Datalogger för system på vattennätet

SEBASTIAN LINDGREN
BACHELOR´S THESIS
DEPARTMENT OF ELECTRICAL AND INFORMATION TECHNOLOGY |
FACULTY OF ENGINEERING | LTH | LUND UNIVERSITY

Datalogger för system på vattennätet

Av

Sebastian Lindgren

Department of Electrical and Information Technology
Faculty of Engineering, LTH, Lund University
SE-221 00 Lund, Sweden

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 2019

Sammanfattning

I dagens industri finns en önskan om att samla in data från många typer av anläggningar och installationer. Denna data används generellt för att övervaka och utveckla, men det är inte alltid lätt att hitta system som erbjuder precis vad som önskas.

Detta examensarbete har undersökt en skräddarsydd lösning för att samla och presentera data, i samarbete med "Hässleholms Vatten AB". Det har ingått tre olika givare i arbetet; en regngivare, en nivågivare och en flödesgivare, men endast regngivaren har presenterats i lösningen.

Mätdata från regngivare kan användas till att dimensionera ledningar för att omhänderta vatten i tätorterna. Sedan kan denna mätdata även användas som grund för eventuella försäkringsärenden.

Under arbetet har en prototyp av en datalogger för att skicka mätdata till en server utvecklats, där data blir åtkomlig enligt ett önskat format. Detta samtidigt som den saknar trådbunden förbindelse för både elektricitet och kommunikation. Dessa förutsättningar har lett till användning av solceller för att producera elektricitet och 3G-nätet för kommunikation. Arbetet har även omfattat en enkel server för att kommunicera med dataloggern, samt ett användargränssnitt för att presentera mätdata.

Funktionalitet och användbarhet hos prototypen har prioriterats under arbetets gång. Detta har lett till att aspekter som hållbarhet, energiförbrukning, kostnad och enkelhet har styrt arbetets utveckling.

Nyckelord: Datalogger, Givare, Server, Visualisering, Mikrokontroller

Abstract

In today's industry there is a desire to collect data from different types of facilities and installations. This data is generally used for the purpose of monitoring and developing, but it is not always easy to find a system that offers exactly what is desired.

This bachelor thesis has examined a tailor-made solution for collecting and presenting data in collaboration with "Hässleholms Vatten AB". It has included three different sensors; a rain sensor, a level sensor and a flow sensor, but only the rain sensor has been implemented in the solution.

Data from rain sensors can be used to determine the size of pipes which is used for diverting water in urban areas. This data can also be used as a basis for insurance matters.

This assignment has consisted of the development of a prototype, for sending data to a server, where it becomes accessible according to a desired format. This, at the same time as it lacks a wired connection for both electricity and communication. These conditions have led to the use of solar cells for producing electricity and the 3G-network for communication. The thesis has also consisted of a simple server for communication with the logger, and a user interface for data accessibility.

Functionality and usability of the prototype have been prioritized during this assignment. This has led to aspects such as sustainability, energy consumption and cost controlling the development.

Keywords: Datalogger, Sensor, Server, Visualization, Microcontroller

Innehållsförteckning

1. Inledning

Inledningen beskriver examensarbetet uppdelat i sex olika delar; bakgrund, syfte, målformulering, problemformulering, motivering och avgränsningar. Det är ur dessa delar som både uppgiften och lösningen har tagits fram.

1.1. Bakgrund

Examensarbetet är utfört i samarbete med Hässleholms Vatten AB, som är etablerat i Hässleholm, Skåne. Företaget ansvarar för ett vidsträckt vattennät som täcker Hässleholms kommun, varifrån kunder erbjuds lösningar för renvatten, spillvatten och mera. Företagets yttersta mål är leverans av renvatten till kunder, rening av spillvatten från kunder och avlägsning av dagvatten från tätorterna. Hässleholms Vatten kan anses vara en viktig del av infrastrukturen i samhället då de har en omfattande insyn i hur olika vatten från både vattennätet och naturen kan påverka både tätorten och glesbygden.

Uppdragsgivaren ville undersöka och utveckla en prototyplösning för att logga data från sina givare på vattennätet. Givare som idag saknar både trådbunden förbindelse för elektricitet och kommunikation.

För en regnmätare sker drift idag över ett batteri och all data loggas lokalt på plats. Detta kräver manuell hämtning av data och utbyte av batteri. Formatet all data loggas i är ett medelvärde över en viss tid vilket gör den mindre användbar. Detta har skapat en tidskrävande arbetsbörda som ger ett oönskat resultat.

Förutom regnmätaren omfattar uppgiften även en nivågivare för spillvattenbrunnar och flödesgivare för renvattenledningar.

1.2. Syfte

Examensarbetets syfte har varit att undersöka tre olika givare för utveckling av en prototyplösning. En prototyplösning bestående av insamling och presentation av önskad mätdata från olika punkter i vattennätet.

1.3. Målformulering

Målet är en prototyp för ett komplett system som gör det möjligt för en användare att hämta all mätdata från specifika mätpunkter genom sin arbetsdator. Detta genom att utveckla en datalogger, en server och ett användargränssnitt. Målet omfattar en hållbar och underhållsfri lösning som fungerar i en krävande miljö över en obestämd tid.

1.4. Problemformulering

Examensarbetet har undersökt ett antal problem, där problemformuleringen har utgått från problemen i syftet och målformuleringen.

- Hur ska data loggas? Det vill säga vilken typ av hårdvara och mjukvara kan användas för att läsa av givaren och skapa trådlös kommunikation.
- På vilket sätt kan hårdvaran försörjas med elektricitet? Med batteri och solceller?
- Hur ska kommunikation mellan dataloggern (klienten) och servern ske?
- Hur kan en server utvecklas för kommunikation med en eller flera klienter samtidigt?
- Vilken data ska presenteras och på vilket sätt?

1.5. Motivering av examensarbete

Valet av examensarbete skedde i samband med vilken typ av uppgift som erbjöds, hur väl denna uppgift kunde utföras som ett examensarbete och ett bakomliggande intresse för uppgiften och företaget. Utöver detta spelade det stor roll var företaget var etablerat och hur god kommunikationen till företaget var.

1.6. Avgränsningar

Arbetet har omfattat utveckling av en prototyp för Hässleholms Vatten. Utvecklingen har skett utan tillgång till företagets interna server, därför är mjukvaran utvecklad för att vara kompatibel med bland annat windows 7, 8 och 10. Programvara som krävs är den senaste versionen av Java 8 och en webbserver. Till servern krävs det att en ej upptagen port öppnas, medan serverns port och adress behöver anges för klienten.

Lösningen har ej integrerats för företagets egna system, därför har ett “open source” användargränssnitt med en webbserver använts för att presentera data. Observera att examensarbetet inte har omfattat utveckling av dessa ramverk, endast användning och modifiering av dem.

Hårdvara och material för arbetet har valts utifrån funktion och dess specifikation, men ej testats utifrån alla specifikationerna. Lösningen som har undersökts för examensarbetet har omfattat en regnmätare och till viss del en nivågivare.

2. Teknisk bakgrund

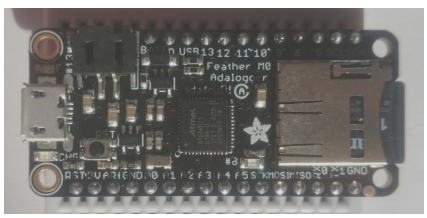
Detta kapitel beskriver de tekniska metoder och tekniker som har använts, samt valet av hårdvara för examensarbetet.

2.1. Hårdvara

För prototypen har olika typer av hårdvara testats beroende på funktion, specifikation och utvecklingsmöjligheter. Mer hårdvara har testats men har inte ingått i lösningen. Valet av komponenter har skett i samband med tester och undersökning utifrån olika källor. Dessa källor är till specifikationer för hårdvara eller till produktbeskrivning.

2.1.1. Mikrokontroller

Kontrollenheten som har använts i examensarbetet är mikrokontrollern "Adafruit Feather M0 Adalogger" enligt figur 1 från "Adafruit Industries". Enheten är enkeltrådad och programmerbar med "Arduinos IDE" eller "CircuitPython" som liknar språken C/C++ och Python. Enheten använder sig av ett mikrochip "ATSAMD21G18 ARM Cortex M0" som arbetar med en högre frekvens samt har mer än flerdubbelt FLASH och RAM än liknande arduino enheter. På enheten finns flera digitala och analoga ingångar. Enheten har även stöd för kommunikationsprotokollen I^2C och SPI som skickar data med hjälp av en genererad klocksignal.

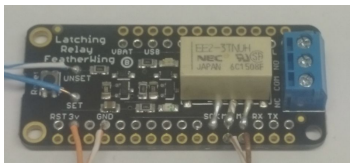


Figur. 1. Mikrokontroller, Feather M0 Adalogger [1]

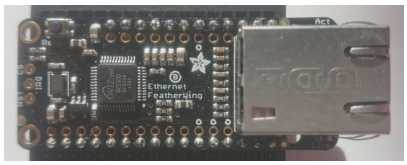
Hårdvara för SD kort finns även integrerad på enheten, vilket tillåter läsning och skrivning till ett SD kort. SD-kortet som har använts med mikrokontrollern är ett "Xmore industrial microSD" [9]. Detta kort är främst anpassat för industri då det under sin livstid ska klara en krävande miljö.

2.1.2. Utvecklingskort

Utvecklingskort eller “featherwings” som de även kallas, är kort med olika kretsar och funktioner, anpassade för att fungera med mikrokontrollern. De som har använts är “Mini Relay FeatherWing” i figur 2 och “Ethernet FeatherWing” i figur 3. Det första kortet består av ett spärrelä medan det andra kortet är konstruerat för Ethernet-anslutning. Ethernet-kortet använder sig av ett “WIZ 5500” mikrochip, detta chip tillåter bland annat TCP-kommunikation över en Ethernet-anslutning.



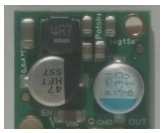
Figur. 2. Relä-kort, Latching Relay FeatherWing [2]



Figur. 3. Ethernet-kort, Ethernet FeatherWing [3]

2.1.3. Variabel spänningsregulator

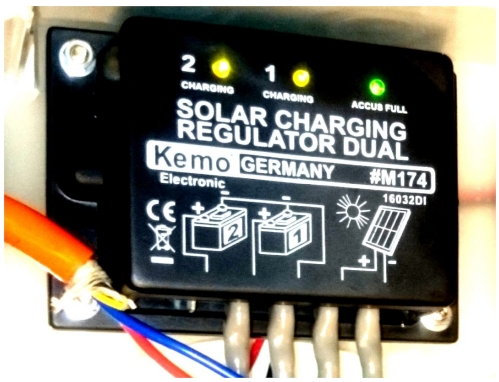
En variabel spänningsregulator är en step-down regulator som skapar en konstant utspänning med en variabel inspänning. Regulatorn i figur 4 har en vilande strömförbrukning och arbetar med en viss effektivitet beroende på vilken spänning och ström som levereras. Regulatorn har omvänt spänningsskydd vilket skyddar enheten om strömmen skulle anslutas från regulatorns utgång. Utöver detta finns skydd för kortslutning och värme där enheten stängs av vid en viss temperatur.



Figur. 4. Variabel spänningsregulator, D24V25F5 [4]

2.1.4. Laddningsregulator

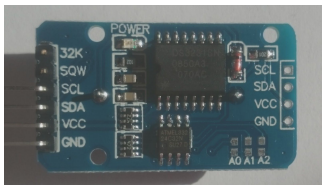
En laddningsregulator enligt figur 5 tillåter laddning av olika batteri från solceller. Regulatorn har en vilande strömförbrukning och arbetar med en viss effektivitet. Regulatorn är konstruerad med ett omvänt spänningsskydd och ett överladdningsskydd för anslutna batteri. Vid höga effekter krävs säkring för lasten och värmeledande bakstycke på grund av hög värmebildning.



Figur. 5. Laddningsregulatorn, M174 [5]

2.1.5. RTC

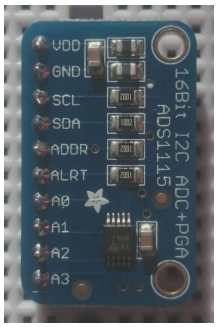
Realtidsklockan i figur 6 består av ett DS3231 chip, som håller en tid med en hög precision. Chipet har en inbyggd temperatursensor för temperaturkompensering och kan drivas externt eller med möjlighet för egen batteridrift med en inbyggd laddningskrets. Kommunikation till klockan sker med I^2C protokollet och har enligt specifikation en avvikelse på en minut per år.



Figur. 6. RTC, DS3231 [6]

2.1.6. ADC

En ADC eller analog till digital omvandlare är en krets vars uppgift är att läsa av en analog spänning och översätta denna till en digital signal, vilket gör avläsning för en kontrollenhet möjlig. Omvandlare finns i olika upplösningar, den 16-bitars omvandlare som användes konverterar en analog signal till en 2^{15}-bitars digital signal. Om denna omvandlare har en maxspänning på 6.144 V ger det teoretiskt en precision på $6.144V/2^{15} bitar = 0.1875\ mV$. Figur 7 visar en ADC med fyra ingångar för avläsning av fyra olika analoga signaler med gemensam jord, eller för avläsning av två differentiella signaler.



Figur. 7. ADC, ADS1115 [7]

2.1.7. 3G-router

En 3G-router är en router som skapar en anslutning till internet via 3G-nätet. För denna anslutning krävs ett SIM-kort från en valfri operatör, 3G-täckning från denna operatör och elektricitet. Routern som har använts är en UR5I [8], detta är en router framtagen för en industriellt krävande miljö.

2.1.8. Batteri

Batteriet i lösningen är ett blyackumulatorbatteri "RT 1223" [10] från Ritar. Batteriet har en livslängd på 6-8 år och är märkt; 2.3Ah, 12V.

2.1.9. Solceller

Solpanelerna i lösningen är av typ polykristallina. Denna typ av solcell kan maximalt omvandla 19 procent av solens energi till elektricitet. Solceller av polykristallina har oftast en livslängd på minst 25 år och under sin livslängd ett effektfall på ca 20 procent. Solcellerna i lösningen, “SOL10P” [11] från Velleman, är märkta; 10W, 12V.

2.2. IP-klassning

Kapslingen som har valts till dataloggern har valts utifrån hållbarhet. Kapslingen användes för att skydda elektroniken i dataloggern från krävande miljöer, främst skydd mot fukt. Kapslingen i lösningen är märkt för IP67, där sexan respektive sjuan anger att kapslingen är dammtät och vattentät.

2.3. Givare

De givare som har undersökts för implementation under arbetets gång är en flödesgivare, en nivågivare och en regngivare.

2.3.1. Flödesgivare

En turbinflödesmätare [12], bestående av en magnetresistiv sensor och en propeller. Magneter fanns placerade i bladen hos propellern vilket tillät avläsning som kunde tolkas som en frekvens vid rotation. Denna frekvens representerade hur snabbt magneter passerade sensorn vilket kunde översättas till en rotation och genom det, ett flöde.

2.3.2. Nivågivare

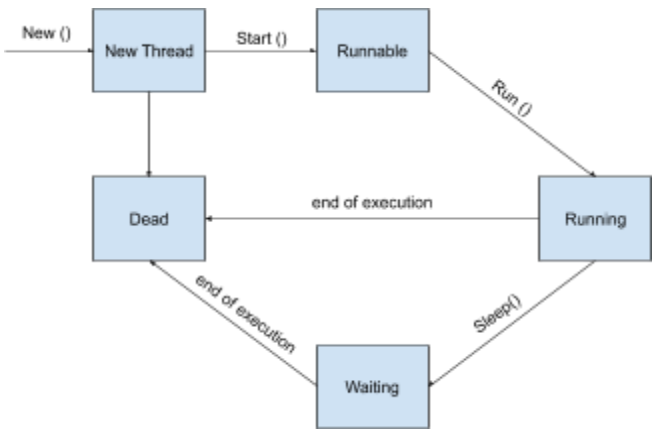
En nedsänkbar givare [13] med en piezoresistiv trycksensor med atmosfärskompensation. Givaren kunde ge två olika utsignaler, en digital datasignal eller en 4 till 20 mA analog signal. Enheten fanns i flera varianter, men den i examensarbetet hade ett mätområde mellan 0 och 3 meter, där den analoga utsignalen var direkt proportionell med mätvärdet.

2.3.3. Regngivare

En digital regnmätare [14] bestående av en vippa som registrerade en massa. Där massan var direkt proportionell med mängden regn som hade fallit över mätaren, detta då den var byggd med en bestämd uppsamlingsyta. Givaren bestod av två kretsar, en normalt öppen och en normalt stängd krets, som representerade de olika lägena. Var mätpuls (vippning) registrerade 0.2 mm eller 4 gram regnvatten och varade strax under 0.5 sekunder.

2.4. Java multitrådning

Java multitrådning är en teknik för att tråda program i Java. Trådad programmering används för att kunna exekvera flera instanser av kod på samma gång, detta för att uppnå en önskad funktionalitet. Exekvering av en tråd i Java kan beskrivas enligt en cykel som illustreras av figur 8 [15].



Figur. 8. Livscykel för en tråd i JAVA

- "New Thread" skapar och initierar en ny tråd.
- När tråden har skapats övergår den till "Runnable" där den förbereds för exekvering.
- När förberedelserna för exekveringen är färdiga utför tråden sin uppgift i "Running".
- Trådens uppgift kan vara fördröjt vilket tvingar den att stanna i ett viloläge "Waiting", men när tråden har utfört sin uppgift och exekverat färdigt kommer den att dekonstrueras och dö ut i "Dead".

2.5. Server socket

Server socket eller “WebSocket” är ett kommunikationsprotokoll som tillåter en direkt tvåvägskommunikation mellan server och klient. WebSocket är en öppen anslutning och utnyttjar TCP protokollet (transmission control protocol) för att skicka okrypterad data i realtid. TCP kommunikation via socket beskrivs i boken “Computer Networking” [16].

2.6. JSON

JSON (JavaScript Object Notation) [17], är ett dataformat och en standard som tillåter en lätt och användbar struktur på lagring och användning av data. Dataformatet är baserat på programspråket JavaScript och är utvecklat sedan början av 2000 talet . Ett mindre exempel visas enligt figur 9 som är taget från ett testprogram utvecklat för arbetet.


```
[
    {"01-03-2019 13:19:54":["0123456789","0123456789","0123456789","0123456789"]},
    {"01-03-2019 13:20:11":["0123456789","0123456789"]},
    {"01-03-2019 13:20:20":["0123456789","0123456789","0123456789","0123456789"]},
    {"01-03-2019 13:20:29":["0123456789","0123456789"]},
    {"01-03-2019 13:20:34":["0123456789","0123456789","0123456789","0123456789"]},
    {"01-03-2019 13:21:54":["0123456789","0123456789"]}
]
```


Figur. 9. JSON dataformat, exempel

2.7. Java - JFrame

JFrame är ett grundläggande grafiskt användargränssnitt i Java. Gränssnittet används för att skapa grafiska fönster med text, knappar och andra funktioner. Boken “Objektorienterad Programmering” tar upp flera exempel för användning och implementering av JFrame [18].

2.8. Visualisering

Visualiseringen av data för användaren är ett enkelt webbgränssnitt baserat på flera “open source” ramverk, tillsammans med en webbserver.

2.8.1. PHP - codeigniter

PHP codeigniter [19] är ett ramverk för webbapplikationer som är byggt för utvecklare. Ramverket består av flera exempel och bibliotek för att enkelt kunna utveckla webbapplikationer.

2.8.2. JavaScript - jQuery

jQuery [20] är ett javaScript bibliotek som är byggt för utveckling av webbapplikationer. Biblioteket kan användas för hantering och manipulering av dokument.

2.8.3. AdminLTE

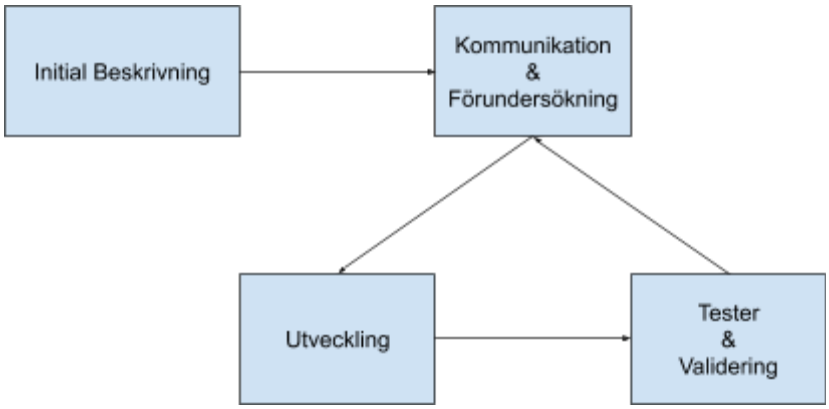
AdminLTE [21] är ett färdigt ramverk för en instrumentbräda till en webbapplikation. En instrumentbräda, eller “dashboard” är ett gränssnitt för att presentera data, i detta fall för en webbläsare.

2.8.4. WampServer

WampServer [22] är en webbserverapplikation för windows som kan användas för att utveckla webbapplikationer. Wamp står för och tillåter utveckling med Windows, Apache, MySQL och PHP. Wampserver är ett gratis verktyg för webbutvecklare.

3. Metod

Detta kapitel beskriver hur examensarbetet delades upp. De olika faserna omfattade den initiala beskrivningen, kommunikation till företaget, undersökning, utveckling av arbetet samt tester och validering för lösningen. Med få undantag var detta följden arbetet tog, vilket presenteras i figur 10.



Figur. 10. Arbetsschema

Ett av undantagen har varit det webbaserade användargränssnittet. Det var från början inget som skulle ingå i lösningen då arbetet skulle implementeras med företagets egna system. Därför är detta något som har tillkommit efter den initiala beskrivningen men som fortfarande har följt samma arbetsmetod.

3.1. Initial beskrivning

Den första delen av examensarbetet var den initiala beskrivningen. Denna skrevs utifrån uppgiften som Hässleholms Vatten ville ha en lösning för. Uppgiften var till en början väldigt vag där den endast beskrev flera problem som skulle lösas, inte hur eller på vilket sätt, dock med förslag. Den initiala beskrivningen skapade en konkret uppgift med en problembeskrivning och satte ett mål för examensarbetet. Det var efter att denna beskrivningen blivit godkänd av handledare Mats Lilja och examinator Christian Nyberg, som arbetet påbörjades.

3.2. Kommunikation

Kommunikation med företaget har förts efterhand som examensarbetet har utvecklats. Att föra samtal och diskutera har varit en viktig del då en tydlig beskrivning av målet har förenklat arbetsprocessen. Kommunikation fördes med en kontaktperson från företagets tekniska avdelning. Ett par möten hölls med ett par veckors mellanrum för att diskutera arbetets framgång och eventuella problem och frågor som hade uppstått. Denna kommunikation skapade en klar bild av målet efterhand som uppgiften utvecklades. Efter att den tekniska delen av arbetet var avklarad hölls samtal med en kontaktperson från företagets administrativa avdelning för att ta reda på hur insamlad data skulle presenteras. Det är utifrån dessa samtal som funktionen hos det webbaserade användargränssnittet har tagits fram, vilket visas i appendix A.

3.3. Förundersökning

Efter att problembeskrivning och ett mål fastställts, skedde en förundersökning av hur målet skulle uppnås. Här låg störst fokus på inlärning med syftet att skapa en struktur över hur arbetet skulle utföras, för att nå målet. Utgångspunkten för arbetet var kunskap som ackumulerats under utbildningen. Därför granskades först studielitteratur som ansågs vara lämplig för examensarbetet. Litteraturen som främst eftersöktes var för programmering, kommunikation och elektriska system. För att skapa struktur till en server med kommunikation över internet användes böcker som "*Computer Networking*"[16] och "*Objektorienterad Programmering*"[18]. Dessa böcker bidrog till en ökad förståelse och användning av kommunikationsprotokoll och grafiska ramverk för java programmering. För att undersöka elektriska komponenter och kretsar lämpliga för arbetet har en encyklopedi använts. De två första volymerna för "Electronic Components" [23], [24] har fungerat som referenser när det kom till att hitta komponenter lämpliga för arbetet. Förutom dessa böcker nyttjades även webben för att hitta information, främst för kontrollenheter och verktyg.

Strukturen som togs fram under förundersökningen bestod till större del av krav, för funktionalitet och hårdvara. Dessa krav beskrev vilken hårdvara som kunde användas för utomhusbruk med önskad funktionalitet och hållbarhet. Ett exempel är hur valet av olika komponenter påverkade ett batteridrivet system. Hela strukturen som togs fram i förundersökningen byggde på vissa konstanter i examensarbetet, dessa konstanter var de olika mätsystemen. Det yttersta kravet, var att försöka använda dessa system.

Ett annat krav begränsade även budgeten. Prototypen har utvecklats för att vara ekonomiskt hållbar där flera enheter ska kunna kopplas till samma system. Då arbetets omfattning är i form av ett examensarbete för en utbildning, där erfarenhet och professionell expertis saknas, har billig och lätt utbytbar utrustning prioriterats.

3.4. Utveckling

Efter att en klar struktur för arbetet tagits fram, skulle utveckling av prototypen påbörjas. Utvecklingen skedde både i en teoretisk och en praktisk del, som ofta gick hand i hand. Den teoretiska delen bestod till stor del av vilka funktioner i programmeringen som krävdes och kraven för hårdvaran. Medan den praktiska delen bestod av konstruktionen.

3.4.1. Teoretisk del

Denna del av examensarbetet var den mest tidskrävande, då den var direkt kopplad till tester och validering. I denna fas behandlades i teori vad som krävdes av hårdvaran och mjukvaran, för att nå önskade funktioner. På detta sätt utvecklades logiken bakom dataloggern. Logiken för den enkeltrådade dataloggern beskriver exempelvis när data ska loggas, när en kommunikationsförbindelse till servern ska skapas eller hur problem ska hanteras. Det största kravet som ställdes på logiken var hållbarheten och hur autonom lösningen var. Tekniskt sett ska dataloggern aldrig behöva någon typ av närvaro förutom vid installation. Exempelvis ska logiken finnas för att lösa eller undgå olika typer av fel, med undantag för om hårdvaran havererar. Utöver dataloggern skapades även en server och ett användargränssnitt med sina egna krav. Krav på servern var exempelvis simultan anslutning av flera klienter och ett enkelt och användbart användargränssnitt.

3.4.2. Praktisk del

Hårdvaran valdes utifrån sina specifikationer och testades för vilka funktioner den skulle utföra. Om hårdvaran fungerade utifrån den teoretiska delen, integrerades den i lösningen. Integrationen i lösningen bestod av en del programmering, och en del konstruktion. De största kraven för den praktiska delen var fysisk hållbarhet hos dataloggern. Exempelvis hur utsatt och hur skyddad mot miljön den var.

3.5. Tester och validering

Alla delar av examensarbetet har testats utifrån dess funktion och validerats beroende på resultatet. Hårdvara har ej testats utifrån specifikation när det kommer till bland annat temperatur för hållbarhet. Dock är den testad för förbrukning och produktion av elektricitet. Testerna som har utförts har varit konstruerade utifrån förväntad funktion och önskat resultat. Exempelvis skapades ett program med flera klienter för anslutning till den multitrådade servern, med syftet att verifiera hur väl flera anslutningar samtidigt fungerade. Majoriteten av testerna för hårdvaran har utgått från energiförbrukningen.

3.6. Källkritik

Källorna [1], [2], [3], [4], [5], [6], [7], [8], [9], [10] och [11] beskriver hårdvaran som har använts i examensarbetet. Hårdvaran har valts utifrån dokumentationen i dessa källorna och även testats därefter. Om hårdvaran har fungerat utifrån vad som angetts i sin dokumentation, exempelvis strömförbrukning, har källan ansetts vara pålitlig. På samma sätt har källor (tillverkare och återförsäljare) ansetts vara opålitliga om hårdvaran inte har fungerat enligt dokumentation. Dokumentation för flödesgivaren, nivågivaren och regnmätaren som har undersökts i examensarbetet finns i källorna [12], [13] och [14]. Dessa har testats på samma sätt som resterande hårdvara, med hjälp av olika mätinstrument. Med undantag för flödesgivaren [12], där tillräckligt med information inte fanns tillgänglig för modellen som användes i examensarbetet. Detta gjorde inte källan opålitlig då informationen kan vara sekretessbelagd, dock gjorde det tester svårare att utföra.

Till mjukvaran har det använts flera olika ramverk och program, dessa är alla väldokumenterade och välanvända, samt finns mycket information att hämta från olika forum där användare själva har delat kodexempel, problem och lösningar. De olika källorna som har använts för mjukvaran är bland annat; Java multithreading [15], socket kommunikation [16], JSON [17], jFrame [18], Codeigniter [19], jQuery [20], AdminLTE [21] och WampServer [22]. Fyra böcker har även använts i examensarbetet, främst två böcker; "Computer Networking" [16] och "Objektorienterad Programmering" [18] som kan anses vara trovärdiga, då dem är studielitteratur som används på bland annat Lunds Universitet. Medan två av böckerna är olika volymer för en encyclopedia för elektroniska komponenter [23] och [24]. Dessa uppslagsverk beskriver elektronik, elektroniska komponenter och olika kretsar men går inte in på djupet för ämnena. Detta gör uppslagsverken trovärdiga för vad dem används till, detta då det är lätt att verifiera informationen med andra källor, via webben. Källorna [25], [26], [27], [28], [29] och [30] hänvisar till hårdvara som har testats men inte använts i lösningen. Bortsett från "Velleman Solar Charger" [27] har källorna ansetts vara trovärdiga med samma anledning som resterande hårdvara. Den enda källa som inte har ansetts vara trovärdig under examensarbetet är [27], detta då egna mätningar jämfört med specifikationer för energiförbrukning visade avvikelser på upp till 300%. Till all hårdvara förväntas en viss avvikelse från specifikation, detta då hårdvara ofta testas i någon typ av labbmiljö. Avvikelser kunde accepteras beroende på hur prototypen i helhet påverkades, där en avvikelse på 300% för en laddningsregulator hade påverkat batteridrift väsentlig.

4. Analys

I detta kapitel beskrivs bakgrunden till de olika besluten. Av dessa beslut kommer det att framgå varför en lösning för regnmätaren togs fram, men inte för flöde och nivå, hur valet av hårdvara skedde och mätresultat. Förutom detta kommer det att anges vilka verktyg som har använts och problem som har förekommit, tillsammans med deras lösningar.

4.1. Givare

Tre olika typer av givare har undersökts, givare för flöde, nivå respektive regn. Problemet som uppstod för flödesgivaren var brist på information, då ingen information om vad den analoga utsignalen motsvarade, var tillgänglig. Detta gjorde att planerna för att implementera dataloggern för flödesgivaren togs bort i förundersökningen. För nivågivaren var problemet snarare en brist på tid och möjligheten att utföra tester. Nivågivaren hade ett mätområde på 0 till 3 meter, att testa 3 meter för att fastställa funktion vid max utsignal krävde utrustning som inte fanns tillgänglig. Utöver detta var nivågivaren kostsam vilket höjde kraven för kvalitet.

Regnmätaren vars funktionalitet var enkel, samtidigt som den var lätt att testa har därför prioriterats under examensarbetet. Bara regnmätaren ingick i lösningen men nivågivaren var till viss del implementerad då både hårdvara och mjukvara var testad för att mäta den analoga signalen från givaren. Dessutom var servern även anpassad för klienter som skickade olika typer av data, vilket enheter för både regn och nivå hade gjort.

4.2. Övrig hårdvara

Valet av mikrokontroller utgick främst från om det var möjligt att programmera mikrokontrollern i Arduino IDE. En lösning bestående av ett redan känt programspråk skulle erbjuda större kvalitet på det slutliga resultatet. Enheter från märkena “Arduino” och “Adafruit” testades utifrån internt minne, arbetsfrekvens och hur stor energiförbrukningen var. Energiförbrukningen för den valda hårdvaran visas i tabell 1. Efterhand som examensarbetet utvecklades uteslöts användning av enheter från märket

“Arduino” då de tillgängliga enheterna inte hade tillräckligt med internminne för att använda alla bibliotek som krävdes. Därför användes enheter från “Adafruit” som hade mycket internminne och arbetade med en högre frekvens för att kunna exekvera kod snabbare.

Den första mikrokontrollern som testades var den nyutvecklade “Adafruit Metro M4 express” [25] som använde sig av ett nytt mikrochip. Detta ledde till problem då bibliotek som krävdes för kommunikation inte hade implementerats för dessa mikrochip. Den andra mikrokontrollern som testades var av samma serie men använde sig av ett äldre mer väldokumenterad mikrochip, detta var en “Adafruit Metro M0 express” [26]. Denna enhet användes under större del av examensarbetet, dock byttes den ut när energilösningen utvecklades. Energilösningen var dyrare därför var det mer ekonomiskt hållbart att byta mikrokontrollern till en “Adalogger Feather M0 express” [1] vilket uppfyllde alla krav.

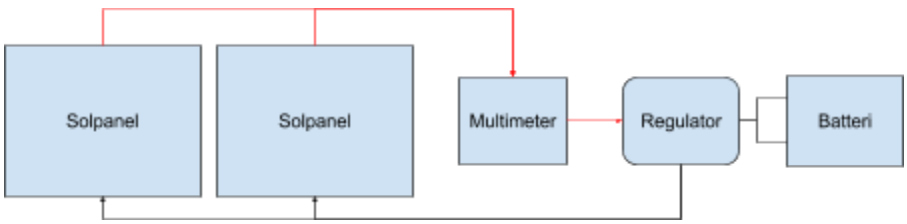
Regulatorerna till arbetet valdes utifrån aspekter som strömförbrukning och effektivitet, tagna från encyclopedia för elektriska komponenter [23],[24]. Högre effektivitet och lägre strömförbrukning tillät mindre och billigare solceller, samt bidrog till minskad värmebildning. Två olika regulatorer användes i prototypen, en laddningsregulator för att förse prototypen med energi från solceller och en variabel spänningsregulator för att mata mikrokontrollern med korrekt spänning. Under arbetets gång har dessa regulatorer bytts ut en gång vardera av samma anledning, för hög strömkonsumtion. Laddningsregulatorn som testades först var en “Velleman Solar Charger SOL10UC3” [27] denna regulator skulle enligt specifikation ha en strömkonsumtion på mindre än 6mA, medan det uppmätta värdet låg på upp till 20mA. Detta var oacceptabelt därför valdes en ny regulator “Kemo Electric M174” [5] som enligt specifikationer hade en strömkonsumtion på mindre än 2mA, den uppmätta strömkonsumtionen låg på ca 0.5mA. Den första spänningsregulatorn som valdes för prototypen var en “Digilent Voltage Regulator Module Rev. B” [28] bestående av kretsen “TPS54620” [29] som är framtagen av Texas Instrument. Kretsen är väldokumenterad men ett misstag gjordes då den inte är byggd för en last med en låg strömkonsumtion. Regulatorns uppmätta strömkonsumtion låg på ca 30mA över en batterispänning på ca 12V. Efter detta valdes spänningsregulatorn “Pololo step-down voltage regulator D24V25F5” [4] med en dokumenterad strömkonsumtion på 0.7mA. Den uppmätta strömkonsumtion mättes till ca 0.5mA.

Utvecklingskorten för arbetet valdes utifrån hur kompatibla de var med mikrokontrollern. Valet av relä gjordes med hjälp av en encyklopedi [23], [24] där funktionen för ett spärrelä beskrevs. Ett spärrelä valdes då det viktigaste kravet för relät var energiförbrukningen, vilket ett spärrelä löste.

Utvecklingskortet för kommunikation valdes utifrån tillgänglighet, istället för att använda en 3G-modul för att direkt kommunicera med servern användes ett Ethernet-kort [3] tillsammans med en 3G-router [8]. Detta då router av hög kvalite fanns tillgängligt. Routern testades i början av examensarbetet för hur väl det skulle fungera i prototypen. Olika praktiska tester utfördes för att avgöra bland annat; strömkonsumtion vid standby och dataöverföring, signalstyrkor med uppkopplingstid på olika platser samt funktionalitet när routern startades och stängdes flera gånger under en längre tid. Routern testades främst genom att utveckla en klient och en server där klientens kommunikation till servern testades för en anslutning över 3G-nätet.

Motivering av valet för arbetets eventuellt viktigaste del, batteriet, har skiftat under arbetets gång. Till en början var mängden energi som batteriet kunde lagra den viktigaste delen, det vill säga antalet wattimmar. Detta ansågs vara viktigt då fler wattimmar utökade tiden för batteridrift. Efter tid som arbetet utvecklades gick detta över till hållbarhet och kompatibilitet. Det initiala batteriet var ett litiumbatteri från Ansmann [30] med en ström på 2600mAh och en spänning på 14,8V. Nackdelen med dessa var kostnaden och hur livslängden och funktionaliteten drastiskt sjunker i en utomhusmiljö, speciellt vid lägre temperaturer. Istället valdes en större och klumpigare blyackumulatorbatteri [10] med en ström på 2300mAh och en spänning på 12V. Detta batteri valdes även med hänsyn till hur laddningsregulatorn [5] fungerade, då regulatorn är utvecklad för laddning av blyackumulatorbatteri, inte litiumbatteri. Livslängden för batterier påverkas av vilken miljö de utsätts för. Därför ansågs det även vara mer ekonomiskt hållbart att vid haveri byta ett blyackumulatorbatteri vars kostnad är betydligt mindre än ett litiumbatteri. Dessutom krävs solceller för kontinuerlig drift av prototypen vilket innebär att ett batteri med mindre kapacitet kan användas.

Valet av solceller skedde initialt enligt dess specifikation och kostnad, där produktionen av elektricitet skulle ligga minst 4 gånger högre än vad som användes vid normal drift vilket visas i tabell 3 och tabell 4. Detta krav är hypotetiskt satt till 4 gånger strömkonsumtionen med hänsyn till de sämsta förutsättningar dataloggern kan utsättas för. Dataloggern ska fungera dygnet runt, året runt, för en miljö med ett antal ljusa respektive mörka timmar, i ett nordiskt klimat där solen kan vara skymd hela dagar. Initialt ska det produceras 3 gånger så mycket elektricitet än vad som används för att kunna driva systemet kontinuerligt. Drift över ett dygn ska vara möjlig med en elektrisk produktion över 8 ljusa timmar (vintertid). Sedan krävs ett överskott av energi som kan användas vid dagar då solen är skymd då det inte produceras tillräckligt med elektricitet. Dessutom kommer ett visst slitage uppstå vilket kommer att minska effektiviteten hos solcellerna. Därför sattes den undre gränsen till 4 gånger vad som används. Med detta krav testades solceller [11], detta för att verifiera funktionaliteten. Tabell 2 visar mätdata för två solpaneler som har kopplats parallellt. Solpanelerna testades genom att ansluta en bänkmultimeter mellan solpanelerna och regulatorn, vilket illustreras enligt figur 11.



Figur. 11. Test av solceller

4.3. Verktyg

De viktigaste verktygen som har använts under examensarbetet listas här.

- **Instrument:**

Multimeter, bänkmultimeter, oscilloskop, miniräknare, dator (windows), lödstation och laserskärare.

- **Programvara:**

Arduino IDE, Visual Studio IDE, Eclipse IDE, WampServer, Notepad++ och Google docs.

4.4. Mätningar

Här redovisas mätningar och specifikationer som har använts under arbetet. Observera att medelvärden är angivna för uppmätt data. Figur 17 till och med figur 21 i appendix A visar ytterligare mätningar för analog spänning och spänningsfall över det använda blyackumulatorbatteriet. Tabell 1 redovisar specifikationer för hårdvara tagna från källorna enligt Teknisk bakgrund avsnitt 2.1. Tabell 2 och tabell 3 visar mätningar, där tabell 2 visar mätningar gjorda för energilösningen medan tabell 3 visar mätningar för den konstruerade prototypen.

Tabell. 1. Specifikation för hårdvara

	Spänning (V)	Ström (mA)	Effekt (W)	Temp min (°C)	Temp max (°C)
Mikrokontroller	3,7	11	0,0407	-40	+85
Ethernet	3,3	128	0,4224	-40	+85
Relä	3,3	50	0,165	-40	+85
Spänningsregulator	6 - 38	0,7	-	-	-
Laddningsregulator	12	< 2	< 0,024	-	+50
RTC	3,3	1	0,0033	-40	+85
ADC	2 - 5	0,15	-	-40	+125
Router	10 - 30	-	5,5	-30	+60
Batteri	12	2300	27,6	-	-
Solpanel (max)	17,6	580	10	-40	+85
Micro SD kort	-	< 0,25	-	-45	+85

Tabell. 2. Solceller, 2 paneler, elektrisk produktion

	Mörker (mA)	Indirekt solljus (mA)	Direkt solljus (mA)
Batterispänning (ca 12V)	0	5 - 15	50 - 100

Tabell. 3. Mätdata för hårdvara

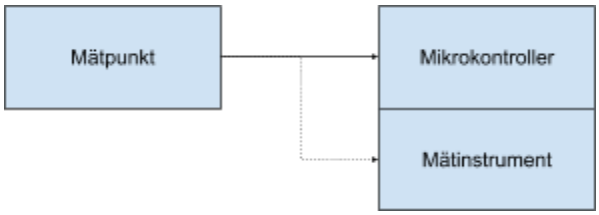
	Spänning (V)	Ström (mA)	Effekt (W)
Normal drift	12	6,7	0,0804
Dataöverföring	12	~250	3

4.5. Problem och lösningar

Examensarbetet har till större del bestått av problem och lösningar. I kronologisk ordning kommer de mest tidskrävande problemen och deras lösningar att redovisas.

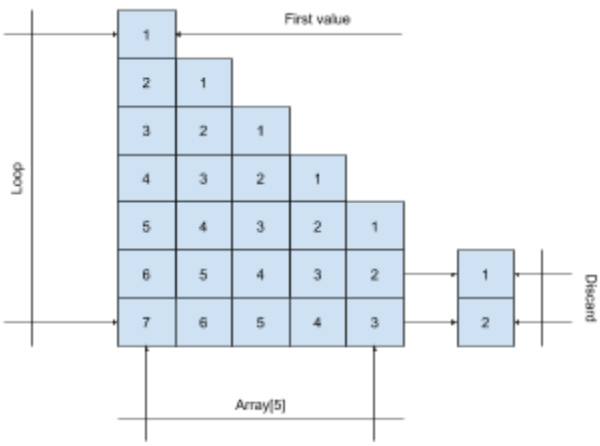
4.5.1. Analog signal

Mätning av analoga signaler för nivå och flöde visade sig ej vara genomförbar med mikrokontrollern. Mikrokontrollern kunde inte mäta de analoga signalerna med tillräcklig precision samtidigt som störningar gjorde signalen oanvändbar. Alla tester utfördes parallellt med mätinstrument enligt figur 12. Mätinstrumenten som användes var en bänkmultimeter och ett oscilloskop för att verifiera och validera funktioner hos all hårdvara. Alla mätningar utgick från vad som angivits i specifikationerna i tabell 1, dock valdes de utifrån funktion där en viss avvikelse var acceptabel. Nivågivaren testades genom att sänka ner den i vatten tillsammans med ett måttband medan flödesgivaren kunde testas genom att rotera den magnetiska propellern. Under testerna gjordes dock aldrig mätningar samtidigt med både mikrokontrollern och mätinstrumenten, detta då inre resistanser och liknande kunde ge avvikelser i resultatet.



Figur. 12. Mätning; hårdvara

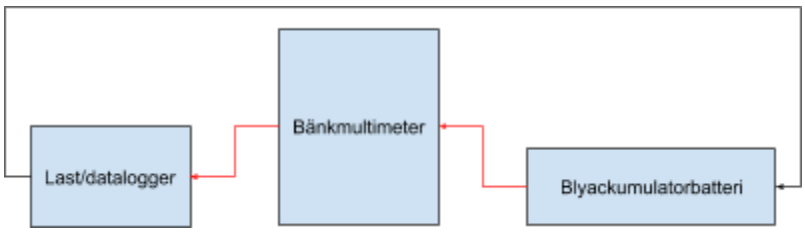
För att korrekt mäta den analoga signalen användes en separat analog till digital (ADC) 16-bitars omvandlare. Denna omvandlare mätte en differentiell spänning mellan signal och jord med en 16-bitars precision och vidarebefordrade en digital I^2C datasignal till kontrollenheten. Detta minskade störningarna och ökade precisionen för hårdvaran. För att göra signalen mer användbar skapades även ett medelvärde i mjukvaran, detta medelvärdet presenteras som ett "flytande medelvärde" i figur 13. Medelvärdet beskrivs som "flytande" då det uppdateras var gång ett nytt värde registreras och ger en långsam men direkt förändring. Resultatet för denna metod presenteras enligt figur 22, 23 och 24 i appendix A. Figur 21 visar mätningar för kontrollenheten, kurvan som presenteras kan anses vara väldigt instabil då många värden kan tolkas som outliers. Dessa outliers betraktades som felmätningar och detekterades genom att registrera en avvikelse från det flytande medelvärdet. För att korrigera problemet sattes en undre och en övre gräns för hur stora avvikelser som godtogs för att ingå i det flytande medelvärdet.



Figur. 13. Flytande medelvärde

4.5.2. Strömförbrukning

Strömförbrukningen för hårdvara och olika operationer har varit ett problem under hela arbetet. Då hårdvara som tillåter 3G-kommunikation [8] och kommunikation via Ethernet [3] kräver en stor mängd energi. Strömförbrukningen mättes genom att koppla en bänkmultimeter i serie med batteriets positiva pol, utan att ansluta solpaneler enligt figur 14.



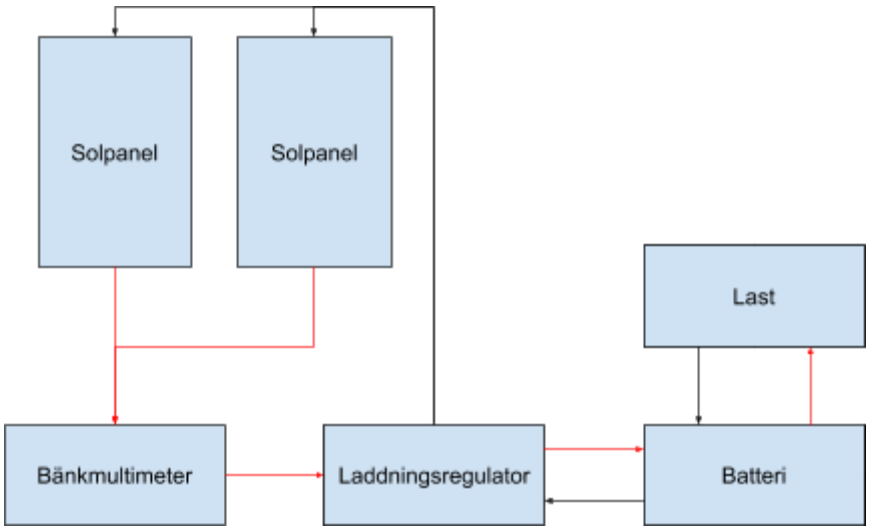
Figur. 14. Mätning; strömförbrukning

För att lösa detta problem användes ett relä för att switcha hårdvaran för kommunikation, problemet med relä är att även de använder mycket ström. Utifrån detta valdes ett spärrelä [2], där den resulterande strömförbrukningen för prototypen visas i tabell 3. Till skillnad från ett vanligt relä kräver ett spärrelä endast en temporär signal för att byta operationsläge. Ur ett energimässigt perspektiv är denna typen av relä fördelaktig, dock inte utan brister. Då en konstant signal inte krävs vet man tekniskt inte vilket läge reläet befinner sig i, detta kan lösas till viss grad med programmering. Reläet kopplades enligt figur 16, där ett relä switchar två olika strömmar för kommunikationen, en spänning för 3G-kommunikation och en spänning för Ethernet-kommunikation.

4.5.3. Solceller

Solceller har undersökts och deras funktion har visat sig vara mindre pålitlig i ett svenskt klimat. Solcellers märkning visar operation för optimala förhållande vilket under bästa förutsättningar inte är möjliga i Sverige. Vad som krävdes för att lösa detta problem var mer solceller eller betydligt bättre kvalitet för att öka effektiviteten. För att driva systemet under sämre väderförhållande har solceller märkta för 20W använts. Detta trots att hårdvaran i snitt drar 2 Wh/dygn. Bakgrunden till detta valet har varit

elektrisk produktion under en testperiod i ett svenskt vinterförhållande. Testet genomfördes enligt figur 15 genom att koppla två solpaneler via en bänkmultimeter och en laddningsregulator till ett batteri över en längre tid. Resultatet visas i tabell 2 vilket i sin tur visar hur mycket den elektriska produktionen varierar beroende på väderförhållande. Tester genomfördes även för när bänkmultimetern var kopplad mellan laddningsregulatorn och batteriet, där den enda skillnaden var laddningsregulatorns egna strömförbrukning på 0.5mA.



Figur. 15. Mätning; solpaneler

4.5.4. Autonom funktion

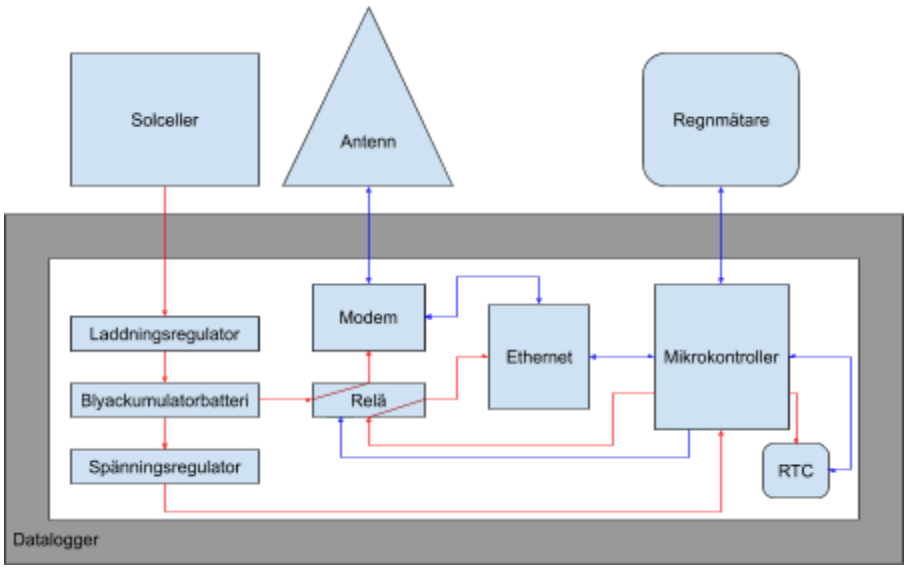
Teoretiskt ska mikrokontrollern aldrig sluta fungera, dock finns alltid risken att en enhet "hänger sig", på grund av konflikter i mjukvaran eller liknande. Detta är ett problem i alla digitala enheter som ofta kan lösas genom en omstart. Problemet kan lösas genom implementation av mjukvara eller hårdvara. Lösningen har utnyttjat mjukvara för att automatiskt säkerställa funktion. Mikrokontrollerns processor har en separat tråd med begränsat minne som kan driva en klocka. Klockan kan för enheten max räkna till 16 sekunder och exekverar omstart av hårdvaran om den inte blir återställd inom detta intervall. Fördelen är hög funktionalitet för en mindre lösning, medan nackdelen är det låga minnet, vilket kräver att funktionen pausas när operationer som tar mer tid än 16 sekunder utförs.

5. Resultat

Resultatet redovisas utifrån sina tre delar som gör upp hela lösningen. Datalogger för loggning av mätdata, server för kommunikation och användargränssnitt för presentation.

5.1. Datalogger

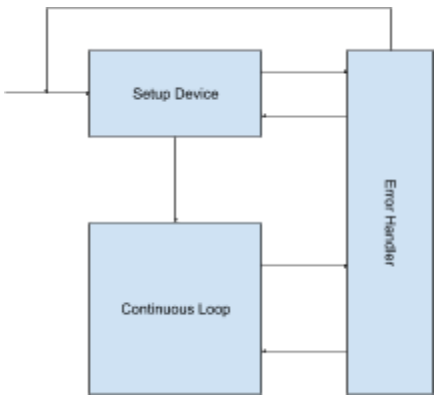
Dataloggern är en mindre kapsling bestående av hårdvara enligt figur 16. All hårdvara som redogörs i kapitel 2.1 finns placerade i denna kapsling och är kopplade enligt figuren. Dataloggern kan delas upp i tre delar; en mätande del från regnmätaren, en kommunikationsdel från antennen och en energidel från solcellerna.



Figur. 16. Strukturen för dataloggerns hårdvara; röd färg representerar ström medan blå färg representerar datasignal

Dataloggerns logik är utvecklad utifrån funktionalitet och hur väl hårdvaran fungerar över kontinuerlig batteridrift. För att konfirmera status skickar dataloggern en ping till servern varje dygn, om denna ping inte kommer fram meddelar servern användaren via ett konsolfönster. Vid regn

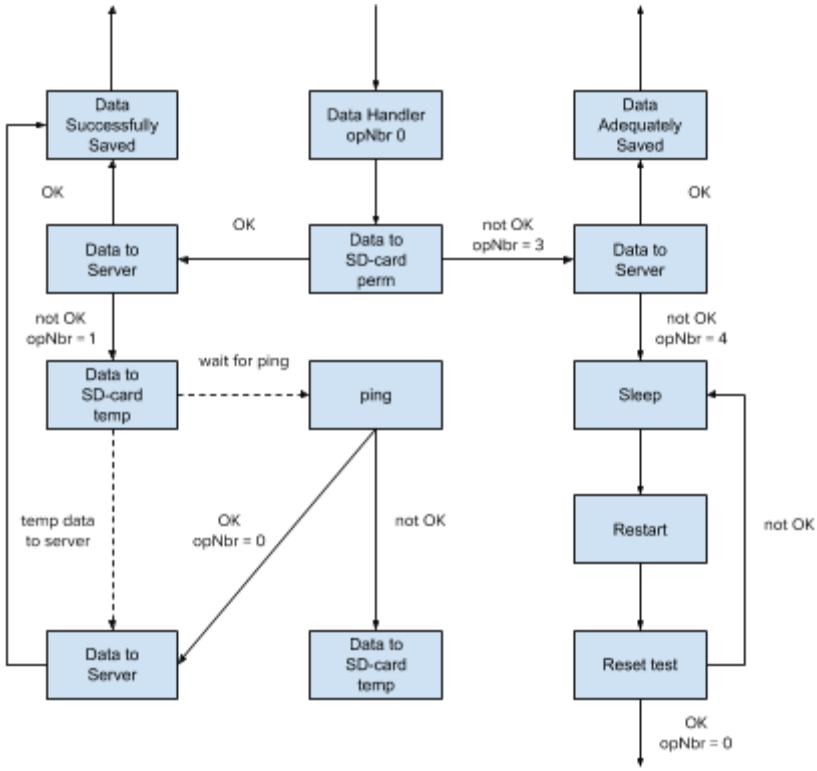
registrerar dataloggern värden i form av unix tidsstämplar, där en tidsstämpel motsvarar 0,2 mm regn. Denna data skickas sedan vidare till servern, 60 minuter efter sista registreringen. Vid problem då dataloggern inte fungerar som den ska, exempelvis om anslutning eller hårdvara inte fungerar, går loggern in i ett tillfälligt driftläge. Om problemet är känt och logik för dess hantering existerar, kommer enheten att försöka logga data på ett annat sätt. Om enheten inte kan logga data kommer en omstart att ske. Om felet kvarstår påtvingas ett sömnläge. För att konfirmera funktionalitet hos dataloggern testas all hårdvara varje gång en ping ska skickas till servern. Vid fel avgör detta test om enheten kan återgå till normal drift eller om en omstart krävs. Felkod som representerar driftläget sparas i FLASH-minnet, vilket kvarstår efter omstart. Om mer än ett driftläge utöver det normala försöker exekveras kommer enheten att testa all hårdvara, innan den läggs i ett sömnläge. Detta då enheten inte kan anses fungera om det finns två eller fler problem. Figur 17 visar den underliggande logiken för loggern.



Figur. 17. Underliggande logik bakom dataloggern

Enligt figur 17 initieras all hårdvara och mjukvara i “Setup Device”. Om denna fas exekverades utan fel hamnar systemet i “Continuous Loop”; som bland annat läser av givaren, loggar data och skickar data till servern. Vid fel i hårdvara eller för anslutning till servern, går systemet in i “Error Handler”. Detta steg hanterar kända fel, försöker lösa dem eller sätter enheten i ett driftläge för att undvika felen. Vid fel som stoppar den underliggande funktionaliteten hos dataloggern sätts systemet i ett sömnläge där systemet vaknar upp periodiskt för att försöka lösa felet.

Vid normal operation loggas data sällan, först sparas data temporärt på primärminnet och loggas permanent efter att en viss tid har passerat eller en viss mängd data har registrerats. Hur data loggas och hur eventuella fel hanteras visas i figur 18.



Figur. 18. Loggning av data

Vid eventuella fel som sker enligt figur 18 så hamnar systemet i olika driftlägen för att logga data. Driftlägen beskriver vilken logik dataloggern ska följa när den loggar data och driftläget definieras av ett operationsnummer (en integer). Det normala driftläget, med operationsnummer “0” visas i figur 18. Vid kända fel enligt figur 18, kommer ett annat driftläge tvinga dataloggern till att följa en annan logik där felet undgås. En ping till servern kommer dock alltid att testa all hårdvara och anslutning till servern, med syftet att återställa driftläget till det för normal drift. Dessutom kommer all hårdvara att testas efter omstart och enheten kommer endast att gå ner i sömnläge om enheten inte kan logga data, efter att felhanteraren har försökt lösa problemet.

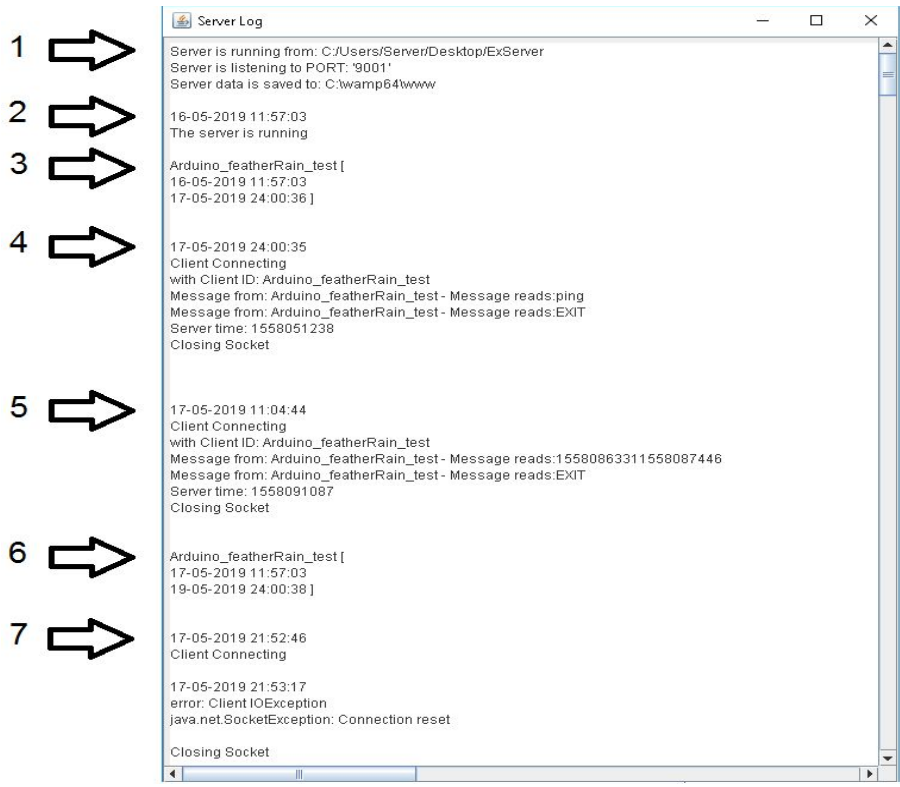
Dataloggern drivs av solceller och kräver därför ett soligt väderförhållande för att fungera. Solcellerna är testade för vintermånader och producerade mellan 5-50 mA för batterispänning, samtidigt som de inte var idealt placerade för att direkt exponeras av solljus. Solcellerna antas fungera bättre under resterande del av året då antalet ljusa timmar på dygnet ökar, samtidigt som solcellerna installeras korrekt. Korrekt installerade är med en södergående riktning och en vinkling uppåt, i detta fall sattes dem med ca 45 graders vinkel uppåt. Utifall solcellerna skulle sluta fungera eller täckas av diverse material, så kan loggern drivas ett antal dagar beroende på vilka operationer som utförs. Detta beskrivs enligt ett beräkningsexempel i tabell 4. I exemplet förutsätts batteriet vara i nyskick. Beroende på 3G-täckning kommer tiden för en anslutning variera, dock kommer det aldrig ta längre tid än en minut, förutsatt att allt fungerar. Därför är energiförbrukning för loggning av data beräknad över en minut.

Tabell. 4. Hypotetisk energiförbrukning vid batteridrift

	Effekt per timme för ett dygn (W/h/dygn)	Effekt över ett dygn (W/dygn)	Dygn med drift (dygn)	Timmar med drift (h)
1 - ping (ej regn)	0,0825	1,9796	13,9	334,6
2 - ping (1x regn)	0,0846	2,0296	13,6	326,4
3 - ping (2x regn)	0,0867	2,0796	13,3	318,5
4 - ping (3x regn)	0,0887	2,1296	13	311
5 - ping (4x regn)	0,0908	2,1796	12,7	303,9

5.2. Server

Den multitrådade java-servern lyssnar konstant på inkommande trafik. Om en klient (datalogger) ansluter, påbörjas en kommunikation med denna, där servern tar emot all data loggern har samlat och skickar tillbaka en tidsstämpel för att konfirmera anslutningen. Genom att konfirmera anslutningen meddelas dataloggern att all data har nått servern, på så sätt kan dataloggern återgå till sin normala operation. När data är mottagen sparas den som JSON-strängar till en textfil. Om servern inte har kommunicerat med en specifik klient under en bestämd tid, anges detta i en konsol för programmet. Detta indikerar att klienten till stor sannolikhet inte fungerar. Konsolen är konstruerad med Java JFrame och representerar och presenterar all data som under utveckling skrivits ut i konsolen i utvecklingsmiljön. Figur 19 visar ett exempel för uppstart av servern.



Figur. 19. Serverkonsol

Figur 19 visar uppstart och exekvering av vissa funktioner hos servern.

1. Uppifrån och nedåt skrivs först sökvägen för var applikationen exekveras. Sedan anges det för vilken port servern lyssnar på och till vilken sökväg all data ska sparas till. Porten och sökvägen måste anges i en separat textfil (settings.txt), denna textfil behöver sedan vara placerad i samma mapp varifrån servern exekveras.
2. Om och när servern kommer hit, anges det att servern startade korrekt för angiven port. Om sökvägen för var data ska sparas inte finns, kommer denna att skapas automatiskt.
3. En separat tråd som normalt är vilande kontrollerar vilka klienter som har varit anslutna till servern. Den första tiden anger när funktionen exekverades och den andra tiden anger när klienten senast behöver ansluta till servern för att anses fungera.
4. En klient, i detta fall datalogger ansluter för att pinga servern. Klienten skickar tre meddelande, sitt ID för att verifiera sin identitet, sin data och en end-flagga för att avsluta kommunikationen. Efter detta anges en unix tidsstämpel som skickas till klienten för att synkronisera klockan.
5. En datalogger ansluter för att skicka loggad data till servern.
6. Den vilande tråden kontrollerar igen vilka klienter som har anslutit sig till servern.
7. En klient försöker ansluta, inte en datalogger. En anslutning görs utan att någon data skickas, då anslutningen inte identifierar sig inom en bestämd tid stängs anslutningen och därmed tråden.

En unix tidsstämpel används för både synkronisera dataloggerns klocka med serverns klocka samtidigt som den verifierar att anslutningen till servern. På detta sätt meddelar servern klienten att all data är mottagen, så klienten kan radera all mätdata på sitt primärminne.

5.3. Användare

Användargränssnittet är byggt för att enkelt och effektivt komma åt och presentera data som har samlats in från vattennätet. Användargränssnittet består av en webbapplikation och en webbserver som drivs från servern där all data loggas. För att komma åt denna data skapas en anslutning via webbläsaren till applikationen där användaren först får bekräfta sin identitet genom en inloggning. Efter detta möts användaren av en tom lista med val för specifikationer, för vilken data som ska kommas åt. Dessa specifikationer är enhetens ID och datum i form av år och månad.

Mätdata presenteras i en lista med objekt som representerar regnskurar. Dessa objekt visar en tid för den sista registreringen för en regnskur, antalet registreringar och detta översatt till mängd regn per skur. För alla objekt kan en lista med minutvärden hämtas, den listan visar antalet registreringar som har gjorts för var minut under ett regn. Dessa värden kan matas in i exempelvis ett Google kalkylark för att skapa en kurva över nederbörd och intensitet, vilket visas i figur 33. Figur 25 till och med figur 32 i appendix A visar användargränssnittet.

6. Slutsats

Vad som har utvecklats under examensarbetet är en prototyplösning bestående av tre delar; en datalogger för regn, en server för kommunikation och ett grafiskt användargränssnitt för presentation. Arbetet har undersökt tre olika typer av givare, dock har bara en av dessa implementerats för lösningen. I lösningen har en regnmätare använts och all mätdata har loggats i form av unix-tidsstämplar, där var tidsstämpel motsvarar 0.2 millimeter regn.

Hårdvaran som har gjort avläsning av regnmätaren och kommunikation över 3G-nätet möjlig är en mikrokontroller programmerbar med Arduino IDE. För att driva dataloggern utan tillgång till det elektriska nätet har en energilösning bestående av solpaneler, en laddningsregulator och ett blyackumulatorbatteri använts. Lösningen drivs från ett batteri vilket har gjort funktionaliteten för kommunikationen begränsad då trådlös kommunikation är energikrävande. Därför har en logik tagits fram för hur data ska loggas, detta innebär att vissa bestämda parametrar behöver uppfyllas innan vissa operationer kan ske, varav kommunikation till servern.

Lösningen består av endast en datalogger men servern är uppbyggd för att kunna kommunicera med flera enheter samtidigt, detta har möjliggjorts genom att tråda servern. Funktionaliteten hos servern kan anses vara enkel, detta då en tråd lyssnar och skapar nya trådar för kommande anslutningar. Efter att servern tagit emot mätdata presenteras den i en webbapplikation med en bakomliggande webbserver. För var datalogger som har anslutit till servern presenteras data i form av antal registreringar som har gjorts för olika regnskurar, där en regnskur differentieras med ett bestämt mellanliggande uppehåll. Minutvärden kan även hämtas för var regn som har registrerats. Dessa minutvärden beskriver hur många registreringar som har tagits emot per minut för hela regnet och kan användas för att konstruera kurvor för nederbörd och intensiteten till specifika regnskurar. Alla delar av webbapplikationen presenteras i figur 25 till och med figur 32.

6.1. Frågor med svar

Detta kapitel svarar på frågorna angivna i problemformuleringen avsnitt 1.4.

- Hur ska data loggas?

  Data loggas med en mikrokontroller utvecklad av Adafruit, programmerbar med arduino IDE. Mikrokontrollern är väldokumenterad och lättanvänd, samt är enheten utvecklad med flera utvecklingskort med olika funktioner och kretsar som tillåter bland annat kommunikation och switchning av ström.

- Hur ska dataloggern försörjas med elektricitet?

  Dataloggern drivs med ett blyackumulatorbatteri och solceller. Batteriet kan driva dataloggern upp till 13 dygn och ska klara av kontinuerlig drift på obestämd tid med hjälp av solceller. Den största fördelen solceller har är att göra lösningen autonom utan underhåll där batteri behöver bytas.

- Hur ska kommunikation till servern ske?

  Dataloggern ansluter sig till servern via 3G-nätet där en bakomliggande logik styr hur kommunikation till servern ska ske, detta med hänsyn till bland annat strömkonsumtion. 3G-nätet används då det är välutvecklat och har stor täckning, förutom det finns mycket hårdvara och mjukvara tillgänglig för kommunikation över 3G-nätet.

- Hur ska servern utvecklas?

  Servern är utvecklad i Java och är multitrådad för att kunna ta emot flera klienter samtidigt. Genom att använda flera trådar kan klienter ansluta när dem vill, och alla klienter kan pinga servern samtidigt. Då klienterna kan kommunicera med servern när de vill behöver dem inte konfigureras för detta.

- Hur och vad ska presenteras?

  All data som har tagits emot av Java-servern presenteras i ett webbaserat användargränssnitt. Mätdata presenteras för hur mycket regn som har kommit vid olika regnskurar. Förutom detta kan ett minutvärde hämtas för alla regnskurar om man vill bygga en kurva för regnskurens intensitet. Dessa minutvärden presenteras enligt företagets önskan.

6.2. Brister

Bristerna som har upptäckts för lösningen anses vara acceptabla. Dock är det fortfarande brister, vilka kan undersökas och korrigeras med hjälp av vidare utvecklad mjukvara och ny hårdvara, det vill säga med mer tid och pengar. All hårdvara har brister, dessa brister är direkt kopplade till hållbarhet och kvalitet. Gemensamma brister för all hårdvara är bristfällig tolerans för bland annat fukt, temperaturskillnader samt UV-strålning i en utomhusmiljö. Att energiförbrukningen är låg medför en låg värmeproduktion för komponenterna, vilket i sin tur leder till mindre risk för kondens i den täta kapslingen. Men risken finns och komponenternas livslängd påverkas trots att det ligger inom området för vad komponenterna är märkta för.

FLASH-minnet i mikrokontrollern kan anses vara en förbrukningsvara då det sker ett visst slitage varje gång en läsning eller skrivning görs. Detta är inget som normalt slits ut, då FLASH-minnet håller programkoden som är konstant, förutom vid utveckling. Vid fel i hårdvaran eller mjukvaran kommer dock enheten att skriva ett par bitar till FLASH-minnet, detta för att hålla kvar felkoden efter omstart, så att enheten inte fastnar i en oändlig loop. Förutsatt att funktioner inte slutar fungera flera gånger om dagen, bör detta inte påverka hållbarheten anmärkningsvärt då minnet är utvecklad för ca 10 000 skrivningar.

Solceller och batterier kan även anses vara förbrukningsvaror, detta då effekten hos båda minskar under långvarig användning. Livslängden hos dessa beror både på hur mycket de används och vilken miljö de utsätts för. Hållbarheten för en blyackumulator minskar även beroende på hur den är positionerad vid laddning. Dessa brister hade kunnat minskas genom att öka kvaliteten. Detta ansågs dock inte vara nödvändigt för en prototyp.

6.3. Reflektion över etiska aspekter

Om lösningen för examensarbetet fungerar och används som förväntat, kan den vara till stor nytta för samhället. Genom att samla data, i detta fall för regn kan information tas fram för mängden som kommer över specifika regnskurar. Den data kan användas för att dimensionera vattenledningar i tätorter för att avleda regnvatten och undvika översvämningar. Översvämningar kan ske i både bostäder och brunnar, vilket kan påverka spillvattennätet i sämre förutsättningar. Om detta nät påverkas kan följderna innebära utsläpp och hög belastning för reningsverken. Problemen kan ofta lösas genom att överdimensionera, men detta kan inte ses som ett hållbart alternativ ur aspekter som ekonomi och miljö.

6.4. Framtida utvecklingsmöjligheter

För en utvecklare finns det stora utvecklingsmöjligheter för prototypen. Detta då alla delar av lösningen kan anses vara enkelt konstruerade. Mjukvaran är inte låst till specifika programspråk och dataloggern är inte låst till sin hårdvara. Detta betyder att både hårdvara och mjukvara kan modifieras och ändras om det finns kompetens för det. Om lösningen ska användas i en bredare utsträckning med flera enheter, bör en ordentlig databas implementeras. En databas implementerades inte, då fördelarna för prototypen inte var tillräckliga för arbetet som krävdes.

Referenser

[1] Adafruit Feather M0 Adalogger, hämtad 16 april 2019,
https://learn.adafruit.com/adafruit-feather-m0-adalogger

[2] Latching Relay FeatherWing, hämtad 16 april 2019,
https://learn.adafruit.com/mini-relay-featherwings

[3] Ethernet FeatherWing, hämtad 16 april 2019,
https://learn.adafruit.com/adafruit-wiz5500-wiznet-ethernet-featherwing

[4] Spänningsregulator, hämtad 16 april 2019,
https://www.pololu.com/product/2850/resources

[5] Laddningsregulator, hämtad 16 april 2019,
https://www.kemo-electronic.de/en/Light-Sound/Solar/M174-Solar-charging-regulator-Dual-16-A.php

[6] RTC, hämtad 16 april 2019,
https://datasheets.maximintegrated.com/en/ds/DS3231.pdf

[7] ADC, hämtad 16 april 2019,
https://learn.adafruit.com/adafruit-4-channel-adc-breakouts

[8] 3G-router, hämtad 16 april 2019,
https://www.induo.com/kop/routrar/3g-router/bb-smartworx-conel-ur5i-v2/

[9] SD-kort, hämtad 16 april 2019,
https://www.elfa.se/Web/Downloads/-t/ds/xmore-industrial-microSD-eng-tds.pdf

[10] Blyackumulatorbatteri, hämtad 16 april 2019,
http://www.ritarpower.com/upimg/201763031747252.pdf

[11] Solcell, hämtad 16 april 2019,
https://www.velleman.eu/downloads/6/sol10p_sol30p_sol60pa5v02.pdf

[12] Flödesgivare, hämtad 16 april 2019,
http://www.globalw.com/downloads/SMProduct/tx100B.pdf

[13] Nivågivare, hämtad 16 april 2019,
http://www.mjk.se/fileadmin/datablad/d_7060.pdf

[14] Regnmätare, hämtad 16 april 2019,
http://www.mjk.se/fileadmin/datablad/d_regn_prof.pdf

[15] Java multithreading, hämtad 13 mars 2019, https://www.tutorialspoint.com/java/java_multithreading.htm

[16] J. Kurose, K. Ross, *"Computer Networking A Top-Down Approach"*, Harlow, Pearson Educational Limited, 978-0-273-76896-8, 2013

[17] JSON, hämtad 13 mars 2019, https://www.json.org/

[18] P. Holm, *"Objektorienterad programmering och Java"*, Lund, Studentlitteratur AB, 978-91-44-04830-7, 2014

[19] Codeigniter, hämtad 13 mars 2019, https://www.codeigniter.com/

[20] jQuery, hämtad 13 mars 2019, https://jquery.com/

[21] AdminLTE, hämtad 13 mars 2019, https://adminlte.io/

[22] WampServer, hämtad 13 mars 2019, http://www.wampserver.com/en/

[23] C. Platt, "Encyclopedia of Electronic Components", (Vol 1), Sebastopol, Maker Media, 978-1-449-33389-8, 2012

[24] C. Platt, F. Jansson, "Encyclopedia of Electronic Components", (Vol 2), Sebastopol, Maker Media, 978-1-449-33418-5, 2014

[25] Adafruit Metro M4 express, hämtad 18 april 2019, https://learn.adafruit.com/adafruit-metro-m4-express-featuring-atsamd51

[26] Adafruit Metro M0 express, hämtad 18 april 2019, https://learn.adafruit.com/adafruit-metro-m0-express-designed-for-circuitpython

[27] Velleman Solar Charger SOL10UC3, hämtad 18 april 2019, https://www.velleman.eu/downloads/6/sol10uc3a6v01.pdf

[28] Digilent Voltage Regulator Module Rev. B, hämtad 18 april 2019, https://reference.digilentinc.com/vrm/vrm

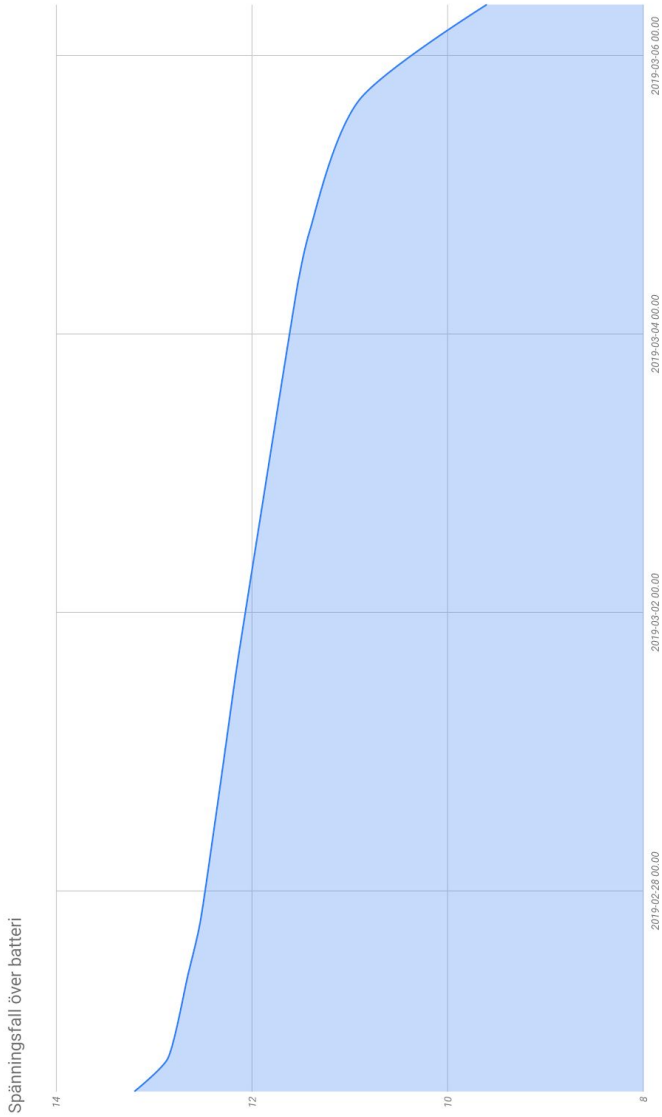
[29] TPS54620, hämtad 18 april 2019, http://www.ti.com/lit/ds/symlink/tps54620.pdf

[30] Ansmann litiumbatteri, hämtad 18 april 2019, https://www.ansmann-energy.com/images/stories/akkupacks/standard/2447-3036-01_V01_0DA.pdf7349-da-01-ml-ANSMANN_LI_ION_AKKUPACK_14_8V_2_6A_de_en.pdf

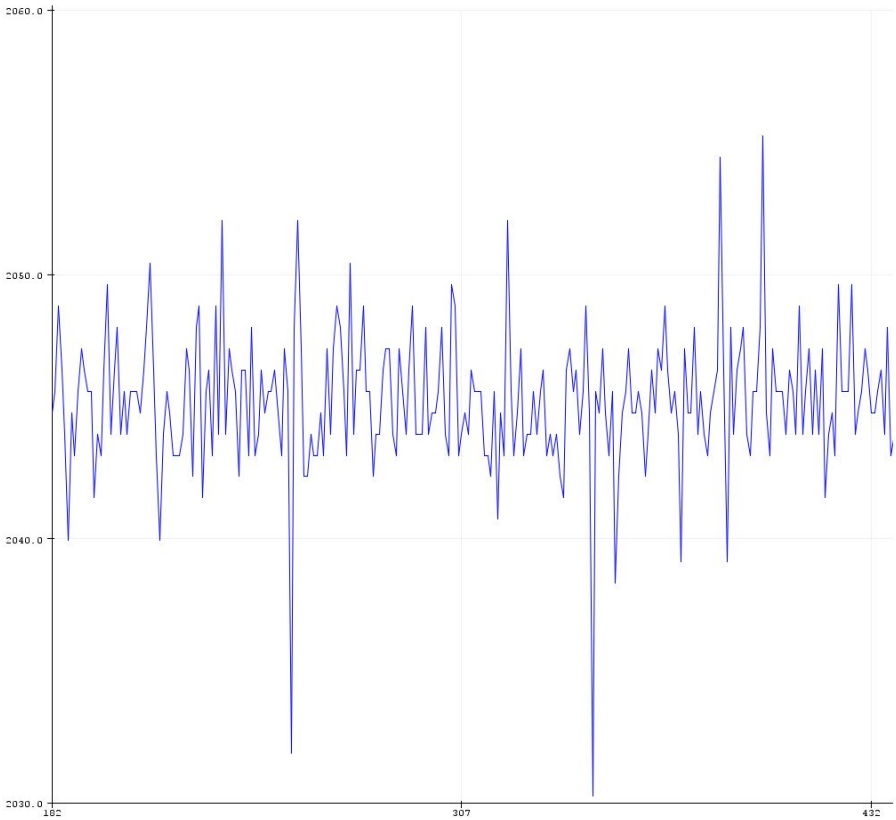
Figurlista

Tabellista

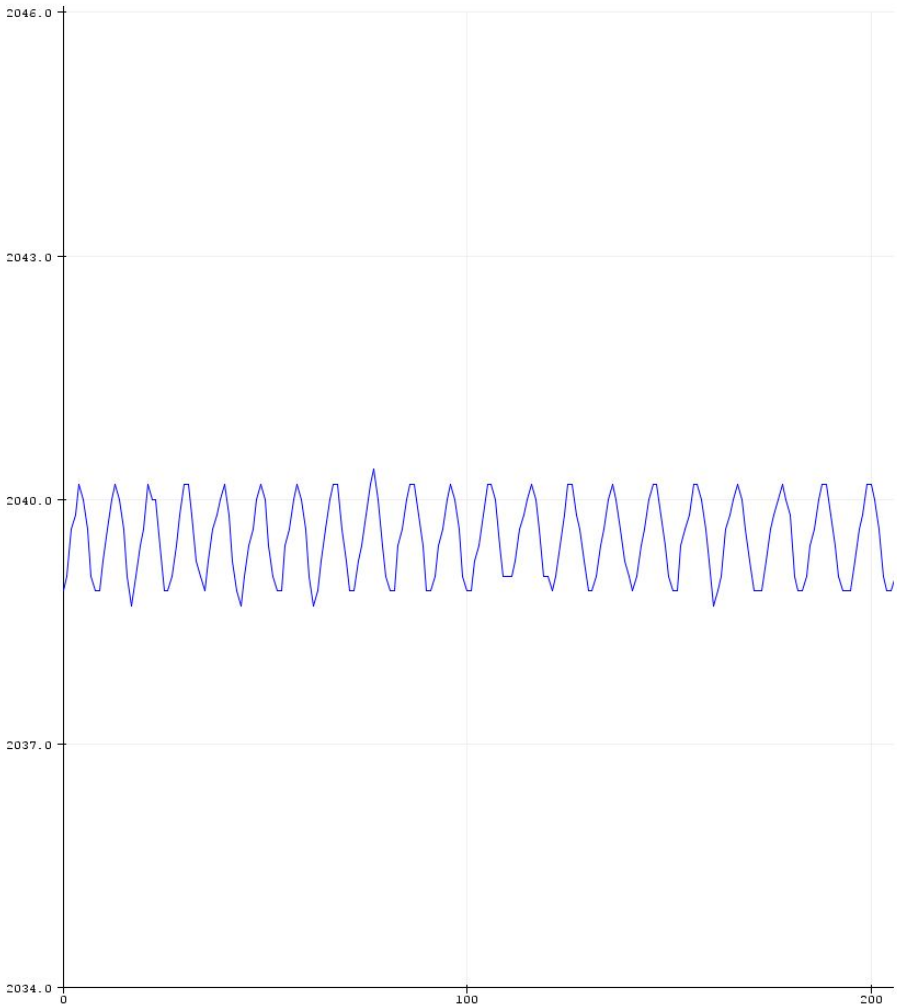
Appendix A:



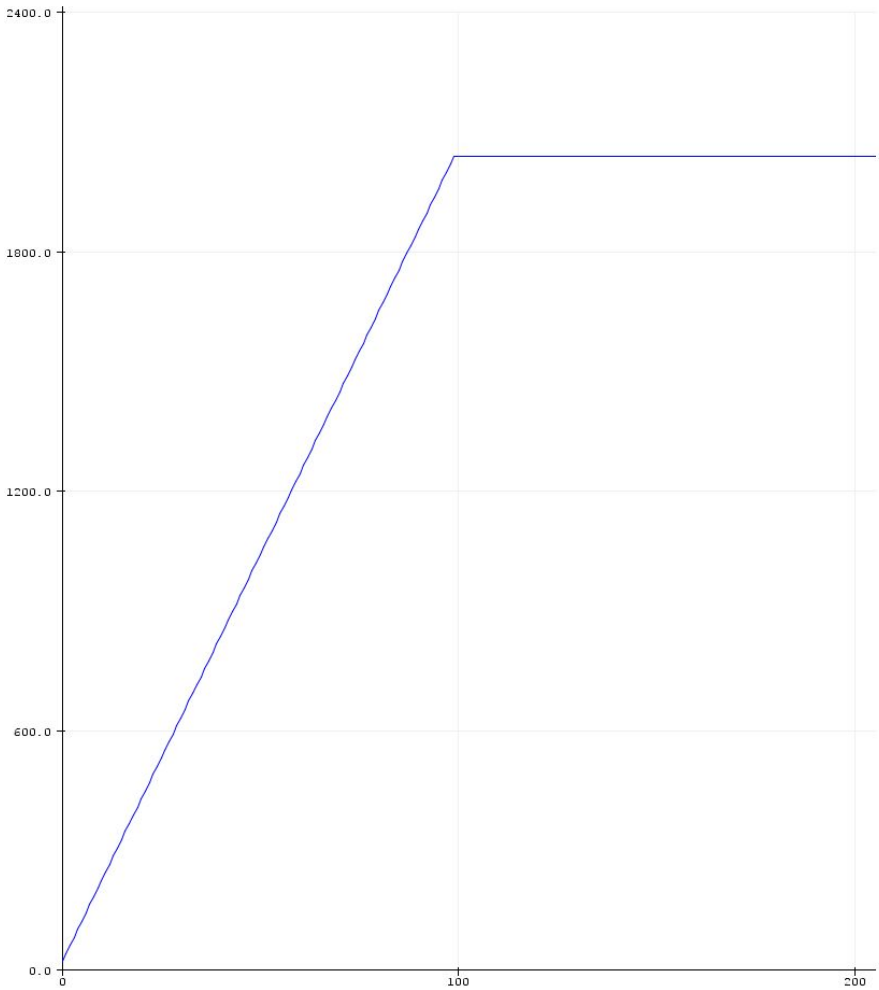
Figur. 20. Spänningsfall för batteri med last på ca 11 mA, rumstemperatur, Y-axel i spänning (V), X-axel i tid (dygn)



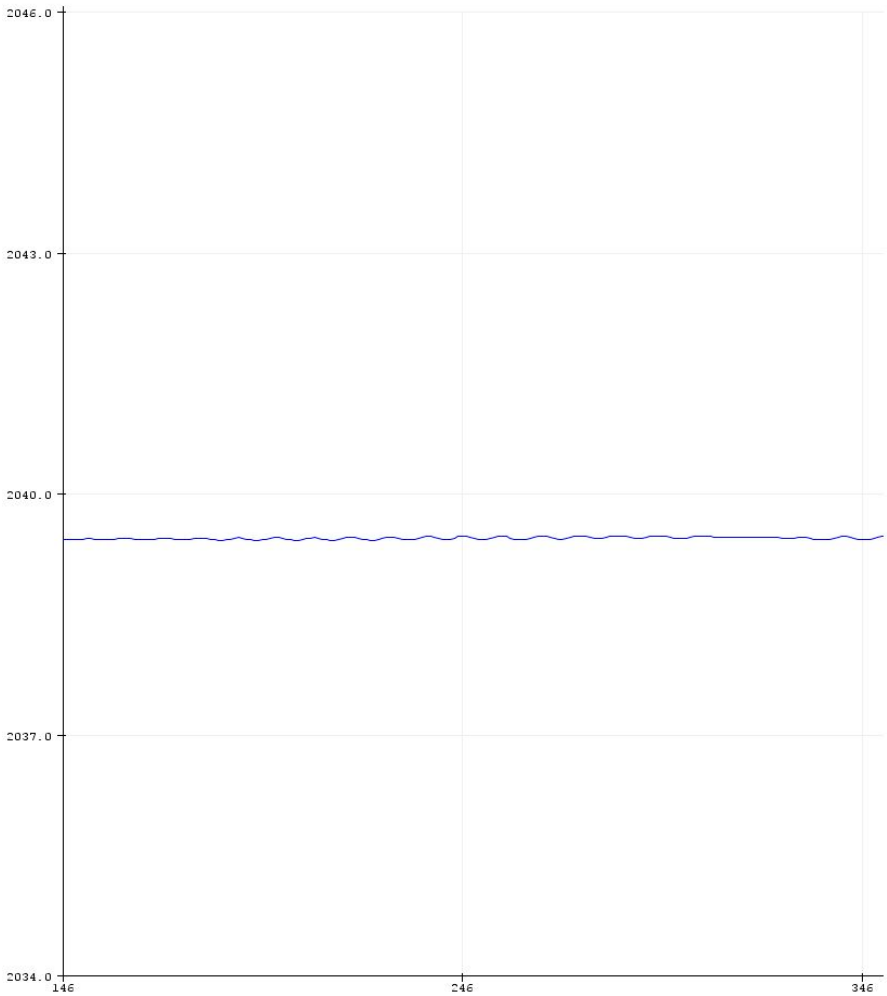
Figur. 21. Analog signalavläsning för mikrokontroller, Y-axel i spänning (mV), X-axel i mätningar (antal)



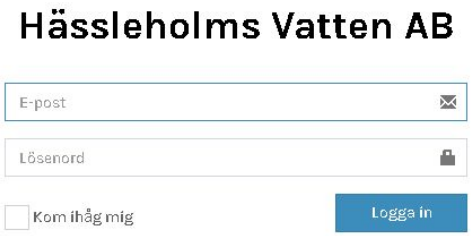
Figur. 22. Analog signalavläsning för ADC-chip, Y-axel i spänning (mV), X-axel i mätningar (antal)



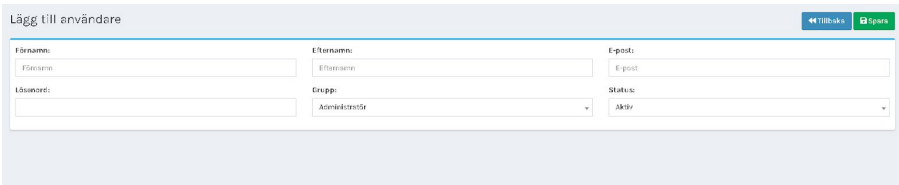
Figur. 23. “Flytande medelvärde”, ADC-chip, Y-axel i spänning (mV), X-axel i mätningar (antal)



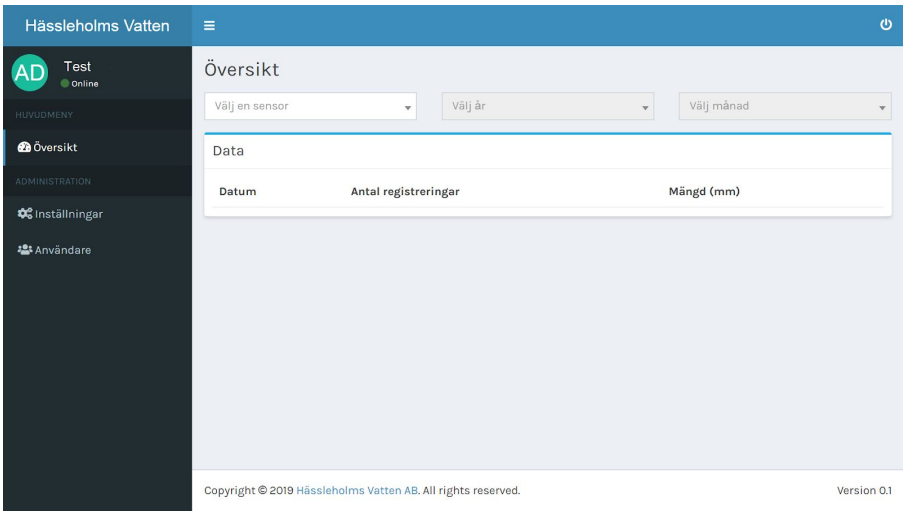
Figur. 24. “Flytande medelvärde”, inzoomat, ADC-chip, Y-axel i spänning (mV), X-axel i mätningar (antal)



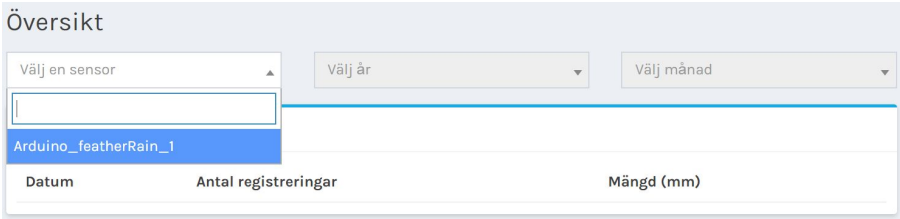
Figur. 25. Användargränssnitt, inloggning



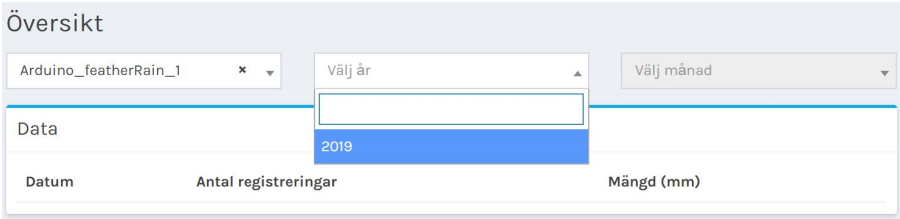
Figur. 26. Användargränssnitt, registrering av användare



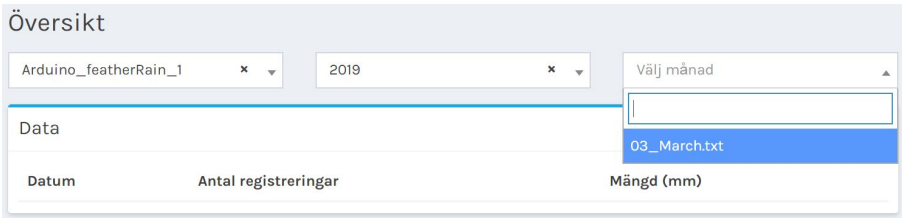
Figur. 27. Användargränssnitt, översikt av webbapplikationen



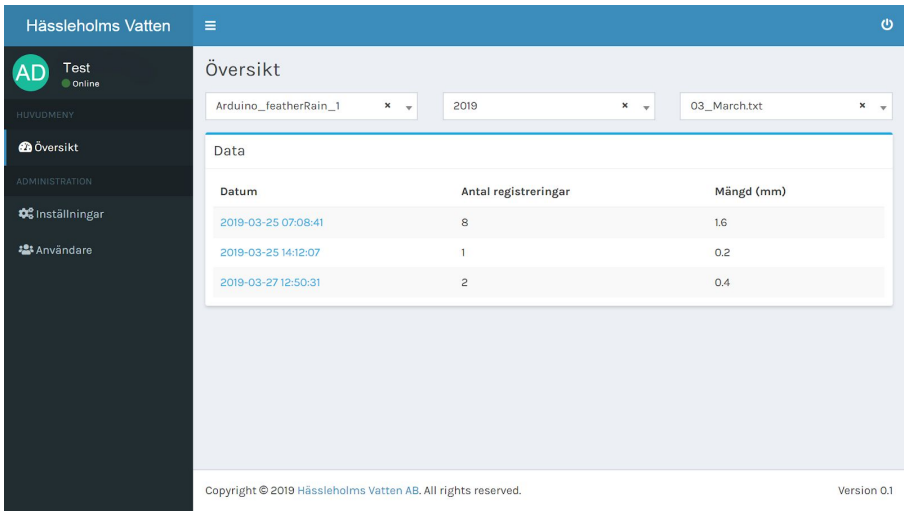
Figur. 28. Användargränssnitt, val av enhet (datalogger)



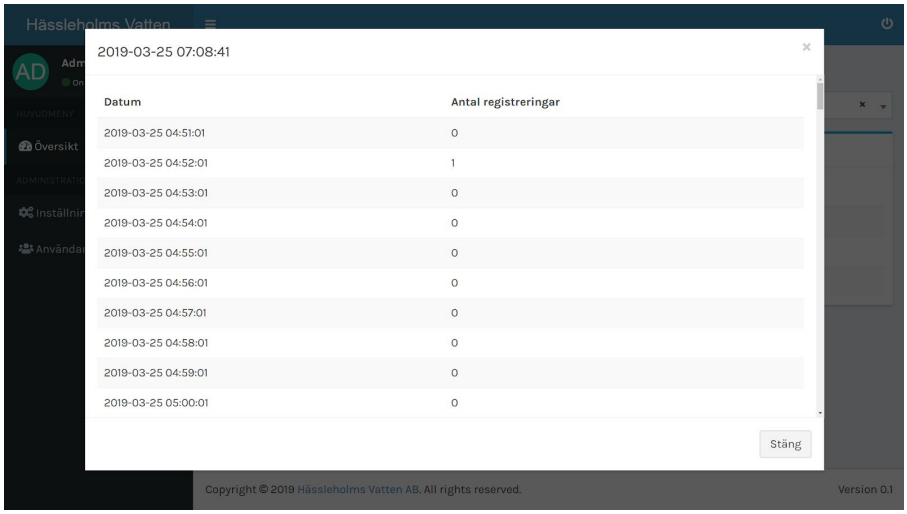
Figur. 29. Användargränssnitt, val av år för enhet



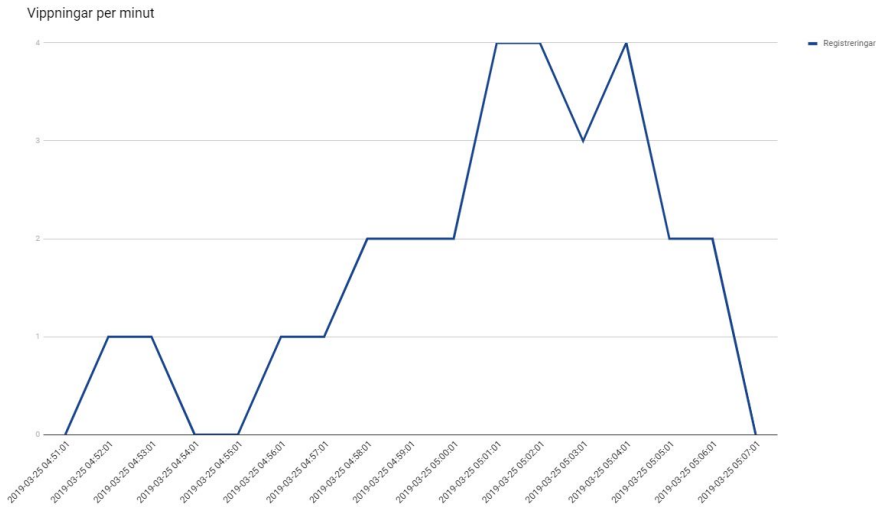
Figur. 30. Användargränssnitt, val av månad för enhet



Figur. 31. Användargränssnitt, lista med regn som har loggats för en månad



Figur. 32. Användargränssnitt, alla registreringar beräknade över minuter för en regnskur



Figur. 33. Användargränssnitt, hypotetisk kurva för minutvärden över ett regn

Printed by Tryckeriet i E-huset, Lund 2019

LUND
UNIVERSITY

Series of Bachelor's theses
Department of Electrical and Information Technology
LU/LTH-EIT 2019-700
http://www.eit.lth.se