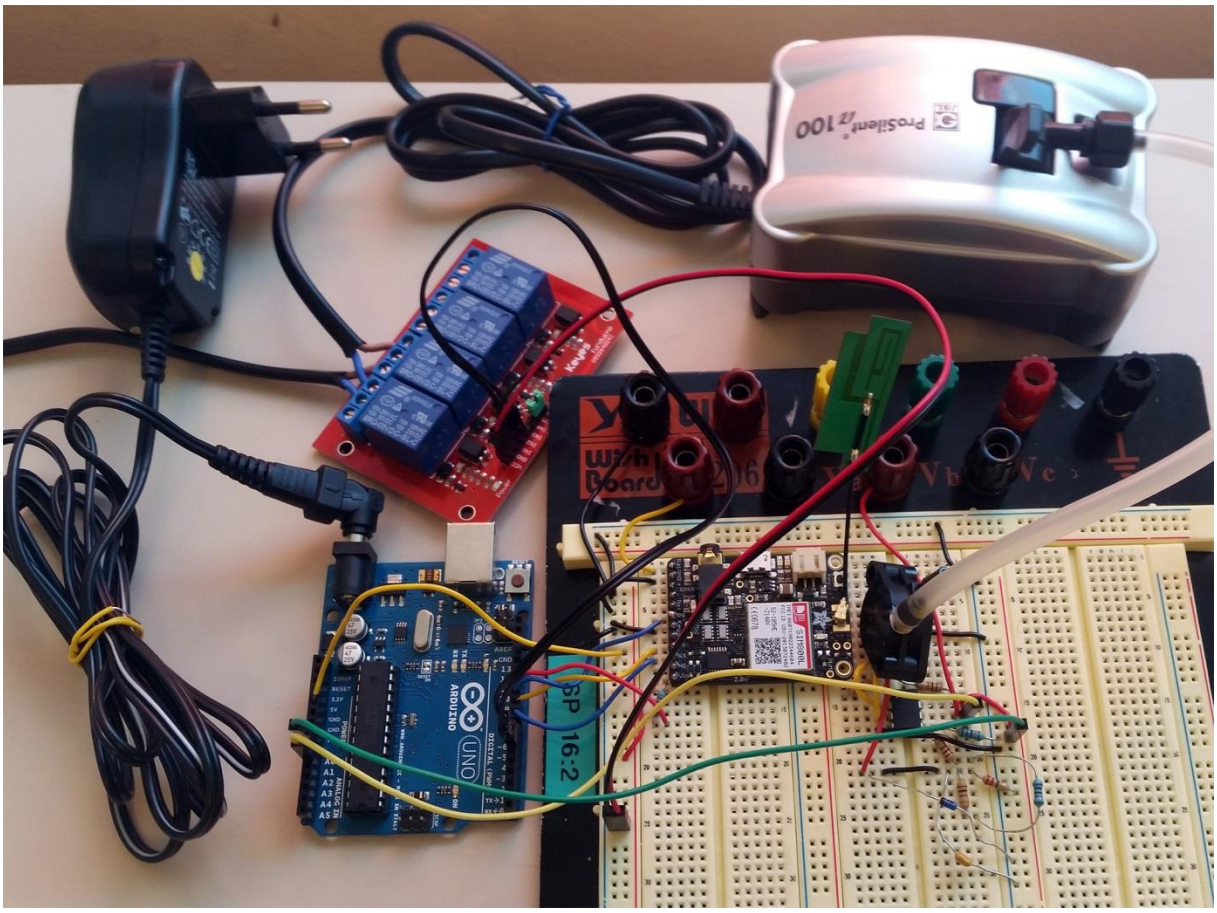
CHALMERS



Överföring av mätdata över IP-nätverk

Examensarbete inom högskoleingenjörsprogrammet Elektroingenjör

Eric Groseclos
Anders Hansson

Institutionen för Signaler och system
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2015

Överföring av mätdata över IP-nätverk
Examensarbete inom högskoleingenjörsprogrammet Elektroingenjör
ERIC GROSECLOS
ANDERS HANSSON

Institutionen för Signaler och System
Chalmers tekniska högskola
SE-412 96 Göteborg
Sverige

Telefon +46 (0)31-772 1000

FÖRORD

Ett examensarbete har utförts under våren 2015. Detta är den avslutande delen i utbildningen Elektroingenjör på Chalmers tekniska högskola. Arbetet motsvarar 15 högskolepoäng av utbildningens totala 180 högskolepoäng.

Vi vill tacka Manne Stenberg som har varit vår handledare under denna tid.

Tack till Sakib Sistek, som har försett oss med komponenter.

Eric Groseclos och Anders Hansson, Göteborg den 13 juni 2015

SAMMANFATTNING

Den globala temperaturhöjningen sker i allt snabbare takt. Skador på såväl land som hav och djurliv kommer att uppstå. Därav följer bland annat stora finansiella åtaganden. För att minimera dessa skador krävs system, såsom ”early warning system” där till exempel vattennivåförändring, koldioxidkoncentration med mera snabbt kan registreras och göras tillgängliga via moderna IP-nätverk. På så sätt kan man undersöka fysikaliska storheter utan att befinna sig på mätplatsen. Detta arbete har tagit fasta på denna utmaning. Arbetet skedde vid Chalmers Lindholmen och utfördes för Institutionen Signaler och System. Mikrokontrollern Arduino Uno hade de egenskaper som krävdes för arbetet. Till denna anslöts en GSM-modul med tillhörande antenn. De fysikaliska storheterna som mättes var temperatur och vattennivå. För mätning av vattennivå användes en tryckgivare och luftpump. För mätning av temperatur användes en digital temperaturgivare. En hemsida konstruerades, där mätdata redovisas i form av en graf. Användaren har möjligheten att styra mätintervallet samt kalibrera systemet genom att ange det på hemsidan.

ABSTRACT

The global temperature increase occurs at a faster pace. Damage to land, sea and animal wildlife will occur. It will require major financial commitments. To minimize these injuries, systems such as the "early warning" system will be required. Such a system can measure water level change and carbon dioxide concentration. This can quickly be recorded and made available through a modern IP network. Thus, one can examine physical quantities without being on the measurement site. The goal of this work was to solve the mentioned problem. The solution was a development platform that transferred data via an IP network. The work was done at Chalmers University and conducted for the Department of Signals and Systems. The microcontroller Arduino Uno had the capabilities required. This device was connected to a GSM module and its associated antenna. The measured values were temperature and water level. For measuring the water level, a differential pressure sensor and an air pump were used. For measuring temperature, a digital temperature sensor was used. A website was constructed, in which data are observed in the form of a graph. The user has the possibility to control the measurement interval by entering a value on the website.

INNEHÅLLSFÖRTECKNING

BETECKNINGAR

A/D = Analog till digital

AT = Attention Commands

FTP = File Transfer Protocol

GPRS = General Packet Radio Services

GSM = Global System for Mobile Communications

HTML = HyperText Markup Language

IP = Internet Protocol

M2M = Machine to Machine

PHP = Hypertext Preprocessor

SIM = Subscriber Identity Module

SQL = Structured Query Language

USB = Universal Serial Bus

1 INLEDNING

1.1 Bakgrund

I dagens samhälle är efterfrågan på trådlösa givare stort. Det ligger i tiden att presentera data på Internet, vilket gör det lättillgängligt för användaren att se givardata var man än befinner sig. Ett exempel på detta är att bevaka havsnivåhöjning, koldioxidkoncentration i havet, sjöar och floder. Vätskenivå i brunnar, tankar och reservoarer är andra exempel. En annan beteckning på trådlös överföring av mätdata är telemetri. Man kan även tala om M2M (maskin till maskin) och telematik.

Det kan vara relevant att styra olika mätparametrar, till exempel hur ofta mätningarna ska ske. Kommunikation åt andra hållet, det vill säga att givaren styrs via webbservern, är också möjlig. Eftersom detta system är en utvecklingsmiljö, finns möjlighet för påbyggnader. Utvecklingsmiljön kan anses vara både billig och enkel.

1.2 Syfte

Syftet är skapa en utvecklingsmiljö för överföring av mätdata. Det som mäts är vattennivå och temperatur. Mätdata fås via givare som skickas över IP-nätverk. Webbservern som tar emot data presenterar innehållet med hjälp av en graf. Det skall vara så pass standardiserat att fler givare kan läggas till utan förhinder. Med detta menas att utvecklingsmiljön ska kunna ändras utan att dess funktion försämras. Användaren ska kunna ange ett mätintervall, som systemet ska anpassa sig efter.

1.3 Avgränsningar

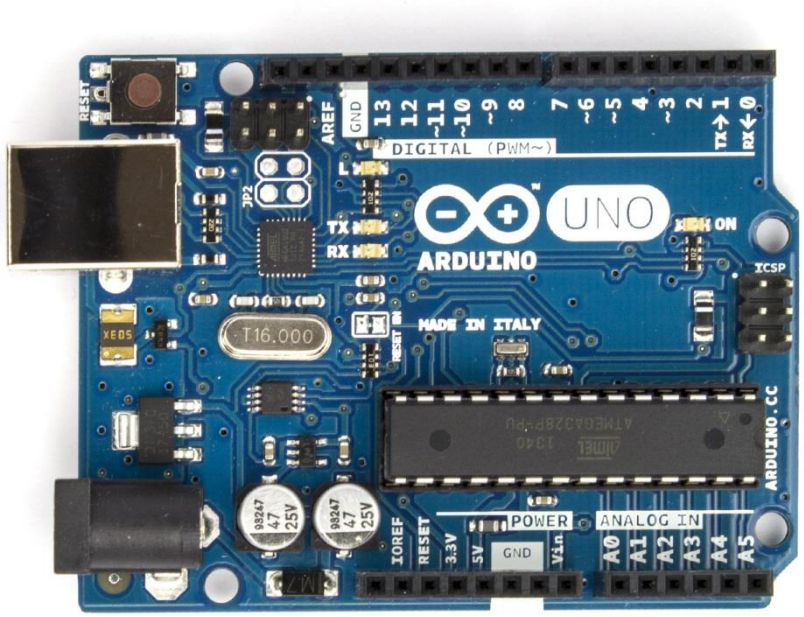
I huvudsak kommer arbetet att fokusera på att få fram en fungerande funktionsmodell. Ingen färdig produkt kommer att utvecklas. Denna funktionsmodell har till uppgift att demonstrera och exemplifiera vad som kan göras i utvecklingsmiljön. Antalet givare kommer att vara två stycken. De ska mäta temperatur respektive vattennivå.

1.4 Precisering av frågeställningar

- Är det säkert att överföra data via GPRS?
- Går det att tillförlitligt mäta vattennivå med en luftpump och tryckmätare?
- Går systemet att styra via webbservern?
- Vilket överföringsprotokoll lämpar sig för små datamängder?
- Är det enkelt att koppla in fler givare till utvecklingsmiljön?

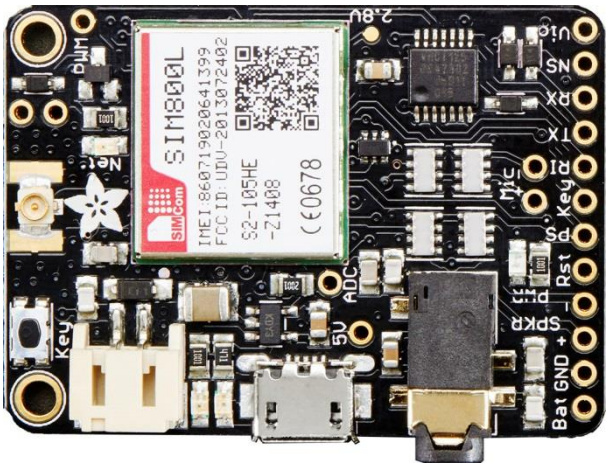
2 TEKNISK BAKGRUND

Hårdvaran i detta projekt kretsar kring Arduino Uno. Arduino Uno är en mikrokontroller med öppen källkod och hårdvara. Den är baserad på ATmega328, som är en integrerad krets [1]. Uno programmeras via en USB-kabel. Det programmeringsspråk som används är C/C++. I den mjukvara som hör till Arduino finns en monitor, där man kan observera den seriella kommunikationen. Strömförsörjning sker också via USB-kabel eller med 6-20 V externt. Kortet har sex analoga ingångar och 14 digitala in- och utgångar. A/D-omvandlaren omvandlar spänningar mellan 0 V och 5 V. Den har en upplösning på 10 bitar, det vill säga ett värde mellan 0 och 1023.



Figur 2.1 Mikrokontrollern Arduino Uno [1]

GSM-modulen ansluter och kommunicerar med svenska mobilnätverket, mer specifikt 2G-nätverket [2]. Modulen behöver en matningsspänning mellan 3,4 V till 4,4 V. Spänningskällan behöver även klara kortvariga strömmar upp till 2 A. Till GSM-modulen kan ett SIM-kort och en antenn anslutas. SIM-kortet som används är ett kontantkort från operatören Comviq. Kontantkortet måste laddas minst en gång per år för att kortet inte ska bli inaktiverat. Varje dataöverföring och SMS kostar pengar. Dataöverföringen kostar 0,35 öre per megabyte och 99 öre per SMS [3]. Det går att kontrollera saldot samt ladda på kontantkortet via Comviqs hemsida.



Figur 2.2 GSM-modul [4]

GSM-modulen har inbyggda funktioner såsom SMS-funktion samt FTP-tjänst. För att skicka data använder GSM-modulen sig av GPRS. Enligt telekombolaget Nokia innehåller GPRS två säkerhetsfunktioner: Abonnentverifiering och datakryptering [5]. Det existerar en särskild nyckel, kallad ”Ki”, som säkerhetsfunktionerna härrör från. Denna nyckel lagras på SIM-kortet samt hos operatören. Med denna nyckel krypteras data från enheten och dekrypteras hos operatören.

3 METOD

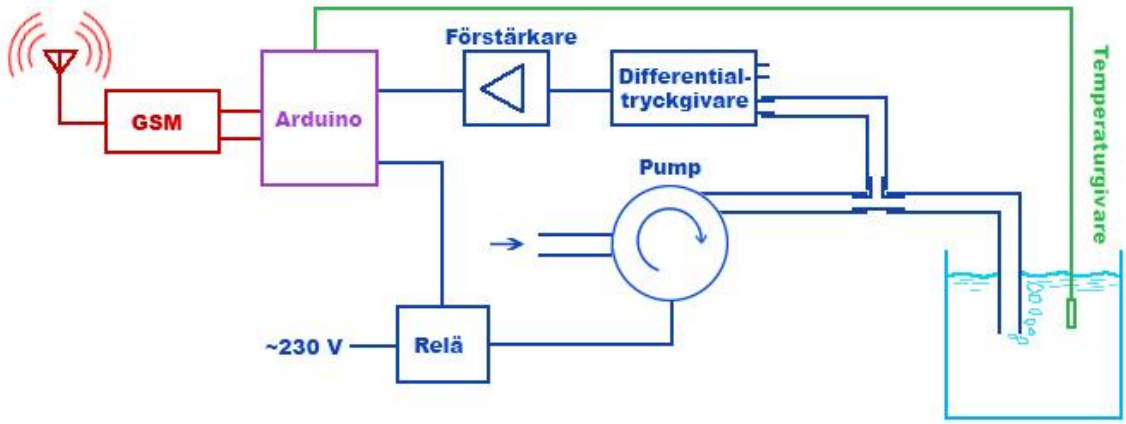
Arbetet utfördes i en laborationssal på Chalmers Lindholmen. Där fanns de komponenter och mätutrustning som behövdes. Detta arbete fokuserade främst på konstruktionsarbete. Systemet togs gradvis fram. När ett segment var klart övergick man till nästa del. Varje del som blev klar funktionstestades. Att bygga i sektioner gör det lättare att felsöka om det uppstår felaktigheter. Kunskaper hämtades från Internet vid behov. I huvudsak var det programmeringskunskaper som inhämtades. Även datablad studerades för att få förståelse för specifika komponenter. I programmeringen experimenterades det fram lösningar, det vill säga med trial-and error metoden.

4 ARBETSGÅNG

I början av arbetet byggdes en förstärkarkoppling med matningsspänningen ±9 V. Förstärkarkopplingens uppgift är att förstärka utspänningen från differentialtryckgivaren. Den nämnda givaren testades genom att blåsa i den, och sålunda generera en tryckskillnad. En utspänning skapades, som kunde verifieras med multimeter. Därefter kopplades Arduino Uno och GSM-modulen samman. GSM-modulens inbyggda FTP funktioner testades genom att programmera Arduino Uno. Mätdata laddades upp till en webbserver, i form av A/D-omvandlade spänningar.

En hemsida skapades sedan med HTML- och PHP-kod. Kunskaper om dessa programmeringsspråk var begränsande och information fick inhämtas från Internet. PHP-kod skrevs så att mätdata som befinner sig på webbservern automatiskt lagras i en databas. Att lagra data i en databas är fördelaktigt för grafpresentation på hemsidan. Sålunda kunde en graf göras för att redovisa data. Efter detta hade gjorts lades fokus på kommunikationen från hemsidan till Uno. En textbox skapades på hemsidan, där användaren kan ange intervall. Detta intervall sparas i en textfil på webbservern. Arduinon programmerades sedan för att inhämta denna textfil. Värdet i textfilen styr hur länge Uno väntar innan nästa mätning sker. Därefter implementerades en SMS funktion i Arduinon. Genom att skicka ett meddelande till GSM-modulen kan färska mätvärden skickas till användarens mobiltelefon. Även en mobilapplikation skapades. Denna applikation hade syftet att läsa av mätvärden från hemsidan.

Efter att detta hade gjorts klart lades fokus på de analoga komponenterna. Ett tvåkanaligt relä kopplades in till systemet. Därefter anslöts en akvariepump med slang till differentialtryckgivaren. Förstärkarkopplingen fick göras om till enkelmatning på 12 V. Temperaturgivaren kopplades in till Arduino Uno. Därefter lades en grund till kalibrering av vattennivåsystemet. Med detta gjort blev systemet komplett. Till sist prövades hela systemet genom att använda en bägare med vatten som testobjekt. Nivån kunde verifieras med tumstock. Figur 4.1 visas en förenklad bild av hela systemet.



Figur 4.1 En illustration som redogör för systemets huvudsakliga utseende.

5 ARDUINO UNO OCH GSM-MODUL

Mätdata som Arduino inhämtar skickas över GSM-nätet. För att göra detta anslöts Arduino Uno till en GSM-modul. Kommunikationen sker seriellt mellan dessa två enheter med en baudrate på 4800 Bd. Under utvecklingsstadiet strömförsörjdes Uno via USB-kabeln och GSM-modulen via ett nätaggregat. I slutet av arbetet strömförsörjdes Arduinon externt med en adapter på 12 V.

Flera färdiga bibliotek utnyttjas i programmet. Dessa är OneWire, AdaFruit_Fona och SoftwareSerial. OneWire används till temperaturmätning. AdaFruit_Fona används till att kommunicera med GSM-modulen. SoftwareSerial används till den seriella kommunikationen mellan enheter. Arduino Uno styr GSM-modulen via instruktioner, så kallade ”AT-kommandon”. Dessa kommandon överförs som textsträngar. Varje överförd textsträng ger respons, som användaren kan observera via monitorn. I koden finns en egen funktion som har till uppgift att hantera AT-kommandon. Grunden till funktionen hämtades från Internet [6]. Fördelen med denna funktion är att det existerar en timeout och att den returnerar responsen från GSM-modulen.

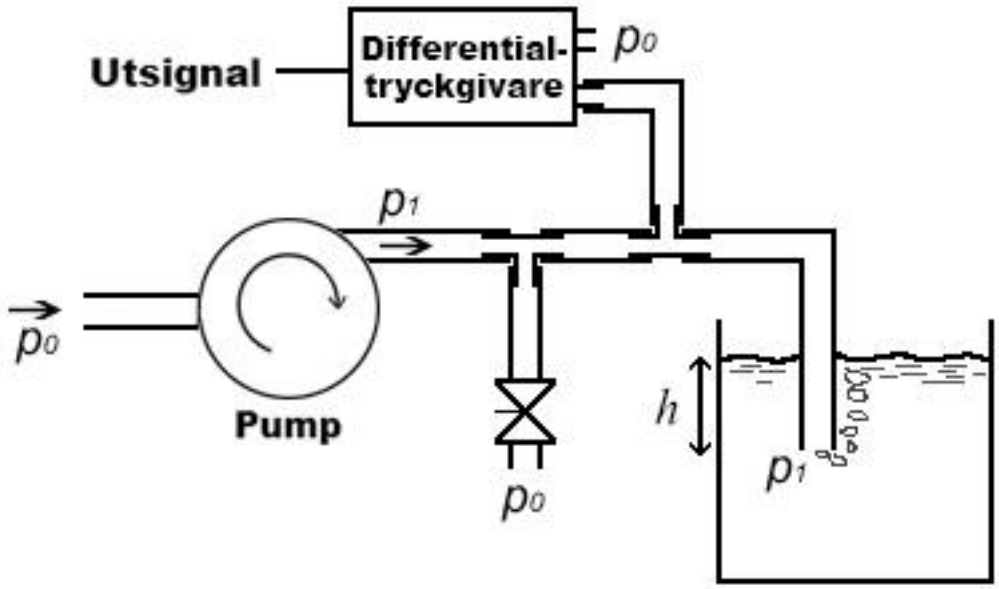
För att få täckning anslöts en separat antenn till GSM-modulen. Mätdata skickas med överföringsprotokollet FTP via GPRS. GSM-modulen har en inbyggd funktion som tillåter detta protokoll. Denna komponent ansluter till webbserverns FTP. Mätdata överförs till FTP:n med en textfil, vilket innehåller temperatur- och nivådata. Den överförda textfilen skriver över den gamla. Sålunda existerar inte minneskapacitet på FTP:n. Efter en lyckad uppladdning läser Uno av intervalltextfilen på FTP:n. Innehållet i denna fil styr hur lång tid det går mellan varje mätning.

Till GSM-modulen finns möjligheten att skicka SMS. När ett SMS mottags av GSM-modulen kontrolleras första bokstaven i meddelandet. Första tecknet i meddelandet måste vara bokstaven Q, för att säkerställa användaren. Därefter mäts aktuella mätvärden. Dessa mätvärden skickas tillbaka i ett SMS till användaren. Då SIM-kortet har begränsad lagringskapacitet, raderas meddelandet direkt efter. I bilaga 10.7 kan man observera resultatet av SMS-funktionen.

6 SYSTEMETS GIVARE

Detta system mäter två fysikaliska storheter, temperatur och vattennivå. Temperaturgivaren som användes är digital och skickar mätdata seriellt till Arduino. Den är dessutom vattentät för att kunna mäta vattentemperaturer. Mätfelet är ±0,5 grader Celsius inom temperaturintervallet -10 till 85 grader Celsius [7]. Upplösningen på temperaturen valdes till 12 bitar, som har precisionen 0,0625 grader Celsius. Det intervall som finns tillgängligt är mellan 9 och 12 bitar. Upplösningen kan justeras efter användarens behov. Ju högre upplösning, desto långsammare går mätningen. 12 bitars upplösning ger exempelvis en mättid på 750 ms. Arduino omvandlar inkommande mätdata till enheten Celsius. Detta värde delas upp i två heltal. Det ena är för heltalsvärdet. Det andra är för decimalvärdet. Innan överföringen sker slås de två nämnda talen samman med vattennivåvärdet till en textsträng. Sålunda innehåller en textsträng alla mätvärden. En 4,7 kΩ pull-up resistor anslöts mellan Arduino och temperaturgivaren. Signalen blir aldrig odefinierad i en sådan koppling.

För att mäta vattennivå användes en luftpump och en differentialtryckgivare. Denna metod kallas bubbelrörsmetoden [8]. Figur 6.1 illustrerar metoden. I figuren kan man observera sammankopplingen mellan differentialtryckgivaren, pumpen och mätobjektet.



Figur 6.1 Bilden illustrerar bubbelrörsmetoden. Variabeln h är höjden som mäts, relativt vattenytan. Luftpumpen tränger kontinuerligt undan vatten. Detta skapar ett mottryck, p_1, vilket differentialtryckgivaren mäter i förhållande till atmosfärstrycket p_0. Någon ventil anslöts aldrig under arbetets gång. Den skulle ha haft uppgiften att kalibrera systemet.

Den undanträngda vattenpelarens tryck ger ett mottryck, som tryckgivaren registrerar. Under arbetet användes en akvariepump för att tillföra luft i en slang. Slangen är kopplad till tryckgivaren via en T-koppling. Därifrån går slangen ner i vätskan som ska mätas. Pumpen körs i intervall och aktiveras bara när nivån ska mätas. Strömförsörjningen till pumpen är 230 V. Via ett 2-kanaligt relä kan spänningen slås till och från. Två kanaler användes för att säkerställa att reläet inte leder ström. För att styra reläet användes två digitala utgångar från Arduino Uno. Strömförsörjningen till reläet är 5 V, vilket fås från Uno.

Instrumentförstärkaren INA126 används till att förstärka spänningen från tryckgivaren. Denna komponent behöver en matningsspänning på cirka 12 V, vilket fås ifrån Arduinon. Tryckgivaren mäter tryck upp till 50 kPa och pumpen kan ge ett maximalt tryck på 19 kPa. Ett tryck på 1 kPa motsvarar en vattenpelare på cirka en decimeter. Sålunda klarar luftpumpen att tränga undan en vattenpelare på 1,9 m. Förstärkaren är designad att klara tryck upp till 28,5 kPa, vilket motsvarar 5 V. Detta betyder att förstärkarkopplingen valdes till att ha en förstärkning på 145 gånger, se ekvation (1) nedan. Ekvationen erhölls från databladet för INA126. Variabeln G är förstärkningen och R_G är den valda resistansen som styr förstärkningen. Anledningen till att klara tryck upp till 28,5 kPa var att kunna möjliggöra en starkare luftpump i framtiden.

$$G = 5 + \frac{80000}{R_G} = 5 + \frac{80000}{570} = \ 145\ gånger \qquad (1)$$

Referensspänningen på INA126 är satt till 1,6 V. Det betyder att utspänningen aldrig går under 1,6 V. Sålunda undviker man att spänningen blir negativ. Då komponenter kan ändra sina värden med tiden, finns det en god säkerhetsmarginal. Spänningen från förstärkaren skickas till Arduinos A/D-omvandlare. Förstärkarkopplingen kan observeras i bilaga 10.5. Spänningen konverteras i A/D-omvandlaren till en skala mellan 0 och 1023. Med den valda referensspänningen och aktuell pump fås ett värde i intervallet 326-450. A/D-omvandlingen är programmerad att ske 20*200 (4000) gånger. Det tar cirka en halv sekund att genomföra omvandlingen. Ett medelvärde beräknas utifrån dessa mätvärden. Detta fick göras eftersom luftpumpens flöde varierar med tiden. Tryckgivarens utsignal anslöts till oscilloskop, där variationen kunde observeras. Det erhållna medelvärdet konverteras till enheten centimeter. Detta åstadkoms genom att först subtrahera medelvärdet med 326 på grund av referensspänningen. För att erhålla korrekt nivå multipliceras värdet sedan med 0,44. Det teoretiska värdet kan observeras i ekvation (2).

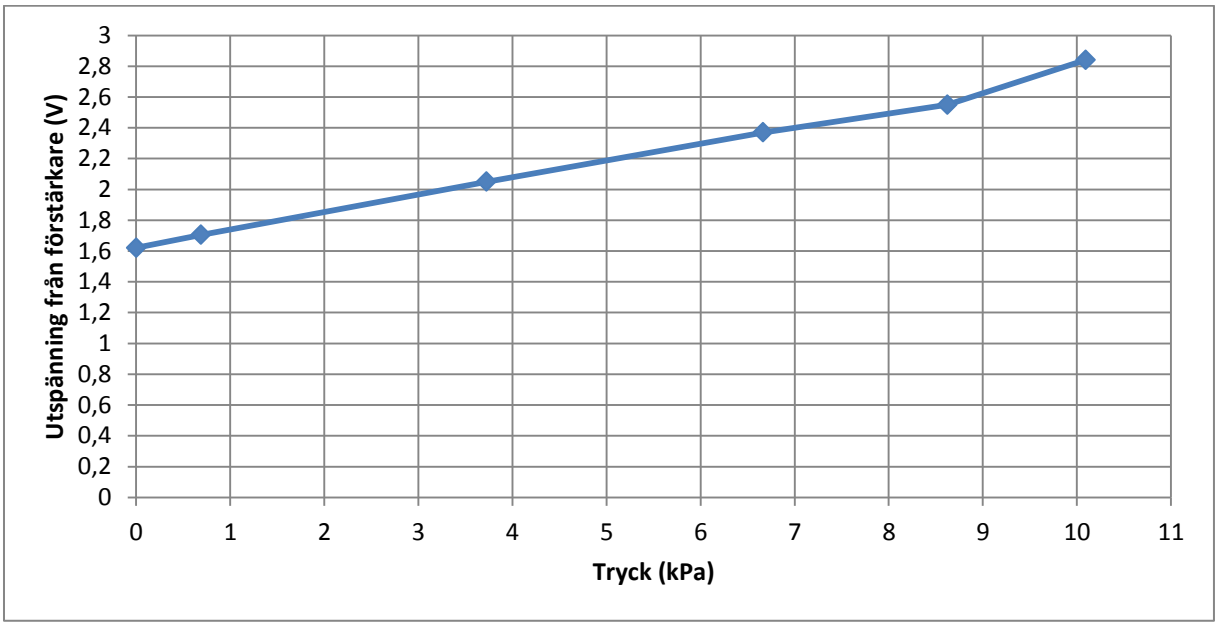
$$antal\ centimeter = (medelvärde - 326) * H \qquad (2)$$

$$där\ H = \frac{maximalt\ tryck\ (hPa)}{maximalt\ värde A/D-omvandlare} = \frac{190}{450} \approx 0{,}42$$

För att verifiera att ekvationen stämmer, utfördes en serie mätningar. En grov slang fylld med vatten användes som mätobjekt. Sedan monterades systemets slang på en tumstock. Därefter fördes tumstocken ner i mätobjektet. Mätningar ner till 105 cm kunde utföras. I de mätningar

som gjordes observerades det att ekvationen inte var korrekt. Konstanten *H* i ekvation (2) fick ändras till 0,44 för att erhålla korrekta mätvärden. Felet i ekvationen är att avrundningar har gjorts.

Nedanstående graf visar sambandet mellan tryck och utspänning från förstärkaren.



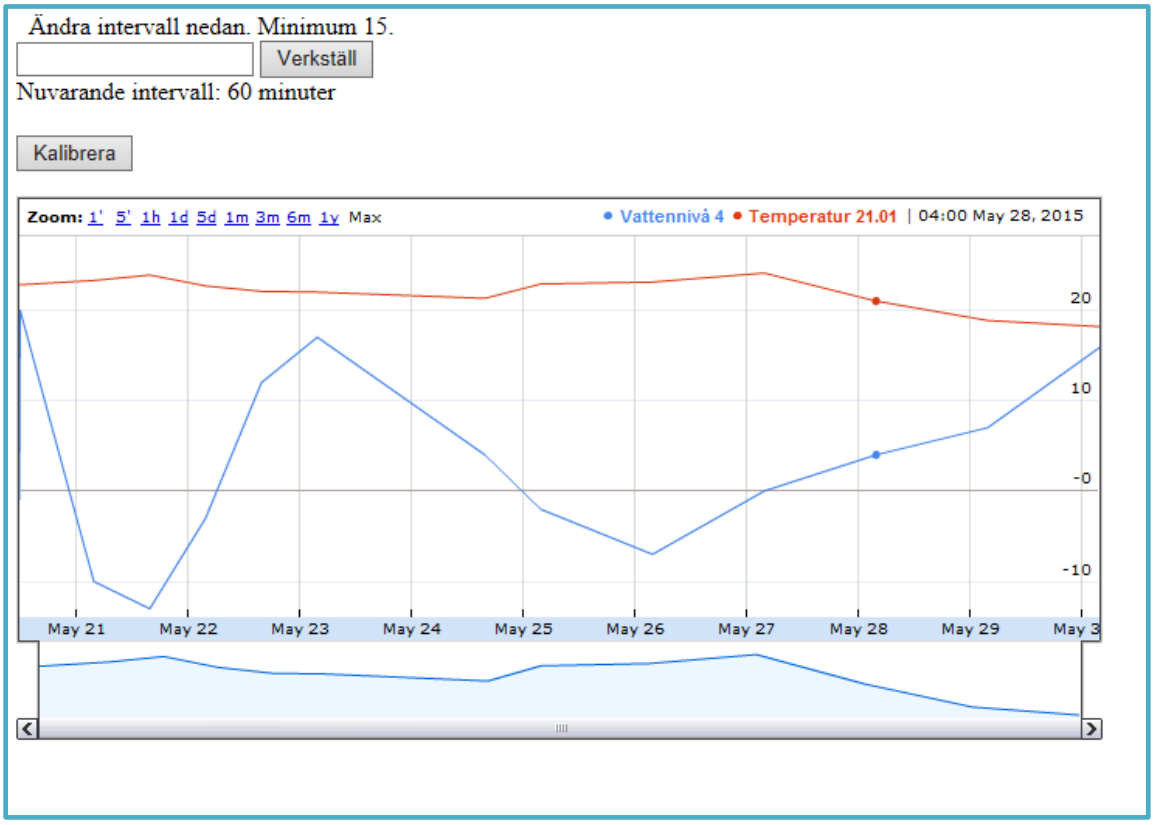
Figur 6.2 Grafen visar sambandet mellan tryck och utspänning från förstärkaren. Referensspänningen på 1,6 V kan observeras i grafen.

Som det har nämnts tidigare, kan komponenter få förändrade egenskaper med tiden. Detta påverkar mätresultatet. För att kompensera det mätfel som uppstår förbereddes en kalibrering av nivåsystemet. En ventil skulle kunna lösa problemet. Ventilens syfte skulle vara att jämna ut trycket i slangen. Därefter tas en mätning för att erhålla en ny referensnivå. Efter en körd kalibrering skickas en bekräftelse till hemsidan. Kalibreringen styrs via hemsidan.

7 HEMSIDA OCH LAGRING AV DATA

En webbserver behövs för att presentera och lagra data på webben. Ett krav på webbservern var att ha tillgång till mySQL- databas. I en mySQL- databas kan data lagras och hämtas. I bilaga 10.6 kan man observera hur en sådan databas ser ut. På webbservern skapades en databas och en hemsida. Strukturen till hemsidan skapades med hjälp av HTML-kod. För interaktiva funktioner på hemsidan och åtkomsten av databasen användes PHP-kod. Textfilen som innehåller mätdata läses av varje kvart av webbservern. Detta åstadkoms med hjälp av ”cron jobs”, en slags funktion som tillåter kontinuerlig exekvering av PHP-kod. Se bilaga 10.3 för uppbyggnaden av filen cron. Databasen uppdateras bara om textfilen innehåller nyare mätdata än vad som redan finns i databasen.

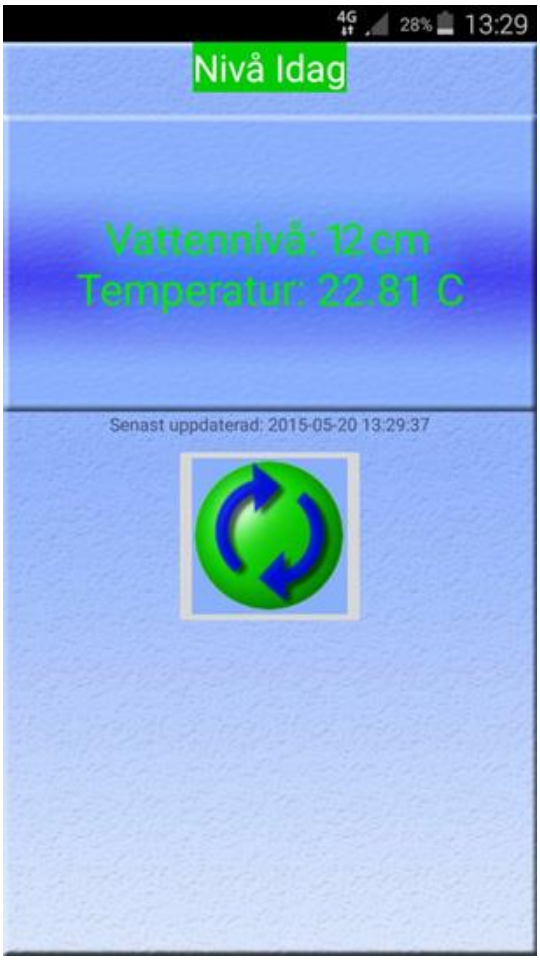
Grafen på hemsidan skapades med hjälp av Google Charts. Google Charts är en gratistjänst som genererar en graf utifrån given data. Databasen paketeras till en matris innehållande datum och tid samt alla mätvärden. Matrisen skickas sedan till Google. En graf returneras, som är fullt modifierbar och kan skräddarsys efter behov. Det är möjligt för användaren att zooma in på specifika delar av grafen. Datormusen används till denna uppgift. Det går även att observera exempelvis det senaste dygnet, senaste månaden eller senaste året. X-axeln redovisar den tidpunkt mätvärdet togs. Noggrannheten är på minutnivå. Man får beakta att webbservern ligger i en annan tidszon. Sålunda får man antingen addera eller subtrahera ett tal för att erhålla den korrekta tiden. Y-axelns enheter är centimeter respektive grader Celsius.



Figur 7.1 Bilden visar hemsidans utseende och grafen som presenterar mätdata. Den röda linjen visar temperaturen i Celsius och den blå linjen vattennivån i centimeter.

På hemsidan kan Arduinos mätintervall styras. Mätintervallet ges i minuter. Det sker en kontroll om intervallet innehåller ogiltiga tecken. Om detta skulle finnas, uppdateras inte intervallet. Då webbservern läser av mätdatafilen var 15:e minut, är minimumintervallet 15 minuter. Det angivna intervallet sparas till en textfil på servern, vilket Arduinon läser av senare. Användaren kan observera det aktuella mätintervallet på hemsidan. En annan funktion på hemsidan är att kunna styra kalibreringen av nivåsystemet. Detta sker via en tryckknapp. Knappen redigerar en textfil, innehållande en boolesk etta. Textfilen är vanligtvis en boolesk nolla för att indikera att kalibrering inte ska köras. Denna textfil läses av Uno vid varje mätning, exempelvis varje kvart. Efter att en kalibrering har körts skickas en bekräftelse tillbaka till hemsidan. När bekräftelsen är mottagen ändras textfilen tillbaka till en boolesk nolla.

En Android applikation skapades, vars huvudsakliga syfte är att läsa av en fil från FTP:n. Denna fil används bara till Android applikationen och filen uppdateras när databasen uppdateras. I denna fil finns de senaste mätvärdena. Applikationen gjordes i mjukvaran Android Studio. Java var det programmeringsspråk koden skrevs i. Innan applikationen överfördes till en mobiltelefon testades det i ett simuleringsprogram. Figur 7.2 visar resultatet av Android applikationen.



Figur 7.2 Bilden visar applikationens utseende.

8 SLUTSATS

8.1 Resultat

Arbetet blev lyckat och uppnår specifikationskraven. Det gick att integrera systemets olika delar till en samverkande enhet. Under slutet av arbetet gjordes en modifiering av spänningsförsörjningen till förstärkaren. I stället för ±9 V användes 12 V. Först togs 12 V ifrån ett spänningsaggregat. Senare skedde matningen direkt ifrån Uno. Detta resulterade i att referensspänningen sjönk en aning, då Arduinon bara kunde leverera 11,5 V. Slutprodukten påverkades dock inte av den lägre spänningen. Pumpen kan styras med ett relä på ett säkert sätt genom två kanaler. Grunden till en kalibreringsfunktion lades till systemet. Kod skrevs såväl för hemsidan som för Arduino Uno för denna funktion. En styrbar ventil lades dock aldrig till. Temperaturgivarens värden verifierades vid sidan om med en portabel termometer. Värdena stämde bra överens.

Mätdata presenteras på hemsidan med den interaktiva grafen, skapad med Google Charts. Då det fanns tid över, kunde en SMS-funktion läggas till systemet. Detta låter användaren skicka ett meddelande till SIM-kortet i GSM-modulen. Ifall förutsättningarna stämmer, skickar modulen mätdata till användarens telefon. Även en Android applikation skapades som visar aktuella värden.

8.2 Diskussion

Vi är nöjda med vad som åstadkommits under detta arbete. Planeringen blev inte riktigt som vi hade tänkt oss. I första veckan fanns inte GSM-modulen tillgänglig. Då kopplades en givare in till Arduinon istället. Överlag så låg vi framför vår planering hela tiden. Detta gjorde att vi kunde göra tillägg till systemet i form av SMS-funktion och Android applikation. Arduino miljön var enkel att arbeta med, då det finns mycket dokumenterat på Internet. Av diverse anledningar har vi växlat mellan Arduino Uno och Yún under arbetets gång. Detta gör att vårt program är kompatibelt med flera olika Arduino enheter.

Tidigare, i kursen LEU240 Mikrodatorsystem, har vi arbetat med en ARM-processor. Erfarenheter från den kursen var till stor användning under arbetet. GSM-modulen var svår att lära sig, då det fanns lite dokumenterat och hade en svårläst manual. Det var första gången vi stötte på AT-kommandon. Dessa kommandon var svåra att förstå, då de resulterade ofta i felmeddelanden från GSM-modulen. Hemsidan kodades med PHP som var nytt för oss. PHP påminner om programmeringsspråket C, vilket underlättade inlärningen eftersom vi kunde detta språk sedan tidigare. Hemsidan tyckte vi blev adekvat, framförallt den interaktiva grafen. Google Charts tycker vi är ett lätthanterligt och praktiskt verktyg. Grafen kan enkelt bytas ut mot andra graftyper.

Utöver bubbelrörsmetoden testades en annan metod för att mäta vattennivå. Denna metod fungerade på samma sätt som tidigare. Det enda som skiljer dem åt är att trycket mäts efter pumpen har stängts av. Denna metod blev ej lyckad för oss eftersom slangsystemet inte var tätt. Detta gjorde att trycket i slangen eventuellt närmade sig atmosfärstrycket. Sålunda påverkades mätresultatet.

8.3 Svar på frågeställningar

Dataöverföring via GPRS kan anses vara relativt säkert, då överföringen sker krypterat.

De två enheterna centimeter och Celsius kan mätas med precision och överföras till hemsidan. Nivågivarens precision blev bättre ju fler mätvärden som togs. Ett medelvärde på dessa värden gav ett stabilt resultat. Den uppmätta felmarginalen på mätvärdet kan vara ±1 cm.

Mätintervallet kan ställas in på hemsidan utan problem. Sålunda kan användaren styra hur lång tid det går mellan varje mätning.

Överföringsprotokollet FTP fungerade bra för små mängder data. Den totala datamängden under ett år kan variera från 1 MB till 25 MB beroende på angivet intervall. Detta innebär att datatrafikkostnaden endast uppgår till cirka 9 kr per år.

För att kunna lägga till en givare måste användaren ha kunskap inom områdena programmering och elektronik. Vissa parametrar i koden behöver ändras för att ackommodera givaren. Även hemsidan behöver modifieras. Slutligen måste användaren koppla in givaren på ett korrekt sätt, där man tar hänsyn till faktorer som A/D-omvandlaren och matningsspänningar. En digital givare är lättare att ansluta. Det är upp till den som ska ansluta ytterligare givare, att bedöma om det beskrivna är enkelt eller inte.

8.4 Rekommendationer till fortsatt arbete

För att uppdatera mätdata mer än en gång i kvarten behöver webbserver hyras från ett webbhotell. Genom att betala för denna tjänst kan ”cron jobs” köras under mindre intervall, till exempel fem minuter. Systemet har en färdig funktion för kalibrering. Sålunda kan man i det fortsatta arbetet lägga till en lämplig styrbar ventil till systemet för att göra funktionen komplett.

Då pumpen kan tränga undan maximalt 1,9 m vattenpelare, kan man överväga att använda en starkare pump för att mäta större vattennivåskillnader. Om pumpen inte byts ut, kan man överväga att övergå till en annan differentialtryckgivare. För en vattenpelare på 1,9 m rekommenderas en tryckgivare på 25 kPa, för bättre precision. Slutligen kan det vara värt att notera att slangen är böjd, vilket gör det svårare att mäta djupare. Sålunda rekommenderas det att slangen sätts fast i en rak anordning. På så sätt förs slangen ner i vattnet utan förhinder.

9 REFERENSER

[1] Arduino Uno, Arduino
http://www.arduino.cc/en/Main/ArduinoBoardUno
(Acc 2015-05-04)

[2] Adafruit FONA overview, Adafruit
https://learn.adafruit.com/adafruit-fona-mini-gsm-gprs-cellular-phone-module/overview
(Acc 2015-05-04)

[3] Standardprislista, Comviq
https://www.comviq.se/files/F130987-113_Standard_Prislista_update27nov_HR.pdf
(Acc 2015-05-15)

[4] Modifierad bild. Ägare: Lady ada. Licens: http://creativecommons.org/licenses/by-sa/3.0/
https://learn.adafruit.com/assets/17736
(Acc 2015-05-11)

[5] Nokia D211 Datasäkerhet. Nokia (sid 7-8)
http://nds1.nokia.com/phones/files/guides/D211_datasecurityguide_sv.pdf
(Acc 2015-05-07)

[6] GPRS/GSM Quadband Module for Arduino Tutorial (SIM900), cooking hacks
http://www.cooking-hacks.com/documentation/tutorials/arduino-gprs-gsm-quadband-sim900
(Acc 2015-05-18)

[7] DS18B20, maxim integrated
http://datasheets.maximintegrated.com/en/ds/DS18B20.pdf
(Acc 2015-05-19)

[8] Thomas Bertil: Modern Reglerteknik, sid. 420 (fjärde upplagan), Liber AB, Stockholm, 2011.

10 BILAGOR

10.1 Arduino Uno-kod

```
#include <SoftwareSerial.h>
#include <OneWire.h>
//#include <Bridge.h> //Används till Yún
#include "Adafruit_FONA.h"
#define FONA_RX 8
#define FONA_TX 9
#define FONA_RST 10
#define niva 0

//Variabler
unsigned long sum;
int cal, sensor_read, raw, celsius;
int kommando, celsius_rest, first, second, kali_var = 332;
byte i, addr_temp[8], data_temp[12], present = 0;
OneWire  ds(11);
char sms_med[10], telefonnummer[15];
float celsius_rest_float, raw_float;

//Initierar den seriella kommunikationen med GSM-modulen.
SoftwareSerial fonaSS = SoftwareSerial(FONA_TX, FONA_RX);
Adafruit_FONA fona = Adafruit_FONA(FONA_RST);

void setup() {
  //Bridge.begin();  //Används till Yún
  Serial.begin(115200);
  pinMode(6, OUTPUT); //Relä-pin
  pinMode(7, OUTPUT); //Relä-pin
  //pinMode(5, OUTPUT); //ventil
  Serial.println(F("Ansluter till GSM-modul"));
  fonaSS.begin(4800);
  //Ansluter til GSM-modul
  if (! fona.begin(fonaSS)) {
    Serial.println(F("GSM FEL!"));
    while (1);
  }
  Serial.println(F("Startar GPRS"));
  //GPRS startup
  fona.setGPRSNetworkSettings(F("internet.tele2.se"));
  if (!fona.enableGPRS(false)) {}
  if (!fona.enableGPRS(true)) {}

  //Ansluter till serverns FTP
  sendATcommand("AT+FTPSERV=nivaidag.host22.com", 2000);
  sendATcommand("AT+FTPPORT=21", 2000);
  sendATcommand("AT+FTPMODE=1", 2000);
  sendATcommand("AT+FTPUN=a2024573", 2000);
  sendATcommand("AT+FTPPW=exjobb15", 2000);
}

//Huvudprogram
void loop() {
  /*
  //Kalibrering
  cal=0;
    sendATcommand("AT+FTPGETNAME=cal.txt", 2000);
  sendATcommand("AT+FTPGETPATH=/public_html/", 2000);
```

```
  sendATcommand("AT+FTPGET=1", 10000);
  cal = sendATcommand("AT+FTPGET=2,2", 10000);
  if (cal) {
    //digitalWrite(5, HIGH); //öppnar ventil
    digitalWrite(6, HIGH);
    digitalWrite(7, HIGH);
    delay(5000);
    for ( second = 0, sum = 0 ; second < 200; second++) {
      sum += analogRead(niva);
    }
    kali_var = sum/second;
    //digitalWrite(5, LOW); //Stänger ventil
    digitalWrite(6, LOW); //stänger av luftpump
    delay(2000);
    digitalWrite(7, LOW);
    delay(2000);
    Serial.println(kali_var);
  }*/

  //Ansluter till servern en gång till. Denna gång för uppladdning av
mätdata
  sendATcommand("AT+FTPSERV=nivaidag.host22.com", 2000);
  sendATcommand("AT+FTPPORT=21", 2000);
  sendATcommand("AT+FTPMODE=1", 2000);
  sendATcommand("AT+FTPUN=a2024573", 2000);
  sendATcommand("AT+FTPPW=exjobb15", 2000);
  sendATcommand("AT+FTPPUTNAME=datalog.txt", 2000);
  sendATcommand("AT+FTPPUTPATH=/public_html/", 2000);

  sensor_read = hamtar_niva(); //hämtar aktuell vattennivå
  //sensor_read = 100 - sensor_read; // hur långt ner röret går ner i
vattnet. Talet 100 visar exempel på 100 cm från vattenytan.
  raw = hamtar_temp(); //hämtar aktuell temperatur
  //omvandlar temperaturen till celsius till två stycken int.
  celsius = raw / 16;
  raw_float = raw;
  celsius_rest_float = (abs(raw_float) / 16);
  celsius_rest = floor(celsius_rest_float);
  celsius_rest = (celsius_rest_float - celsius_rest) * 100;

  //slår ihop strängarna till en sträng
  char result[20];
  sprintf(result, "%d %d.%d %d", sensor_read, celsius, celsius_rest, cal);
  Serial.println(result);
  //resterande tecken i strängen blir mellanslag.
  unsigned int len = strlen(result);
  for (int i = len; i < 20; i++) {
    result[i] = ' ';
  }

  //överföring till servern
  sendATcommand("AT+FTPPUT=1", 10000);
  sendATcommand("AT+FTPPUT=2,20", 10000);  //Mycket känslig, måste överföra
exakt datamängd(20 bytes).
  fonaSS.write(result); //skickar strängen
  sendATcommand("AT+FTPPUT=2,0", 10000);

  //Hämtar kommando.txt, vilket innehåller fördröjningstiden
  sendATcommand("AT+FTPSERV=nivaidag.host22.com", 2000);
  sendATcommand("AT+FTPPORT=21", 2000);
  sendATcommand("AT+FTPMODE=1", 2000);
```

```
  sendATcommand("AT+FTPUN=a2024573", 2000);
  sendATcommand("AT+FTPPW=exjobb15", 2000);
  sendATcommand("AT+FTPGETNAME=kommando.txt", 2000);
  sendATcommand("AT+FTPGETPATH=/public_html/", 2000);
  sendATcommand("AT+FTPGET=1", 10000);
  kommando = sendATcommand("AT+FTPGET=2,5", 10000);
  Serial.print("Fordrojning i ");
  Serial.print(kommando);
  Serial.print(" minuter");
  unsigned int antal_delay = 6 * kommando, antalggr = 0;
  //Fördröjning och kontrollerar om SMS har kommit in
  while (antalggr < antal_delay ) {
    delay(10000);
    if (fona.getNumSMS() > 0) {
      sms_func();
    }
    antalggr++;
  }
}

//Sköter AT-kommandon
unsigned char sendATcommand(char* ATcommand, unsigned int timeout)
{
  unsigned char x = 0,  answer = 0;
  char response[100];
  unsigned long previous;

  memset(response, '\0', 100);    // Initierar strängen
  delay(100);
  while ( fonaSS.available() > 0) fonaSS.read();   // Rensar input
bufferten
  fonaSS.println(ATcommand);    // Skickar AT-kommandot
  x = 0;
  previous = millis();
  // Väntar på response från GSM-modul
  do {
    if (fonaSS.available() != 0) {
      response[x] = fonaSS.read();
      x++;
    }
  }
  // Timeout
  while ((answer == 0) && ((millis() - previous) < timeout));
  Serial.print(response);
  //Retunerar värde
  char *b = response + 16; //Plus 16 för att få korrekt tecken
  int num = atoi(b);
  answer = num;
  return answer;
}

//SMS-funktion
void sms_func(void) {
  uint8_t smsn = 1;
  // Hämtar sändarens telefonnummer
  if (! fona.getSMSSender(smsn, telefonnummer, 250)) {
    Serial.println("Kunde ej fa hamta telefonnummer");
  }
  Serial.println(telefonnummer);

  // Läser in SMS
```

```
  uint16_t smslen;
  if (! fona.readSMS(smsn, sms_med, 250, &smslen)) {
    Serial.println("Kunde ej lasa in SMS meddelande!");
  }
  Serial.println(sms_med);
  //Raderar SMS föt att spara utrymme på SIM-kort
  if (fona.deleteSMS(smsn)) {
    Serial.println(F("SMS borttaget"));
  } else {
    Serial.println(F("Radering av SMS gick inte!"));
  }
  if (sms_med[0] == 'Q') { //kontrollerar första tecknet i sms
    //Hämtar nya mätvärden
    sensor_read = hamtar_niva(); //hämtar aktuell vattennivå
    raw = hamtar_temp(); //hämtar aktuell temperatur
    celsius = raw / 16;
    raw_float = raw;
    celsius_rest_float = (abs(raw_float) / 16);
    celsius_rest = floor(celsius_rest_float);
    celsius_rest = (celsius_rest_float - celsius_rest) * 100;
    char sms_result[50];  //Skapar sträng som ska skickar tillbaka till
användaren
    sprintf(sms_result, "Vattenniva: %d cm\nTemperatur: %d.%d C",
sensor_read, celsius, celsius_rest);
    if (!fona.sendSMS(telefonnummer, sms_result)) {  //Skickar textsträng
till användaren
      Serial.println(F("Att skicka SMS gick snett!"));
    } else {
      Serial.println(F("Skickar meddelande: "));
      Serial.println(sms_result);
    }
  }
}

//Denna funktion hämtar aktuell vattennivå
int hamtar_niva(void) {
  digitalWrite(6, HIGH); //Sätter på luftpump
  digitalWrite(7, HIGH);
  delay(10000);
  //20*200 mätningar görs
  for (first = 0, sensor_read = 0; first < 20; first++) {
    for ( second = 0, sum = 0 ; second < 200; second++) {
      sum += analogRead(niva);
    }
    sensor_read = sensor_read + sum / second;
  }
  Serial.println(sensor_read / first);
  sensor_read = ((sensor_read / first) - kali_var) * 0.44; //omvandlar till
centimeter
  digitalWrite(6, LOW); //stänger av luftpump
  delay(2000);
  digitalWrite(7, LOW);
  delay(2000);
  return sensor_read;
}
//Hämtar aktuell temperatur
int hamtar_temp(void) {
  if ( !ds.search(addr_temp)) {
    Serial.println();
    ds.reset_search();
    delay(250);
```

```
  }
  Serial.println();
  ds.reset();
  ds.select(addr_temp);
  ds.write(0x44);         // starta konvertering
  delay(1000);
  ds.reset();
  ds.select(addr_temp);
  ds.write(0xBE);         // Läser temperatur som har upplösningen 12 bitar
  for (i = 0; i < 2; i++) {
    data_temp[i] = ds.read();
  }
  int16_t raw = (data_temp[1] << 8) | data_temp[0]; //Temperaturen ligger i
två ord som adderas ihop.
  return raw;
}
```

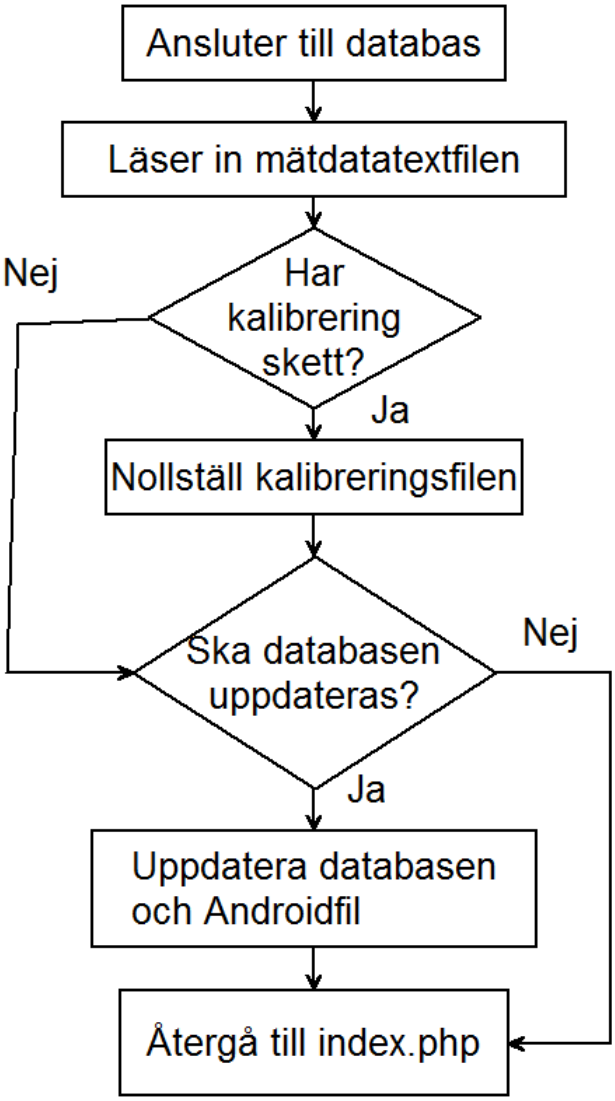
10.2 index.php

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<title>Nivå Idag</title>
<?php include "cron.php";

//Skapar en matris och hämtar data från databas. Denna matris används till
google charts.
$query = "select * from nivadata";
if ($sth = $mysqli->query($query)) {
   $data = array (
   'cols' => array(
   array('id' => 'date', 'label' => 'date', 'type' => 'datetime'),
   array('id' => 'Niv', 'label' => 'Vattennivå', 'type' => 'number'),
   array('id' => 'Temp', 'label' => 'Temperatur', 'type' => 'number')
   ),
   'rows' => array()
);}

//Omvandlar formatet på datum/tid, till javaformat och lägger in data i
matrisen.
while ($res = $sth->fetch_assoc()) {
      preg_match('/(\d{4})-(\d{2})-(\d{2})\s(\d{2}):(\d{2}):(\d{2})/',
$res['reg_date'], $match);
      $year = (int) $match[1];
      $month = (int) $match[2] - 1;
      $day = (int) $match[3];
      $hours = (int) $match[4] + 6; //plus sex eftersom webbservern ligger
i annan tidszon.
      $minutes = (int) $match[5];
      array_push($data['rows'], array('c' => array(
      array('v' => "Date($year, $month, $day, $hours, $minutes)"),
      array('v' => $res['akt_niv']),
      array('v' => $res['akt_tem'])
)));}
$jsonTable1 = json_encode($data);
?>
<!-- Skickar matrisen till google -->
```

```
<script type='text/javascript' src='https://www.google.com/jsapi'></script>
<script type='text/javascript'>
google.load('visualization', '1', {'packages':['annotatedtimeline']});
google.setOnLoadCallback(drawChart);
function drawChart() {
 var data = new google.visualization.DataTable(<?=$jsonTable1?>);
 var chart = new
google.visualization.AnnotatedTimeLine(document.getElementById('chart_div')
);
 chart.draw(data, {displayAnnotations: true});
}
</script>
</head>
<body>

<!-- Skapar textbox och knapp -->
Ändra intervall nedan. Minimum 15.
<FORM NAME ="form1" METHOD ="GET" ACTION = "index.php">
<input type="text" value="" name="name" id="name" >
<input type="submit" name="submit" value="Verkställ"><br/>

<?php
//Vad som händer när man trycker på knappen.
if(isset($_GET['submit'])){
 $name = $_GET['name'];
  if (preg_match('/[0-9]/', $name)){  //Kontrollerar ogiltliga tecken
   if ($name>14){                    //minimum är 15 minuter
    $myfile = fopen("kommando.txt", "w") or die("Unable to open file!");
//skriver till textfil
    fwrite($myfile, $name);
    fclose($myfile);   }
   else { echo ("Ogiltligt! ");}
  }
 else { echo ("Ogiltligt! ");}
}
//Läser av nuvarande intervall
$myfile1 = fopen("kommando.txt", "r") or die("Unable to open file!");
$nu_var = fread($myfile1,filesize("kommando.txt"));
fclose($myfile1);
echo ("Nuvarande intervall: ");
echo ($nu_var);
echo (" minuter");
?>

<!-- kalibreringsknapp -->
</br></br>
<form action="index.php" method="get">
<input type="submit" name="submit1" value="Kalibrera">
</form>

<?php
// vad händer när man trycker på kalibreringsknappen. Skriver till cal.txt.
if(isset($_GET['submit1'])){
 $myfile = fopen("cal.txt", "w") or die("Unable to open file!");
 fwrite($myfile, '1');
 fclose($myfile);
 echo ("Kalibrering har skett!");
}
?>

</FORM>
```

```
<!-- Prensentation av grafen-->
<div id='chart_div' style='width: 700px; height: 350px;'></div>
</body>
</html>
```

10.3 Flödesschema för cron.php

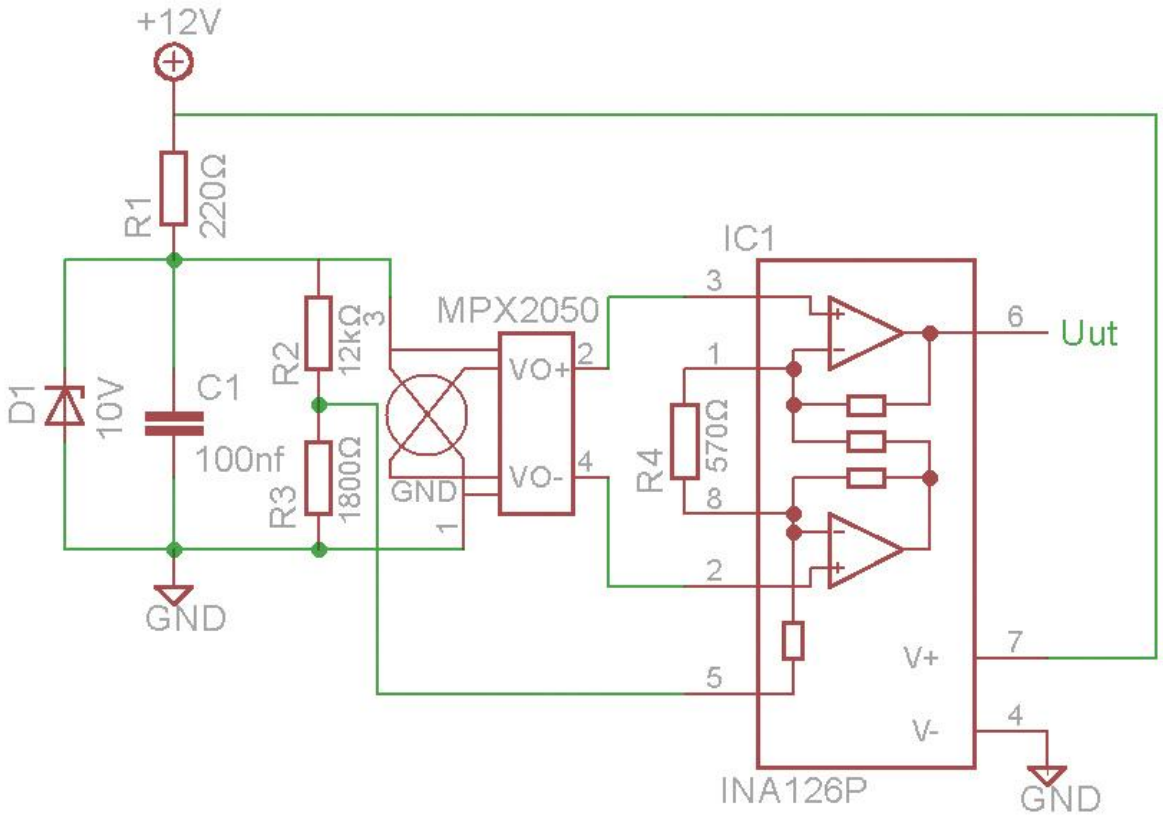


10.4 cron.php

```
<?php
  //ansluter till databasen
  $mysqli = mysqli_connect("Servername or IP Address", "user", "pasword","user");

  if (!$mysqli) { die("Connection failed: " . mysqli_connect_error());}

  //Hämtar tiden då senaste raden lades till i databasen och tiden då
textfilen skapades.
  $qstring = ("select * from nivadata order by reg_date desc limit 1");
  $last_timestamp = $mysqli->query($qstring)->fetch_object()->reg_date;
  $filename = 'datalog.txt';
```

```
  if (file_exists($filename)) {
    $file_time = date ("Y-m-d H:i:s", filemtime($filename));}

    //Läser in mätdata
  $myfile = fopen("datalog.txt", "r") or die("Unable to open file!");
  $line = fread($myfile,filesize("datalog.txt"));
  $txtstring = explode(" ", $line); //delar upp textsträngen till flera.
Det ligger ett mellanslag mellan varje mätvärde.
  fclose($myfile);

   //nollställer kalibrering om arduino säger att den har skett.
  if ($txtstring[2]){
    $myfile = fopen("cal.txt", "w") or die("Unable to open file!");
    fwrite($myfile, "0");
    fclose($myfile);
    }

  //Ska databasen uppdateras
  if( strtotime($file_time) > strtotime($last_timestamp)){

    //lägger in mätvärderna i en ny rad i databasen
    $insert_row = $mysqli->query("INSERT INTO nivadata(akt_niv, akt_tem)
VALUES($txtstring[0],$txtstring[1])");
    if(!$insert_row){die('Error : ('. $mysqli->errno .') '. $mysqli-
>error);}

    // skapar en php fil som android Appen kommer att läsa av.
    $android_file = fopen("android_read.php", "w") or die("Unable to open
file!");
    $android_string = "Vattennivå: " . $txtstring[0]. " cm " ."\r\n" .
"Temperatur: " . $txtstring[1] . " C";
    fwrite($android_file, $android_string);
    fclose($android_file);
    }
?>
```

10.5 Kretsschema över förstärkarkopplingen



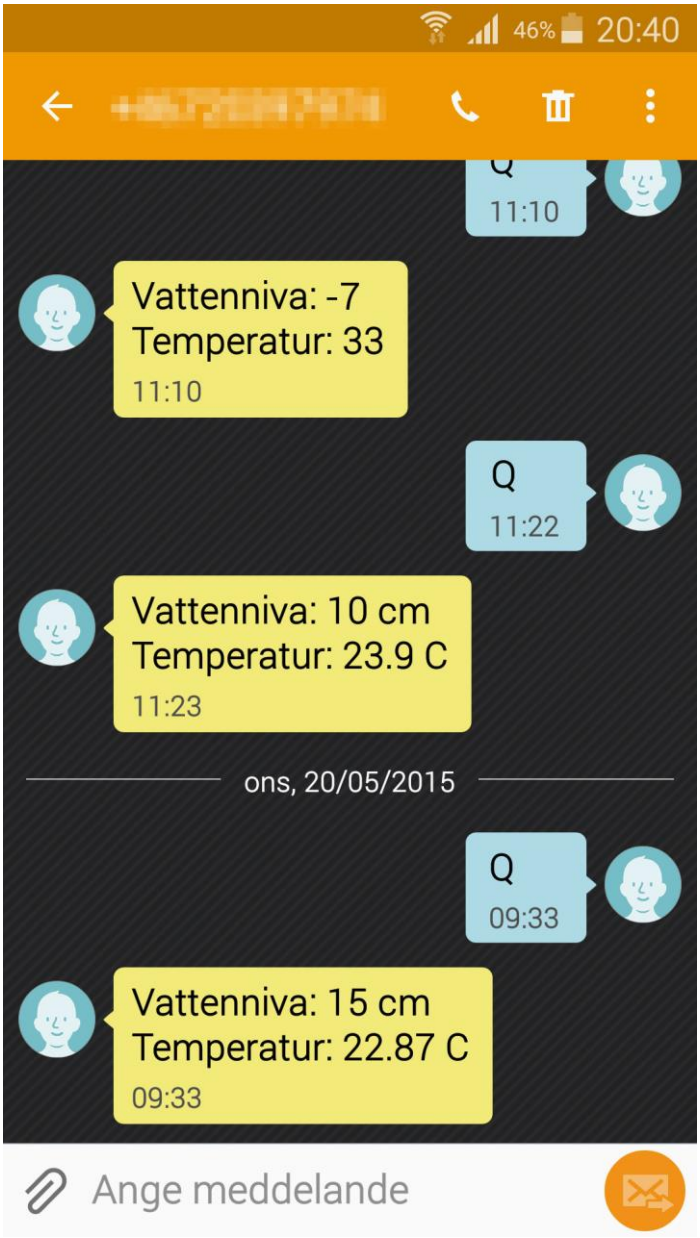
Kretsschema över förstärkarkopplingen. MPX2050 är differentialtryckgivaren. INA126P är instrumentförstärkaren som användes under arbetet. U_{ut} är den spänning som går till Arduino Uno.

10.6 MySQL-databasens utseende

	id	akt_niv	akt_tem	reg_date
	220	16	18.17	2015-05-29 22:00:00
	219	7	18.88	2015-05-28 22:00:00
	218	4	21.01	2015-05-27 22:00:00
	217	0	24.11	2015-05-26 22:00:00
	216	-7	23.12	2015-05-25 22:00:00
	215	-2	22.89	2015-05-24 22:00:00
	214	4	21.30	2015-05-24 10:00:00
	213	17	22.00	2015-05-22 22:00:00
	212	12	22.10	2015-05-22 10:00:00
	211	-3	22.70	2015-05-21 22:00:00
	210	-13	23.90	2015-05-21 10:00:00
	209	-10	23.30	2015-05-20 22:00:00
	208	-1	22.81	2015-05-20 06:00:04
	207	20	22.81	2015-05-20 06:12:11

Så här ser databasen ut på webbservern. Den innehåller fyra kolumner med information. Kolumnen ”id” är en numrering. akt_niv innehåller vattennivådata. akt_tem innehåller temperaturdata. reg_date visar när mätvärdena registrerades.

10.7 SMS-funktion



När användaren skickar ett Q till SIM-kortet får hen tillbaka ett meddelande som innehåller färska mätvärden. Om meddelandet inte börjar med bokstaven Q utförs ingen mätning.