tur skapade med hjälp av Zigbee Cluster Library. I grund och botten så är det en mängd av kommandon som är indelade i områden efter funktionalitet och som kombineras med varandra för att skapa applikationsprofiler.

Bild 2.4 illustrerar hur olika kluster kan sättas samman till en profil. Färgerna som används har dock inget med att göra med de som användes i föregående stycke om enhetstyperna.

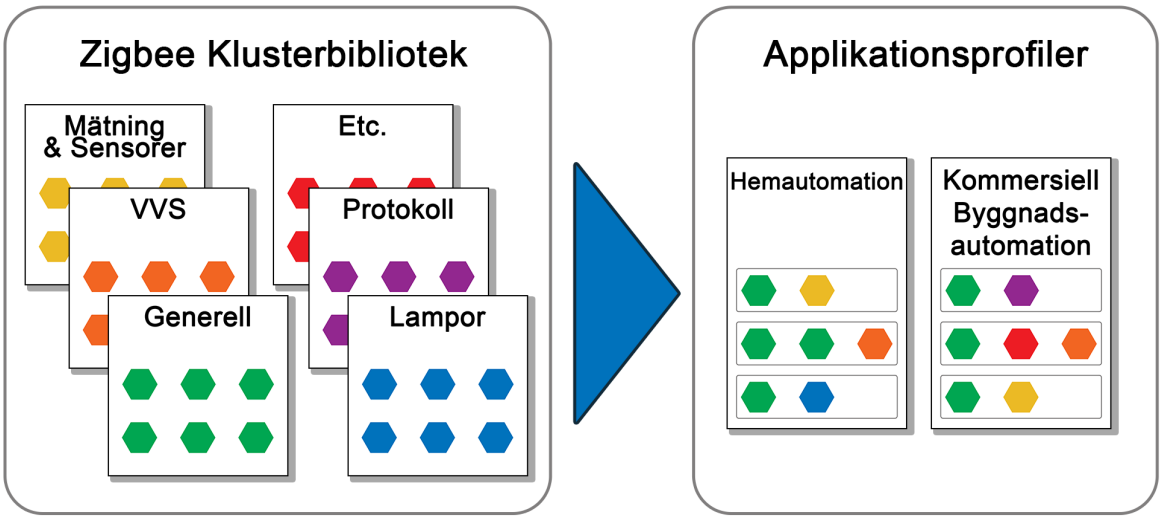

Bild 2.4. Illustration över hur kluster av olika typer konfigureras till applikationsprofiler.

2.2.4 Räckvidd

Enheterna kommunicerar med varandra med hjälp av en radio-transceiver. Chippet arbetar på IEEE 802.15.4 protokollet över 2.4Ghz. Dvs samma frekvensband som både WiFi och Bluetooth använder, men till skillnad från WiFi är Zigbee:s räckvidd kortare och inom intervallet 10-20 meter. Detta är ett designval för att göra Zigbee strömsnålare och på så vis förlänga batteritiden hos enheterna.[12]

Eftersom Zigbee-enheter har förmågan att skicka och ta emot data från andra Zigbee-enheter, medför detta att det går att bygga upp ett nätverk enligt en daisy-chain-konfiguration. som då kan vara mycket större än 20 meter i radie. Se avsnitt 2.3.2, bild 2.3, Mesh-konfiguration.

2.2.5 Säkerhet

Säkerheten är lägre i ett trådlöst nätverk gentemot ett trådbundet då signaler färdas genom luften och lätt kan lyssnas av, av en utomstående. I ett trådbundet nätverk är det svårare då man måste ha fysisk åtkomst till kablar i vilka signalerna går. Oavsett vilket, behövs det en säkerhetsplan. I Zigbee fungerar detta enligt följande scenario: Controllern delar ut en nyckel för kommunikationen mellan inkluderade noder. En Zigbee-enhet som inte är parad med controllern har alltså inte möjlighet att kommunicera med övriga enheter i systemet. 128 bitars AES-kryptering används, som är en vanlig standard inom digital kryptoteknik.

2.2.6 Kompatibilitet

Det är inte alla typer av smarta enheter som stöder Zigbee. Zigbee dominerar inom smarta lampor, men det är högst sannolikt att du kommer behöva stöd för annat protokoll om du skulle vilja styra TV, fläktar eller andra typer av enheter där tillverkaren ej använt Zigbee-protokollet.

Med dess låga strömförbrukning lämpar det sig bättre för vissa ändamål än andra. Den sänder bara data på begäran. Detta innebär att det är optimalt för knappar och sensorer, men mindre lämpligt för enheter som kräver högre bandbredd. T.ex trådlösa högtalare.

2.2.7 Systemuppbyggnad

IEEE 802.15.4-specifikationen är ett paketbaserat radio-protokoll och är vad Zigbee-protokollet bygger vidare på. Det ingår i gruppen av nätverkstyper LR-WPAN som står för low-rate wireless personal area network. Ett grundkonceptet inom WPAN är ad-hoc-nätverk. Det innebär att enheter tillåts att temporärt vara en del av nätverket genom att ansluta och koppla ifrån när som helst och lämpar sig för mobila enheter som laptops, tablets och mobiltelefoner.

Ett Zigbee-system kan teoretiskt sett ha maximalt 65 000 noder. I praktiken kan det antagligen inte nås i verkligheten, men det finns fungerande Zigbee-system som innehåller tusentals noder. Det har förmågan att självläka(eng. self heal) genom så kallad “mesh networking”. Tack vare att nätanslutna (ej batteridrivna) Zigbee-noder kan skicka vidare signalen till nästa, i ett mesh-nät, är dock räckvidden inte ett problem i större system. Till skillnad från Z-wave har inte Zigbee-tekniken någon begränsning hur många hopp signalen kan ta. Det gör att det går att bygga stora Zigbee-nätverk med god täckning och tillförlitlighet.

2.3 Raspberry Pi

En Raspberry Pi är en enkortsdator skapad av Raspberry Pi Foundation, med målet att utbilda i datoranvändning.[13] Raspberry Pi:n är som en vanlig dator som kör Linux, men tillhandahåller också GPIO-kontakter (general purpose input/output) som gör det möjligt för användaren att kontrollera olika sorters elektroniska komponenter.[14]

En sådan komponent är RaspBee2, som gör det möjligt för en Raspberry Pi att agera som en Zigbee-gateway.[15] RaspBee2 utvecklades av Dresden Electronics och har som mål att koppla ihop Zigbee-enheter från olika tillverkare.[16] På så sätt kopplas Raspberry Pi:n in till Zigbee-nätverket där den kan skicka och ta emot signaler från andra enheter.

I vårt arbete använder vi en Raspberry Pi för att köra programmen som krävs för att möjliggöra kommunikation mellan Palcom och Zigbee. Raspberry Pi:n kan då skicka kommandon till Zigbee-nätverket, som den tar emot från Palcom.

2.3.1 deCONZ

deCONZ är ett program som möjliggör kommunikation mellan Zigbee-enheter och gateways. Programmet visar en grafisk bild av Zigbee-nätverket, och ger en överblick över alla inkopplade enheter(se bild 4.2). deCONZ tillhandahåller även ett REST-API för system, vilket gör det möjligt att kontrollera enheter i systemet. Detta REST-API:et kan på ett enkelt

sätt användas från ett Java-program, varifrån man då kan skicka kommandon till sina enheter.[17]

2.4 Hårdvara

För att utföra arbetet användes några olika komponenter. Dessa inkluderar Raspberry pi, gateways, och lampor. Vi hade minst två av varje del, så att vi kunde använda varsin. Nedan följer en lista över de delarna som vi använt oss av.

Hårdvaran som användes under examensarbetet är:

- 2x OKDO Raspberry Pi 4 4GB Starter Kit
- 2x IKEA Trådfri LED E27 806lm (dessa lyckades vi inte få att fungera)
- 3x Philips Hue White ambiance E27
- 2x Dresden Elektronik RaspBee2 Zigbee Gateway

Det behövdes även vissa egna komponenter:

- Android-telefon med Palcom-applikation
- Egna datorkomponenter (Skärm, Mus, Tangentbord, o.s.v.)

3 Metod

Man kan dela upp arbetet under examensarbetet i olika delar: Palcom, Zigbee.
Dessa delar är inte i helt kronologisk ordning, eftersom att vi arbetade med de olika delarna samtidigt.

3.1 Litteratursökning

Under examensarbetet utförde vi en stor del litteratursökning. Detta gjordes för att lära oss om hur Zigbee och Palcom fungerar.

Vi läste även en del relaterat arbete, och jämförde dessa arbeten med vårt eget arbete. Dessa jämförelser kan läsas om i Kapitel 8.

3.2 Zigbee

När vi började använda Zigbee så började vi med att försöka nå lamporna från en REST-klient i Google Chrome, som heter Advanced REST Client och därefter började vi planera en implementation av detta i ett Java-program. Det visade sig att våra första lampor (IKEA Trådfri LED E27) ej var kompatibla med Raspbee2, så därför behövde vi beställa en annan sorts lampor. Dessa lampor (Philips Hue White ambiance E27) fungerade direkt, vilket gjorde att arbetet kunde fortsätta. När vi fick lamporna att fungera så skapade vi ett Java-program som vi kunde skicka REST-kommandon från.

3.3 Palcom

Samtidigt som vi arbetade med Zigbee, så började vi testa hur programmen inom Palcom fungerade, så att vi skulle kunna skapa program med hjälp av dessa. Eftersom Palcom-services var en stor del av arbetetet så började vi tidigt utföra tester med hjälp av enkla services. När vi hade en fungerande test-service så började vi arbeta på en mer användbar service, som skulle kunna användas i den faktiska Android-applikationen. När servicen var färdig började vi testa att koppla den till PML-editor, så att den skulle kunna köras från en Android-telefon med ett GUI.

När Zigbee-delen var färdig så kunde vi lägga in deCONZ-kommandona i vår Palcom-service, och på så sätt hade vi skapat en enkel applikation för att kontrollera lampor. Denna Android-applikationen utvecklades sedan så att den kunde spara data till lagringsutrymmet, och så att knappar kunde skapas dynamiskt beroende på hur många lampor som fanns i systemet.

3.4 Esoteric API

Vi implementerade även en ny service genom att använda oss av ett bibliotek som heter deCONZ REST Java API för att på ett smidigare sätt kunna exekvera REST-kommandon från vår Palcom-Service. Detta gick ut på att vi klonade biblioteket och länkade det till vårt Java Palcom-projekt utifrån vilket vi nu kunde skapa objekt i Java med hjälp av detta externa bibliotek för att på ett smidigare sätt exekvera REST-kommandon och snabbare skapa services.

3.5 Källkritik

[1], [2], [3], [6], [7] är källor av mer allmängiltig natur som behandlar IoT, REST API samt vad en brygga i detta sammanhang innebär. Några av sidorna marknadsför produkter vilket riskerar att informationen blir ensidig. Därför granskades informationen med större omsorg för att undvika detta.

[4], [13], [14], [15], [16] och [17] är webbsidor som är uppbyggda av företagen som referenserna handlar om. De kan därför ses som trovärdiga, eftersom att de förväntas ha bra kännedom om sig själva.

[5] och [8] är artiklar om Palcom. Dessa är skrivna av personer som har varit med och byggt upp Palcom, och är därför trovärdiga källor.

[9] och [10] är webbsidor som används för information om Zigbee. Dessa sidor används för försäljning av produkter som använder sig av Zigbee, och skulle därför vinna på att få Zigbee att verka så bra som möjligt.

[11] En bok om Zigbee-protokollet där medlemmar av Zigbee Alliance agerat konsulter. Anses därför vara tillförlitlig.

[12] Texas Instruments har arbetat mycket med Zigbee och har produkter med Zigbee på marknaden.

[18], [19], [20], [21], och [22] är referenser till artiklar som användes som relaterat arbete. Dessa artiklar hade inte så stor inverkan på utförandet av vårt arbete, men kan ses som trovärdiga källor i sina områden. Detta eftersom att de är skrivna av personer som är utbildade inom sina områden, och förväntas kunna mycket inom dessa.

4 Bryggans Arkitektur

Under examensarbetet har vi konstruerat en brygga, som har i uppgift att koppla ihop Palcom och Zigbee. Bryggan består av två olika delar, som ligger på en Raspberry Pi. Dessa delar är theThing och deCONZ. theThing och deCONZ är inte något vi har skapat själva, utan delar som vi använder för sammankopplingen. Det vi har gjort är att koppla ihop dessa delar, med hjälp av en Palcom service som vi har gett namnet "Zigbee Bridge". Bryggans delar används för att koppla ihop mobilapplikationen med lampa, enligt bild 4.1. För mer information om hur sammankopplingen fungerar, se kapitel 4.3.

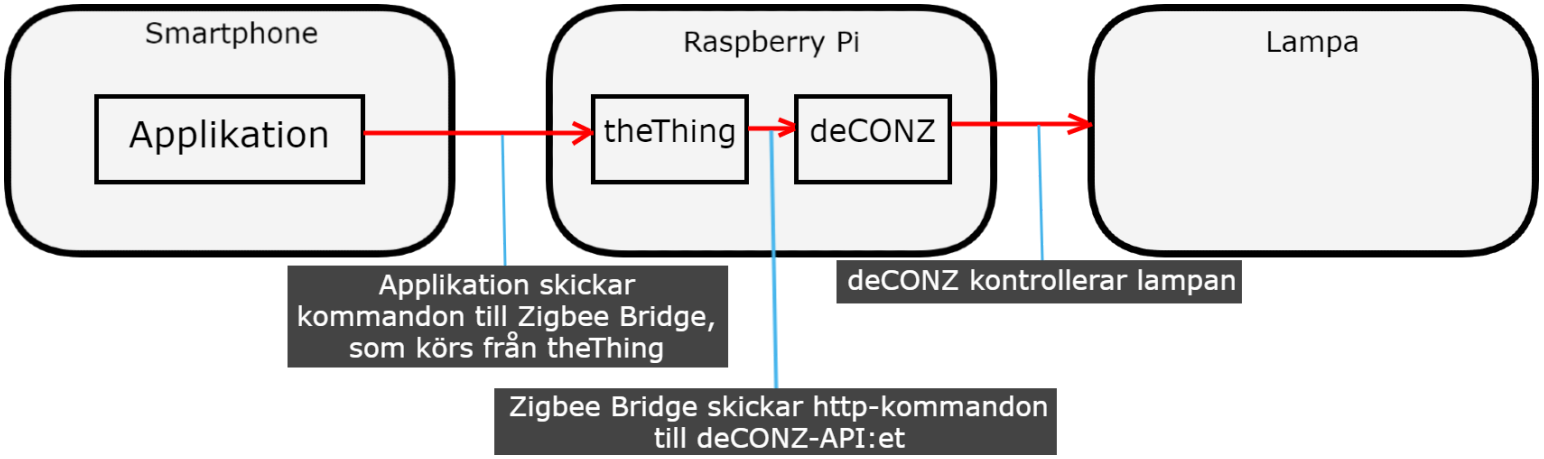

Bild 4.1: Sammankoppling av smartphone och lampa med hjälp av bryggan

4.1 theThing

Vi använder theThing för att köra en service, som vi har kallat för "Zigbee bridge", och denna innehåller flera olika funktioner. Funktionerna som är tillagda i servicen kan ses i del 5.2. Servicen kan sedan köras från en applikation via PUIDI. En palcom-service är koden som körs av theThing, och innehåller flera olika metoder som kan skicka signaler till lamporna. Detta görs genom att man skickar ett kommando till Zigbee-enheterna via REST-API:et. Se kapitel 2.1.4 för mer information om theThing.

4.2 deCONZ

Detta programmet kopplar samman Raspberry Pi:n till Zigbee-nätverket vilket gör det möjligt för Raspberry Pi:n att skicka kommandon till våra lampor. Dessa kommandon körs från vår Palcom service (5.2). Ett deCONZ-kommando består av en http-förbindelse till en enhet, som man kan använda för att skicka requests. Om vi till exempel skickar ett PUT-request med texten "on: true" till en lampa, så tänds lampan. Ni kan hitta mer information om deCONZ i kapitel 2.3.1.

deCONZ kan också användas för att grafiskt visa hur enheterna är kopplade till varandra. Bild 4.2 visar en bild av vårt system i deCONZ där vår Raspberry Pi(Configuration tool 1) har en anslutning(grön linje) till vår lampa(Philips Hue 1). Lampan "Color temperature light

3” finns också med i systemet, men är inte kopplad till RaspBerry Pi:n. Detta visas av att där saknas en linje till denna lampan:

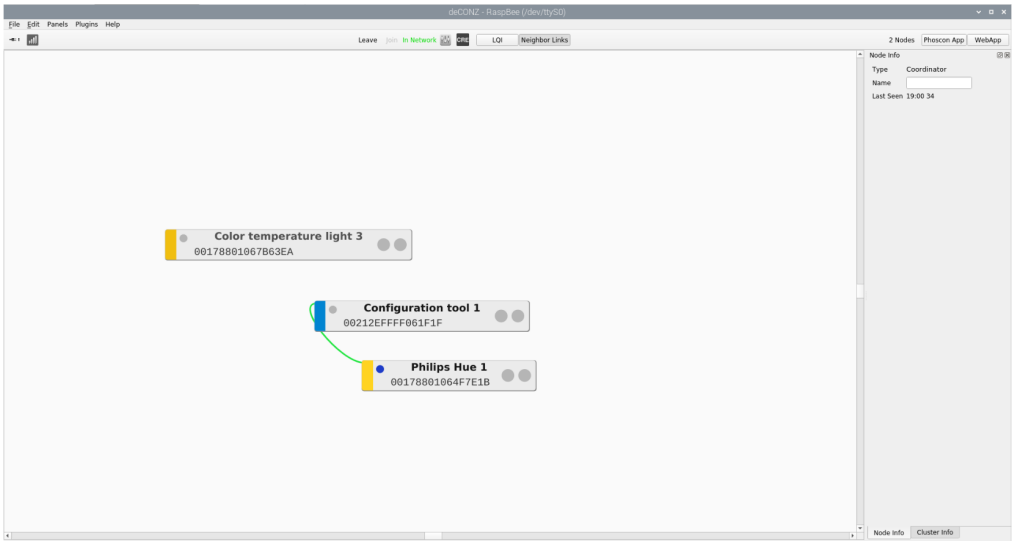

Bild 4.2: Grafisk bild i deCONZ

4.3 Sammankoppling

Bryggan kopplar en applikation till Zigbee-nätverket, som i vårt fall består av en lampa. Det görs genom att applikationen kör en av servicens funktioner när användaren trycker på en knapp. Detta kan enkelt läggas till från PML-editor. Servicen körs från theThing, som är igång på Raspberry Pi:n. Från servicen kan man sedan skicka kommandon via REST-API:et. Dessa kommandon kommer att gå via deCONZ, som då kan kontrollera lampan.

5 Exempel på applikation av bryggan

Under examensarbetet byggde vi en applikation som använder sig av alla tidigare nämnda delar. Detta för att kunna se ifall Palcom och Zigbee är korrekt sammankopplade. Syftet med applikationen är att styra en lampa från en Android-telefon. För att åstadkomma detta använder sig applikationen av både Palcom och Zigbee. Man kan dela upp applikationen i tre huvuddelar som arbetar tillsammans för att få den att fungera. Dessa delar är Applikationen, Raspberry Pi:n, och en Palcom service. Raspberry Pi:n används som brygga, och är beskriven i del 4.

5.1 Applikation

Med applikation menar vi designen och knapparna som byggs med hjälp av PML-editorn. Den består av olika knappar, som kör olika kommandon från servicen i theThing:

- Knappen "Länka Zigbee" används för att hämta mobilens API-nyckel, och spara denna i dess lagringsutrymme. API-nyckeln används av REST-API:et för att identifiera användaren och nyckeln används och krävs för alla framtida API-requests. Denna knapp behöver bara användas första gången man kör appen.
- Knappen "Hitta lampa" tar reda på hur många lampor som finns i systemet, och skapar två knappar för varje lampa. Den länkar sedan de nya knapparna till rätt lampa, så att en av knapparna stänger av lampan, och den andra sätter på den.
- Knapparna "LampaXPå" och "LampaXAv" skapas när man trycker på "hitta lampor" och X:et är här en referens till lampans nummer(i den ordning de förekommer i API:et). På-knappen sätter igång lampan, och av-knappen stänger av den.

Eftersom det inte gick att skapa knappar dynamiskt i PML-editor fick vi skapa en "custom part" istället (se 2.1.5.1 för beskrivning av custom part). Vi ville att knappen "Hitta Lampa" skulle ta reda på hur många lampor som fanns i systemet, och sedan skapa av- och på-knappar för för varje lampa. Det var dock inte möjligt eftersom man inte kunde skapa nya knappar direkt från applikationen. PML-editor saknar helt enkelt stöd för att skapa knappar dynamiskt. Med hjälp av en custom part kunde vi dock få detta att fungera, eftersom de skrivs i generell android-kod, och då kan alltså knappar skapas för varje lampa.

Bild 5.1 visar hur appen ser ut, inklusive alla knappar:

Bild 5.1: Vår applikation

5.2 Palcom service

Palcom-servicen kan köras via theThing, och innehåller flera metoder. Dessa metoder exekveras när den motsvarande knappen trycks ned, i applikationen. Metoderna kör deCONZ-kommandon (se kap 4.2), för att på olika sätt kontrollera Zigbee-lampor.

En bild med metoderna i servicen visas i bild 5.2:

Bild 5.2: Vår service

Här är en länk till servicen, för möjlighet till granskning. https://drive.google.com/file/d/1YqDNJDoGSZvQ-FO3q3FADR8WJtLJr86q/view?usp=sharing
Nedan beskrivs vad de olika metoderna gör:

- "link"-metoden hämtar API-nyckeln, och sparar den på enheten. API-nyckeln är en identifierare som används för autentisering inom systemet.
- "findLamps" hämtar mängden lampor som finns i systemet
- "findSensors" hämtar mängden sensorer i systemet
- "on…" sätter igång en vald lampa
- "off…" stänger av en vald lampa
- "SetName…" sätter namnet på vald lampa till valt namn

5.3 Scenario

För att göra det lättare att förstå hur bryggan fungerar har vi beskrivit ett scenario nedan, som beskriver vad som händer när man trycker på "link" i appen.

1. Användaren trycker på "link"-knappen i applikationen. Som tidigare förklarat i 4.1 så behöver man bara göra detta första gången efter att man startat appen.
2. Signal skickas från applikationen till theThing
3. theThing får signalen och kör då servicens "link"-metod
4. "link"-metoden skickar REST-kommandon för att hämta api-nyckeln (via deCONZ), och sätter nyckeln som utdata.
5. theThing sparar utdatan från servicen
6. Applikationen läser av den nya utdatan, och sparar API-nyckeln i lagringsutrymmet

6 Esoteric deCONZ Java API

6.1 Palcom service

Vi ville utforska andra sätt att arbeta mot deCONZ Rest API:et eftersom det behövdes skrivas en hel del kod för att göra något ganska enkelt som att få en lista på alla lampor i systemet. Därför undersöker vi i detta avsnitt en annan lösning och jämför sedan de båda. Denna andra lösning involverar ett externt bibliotek i Java för att arbeta mot deCONZ, kallat deCONZ REST Java API. https://github.com/EsotericSoftware/deconz

Vi hade tidigt en tanke om att utforska möjligheten att skapa virtuella lampor och vi ville testa vår applikation i ett större scenario eftersom vi enbart hade 1 resp 2 lampor var. Därför tänkte vi att det skulle kunna vara lämpligt att göra det med detta nya bibliotek. Att skapa virtuella lampor visade sig dock ej vara möjligt eftersom deCONZ API:et helt enkelt saknar stöd för att skapa virtuella lampor. I stället fann vi i dokumentationen, möjligheten att skapa en virtuell sensor av en särskild sensortyp. En så kallad CLIP SWITCH. Denna kom med olika attribut som går att ställa på olika vis och i vårt test så valde vi att växla dess config-attribut till antingen av eller på för att simulera en lampas status som antingen av eller på.

Biblioteket som vi använder är skrivet av en privatperson som själv använder det i ett program i vilket han hanterar flera olika protokoll och skapar kopplingar mellan dem. För detta har han skapat ett Java API för varje protokoll som han behöver stöd för i sitt program. Hans program är däremot inte möjligt att ladda ner. Hans bibliotek för deCONZ är emellertid öppet och fritt att ladda ner och det erbjuder alltså ett Java API för åtkomst till deCONZ REST API:et.

För att kunna använda biblioteket, klonade vi både Esoteric deCONZ Java-biblioteket och Palcom-projektet från respektive “GIT-repositories”. Sedan importerades dessa projekt i Eclipse som är det IDE som användes för denna del av utvecklingen. Ett nytt projekt skapades som vi kallade RunnerClassTest och dess Build Path konfigurerades. Detta innebar att Palcom-projektet samt Esoteric deCONZ-biblioteket länkades till vårt nya projekt, RunnerClassTest så att vi kunde använda de klasser som ingår i Palcom och de klasser som hör till deCONZ-biblioteket, tillsammans. Därefter kunde vi sedan börja skriva vår service som vi kallade EsotericService.

Vi skapade en liknande service som i 4.1 fast nu med Java-biblioteket som stöd. Vår service finns att ladda ner härifrån:
https://drive.google.com/file/d/1oq7gqbp2JpWN0TxE1wKsZ08Vxb0IP1WE/view?usp=sharing

Det som skiljer sig från servicen i 4.2 är egentligen bara att objekt behöver skapas för den data som önskas från systemet, i vårt fall lampor och sensorer. Nedan följer tillvägagångssätt och kodexempel på hur biblioteket används för att skapa metoder som kan exekveras av Palcom.

6.2 Bibliotekets användning för att skapa en service

1. Ett deCONZ objekt skapas med inparametrarna: nyckel, adress, port samt connection pool size.

```
DeCONZ myDeCONZ = new DeCONZ("57815C6C64", "192.168.1.149", 80, 1);
String apiKey  = myDeCONZ.getGateway().register("test app", null);
myDeCONZ.setApiKey(apiKey);
```

2. Objekt skapas för Lamporna och för sensorerna på liknande vis. Nedan illustrerat för lamporna. Observera att variablerna har tidigare deklarerats som static(visas ej).

```
myLightState = new Light.LightState();
myLightStateChange = new Light.LightStateChange();
myLights = myDeCONZ.new Lights();
```

3. För att ändra status hos en lampa används metoden “apply” som tar de objekt som tidigare skapats i punkt 2 och ändrar dess status:

```
Light currentLight = myLights.get(lightID);
myLights.apply(lightID,myLightStateChange.on(!currentLight.state.on));
```

4. Exempel på en av dessa metoder i en service skriven med "DeCONZ REST Java API" Här metoden toggleLight, som tar en inparameter LightID vilken är id:t hos en lampa i systemet.

```
public static void toggleLight(String lightID) throws DeCONZException {
    Light currentLight = myLights.get(lightID);
    myLights.apply(lightID, myLightStateChange.on(!currentLight.state.on));
}
```

Nedan följer nedsparade bilder av servicen när den exekveras inifrån TheBrowserThing.
Här trycks knappen “Find Sensors” ned och till höger skrivs resultatet ut. Här returneras antalet sensorer som finns i systemet.

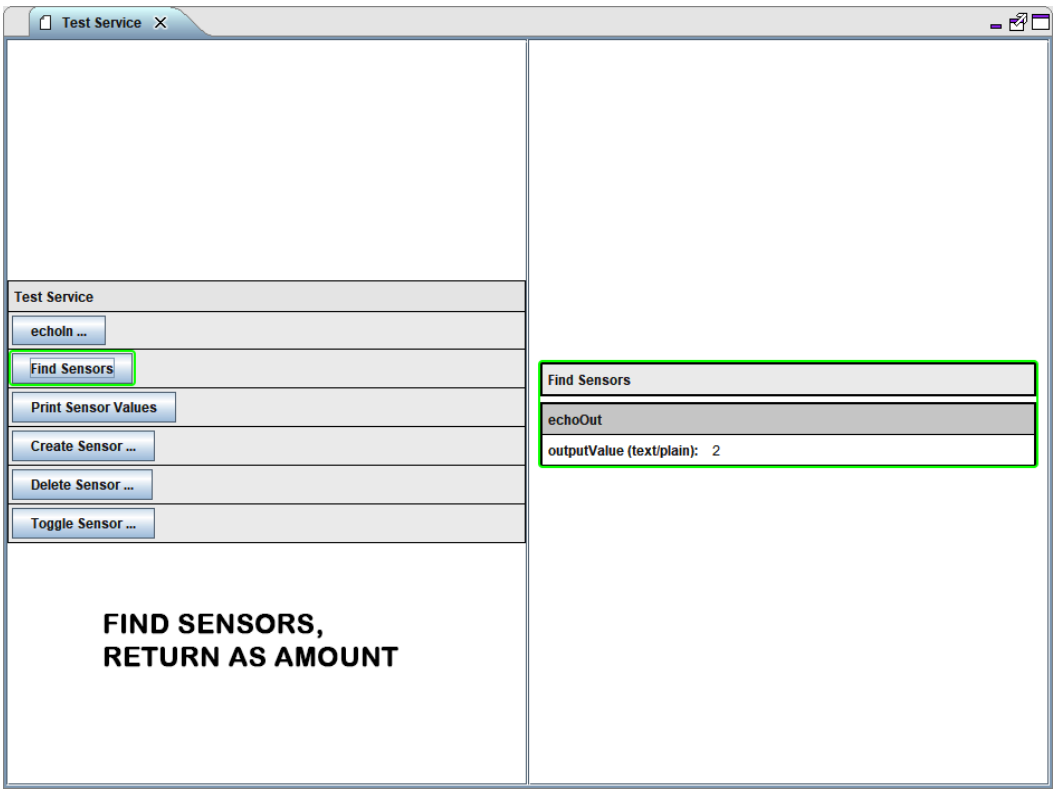

Bild 6.1: Som servicen ser ut när den används inuti TheBrowserThing. Här exekveras metoden “Find Sensors”.

Liknande scenario för “Print Sensor Values”. Den skriver ut den infon som specificerats i metoden. Här ID, namn på sensorn och dess av/på-värde.

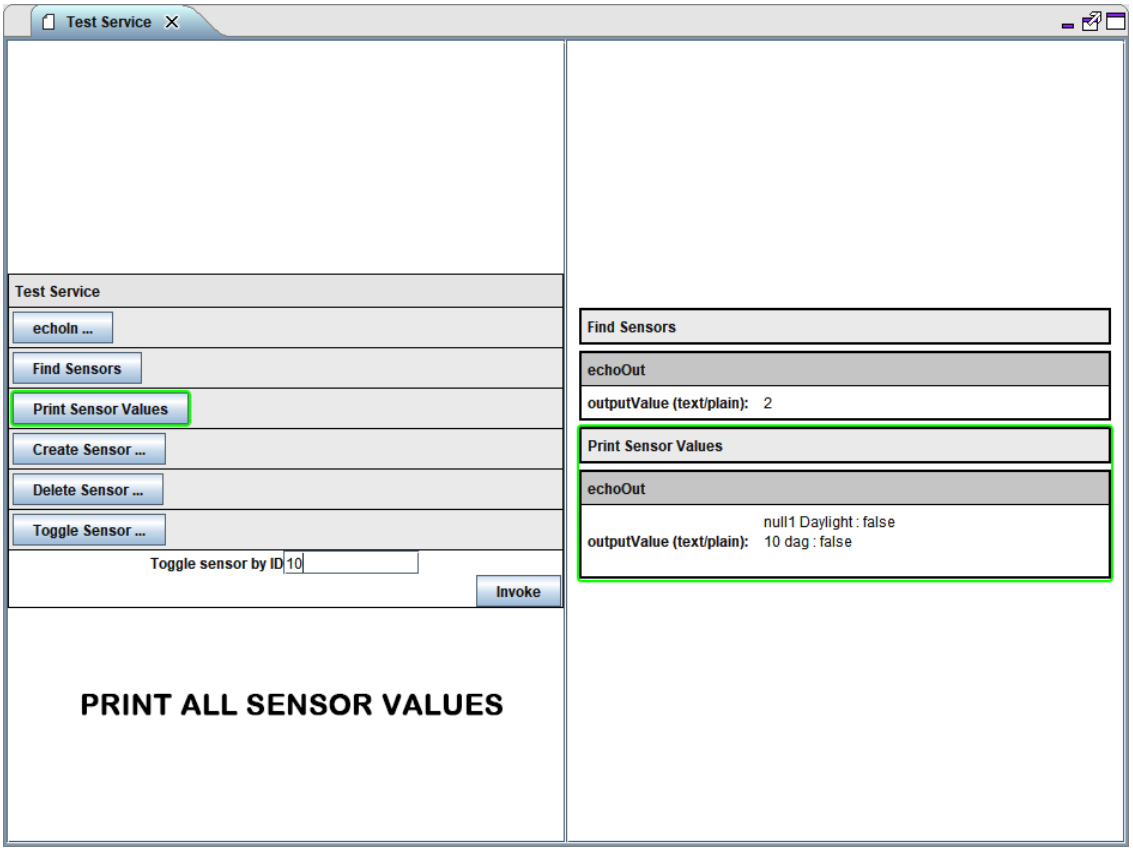

Bild 6.2: Metod som skriver ut varje sensor, dess ID nummer, namn och av/på-värde.

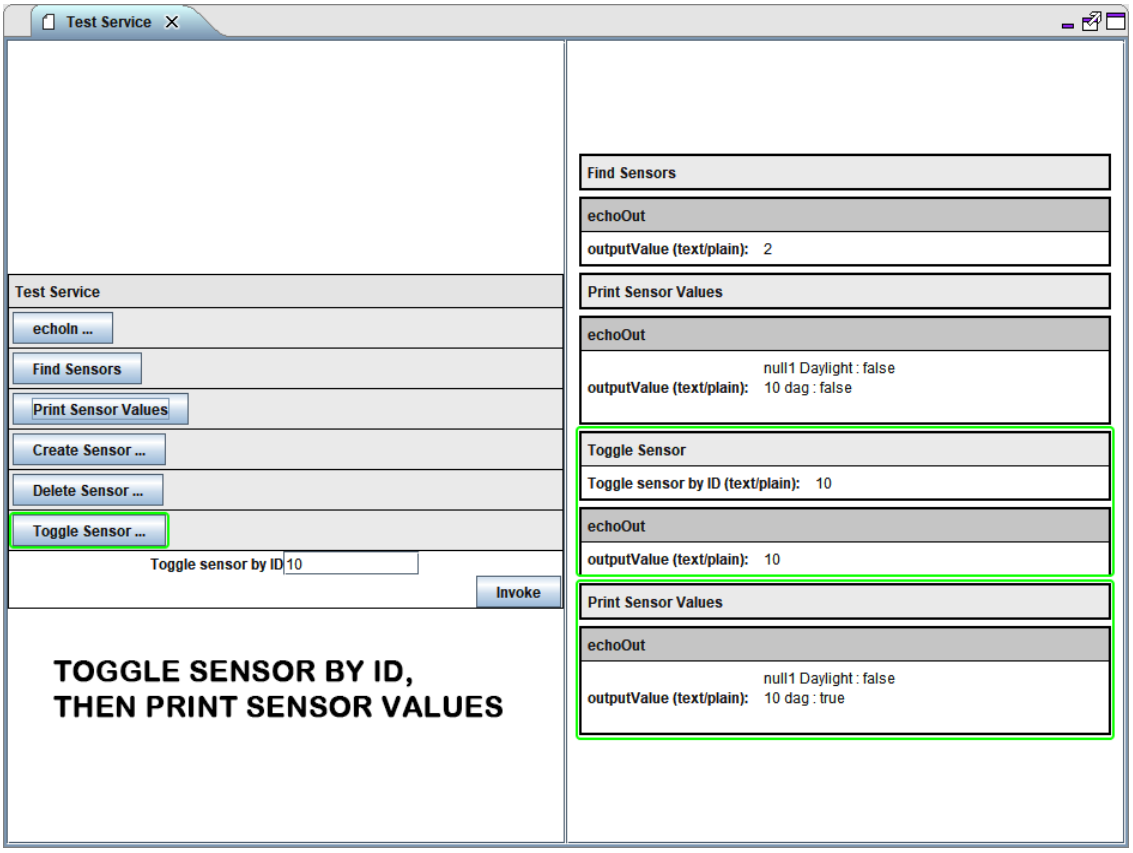

Bild 6.3: Metod som växlar specificerad sensors värde. Tar dess ID nummer som inmatning.

6.3 Jämförelse mellan lösningarna

Nedan följer en jämförelse av samma metod(bild 6.4 samt 6.5), implementerade i våra 2 services.

```
public static int findLamps(String IP, String port, String api) throws
IOException {
        HttpURLConnection connection = (HttpURLConnection) new URL("http://" +
IP + ":" + port + "/api/" + api + "/lights/").openConnection();
        connection.setRequestMethod("GET");
        connection.setDoOutput(true);

        int lamps = 2;
        int responseCode = connection.getResponseCode();
        String response = "";
        if(responseCode == 200){
            Scanner scanner = new Scanner(connection.getInputStream());
            while(scanner.hasNextLine()) {
                response += scanner.nextLine();
                response += "\n";
            }
            scanner.close();
```

```
            boolean hasMore = true;
            while (hasMore) {
                if (response.contains("\"" + lamps + "\":")) {*
                    lamps++;
                } else {
                    hasMore = false;
                }
            }
            return lamps-2;
        } else if (responseCode == 403){
            throw new IOException("connection failed");
        }
        return 0;
    }
```

Bild 6.4: findLamps-metoden som den ser ut i Zigbee Bridge.

```
public static int findLamps() throws DeCONZException {
    return myLights.getAll().size()-1;
}
```

Bild 6.5: findLamps-metoden som den ser ut i EsotericService. Vi drar av 1 från antalet eftersom raspBee2-dongeln returneras som en lampa också. Detta är en bugg som ska vara åtgärdat i senare version av deCONZ API:et.

Det finns både fördelar och nackdelar med de olika lösningarna, men vi vill hävda att ett bibliotek för Java är att föredra. Fördelen med detta är att det gör det mycket mer attraktivt att arbeta mot deCONZ eftersom det paketerar modulerna(Lights, Sensors etc.) i klasser med vilkas objekt du använder och gör direkta metodanrop med. Dessutom är det Java-kod som skrivs så det ser renare ut än att behöva konkatenera strängar för att bygga upp REST-kommandon.

Biblioteket var lite svårt att sätta sig in i då tidigare erfarenhet av att jobba med ett REST-API, saknades. Det var inte helt självklart att förstå hur det ska användas, utan det krävs att man experimenterar en del och försöker följa koden för att skapa de objekt som krävs och lista ut hur de ska kombineras för en korrekt HTTP-förfrågan.

Det klarnade emellertid så småningom och det gick sedan väldigt snabbt att bygga upp en service för att kunna göra lämpliga förfrågningar på de sensorer och lampor som ingick i nätverket. Även om allt finns där i form av funktionalitet till deCONZ-API:et så måste man fortfarande ha programmeringskunskaper för att kunna använda det och sätta ihop det till ett fungerande program. Detta betyder att det dessvärre inte är särskilt “plug-and-play”-vänligt för en person som saknar programmeringsvana.

Vad gäller Esoteric deCONZ Java API saknas dokumentation, utan det hela är distribuerat utan någon form av handledning som sådan, vilket är en klar nackdel. Det innebär att det ställer vissa krav på programmeringsvanan hos användaren. Samtidigt påstår vi att jobba direkt med HTTP-förfrågningar som andra alternativet är, ligger på samma tekniskt avancerade nivå.

7 Contribution statement (vem har gjort vad?)

7.1 Simon

- Skapade service som används i appen.
- Skapade första test-program, med demo
- Byggde programmet beskrivet i kapitel 5
- Skrev delar i rapporten:
    - Kapitel 1.1 (Bakgrund) (Några stycken)
    - Kapitel 2.1.1 (Arkitektur) (Ett stycke)
    - Kapitel 2.2.5.1 (Custom part)
    - Kapitel 2.3 (Raspberry Pi)
    - Kapitel 2.4 (Hårdvara)
    - Kapitel 3 (Några stycken)
    - Kapitel 4 (Bryggans arkitektur)
    - Kapitel 5 (Exempel på applikation av bryggan)
    - Kapitel 8 (Relaterat arbete)
    - Kapitel 9 (Slutsats) (Del av 9.1, 9.2, ett stycke av 9.3, 9.4)

7.2 Dag

- Skapade en service med hjälp av ett externt bibliotek.
- Skrev delar i rapporten:
    - Kapitel 1.1 (Bakgrund)(Några stycken)
    - Kapitel 2.1 (Palcom)
    - Kapitel 2.2 (Zigbee)
    - Kapitel 2.3 (Raspberry Pi, Några stycken)
    - Kapitel 3 (Zigbee 3.2 samt Esoteric 3.4)
    - Kapitel 6 (Esoteric Java API)
    - Kapitel 9.1 (Slutsats)(Ett stycke av 9.1, 9.3, 9.4)

8 Relaterat arbete

Denna del beskriver andra arbeten som har gjorts, som liknar vårt arbete. Flera av dessa använder sig av Zigbee och Raspberry Pi för automation. Av dessa arbeten är där dock inget som använder sig av Palcom. Vårt arbete är det första som kopplar Zigbee till just Palcom.

Smart Healthcare Monitoring System

I artikeln "Smart Healthcare Monitoring System" beskriver Abhishek Kumar, Gaurav Chattree och Sasikumar Periyasamy hur man kan bygga ett system för att lättare sköta sjukvård med hjälp av en Raspberry Pi och en Linux-dator.[18]

Denna artikel handlar om ett system för att hämta information från patienter, via en raspberry pi. Det blir ganska likt vårt arbete, då systemet använder sensorer för att skicka information till en raspberry pi. Raspberry Pi:n skickar sedan vidare informationen till en monitor. På samma sätt skickar vi information mellan en smartphone och lampor/sensorer via en raspberry pi. Till skillnad från oss används dock flera olika sorters sensorer, medan vi mest har använt lampor.

Remote-Control of High Efficiency and Intelligent Street Lighting Using a Zigbee Network of Devices and Sensors

I artikeln "Remote-Control of High Efficiency and Intelligent Street Lighting Using a Zigbee Network of Devices and Sensors" beskriver Fabio Leccese uppbyggnaden av ett Zigbee-nätverk för kontroll av gatulampor.[19]

Denna artikel handlar om ett Zigbee-nätverk för att kontrollera gatulampor. Skalan på detta projekt är såklart mycket större än vårt, men precis som vi använder de sig av ett Zigbee-nätverk för att hämta och skicka information till/från lampor.

Internet of Things–based smart home system using a virtualized cloud server and mobile phone app

I artikeln "Internet of Things–based smart home system using a virtualized cloud server and mobile phone app" beskriver Gou-Ming Sung, Yen-Shin Shen, Jia-Hong Hsieh och Yu-Kai Chiu utvecklingen av ett Zigbee-system för automatisering i hemmet.[20]

I deras lösning kopplas Zigbee-nätverket till en cloud-server, som skickar och tar emot data via en Raspberry Pi. Detta är ganska likt vårt arbete i att de också använder sig av Zigbee och Raspberry Pi. Till skillnad från vårt arbete handlar det mer om hemövervakning, snarare än lampor och sensorer.

Istället för att använda Palcom, som i vårt arbete, användes här en cloud-server för att skicka data till en webbläsare och en app. Det gör att man kan spara data i en databas istället för i själva appen. Fördelen med detta är att man kan nå datan från både en smartphone och en webbläsare. En databas kan även användas av fler personer, eftersom att data som ligger i databasen kan nås av flera smartphones.

I vårt arbete sparade vi istället data i applikationens lagringsutrymme, vilket är enklare eftersom att vi då inte behövde bygga en hel databas. En fördel med att använda lagringsutrymmet på telefonen är att man inte behöver ha tillgång till internet, utan kan nå datan offline.

Interoperable Internet-of-Things platform for smart home system using Web-of-Objects and cloud

I texten "Interoperable Internet-of-Things platform for smart home system using Web-of-Objects and cloud" från boken "Sustainable Cities and Society beskriver Asif Iqbal, Farman Ullah, Hafeez Anwar, Kyung Sup Kwak, Muhammad Imran, Waseef Jamal och Atta ur Rahman hur man kan möjliggöra hemautomation med hjälp av en IoT-platform.[21]

Denna text beskriver en IoT-platform som med hjälp av en Raspberry Pi används för att möjliggöra hemautomation. De använder sig av Raspberry Pi och REST-API, precis som vi gör. Även om där inte finns så mycket fokus på lampor som i vårt arbete, så beskrivs hur systemet används med olika sorters sensorer.

Wireless Sensor Network System Design Using Raspberry Pi and Arduino for Environmental Monitoring Applications

I texten "Wireless Sensor Network System Design Using Raspberry Pi and Arduino for Environmental Monitoring Applications" från boken "Procedia Computer Science" skriver Sheikh Ferdoush och Xinrong Li om byggandet av ett sensornätverk för miljöövervakning.[22]

Texten beskriver skapandet av ett Zigbee-nätverk av sensorer. Precis som vi gjorde, använde de också Raspberry Pi som en gateway mellan Zigbee och webbservern. Men till skillnad från oss använder de sig av en Arduino för att koppla ihop sensorerna.

9 Slutsats

Detta kapitel innehåller slutsatser och alternativ för vidareutveckling.

9.1 Återkoppling på frågor

Här ges svar på frågorna från problemformuleringen i del 1.3.

- Hur kan man använda en Raspberry Pi för att styra Zigbee enheter?

Med hjälp av en RaspBee2 som man kan sätta på en Raspberry Pi kan man få Pi:n att agera som en Zigbee-gateway. Detta betyder att man kan skicka kommandon till Zigbee-enheter via REST-api:et. För att läsa mer om Raspberry Pi och RaspBee2, se 2.3.

- Kan man skapa en generell lösning för kommunikation med andra Zigbee-enheter än lampor? Såtillvida att varje typ av enhet går att ansluta och användas i systemet utan att först behöva implementera särskilt stöd för just den typen?

Det går att skapa en lösning för kommunikation med sensorer, vilket vi gjorde med hjälp av ett bibliotek i vårt fall. Detta bibliotek gör egentligen inget mer än vad vi åstadkom med att inte använda det utan det förenklar bara processen att jobba med deCONZ API:et.

Det vi har gjort hittills är att skapa en lösning med ett GUI för att tända och släcka lampor. Att utveckla detta till en mer generell lösning för alla olika sorters Zigbee-enheter ser vi inte som en omöjlighet. Frågan vi ställer oss är vad som räknas som en generell lösning. Det hade varit möjligt att via HTTP-förfrågningar ge oss alla enheter i systemet och skapa knappar om det är en lampa, kanske en ikon om det är en sensor för att avläsa status etc. Att ha stöd för alla sorters enheter i vårt Zigbee-nätverk som är specificerat i deCONZ API:et.

Att jobba med HTTP-förfrågningar för detta ändamål blir däremot lätt klumpigt och där ser vi stora fördelar med att ha ett Java-API så att vi jobbar med ren Java-kod och objekt i stället. Hade vi haft mer tid och/eller kunskap om API-programmering så skulle det vara lämpligt att göra.

- Hur tar man emot data från Palcom i en Raspberry Pi?

Med hjälp av programmen i Palcom kunde vi använda services för att skicka data till Raspberry Pi:n. För att läsa mer om program och services, se 4.1.

- Vilka är skillnaderna mellan Palcom och Zigbee?

Palcom är en arkitektur som används för att koppla ihop olika enheter och system. I nuläget finns det dock inte några lampor som är byggda för att köras med Palcom. Zigbee är däremot ett protokoll som används för att styra enheter i ett nätverk. Den används av många olika tillverkare för att skapa kopplingar mellan enheter. Palcom är mer av en forsknings-prototyp, medan Zigbee är en standard för styrning av enheter. Se 2.1 för mer information om Palcom, och 2.3 för mer information om Zigbee.

- Hur kan man lättast översätta mellan signalerna inom Palcom och Zigbee?

En Raspberry Pi kan ta emot Palcom-kommandon via services, som den sedan skickar vidare som deCONZ-kommandon. Denna processen beskrivs i Kapitel 4.

9.2 Diskussion

Målet med examensarbetet har uppnåtts på ett till viss del tillfredsställande sätt. De viktigaste målen har uppfyllts då lösningen innefattar en koppling mellan Palcom och Zigbee. Denna består av en brygga som tar emot Palcom-signaler och skickar vidare dessa som Zigbee-signaler. Vi har även byggt en applikation som fungerar och kan användas för testning och demonstrering av bryggan. Bryggan kan utvecklas med fler funktioner utan större svårigheter.

Däremot är översättningen inte speciellt generell, utan kan bara användas för ett antal fördefinierade funktioner. Om man vill ha andra funktioner så måste man lägga till dessa funktioner manuellt. I nuläget kan bryggan användas för en begränsad mängd funktioner för kontrollering av lampor, och innehåller även en bas för användning av sensorer.

Vi har kunnat ge svar på nästan alla frågorna från vår problemformulering, och vi har utforskat de andra frågorna efter vår förmåga. Frågorna som saknar utvecklade svar handlar om skapande av en lösning för andra sorters enheter. Detta är något som vi har testat med ett program för sensorer, som tycks fungera. Däremot kan vi inte lägga till andra sorters enheter utan att lägga till dessa i ett program.

9.3 Olika lösningar

I arbetet skapade vi två olika lösningar, som fungerar på lite olika sätt. Den ena lösningen används för att kontrollera en lampa, och består av ett program med funktioner för detta. Det är denna lösning som framförallt är beskriven i kapitel 5.

Den andra lösningen går ut på att använda ett bibliotek för Java för att lättare jobba med deCONZ REST API:et. Denna metod beskrivs i avsnitt 6. För utvärdering har vi valt att utforska hur det skulle kunna se ut att jobba med virtuella enheter. För detta ändamål implementerade vi möjligheten att skapa virtuella sensorer och ändra dess attribut för att simulera en lampas av- och på-tillstånd.

9.4 Vidareutveckling

Efter examensarbetet finns flera olika möjligheter för vidareutveckling av bryggan.

Med nuvarande system kan man endast kontrollera lampor, med endast en liten mängd funktioner. Det kan vara värt att lägga till ytterligare funktioner till dessa, och även att lägga till stöd för andra sorters enheter. Man hade till exempel kunnat lägga till funktioner som schemaläggning av lampors på-/avstängning, eller gruppering av enheter.

För säkrare användning av systemet kan det även vara nödvändigt att se över dess säkerhet, då detta är något som vi inte har tagit hänsyn till i vår utveckling. För att öka säkerheten

skulle man behöva göra tester, och se hur systemet klarar sig mot olika sorters intrång. Därefter hade man kunna åtgärda eventuella problem med säkerheten.

10 Referenser

[1] "Vad är IoT?". iotsverige.se. https://iotsverige.se/om-oss/vad-ar-iot/. [Accessed May 15th 2021]

[2] C. Patel. "Bright future of IoT". metizsoft.com. https://www.metizsoft.com/blog/bright-future-of-iot. [Accessed May 14th 2021]

[3] "Network basics: Bridges". dummies.com. https://www.dummies.com/programming/networking/network-basics-bridges/. [Accessed Apr. 7th 2021]

[4] "Zigbee - Zigbee Alliance". zigbeealliance.org. https://zigbeealliance.org/solution/zigbee/. [Accessed Apr. 21st 2021]

[5] David Svensson Fors, Boris Magnusson, Sven Gestegård Robertz, Görel Hedin, & Emma Nilsson-Nyman. "Ad-hoc composition of pervasive services in the PalCom architecture". Pervasive Services, pp. 83–92, July 2009. https://doi-org.ludwig.lub.lu.se/10.1145/1568199.1568213

[6] "What is a REST API?". redhat.com. https://www.redhat.com/en/topics/api/what-is-a-rest-api. [Accessed Apr. 8th 2021]

[7] "What is an API key?". rapidapi.com. https://rapidapi.com/blog/api-glossary/api-key/. [Accessed Apr. 8th 2021]

[8] Johnsson, B. A. and Weibull, G. (2016) 'End-User Composition of Graphical User Interfaces for PalCom Systems', Procedia Computer Science, 94, pp. 224–231. doi: 10.1016/j.procs.2016.08.035. https://doi-org.ludwig.lub.lu.se/10.1016/j.procs.2016.08.035

[9] **"**What is Zigbee?". homey.app. https://homey.app/en-au/wiki/what-is-zigbee/. [Accessed Jan. 20th 2021]

[10] "Zigbee technology". ubisys.de. https://www.ubisys.de/en/technology/zigbee-technology/. [Accessed Jan. 23rd 2021]

[11] Zigbee Wireless Networking

https://search.ebscohost.com/login.aspx?direct=true&db=cat02271a&AN=atoz.ebs45677e&site=eds-live&scope=site
[Accessed Jan. 22nd 2021]

[12] "What is Zigbee? Introduction and look at TI's solutions". training.ti.com. https://training.ti.com/what-zigbee-introduction-and-look-tis-solutions. [Accessed Jan. 23rd 2021]

[13] "Raspberry Pi Foundation - About Us". raspberrypi.org. https://www.raspberrypi.org/about/. [Accessed Mar. 4th 2021]

[14] "What is a Raspberry Pi?". raspberrypi.org. https://www.raspberrypi.org/help/what-%20is-a-raspberry-pi/. [Accessed Mar 4th 2021]

[15] "RaspBee 2 Overview". phoscon.de. https://phoscon.de/en/raspbee2. [Accessed Mar. 4th 2021]

[16] "The RaspBee 2". dresden-elektronik.com. https://www.dresden-elektronik.com/tidings/the-raspbee-ii-now-available-in-the-shop.html. [Accessed Mar 4th 2021]

[17] "ConBee Software". phoscon.de. https://phoscon.de/en/conbee/software. [Accessed Mar. 12th 2021]

[18] Kumar, A., Chattree, G. & Periyasamy, S. Smart Healthcare Monitoring System. Wireless Pers Commun 101, 453–463 (2018). https://doi.org/10.1007/s11277-018-5699-0

[19] F. Leccese, "Remote-Control System of High Efficiency and Intelligent Street Lighting Using a Zigbee Network of Devices and Sensors," in IEEE Transactions on Power Delivery, vol. 28, no. 1, pp. 21-28, Jan. 2013, doi: 10.1109/TPWRD.2012.2212215. https://doi.org/10.1109/TPWRD.2012.2212215

[20] Sung, G.-M. (1,2), Y.-S. (1,2) Shen, J.-H. (1,2) Hsieh, and Y.-K. (1,2) Chiu. "Internet of Things–based Smart Home System Using a Virtualized Cloud Server and Mobile Phone App", International Journal of Distributed Sensor Networks 15, no. 9. doi:10.1177/1550147719879354. https://doi.org/10.1177/1550147719879354

[21] Asif Iqbal, Farman Ullah, Hafeez Anwar, Kyung Sup Kwak, Muhammad Imran, Waseef Jamal, Atta ur Rahman, "Interoperable Internet-of-Things platform for smart home system using Web-of-Objects and cloud", Sustainable Cities and Society, vol. 38, 2018, pp. 636-646, ISSN 2210-6707. https://doi.org/10.1016/j.scs.2018.01.044.

[22] Sheikh Ferdoush, Xinrong Li, “Wireless Sensor Network System Design Using Raspberry Pi and Arduino for Environmental Monitoring Applications”, Procedia Computer Science, vol. 34, 2014, pp. 103-110, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2014.07.059.