Industrial Electrical Engineering and Automation

Feltolerant IoT för styrning av fontän

Albin Källner

Division of Industrial Electrical Engineering and Automation

Faculty of Engineering, Lund University

Feltolerant IoT för styrning av fontän

LUNDS
UNIVERSITET
Lunds Tekniska Högskola

LTH Ingenjörshögskolan vid Campus Helsingborg
Industriell Elektroteknik och Automation

Examensarbete:
Albin Källner

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 2020

Sammanfattning

Det här examensarbetet har gått ut på att kunna styra en fontän, hämta och spara data från en DUC samt att göra den så feltolerant som möjligt. Arbetet har utförts åt Stadsbyggnadsförvaltningen på Drift och Underhållsavdelningen i Helsingborg för att bespara dem pengar, tid, undvika skador på personal och åtkomst till data.

För att detta skulle vara möjligt behövdes en styrenhet som kan styra fontänen och ett sätt att hämta och lagra data. Styrenheten som användes var en DUC från företaget Bastec och för att kunna hämta data användes programmet Node-RED. Mycket av förstudierna gick till att lära sig hur DUC:en och Node-RED skulle programmeras.

Resultatet blev att DUC:en programmerades utefter kravlista och för att den skulle kunna skicka data till Node-RED användes protokollet Modbus TCP/IP. Data skickas sedan vidare till en Gmail där den kan sparas, hämtas och appliceras på andra tjänster. Fontänen gjordes feltolerant då fontänpumpen stängs av ifall en bestämd händelse inträffar.

Tanken är att vidareutveckla detta system och applicera det på fler fontäner i Helsingborg med ett flöde i Node-RED som styr hämtningen av all data från flera fontäner.

Nyckelord

DUC, Node-RED, feltolerant, Modbus TCP/IP, data, flöde

Abstract

The purpose of this scientific report has been to steer a fountain, collect and save its data from the control unit and to make it as fault tolerant as necessary. The work has been done for Stadsbyggnadsförvaltningen on Drift and Underhållsavdelningen in Helsingborg so they can save money, time, delay injuries and have access to data.

For this to be possible was a control unit needed for the steering of the fountain and a way to collect and store the data. The control unit that was used was a DUC from the company Bastec and to collect data Node-RED was used. Most of the pre-study went to learning how to program the DUC and Node-RED.

The result was a programmed DUC with the met requirements and for it to send data to Node-RED the protocol Modbus TCP/IP was used. The data is then sent to a Gmail where it can be stored, be retrieved and applied to other services. The fountain was also made as fault tolerant as necessary as it shuts down if certain event happens.

The idea is to further develop this system and apply it to more fountains in Helsingborg with a flow in Node-RED that controls the collection of data from several fountains.

Keywords

DUC, Node-RED, fault tolerance, Modbus TCP/IP, data, flow

Förord

Jag vill tacka Anders Söderberg som gav mig detta examensarbete och all lärdom jag fått från det.

Tack till Markus Andersson och Jonas från Bastec för all hjälp när det kom till programmeringen av DUC:en och att ni även på eran semester hjälpte mig om jag hade några frågor

Tack till Andreas Postl som också på sin semester fanns där ifall jag behövde hjälp med något.

Innehållsförteckning

1 Inledning **10**

1.1 Bakgrund **10**

1.2 Syfte **10**

1.3 Målformulering **10**

1.4 Problemformulering **10**

1.5 Motivering av examensarbetet **10**

2 Teknisk bakgrund **12**

2.1 Hur fontänen fungerade innan examensarbete **12**

2.2 Data under central (DUC) **12**

2.3 Modbus TCP/IP **13**

2.4 Node-RED **13**

2.5 Portvidarebefordran **14**

2.6 Infracontrol **15**

2.7 MQTT **15**

3 Metod **16**

3.1 Planering **16**

3.2 Förstudier **17**

3.3 Programmering av DUC **17**

3.4 Systemet i Bas2 ..... **18**

3.5 Programmering i Node-RED och datahämtning **19**

3.6 Driftsättning **21**

3.7 Källkritik **22**

4 Analys **23**

4.1 Förstudier **23**

4.2 DUC **23**

4.3 Node-RED 23
4.4 Sparning av data 25
4.5 Driftsättning 25
4.6 Feltolerant fontän 25
5 Resultat 27
5.1 Styra fontän efter kravspecifikation 27
5.2 Feltolerant fontän 31
5.3 Spara data från DUC 31
5.4 Kommunikationen 33
5.5 Driftsättning av fontän 35
6 Slutsats 36
6.1 Reflektion över etiska aspekter 36
6.2 Framtida utvecklingsmöjligheter 37
7 Terminologi 38
8 Källförteckning 39

1 Inledning

1.1 Bakgrund

Examensarbetet har utförts åt Stadsbyggnadsförvaltningen på Drift och Underhållsavdelningen i Helsingborg. Deras ansvar ligger i att planera, bygga och förvalta det offentliga rummet. Detta medför att sköta stadens gator, skog, parker med mera.

Genom att automatisera fontänerna i Helsingborg kommer uppdragsgivaren att spara pengar, tid, fördröja skador, optimera driften och kunna använda sparad data till diverse andra projekt. Om uppdragsgivaren vill att fontänen ska vara igång på en icke schemalagd tid kan han lägga till det via app eller hemsida istället för att behöva ringa och skicka ut personal för att göra det manuellt. Eftersom data från givare ska sparas kan det även användas till andra projekt till exempel om en vindmätare används till en fontän kan denna data även användas till att visa hur mycket det blåser i Helsingborg på den platsen i en annan tjänst. Fontänpumpen kommer att slitas mindre vilket leder till färre reparationer som leder till färre skador på servicetekniker eftersom enligt uppdragsgivaren skadar sig 1 av 100 och är borta upp till fjorton dagar.

1.2 Syfte

Syftet med examensarbetet är att automatisera en av Helsingborgs fontäner och att göra den så feltolerant som möjligt. Data från givare ska sparas och kunna appliceras på diverse tjänster.

1.3 Målformulering

Målen med examensarbetet är att kunna styra fontänen efter kravspecifikation, göra den feltolerant och kunna hämta data från DUC.

1.4 Problemformulering

* Vad händer om en givare slutar fungera?

* Hur skyddas fontänpumpen vid manipulation av givare?

* Hur kommer data att sparas och appliceras?

1.5 Motivering av examensarbetet

Helsingborgs fontäner ska automatiseras på grund av att staden har som mål att vara ett av Europas mest innovativa städer år 2022 samt besparing av pengar. När styrsystemet är implementerat kommer det att skicka ut larm när det är lågt vattentryck eller till exempel låg vattennivå och inträffar någon av dessa kommer även fontänpumpen att stängas av vilket leder

till att den inte kommer gå sönder. I dagsläget finns det inget system som stänger av pumpen eller skickar ut larm så om det skulle bli låg vattennivå kommer inte pumpen ha något vatten att pumpa ut vilket leder till att den kan gå sönder och då måste en ny pump köpas. Detta var ett av flera scenarion som kan inträffa, därför är det viktigt att ha ett system implementerat som kommer att undvika detta så det inte behövs lägga onödiga pengar på nya pumpar.

2 Teknisk bakgrund

2.1 Hur fontänen fungerade innan examensarbete

Innan implementering av styrning gjordes allt manuellt på fontänen. Av- och påsättning fick göras manuellt varje dag, liksom för påfyllning av vatten. Det gick inte heller att se hur mycket vatten som förbrukades i realtid då vattenmätaren uppdaterades en gång i månaden.

2.2 Data under central (DUC)

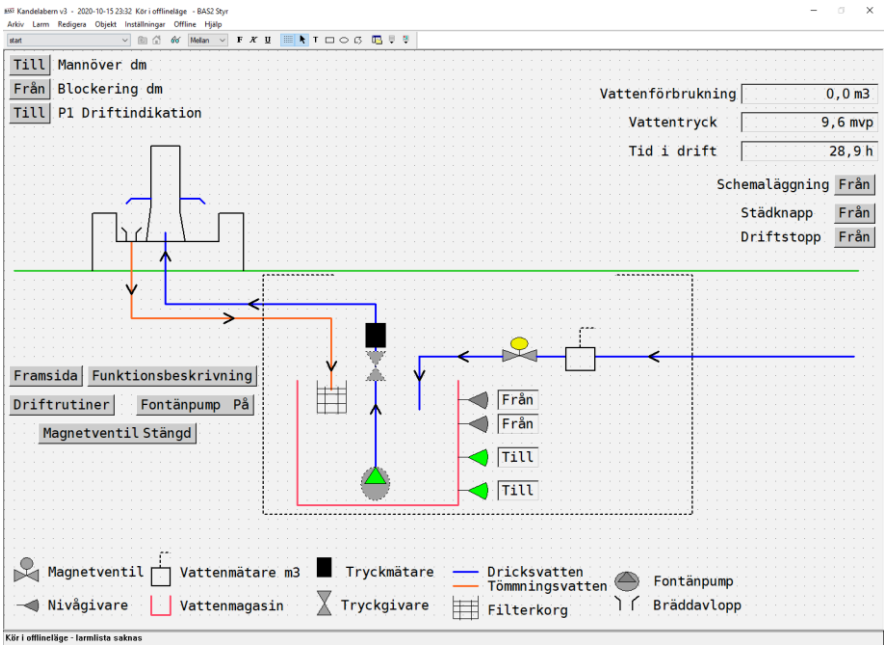
BAS2 XE16-COM är en DUC som oftast används inom byggnadsautomation för till exempel styrning av ventilation. Den har en inbyggd webbserver som får nätverksanslutning via 3G-modem eller LAN/WAN [1].

Den har också Modbus TCP/IP inbyggt och stödjer klient/server Mode (mer om det i kap 2.3). Detta gör det möjligt för DUC:en att kommunicera med andra DUC:ar eller med ett program på en PC [1].

Den har åtta ingångar som kan användas som digitala och analoga samt åtta utgångar därav fyra är digitala och fyra är analoga. Skulle det behövas fler in- och utgångar kan expansionsmoduler kopplas till [1].

Figur 1: BAS2 XE16-COM

Uppkopplingen behöver inte vara på plats utan anläggningen kan nås via internetanslutning, GSM-modem, 3G-modem eller telefonmodem [10].



Figur 2: Exempel på anläggning i Bas2

2.3 Modbus TCP/IP

TCP står för Transmission Control Protocol och IP står för Internet Protocol, tillsammans används de som ett transportprotokoll över internet. När dessa protokoll används för Modbus information kommer data att skickas till TCP där ytterligare information läggs till som sen ges vidare till IP. IP skickar iväg data i ett paket [2].

Modbus protokollet var skapat av Modicon 1979 för deras PLCs och industriella automationssystem. Sedan dess har protokollet blivit en industristandard för transport av I/O information [3].

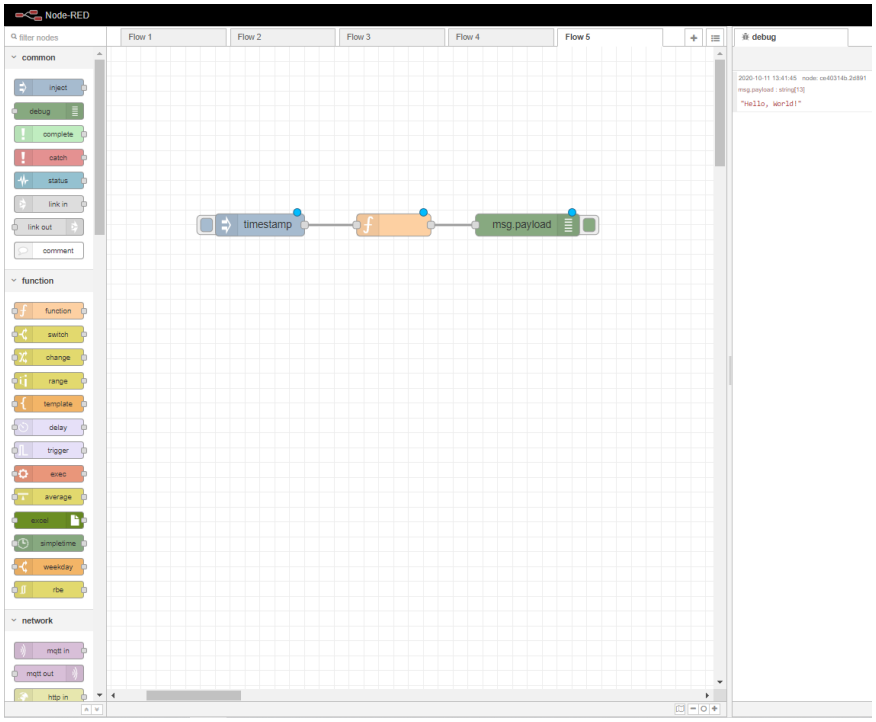
För att Modbus enheter ska kunna kommunicera med varandra använder de en klient-serverteknik där klienten (Node-RED) kan inleda en förfrågan. Den andra enheten som är en server (DUC) svara på förfrågan genom att leverera data till klienten [3].

En klients förfrågan innehåller en adress och funktionskod som talar om vilken funktion som ska användas [3]. Adressen säger vart i klienten informationen ska hämtas och funktionskod innebär om klienten kan skriva, läsa eller båda till adressen [4].

2.4 Node-RED

Node-Red är ett visuellt verktyg med ett webbaserat flödesprogram som bland annat kan användas till att skapa JavaScript funktioner eller koppla ihop hårdvara och internettjänster [5].

I Node-RED används flödesbaserad programmering och detta betyder att block används för att utföra en funktion. När blockets funktion har utförts går det vidare till nästa block i flödet. Det finns olika kategorier av block men de mest väsentliga är input, function och output. Input-blocken är de som startar flödet, function-blocket manipulerar informationen och skickar vidare den och output-blocken är det sista blocket i flödet som visar den slutgiltiga informationen [6].



Figur 3: Node-RED gränssnitt

I Figur 3 visas ett program i Node-RED. Gränssnittet är uppdelat i 3 huvudsakliga delar. Byggblocken är till vänster, flödet är i mitten och till höger är debug informationen. Blocket timestamp är ett input-block som startar flödet och skickar vidare dess information till nästa block som är ett function-block som kan ändra informationen från input-blocket och sedan skickas det till msg.payload som är ett output-block. Informationen visas sedan i debug-fönstret.

2.5 Portvidarebefordran

En port i en router kan användas för utgående trafik, inkommande trafik eller båda. Några standardportar som alltid är öppna är HTTP via port 80 och FTP via port 21. Anledningen till att öppna en ny port är för att kunna sortera nätverkstrafiken. När en ny port ska öppnas behöver den tilldelas en tjänst. Tjänsten kan vara ett spel eller program där portnumret redan finns tillgängligt för att öppnas i listan Tjänstnamn men om det inte visas i listan läggs portnumret till manuellt [7].

För att förtydliga öppnades en port i routern för att kunna skicka ut larm från DUC:en, en annan port öppnades för att skicka data med Modbus och en annan port öppnades för att anläggningen ska kunna visas på webbsida och app.

2.6 Infracontrol

Infracontrol är ett företag som är specialiserade på IT-lösningar och agerar som en oberoende systemintegratör. Helsingborg stad använder dem för bland annat driftövervakning och jobbet de utför är att hämta information från olika system. Deras roll i examensarbetet är att ta hand om driftlarmen från dessa system som skickas från Node-RED. Larmen samlas i Infracontrol Online och distribueras sedan till rätt person [8][9].

2.7 MQTT

MQTT står för MQ Telemetry Transport, protokollet skapades 1999 av anledningen att det behövdes ett protokoll för minimal batteriförlust och bandbredd. MQTT är en klient/server och publicera/prenumerera transportprotokoll. Det är lätt att implementera, öppet, simpelt och meddelanden som skickas väger väldigt lite [13].

Klienten som skickar meddelandet kallas för sändare och klienten som får meddelandet kallas för mottagare. Sändaren och mottagaren har aldrig kontakt med varandra utan kopplingen mellan dem sker via en broker. Brokerns jobb är att filtrera alla meddelanden och distribuera dem till rätt mottagare [14].

3 Metod

3.1 Planering

Ett av de första momenten i examensarbetet var att handledaren Anders Söderberg gjorde en kravlista på vad han ville att fontänens funktioner skulle vara. Den såg ut såhär:

- Städknapp
    - Knapp för att stänga av alla funktioner och larm
- Driftstopp
    - Samma som städknapp
- Säsongsavslut
    - Knapp för att stänga av fontän enligt ovan
- Säsongstart
    - Knapp för att aktivera alla larm och funktioner
- Nivågivare/magnetventil
    - Nivågivare ska reglera magnetventil för att bibehålla vattennivå
        - Öppna magnetventil vid låg vattennivå
        - Stänga magnetventil vid rätt vattennivå
        - Larm vid mininivå samt att fontänpump stängs av
        - Larm vid maxnivå
        - Kunna öppna och stänga magnetventil manuellt i Bas2
- Vattenmätare
    - Aktuell vattenmätarställning
    - Medelvattenförbrukning per dygn
    - Medelvattenförbrukning per vecka
- Pump/tryckgivare/Tidur
    - Vid lågt tryck ska pump stängas av
    - Kunna stänga av/på fontänpump via knapp i Bas2
    - Tidur ska reglera start och stopp av fontänpump
        - Ska kunna redigera tider, veckoschema
    - Visning pump av/på
        - Grön = Pump på, ska vara det enligt tidur, samt tryck i ledning
        - Gul = Pump av, ska vara det enligt tidur, samt inget tryck i ledning
        - Röd = Pump av, ska vara på enligt tidur

- Info om varför den inte är på

För att veta om detta var möjligt kontaktades Bastec och de tyckte kraven var rimliga.

Uppdragsgivaren ville även att data från DUC:en kan hämtas och sparas regelbundet. Den kravlistan såg ut såhär:

- Skicka larm till Infracontrol
- Medel vattenförbrukning per 24h
- Medel vattenförbrukning per vecka
- Total vattenförbrukning
- Info om pump är av/på
- Info om städ- och driftstoppknapp är till/från
- Skicka info till karttjänst

3.2 Förstudier

För att kunna styra fontänen användes Bastecs DUC och för att lära sig hur den ska programmeras bokades en genomgång på Bastecs kontor. Under genomgången gick dem igenom hur deras Bas2 program fungerade. När det var klart lånades en DUC och simuleringsplatta för att kunna rita och programmera anläggningen till fontänen, se Figur 4.

Figur 4: DUC och simuleringsplatta

För att lära sig att hur Node-RED fungerade användes information och guider från internetbaserade sidor som Youtube och Node-RED Forum.

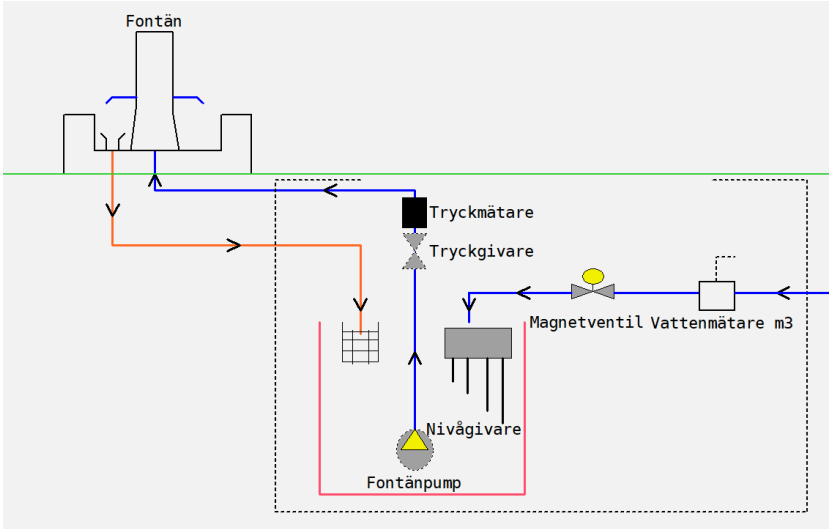
3.3 Programmering av DUC

En viktig del var att flödesbilden för fontänen skulle vara så användarvänlig att nästan vem som helst ska kunna koppla upp sig till den och förstå vad de kan göra och vad de olika symbolerna

betyder. Med detta i åtanke ritades flödesbilden så det klarligen visade en fontän, vart pumpen, magnetventil och givare var placerade samt hur vattnet cirkulerade. För att tydligt visa vad som var vad i ritningen gjordes en lista med namn till de olika symbolerna och ritningar. Därefter lades visuella och manuella funktioner till, som till exempel städknapp, driftstopp vattenmätarställning med mera, se Figur 11.

Det följande steget var att programmera DUC:en utefter kravlistan uppdragsgivaren hade givit. Utöver genomgången fanns det inte mycket information om hur den skulle programmeras vilket ledde till skapandet av många olika anläggningar för test och lärdom. Om det var något som var svårt hjälpte de anställda på Bastec till. Givarna är kopplade till DUC:ens in- och utgångar och funktionerna till dem programmerades med logiska uttryck.

3.4 Systemet i Bas2



Figur 5: Övergripande bild på anläggningen i Bas2

I Figur 5 visas en mer övergripande bild över hur givarna och andra komponenter är placerade. Det som kan göras i systemet manuellt är att kunna stänga av/på fontänpumpen, öppna/stänga magnetventilen för påfyllning av vatten och stänga av alla funktioner och larm via städ- och driftstoppsknapp. Funktion för att kunna schemalägga när fontänen ska vara igång finns också. Detta medför att uppdragsgivaren kan välja att fontänen ska vara på varje vardag mellan en viss tid men han kan även lägga till ytterligare scheman över en dag på till exempel helgen. För att se den fullständiga bilden i systemet kolla Figur 11 i resultatet.

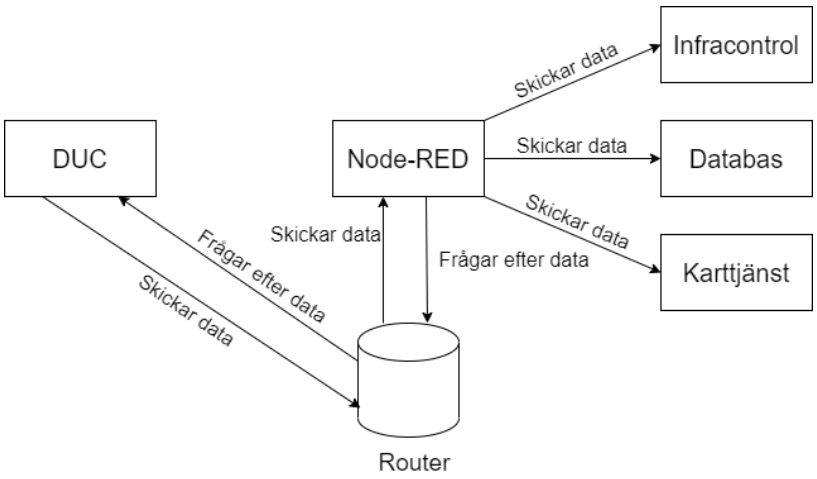
På nivågivaren visas dess fyra spröten i figuren. Dessa har programmerats för olika funktioner beroende på vattennivån i tanken. Det längsta sprötet är referens till vattennivån och om vattennivån skulle sjunka förbi det näst längsta sprötet kommet ett larm inträffa som säger att

det är låg vattennivå och fontänpumpen stängs av till servicetekniker har kvitterat det. Mellan de två sista spröten ska vattennivån regleras. Om vattennivån sjunker förbi det näst kortaste sprötet har magnetventilen programmets till att öppnas och stängas när vattennivån når det kortaste sprötet.

Tryckgivaren har programmerats för att stänga av fontänpumpen ifall trycket blir för lågt eftersom det kan skada pumpen i långa sträckor. Anledningen till att det är just lågt vattentryck är på grund av att folk häller diskmedel i fontänen enligt uppdragsgivaren.

3.5 Programmering i Node-RED och datahämtning

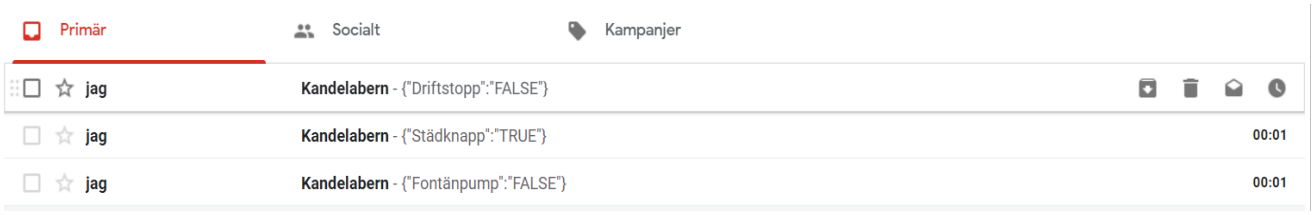
För att kunna hämta data behövde DUC:en och Node-RED kunna kommunicera med varandra. Detta sker via Modbus TCP/IP. Det första som gjordes var att ge DUC:en en fast IP-adress i nätverket och sedan öppna en port i routern där informationen mellan DUC:en och Node-RED kommer att skickas. Efter det fylldes IP-adressen och portnumret in i Node-RED som agerar klient till DUC:en som är server. För att kunna hämta data från DUC:en användes en lista med Modbus-adresser som överlämnades från Bastec. Se Figur 6 för hur kommunikationen fungerar.



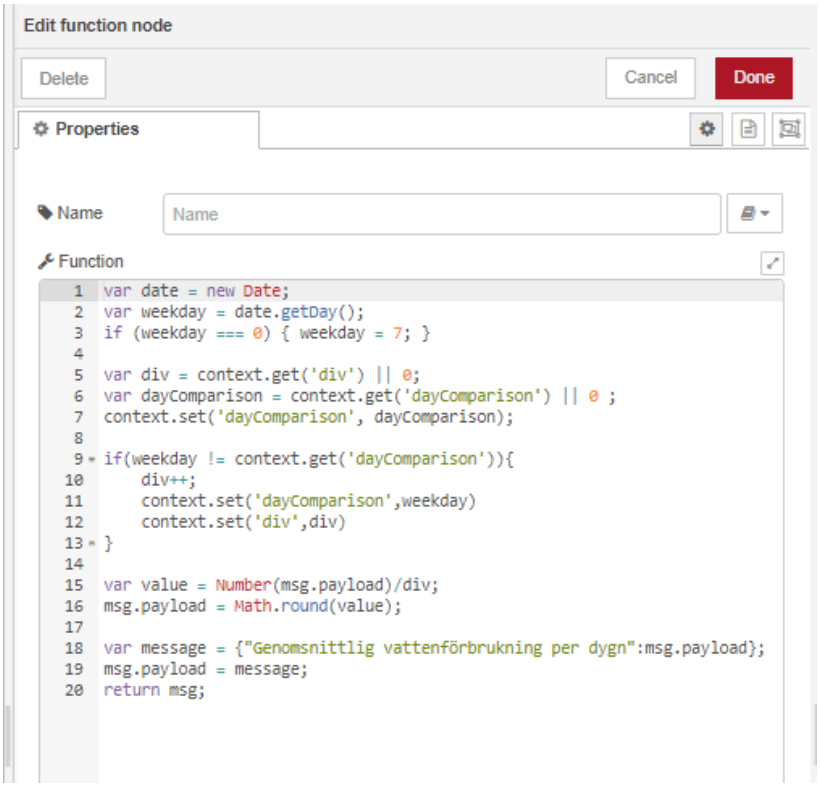
Figur 6: Hur kommunikationen fungerar

Det skapades tre Node-RED-flöden, ett som hämtar data uppdragsgivaren ville ha, ett flöde som skickar larm till Infracontrol och ett flöde som skickar data till kartjänst. Uppdragsgivaren ville ha statusen på fontänpumpen, städ- och driftstoppknapp, totalvattenmätarställning, genomsnittlig vattenförbrukning för varje dygn och genomsnittlig vattenförbrukning för varje vecka. Statusen på pumpen, städ- och driftstoppknappen löstes genom att använda deras Modbusadresser från DUC:en i Node-RED och varje gång flödet uppdateras kommer deras status att skickas från DUC:en till flödet och sedan vidare till en Gmail med statusen TRUE eller FALSE, se figur 7. Givaren som mäter hur mycket vatten som förbrukas visar den totala

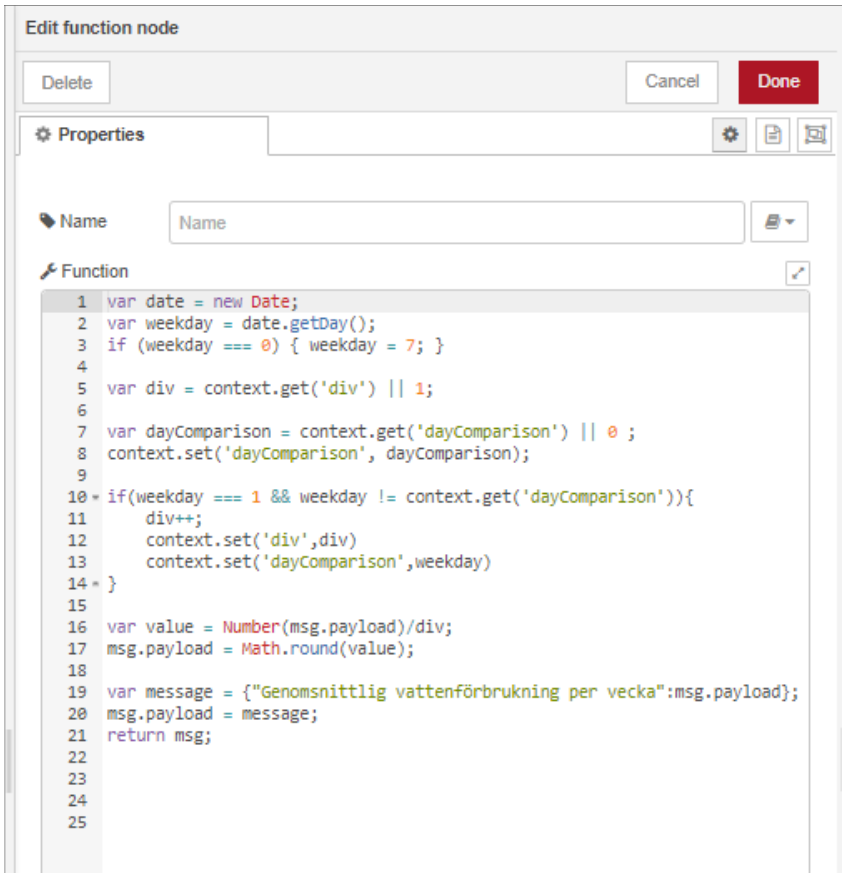
vattenförbrukningen så för att beräkna genomsnittet för varje dygn och vecka behövdes kod skrivas i ett function-block i Node-RED, se Figur 8 och 9.



Figur 7: Hur det ser ut i Gmail



Figur 8: Kod för beräkning av genomsnittlig vattenförbrukning per dygn



Figur 9: Kod för beräkning av genomsnittlig vattenförbrukning per vecka

När ett larm inträffar skickar DUC:en ut det till angivna personer via mejl eller sms men uppdragsgivaren ville också att larmen skulle skickas via Infracontrol. Detta löstes genom att använda Modbusadressen för varje larm i Node-RED som skickar statusen på larmet i JSON format till en Gmail. Eftersom flödet uppdateras i ett regelbundet intervall tillades att ett mejl bara kan skickas om statusen har ändrats på larmet. För att Infracontrol ska kunna använda sig av detta utvecklade dem ett eget program som kan hämta informationen från mejlet.

Uppdragsgivaren ville även att statusen av fontänpumpen, larm, städ-och driftstoppknapp ska visas i en karttjänst. Detta löstes på samma sätt som hur larmen skickas till Infracontrol.

3.6 Driftsättning

Under tiden som DUC:en programmerades installerades de givare som skulle användas på fontänen samt ett nytt elskåp med en ny DUC och router. När allt var klart överfördes anläggningen som gjordes under examensarbetet till den nya DUC:en för att styra fontänen.

3.7 Källkritik

[1] och [10] är hämtade från företagets hemsida som skapade produkten och borde då anses vara en säker källa eftersom det är deras egen produkt.

[2],[3] och [4] är säkra källor eftersom de säljer tjänster eller produkter som har med protokollet att göra. Det har även testats praktiskt under examensarbetet vilket visar på att deras information stämde.

[5] och [6] anses vara säkra eftersom deras information testats praktiskt under examensarbetet och stämt.

[7] anses vara säker eftersom det är ett stort företag med expertkunskap inom området samt att deras information har testats praktiskt under examensarbetet.

[8] och [9] är från företagets egen hemsida och borde därför anses som en säker källa då deras information måste stämma om deras egen tjänst.

[10] anses vara säker på grund av att författaren kontaktuppgifter finns med och att andra källor säger samma information.

[11] är säker eftersom de har det största det största nätverket i Sverige för högskoleutbildade ingenjörer samt att de är ett fackförbund för dem.

[13] och [14] är säker på grund av att MQTTs egna hemsida refererar till dem.

[15] är säker på grund av att det är en publicerad vetenskaplig arktikel som hittades genom LUBsearch.

4 Analys

4.1 Förstudier

Under förstudierna var det oklart hur data skulle hämtas eftersom ingen riktigt visste hur det skulle kunna gå till. Bastecs DUC har möjligheten att logga värden och sedan överföra dem till en Excel fil. Problemet med detta är att personal måste gå till elskåpet varje gång och hämta det manuellt från DUC:en till datorn. Detta var ingen bra lösning då uppdragsgivaren vill kunna se data vart han än var utan att behöva gå och hämta den på plats. När Bastec rådfrågades om detta för att se om de hade något förslag nämnde de Modbus TCP/IP protokollet som deras DUC stödjer. Problemet var att de har bara använt det för DUC till DUC kommunikation och inte med något separat program. Detta ledde till mycket sökningar om information av Modbus TCP/IP och om det fanns andra lösningar på hur data kan hämtas. Ett annat alternativ som kom fram var att försöka använda sig av en HTTP API för att hämta data via Node-RED från DUC:ens webbserver. Detta fungerade inte men efter många tester fungerade det med Modbus TCP/IP.

4.2 DUC

DUC:en behövdes göras till en server så att klienten (Node-RED) kunde hämta data från den. Problemet var att det inte fanns någon information hur detta skulle göras. Detta löstes genom att kontakta Bastec som sa att systemvariablerna i DUC:en behövdes ändras, dessa var Modbus slavport och Modbus slav via TCP.

Ett annat problem var hur Node-RED skulle veta vart i DUC:en till exempel statusen av städknappen låg. Efter påläsning av Modbus TCP/IP förstods det att en specifik adress behövdes men det gick inte att hitta i någon manual som Bastec hade. Dem kontaktades igen och de skickade en lista med alla adresser som fanns till DUC:en.

4.3 Node-RED

Vid val av program för att hämta data valdes Node-RED på grund av att det har ett användarvänligt system, det är gratis samt att det stödjer Modbus TCP/IP. Det finns liknande program som övervägdes att användas som till exempel n8n.io eller ThingsBoard. Anledningen till att dem inte valdes var på grund av att n8n.io inte stödjer Modbus TCP/IP och ThingsBoard kostar pengar.

Det förekom ett problem när den genomsnittliga vattenförbrukningen per dygn och vecka skulle implementeras. Givaren som mäter vattenförbrukningen uppdateras varje gång det fylls på med

vatten, det som visas i DUC:en är då den totala vattenförbrukningen under givarens livstid. För att kunna beräkna den genomsnittliga vattenförbrukningen per dygn och vecka var lösningen tvungen att revolvera kring den totala vattenförbrukningen. För att kunna beräkna den genomsnittliga vattenförbrukningen per dygn behövdes kod skrivas i ett function-block i Node-RED som jämförde de olika dagarna på veckan och delade den totala vattenförbrukningen med antalet dagar. För att beräkna det per vecka gjordes en liknande lösning men istället för att dela med antalet dagar delades den totala vattenförbrukningen med antalet veckor. Problemen som uppstod var hur Node-RED skulle kunna veta vilken dag det var och hur den ska kunna jämföra nuvarande dag med föregående så den vet om när en ny dag inträffar. Det sistnämnda löstes med context och det den gör är att den sparar ett värde och varje gång flödet uppdateras kollar context sitt nuvarande värde och jämför det med sitt föregående. Om värdena skiljer sig åt har något nytt inträffat. För att veta vilken dag det var användes objektet Date som håller koll på vilken dag på veckan det är och för att veta om det har blivit en ny dag jämfördes värdet av Date med en context av objektet Date. Exempel på hur det fungerar är om flödet i Node-RED uppdateras tio gånger per dygn kommer Date ha samma värde alla tio gånger och det har även context som jämförs med Date, men vid elfte uppdateringen är det en ny dag och då får Date ett nytt värde som inte är samma som context. Detta betyder att det är en ny dag och antalet dagar ökas som sedan delas med den totala vattenförbrukningen. Lösningen för att beräkna den genomsnittliga vattenförbrukningen per vecka löstes på liknande sätt fast istället för att jämföra om det är ny dag så ökar antalet veckor genom att jämföra varje dag med om det är en måndag, ifall det är en måndag är det en ny vecka och antalet veckor ökar.

4.4 Sparning av data

Hur data hämtas har etablerats genom Modbus TCP/IP men hur och vart data ska sparas var det olika metoder som övervägdes. Först övervägdes att använda en databas där data skulle sparas. Anledningen till att detta inte användes var på grund av att uppdragsgivaren inte hade någon tillgänglig för detta projekt och att göra en ny databas var inte angeläget då det kostar pengar och styrningen har inte implementerats på fler fontäner än så det hade varit onödigt med en databas för en enda fontän. Ett annat alternativ var att använda sig av MQTT där en broker skulle prenumerera på alla ämnen som till exempel statusen på fontänpumpen och den totala vattenmätarställningen. Värdena skulle sparas på HiveMQ som agerar broker och sen hade den publicerat vidare data till Infracontrol och karttjänst. Detta valdes inte på grund av att när detta testades kunde hämtad data visas i brokern men för att transportera vidare den behövdes en betald version av den. Detta ledde till att använda sig av en Gmail som agerar databas. Det är gratis och data sparas i form av mejl för evigt och det är lätt att hämta data med till exempel ett flöde i Node-RED som lägger det i en Excel-fil.

4.5 Driftsättning

Under driftsättningen överfördes anläggningen som gjorts under examensarbetet till den nya DUC:en som ska styra fontänen. När funktionerna testades var det problem med styrningen för påfyllning av vatten. Efter nedstigning under fontänen där givaren satt upptäcktes det att nivågivaren inte var installerad. Detta resulterade i att denna funktion inte kunde testas. Utöver det fungerade resterande funktioner och den anställde från Bastec som var med under driftsättningen sa att det inte borde vara några problem sen när nivågivaren är installerad.

Funktionen av att hämta data blev inte installerad under driftsättningen på grund av att det behövdes mer tid att komma på hur det ska installeras på bästa sätt. Ett alternativ är att lägga till en Raspberry PI i elskåpet som har Node-RED programmet installerat som hämtar och skickar iväg data.

4.6 Feltolerant fontän

Order feltolerant syftar på IT-system som inte slutar fungerar om ett fel skulle inträffa. Om de tekniska felen skulle bli för stora är det givetvis svårt att ha en avbrottsfri drift, men målet är att mindre fel inte ska stoppa driften. Ett feltolerant system kan också ha funktioner som stoppar driften om en bestämd händelse inträffar [11].

Fler och fler Internet of things system börjar integreras i världen och detta resulterar i mycket data som ska hanteras och lagras. Ju mer data som sparas desto mer ökar chansen att denna data innehåller fel information. I den vetenskapliga artikeln ”*To Verify the Correctness of IoT Sensor Data in Real-Time*” går Kamioka et al. (2019) igenom hur detta kan detekteras och hur en lösning kan se ut [15].

Deras arbete fokuserar på olika delar som orsakar fel data, dessa är Bias, Drift, Malfunction och Random. Bias och drift är typer av fel som kontinuerligt händer medan Malfunction och Random är tvärt om [15].

För att lösa dessa fel har dem byggt en konstruktion som är medveten om IoT-hårdvara samt IoT-applikationer. Konstruktionen har tre delar, Devices, Edge och Cloud. I första delen samlar givarna in information. I andra delen sparas denna information i till exempel en lokal databas men innan information når dit analyseras den för att se ifall den är korrekt. Ifall information inte är korrekt korrigeras den. Sedan i sista delen skickas det till ett Cloud [15].

Deras resultat blev att konstruktionen upptäckte och korrigerade fel data från givare på mindre än 50ms med en noggrannhet på 85% [15].

Eftersom examensarbetet system liknar mycket av vad Kamioka et al. (2019) har testat hade deras lösning på felaktig data kunnat vara användbart. I dagsläget är fontänen feltolerant genom att bestämda händelser är programmerade för att stänga av fontänpumpen tills servicetekniker har kvitterat det. Dessa händelser är om det blir låg vattennivå, lågt vattentryck och driftfel. Kamiokas et al. (2019) arbete hade gjort fontänen ännu mer feltolerant då driften inte behöver avslutas. Om det till exempel hade blivit låg vattennivå men det visar sig att givarens data inte stämmer kommer fontänpumpen att stängas av i alla fall. Men om deras konstruktion hade använts hade den inte behövts stängas av och kan istället fortsätta vara igång.

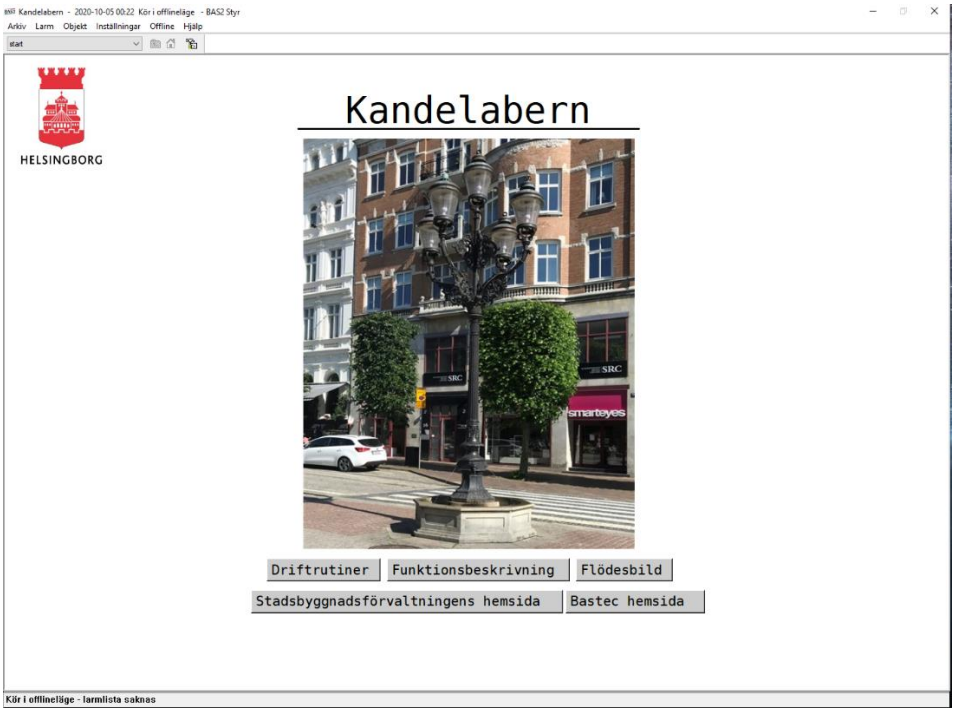
5 Resultat

5.1 Styra fontän efter kravspecifikation

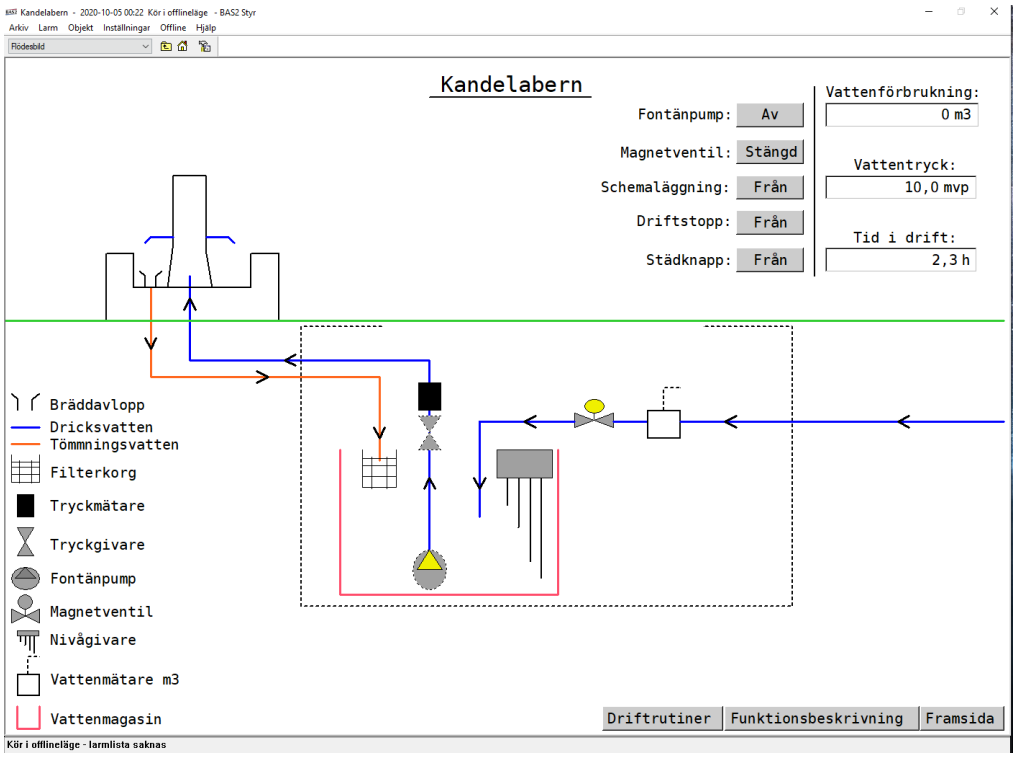
Nästan alla delar av kravspecifikationen blev implementerade. De funktioner som inte lades till var:

- Säsongsavslut
- Säsongsstart
- Larm vid maximal vattennivå

Anledningen till att säsongsavslut och säsongsstart inte lades till var på grund av att städ- och driftstoppsknappen har exakt samma funktion. Larm vid maximal vattennivå kunde inte läggas till eftersom nivågivaren inte hade tillräckligt många spröten (se Figur 11 för antalet spröten) för att det skulle vara möjligt.

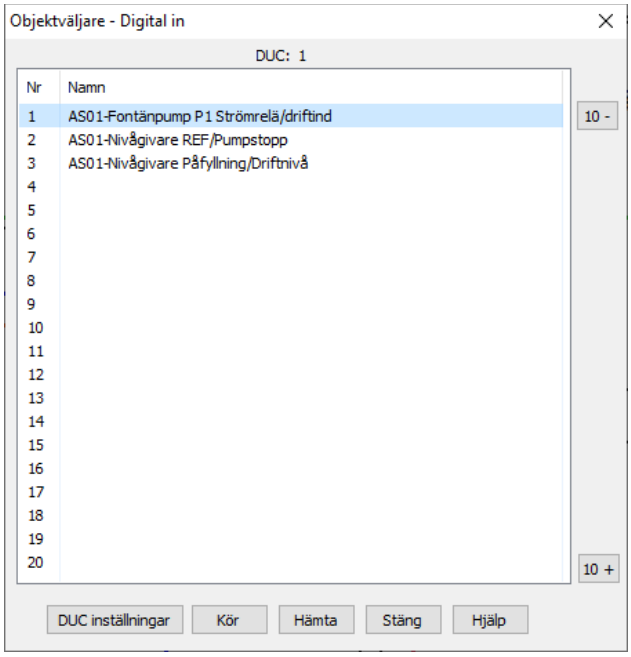


Figur 10: Framsida Bas2

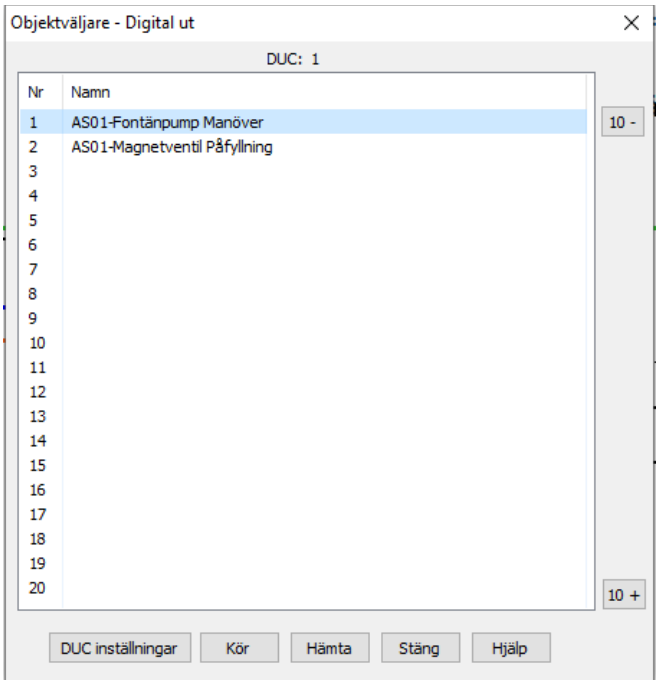


Figur 11: Flödesbild Bas2

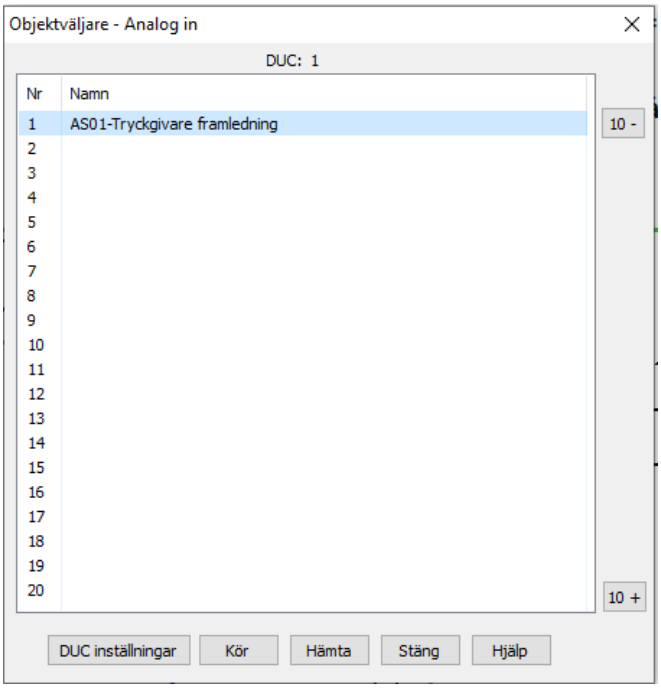
I Figur 10 och 11 visas den slutgiltiga designen av anläggningen till fontänen i Bas2-programmet. Det ser likadant ut i webbläsare och app.



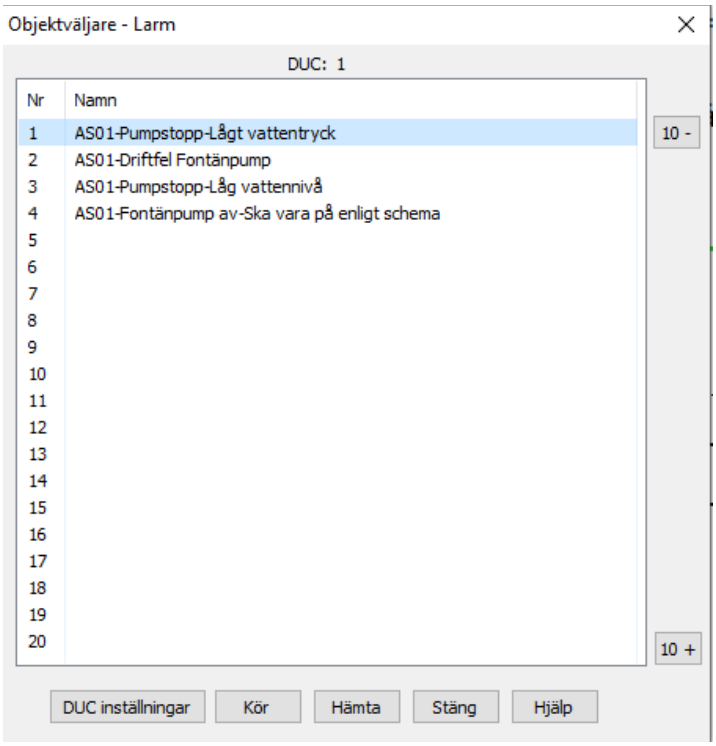
Figur 12: Digitala ingångar DUC



Figur 13: Digitala utgångar DUC

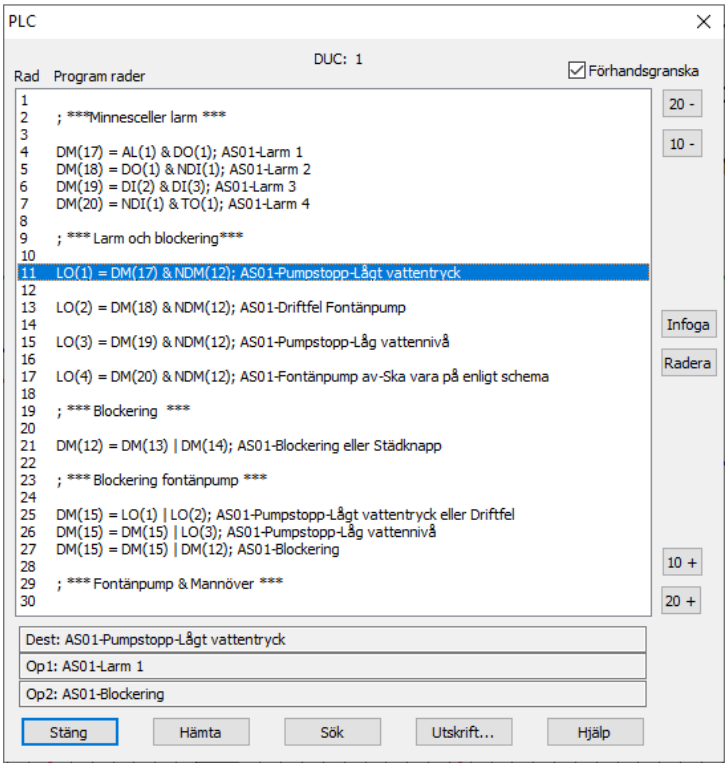


Figur 14: Analog ingång DUC

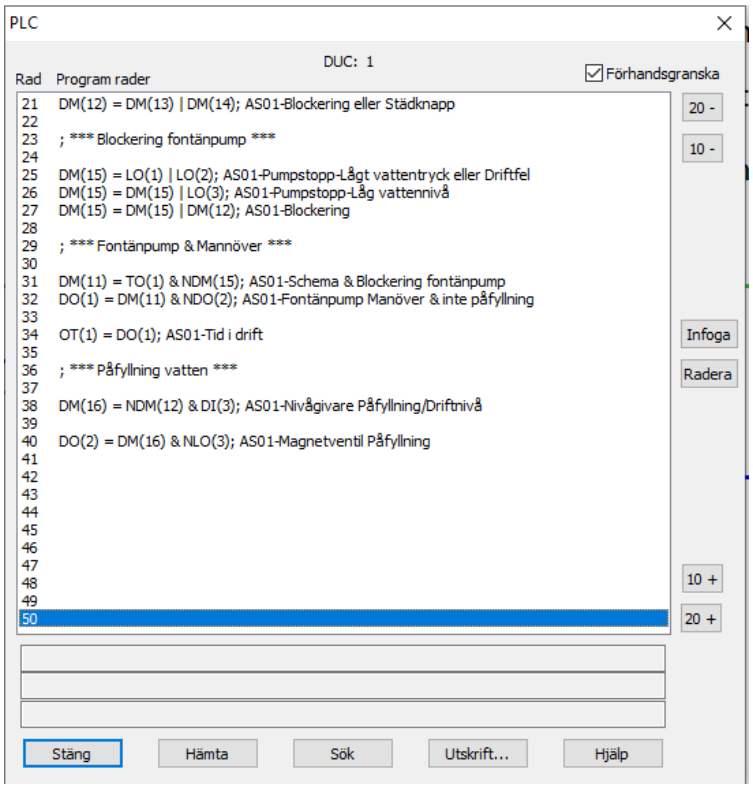


Figur 15: Larmlista DUC

I Figur 12, 13 och 14 visas de olika in- och utgångarna som använts för att styra fontänen. De analoga och digitala ingångarna programmeras för att styra utgångarna. I Figur 15 visas alla larmen som skapades i DUC:en.



Figur 16: Kod för DUC



Figur 17: Kod för DUC

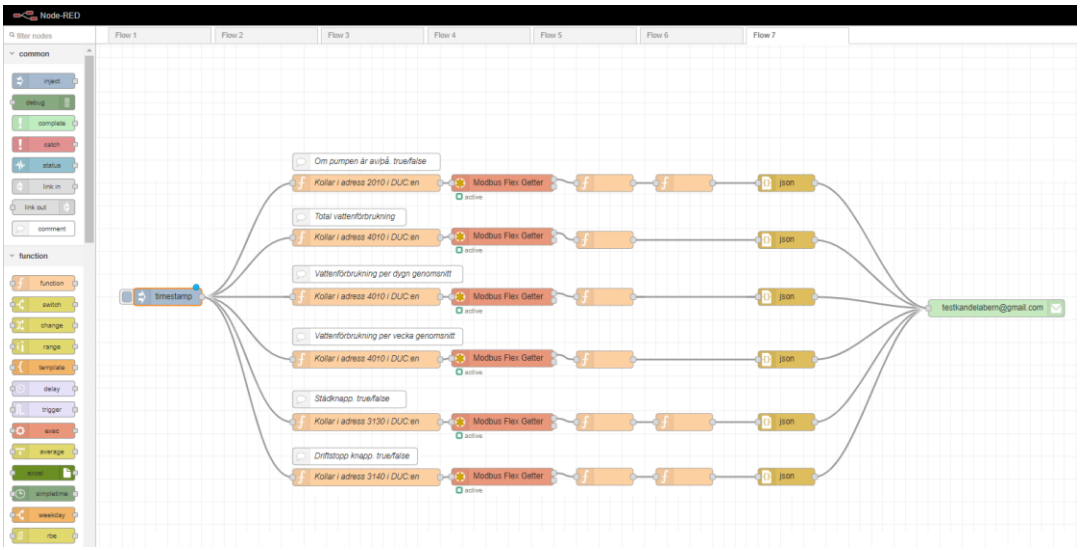
Koden i Figur 16 och 17 är koden i Bas2-programmet för styrning av DUC:en utifrån kravlistan.

5.2 Feltolerant fontän

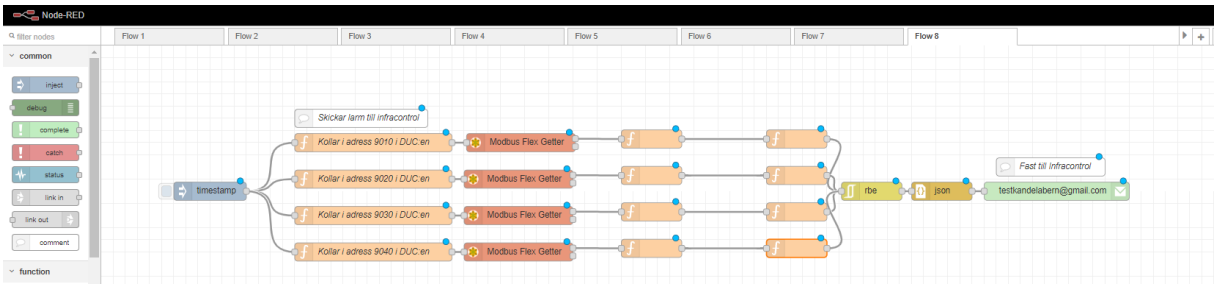
Fontänen gjordes så feltolerant som var nödvändigt. Den har programmerats för att stoppa driften ifall fel som lågt vattentryck, lågvattennivå eller driftfel inträffar.

5.3 Spara data från DUC

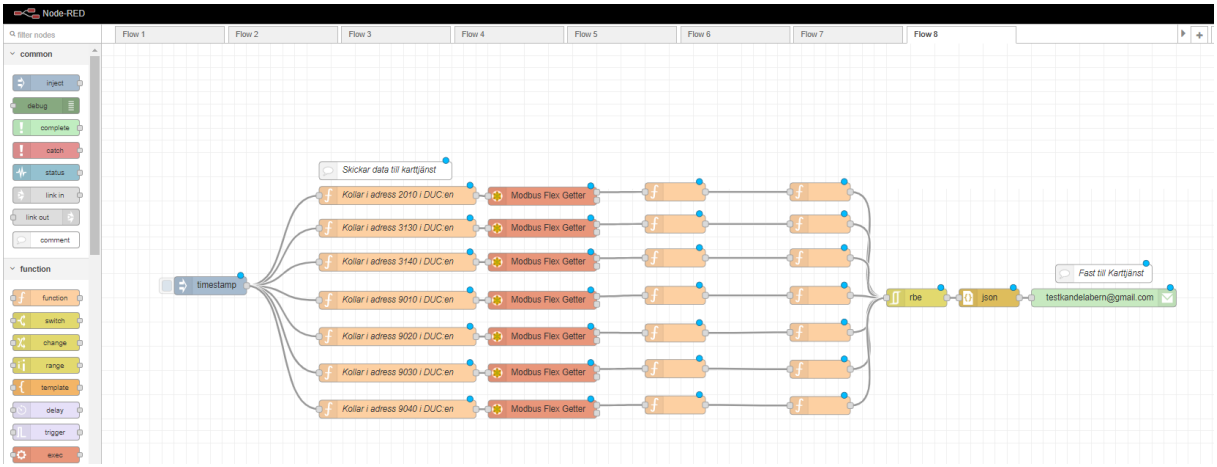
Det skapades tre flöden i Node-RED för att hämta och spara data från DUC:en, se Figur 18, 19 och 20. Ett var för data som uppdragsgivaren ville ha och de andra två för larmutskick till Infracontrol och data till karttjänst. Resultatet blev att data lyckades att hämtas och skickas vidare till en Gmail där data sparas.



Figur 18: Första flödet i Node-RED



Figur 19: Andra flödet i Node-RED

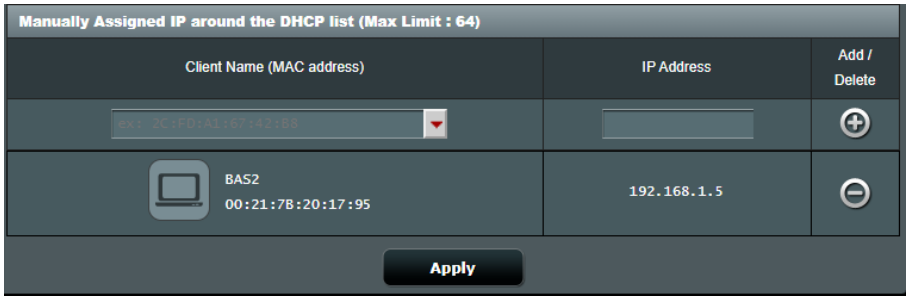


Figur 20: Tredje flödet i Node-RED

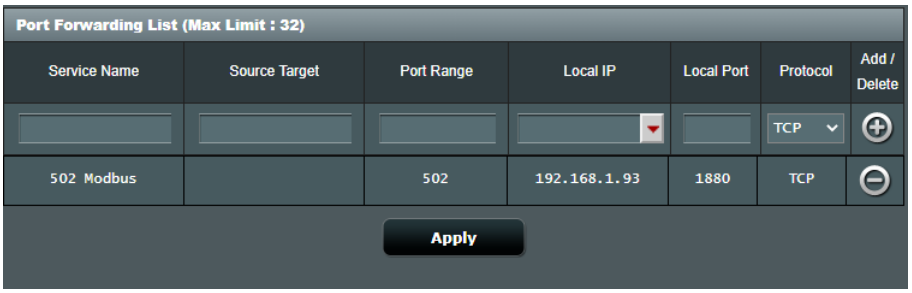
I den första figuren hämtas statusen på fontänpumpen, städknapp och driftstoppsknapp samt den totala vattenförbrukningen, medelvattenförbrukningen per dygn samt medelvattenförbrukning per vecka. Värdet görs sedan om till ett JSON-format och skickas till en Gmail, se Figur 7. I Figur 19 och 20 visas flödena för hämtning av data som skickas till

kartjänst och Infracontrol. Det första av dem är till Infracontrol och den skickar statusen av larmen och det andra flödet är till karttjänsten och den skickar statusen av larmen samt statusen på fontänpumpen, städ- och driftstoppsknapp.

5.4 Kommunikationen

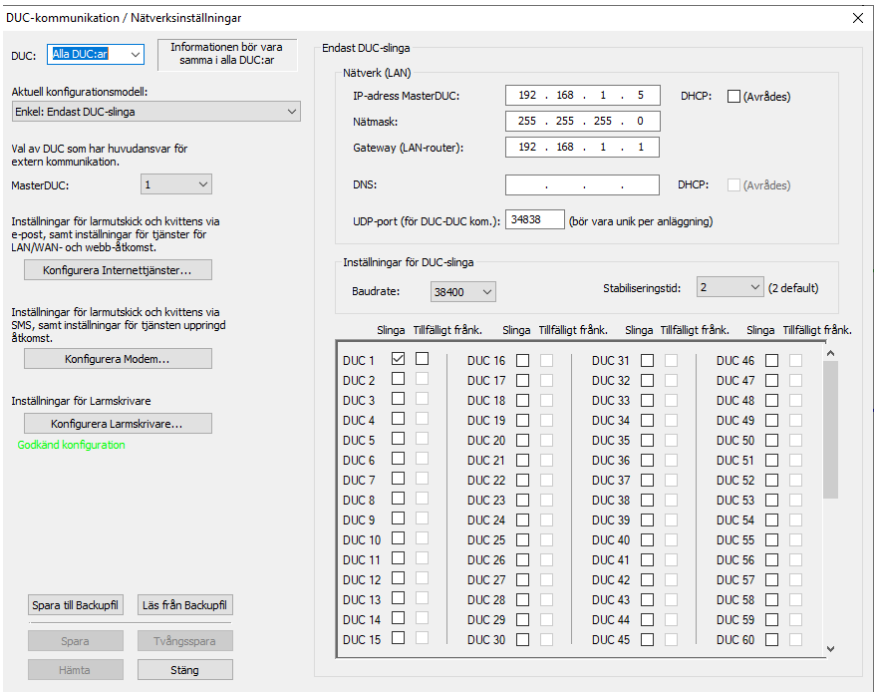


Figur 21: Fast IP i routerinställningar

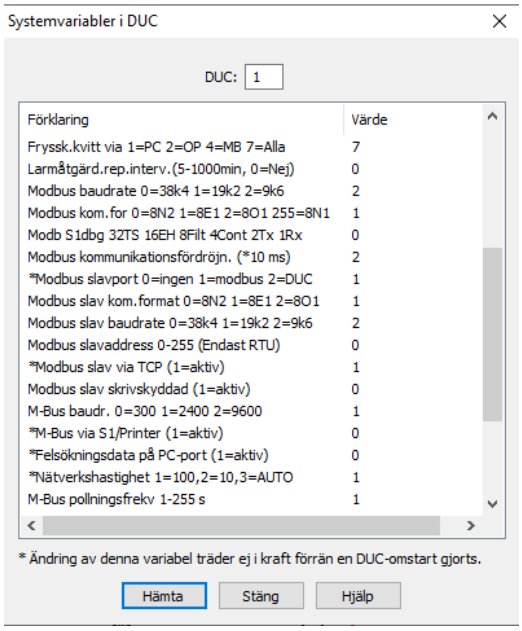


Figur 22: Portvidarebefordran i routerinställningar

För att det ska kunna gå att kommunicera med DUC:en behöver den en fast IP och detta gjordes via routerinställningarna som visas i figur 21. För att data ska kunna skickas från DUC:en via routern behövdes även en portvidarebefordran göras som visas i figur 22 där port 502 öppnas.



Figur 23: Nätverksinställningar DUC



Figur 24: Systemvariabler DUC

I figur 23 visas nätverksinställningarna i DUC:en och där fylls det fasta IP-numret, nätmask och gateway i för att DUC:en ska kunna skicka data, visas på webbläsare och app. För att DUC:en skulle kunna skicka data behövdes även systemvariablerna ändras som visas i figur 24. Variablerna som ändrades var Modbus slavport från 0 till 1 och Modbus slav via TCP från 0 till 1.

5.5 Driftsättning av fontän

I figur 25 visas fontänen som testas för de olika funktionerna under driftsättningen och som det visas på figuren kommer det vatten ur fontänen. Anledningen till att övre delen är borta och det kommer vatten ur de smala blåa slangarna är på grund av att någon hade kört på fontänen tidigare. Det gick även att styra fontänen via Bastecs app. I Figur 26 visas elskåpet där DUC:en är installerad som styr fontänen.

Figur 25: Fontänen

Figur 26: Elskåp med DUC installerad

6 Slutsats

Detta examensarbete har gått ut på att kunna styra en fontän med en DUC samt att kunna hämta data från den. Nedanstående diskuteras problemformuleringarna.

Vad händer om en givare slutar fungera?

Om en givare skulle sluta fungera, slitas bort eller bli kortsluten kommer texten ”Felaktigt värde, givare ej ansluten” att visas i anläggningen. Detta stoppar däremot inte DUC:en att försöka styra fontänen. För att undvika detta kan kod skrivas som ger TRUE ifall något fel skulle inträffa en givare och om detta händer ska fontänpumpen och resterande funktioner att stängas av.

Hur skyddas fontänpumpen vid manipulation av givare?

De givare som är installerade är tryckgivare och nivågivare. Om någon av dessa skulle bli manipulerade så de får felaktiga värden kommer fontänpumpen skyddas via larmen. Om tryckgivarens värde skulle vara väldigt lågt har DUC:en programmerats efter den händelsen så att ett larm skickas ut och när det har skickats kommer även fontänpumpen att stängas av till servicetekniker har kvitterat den. Samma sak gäller för nivågivaren om den skulle säga att det var låg vattennivå. Detta skyddar pumpen då om någon av givarna säger fel värde och att det inte stämmer är det förmodligen något på fel givaren eller så har någon tagit kontroll över den och gör det manuellt. I vilket fall skyddas pumpen då den stängs av och när servicetekniker kommer ut och ser att det egentligen inte har varit något fel kan detta hjälpa dem att komma till slutsatsen att det är fel på givaren eller att någon har manipulerat den.

Hur kommer data att sparas och appliceras?

Data sparas genom att Node-RED programmet hämtar den från DUC:en via Modbus TCP/IP och skickar vidare den till en Gmail som agerar likt en databas. Denna data kan uppdragsgivaren sedan hämta och applicera på till exempel vattenräkningar eller andra projekt som kan ha nytta av den.

6.1 Reflektion över etiska aspekter

Ingenjörernas hederskodex innehåller tio punkter som ser till att tekniken som används är för mänskligheten och samhällets bästa och att det förs vidare till följande generationer. Under examensarbetet har en punkt varit extra betydelsefullt [12].

- Ingenjören bör i sin yrkesutövning känna ett personligt ansvar för att tekniken används på ett sätt som gagnar människa, miljö och samhälle.

Detta examensarbete nyttjar människan och samhället då befolkningen i Helsingborg kommer kunna se när fontänen är igång eller varför den inte är igång via karttjänst istället för att behöva kontakta stadsförvaltningen för svar. Den är till nytta även för de människor som gör lagningar på fontänen då de nu minskar risken för att bli skadade eftersom pumpen kommer slitas mindre och de kan också stänga av alla funktioner på fontänen via städ- eller driftstoppsknapp så inget kan starta under reparation.

Examensarbetet gagnar också miljön genom styrningen för påfyllning av vatten. Eftersom DUC:en är programmerad till att reglera vattennivån kommer inte mer vatten än nödvändigt att användas. Om det skulle bli hål i vattenmagasinet kommer larmet låg vattennivå tillslut att inträffa vilket stoppar påfyllningen av vatten så att magnetventilen inte är öppen och försöker få vattnet till driftnivå onödigt länge.

6.2 Framtida utvecklingsmöjligheter

Att styra fontänen med en DUC visade sig fungera och det finns planer på att implementera det på fler fontäner. För att kunna hämta data från dem används Node-RED. Detta är däremot inte implementerat i dagsläget. Eftersom det kommer vara flera fontäner som data ska hämtas från är den mest optimala lösningen att ett flöde i Node-RED kan hämta data från alla fontäner istället för att varje fontän ska ha varsitt flöde. För att Node-RED ska kunna hämta data från DUC:en är ett alternativ att installera en Raspberry Pi i elskåpet så de delar samma nätverk och kan kommunicera. Detta är inte den mest optimala lösningen men det skulle förmodligen fungera. En annan lösning som potentiellt hade kunnat fungera är att använda sig av en Cloud tjänst som Node-RED kan vara installerad på som kommunicerar med DUC:en och då behövs inte längre någon Raspberry Pi.

Data som hämtas skickas till en Gmail där den sparas. En utveckling av detta hade varit att data skickas direkt till en Databas där andra tjänster som Infracontrol kan ha tillgång och hämta den data de behöver.

7 Terminologi

DUC – Data Under Central

Node-RED – Verktyg för flödesbaserad programmering

Modbus – Protokoll för sändning av data

Anläggning - Programmet som görs i Bas2

Portvidarebefodran - Öppnar port i router

JSON - Textbaserat format

Data - Information som skickas och sparas

8 Källförteckning

[1] Bastec - Datablad XE16-COM https://www.bastec.se/download/bas2-xe16-com-datablad/?wpdmdl=722&refresh=5f2adf8fcb8ab1596645263 [2020-08-05]

[2] Simply Modbus – Modbus TCP/IP http://www.simplymodbus.ca/TCP.htm [2020-09-24]

[3] Acromag - Introduction to Modbus TCP/IP https://www.prosoft-technology.com/kb/assets/intro_modbustcp.pdf [2020-09-26]

[4] Modbus 101 – Introduction to Modbus https://www.csimn.com/CSI_pages/Modbus101.html [2020-09-26]

[5] IoTEDU - What is Node-RED? https://iot4beginners.com/what-is-node-red/ [2020-10-04]

[6] Automatiserar - Guide: Kom igång med Node-Red och hemautomation https://www.automatiserar.se/guide-hemautomation-med-node-red/ [2020-10-04]

[7] Netgear – Hur konfigureras portvidarebefordran? https://kb.netgear.com/sv/22603/Hur-konfigureras-portvidarebefordran [2020-10-05]

[8] Infracontrol – Driftövervakning https://www.infracontrol.com/infracontrol-online/driftovervakning/ [2020-10-05]

[9]Infracontrol – Vår nutid https://www.infracontrol.com/om-oss/ [2020-10-05]

[10] Bastec – Teknisk information om BAS2 https://www.bastec.se/support/teknisk-info-om-bas2/ [2020-10-06]

[11] IT-ord – feltolerant https://it-ord.idg.se/ord/feltolerant/ [2020-10-06]

[12] Sveriges Ingenjörer - Hederskodex https://www.sverigesingenjorer.se/om-forbundet/sveriges-ingenjorer/hederskodex/ [2020-10-25]

[13] HiveMQ – Introduction to the MQTT Protocol https://www.hivemq.com/blog/mqtt-essentials-part-1-introducing-mqtt/ [2020-11-07]

[14] HiveMQ – Publish & Subsribe https://www.hivemq.com/blog/mqtt-essentials-part2-publish-subscribe/ [2020-11-07]

[15] A. L. Nguyen, E. Kamioka and T. Nguyen-Duc, "To Verify the Correctness of IoT Sensor Data in Real-Time," https://ieeexplore-ieee-org.ludwig.lub.lu.se/document/9026398?arnumber=9026398 [2020-11-09]