CHALMERS

Effektiv utrymmesanvändning med datainsamling i realtid

ALADIN BECOVIC
OSCAR ARVERING WALLDEN

Institutionen för Elektroteknik
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2019

Examensarbete 2019:NN

Effektiv utrymmesanvändning med datainsamling i realtid

ALADIN BECOVIC
OSCAR ARVERING WALLDEN

CHALMERS

Institutionen för Elektroteknik
Chalmers Tekniska Högskola
Göteborg, Sverige 2019

Effektiv utrymmesanvändning med datainsamling i realtid

ALADIN BECOVIC
OSCAR ARVERING WALLDEN

Handledare: Magnus Sandin, Collibay AB
Examinatior: Andreas Fhager, Institutionen för Elektroteknik

Examensarbete 2019:NN
Institutionen för Elektroteknik
Chalmers Tekniska Högskola
412 96 Göteborg
Telefon +46 31 772 1000

Omslag: Visualisering av uppkopplade enheter.

Göteborg, Sverige 2019

Efficient space usage using data gathered in realtime

ALADIN BECOVIC
OSCAR ARVERING WALLDEN

Institutionen för Elektroteknik
Chalmers Tekniska Högskola

Abstract

According to Collibay AB the demand for office- and warehouse space will increase while the availability will decrease in the near future. The goal of the company is to provide a platform where those who are in need of these spaces easily can get in touch with those who have unused space to rent. The goal of this project was to develop sensors which can be used to automate the monitoring of the spaces owned or rented by a company so that unused spaces can be rented to those in need of it, thus decreasing the unnecessary costs of the company while providing office- or warehouse space to those in need of it. The aim of this projects was to develop a system which consists of wireless sensors that gather data that is then sent to a main unit which handles the received data. The main goal of the project was to develop a system that doesn't interfere with surrounding wireless networks and devices, this was solved by using the LoRa wireless transmission standard which uses 868MHz for data transmission. By comparing the frequencyspectrum of 868MHz and a more commonly used frequency, 2.4GHz, it's clear that the interference from surrounding systems is almost completely eliminated.

Keywords: transciever, radiofrequency, wireless, transmission, data, sensor.

Sammanfattning

Undersökningar gjorda av Collibay AB visar att endast 40% av kontorsutrymmen och lagerlokaler används utav de företag som äger eller hyr dessa utrymmen. De ser också att förfrågan för dessa utrymmen kommer att öka de kommande åren och har därför skapat en plattform där man kan hyra ut det utrymme som inte används. Målet med detta projektet har varit att skapa ett system som ska kunna användas för att samla in data om hur de olika utrymmen används för att effektivt kunna hyra ut det utrymme som inte används till de som behöver det samtidigt som de onödiga kostnaderna minskar. Det huvudsakliga målet med projektet har varit att skapa ett system som inte stör eller störs av kringliggande system och trådlösa nätverk då ett enda system är tänkt att bestå av ett större antal sensorenheter som samlar in data. Eftersom både bluetooth och Wifi använder sig utan frekvensen 2.4GHz för trådlös överföring var ett av målen att använda en standard för trådlös överföring som inte arbetar på denna frekvens vilket resulterade i att LoRa:s standard för trådlös överföring användes. Anledningen till att just LoRa valdes var för att överföring av data sker på licensfria frekvenser, 868MHz inom EU, och dess väldigt långa räckvidd.

Keywords: mottagare, sändare, trådlös, överföring, data, sensor.

Förord

Detta examensarbete har utförts som ett avslutande momentet på elektroingenjörsprogrammet som är 180hp på Chalmers Tekniska Högskola. Arbetet omfattar 15hp och är utfört på Collibay AB.

Stort tack till Jerker Funnemark och Magnus Sandin på Collibay AB som gett oss möjligheten att utföra detta projekt på deras företag. Vi vill också tacka Magnus för att han har stöttat oss under projektets gång och sett till att vi snabbt har fått de komponenter som fattats.

Vi vill också tack Andreas Fhager som tagit sig tid att vara vår examinator och som har följt oss under arbetets gång.

Aladin Becovic och Oscar Arvering Walldén, Göteborg, Juni 2019.

Innehåll

1
Introduktion

1.1 Bakgrund

Målet med projektet är att skapa ett system som kommer att användas för att samla in information om hur olika utrymmen, t.ex. lagerlokaler, konferensrum och kontor, utnyttjas av de företag som hyr eller äger dessa utrymmen. Systemet kommer alltså att användas i lokaler där det finns ett eller flera trådlösa nätverk samt andra trådlösa enheter. Idag är 2.4 GHz den frekvens som är vanligast vid trådlös överföring av information och både Wi-fi [1] och bluetooth [2] använder sig av denna frekvens för trådlös överföring vilket i sin tur leder till att det system ska skapas har som krav att inte interferera med redan befintliga trådlösa system. Utöver frekvenskravet så ska systemet kunna föra över information trådlöst även vid större avstånd då storleken på de utrymmen där systemet ska användas kan variera. Därför vill man säkerställa att systemet är flexibelt då både avstånd och antalet sensorer i ett enskilt system kan variera.

1.2 Syfte

Syftet med detta arbetet är att ta fram en prototyp av en trådlös sensor. Denna sensor ska samla in och lagra data om utrymmets användning. Företaget Collibay AB ser att inom några år kommer efterfrågan på kontors- och lagerutrymme ha ökat drastiskt och att utbudet minskat. Deras mål är att effektivisera de kontorsutrymmen som finns och erbjuda smart lösningar till företag som behöver det. Denna prototyp ska vara baserad på och ha samma funktion som en av företagets nu inhyrda sensorer. Den ska fungera minst lika bra och ha möjlighet att bli en färdig produkt som ska kunna produceras så att företaget inte ska behöva hyra sensorer.

1.3 Avgränsningar

För att kunna uppnå målen måste ett par avgränsningar deklareras redan innan arbetet börjar. Det första av dessa avgränsningarna är att sensorerna ska vara trådlösa vilket betyder att de ska vara batteridrivna och ha trådlös överföring av data. Som följd av att sensorerna drivs av batterier får inte komponenterna dra för mycket ström. Målet är att sensorn inte ska behöva laddas på minst en månad. En annan avgränsning är över vilka frekvenser det är lagligt och mest optimalt att skicka data. En annan avgränsning är att hela sensorn inte får vara för klumpig vilket betyder att komponenterna inte får vara för stora och ska kunna brytas ut för att bygga en färdig produkt.

1.3.1 Mål med projektet

- En fungerande prototyp som kan sända specifik data från valfri plats till huvudenhet.
- Prototypen ska kunna vara igång utan att behöva byta och ladda batterierna i minst 40 dagar kontinuerligt.
- Ska klara av att kunna skicka data över minst en våning.
- Inte interferera och inte heller få för mycket störningar av andra enheter i närheten.

1.3.2 Systemets funktion

Systemet är tänk att fungera på följande sätt (All kommunikation sker trådlöst):

1. Sensorenheter kopplas samman med en huvudenhet.
2. Huvudenheten skickar en hämtningsförfrågan till sensorenhet.
3. Sensorenheten tar emot en hämtningsförfrågan, läser av nuvarande värde från sensor och skickar sedan den avlästa datan till huvudenheten.
4. Data tas emot av huvudenhet från sensorenhet och behandlas.
5. Återupprepa steg 2-4 för varje kopplad sensorenhet.
6. Vänta en viss tid och påbörja nästa hämtning av data från de kopplade sensorenheterna.

2
Teknisk bakgrund

Eftersom ett av målen för systemet är att kunna föra över data över långa sträckor och att inte störa andra enheter och system i de utrymmen där det är tänkt att användas så jämfördes först olika typer av standarder för trådlös dataöverföring [se tabell 2.1]. I tabellen kan man se att 2.4GHz är den frekvens som idag används av både Wifi [1] och bluetooth [2] och därför behöver systemet använda en standard som inte använder 2.4GHz för trådlös dataöverföring för minimal interferrens med övriga system. Den standard som valdes var LoRa eftersom den används för att föra över data över långa sträckor och dataöverföring sker på licensfria frekvenser, 868 MHz inom EU [3].

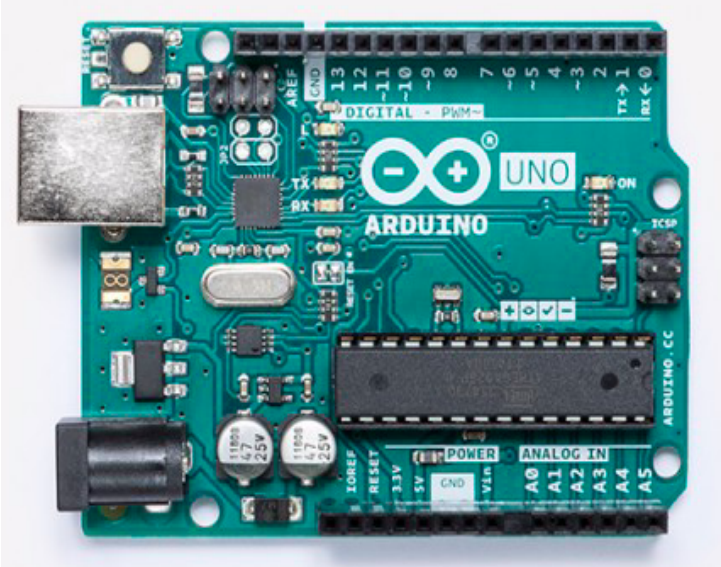
Tabell 2.1: Olika standarder för trådlös dataöverföring

Standard	**Frekvens**	**Överföringshastighet**	**Räckvidd**
LoRa[4]	868Mhz inom EU	<50 kbps	15km fri sikt
Wifi[1]	2.4GHz och 5GHz	<200Mbps	50m
Bluetooth[2]	2.4GHz	1Mbps	50-150m
ZigBee[5]	2.4GHz	250kbps	10-100m

För att programmera sensorerenheterna och huvudenheten som ingår i systemet valdes utveckligskortet Arduino eftersom det finns väldigt många färdiga bibliotek för många olika typer moduler och tillsammans med deras utveckligsmiljö är det väldigt enkelt att snabbt kunna programmera en fungerande prototyp. Den transcievermodul som valdes var Reyax RYLR896 då modulen använder LoRa:s standard vilket säkerställde att systemet uppfyller de krav som ställdes på det. Reyax RYLR896 valdes också för att Arduino och modul enkelt kan kommunicera med varandra via seriell kommunikation genom att skicka AT-kommandon [6] mellan Arduino och modul i form av textsträngar.

2.1 Arduino

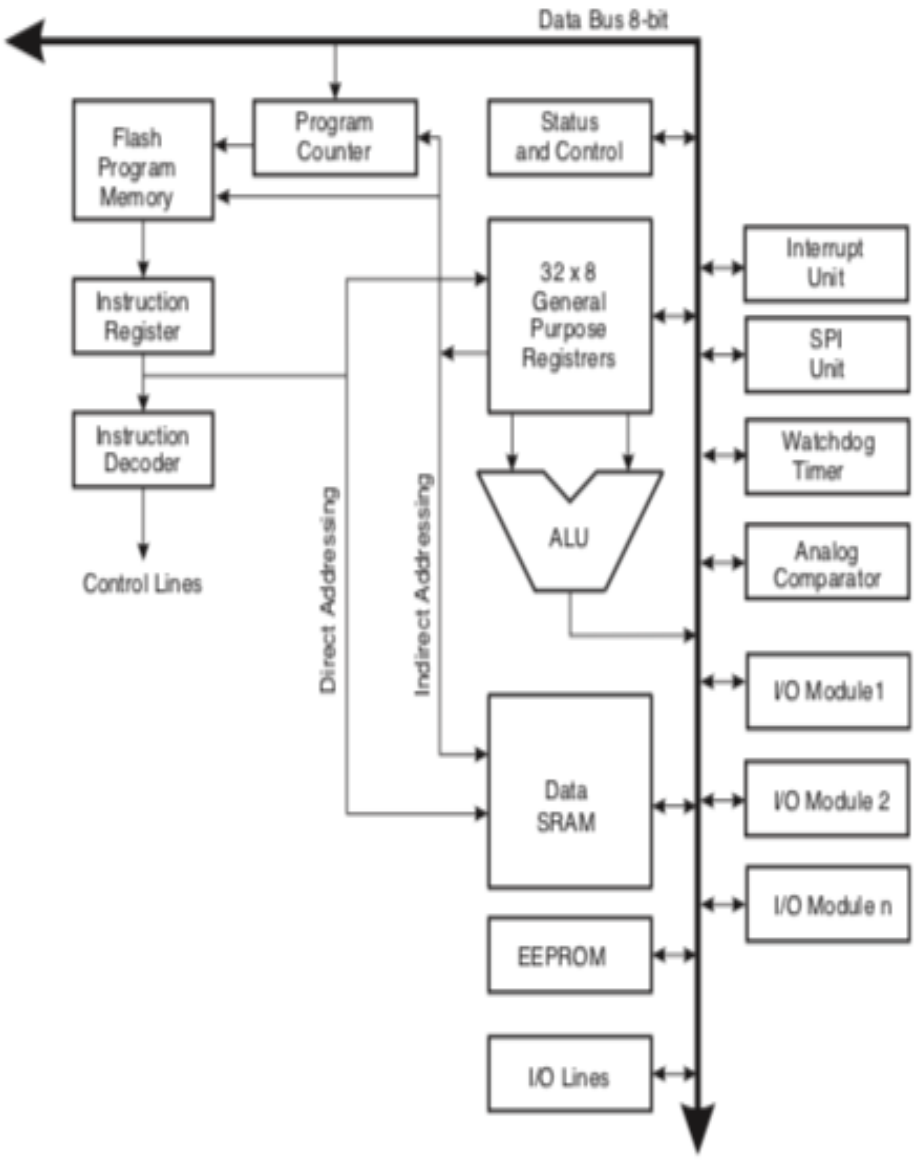
Arduino Uno/Nano är ett open-source utvecklingskort som är baserat på mikrokontrollern ATmega328P. Utvecklingskortet Arduino Uno har 14 digitala in/ut-portar samt 6 analoga in-portar. På kortet finns även en 16 Mhz klocka samt en USB-port för att enkelt kunna ansluta kortet till en datorn och programmera det [7]. Det finns även en färdig utvecklingsmiljö framtagen specifikt för Arduino [8] som underlättar programmeringen av mikrokontrollern samtidigt som det finns väldigt många färdiga bibliotek för just Arduino.



Figur 2.1: Arduino Uno utvecklingskort [7]

2.1.1 ATmega328P

ATmega328P är en 8-bitars AVR(Advanced RISC Architecture) mikrokontroller. ATmega328P använder sig av en Harvard arkitektur [se figur 2.2] Vilket innebär att program och data har separata minnen och bussar. Detta resulterar i att under tiden en instruktion exekveras så laddas nästa instruktion från minnet vilket leder till att instruktioner kan exekveras i varje klockcykel. Detta ger maximal prestanda och parallelism för microcontrollern [9].



Figur 2.2: ATmega328P Arkitektur[9]

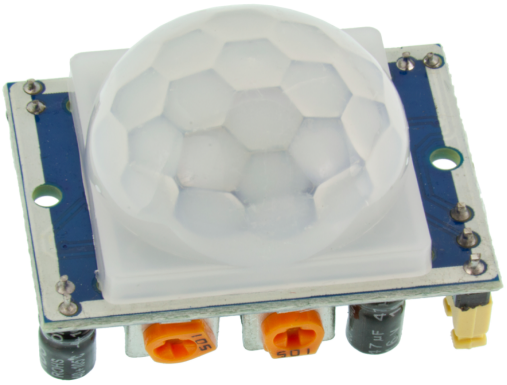
2.2 Reyax RYLR8XX

Reyax RYLR8XX är en trådlös överförings modul som använder LoRa:s dataöverföringsteknik
LoRa används i huvudsak till att bygga trådlösa nät som kan föra över data över väldigt långa sträckor [4]. Reyax RYLR8XX arbetar på frekvensen 868 MHz vilket i Sverige är en licensfri frekvens [3]. Modulen har en räckvidd på upp till 15km samtidigt som strömförbrukningen är låg. Då modulen tar emot data är förbrukningen 16.5mA och när modulen skickar data så är förbrukningen 43mA, utöver detta så kan modulen även sättas i strömspararläge vilket leder till en strömförbrukning på 0.5μA [10].

Figur 2.3: Reyax RYLR8XX [10]

2.3 HC-SR501

HC-SR501 är en rörelsedetektor med en PIR(Passive infrared)-sensor som används för att registrera rörelse genom att detektera en förändrig i temperatur framför sensorn med hjälp av sensorer som detekterar infrarött ljus [11]. En fördel med HC-SR501 är dess låga strömförbrukning på endast 50μA om värde inte hämtas, går upp till 65mA när värde hämtas från sensor. Sensorn har ett avkänningsområde på 110° och kan detektera förändringar inom avståndet 3-7m [12].



Figur 2.4: HC-SR501 [12]

2.4 Mätutrustning

Den mätutrustning som använts för att mäta strömförbrukningen och för att analysera och jämföra olika frekvensspektrum.

2.4.1 Oscilloskop

Ett oscilloskop är ett mätinstrument som används för att skapa en grafisk representation över hur spänningen varierar över tid. Den två-dimensionella plotten som ritas upp av oscilloskopet kan sedan användas för att utföra olika mätningar, t.ex. mätning av amplitud och frekvens [13]. För att kunna beräkna strömförbrukningen över tid med ett oscilloskop så mäts spänningen över en känd resistans och strömmen genom resistansen beräknas med hjälp av Ohms lag [14].

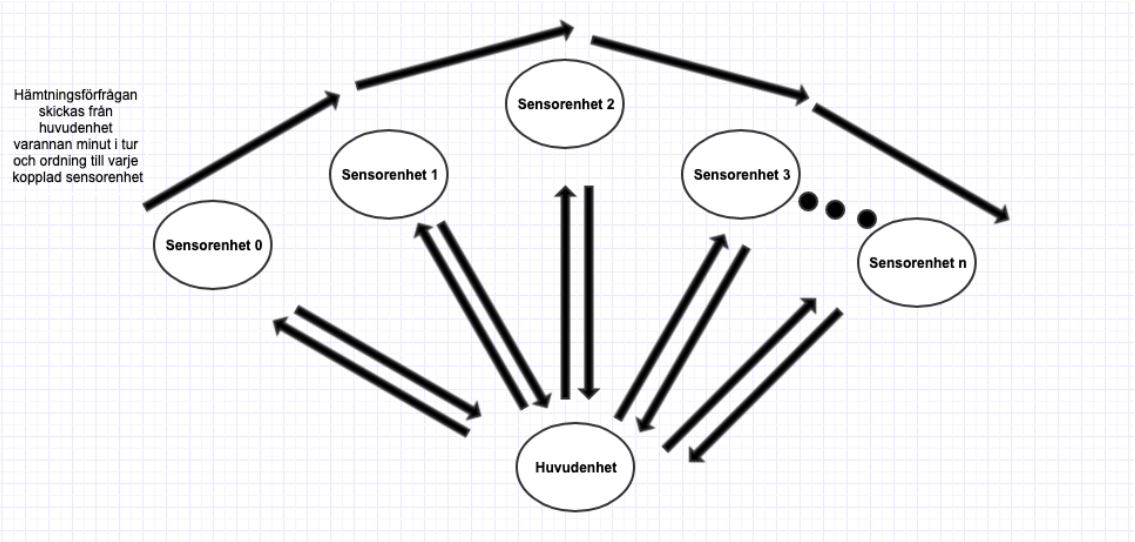
2.4.2 Spektrumanalysator

En spektrumanalysator är ett mätinstrument som används för att mäta amplitud mot frekvens jämfört med ett oscilloskop som mäter amplitud mot tid. Spektrumanalysatorn används för att analysera dominant frekvens, bandbredd, effekt och andra spektrumkomponenter vilket lämpar sig för analys av t.ex. trådlösa sändtagare [15].

3
Metod

Systemet är tänk att bestå av två huvuddelar, en huvudenhet som hämtar data från varje kopplad sensorenhet med jämna mellanrum och en sensorenhet som kan placeras på de platser där man vill utföra mätningarna. Huvudenheten kommer att bestå av en Arduino som sköter all logik, till Arduino kopplas en Reyax RYLR8XX så att data ska kunna skickas mellan Arduino:s trådlöst. Sensorenheten består också av en Arduino och en Reyax RYLR8XX men till den kopplas också en rörelsesensor som data kommer hämtas från varje gång som sensorenheten får en hämtningsförfrågan.

Eftersom systemet kommer att bestå av en huvudenhet och flera sensorenheter så kommer sensorenheterna vid start, om de inte redan är kopplade till huvudenhet, starta en initiering med huvudenheten där huvudenheten skickar sin egna unika adress till sensoreneheten som sparar den och sedan kommer huvudenheten att skicka en egen unik adress till sensorenheten. Huvudenheten kommer också att spara ner de unika sensoradresserna för att varannan minut skicka en hämtningsförfrågan till de sensorenheter som är kopplade till huvudenheten och sedan skickar sensorenheterna datan som hämtas från deras rörelsesensorer till huvudenheten.
Efter att sensorenheten är initierad så kommer den 10 sekunder efter senaste överföring att gå in i strömsparläge, detta görs genom att strömförsörjningen till Reyax RYLR8XX bryts med hjälp av en extern transistor och efter det så kommer Arduino att gå in i strömsparläge. Tiden för hur länge sensorenheten ska vara i strömsparläge kommer sedan att beräknas eftersom tiden för nästa hämtning varierar med antalet sensorenheter kopplade till huvudenhet. Huvudenheten påbörjar nästa hämtning två minuter efter att föregående hämtning av data är avslutad, en hämtning innebär att huvudenheten skickar en hämtningsförfrågan i tur och ordning till varje kopplad sensorenhet och väntar på att sensorenhet skickar datan som den läser av från rörelsesensor innan huvudenheten skickar en hämtningsförfrågan till nästan sensorenhet [se figur 3.1]. Anledningen till varför tiden mellan hämtning av data från sensorenheterna valdes till två minuter berodde på att det gjordes en avvägning mellan hur pass realtid systemet ska vara och hur lång batteritid varje enskild sensorenhet borde ha. En periodisk hämtning av data från alla kopplade sensorenheter valdes eftersom det ansågs att inte varje enskild rörelse behöver registreras då det hade gett en väldigt varierande batteritid mellan de olika sensorenheterna, beroende på hur ofta de aktiveras. Just två minuter valdes på grund av den medelförbrukning av ström som det innebär vilket resulterar i att den önskade batteritiden uppnås vilket kan ses i kapitel 4.



Figur 3.1: Kommunikationen mellan huvudenhet och sensorenheter

3.1 Programmering av system

Innan systemet började programmeras så anslöts Reyax RYLR8XX till Arduino och med hjälp av en seriell monitor, som är en inbyggd funktion i Arduino:s utvecklingsmiljö, kunde kommunikationen mellan Reyax RYLR8XX och Arduino enkelt observeras för att säkerställa att Reyax RYLR8XX utförde de instruktioner som skickades från Arduino. Därefter programmerades Arduino så att den med hjälp av Reyax RYLR8XX kunde skicka data till en annan Arduino med tillhörande Reyax RYLR8XX. När datan togs emot av mottagaren så observerades resultatet i en seriell monitor för att kontrollera att den skickade datan kommit fram utan några förluster. Kommunikationen mellan Arduino och Reyax RYLR8XX programmerades så att det inte kan skickas mer än en instruktion var tredje sekund [se figur A.1 i Appendix]. Detta gjordes för att Reyax RYLR8XX först tar emot en instruktion från Arduino och sedan svarar Reyax RYLR8XX med en textsträng som bekräftar att kommandot genomförts korrekt, vid eventuellt fel så är textsträngen istället en felkod som kan användas för att underlätta felsökning.

Efter att ha programmerat och testat de mest fundamentala funktionerna, att skicka [se figur A.1 i Appendix] och ta emot data [se figur A.2 i Appendix], så programmerades det en initieringsfunktion eftersom Reyax RYLR8XX är fabriksinställda så att de har adressen 0 vilket inte är en unik adress för just den enheten och den data som skickas till adress 0 kan tas emot utav alla Reyax RYLR8XX. Initieringsfunktionen programmerades så att först så skickar huvudenheten sin egna unika adress till sensorenheten som i sin tur sparas så att sensorenhet endast kan kommunicera med den specifika huvudenheten. Efter att sensorenheten har tagit emot huvudenhetens adress så skickar huvudenheten en specifik adress för just den sensorenheten som i sin tur tar emot denna adress och sedan ändrar sensorenheten sin egna adress till den unika adress som tagits emot från huvudenhet. Detta säkerställer att sensorerenheterna endast kommunicerar med den huvudenhet som de är anslutna till

och huvudenheten kan kommunicera med specifika sammankopplade sensorenheter utan risk för att datan tas emot av fel sensorenhet.

För att ny sensordata ska hämtas med jämna mellanrum så skapades en fetch funktion i huvudenheten som ser till att sensordata hämtas från alla anslutna sensorer två minuter efter att föregående hämtning är klar [se figur A.5 i Appendix].
Eftersom sensordata hämtas två minuter efter föregående hämtning är klar så kan tiden för nästa hämtning beräknas enkelt då det är 3 sekunder mellan varje hämtning av data. För att uppnå maximal batteritid så programmerades sensorenheten så att de 10 sekunder efter att ha fört över data till huvudenhet går in i ett strömsparar-läge [se figur A.6 i Appendix]. När Arduino går in i strömsparar-läge så bryts strömförsöjningen till Reyax RYLR8XX med hjälp av en transistor som styrs av Arduino, detta görs för att uppnå minimal strömförbrukning under den tiden som sensorenheten väntar på nästa hämtning av data. Den tid som sensorenheterna ska vara i strömsparar-läge implementerades i koden utifrån följande ekvation:

$$t = 120 - 10 + 3 * (n - 1)[s]; \text{n} = \text{antal sensorer kopplade till huvudenhet}$$

Antal sensorenheter kopplade till huvudenhet skickas till sensorenhet varje gång som huvudenhet skickar en hämtningsförfrågan till sensorenhet så att den tid som de är i strömsparar-läge anpassas dynamiskt beroende på antalet kopplade sensorenheter. För att tiden alltid ska stämma, även om nya sensorer ansluts till huvudenhet under den tid som hämtning av data pågår, så programmerades huvudenheten så att data inte hämtas från de sensorenheterna som anslutits under pågående hämtning innan nästa hämtning påbörjas.

3.2 Sänkning av systemets strömförbrukning

Efter att systemet var programmerat behövdes målet med att sensorenheten skulle drivas på batteri realiseras. En väldigt stor del utav detta projekt handlar om att sensorenheten ska kunna vara online i minst 40 dagar och hur detta ska realiseras. Detta gjordes i tre steg där det gjordes mätningar för att kunna beräkna hur länge sensorenheten potentiellt sätt skulle kunna drivas med de batterier som företaget valde. Det första steget implementerade strömsparfunktionen för Arduinon vilket resulterar i en märkbar minskning i förbrukningen. Denna minskning skulle dock inte vara tillräcklig för att få enheten att uppnå målen så det gjordes ett försök till att lokalisera vad det är som är energiboven i sensorenheten. Det undersöktes olika metoder för att effektivt sänka Arduino:s passiva energikonsumption.

Det upptäcktes att de två huvudsakliga och enklaste sätten att sänka konsumtionen var att koppla bort en LED som alltid är på som kallas POWER LED. Även att koppla bort den linjära spänningsregulatorn som tillförser Arduinon med rätt spänning. Denna har stor påverkan på hur mycket Arduinon förbrukar. Eftersom den enda spänningingskällan som fanns tillgänglig var ett 9 volt batteri kunde dessvärre inte spänningsregulatorn kopplas bort för då skulle de interna komponenterna komma till

skada [16]. När POWER LED kopplas bort sker en global minskning i förbrukningen för hela enheten men det blir uppenbart att detta inte heller är en tillräckligt hög minskning.

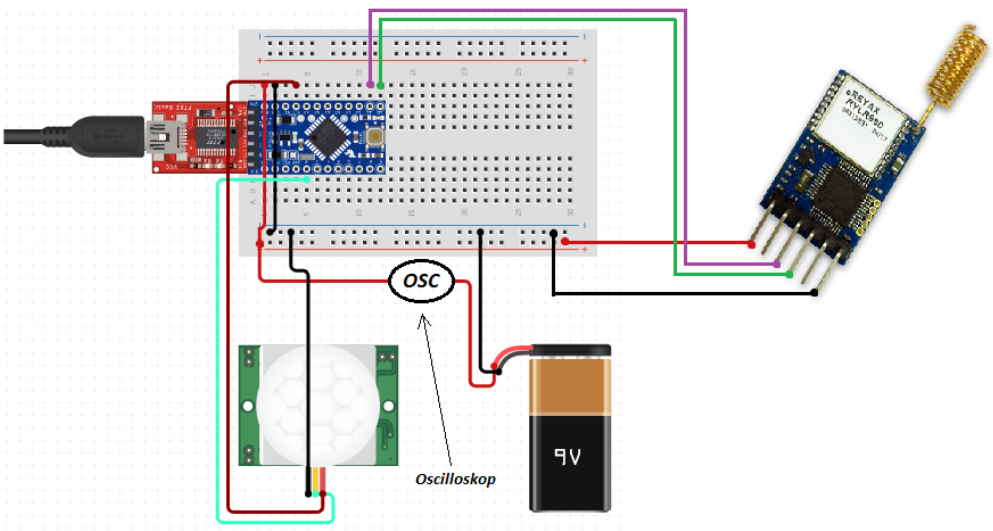
För att uppnå den tillräckligt höga minskningen blev fokus vänt från arduinon till modulerna. Den första modulen som blev undersökt var PIR-senorn men denna hade en statisk förbrukning på mindre än 50 uA. Reyax modulen troddes enbart förbruka när den tog emot eller sände data men detta visade sig vara fel. Den hade även en statisk förbrukning som gjorde att hela sensorenheten inte kunde nå målet strömförbrukningsmålet. Detta åtgärdades som nämnt med att en transistor kopplades mellan Reyax RYLR8XX och dess strömtillförsel för att kunna försätta den i strömsparläge under tiden att Arduino är i strömsparläge.

4

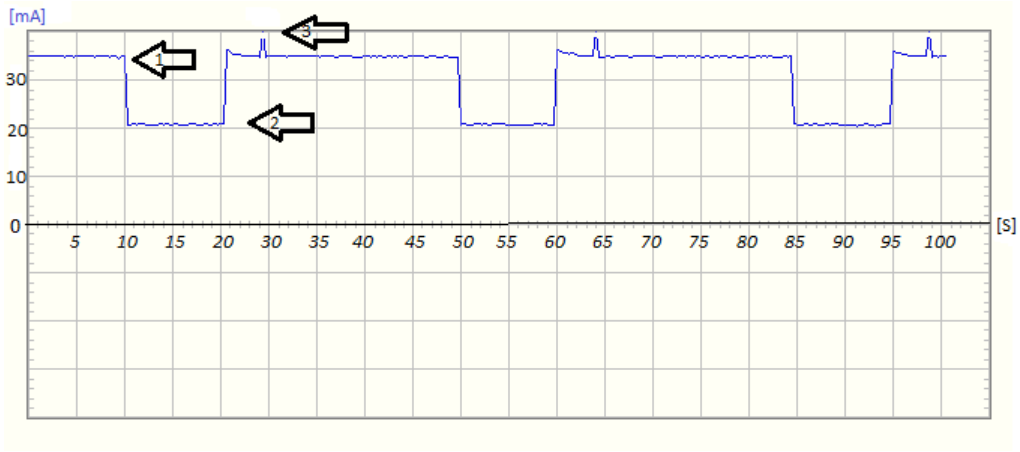
Resultat

4.1 Effektförbrukning

För att mäta hur hög effekt sensorenheten förbrukar som baslinje utan att vara anpassad eller modifierad placeras en resistor i serie med batteriet som driver hela sensorenheten, med en känd resistans på 100Ω. Efter detta anslöts ett oscilloskop över resistansen för att mäta strömmen som flödar genom kretsen under minst en hel cykel av att skicka och vänta på att skicka data. En hel skiss över hur alla moduler är inkopplade och vart oscillioskopet är inkopplat kan ses i figur 4.1.



Figur 4.1: Skiss över sensorenhet med oscilloskop anslutet



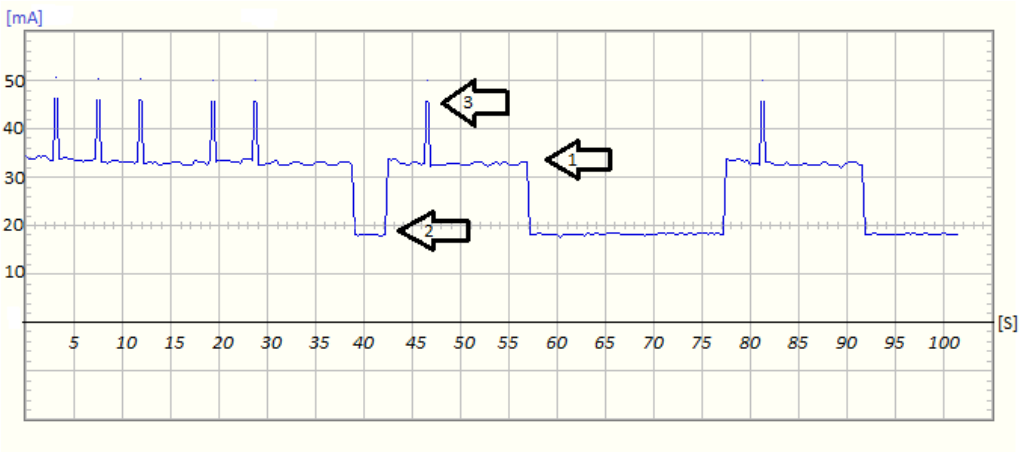
Figur 4.2: Mätning utan modifikation med sömnläge på arduino.

Ur figur 4.2 går det att se att det finns tre olika nivåer, den första nivån som enheten börjar på är när alla system är aktiva och när den väntar på att skicka. Den andra nivån är dalarna och detta är efter att den har skickats och arduinon försätter sitt mikroship i sömnläge för att spara energi. Den tredje nivån är Toppen som uppstår fem sekunder efter att microchipet vaknat och sensorenheten sänder sin data. Som det går att läsa ur denna mätningen ligger genomsnittet någonstans mellan 35mA och 28mA vilket är relativt högt jämfört med en verklig applicering där kretsen ska kunna vara online i över en månad. när arduinon går i sömnläge och där av enbart förbrukar ett enkelsiffrigt antal mA/h, kan det läsas ut att hela sensorenheten förbrukar 20 mA/h.

$$3000maH/(28ma/H) = 107,14H => 107.14/24 = 4,46Dygn.$$

Detta är inte i närheten av målet på minst 40 dagar som blev satt i början och det blev väldigt tydligt att strömförbrukningen behövde sänkas. Först och främst beräknades ett genomsnittligt mål av frörbrukningen att kunna förhålla sig till för framtida mätningar. För att uppnå målet att få prototypens batteri att hålla i minst 40 dagar måste det först bestämmas hur mycket kapacitet batterierna i prototypen ska ha. Efter att ha fått klara direktiv av företaget om att batterierna skulle vara cylindriska kom företaget fram till att två stycken 3.7V, 1500 maH fungerade bäst med företagets krav. Med detta kunde enheten ha en maxkapacitans på 3000 maH och med det kan det beräknas hur länge hela enheten kan vara online innan batterierna får slut på energi. Enligt det första mätvärdet med en genomsnittlig förbrukning på 28 ma/H går det att beräkna hur länge batterierna skulle hålla

$$24 * 40 = 960h => 3000/960 = 3.125mA/h$$



Figur 4.3: Mätning utan POWER LED med sömnläge på arduino.

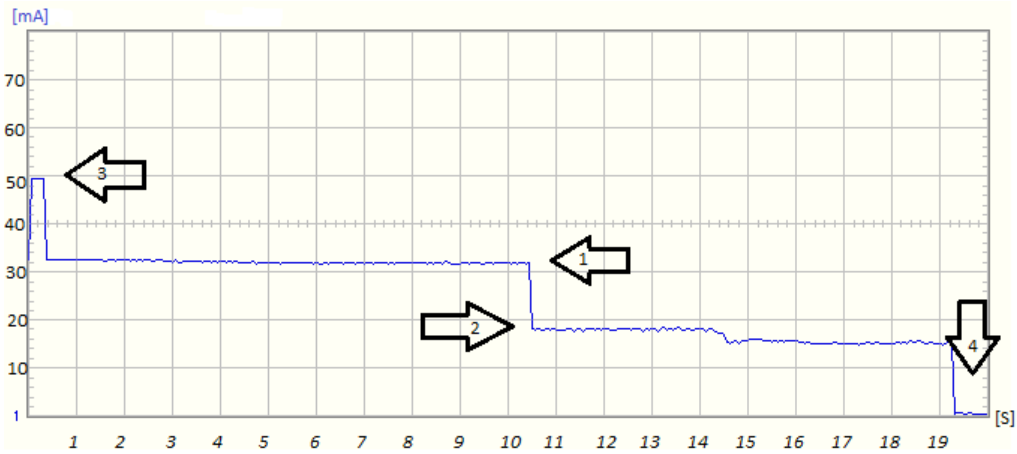
I figur 4.3 finns det tre olika strömnivåer och dessa tre representerar samma nivåer som i förklaring för figur 4.2, topparna är när sensorenheten sänder och i denna grafen går det att se hela magnituden av topparna. I denna grafen råkade det även följa med när sensorenheten gjorde setup vilket resulterade i dem många topparna i de första 30 sekunderna innan den börjar operera som vanligt. Den mellersta nivån som är när alla system är vakna men inget händer alltså när sensorenheten väntar på att få skicka. Den sista och lägsta nivån är när Arduinos microchip är i strömsparläge.

Det första sättet att skära ner på energiförbrukningen som testades och mättes, representerat i figur 4.3, var att koppla bort POWER LED:n från arduino kortet. Detta resulterade i en direkt minskning på ungefär 3 mA/h över hela grafen och kunde läsas av när man jämförde de tre nivåerna i figur 4.3 med figur 4.2

Efter POWER LED:n var urkopplad blev nästa steg att spara energi genom att minska förbrukningen hos Reyax-modulen. Den första lösningen som blev testad var att med hjälp av modulens egna mjukvara försätta den i ett strömspararläge men det resulterade tyvärr bara i en minskning på 9 mA/h. Detta var inte en tillräckligt hög minskning för att resultera i att sensorn inte ska behöva laddas på 10 dagar.

Den andra och slutgiltiga lösningen som blev vald att applicera var att använda en transistor för att koppla bort strömtillförseln till Reyax modulen helt och hållet. Detta resulterade i en dramatisk minskning av förbrukningen när hela sensorn går in i strömspararläge.

I figur 4.4 finns det fyra olika nivåer till skillnad från dem andra mätningarna och detta beror på att i denna infördes avstägningsfunktion av Reyax-modulen. Detta är även anledningen till att mätningen bara var 20 sekunder lång eftersom när mätningen gjordes hade inte koden för att starta modulen blivit implementerad ännu.



Figur 4.4: Mätning utan POWER LED med sömnläge på arduino och avstängd Reyax.

De första tre nivåerna i figur 4.4 representerar samma händelser som förklaringen för 4.2. Skillnaden är att i figur 4.4 tillkommer en till nivå, den fjärde nivån och detta är hur lågt förbrukningen sjunker när reyax-modulen och arduinon försätts i strömsparläge samtidigt. Det finns även en mindre nivåminskning i den andra nivån vid 14.5 sekunder, anledningen till denna minskning är okänd men kan har varit ett resultat av mänsklig faktor.

Som kan bli sett i den sista sekunden av figuren sjunker strömförbrukningen till ett så lågt värde som 0.2 mA/h. Detta värdet var väldigt lovande men betydde inte att sensorenheten hade en generell förbrukning på 0.2 mA/h eftersom den behöver vakna och sända data till huvudenheten. För att beräkna detta måste den aktiva förbrukningen och spikarna som uppstår medan ett meddelande sänds tas i åtanke.

Innan uträkningarna kan göras måste det nämnas igen hur ofta sensorenheten är aktiv och hur länge den sover. I dessa uträkningarna tas det hänsyn till att sensorenheten varannan minut vaknar i 10.5 sekunder för att skicka ett meddelande till huvudenheten detta betyder att varje "cykel" varar i två minuter. Under dessa två minuter så finns det en topp på 50 mA som varar i en halv sekund, en paus innan hela enheten somnar på 10 sekunder där den ligger på 34 mA och att den är i strömspararläge i resterande 109.5 sekunder med en förbrukning på 0.2 mA.

$$\text{Medelförbrukning} = 1/120s * (50mA * 0.5s + 34mA * 10s + 0.2mA * 109.5s) = 3,22mA$$

Denna uträkning gäller ifall sensorenheten skickar data varannan minut under ett helt dygn vilket inte speglar verkligheten då arbetstiderna inte omfattar hela dygnet. Sensorenheten behöver enbart skicka data under dygnets arbetstimmar alltså 8 av 24 timmar.

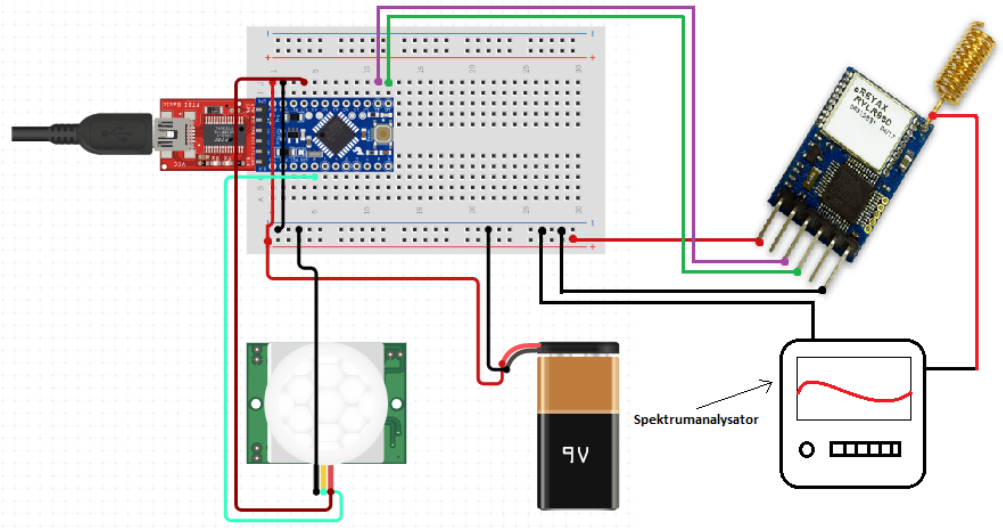
$$\text{Medelförbrukning under ett dygn} = 1/24h * (3.22mA * 8h + 0.2 * 16h) = 1.21mA/h$$

Med denna medelförbrukning uppfylls det satta målet eftersom 1.21 mA/h resulterar i att sensorenheten kan vara aktiv i

$$3000mA/1.21mA = 2479h \rightarrow 2479h/24h = 103Dygn$$

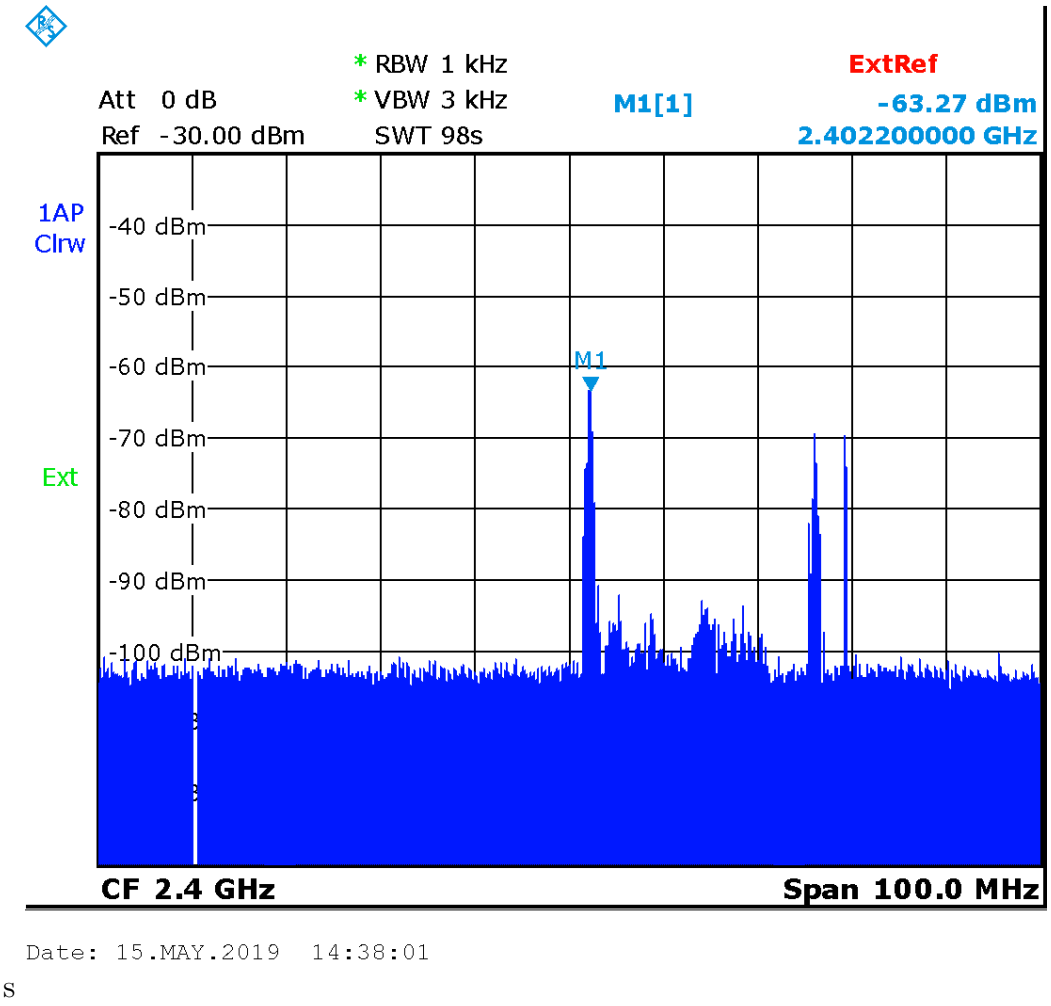
4.2 Spektrumanalys

Nästa mätningar som gjordes var på de olika generella spektrumen som används för att sända och ta emot data. Detta gjordes på samma sändtagare som används under hela projektet för att få en klar bild på hur det påverkar sensorenheten. Spektrumanalysatorn kopplades direkt på antennen och jordades mot Arduinon för att få en så verklighetstrogen bild på hur spektrumen såg ut och aggerade med sensorenheten. se figur 4.5



Figur 4.5: Illustrering av spektrummätning på sensorenhet.

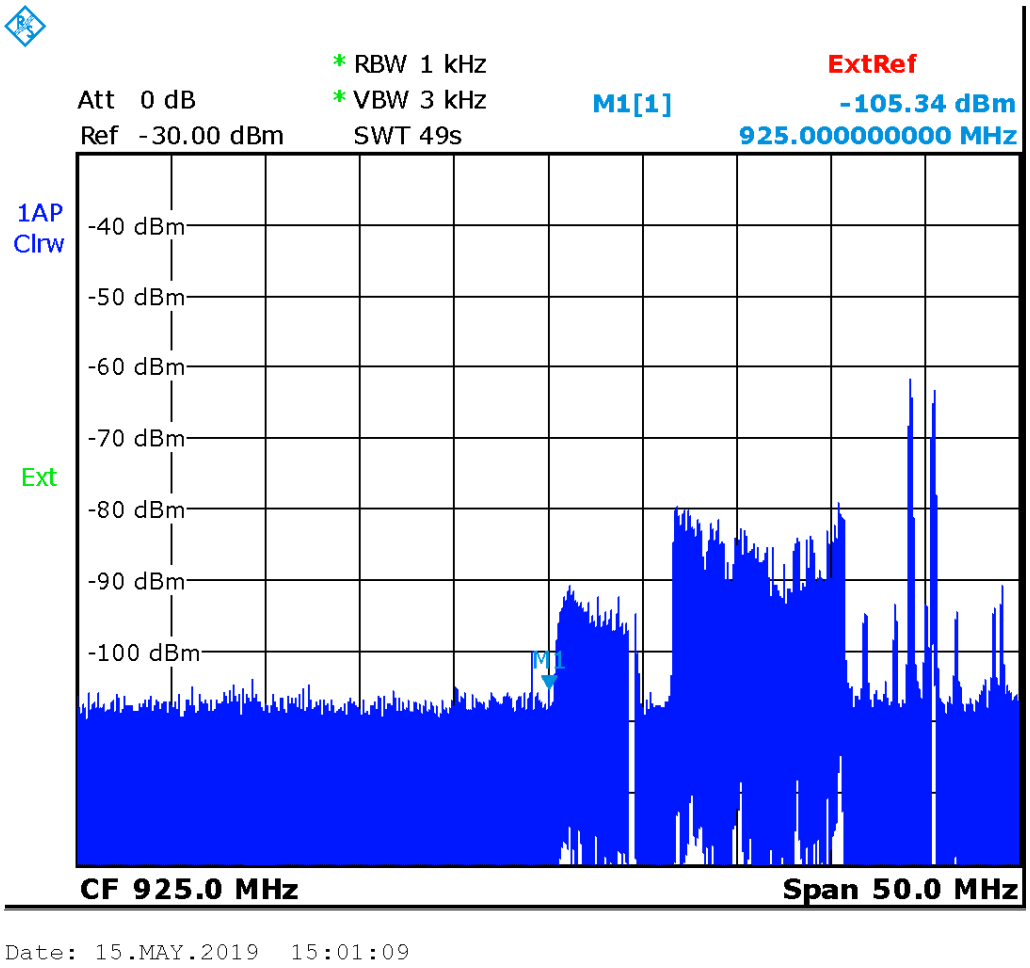
För att mäta ungefär hur mycket störningar och hur sändtagaren i kretsen påverkar och utsätts för i en vanlig kontorsmiljö gjordes det mätningar på ett labb med en Spektrumanalysator på Chalmers Tekniska högskola som omges av kontor och annan tillhörande utrustning. Det första som gjordes var att mäta hur dem olika använda spektrum såg ut, Wifi och GSM. För att se hur det skulle sett ut ifall det istället för 868MHz hade valts att sända över 2.4 GHz eller GSM nätet. Signaler motsvarar allt över -90 dBm i styrka och allt under det betraktas som bakgrundsbrus.



s

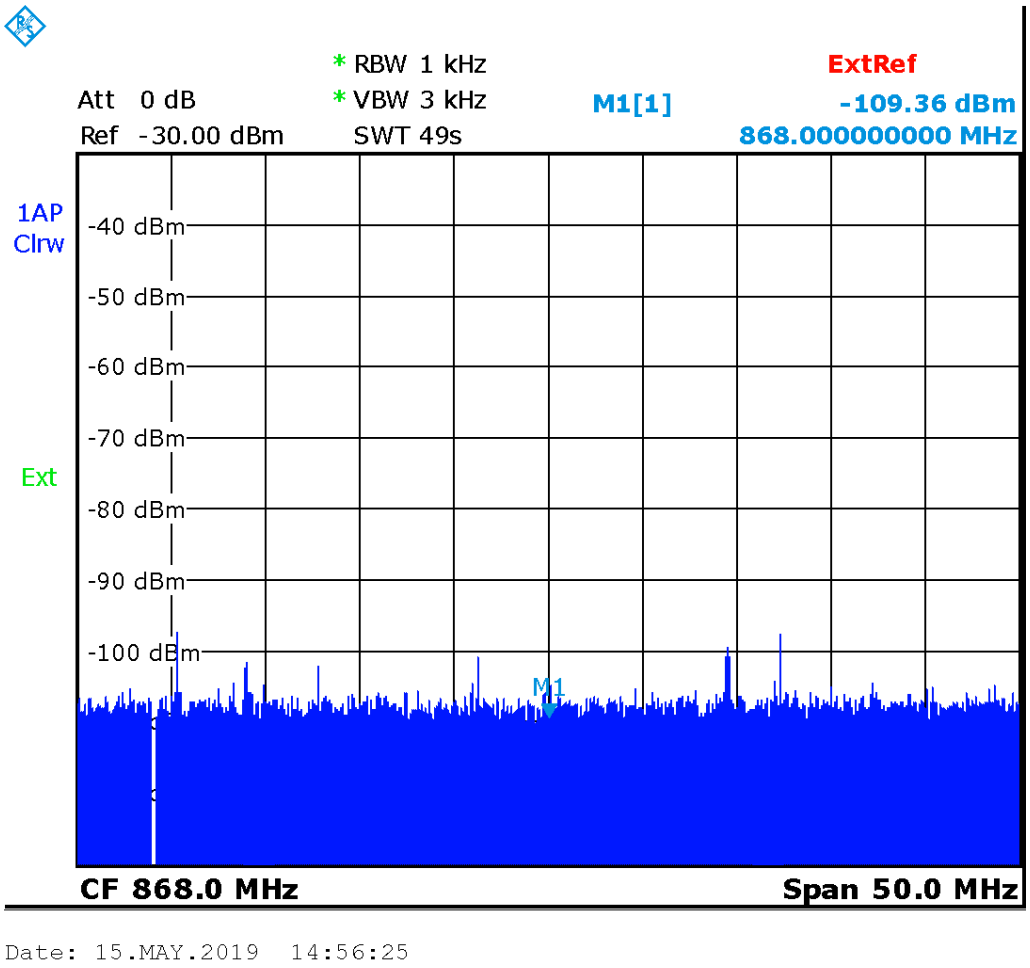
Figur 4.6: Signaler från 2.35 till 2.45 GHz

Det går klart och tydligt att se i figur 4.6 andra signaler utbredda från 2.4 till 2.43 GHz som i detta fall hade resulterat i risk för interferens. Denna mätningen upprepades ett flertal gånger för att se hur stor spridningen är på olika enheter i närheten. Det gick att se att dem olika signalerna var olika starka vid olika tillfällen och att det även dök upp signaler hela vägen upp till 2.45 GHz. Med denna information kunde det konstateras det gjordes rätt val att inte avgränsa sig till 2.4 GHz eftersom det var så mycket andra signaler på den frekvensen, där den högsta detekterade signalstyrkan var strax under -60dBm.
Nästa Mätning presenterad i figur 4.7 var för att se hur mycket störingar som ligger över GSM nätet som referens för hur telefoner skulle påverka sändtagaren. I Sverige finns det en typ av GSM som kan skapa interferens med våra signaler, GSM 900 som jobbar mellan 880 och 960 MHz. [17]



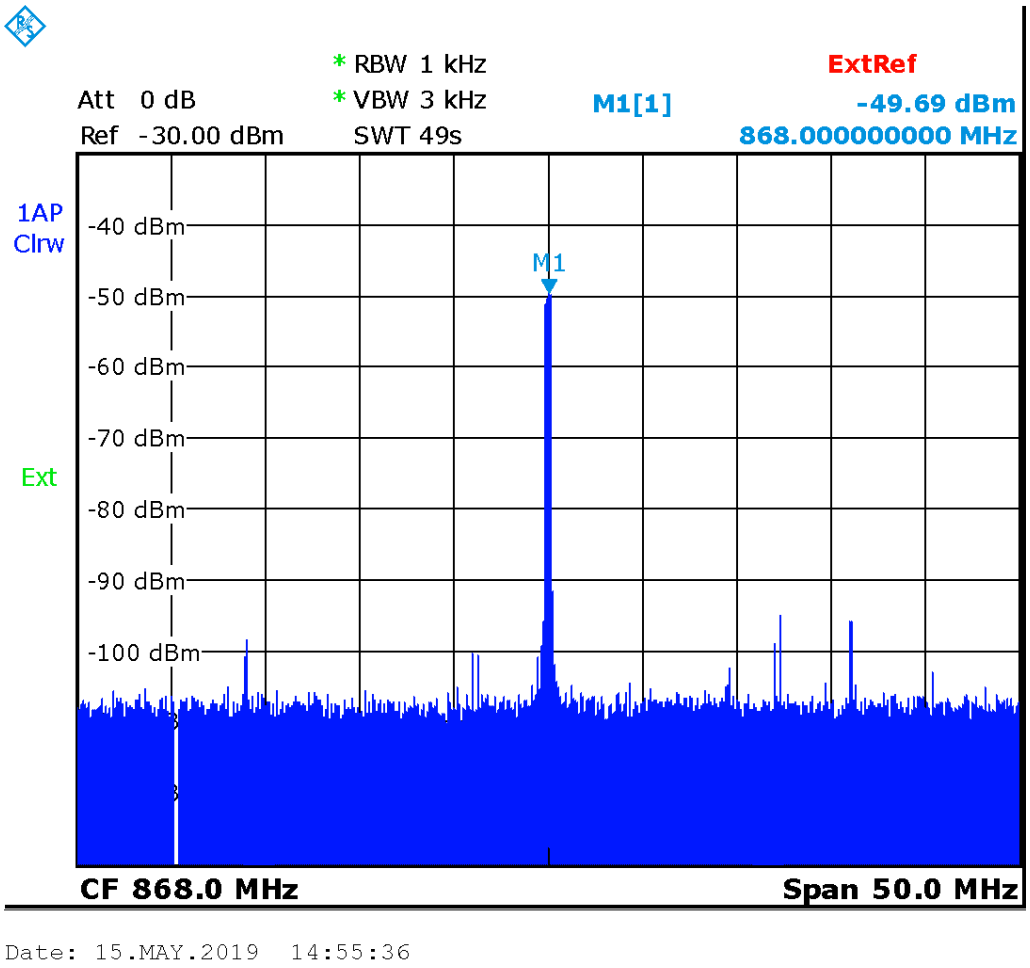
Figur 4.7: Telefonnätet GSM, 900-950 MHz

Som det kan ses ur grafen går det väldigt mycket trafik över telefonnätet i ett vanligt kontor. Dessa frekvenserna hade inte heller varit lämpliga att operera i, ifall det varit ett alternativ, eftersom det med säkerhet alltid kommer ligga störningar här och att det hade varit lättare att lyssna av eftersom mobiltelefoner tar emot dessa signaler. Den sista mätningen och mest relevanta är för den frekvens som sändtagaren som blev vald faktiskt opererar i. Denna frekvensen valdes eftersom det är öppet för allmänheten att använda utan licens för saker som radio eller liknande och är representerad i figur 4.8



Figur 4.8: 868 MHz utan signal från sändtagare, 843-893 MHz

Ur plotten, se figur 4.8 ser man att inga oväntade signaler tas upp av mottagaren utan endast bakgrundsbrus. Detta betyder att denna frekvensen är optimal då konstant interferens inte uppstår jämfört med 2.4GHz och 880MHz-960MHz som blivit presenterade i tidigare mätningar. Detta beror på att väldigt få system använder denna frekvens och störningar från utrustning runtomkring inte påverkar 868MHz. Ifall det gett utslag och det varit annan utrustning som sänt data över samma frekvens betyder inte det nödvändigtvis att sensorenheten plockat upp det som mer än störningar eftersom det med störst sannolikhet inte skickat med samma typ av kryptering. Men det hade självklart givit störningar och hade resulterat i att enheten fått skicka flera gånger för att vara säker att meddelandet kommit fram. Eftersom alla sensorenheter skickar tills dem får ett svar från huvvudenheten hade detta bara ökat fördröjningen mellan sensorenhet och huvudenhet.



Figur 4.9: 868 MHz utan signal från sändtagare, 843-893 MHz

Den tydliga spetsen i mitten av figur 4.9 är signalen från sändtagaren och är överlägset tydlig, med en signalstyrka på -50dBm, till allt annat trots att den ligger i andra änden av labbsalen och sänder. Meddelandena går fram snabbt och utifrån de tester som gjorts märks ingen interferens av från andra trådlösa enheter eftersom inga andra signaler blev detekterade på denna frekvensen.

5
Diskussion

Även om det färdiga systemet uppfyllde de krav som ställdes på det så finns det fortfarande väldigt mycket utrymme för vidare utveckling och förbättring av systemet. Vidare kan systemet utvecklas med en mikrokontroller där inställningarna är anpassade för just vårat system så att endast de nödvändiga funktionerna är aktiva så att strömförbrukningen kan fås ner ytterligare. Övergången från Arduino till eget kretskort kan enkelt göras utan att behöva programmera ett nytt system då det egna kretskortet kan baseras på samma mikrokontroller som Arduino använder, ATmega328P. Fördelen med att skapa ett eget kretskort blir då också att alla extra komponenter som finns på Arduino:s utvecklingskort kan tas bort så att endast de komponenter som vårat system behöver finns på kretskortet. Detta leder till ett mindre kretskort som är designat specifikt för vårt ändamål så att den färdiga slutprodukten inte blir så stor samtidigt som alla komponenter och eventuella moduler kan kopplas samman på ett och samma kretskort.
Efter att ha utfört spektrumanalys och jämfört 868MHz, som vårt system använder, med 2.4GHz [se figur 4.6] som används för både Wifi och bluetooth så kan vi konstatera att störningarna från kringliggande system minskar markant [se figur 4.8]. Vid 868MHz tas endast bakgrundsbrus upp om vårt system inte skickar data jämfört med 2.4GHz där det ständigt tas upp signaler från de kringliggande systemen och trådlösa nätverken. Att inte störa eller störas av kringliggande system var det primära målet vilket vi har åstadkommit med önskat resultat vilket kan ses man jämför bilderna från spektrumanalysatorn när systemet inte är igång jämfört [se figur 4.8] med när systemet är igång [se figur 4.9].
I dagsläget används endast rörelsesensorer för att samlar in data men om det i framtiden blir aktuellt att expandera utbudet av vilken typ av data som samlas in så kan rörelsesensorerna enkelt bytas ut till andra typer av sensorer utan att systemet behöver programmeras om.

6
Slutsats

Vi har fått ner den totala effektförbrukningen för systemet så mycket som det varit möjligt med de nuvarande komponenterna genom att koppla bort POWER LED på Arduino, sätta Arduino i strömsparläge när den inte behöver vara aktiv och i samband med att Arduino går i strömsparläge så bryts strömförsörjningen till Reyax RYLR8XX med hjälp av en extern transistor för att på så sätt nå målet att en sensorenhet ska kunna drivas utav batterier under minst 40 dagar. Vid spektrumanalys så har vi också konstaterat att den frekvens vi valt för trådlös överföring, 868MHz, används mycket mindre jämfört med 2.4GHz där spektrumananlysatorn visade att signaler plockades upp från de kringliggande systemen. På frekvensen 868MHz kunde vi konstatera att det endast var bakgrundsbrus som plockades upp av vår mottagare då mätningen gjordes.

Referenser

[1] "wi-fi alliance." https://www.wi-fi.org.

[2] "radio versions | bluetooth technology website." https://www.bluetooth.com/bluetooth-technology/radio-versions/.

[3] "post- och telestyrelsens allmänna råd (ptsfs 2015:3) om den svenska frekvensplanen." https://pts.se/globalassets/startpage/dokument/legala-dokument/foreskrifter/radio/ptsfs-2015_3-allmanna-rad-frekvensplanen.pdf, 2019.

[4] "what is lora? | semtech lora technology." https://www.semtech.com/lora/what-is-lora, 2019.

[5] "zigbee alliance." https://www.zigbee.org.

[6] "lora at command guide." https://www.ne.se/uppslagsverk/encyklopedi/lång/ohms-lag, 2018.

[7] "Arduino uno product page." https://store.arduino.cc/arduino-uno-rev3, 2019.

[8] "Arduino introduction." https://www.arduino.cc/en/Guide/Introduction, 2019.

[9] "Atmega328p datasheet." https://www.sparkfun.com/datasheets/Components/SMD/ATMega328.pdf.

[10] "Reyax rylr896." http://reyax.com/products/rylr896/, 2019.

[11] H. L. B. Optical Coating Laboratory, "Infrared intrusion detector system," 1972.

[12] "hc-sr501 pir sensor." https://www.lawicel-shop.se/hc-sr501-pir-sensor, 2019.

[13] "oscilloscope fundamentals." http://www.rohde-schwarz-usa.com/rs/rohdeschwarz/images/Oscilloscope-Fundamentals_v1.1.pdf.

[14] "Ohms lag." https://www.ne.se/uppslagsverk/encyklopedi/lång/ohms-lag.

[15] "test measurement catalog 2019." https://scdn.rohde-schwarz.com/ur/pws/dl_downloads/dl_common_library/dl_brochures_and_datasheets/pdf_1/Test_and_Measurement_Cat2019_cat_en_5213-7590-42_v0800_120dpi.pdf, 2019.

[16] "Arduino low power - how to run for a year on coin cell battery." http://www.home-automation-community.com/arduino-low-power-how-to-run-atmega328p-for-a-year-on-coin-cell-battery/, 2019.

[17] "Frekvensbad telefoni." https://www.induo.com/s/g/gsm-3g-4g-frekvensband/, 2019.

A

Appendix

```
int send_data(String string_to_send, int address){
  if((millis() - last_transmission) >= send_data_interval){ //send_data_interval = 3000 (3s)

    LoraSerial.println((String)"AT+SEND="+address+","+string_to_send.length()+","+string_to_send); // AT-kommando för att skicka data
    wait_lora_ret();                                                                               // Vänta på svar från LoRa modul

    last_transmission = millis();
```

Figur A.1: Funktion för att skicka data

```
void serial_event(){
  LoraSerial.listen();

  /* Läs data från LoRa modul och spara i recieve_string */
  while(LoraSerial.available()){
    char in_char = (char)LoraSerial.read();

    recieve_string += in_char;

    if(in_char == '\n'){
      recieve_complete = true;
    }
  }
}
```

Figur A.2: Funktion för att ta emot data

```
int init_sensor(String str){
  // Första strängen som tas emot är huvudenhetens adress
  if(main_unit_addr == 0){
     main_unit_addr = str.toInt();
  // Andra strängen som tas emot är sensorns unika adress
  }else{
    sensor_num = str.toInt();
    int sensor_id = sensor_num + main_unit_addr;
    if(sensor_num != 0){
      LoraSerial.println((String)"AT+ADDRESS="+ sensor_id);
      wait_lora_ret();
  // Slutför initiering om sensorns unika adress
  // har uppdaterats korrekt
      if(get_address() > 0){
        add_to_buffer_last("init done");
        init_setup = false;
      }
    }
  }
  return 0;
}
```

Figur A.3: Ta emot och spara unik adress och huvudenhetens adress (kod från sensorenhet)

```
int init_sensor(String str){
  // Första strängen som tas emot är huvudenhetens adress
  if(main_unit_addr == 0){
     main_unit_addr = str.toInt();
  // Andra strängen som tas emot är sensorns unika adress
  }else{
    sensor_num = str.toInt();
    int sensor_id = sensor_num + main_unit_addr;
    if(sensor_num != 0){
      LoraSerial.println((String)"AT+ADDRESS="+ sensor_id);
      wait_lora_ret();
  // Slutför initiering om sensorns unika adress
  // har uppdaterats korrekt
      if(get_address() > 0){
        add_to_buffer_last("init done");
        init_setup = false;
      }
    }
  }
  return 0;
}
```

Figur A.4: Ta emot och spara unik adress och huvudenhetens adress (kod från sensorenhet)

```
  /* Send data */
  if(data_buffer[0] != ""){
    send_data(data_buffer[0], 0);
  }else if(fetch_active == true){
    if(sensor_num > 1){
      send_data((String)"fetch(" + sensor_num + ")", main_unit_addr + fetch_sensor_num);
    }
  }

  /* Fetch 120 sekunder efter föregående fetch */
  if(init_active == false && fetch_active == false){
    if((millis() - fetch_end_millis) >= fetch_data_interval){ //fetch_end_millis är den tid
      fetch_active = true;                                   //när senaste fetch slutfördes
    }
  }
```

Figur A.5: Starta hämtning 2 minuter efter att föregående avslutats (kod från huvudenhet)

```
void going_to_sleep(){

  // Beräkna antal 8 sek intervall som arduino ska vara i strömspararläge
  int n = round((110 + (3*(active_sensors - 1)))/8);

  for(int i = 0; i < n; i++){
    LowPower.powerStandby(SLEEP_8S, ADC_OFF, BOD_OFF);
  }

  // Slå på transistor för att aktivera LoRa modul
  digitalWrite(loraPower, HIGH);
}
```

Figur A.6: Beräkna den tid som sensorenhet ska vara i strömspararläge