CHALMERS

Universell IoT-enhet

För dataregistrering och processtyrning

Examensarbete inom Högskoleingenjörsprogrammet i Datateknik och Mekatronik

RICKARD EDFAST
OSKAR LINDSTRÖM

Institutionen för data- och informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2014

Universell IoT-enhet
För dataregistrering och processtyrning

Rickard Edfast, Oskar Lindström

Institutionen för data- och informationsteknik
Chalmers tekniska högskola
412 96 Göteborg
Tel: 031-772 1000
Fax: 031-772 3663

Institutionen för data- och informationsteknik
Göteborg, 2014

Förord

Denna rapport är resultatet av ett examensarbete som utfördes under våren 2014 vid Institutionen för Data- och Informationsteknik, Chalmers Tekniska Högskola. Arbetsperioden var tio veckor och motsvarade 15 högskolepoäng. Vi som utfört projektet studerar till högskoleingenjörer inom Datateknik respektive Mekatronik. Handledare från Chalmers har varit Sven Knutsson och examinator Lars Svensson. Vi vill tacka vår Sven Knutsson för all vägledning och även rikta ett speciellt tack till Sakib Sistek som hjälpte till med utformningen av idéen som ledde till projektet.

Sammanfattning

Marknaden för IoT-enheter växer och för att utveckla en enhet krävs kunskap för att kunna programmera mikrokontrollern. Syftet med detta projekt var att utveckla en universell IoT-enhet, som gick att konfigurera grafiskt, för registrering av data samt styrning av processer. Relevant efterforskning gjordes för att specificera kraven för enheten. Kraven nåddes genom en agil arbetsmetod och projektet kan i stort delas in i tre huvudcykler: Utveckling av databasen, programmeringen av mikrokontrollern och utvecklingen av ett webbgränssnitt. Som platform har en Arduino Yún använts och utvecklats för att spara data från sensorer och styra processer i realtid. Ett webbgränssnitt har utvecklats för att kunna presentera kontinuerligt insamlad data. Fokus har legat på att ta fram en produkt som ställer låga krav på slutanvändarens förkunskaper och som inte kräver någon administration efter att den konfigurerats. Slutresultatet mynnade ut i en fungerande prototyp som uppfyller de ställda kraven kring funktionalitet och som har potential att kunna säljas på slutmarknaden efter fortsatt utveckling. Dels behövs vidareutveckling av det grafiska gränssnittet samt en förbättring av databashanteringen.

Abstract

Internet of Things is growing and deploying an IoT unit today requires knowledge in programming of the microcontroller. The purpose of this project was to develop a universal IoT unit with focus on the end user interaction. An IoT unit requires communication between the user interface and the hardware (a microcontroller and a system on a chip) while the microcontroller operates autonomously. Relevant research was used to specify the requirements for the unit. The requirements were met through an agile development method broken down into three main cycles: development of the database, graphical user interface and development of the microcontroller code. The end result was a working prototype based on Arduino Yún with a grapical user interface to detach the user from the microcontroller. There are still areas that need further research and testing, where the main areas would be to test a headless database process and further development of the graphical user interface.

Innehåll

Förkortningar

ACID	Atomicity, consistency, isolation, durability
AD	Analog till digital
AJAX	Asynchronous javascript and XML
API	Application programming interface
BSD	Berkeley software distribution
CGI	Common gateway interface
CLI	Command line interface
DDR2	Double data rate synchronous dynamic random-access memory
EEPROM	Electrically erasable programmable read-only memory
ER	Entity-relationship
HTML	Hypertext markup language
HTTP	Hypertext transfer protocol
I2C	Inter-integrated circuit
IO	Input/output
IoT	Internet of things
ICSP	In circuit serial programming
IP	Internet protocol
JSON	JavaScript object notation
LAN	Local area network
OSI	Open systems interconnection
PHP	PHP: hypertext processor
PID	Proportionell, integrerande och deriverande
PoE	Power over ethernet

POSIX	Portable operating system interface
PWM	Pulsbreddsmodulering
RAM	Random-access memory
RC	Lågpassfilter
RX	Recieve
SD	Secure digital
SPI	Serial peripheral interface
SSH	Secure shell
SQL	Structured query language
TCP	Transmission control protocol
TWI	Two wire interface
TX	Transmit
UART	Universal asynchronous receiver/transmitter
USB	Universal serial bus
VPN	Virtual private network
WAN	Wide area network
WLAN	Wireless local area network

Kapitel 1

Inledning

1.1 Bakgrund

Att lansera en IoT-enhet idag kräver kunskap om mikrokontrollers samt elektronik. Det är en marknad som expanderar idag och fler vardagliga saker kopplas upp till internet av företag och entusiastiska hemanvändare med programmeringskunskap.

1.2 Syfte

Målet med projektet är att skapa en universell IoT-enhet för registrering av data och styrning av olika processer. Enheten ska kunna gå att ansluta till elnätet och till befintligt lokalt nätverk.

Det som skiljer vår enhet från de som finns på marknaden idag är enkelheten för användaren. Denne kommer inte behöva någon kunskap varken om programmering eller om mikrokontrollers. Allt användaren kommer i kontakt med mjukvarumässigt är ett webbgränssnitt där denne kan välja att konfigurera eller övervaka sina processer genom ett grafiskt gränssnitt. Hårdvarumässigt kommer användaren behöva koppla in givare och ställdon vilket dock kräver en viss kunskap. Detta är svårt att komma ifrån om en universell och komplett lösning ska kunna erbjudas. För att underlätta för användaren, skulle exempel på hur de vanligaste givarna och ställdonen kopplas in samt hur de konfigureras, kunna ingå i helhetslösningen. Detta skulle troligtvis göra den attraktiv för privata användare.

Projektet ska resultera i en prototyp som efter vidare utveckling ska vara tillräckligt bra för att kunna säljas på slutmarknaden.

Syftet är att skapa en IoT-enhet med tillhörande system där enkelhet för slutanvändaren ska prioriteras, där endast en grundläggande kunskap om elektronik kommer att behövas. Fokus kommer att ligga på att samarbetet mellan mikrokontrollern och databasen blir

helt självgående och att användaren endast ska komma i kontakt ett webbgränssnitt för konfiguration och övervakning.

1.3 Mål

Systemet baseras på de tre huvudpunkterna databas, mikrokontroller och webbgränssnitt. För var och en av dessa punkter finns en målbeskrivning för önskad funktion.

1.3.1 Databas

Databasen kommer att vara navet i hela systemet, den ska konstrueras och testas för olika scenarion. Målet är att göra databasen så resurssnål som möjligt med tanke på enhetens begränsade beräkningskapacitet samt begränsade mängd minne. Databasen ska spara all information angående styrning och registrering som specificeras av användaren i webbgränssnittet angående styrning och registrering. Den ska även spara inmatad information från datablad tillhörande de använda givarna och kontinuerligt spara data som registreras från givare.

1.3.2 Mikrokontroller

Mikrokontrollern ska initieras automatiskt vid ändrad konfiguration utifrån information som finns i databasen. Mikrokontrollern ska kunna styra processer i realtid samtidigt som den registrerar data från givare och sparar data i databasen.

1.3.3 Webbgränssnitt

Webbgrässnittets ska tillhandahålla konfigurationsmöjligheter för användaren och presentera registrerad data i databasen.

1.3.3.1 Presentation

Målet med presentationssidan är att ge användaren en överskådlig bild av pågående processer. För analoga in- och utgångar ska presentationen ske i en graf med rätt måttenhet, digitala in- och utgångar ska endast presenteras med deras status.

1.3.3.2 Konfiguration

Målet med konfigurationssidan är att användaren ska kunna mata in värden för styrningen och ange vilka ingångar som används vid återkoppling. Datablad för givare ska gå att lägga till, detta för att kunna ange ett värde i rätt måttenhet istället för ett AD-omvandlat värde vid presentationen. Det ska även vara möjligt att konfigurera enheten att skicka ett mejl till användaren då specificerade maximala eller minimala nivåer över- eller underskrids.

1.3.4 Funktioner

Dygnsrapporter från det gångna dygnet med insamlad data ska kunna skickas via mejl.

1.4 Avgränsingar

Gränssnittet för styrning kommer inte att ges någon tyngd för att se grafiskt tilltalande ut, exempelvis likt Simulink i Matlab, då det skulle vara alltför tidskrävande. Detsamma gäller presentationen av data i graf form, där kommer vi att använda oss av ett färdigutvecklat bibliotek då ett sådant hade varit alltför resursödande att utveckla. Projektet kommer inte omfatta hårdvaruutveckling utan istället har en färdig plattform med specifik hårdvara som passar denna lösning valts. Ingen hänsyn kommer att tas till säkerhet, där ansvaret för säkerheten på det lokala nätverket antas ligga på nätverksadministratören och Linux anses tillräckligt säkert.

Kapitel 2

Metod

Projekt ska genomföras med en agil utvecklingsmetod vilket innebär att projektet sker i cykler. Varje cykel baseras på att utveckling sker, därefter testas och implementeras en fungerande del av mjukvaran. Projektets olika delar är i utvecklingsstadiet relativt oberoende av varandra och kan därför utvecklas parallellt för att sedan sammanflätas till en komplett produkt.

Projeket kommer inledas med en förberedelsefas som kommer innebära inhämtning av information kring de olika språkens API.

2.1 Kommunikation

Kommunikation med enheten kommer ske trådlöst eller trådbundet via lokalt nätverk. Detta gör att användare som är uppkopplade på samma nätverk kan komma åt webbservern som är installerad på enheten för att dels konfigurera den och övervaka processerna.

2.2 Datainsamling

Datainsamling kommer att ske via givare kopplade till mikrokontrollern som sedan kontinuerligt ska spara ärvärden i databasen där de konverteras till läsbara värden med rätt måttenhet enligt tillhörande datablad.

2.3 Presentation av data

Javascriptbiblioteket Flot kommer att användas för att kontinuerligt presentera insamlad data i en graf som det går att interagera med.

2.4 Utveckling

Utvecklingen av mjukvaran kommer att ske på datorer lokalt. Utveckling av C för mikrokontrollern kommer ske i Arduinos egen integrerade utvecklingsmiljö. Webbutveckling av PHP, HTML, Javascript sker i textediteringsprogrammet Sublime Text 2 och exekvering kommer att ske via en lokal webbserver.

2.5 Testning

Testning kommer initialt att ske lokalt i varje arbetscykel. När en cykel är färdig implementeras den och testas på enheten tillsammans med ett kopplingsdäck med diverse givare och ställdon. Denna testning sker genom det lokala nätverket.

2.6 Material

Material som kommer användas i detta projekt är följande:

- Plattform med systemkrets och mikrokontroller
- Switch TP-Link TL-SF1005D
- 3 st nätverkskabel RJ-45
- Kopplingsdäck
- Komponenter för testning

Kapitel 3

Teknisk bakgrund

Utvecklingen av IoT har pågått länge, där vardagliga saker som bilar, hushållsmaskiner och verktyg kommer troligtvis vara uppkopplade. En tvättmaskin som behöver service ska till exempel kunna skicka information till reparatören om vilken specifik komponent som havererat vilket sparar tid och minskar kostnaderna för reparationen. Det som gör att utvecklingen snabbt går framåt nu är IP version 6, högre bandbredd och billigare lagring av data. IPv6 ger möjligheten för 2^{128} enheter, vilket är ett enormt stort tal, att vara uppkopplade mot internet. IoT-enheter kan använda sig av 4G-nätet för ökad bandbredd och lagringen av data är dessutom billig. [1]

Xively är den första plattformen som utvecklades som en kommersiell tjänst för IoT, det är menat som en tjänst för företag som vill skapa enheter anslutna till molnet av IoT. Xively vill uppnå detta genom att förenkla och accelerera utvecklingen för företagen som önskar använda IoT genom att erbjuda färdiga API [2]. De har även börjat erbjuda konsultation i vilka det ingår tjänster för att ta fram en färdig lösning för kunden utifrån dennes krav [3]. Vad som är gemensamt för Xively och dess konkurrenter som ThingWorx och Carriots är att de inte erbjuder en färdig lösning för mikrokontrollern utan denna måste programmeras av användaren. Det som erbjuds är exempel och referenser till bland annat Arduinos egen hemsida. [4, 5]

3.1 Hårdvara

För hårdvaran skulle en helt egen lösning med mikrokontroller och systemkrets kunna tas fram men detta hade varit tidskrävande. En annan lösning är att använda en mikrokontroller och kombinera den med en enkortsdator som till exempel Raspberry Pi, vilket Microchip har skapat en lösning för [6].

Arduino Yún är en färdig plattform som har den sökta funktionaliteten och som dessutom är strömsnål då dess maximala effektförbrukning endast är 0,15W [7]. Arduino Yún baseras på mikrokontrollern Atmel ATmega32U4 och systemkretsen Atheros AR933. Enligt Arduino är en Arduinoplattform ett verktyg för att göra datorer mer medvetna om den fysiska omvärlden och kunna kontrollerna den. En Arduinoplattform har i grunden en

mikrokontroller med ett gränssnitt för att kunna kommunicera med en dator. Arduino bygger på öppen källkod, både för mjukvaran och för hårdvaran. [8]

3.1.1 Mikrokontroller

En mikrokontroller består av en processor och kringutrustning i form av till exempel programminne, in- och utgångar, AD-omvandlare och gränssnitt för seriell kommunikation. De kan jämföras med datorer i ett litet format och eftersom de tillverkas i stora serier är de billiga. En mikrokontroller lämpar sig för system som måste köras i realtid med en kristall eller oscillator, denna kan vara antingen extern eller intern och kan ge en klocksignal som används för realtidsberäkningar. Benen kan konfigureras som in- eller utgångar för kommunikation med andra enheter och utrustning som givare och ställdon [9]. Atmel ATmega32U4 är mikrokontrollern som används av Arduino Yún. Den består av en 8-bitars processor som har klockfrekvensen 16 MHz, 32 kB programminne, 2,5 kB minne, 1 kB EEPROM. Det finns möjlighet för 26 IO-ben, varav åtta har PWM och tretton ben kan användas för AD-omvandling. AD-omvandlingen sker med 10-bitars upplösning och PWM-signalen har 8-bitars upplösning. ATmega32U4 har en integrerad 32 kHz RC oscillator och kommunikation kan ske via SPI, I2C, UART och USB. [10]

3.1.2 Systemkrets

En systemkrets är designad kring en mikroprocessor och kringutrustning som kan vara i form av till exempel minne, nätverkskontroller, grafikkontroller och ljudkontroller, där mikroprocessorn och all kringutrustning är integrerade i ett chip [11]. Atheros AR9331 är systemkretsen som används av Arduino Yún, den är främst tänkt för användning som accesspunkt och som router. Den tillverkas med en 32 bitars processor med klockfrekvens 400 MHz, den har gränssnitt för externt minne, gränssnitt för kommunikation via LAN/WAN, UART, USB samt IO-ben och radio för WLAN [12].

3.1.3 Arduino Yún

Mikrokontrollern ATmega32U4 och systemkretsen Atheros 9331 integreras av Arduino till plattformen Yún på ett kretskort. Systemkretsen är ansluten till 64 MB DDR2 ram, 16 MB flashminne, en minneskortsläsare av typen micro-SD, en nätverksport och en USB-port [7]. Den innehåller Linuxdistributionen Linino som är baserad på OpenWrt och den kan kommunicera med mikrokontrollern via UART, där sändningar från mikrokontrollern till systemkretsen tolkas av biblioteket bridge [13]. Det går inte att komma åt IO-benen på systemkretsen utan de som finns på plattformen är knutna till mikrokontrollern. På plattformen finns 20 IO-ben, varav sju är möjliga att definiera som PWM och tolv kan definieras som analoga ingångar. Den kan programmeras via USB som också står för strömmatningen. Strömmatning kan alternativt ske med hjälp av PoE via en extra modul. Mikrokontrollern har en bootloader vilket innebär att programmering av den kan ske utan extern hårdvara. Programmeringen av mikrokontrollern kan göras i C eller C++ [7].

3.1.4 Nätverk

Det är få verksamheter som inte har någon form av nätverk. Det finns flera olika typer av nätverk, till exempel intranät vilket är ett kontrollerat nätverk som begränsas till användare med rätt behörigheter, VPN och internet vilket alla har tillgång till.

Kommunikationen på LAN och WAN sker vanligen över TCP/IP-protokollet, vilket togs fram för att standardisera kommunikationsprotokollen. Det består egentligen av två separata protokoll; TCP-protokollet som bryter ner data till småpaket och sätter ihop data när alla paket kommit fram medan IP-protokollet hanterar hur data tar sig fram. TCP/IP referensmodellen består av fyra lager applikationslagret, transportlagret, internetlagret och nätverkslagret.

Nätverkslagret består av drivrutiner och hårdvara för att fysiskt kunna anslutas till nätverket. Internetlagret levererar paketen över nätverket med hjälp av IP-protokollet, detta genom att bestämma vart det ska skickas och hur det ska skickas. Transportlagret är det som bestämmer hur dataflödet ska ske mellan noderna i nätverket. TCP-protokollet är det som används för dataflödet, det innehåller även information om hur kontroll av paketen ska ske. Applikationslagret får data av de tidigare lagren, det kan vara till exempel HTTP-protokollet som innehåller detaljer om hur applikationen fungerar och dess processer men knappt någon information om hur data skickas. Transporten av data kan även ske krypterat.

Vid varje sändning av data säkerställs det att data inte blivit korrupt då det i så fall skickas på nytt och skulle data saknas skickas det igen. Idag är IP version 4 det mest använda IP-protokollet men IP version 6 är menat att bli ersättaren då det kan erbjuda fler IP-adresser (vilket kommer med fler anslutna enheter, till exempel genom IoT). [14]

3.2 Mjukvara

Mjukvara väljs utifrån att vara funktionell på ett integrerat system vilket medför krav på att mjukvaran ska vara resurssnål, det vill säga liten, snabb och pålitlig. Detta på grund av begränsningar i processorhastighet och lagringskapacitet.

3.2.1 Linino

Linino har utvecklats av Arduino för att skapa en produkt som ska underlätta hemautomation via WLAN och göra det lättförståeligt för användare med begränsad teknisk kunskap. Detta sker genom att kombinera egenskaperna hos OpenWrt och AllJoyn [15]. AllJoyn är ett ramverk och består av olika systemtjänster skrivna i öppen källkod för Linuxbaserade system som tillhandahåller funktioner för olika IoT-enheter [16]. OpenWrt är en Linuxdistribution som från början togs fram för att ha funktionaliteten hos en router. Den öppna källkoden och möjligheten för konfiguration har lett till att det idag används

av integrerade system. OpenWrt har stöd för många olika arkitekturer och grundinstallationen är minimal men erbjuder möjligheten för installation av program med hjälp av en pakethanterare som innehåller över 2000 paket [17]. En alternativ lösning istället för att använda operativsystemet Linino hade varit att själv kompilera OpenWrt med endast de moduler som krävs för ett ännu resurssnålare system. Om det skulle tillkomma ett behov av fler funktioner så har dock Linino bättre stöd för detta än ett avskalat OpenWrt-system eftersom Linino är anpassat för Arduino.

3.2.2 Bridgebiblioteket

För att Linuxmiljön ska kunna kommunicera med mikrokontrollern finns biblioteket bridge. Det är skrivet i Python och gör att mikrokontrollern kan exekvera processer med till exempel kommandon i Linuxmiljö som sedan kan skickas till mikrokontrollern. Samtidigt kan Linuxmiljön kommunicera med mikrokontrollern genom att skicka kommandon i form av att sätta på eller stänga av signaler på portar [13].

3.2.3 SQLite

Ett integrerat system lämnar inte några möjligheter för administration av databasen. OpenWrt har stöd för flera databassystem där de flesta tillämpningarna använder sig av MySQL, MariaDB, PostgreSQL eller SQLite [18]. SQLite3 är en resurssnål databas, som utför transaktioner effektivt och är skrivet i öppen källkod för integrerade system [19]. SQLite utför sina databastransaktioner enligt ACID vilket garanterar pålitlighet. SQLite behöver ingen konfiguration vid installationen eller senare administration och stödjer större delen av SQL92-språket. Hela databasen sparas i en fil som enkelt kan flyttas mellan enheter och det tar upp lite ram minne. Det körs utan en serverprocess, tvärtemot vad de flesta SQL-databaser gör. SQLite behöver endast ha läs- och skrivrättigheter till filen för att fungera. Dock medför detta att man inte kan kontrollera databasen lika exakt som med en serverprocess eftersom administration endast kan göras lokalt på enheten. Alla dessa faktorer resulterar i en liten, snabb och pålitlig databas som inte behöver någon administration [20].

3.2.4 sSMTP

sSMTP är ett enkelt och resurssnålt program för att skicka mejl. Det har inga andra funktioner än att skicka mejl och det körs dessutom inte som en bakgrundsprocess utan exekveras endast vid behov för att spara processorkraft. [21]

3.2.5 Cron

Cron är den standardiserade schemaläggaren i Linux och måste aktiveras i Linino. Cron körs som en bakgrundsprocess och konfigureringen av schemalagda aktiviteter görs enligt:

```
minut(0-59), timme(0-23), dag i månaden (1-31),
månad i året (1-12), veckodag (0-6)
```

Därefter exekveras ett specificerat kommando eller program. [22]

3.2.6 uHTTPd

OpenWrt lämnar möjligheter att använda många olika webbservrar som bland annat Apache, Lighthttpd, uHTTPd, axhttpd och mini-httpd [23]. uHTTPd är en webbserver skriven i öppen källkod av OpenWrt/LuCI utvecklarna. Den har all funktionalitet som en vanlig webbserver och den är effektiv och stabil samtidigt som den är utvecklad för integrerade system och OpenWrt specifikt. Eftersom uHTTPd har samma funktionalitet som en vanlig webbserver fungerar till exempel kryptering, CGI och möjligheten till flera webbservrar på olika portar [24], vilket gör att konfigurationsgränssnittet kan bibehållas samtidigt som gränssnittet för systemet körs. axhttpd är den enda webbservern som är resurssnålare än uHTTPd men axhttpd är utvecklad för BSD [23], dock väljs främst uHTTPd för sitt stöd av OpenWrt-utvecklarna då det är skrivet för OpenWrt specifikt.

3.2.7 Flot

I ett Linuxsystem som Linino finns det många möjligheter att presentera data grafiskt. De vanligaste programmen som används för detta är rrdTool som kan skapa bilder som kan presenteras på webbservern och charts.mrtg som är skrivet för att presentera routertrafik med hjälp av rrdTool. Det finns även möjlighet att plotta från en databas direkt på en webbsida med hjälp av ett skriptspråk för att skapa dynamiska webbsidor [25]. Flot är ett bibliotek utvecklat i jQuery för att grafiskt presentera data. Flot ska vara enkelt att använda, se grafiskt bra ut och det ska gå att interagera med den presenterade datan [26]. Det är ett externt bibliotek som lämpar sig mycket väl för mobila enheter och det enda kravet är att data som presenteras är i form av JSON [27] vilket innebär att det kommer fungera väl på integrerade system. rrdTool hade kunnat användas för att skapa grafer men det lämnar ingen möjlighet för ett interaktivt gränssnitt utan det visas endast som en bild.

3.3 Utvecklingsspråk

C används för att programmera mikrokontrollern och SQL används vid databastransaktioner.

3.3.1 PHP

PHP är ett skriptspråk som är skrivet i öppen källkod. PHP används för att exekvera skript på webbservrar och kan integreras med HTML för att skapa dynamiska webbsidor. PHP har stöd för de flesta stora operativsystem, i Linux fungerar PHP som en modul eller genom CGI. CGI är ett protokoll som används som gränssnitt mellan en webbserver och ett program som genererar dynamiskt innehåll. Det betyder att alla webbservrar med stöd för CGI kan använda PHP, vilket uHTTPd har. PHP har stöd för datatypen JSON [28] och flertalet databassystem, däribland SQLite [28].

3.3.2 HTML

HTML är ett märkspråk som används för att ge kontext och struktur till innehåll genom att använda olika så kallade taggar. Det skiljer sig från ett programmeringsspråk som beskriver en process av att göra något genom metoder och satser. [29]

3.3.3 Javascript

Javascript är ett skriptspråk som främst används på klientsidan för webbtillämpningar, vilket betyder att användarens webbläsare exekverar skripten. Javascript kan användas för att ge användaren möjlighet att interagera med webbsidan på olika sätt utan skicka en ny förfrågan för sidhämtning. Exempel på användningsområden är att direkt validera information som skrivits in i en textruta. [30]

3.3.4 jQuery

jQuery är ett lättviktigt Javascriptbibliotek skrivet i öppen källkod som ska förenkla utvecklandet av webbapplikationer i Javascript [31]. Grundutförandet av jQuery har färdiga metoder för vanliga uppgifter som annars kräver många rader av kod i Javascript. Flot är baserat på jQuery och använder bland annat AJAX-anrop från jQuerybiblioteket. [32]

3.3.5 AJAX

AJAX är en grupp tekniker som används för ett asynkront datautbyte mellan en webbapplikation och en server, vilket ger möjlighet till dynamiska webbsidor. Det kan exempelvis användas i ett Javascript som körs på klientsidan i en webbapplikation för att skicka förfrågningar till webbservern. Den förfrågade filen exekveras på webbservern som sedan skickar svaret till klienten. Svaret kan sedan i sin tur användas för att kontinuerligt uppdatera webbsidan utan att ladda om den. [33]

3.3.6 JSON

JSON är en öppen standard som huvudsakligen används för att skicka data mellan server och applikation. Det är ett lättviktsalternativ till XML vilket betyder att det kräver mindre kodning för att nå samma resultat. JSON är språkoberoende och kan användas direkt med bland annat PHP och Javascript. JSON kan användas för att konvertera matriser med värden från databaser via PHP och returnera dessa i en överskådlig syntax. [34]

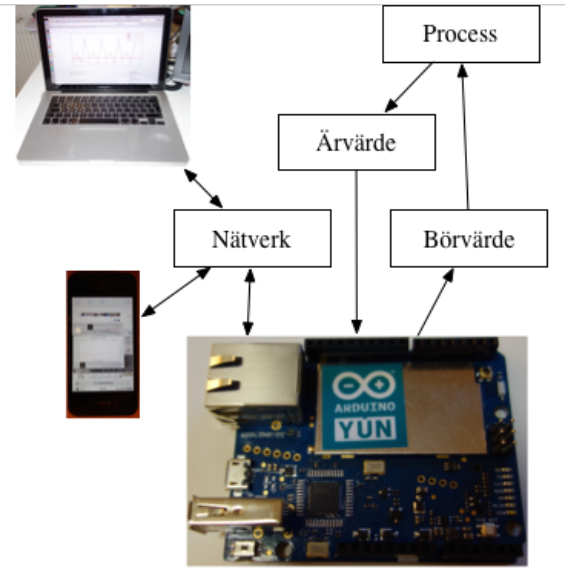
3.3.7 Skal

OpenWrt använder sig av paketet Busybox vilket är en sammansättning av program som är resurssnåla och tänkta att fungera väl för integrerade Linuxsystem [35]. Med Busybox kommer skalet ash som saknar dokumentation men använder sig av POSIX-standarden till skillnad från andra skal som kan avvika från detta. Åtkomst till skalet sker över SSH vilket innebär att all syntax måste vara POSIX-korrekt.

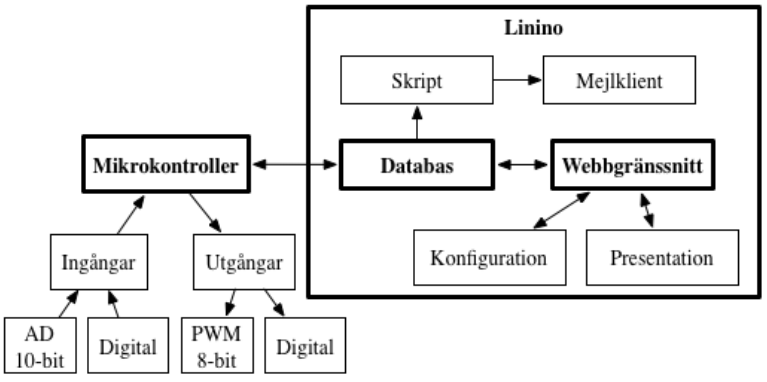
Kapitel 4

Genomförande

En universell IoT-enhet utvecklades med möjlighet för konfiguration och presentation av data genom ett webbgränssnitt. Användaren kunde ansluta till plattformen Via ett lokalt nätverk och då övervaka eller konfigurera processer. Detta framgår i figur 4.1. Mikrokontrollern utvecklades för att arbeta autonomt med registrering av data samt styrning av processer. Kommunikation mellan mikrokontrollern och omvärlden skedde genom en databas i Linux. Databasen innehöll all information för mikrokontrollern och gav möjligheten att spara registrerad data. Program på systemkretsen gjorde databasförfrågningar för att presentera rätt information för användaren i webbgränssnittet eller via mejl. Hur hela enheten var sammankopplad framgår av blockschemat i figur 4.2.



FIGUR 4.1: Plattformens kommunikation med nätverk och processer.



FIGUR 4.2: Blockschema över enheten.

4.1 Kravspecifikation

En kravspecifikation togs fram med krav som är generella för en enhet med den efterfrågade funktionen. En del krav tillkom från användningen av Arduino Yún.

4.1.1 Mikrokontroller

- Data från ingångar ska sparas med en tidstämpel
- Konfiguration ska ske autonomt

4.1.2 Konfiguration

- Konfiguration ska kunna ske genom ett webbgränssnitt
- Analoga utgångar ska styras med PID-regulator
- Digitala utgångar ska kunna fungera med timer och oberoende av ingång
- Alla utgångar ska kunna konfigureras som återkopplade
- Alla analoga ingångar måste ha ett datablad för att presentera värden med rätt måttenhet
- Samplingstid ska kunna anges för alla in- och utgångar
- Alla utgångar ska kunna specificeras med starttid och hur länge reglering ska pågå
- Digitala utgångar ska kunna ha en fördröjning
- Nivåer för specifika ingångar ska kunna specificeras som maximi- och miniminivåer

4.1.3 Presentation

- Analoga ingångar ska presenteras i form av en graf
- Börvärdet för analoga utgångar ska kunna visas
- Måttenheten för graferna ska presenteras från databladet

- Ett värde som motsvarar rätt måttenhet ska visas (istället för AD-omvandlat värde)
- Digitala ingångar ska visas som av eller på
- Digitala utgångar ska visas med det värde de bör ha
- Mejl med statusrapport ska kunna skickas till användaren med specificerat tidsintervall
- Mejl ska skickas till användaren om specificerade nivåer över- eller underskrids

4.1.4 Arduino Yún specifikt

- Tjugo ben som kan konfigureras som in eller utgångar
- Tolv stycken av benen kan vara analoga ingångar med 10-bitars AD-omvandling
- Sju stycken av benen kan fungera som PWM-utgångar i 8-bitars upplösning

4.2 Hårdvara

Valet av den resurssnåla mjukvaran medförde att alla utvecklade filer och program fick plats på systemkretsens minne vilket gjorde att läsningen och skrivningen utfördes effektivt.

4.3 Databas

Databasen spelar en central roll då detta projekt byggde på insamling och lagring av data. Det var därför viktigt att noggrant planera hur och var databasen skulle byggas upp. Efter en analys av olika databastyper och databashanterare valdes SQLite. Det var en databashanterare som lämpade sig väl eftersom enheten krävde snabb och högfrekvent tillgång till databasen. Dessutom erbjöd SQLite möjligheten att använda en lokalt placerad fil på enhetens minneskort för att underlätta åtkomst via mikrokontrollern.

4.3.1 ER-diagram

I bilaga A finns visualiseringar av databasen i form av två ER-diagram. Tabellen "_Variables"visas i en egen figur då den inte har någon koppling till den övriga databasen. De olika tabellerna i diagrammet förklaras nedan:

- Setup: pinNr är ett unikt värde för att specificera varje ben på enheten. Setup innehåller basinformation som både in- och utgångar behöver ha. Där sparas värden för att definiera om givet ben är en in- eller utgång, digital eller analog och vad samplingstiden är.
- Input: Ingångar kan ha flera datavärden med olika tidsstämplar. Analoga ingångar måste ha ett datablad medan digitala inte kan ha något datablad. Ingångar ärver information från Setup och har ett attribut med namnet på givartypen vilket används för att hitta rätt datablad.
- Output: Alla utgångar har en start- och stopptid och är generellt återkopplade däremot behöver inte digitala utgångar vara återkopplade. Alla analoga utgångar har variabler som används för PID-reglering och ett börvärde. Börvärdet sparas som ett 8-bitars tal för att det ska kunna användas direkt av mikrokontrollern för PWM-utgångar. Om utgången är digital kan den ha en digital fördröjning, vilket är en tidskonstant för hur länge efter den sista positiva återkopplingen som processen ska fortsätta vara på.
- Datasheet: Varje datablad tillhör en specifik givartyp. Datablad antas vara linjära och användaren har möjlighet att specificera antalet värden mellan vilka linjaritet antas. Varje datablad består av en måttenhet och minst två datapunkter bestående av ett in- och utdatavärde vardera. Utifrån dessa par av datavärden antas databladet vara linjärt och en inter- eller extrapolering kan ske för att konvertera den tillhörande givarens insamlade data till motsvarande måttenhet. Interpolation och extrapolation var metoder för att skapa datapunkter utifrån redan existerande datapunkter. Interpolation användes då en av koordinaterna för datapunkten som skulle skapas låg mellan de redan existerande datapunkterna och extrapolation i fallet då datapunkten låg utanför de existerande datapunkterna.
- DataIn: Består av sparad data från ingångar, pinNr ärvs från ingångarna. Varje rad med sparad data består av ett ärvärde och en tidpunkt. Analog data innehåller även ett, utifrån tillhörande datablad, konverterat ärvärde.
- ReportAndAlarm: Här sparades en mejladress till användaren och det eventuella tidsintervallet för rapportering av alla ingångars aktuella värde. Tabellen innehöll även en maximi- och miniminivå för varje ingång.
- _Variables: En tabell utan relation till övriga tabeller. Tabellens enda funktion var att temporärt spara variabler som användes vid konvertering av ärvärde.

4.3.2 Databaslösningar

Under projektets gång skedde kontinuerlig utveckling av databasen i takt med att nya funktioner och problem uppstod. När många ärvärden samtidigt skulle hämtas från databasen upptäcktes det att det var ineffektivt att konstant konvertera alla ärvärden via PHP. Varje gång hämtning av data skedde så konverterades ärvärdena enligt databladet, även de värden som redan hade konverterats i tidigare hämtning. Därför implementerades en trigger för att via ett datablad konvertera respektive givares ärvärde direkt när det sparades i databasen.

Triggern gjorde en kontroll om det AD-omvandlade värdet existerade i databladet. I det fallet genomfördes direkt en konvertering till önskat ärvärde, i annat fall skedde konverteringen med hjälp av antingen interpolering eller extrapolering. För att utföra konverteringen krävdes variabelhantering vilket SQLite inte stödde. Då undersöktes alternativet att skapa variabler i temporära tabeller. Detta visade sig inte vara tillåtet att göra inuti en trigger. Lösningen blev att skapa en tabell med kolumner som alternativ till användning av lokala variabler. Triggern gjorde insättningar och förfrågningar till denna tabell för att spara de värden som behövdes vid konverteringen. Variabeltabellen tömdes i slutet av exekveringen då variablerna inte längre var aktuella.

Det implementerades även två triggers vilkas syfte var att direkt då en givare konfigurerades eller togs bort, göra en insättning eller borttagning av dess specifika bennummer i tabellen som hanterar rapportering och alarm för givare i systemet. Dessa triggers användes för att kunna skapa en lista med alla existerande givares namn i webbgränssnittet för konfigurering av maximi- och miniminivåer.

Ett problem som uppstod med valet av SQLite som databashanterare var att databasen låstes vid skrivning eftersom databasen bestod av en fil. Det resulterade i att åtkomst av databasen inte var möjlig när skrivning utfördes. Tester utfördes och skrivning med tre sekunders intervall var den kortaste kontinuerligt fungerande tiden. Skrivning av mikrokontrollern begränsades då till fem sekunder för att ha en god marginal. Problem uppstod dock igen vid ritning av grafen samtidigt som skrivning till databasen skedde, när dessa två databastransaktioner utfördes samtidigt låstes databasen.

4.4 Linux

Linino har OpenWrts gränssnitt för konfiguration, LuCI vilket gjorde att det gick fort att komma igång med konfigurationen. Detta gränssnitt finns för att användaren ska slippa konfigurationsfiler i Linux, konfigurationsfilerna lämnade dock större möjlighet för egen konfiguration. Det första som gjordes var att ansluta enheten till ett nätverk, ett statiskt IP-nummer valdes för enkel åtkomst genom att ställa in parametrarna:

```
# cat /etc/config/network
    config interface 'wan'
        option _orig_ifname 'eth1'
        option _orig_bridge 'false'
        option proto 'static'
        option ipaddr '192.168.1.242'
        option netmask '255.255.0.0'
```

Nätverksanslutningen behövdes för åtkomst via SSH samt för att kunna installera program för den önskade konfigurationen.

För att installera program i Linino användes pakethanteringssystemet opkg och för att spara utrymme har utvecklarna valt att inte spara en lokal kopia av listan med tillgängliga paket. Det innebär att listan måste uppdateras varje gång ett program installeras. Programmen som installerades ses i tabell 4.1.

Paket	Funktion
nano	Texteditor
uhttpd	Webbserver
unzip	Packa upp filer
php5	Stöd för PHP
php5-cgi	PHP-modul för CGI
php5-mod-json	PHP-modul för JSON
php5-mod-sqlite3	PHP-modul för databashanterare
libsqlite3	Databashanterare
sqlite3-cli	Gränssnitt för databashanterare
ssmtp	Klient för att skicka epost
zoneinfo-core	För tidszonhantering
zoneinfo-europe	Europeiska tidszoner

TABELL 4.1: Installerade program.

Därefter konfigurerades uHTTPd med endast de parametrar som krävdes för grundläggande funktionalitet enligt följande:

```
# cat /etc/config/uhttpd
    config uhttpd 'main'
        list listen_http    '0.0.0.0:80'
        option home        '/mnt/root/arduino/www'
        option cgi_prefix    '/cgi-bin'
        option interpreter    '.php=/usr/bin/php-cgi'
        option index_page    'index.html index.htm default.html
        default.htm index.php'

    config uhttpd 'secondary'
        list listen_http    '0.0.0.0:8080'
        option home        '/www'
        option cgi_prefix    '/cgi-bin'
        option config        '/etc/httpd.conf'
```

Mikrokontrollern behövde arbeta mot en mapp som hette arduino, detta på grund av hur bryggan var programmerad. Mappen arduino visade sig dock endast vara för att styra digitala utgångar via webbservern. Porten för enhetens gränssnitt valdes till port 80, vilket är standardporten för HTTP-förfrågningar medan LuCI valdes till port 8080. Detta innebar att åtkomst till enhetens gränssnitt respektive LuCI skedde genom:

```
http://arduino
http://arduino:8080
```

För att kunna skicka mejl behövde sSMTP konfigureras, den konfigurationen ses nedan:

```
# cat /etc/ssmtp/ssmtp.conf
    root=user@domain.com
    mailhub=smtp.gmail.com:587
    rewriteDomain=
    hostname=user@domain.com
    UseSTARTTLS=YES
    AuthUser=user@domain.com
    AuthPass=XXXXXXXX
    FromLineOverride=YES
```

Mejladressen och lösenordet hårdkodades i filen, tanken är dock att dessa ska kunna ändras utifrån databasen där användaren kan specificera egna uppgifter.

SQLite glömde de pragman vi satt efter att sessionen stängts, det löstes genom att skapa en konfigurationsfil med inställningar som läses in vid varje programexekvering enligt:

```
# cat /root/.sqliterc
    PRAGMA synchronous = 0;      #Synkronisering avstängd
    PRAGMA foreign_keys = 1;     #Användning av främmande nycklar
```

Två skript utvecklades, det ena för att skicka mejl vid under- eller överskridna maximi- eller miniminivåer och det andra för att skicka mejl för rapportering. Problem uppstod då skripten behövde vara enligt POSIX-standard, datorerna där utvecklingen skedde använde bash vilket var friare med syntaxen. Dock räckte det med några få omskrivningar för att de skulle fungera i ash enligt POSIX-standarden.

Skriptet för larmnivåerna hämtade mejladressen för användaren och information om vilka ingångar som var aktiva från databasen. Därefter skedde en jämförelse med de satta maximi- och minimivärdena mot vad aktuellt värde var. Om maximi- och minimivärdena över- respektive underskreds skapas en fil innehållande information om givaren och dess status som därefter skickades till användaren. Ett alternativ för att larmet skulle vara mer aktuellt hade varit att konfigurera larmet med hjälp av avbrott i mikrokontrollern, vilket hade gett ett larm direkt när nivån över- eller underskreds. Detta hade krävt en inläsning av larmnivåerna till mikrokontrollern, vilket i sin tur hade krävt mer minne och processortid. Eftersom systemkretsen har en snabbare processor och mer minne valdes Linuxmiljön över mikrokontrollern då en minut var en acceptabel fördröjning för tänkta användningsområden. Detta hade emellertid inte fungerat om processen behövt åtgärdas omedelbart, dock hade en sådan process förhoppningsvis inte lämnats oövervakad.

För rapportering utvecklades ett skript vilket hämtade intervallet för rapportering och mailadressen till användaren, därefter skedde en jämförelse med klockan för att se om intervallet stämde överens. Om intervallet stämde, hämtades den aktuella statusen för ingångarna och sparades i en fil som därefter skickades till användaren.

Cron konfigurerades att exekvera skripten som utvecklades enligt:

```
# cat /etc/crontabs/root
    */1 * * * * /root/alarm.sh  #Skriptet för alarmfunktionen
    0 * * * * /root/report.sh   #Skriptet för rapportfunktionen
```

Där alarm.sh exekverades varje minut och report.sh exekverades när minuterna var 0, det vill säga varje hel timme.

I ett senare skede när all konfiguration av Linino är helt färdig bör alla program som inte används tas bort, exempel på sådana är wget och nano. Lininos grundkonfiguration innehåller även program som inte fyller någon funktion för tänkt användningsområde. Istället hade kompilering av OpenWrt kunnat ske med endast de program som är nödvändiga och därefter hade en diskavbildning kunnat skapas för en enkel konfiguration av andra enheter.

4.5 Mikrokontroller

Mikrokontrollern programmerades i Arduinos egna program som innehåller förenklade sätt att skriva, ett exempel på detta är att det räcker att skriva ”analog_read(XX)” för AD-omvandling istället för till exempel:

```
ADCON0 = 0b00111101;
ADCON0bits.GO_DONE = 1;
while(ADCON0bits.GO_DONE != 0);
```

Detta gjorde att många funktioner fanns att tillgå, dock är de möjligen långsammare. För programmet inkluderades process- och bridgebiblioteket vilket innehåller funktioner för kommunikationer med systemkretsen samt EEPROM-biblioteket vilket innehåller funktioner för att skriva till det icke-flyktiga minnet.

Hur hela programkörningen gick till framgår av bilaga B. Programmet började med en initiering av bryggan för kommunikation med systemkretsen samt initiering av den seriella kommunikationen. Mikrokontrollern ATmega32U4 har 20 ben tillgängliga i Arduino Yúns konfiguration, dessa kunde fungera enligt tabell 4.2. Benkonfigurationen lästes in genom att en process skapades via bryggan i Linuxmiljön mot databasen, resultatet från databasen sparades med tillhörande parametrar samt initierades som in- eller utgångar. Ingångarna sparades som variabler i minnet medan utgångarna sparades på det icke-flyktiga minnet, detta för att två olika möjligheter att spara information skulle jämföras.

Ben	Funktion
0	Seriell RX, externt avbrott
1	Seriell TX, externt avbrott
2	TWI (SDA), externt avbrott
3	PWM, TWI (SDC), externt avbrott
4	AD-omvandling
5	PWM
6	AD-omvandling, PWM
7	Externt avbrott
8	AD-omvandling
9	AD-omvandling, PWM
10	AD-omvandling, PWM
11	PWM
12	AD-omvandling
13	PWM
A0	AD-omvandling
A1	AD-omvandling
A2	AD-omvandling
A3	AD-omvandling
A4	AD-omvandling
A5	AD-omvandling
ICSP	SPI för shield
GND	Jord
AREF	Analog referensspänning

TABELL 4.2: Benkonfiguration på mikrokontroller.

Att spara till minnet var en enklare process, nackdelen med detta var att informationen försvinner när strömmen bryts. Mikrokontrollern startar på några sekunder jämfört med systemkretsen som kan behöva uppemot en minut, eftersom det är systemkretsen som innehåller information om konfigureringen kan detta vara problematiskt om det är en känslig process som styrs. Utgångarnas konfiguration fanns då att tillgå för mikrokontrollern vid uppstart, nackdelen är att

varje adress är en byte stor och endast ett begränsat antal (färre än 100000) skrivningar får göras.

Efter att initieringen var klar startades huvudprogrammet i en oändlig loop. I den jämfördes tiden sedan mikrokontrollern startades med sampeltiden för in- respektive utgångarna som var satt i millisekunder.

Programmet började med att kontrollera alla utgångar och avgöra om de skulle regleras, om de skulle regleras exekverades en subrutin för reglering. Subrutinen jämförde först mikrokontrollerns tid mot starttid för utgången samt dess körtid, om de är uppfyllda startar regleringen som skedde olika för analoga och digitala utgångar. All reglering började med en inläsning av ingången, detta för att ha ett värde så nära realtid som möjligt.

För analoga utgångar användes en PID-regulator enligt ekvation 4.1.

$$u(t) = K\left[e(t) + \frac{1}{T_I}\int e(t)\mathrm{d}t + T_D \times e'(t)\right] \tag{4.1}$$

K motsvarar förstärkningen, e(t) motsvarar felet, T_I motsvarar integrationstid och T_D deriveringstiden [36]. Konstanterna för PID-regulatorn lästes in från databasen av mikrokontrollern och dessutom kompletterades PID-regulatorn som implementerades med integratoruppvridningsskydd.

Om det var en digital utgång gjordes först en kontroll om det fanns en koppling till en ingång, om ingen koppling existerade (det vill säga timerbaserad) skickades ”HÖG” utsignal. Fanns däremot en tillhörande koppling gjordes en kontroll av insignalen och då statusen var ”HÖG” skickades en ”HÖG” utsignal. Var insignalens status däremot ”LÅG” gjordes en kontroll om fördröjningstimern var klar och i så fall skickades en ”LÅG” utsignal annars fortsatte programmet till att läsa ingångarnas status.

Då tidsvilkoren ej var uppfyllda gjordes en kontroll om utgången var digital eller analog och därefter skickades ”LÅG” för digitala utgångar respektive ”0” för analoga som utsignal.

Vid läsning av ingångarna gick programmet alla ingångar och en kontroll gjordes om de var digitala eller analoga, därpå lästes ingången av och en subrutin anropades för att spara de inlästa värdena till databasen på systemkretsen. Alla utgångar som hade lästs in samlades ihop och en process exekverades asynkront via bryggan för att spara ingångarnas värde i databasen. Processen valdes att köras asynkront för att mikrokontrollern inte skulle behöva vänta på att processen var klar på systemkretsen.

4.6 Webbgränssnitt

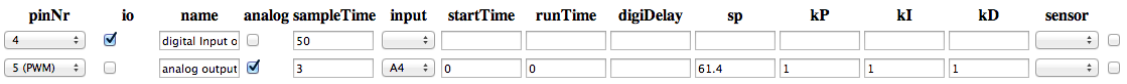
Webbgränssnittet programmerades främst i PHP men även Javascript användes för att skapa en dynamisk presentationssida som det gick att interagera med. En gemensam funktionsfil för PHP skapades, delvis för att göra koden mera överskådlig men främst för att göra programmeringen mer modulär. Funktionerna kunde anropas av alla separata filer, vilket sparade mycket tid.

4.6.1 Konfiguration

Konfigurationssidan fanns tillgänglig att kunna konfigurera enheten via ett grafiskt gränssnitt. Eftersom mikrokontrollern endast hade möjligheten att använda vissa ben för PWM och AD-omvandling, sattes denna begränsning i en matris som det gick att ändra på om till exempel en annan mikrokontroller skulle användas.

Vid exekvering av sidan lästes den nuvarande konfigurationen in från databasen tillsammans med kolumnnamn, kolumnnamn valdes då produkten var i en utvecklingsfas och databasen kunde ändras utan att ge alltför omfattande omskrivningar.

Konfigurationen presenterades i tabellform i ett formulär som kunde skickas till databasen. De olika kolumnnamnen krävde tre olika typer av formulärelement; textruta, kryssruta och lista vilket sköttes dynamiskt utifrån kolumnnamnen. Fanns det information om konfigurationen i databasen skrevs det in, annars presenterades tjugo rader. Hur två rader kunde konfiguraras framgår av figur 4.3.



FIGUR 4.3: Ett exempel med två rader av konfiguration.

Raderna för konfiguration kunde tas bort antingen individuellt eller flera rader samtidigt. På grund av databasutformningen behövde vald rad först tas bort som in- eller utgång varpå den kunde tas bort från setup. Detsamma gällde för att lägga till en rad som konfiguration, först i setup och därefter in- eller utgång. Om det var en analog utgång behövde det av användaren specificerade börvärdet omvandlas till ett 8-bitars tal för att sparas i databasen. Den omvandlingen skedde genom interpolering eller extrapolering utifrån värden i datablad som var kopplade till den analoga utgångens återkopplade ingång. Allt som skulle läggas till i databasen behövde först passera en validering för att inte felaktig inmatning skulle ske, samt för att göra en kontroll att inte någon felaktig konfiguration skett. Valideringen sköttes dels genom en typomvandling för värden som kan vara ”0” i databasen, andra värden kontrollerades för att vara korrekt inmatade. Denna kontroll hade kunnat kompletteras av Javascript för att direkt informera användaren om felaktig inmatning hade skett. Dock går Javascript att stänga av i webbläsare vilket gjorde att valideringen i PHP var nödvändig.

Om konfigurationen hade ändrats krävdes en omstart av mikrokontrollern för ny inläsning, PHP programmerades då att exekvera följande kommando i Linuxmiljön:

```
/usr/bin/reset-mcu
```

4.6.2 Datablad

Databladens funktion var att utifrån det AD-omvandlade värdet kunna skapa ett värde som motsvarade ett intressant värde för användaren. Exempelvis kan temperaturen visas i grader istället för ett 10-bitars tal i form av det AD-omvandlade värdet. Utifrån de av användaren angivna värdena antogs databladet vara linjärt och punkter som ligger inom respektive utanför intervallet interpolerades respektive extrapolerades. Databladen konfigurerades på ett snarlikt sätt som mikrokontrollern i webbgränssnittet, den stora skillnaden var att det inte fanns något fördefinierat antal datablad. Användaren kunde skapa nya datablad och ändra på befintliga datablad.

Sidan skapades genom att de konfigurerade databladen lästes in och önskat datablad gick att välja via en lista eller så fanns möjligheten att skapa ett nytt datablad. Tabellen skapades efter kolumnnamn, dock endast med textfält samt kryssruta för att kunna ta bort rader i databasen. Denna information skickades till databasen, därpå valde sidan rätt datablad och presenterade all information som fanns i databasen, alternativt skapades ett nytt datablad med användarens önskade antal rader. Om ett nytt datablad skapades kunde användaren lägga till information som namn, måttenhet och värden. Raderna kunde tas bort och läggas till med samma funktion som i konfigurationssidan. Inmatad text validerades också med samma funktion som i konfigurationssidan, dock med andra gällande villkor.

Vid ändring av datablad interpolerades eller extrapolerades tidigare konverterade ärvärden utifrån det nya databladet. Denna funktion var nödvändig då felaktiga värden kunde ha angivits i databladet samt extrapolationen kunde ge felaktiga värden, vilka då kan korrigeras.

4.6.3 Larm och rapport

Konfigurationssidan för att ställa in larmnivåer för maximi- och miniminivåer skapades utifrån databasen där nya rader skapades av triggern när en ingång konfigurerades. Detta innebar att hela tabellen hämtades från databasen och visades i tabellformat. I tabellen kunde användaren specificera sin mejladress, ange inom vilket intervall statusrapporten önskades samt definiera maximi- och minimivärden när larm skulle skickas för varje befintlig ingång. Namnet på ingången tilläts inte att ändras i denna sida utan baserades på tidigare konfigurerade namn. Funktionellt var principen samma för denna sida som tidigare konfigurationssidor, vid ändring sparades det i databasen. Rader som användaren önskade ta bort lämnades tomma och rader som lades till validerades innan de lades in i databasen.

4.6.4 Hämtning av analog data

Data att presenteras i Flot hämtades i två olika filer, varav den ena filen hämtade namnen på alla analoga in- och utgångar som skulle visas som kryssrutor. Anledningen till att

detta var uppdelat var att Flot skrev över all tidigare hämtad information vid varje ny hämtning, vilket inte var önskvärt för kryssrutorna.

Den andra filen hämtade data att visa som en graf i Flot. Från databasen hämtades ett av användaren specificerat antal datapunkter med namnet på aktiva ingångar, måttenhet, tider samt de konverterade ärvärdena. Därpå hämtades börvärden med måttenhet och namn. Eftersom börvärdet var ett 8-bitarstal för PWM-signalen var konvertering nödvändig för att kunna visa det intressanta börvärdet.

En matris skapades med färger för att uppnå en gemensam färgkodning, en återkopplad ingång fick en mörkare färgton än associerade utgången fick en ljusare ton av samma färg. Matrisen möjliggjorde även att det gick att lägga till nya färger samt ändra de befintliga färgerna.

Sedan konverterades all data till en JSON-sträng som innehöll namnet på respektive in- eller utgång, tid, måttenhet och konverterat ärvärde eller börvärde. Ett exempel på JSON-formatering för en ingång:

```
{
    "Analog ingång A0":{
       "label":"PV för: Analog ingång A0",
       "data":[
                  [
                     0,
                     30
                  ],
                  [
                     10,
                     25
                  ]
              ],
       "unit":"Grader",
       "color":"#ff0000"
    }
 }
```

4.6.5 Hämtning av digital data

Den digitala data som skulle presenteras hämtades via en fil som först hämtade den senaste statusen för alla digitala ingångar samt ingångarnas namn. För utgångar hämtades sedan tidsvariablerna innehållande starttid, körtid och eventuell fördröjning samt deras eventuellt återkopplade ingång. Därefter skedde en kontroll för att bestämma om utgången var återkopplad. Om så var fallet kontrollerades den återkopplade ingångens status och tidsvariablerna tillhörande utgången för att bestämma vad utgångens status borde vara. I fall utgången ej var återkopplad bestämdes vad statusen borde vara endast utifrån tidsvariablerna.

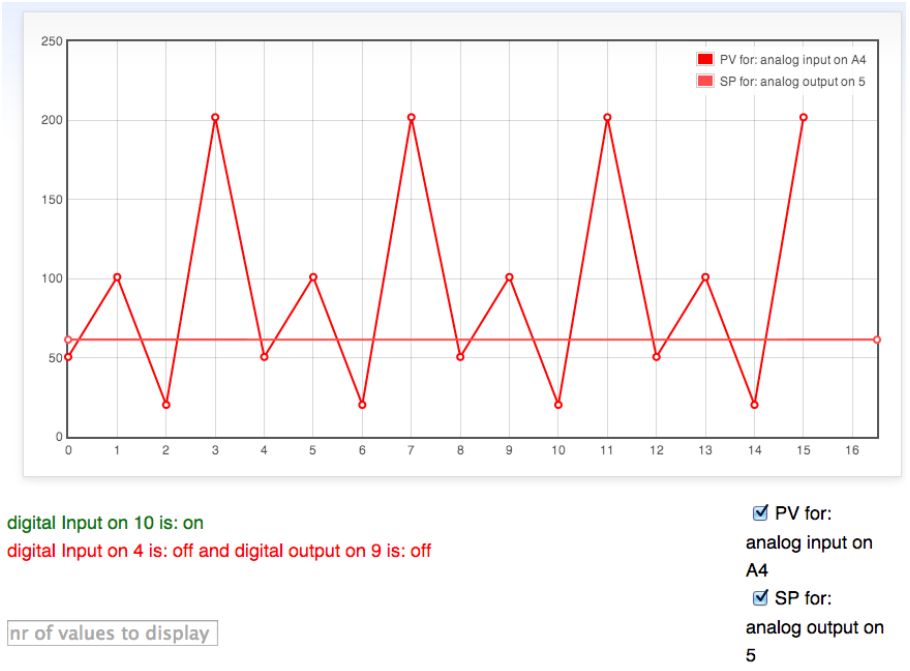
Sedan konverterades all data till en JSON-sträng som innehöll namnet på respektive in- eller utgång, tid samt status. Data som ej var applicerbar sattes till NA, det vill säga att den inte fanns tillgänglig. Ett exempel på JSON-formateringen för en ingång:

```
{
        {
                "sensorName":"Digital ingång på 1",
                "sensorStatus":"on",
                "digOutputName":"NA",
                "outputStatus":"NA"
        }
}
```

4.6.6 Presentation av registrerad data

Flot hade ett till stora delar väldokumenterat API och var enkelt att komma igång med. Svårigheten låg i att skapa JSON-strängar som var rätt formaterade enligt Flot av respektive PHP-fil. Flot hade exempel för hur kryssrutor skulle implementeras samt för hur AJAX skulle användas för asynkron datahämtning med ett specifikt tidsintervall.

Ett problem uppstod när data som skulle presenteras låg inom ett stort tidsintervall så blev tidsaxeln i grafen oläslig eftersom tidsaxeln skalades om för att visa varje sekund som passerat i grafen som var av fast storlek. Det implementerades då en funktion att begränsa antalet datapunkter som skulle visas i Flot. Ett exempel på hur presentationen kunde se ut framgår av figur 4.4.



FIGUR 4.4: Presentationen av registrerad data.

Först hämtades JSON-strängen med information för kryssrutorna som skapades i Flot. Kryssrutornas syfte var att ge användaren möjlighet att kunna välja vilka kurvor som skulle presenteras i grafen. Sedan skedde en hämtning av data för att presentera den i grafen utifrån antalet punkter som efterfrågat av användaren. Data hämtades som en JSON-sträng från PHP-filen med hjälp av AJAX och presenterades i grafen utifrån de av användaren valda kurvorna. Hämtningen av data att presentera i grafen skedde var femte sekund, vilket var samma tidsintervall som data kunde registreras av ingångarna.

Därefter hämtades digital data som en JSON-sträng via PHP med hjälp av AJAX. Beroende på om en ingång existerade visades namnet för ingången, ingångens status, utgångens namn samt vad utgångens status borde vara. Ifall ingen ingång var återkopplad till utgången presenterades vad utgången borde vara och för ingångar visades namn och status. Detta presenterades som text och utifrån vad statusen på ingången var eller hur utgångens status skulle vara presenterades detta med olika färg på texten, röd text för ”LÅG” och grön text för ”HÖG”.

Kapitel 5

Slutsats

Målet med projektet var att skapa en universell enhet för registrering av data och styrning av processer enligt given kravspecifikation. Hårdvaran som valdes var Arduino Yún eftersom det var en enhet som hade de enligt kravspecifikationen eftersökta egenskaperna gällande både trådlöst och trådbundet nätverk, systemkrets och mikrokontroller. Arduino Yún leverades med Linino som byggde på OpenWrt vilket var en Linuxdistribution för inbyggda system. Linino hade ett pakethanteringssystem som erbjöd flertalet program och de som användes valdes främst utifrån sina låga prestandakrav. SQLite valdes som databashanterare eftersom det uppfyllde kraven gällande enkelhet och funktionalitet, uHTTPd var webbservern som valdes utifrån sin prestanda och sSMTP var en lättviktsklient för att endast skicka mail. Databasen var relationsbaserad och utformades enligt kraven på enheten. En funktion utvecklades för att kunna visa ärvärden med rätt måttenhet för presentation. Mikrokontrollern programmerades i C för att arbeta autonomt. Det åstadkoms genom att den först initierades från databasen och styrning kunde ske i realtid samtidigt som registrering av data skedde. Gränssnittet för presentation och konfiguration utvecklades för att fungera webbaserat. Insamlad data presenterades i en sida utvecklad utifrån Flot med databashämtningar i PHP. En graf presenterades för analoga in- och utgångar samt endast en textbaserad status för digitala in- och utgångar. Sidor utvecklades i PHP med tabeller för att kunna erbjuda ett gränssnitt för konfiguration av enheten, rapportering och datablad. De ställda kraven på funktion för enheten uppfylldes och efter att enheten var ansluten till ett lokalt nätverk kunde åtkomst till enheten ske via trådlöst eller trådbundet nätverk. Genom webbservern på enheten kunde användaren kommunicera med mikrokontrollern.

5.1 Kritisk diskussion

Inledningsvis lades mycket tid på analys för hur databasen skulle utformas, vilket gick emot projektets agila arbetsmetoden. Eftersom specifikationen för databasen uppdaterades kontinuerligt under projektets gång blev det första utkastet av databasen snabbt föråldrat.

Problemet med att databasen var låst vid skrivning löstes ej, efter testning drogs slutsatsen att det berodde på att databasen låg lokalt i form av en fil. Alternativet hade varit en databashanterare som exekverades som en bakgrundsprocess men den typen av databashanterare valdes bort under inhämtningen av information i förberedelsefasen av projektet då det ansågs för resurskrävande. Detta var ett problem som behövde mer tid för att lösas genom testning av olika alternativ.

Eftersom arbetet med webbutvecklingen skedde parallellt blev utvecklingen av konfigurationssidorna klar innan utvecklingen av presentationen som mest utvecklades i Javascript. När presentationssidan var färdigutvecklad hade ny kunskap inhämtats om Javascript vilket öppnade för nya möjligheter att utveckla konfigurationssidorna på ett mer användarvänligt sätt. Dessutom borde fler PHP-sidor skapats dynamiskt genom funktioner eftersom PHP kunde generera den HTML-kod som önskades. Två triggers som skapades i databasen hade kunnat lösas effektivare med hjälp av en databasförfrågan med hjälp av PHP. Däremot hade en trigger varit effektivare för att hantera insättning och borttagning av in- eller utgångar.

Lagringen av variabler på mikrokontrollern för styrningen placerades på det icke-flyktiga minnet för att de skulle finnas tillgängliga vid uppstart om det var en känslig process. Detta var dock inte nödvändigt då enheten inte kommer vara utvecklad för dessa typer av processer, en bättre lösning hade varit att spara variablerna i minnet och lämna samma uppstartstid för mikrokontrollern som systemkretsen. Lösningen med variablerna i minnet hade troligen resulterat i ett snabbare program.

Efter testning visade det sig att mikrokontrollern arbetade autonomt och utifrån en inläsning av information från databasen. Webbgränssnittet för grafiskpresentationen av data uppfyllde kraven på interaktivitet och visning av önskad data. Enheten har en stor potential för att spara energi vilket hade resulterat i god miljöpåverkan, enheten hade till exempel kunnat användas för att stänga av elektronik i hemmiljö.

5.2 Fortsatt utveckling

Enheten har flera användningsområden och utveckling behöver ske för följande scenarion:

- Periodvis användning för processer där den registrerade datan kan sparas eller tas bort.
- Användning då enheten blir strömlös och registrerar data kontinuerligt, i det fallet behöver mikrokontrollern börja sin programkörning utifrån den senast sparade tiden i databasen. En kompletterande lösning till detta hade kunnat vara strömförsörjning via en batterikälla.

Alternativa lösningar för databashantering:

- En databashanterare som exekveras som en bakgrundsprocess hade varit en möjlig lösning. Detta hade troligtvis varit mer prestandakrävande och det behöver undersökas på enheten innan det kan implementerades som en lösning. Det hade dock medfört en för mikrokontrollern mer komplicerad hämtning och registrering av data.
- En annan lösning hade varit att ha en central databas som tar hand om databasfunktionaliteten för alla produkter vid eventuell serieproduktion.
- Det finns även ett alternativ att använda mellanlagring av data vid skrivning. Detta skulle kunna ske i en textfil och sedan till SQLite eller genom att använda sig av en bakgrundsprocess samtidigt som mikrokontrollern endast skriver till en SQLite-databas.

Fler alternativ till interaktivitet med grafen hade kunnat ges genom att implementera flera funktioner som Flot har stöd för, exempelvis genom att kunna välja ett tidsintervall att visa i grafen. Ett interaktivt gränssnitt för konfiguration av utgångar likt Simulink i Matlab hade varit önskvärt.

Vid eventuell serieproduktion av enheten skulle det lönat sig att bygga enheten utifrån den önskade funktionaliteten med lämplig systemkrets samt mikrokontroller. En möjlig komplettering skulle kunna vara med ett 4G-modem med en central databaslösning då all konfiguration och presentation hade kunnat ske i molnet. Eftersom det kan vara svårt att ställa in en PID-regulator rätt hade en självinställande PID-regulator kunnat implementeras för ökad användarvänlighet.

En annan möjlig lösning för att få ner kostnaderna vid eventuell serieproduktion hade varit att endast använda en mikrokontroller som kommunicerar med en central databas. Arduino Yún kostar 621 kr [37] medan mikrokontrollern som finns på Yún-plattformen tillsammans med en WLAN-modul hade kostat cirka 50 kronor [38, 39] vid serieproduktion.

Litteraturförteckning

[1] N. Valéry, "Welcome to the thingternet," *The Economist*, nov 2012, Tillgänglig: http://www.economist.com/news/21566428-things-brather-people-are-about-become-biggest-users-internet-welcome [Hämtad: 2014-05-21].

[2] Xively. About. [Online] Tillgänglig: https://xively.com/about [Hämtad: 2014-03-24].

[3] "Xively launches consulting services for internet of things," *Telecomworldwire*, sep. 18 2013, Tillgänglig: ProQuest, http://search.proquest.com.proxy.lib.chalmers.se/docview/1433276503 [Hämtad: 2014-04-18].

[4] ThingWorx. What is ThingWorx? [Online] Tillgänglig: http://www.thingworx.com/platform/ [Hämtad: 2014-05-22].

[5] Carriots. Tutorials. [Online] Tillgänglig: https://www.carriots.com/tutorials [Hämtad: 2014-05-22].

[6] A. Bawaba, "Microchip and element14 announce raspberry pi(r) chipkit expansion board; worlds first with prototyping-friendly 32-bit mcu package," *MENA Report*, 19 sep 2013, Tillgänglig: ProQuest, http://search.proquest.com/docview/1433440464 [Hämtad: 2014-03-20].

[7] Arduino. Yún. [Online] Tillgänglig: http://playground.arduino.cc/Hardware/Yun [Hämtad: 2014-03-20].

[8] Arduino. Introduction. [Online] Tillgänglig: http://arduino.cc/en/Guide/Introduction [Hämtad: 2014-03-20].

[9] M. C. och J. Sanchez, *Microcontrollers High-Performance Systems and Programming*, Boca-Raton: CRC-Press, 2013.

[10] Atmel, ATmega32U4, Feb 2014, [Online] Tillgänglig: http://www.atmel.com/Images/Atmel-7766-8-bit-AVR-ATmega16U4-32U4_Summary.pdf [Hämtad: 2014-03-20].

[11] R. Rajsuman, *System-on-a-chip: design and test*, Norwood: Artech House, 2000.

[12] Atheros, AR9331, Dec 2010, [Online] Tillgänglig: https://www1.elfa.se/data1/wwwroot/assets/datasheets/AR9331_eng_tds.pdf [Hämtad: 2014-03-21].

[13] Arduino. Bridge Library for Arduino Yún. [Online] Tillgänglig: http://arduino.cc/en/Reference/YunBridgeLibrary [Hämtad: 2014-03-21].

[14] R. B. och J. Edwards, *Networking: OSI, TCP/IP, LANs, MANs, WANs, Implementation, Management, and Maintenance*, Hoboken: Wiley Publishing, 2009.

[15] Linino. Welcome to linino.org. [Online] Tillgänglig: http://wiki.linino.org/doku.php [Hämtad: 2014-03-22].

[16] L. Garber, "News brief (companies will develop framework for the internet of things)," *Computer*, vol. 47, no. 2, pp. 16 – 20, mar 2014, Tillgänglig: IEEE Xplore, http://ieeexplore.ieee.org.proxy.lib.chalmers.se/xpl/articleDetails.jsp?arnumber=6756863 [Hämtad: 2014-04-18].

[17] OpenWrt. About. [Online] Tillgänglig: http://wiki.openwrt.org/about/start [Hämtad: 2014-03-22].

[18] OpenWrt. Database Mananagement systems. [Online] Tillgänglig: http://wiki.openwrt.org/doc/howto/database.overview [Hämtad: 2014-03-24].

[19] X. S. L. Junyan and L. Yijie, "Application research of embedded database sqlite," in *International Forum on Information Technology and Applications*, Changdu, 15 - 17 maj 2009, pp. 539 – 543.

[20] SQLite. About. [Online] Tillgänglig: https://sqlite.org/about.html [Hämtad: 2014-03-22].

[21] Debian. sSMTP. [Online] Tillgänglig: https://wiki.debian.org/sSMTP [Hämtad: 2014-05-21].

[22] dd-wrt. Cron. [Online] Tillgänglig: http://www.dd-wrt.com/wiki/index.php/CRON [Hämtad: 2014-05-22].

[23] OpenWrt. Web Server Overview. [Online] Tillgänglig: http://wiki.openwrt.org/doc/howto/http.overview [Hämtad: 2014-03-24].

[24] OpenWrt. uHTTPd. [Online] Tillgänglig: http://wiki.openwrt.org/doc/howto/http.uhttpd [Hämtad: 2014-03-22].

[25] OpenWrt. Statistical Data Overview. [Online] Tillgänglig: http://wiki.openwrt.org/doc/howto/statistical.data.overview?s[]=chart [Hämtad: 2014-03-24].

[26] O. Laursen. Flot: Attractive JavaScript plotting for jQuery. [Online] Tillgänglig: http://www.flotcharts.org/ [Hämtad: 2014-03-22].

[27] R. H. et al., *Oracle Application Express for mobile web applications*, New York: Springer Science+Business, 2013.

[28] PHP. What can PHP do? [Online] Tillgänglig: http://www.php.net/manual/en/intro-whatcando.php [Hämtad: 2014-03-24].

[29] W3Schools. HTML Introduction. [Online] Tillgänglig: http://www.w3schools.com/html/html_intro.asp [Hämtad: 2014-05-26].

[30] J. Pollock, *JavaScript - A Beginner's Guide*, 4th ed., New York: McGraw-Hill, 2013.

[31] "appendto: Jquery overtakes flash on world's top websitess," *Entertainment Close - Up*, aug 29 2011, Tillgänglig: ProQuest, http://search.proquest.com/docview/885540951 [Hämtad: 2014-04-18].

[32] jQuery. jQuery. [Online] Tillgänglig: http://jquery.org [Hämtad: 2014-05-26].

[33] jQuery. jQuery.ajax(). [Online] Tillgänglig: http://api.jquery.com/jQuery.ajax [Hämtad: 2014-05-26].

[34] JSON. Introducing JSON. [Online] Tillgänglig: http://json.org [Hämtad: 2014-05-26].

[35] OpenWrt. Busybox. [Online] Tillgänglig: http://wiki.openwrt.org/doc/devel/packages/busybox [Hämtad: 2014-05-21].

[36] B. Thomas, *Modern Reglerteknik*, 3rd ed., Falköping: LIBER, 2003.

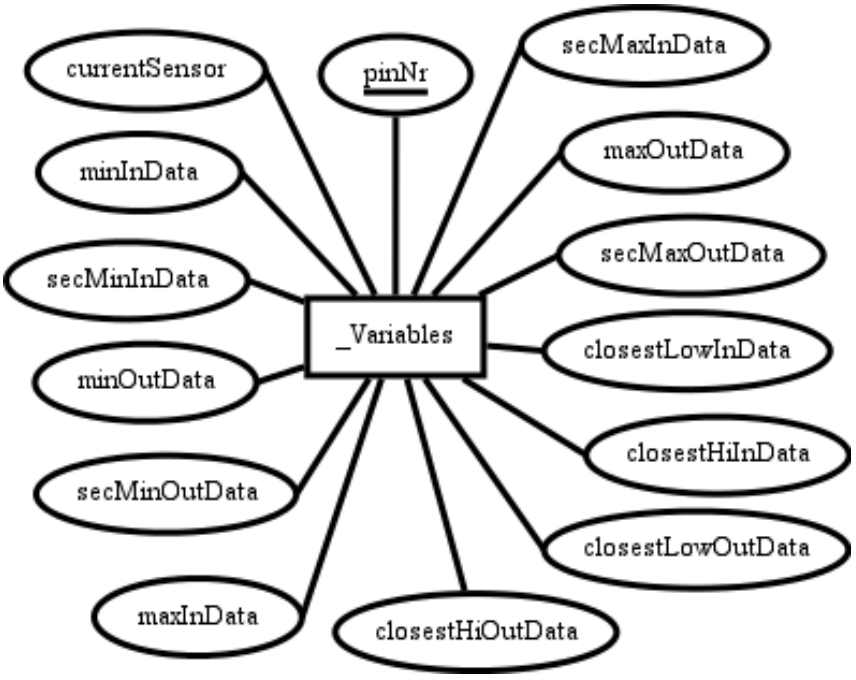
[37] Farnell element14. Arduino A000008. [Online] Tillgänglig: http://se.farnell.com/arduino/a000008/atmega32u4-yun-wifi-evaluation/dp/2356914?Ntt=yun [Hämtad: 2014-06-18].

[38] Farnell element14. ATMEL ATMEGA32U4. [Online] Tillgänglig: http://se.farnell.com/atmel/atmega32u4-mu/mcu-8bit-atmega-16mhz-vqfn-44/dp/2425127 [Hämtad: 2014-06-18].

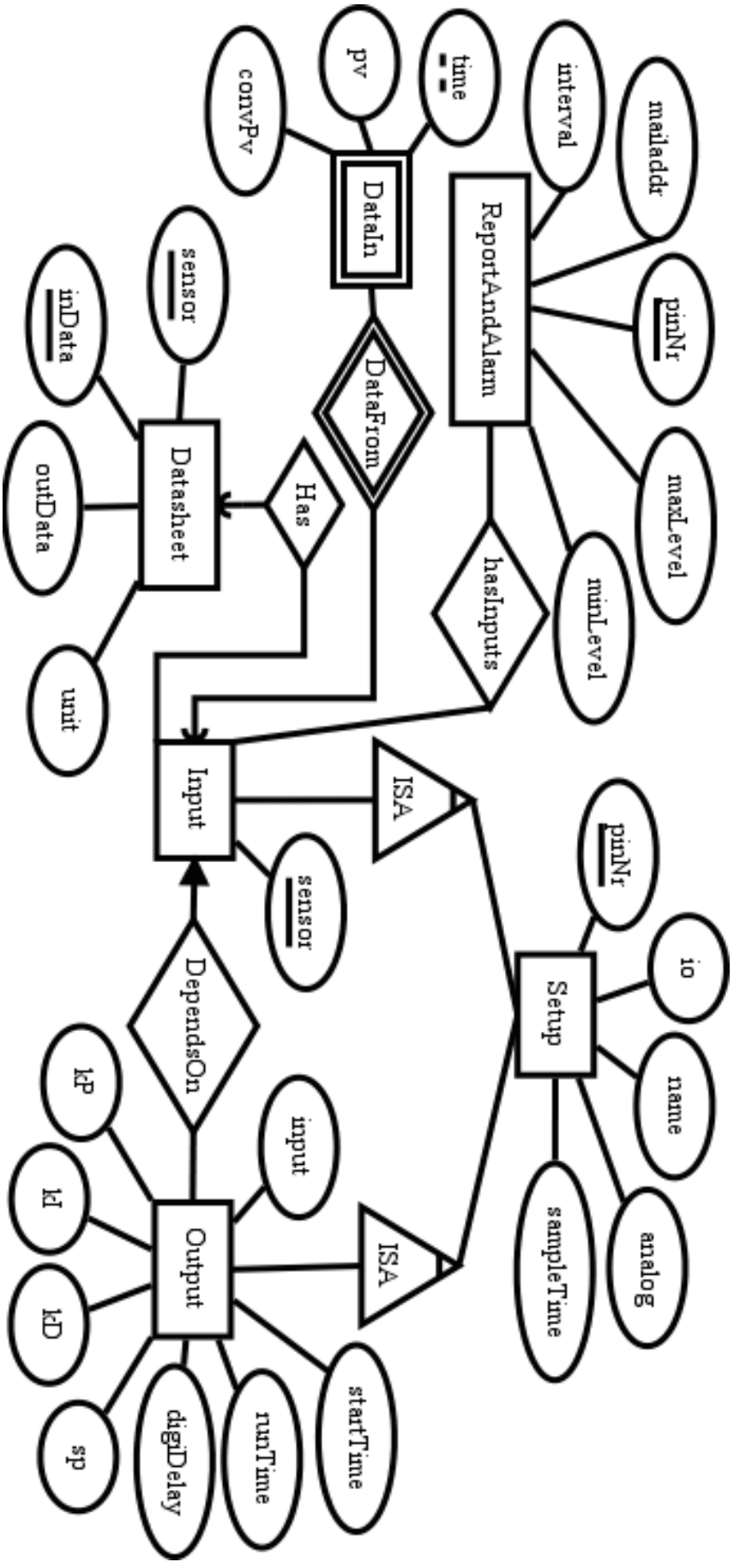
[39] Farnell element14. ATMEL ATR2406-PNQG 86. [Online] Tillgänglig: http://se.farnell.com/atmel/atr2406-pnqg-86/ic-rf-if-trans-2-4ghz-qfn-32/dp/2075856 [Hämtad: 2014-06-18].

Bilaga A

ER-diagram



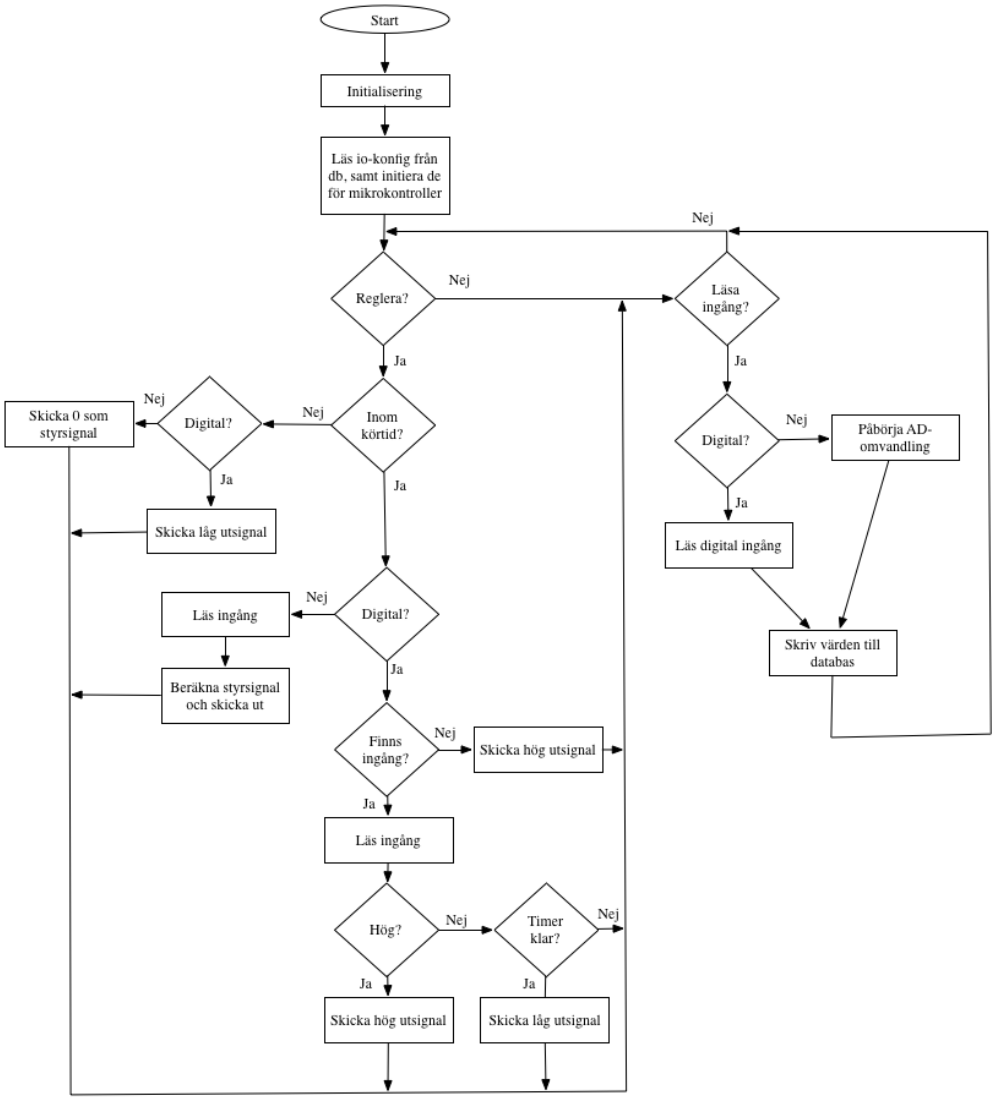
FIGUR A.1: ER-diagram över tabellen _Variables



FIGUR A.2: ER-diagram över enhetens databas.

Bilaga B

Mikrokontrollerns programkörning



FIGUR B.1: Flödesschema programkörning mikrokontroller.

Bilaga C

Skript

Alarm.sh

```
# cat /root/alarm.sh
  #!/bin/sh
  #

  dbFile="/root/arduino/www/main.db"

  queryGetMail="SELECT DISTINCT mailaddr FROM ReportAndAlarm"
  mail="$(sqlite3 $dbFile "$queryGetMail")"
  subject="Subject: ALARM Yun
   "

  queryGetActive="SELECT DISTINCT pinNr FROM DataIn"
  YUN_pins="$(sqlite3 $dbFile "$queryGetActive")"
  alarm=""

  for pin in $YUN_pins
  do
      queryAlarm="SELECT status FROM (SELECT MAX(time),
      pv, maxLevel, minLevel, CASE WHEN maxLevel < pv THEN 'HIGH'
      WHEN minLevel > pv AND minLevel != '' THEN 'LOW' END AS status
      FROM DataIn NATURAL JOIN ReportAndAlarm WHERE pinNr = $pin)"
      temp_alarm="$(sqlite3 $dbFile "$queryAlarm")"
      if [ "" != "$temp_alarm" ] #check if set
      then
          alarm=$alarm+"$pin is $temp_alarm
             "
      fi
  done

  if [ "" != "$alarm" ]
```

```
        then
            echo "$subject$alarm\n" > alarm.mail
            sendmail -v "$mail"@gmail.com < alarm.mail
```

Report.sh

```
# cat /root/alarm.sh
  #!/bin/sh
  #

  dbFile="/root/arduino/www/main.db"

  queryGetInterval="SELECT DISTINCT interval FROM ReportAndAlarm"
  interval="$(sqlite3 $dbFile "$queryGetInterval")"
  hour=$(date +%H)
  hour=$(echo $hour | sed 's/^0*//')
  hour=$((hour+1))
  report=""

  if [[ $interval -ne 0 ]] && [[ $((hour%interval)) -eq 0 ]]
          then
          queryGetMail="SELECT DISTINCT mailaddr FROM
          ReportAndAlarm"
          mail="$(sqlite3 $dbFile "$queryGetMail")"
          subject="Subject: Report Yun
           "

          queryGetActive="SELECT DISTINCT pinNr FROM DataIn"
          YUN_pins="$(sqlite3 $dbFile "$queryGetActive")"

          for pin in $YUN_pins
          do
                  queryDataIn="SELECT pvPick FROM (SELECT MAX(time),
                  pv, convPv, CASE WHEN convPv != '' THEN convPv
                  ELSE pv END AS pvPick FROM DataIn WHERE
                  pinNr = $pin);"
                  temp_rep="$(sqlite3 $dbFile "$queryDataIn")"
                  if [ "" != "$temp_rep" ]        #check if set
                  then
                          report=$report"$pin is $temp_rep
                              "
                  fi
          done

          echo "$subject$report" > report.mail
          #sendmail -v "$mail"@gmail.com < report.mail
  fi
```

Bilaga D

C-kod

```
#include <Process.h>
#include <EEPROM.h>

const String dbFile = "/root/arduino/main.db "; //Location of db
const int unDef = 255;
int inPins[20][4] = {unDef};  // Definition of input pins and
//their state
int led = 13;
const byte ctrlParm = 11; // Nr of attributes in output
int **oldCtrl;  //Old time and error for control
unsigned long lastRun = 0;

void setup(){
  Bridge.begin();  //Initilize bridge
  Serial.begin(9600); //Initilize serial with baud rate 9600 b/s
  while(!Serial); //Wait for serial to become active.

  initOutputs();
  initInputs();
}

void loop(){
  int i = 0, pv, rest, analog, pinNr, sampleTime, nrOfOutputs;
  unsigned long time, timeSec;

  time = (millis() / 100);  //Time since sketchstart in tenths of
  //a sec
  timeSec = time / 10;

  nrOfOutputs = EEPROM.read(0);

  for(i = 0; i < nrOfOutputs; i++){
```

```
    sampleTime = EEPROM.read((ctrlParm * i) + 4);
    rest = time % sampleTime;  //Check if time to run

    if((rest == 0) || (sampleTime == 0)) {
      // Time for the output to control and pin#
      //(address in EEPROM)
      control(time, (ctrlParm * i + 1));
    }
  }

  if(timeSec > lastRun){
    for(i = 0; i < 20; i++){// Read inputs
      pinNr = inPins[i][0];
      analog = inPins[i][1];
      sampleTime = inPins[i][2];
      rest = time % sampleTime;  //Check if time to run

      // Check if pin is analog, set and time for sampling
      if((pinNr != unDef) && (analog == 1) && (rest == 0)) {
        inPins[i][3] = analogRead(pinNr);
      }
      //Check if pin is analog, set and time for sampling
      else if((pinNr != unDef) && (analog == 0) && (rest == 0)) {
        inPins[i][3] = digitalRead(pinNr);
      }
      else{
        inPins[i][3] = unDef;
      }
    }
    lastRun = timeSec;
    dbAdd(timeSec);
  }
}


//Function to initialize controls to EEPROM
int initOutputs(){
  int i, post = 0, nrOfOutputs, numOld = 3, address = 1;
  char c;
  String dbGetOutputs = "\"SELECT pinNr, input, analog, sampleTime,
  startTime, runTime, sp, kP, kI, kD, digiDelay FROM Output
  NATURAL LEFT JOIN Setup\"";
  String dbGetNrOfOutputs = "\"SELECT COUNT(pinNr) FROM Output\"";
  Process initdbOutputs;

  initdbOutputs.runShellCommand("sqlite3 " + dbFile +
  dbGetNrOfOutputs);
```

```
while(initdbOutputs.available()) {
   c = initdbOutputs.read();
    if(isDigit(c)){ //Save integer value
     //char from output to integer
      post = (c - '0') + post * 10;

    }else{
      nrOfOutputs = post;
    }
}
Serial.flush();
EEPROM.write(0, nrOfOutputs);

//Allocate memory for outputs
oldCtrl = (int **)malloc(nrOfOutputs * sizeof(int *));
if(oldCtrl == NULL){
  Serial.print("out of memory");
  return 0;
 }

 for(i = 0; i < nrOfOutputs; i++){
   oldCtrl[i] = (int *)malloc(numOld * sizeof(int));
   if(oldCtrl[i] == NULL){
     Serial.print("out of memory");
     return 0;
   }
 }

initdbOutputs.runShellCommand("sqlite3 " + dbFile +
dbGetOutputs);

i = 0;
post = 0;

while(initdbOutputs.available() > 0){
  c = initdbOutputs.read();  //Read char from output

  if(isDigit(c)){ //Save integer value
    post = (c - '0') + post * 10;  //char from output to integer
  }else if((c == '|') || (c == '\n')){  //Next col
      EEPROM.write(address, post);

      if(((address - 1) % ctrlParm) == 0 ){ // Check if post
      is pinNr
        pinMode(post, OUTPUT); // Init output
        oldCtrl[i][0] = post; //Only pin#
        oldCtrl[i][1] = 0; //First run, reset old time
```

```
          oldCtrl[i][2] = 0; //First run, reset old error
        }
        address++;
        post = 0;

      }
  }

  Serial.flush();
  return 0;
}


//Function to initialize in/outputs
void initInputs(){
  int post = 0, i = 0, j = 0;
  char c;
  Process initdbInputs;    //Create process
  String dbGetInputs = "\"SELECT pinNr, analog, sampleTime
  FROM Input
  NATURAL LEFT JOIN Setup\""; //Query to get inputs

    initdbInputs.runShellCommand("sqlite3 " + dbFile +
    dbGetInputs);

    while(initdbInputs.available() > 0){
      c = initdbInputs.read();  //Read char from output

      if(isDigit(c)){ //Save integer value
        post = (c - '0') + post * 10;  //char from output to
        integer
      }else if((c == '|') || (c == '\n')){  //Next col
        inPins[i][j] = post;
        post = 0;
        j++;
        if(c == '\n'){  //Next row
          if(inPins[i][1] == 0) { //Define digital inputs
            pinMode(inPins[i][0], INPUT); //Init inputs
          }
          j = 0;
          i++;
        }
      }
    }
    Serial.flush();
}
```

```
void control(int timeMs, int address){
  int ctrlOutpin, ctrlInpin, ctrlAnalog, startTime, runTime, sp,
  pv, i,
  ctrlSignalInt, on, error, timeMin, dT, oldError, digiDelay;
  float ctrlSignal, kP, kI, kD;

  timeMin = timeMs / (60 * 10);
  // calculate indexes for old values
  i = (address - 1) / ctrlParm;

  ctrlOutpin = EEPROM.read(address);
  // 50 if there is no inpin
  ctrlInpin  = EEPROM.read(address + 1);
  ctrlAnalog = EEPROM.read(address + 2);
  startTime  = EEPROM.read(address + 4);
  runTime    = EEPROM.read(address + 5);
  digiDelay  = EEPROM.read(address + 10);

  // Checks if it's time to start and runtime isn't over or
  //indefinite
  if( ( (startTime <= timeMin) && ( ((startTime - timeMin) == 0) ||
  (startTime + runTime > timeMin) ) ) || (runTime == 0) ) {
    if(ctrlAnalog == 0) { // checks if the output is digital = 0
       dT = timeMin - oldCtrl[i][2] - digiDelay;
      if(ctrlInpin == 50) { // if no inpin
        digitalWrite(ctrlOutpin, HIGH);
      }else {
        on = digitalRead(ctrlInpin);
        if(on == HIGH) { // if inpin is on, turn on digital output
          digitalWrite(ctrlOutpin, HIGH);
          oldCtrl[i][2] = timeMin;
        }
        // if no delay or delay time is over and inpin is low,
        // turn off digital output
        else if( ((digiDelay == 0) || (dT >= 0)) && on == LOW){
          digitalWrite(ctrlOutpin, LOW);
        }
      }
    }else{ // if the output is analog
        sp = EEPROM.read(address + 6);
        //Convert to float with more decimals
        kP = EEPROM.read(address + 7) / 10;
        kI = EEPROM.read(address + 8) / 10;
        kD = EEPROM.read(address + 9) / 10;

        //Input in 10 bit - output in 8 bit
```

```
        pv = analogRead(ctrlInpin) / 4;

        error = sp - pv;
        dT = timeMs - oldCtrl[i][1];

        // Calculates the output signal using a PID-algorithm
        ctrlSignal = kP * error + kI * (error * dT) +
        kD * ((error - oldError) / dT);

        if(ctrlSignal < 0){
          ctrlSignal = 0;
        }
        if(ctrlSignal > 255){ //Maxmimum value for PWM output
          ctrlSignal = 255;
        }
        ctrlSignalInt = (int) ctrlSignal;

        analogWrite(ctrlOutpin, ctrlSignalInt);

        oldCtrl[i][1] = timeMs;  //Save old time for dT
        oldCtrl[i][2] = error;  //Save old error
    }
  }else{
    if(ctrlAnalog == 0) { // checks if the output is digital = 0
      digitalWrite(ctrlOutpin, LOW);
    }else{
      analogWrite(ctrlOutpin, 0);
    }
  }
}


//Function to add to DB
void dbAdd(int timeSec){
  int i;
  Process addPv;     //Create process
  String dbAddLine = "", pv, pinNr, time;
  time = String(timeSec);

  for(i = 0; i < 20; i++){
    if(inPins[i][3] != unDef){
      pinNr = String(inPins[i][0]);
      pv = String(inPins[i][3]);
      dbAddLine = "INSERT INTO DataIn (pinNr, time, pv)
      VALUES ('" + pinNr + "','" + time + "', '" + pv + "'); "
      + dbAddLine;
    }
```

```
  }

  addPv.runShellCommandAsynchronously("sqlite3 " + dbFile + "\""
  + dbAddLine + "\" &");
  Serial.flush();  //Ensure last bit is sent
}
```

Bilaga E

PHP-kod för funktionsfil

```
<?php

$numPins = 20; //Number of physical pins
$IOPins = array(0 => 0, 1 => 1, 2 => 2, 3 => 3, 4 => 4, 5 => 5,
    6 => 6, 7 => 7, 8 => 8, 9 => 9, 10 => 10, 11 => 11, 12 => 12,
    13 => 13, 18 => "A0", 19 => "A1", 20 => "A2", 21 => "A3",
    22 => "A4", 23 => "A5", 24 => "A6", 25 => "A7", 26 => "A8",
    27 => "A9", 28 => "A10", 29 => "A11");
$PWM = array(3, 5, 6, 9, 10, 11, 13); //Pins that will work as PWM
$maxEEPROM = 255; //Maximum size to write to on EEPROM

$setIO = array();
$setDatasheets = array();
$allSetDatasheets = array();
$nameDatasheetCols = array();
$nameSetupCols = array();

$analogData = array();
$analogInputs = array();
$analogInData = array();
$analogSP = array();
$digitalInputs = array();
$digitalInData = array();
$digitalOutData = array();
$datasheets = array();

$queryReadDatasheetForOutput = "SELECT DISTINCT sensor FROM
Datasheet;";
$queryReadSetupDb = "SELECT * FROM Setup NATURAL LEFT JOIN Output
NATURAL LEFT JOIN Input;";
$queryReadAllSetDatasheets = "SELECT * FROM Datasheet;";
```

```
$queryReadDatasheetBasedOnInput = "SELECT DISTINCT sensor, unit FROM
Datasheet NATURAL LEFT JOIN Input WHERE pinNr = :pinNr";
$queryReadSetupForAlarm = "SELECT pinNr FROM Setup WHERE
name = :name";
$queryReadUnit = "SELECT DISTINCT unit FROM Datasheet WHERE
sensor = :sensor;";
$queryReadAnalogInputs = "SELECT DISTINCT DataIn.pinNr, name, sensor
FROM DataIn NATURAL LEFT JOIN Setup NATURAL LEFT JOIN Input
WHERE analog = 1";
$queryReadDigitalInputs = "SELECT DISTINCT DataIn.pinNr, name FROM
DataIn NATURAL LEFT JOIN Setup WHERE analog = 0";
$queryReadAnalogSP = "SELECT name, input, startTime, runTime, sp
FROM Output NATURAL LEFT JOIN Setup WHERE (analog = 1)";
$queryReadAnalogInDataLimit = "SELECT time, convPv FROM DataIn
WHERE (pinNr = :currentPin) ORDER BY time DESC LIMIT ";
$queryReadAnalogInData = "SELECT time, convPv FROM DataIn
WHERE (pinNr = :currentPin);";
$queryReadDigitalInData = "SELECT name, MAX(time), pv FROM DataIn
NATURAL JOIN Setup WHERE (analog = 0) AND (pinNr = :currentPin)";
$queryReadTimeLastZeroInput = "SELECT MAX(time) FROM DataIn
WHERE (pinNr = :currentPin) AND pv = 0;";
$queryReadTimeLastActiveInput = "SELECT MAX(time) FROM DataIn
WHERE (pinNr = :currentPin) AND pv = 1;";
$queryReadDigitalOutData = "SELECT name, input, startTime, runTime,
digiDelay FROM Output NATURAL LEFT JOIN Setup WHERE analog = 0";
$queryReadTimePassed = "SELECT MAX(time) FROM DataIn NATURAL JOIN
Setup WHERE analog = :analog;";
$queryReadRepAndAlarm = "SELECT mailaddr, interval, name, maxLevel,
minLevel FROM ReportAndAlarm NATURAL JOIN Setup;";

$queryAddToInput = "REPLACE INTO Input (pinNr, sensor)
VALUES (:pinNr, :sensor);";
$queryAddToOutput = "REPLACE INTO Output (pinNr, input, startTime,
runTime, digiDelay, sp, kP, kI, kD) VALUES (:pinNr, :input,
:startTime, :runTime, :digiDelay, :sp, :kP, :kI, :kD);";
$queryAddToDatasheet = "REPLACE INTO Datasheet (sensor, unit,
outData, inData) VALUES (:sensor, :unit, :outData, :inData)";
$queryAddToRepAndAlarm = "REPLACE INTO ReportAndAlarm (pinNr,
mailaddr, interval, maxLevel, minLevel) VALUES (:pinNr, :mailaddr,
:interval, :maxLevel, :minLevel)";

$queryDeleteFromInput = "DELETE FROM Input WHERE pinNr = :pinNr";
$queryDeleteFromOutput = "DELETE FROM Output WHERE pinNr = :pinNr";
$queryDeleteDatasheetPost = "DELETE FROM Datasheet WHERE
sensor = :sensor AND inData = :inData";
$queryDeleteDatasheet = "DELETE FROM Datasheet WHERE
sensor = :sensor";
```

```
function getDbConnection(){
$db = new SQLite3("../main.db") or
die('Unable to open database');

return $db;
}


function restartMCU(){
exec('/usr/bin/reset-mcu');

}


function readDbAssoc($query, $bindName, $bindValue){
$db = getDbConnection();
$dbResult = array();

$readDb = $db->prepare($query);
if(isset($bindValue)){
$readDb->bindValue($bindName, $bindValue);
}
$result = $readDb->execute() or die ('Unable to read from db');

while($row = $result->fetchArray(SQLITE3_ASSOC)){
$dbResult[] = $row;
}

return $dbResult;
}


function minutesPassed($query, $bindName, $bindValue){
$timePassed = readDbAssoc($query, $bindName, $bindValue);

$minutesPassed = $timePassed[0]['MAX(time)'] / 60;

return $minutesPassed;
}


function linInterAndExtrapol($y1, $y2, $x, $x1, $x2){
$y = $y1 + ($y2 - $y1) * ($x - $x1) / ($x2 - $x1);
```

```
return $y;
}


function convertData($currentValue, $sensor, $getOutData){
$queryReadDatasheet = "SELECT * FROM Datasheet WHERE
sensor = :sensor;";

$convertedValue = $currentValue;
$datasheets = readDbAssoc($queryReadDatasheet, ":sensor",
$sensor);
$last = 0;

foreach($datasheets as $sheetRow){
$last++;
$currentOutData = $sheetRow['outData'];
$currentInData = $sheetRow['inData'];
$allInData[$last] = $currentInData;
$allOutData[$last] = $currentOutData;
if($currentInData == $currentValue){
$convertedValue = $currentOutData;
}
 /*If the distance between current pv and the 'closest' is
larger than between current pv and the current row value*/
else if( ($last > 1) && ($currentValue > $lowerInData) &&
($currentValue < $currentInData) ) {

if(isset($getOutData)){
/*Uses Linear Interpolation to calculate the outdata value for
a pv that are between two values in the datasheet*/
$convertedValue = linInterAndExtrapol($lowerOutData,
$currentOutData, $currentValue, $lowerInData,
$currentInData);
} else{
$convertedValue = linInterAndExtrapol($lowerInData,
$currentInData, $currentValue, $lowerOutData,
$currentOutData);
}
}
$lowerInData = $currentInData;
$lowerOutData = $currentOutData;
}

/*Uses Linear Extrapolation to calculate a decent outdata for
the values that are outside of the given interval in the
datasheet*/
if (($last >= 2) && ($convertedValue == $currentValue)){
```

```
if($currentValue < $allInData[1]){
if(isset($getOutData)){
$convertedValue = linInterAndExtrapol($allOutData[1],
$allOutData[2], $currentValue, $allInData[1],
$allInData[2]);
} else{
$convertedValue = linInterAndExtrapol($allInData[1],
$allInData[2], $currentValue, $allOutData[1],
$allOutData[2]);
}
}
else if($currentValue > $allInData[$last]){
if(isset($getOutData)){
$convertedValue = linInterAndExtrapol(
$allOutData[$last-1], $allOutData[$last],
$currentValue, $allInData[$last-1],
$allInData[$last]);
} else{
$convertedValue = linInterAndExtrapol(
$allInData[$last-1], $allInData[$last],
$currentValue, $allOutData[$last-1],
$allOutData[$last]);
}
}
}

return $convertedValue;
}

function reConvPv($sensor){
$db = getDbConnection();

$queryReadInputsForConv = "SELECT pinNr FROM Input WHERE
sensor = :sensor;";
$queryReadPvForConv = "SELECT time, pv FROM DataIn WHERE
pinNr = :currentPin;";
$queryUpdateConvPv = "UPDATE DataIn SET convPv = :convPv WHERE
pinNr = :currentPin AND time = :pvTime;";

$inputsForConv = readDbAssoc($queryReadInputsForConv, ":sensor",
$sensor);
foreach ($inputsForConv as $input) {
$valuesForConv = readDbAssoc($queryReadPvForConv,
":currentPin", $input['pinNr']);
foreach($valuesForConv as $row){
$convPv = convertData($row['pv'], $sensor, TRUE);
```

```
$updateConvPv = $db->prepare($queryUpdateConvPv);
$updateConvPv->bindValue(':currentPin', $input['pinNr']);
$updateConvPv->bindValue(':convPv', $convPv);
$updateConvPv->bindValue(':pvTime', $row['time']);

$updateConvPv->execute() or
die ('Unable to read from db');
}
}
}

function readColNameDb($query){
$db = getDbConnection();
$dbResult = array();

$readDb = $db->prepare($query);
$result = $readDb->execute() or
die('Unable to read column names from db');

$numSetupCols = $result->numColumns();
$i = 0;
while($i < $numSetupCols){
$dbResult[] = $result->columnName($i);
$i++;
}

return $dbResult;
}


function deleteFromDb($rowToDelete, $query){
$db = getDbConnection();

//If input or output, setup will need them first
if(is_int($rowToDelete) && (is_array($rowToDelete) == false)){
$deleteFromSetup = $db->prepare("DELETE FROM Setup WHERE
pinNr = :pinNr");

$deleteFromSetup->bindValue(':pinNr', $rowToDelete);
$deleteFromSetup->execute() or
die('Unable to delete row from setup');

$deleteFromIO = $db->prepare($query);

$deleteFromIO->bindValue(':pinNr', $rowToDelete);
$deleteFromIO->execute() or
die('Unable to delete row from input or output table');
```

```
}elseif(array_key_exists('sensor', $rowToDelete)){
$deleteFromDatasheet = $db->prepare($query);
$deleteFromDatasheet->bindValue(':sensor',
$rowToDelete['sensor']);
if(array_key_exists('inData', $rowToDelete)){
$deleteFromDatasheet->bindValue(':inData',
$rowToDelete['inData']);
}

$deleteFromDatasheet->execute() or
die('Unable to delete from datasheet');
}

}


function addToDb($arrayToAdd, $query){
$db = getDbConnection();

//If input or output, setup will need them first
if(array_key_exists('pinNr', $arrayToAdd[0]) &&
(array_key_exists('mailaddr', $arrayToAdd[0]) == false)){
$queryAddToSetup = "REPLACE INTO Setup (pinNr, io, name,
analog, sampleTime) VALUES (:pinNr, :io, :name, :analog,
:sampleTime);";
$addToSetup = $db->prepare($queryAddToSetup);

foreach($arrayToAdd as $add) {
$addToSetup->bindValue(':pinNr', $add['pinNr']);
$addToSetup->bindValue(':io', $add['io']);
$addToSetup->bindValue(':name', $add['name']);
$addToSetup->bindValue(':analog', $add['analog']);
$addToSetup->bindValue(':sampleTime',
$add['sampleTime']);

$addToSetup->execute() or die('Unable to add to setup');
}

if(array_key_exists('input', $arrayToAdd[0])){//Output
$addToOutput = $db->prepare($query);

foreach($arrayToAdd as $add){
$addToOutput->bindValue(':pinNr', $add['pinNr']);
$addToOutput->bindValue(':input', $add['input']);
$addToOutput->bindValue(':startTime',
$add['startTime']);
$addToOutput->bindValue(':runTime',
```

```
$add['runTime']);
$addToOutput->bindValue(':digiDelay',
$add['digiDelay']);
$addToOutput->bindValue(':sp', $add['sp']);
$addToOutput->bindValue(':kP', $add['kP']);
$addToOutput->bindValue(':kI', $add['kI']);
$addToOutput->bindValue(':kD', $add['kD']);

$addToOutput->execute() or
die('Unable to add to output');
}
}else{ //Input
$addToInput = $db->prepare($query);

foreach($arrayToAdd as $add) {
$addToInput->bindValue(':pinNr', $add['pinNr']);
$addToInput->bindValue(':sensor', $add['sensor']);

$addToInput->execute() or
die('Unable to add to input');
}
}
}//Datasheet
elseif(array_key_exists('unit', $arrayToAdd[0])){
$addToDatasheet = $db->prepare($query);

foreach($arrayToAdd as $add){
$addToDatasheet->bindValue(':sensor', $add['sensor']);
$addToDatasheet->bindValue(':unit', $add['unit']);
$addToDatasheet->bindValue(':inData', $add['inData']);
$addToDatasheet->bindValue(':outData', $add['outData']);

$addToDatasheet->execute() or
die('Unable to add to datasheet');
}
}elseif(array_key_exists('mailaddr', $arrayToAdd[0])){
$addToRepAndAlarm = $db->prepare($query);

foreach($arrayToAdd as $add){
$addToRepAndAlarm->bindValue(':pinNr', $add['pinNr']);
$addToRepAndAlarm->bindValue(':mailaddr',
$add['mailaddr']);
$addToRepAndAlarm->bindValue(':interval',
$add['interval']);
$addToRepAndAlarm->bindValue(':maxLevel',
$add['maxLevel']);
$addToRepAndAlarm->bindValue(':minLevel',
```

```
$add['minLevel']);

$addToRepAndAlarm->execute() or
die('Unable to add to datasheet');
}
}
}


function validate($arrayToCheck){
global $PWM;
global $maxEEPROM;
$regExString = "/^[a-zA-Z0-9 ]+$/";

$errormsg = array();
$errorHtml = '';

foreach($arrayToCheck as $check){

//Input or output
if(array_key_exists('sampleTime', $check)){
if(preg_match($regExString, $check['name']) == FALSE){
$errormsg[] = "Only letters and number allowed,
for " . $check['name'];
}

if($check['sampleTime'] < 1){
$errormsg[] = "Sample time needs to be greater
than 0, for ". $check['name'];
}

if(array_key_exists('sensor', $check)){ //Input
if(($check['analog'] == 1) &&
($check['sensor'] == NULL)){
$errormsg[] = "Analog input needs a datasheet,
for ". $check['name'];
}

if(($check['sampleTime'] % 50) != 0 ){
$errormsg[] = "Sample time needs to be in
format 50 * X, for  ". $check['name'];
}
}elseif(array_key_exists('input', $check)){ //Output
if($check['startTime'] < 0){
$errormsg[] = "Start time needs to be 0 or
greater than, for ". $check['name'];
}
```

```
if($check['sampleTime'] < 0){
$errormsg[] = "Sample time needs to be greater
than 0, for ". $check['name'];
}

if($check['runTime'] < 0){
$errormsg[] = "Sample time needs to be greater
than 0, for ". $check['name'];
}

if($check['analog'] == 0){
if($check['digiDelay'] < 0){
$errormsg[] = "Digital output needs a delay,
for ". $check['name'];
}
}else{
if($check['input'] == NULL){
$errormsg[] = "Analog output needs a input,
for ". $check['name'];
}

if(($check['sp'] * $check['kP'] * $check['kI']
* $check['kD']) == 0){
$errormsg[] = "Analog parameters cannot be 0,
for ". $check['name'];
}

if($check['sp'] > $maxEEPROM){
$errormsg[] = "Analog parameters cannot be
larger than " . $maxEEPROM / 10 .", for ".
$check['name'];
}

if($check['kP'] > $maxEEPROM){
$errormsg[] = "Analog parameters cannot be
larger than " . $maxEEPROM / 10 .", for ".
$check['name'];
}

if($check['kI'] > $maxEEPROM){
$errormsg[] = "Analog parameters cannot be
larger than " . $maxEEPROM / 10 .", for ".
$check['name'];
}

if($check['kD'] > $maxEEPROM){
```

```
$errormsg[] = "Analog parameters cannot be
larger than " . $maxEEPROM / 10 .", for ".
$check['name'];
}

if(in_array($check['pinNr'], $PWM) != TRUE){
$errormsg[] = "Analog output needs to be
pwm, for ". $check['name'];
}
}
}
}elseif(array_key_exists('unit', $check)){//Datasheet
if((preg_match($regExString, $check['sensor']) == FALSE)
|| (preg_match($regExString, $check['unit']) == FALSE)){
$errormsg[] = "Only letters and number allowed!";
}

if($check['inData'] < 0){
$errormsg[] = "In data needs to be greater than 0.";
}

if($check['outData'] < 0){
$errormsg[] = "Out data needs to be greater than 0.";
}
}elseif(array_key_exists('mailaddr', $check)){
if(preg_match($regExString, $check['mailaddr']) == FALSE){
$errormsg[] = "Please supply a valid email!";
}

if(($check['interval'] < 0) || ($check['interval'] > 24)){
$errormsg[] = "Enter a valid hourly interval.";
}

if(($check['minLevel'] > $check['maxLevel']) &&
($check['maxLevel'] != NULL)){
$errormsg[] = "Min level can't be larger than max
level.";
}
}
}

if(!empty($errormsg)){
$errormsg = array_unique($errormsg);

foreach ($errormsg as $message) {
$errorHtml .= $message . '<br>';
}
```

```
}

return $errorHtml;
}

?>
```

Bilaga F

PHP-kod för presentation

analog_data_for_checkboxes.php

```
 <?php

  include_once ("db.php");

$analogInputs = readDbAssoc($queryReadAnalogInputs, NULL, NULL);

foreach($analogInputs as $anInput){
$analogInputsCheckbox[$anInput['name']] = array('label' =>
"PV for: " . $anInput['name']);
}

$analogSP = readDbAssoc($queryReadAnalogSP, NULL, NULL);

foreach($analogSP as $anSP){
$analogInputsCheckbox[$anSP['name']] = array('label' =>
"SP for: " . $anSP['name']);
}

echo json_encode($analogInputsCheckbox);

?>
```

analog_data.php

```
 <?php

  include_once ("db.php");

  $nrOfValues = $_GET['nrOfValues'];
```

```
$analogInputs = readDbAssoc($queryReadAnalogInputs, NULL, NULL);
$lineColor = array(
'pv' => array('#ff0000', '#0000ff', '#008000',
'#000000', '#800080', '#ffff00', '#ffc0cb',
'#ffa500', '#a52a2a', '#00ffa5'),
'sp' => array('#ff4c4c', '#4c4cff', '#4ca64c',
'#666666', '#b266b2', '#ffff66', '#ffd2da',
'#ffc04c', '#c06969', '#66ffc9')
  );
$i = 0;

if($nrOfValues > 0){
$queryAnInData = $queryReadAnalogInDataLimit .
$nrOfValues . ";";
}else{
$queryAnInData = $queryReadAnalogInData;
}

foreach($analogInputs as $anInput){

$analogInData = readDbAssoc($queryAnInData, ":currentPin",
$anInput['pinNr']);
$dataUnit = readDbAssoc($queryReadUnit, ":sensor",
$anInput['sensor']);
$dataUnit = $dataUnit[0]['unit'];

foreach($analogInData as $anData){
$analogData[] = array((int) $anData['time'],
$anData['convPv']);
}

$analogDataResult[$anInput['name']] = array('label' =>
"PV for: " . $anInput['name'], 'data' => $analogData,
'unit' => $dataUnit, 'color' => $lineColor['pv'][$i]);

$i++;
unset($analogData);
}

/*Saves the analog SP so that it can be printed as a line from
it's startTime to the end of it's runTime*/
$analogSP = readDbAssoc($queryReadAnalogSP, NULL, NULL);

foreach($analogSP as $anSP){
$sensor = readDbAssoc($queryReadDatasheetBasedOnInput,
```

```
":pinNr", $anSP['input']);
$sensor = $sensor[0]['sensor'];
$dataUnit = readDbAssoc($queryReadUnit, ":sensor", $sensor);
$dataUnit = $dataUnit[0]['unit'];

$startTime = $anSP['startTime'] * 60;
$runTime = $anSP['runTime'] * 60;

if($nrOfValues > 0){
foreach($analogInputs as $anInput){
if($anInput['pinNr'] == $anSP['input']){
$timePassed = $analogDataResult[
$anInput['name']]['data'][0][0];
if($startTime < $analogDataResult[
$anInput['name']]['data'][$nrOfValues-1][0]){
$startTime = $analogDataResult[
$anInput['name']]['data'][$nrOfValues-1][0];
}
}
}
}else{
$timePassed = minutesPassed($queryReadTimePassed,
":analog", "1") * 60;
}


$convertedSp = convertData(($anSP['sp'] * 4), $sensor, TRUE);
$analogData[] = array($startTime, $convertedSp);

if($runTime == 0){ // Run endlessly
$displayRunTime = (($timePassed - $startTime) * 1.1 +
$startTime); // Adding 10% of time to the sp
}else{
$displayRunTime = $runTime + $startTime;
}

$i = 0;
foreach($analogInputs as $input){
if($anSP['input'] == $input['pinNr']){
$colorIndex = $i;
}
$i++;
}
// Sets an ending point based on start time + run time
$analogData[] = array($displayRunTime, $convertedSp);

$analogDataResult[$anSP['name']] = array('label' =>
```

```
"SP for: " . $anSP['name'], 'data' => $analogData, 'unit' =>
$dataUnit, 'color' => $lineColor['sp'][$colorIndex]);
unset($analogData);
}

echo json_encode($analogDataResult);

?>
```

digital_data.php

```
<?php
include_once("db.php");

// Get the status of all digital in- and outputs
$digitalInputs = readDbAssoc($queryReadDigitalInputs, NULL, NULL);
$digitalOutData = readDbAssoc($queryReadDigitalOutData, NULL, NULL);

$timePassed = minutesPassed($queryReadTimePassed, ":analog", "0");

// For every digital input get the latest pv there is
foreach($digitalInputs as $digInput){
$digitalInData = readDbAssoc($queryReadDigitalInData,
":currentPin", $digInput['pinNr']);

if($digitalInData[0]['pv'] == 0){
$sensorStatus = "off";
}else{
$sensorStatus = "on";
}

unset($outputStatus);

foreach($digitalOutData as $digOutData){
if($digOutData['input'] == $digInput['pinNr']){
$timeLastZeroInput = minutesPassed(
$queryReadTimeLastZeroInput, ":currentPin",
$digInput['pinNr']);
$timeLastActiveInput = minutesPassed(
$queryReadTimeLastActiveInput, ":currentPin",
$digInput['pinNr']);
$timeInactive = $timeLastZeroInput -
$timeLastActiveInput;
$timeSinceStart = $timePassed -
$digOutData['startTime'];

/*Set the status of all digi outputs that has a digi
```

```
delay and an input*/
if(($digOutData['digiDelay'] > 0) &&
($timeInactive >= 0)){
if( ($timeSinceStart >= 0) &&
($digOutData['runTime'] > $timeSinceStart) &&
(($sensorStatus == "on") ||
($digOutData['digiDelay'] > $timeInactive)) ){
$outputStatus = "on";
}else{
$outputStatus = "off";
}
}// Set the pv of all digi outputs that has an input
else{
if( ($timeSinceStart >= 0) &&
($digOutData['runTime'] > $timeSinceStart) &&
($sensorStatus == "on") ){
$outputStatus = "on";
}else{
$outputStatus = "off";
}
}
$digitalStatusResult[] = array('sensorName' =>
$digInput['name'], 'sensorStatus' => $sensorStatus,
'digOutputName' => $digOutData['name'],
'outputStatus'=> $outputStatus);
}
}
// Set the status of all sensors that isn't used by any output
if(!isset($outputStatus)){
$digitalStatusResult[] = array('sensorName' =>
$digInput['name'], 'sensorStatus' => $sensorStatus,
'digOutputName' => "NA", 'outputStatus' => "NA");
}
}

/*Set the pv of all digital outputs that doesn't have an input
and is therefore only timer based*/
foreach($digitalOutData as $digOutData){
if($digOutData['input'] == "50"){
if(($timeSinceStart >= 0) &&
($digOutData['runTime'] >$timeSinceStart)){
$outputStatus = "on";
}else{
$outputStatus = "off";
}
$digitalStatusResult[] = array('sensorName' => "NA",
'sensorStatus' => "NA", 'digOutputName' => $digOutData['name'], 'outputStatus' => $
```

```
}
}


echo json_encode($digitalStatusResult);
?>
```

Bilaga G

Presentation av data

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.
org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>AJAX</title>
<link href="style.css" rel="stylesheet" type="text/css">
<!--[if lte IE 8]><script language="javascript"
type="text/javascript" src="../../excanvas.min.js">
</script><![endif]-->
<script language="javascript" type="text/javascript"
src="flot/jquery.js"></script>
<script language="javascript" type="text/javascript"
src="flot/jquery.flot.js"></script>
<script type="text/javascript">

$(function() {

var data = [];
var result = [];

function fetchDigitalData(){
$.ajax({
url: "digital_data.php",
dataType: "json",
success: onDigitalDataReceived
});
}

        function onDigitalDataReceived(data){
var divCreate = "";
var divOnOff = "";
```

```
          $.each(data, function(index, value){

            if(value.sensorStatus == "NA"){
            divOnOff = value.outputStatus;
            result = value.digOutputName + " is: " +
            value.outputStatus;
            }else{
            if(value.outputStatus == "NA"){
            divOnOff = value.sensorStatus;
            result = value.sensorName + " is: " +
            value.sensorStatus;
          }
          else{
            divOnOff = value.outputStatus;
            result = value.sensorName + " is: " +
            value.sensorStatus + " and " +
            value.digOutputName + " is: " +
            value.outputStatus;
        }
            }

            divCreate = "<div id=\"" + divOnOff + "\">" +
            result  + "</div>" + divCreate;
    });

    $("#container").html(divCreate);
        }


setInterval(fetchDigitalData, 1000);



var options = {
series: {
shadowSize: 0 // Drawing is faster without shadows
},
grid: {
hoverable: true
},
lines: {
show: true
},
points: {
show: true
},
xaxis: {
```

```
tickDecimals: 0,
tickSize: 1
}
};

var data = [];
var choiceContainer = $("#choices");
var nr;

function fetchData() {
$("#nrOfValues").change(function(){
nr = $(this).val();
});
console.log(nr);
$.ajax({
type: "POST",
url: "analog_data.php?nrOfValues=" + nr,
dataType: "json",
success: plotAccordingToChoices
});
}

function fetchDataForCheckboxes() {
$.ajax({
url: "analog_data_for_checkboxes.php",
dataType: "json",
success: createCheckboxes
});
}

function createCheckboxes(data) {
var html = '';

$.each(data, function(key, val) {
html = html + "<br/><input type='checkbox' name='" +
key + "' checked='checked' id='id" + key + "'></input>"
+ "<label for='id" + key + "'>" + val.label +
"</label>";
});

choiceContainer.html(html);
}

choiceContainer.find("input").click(plotAccordingToChoices);

function plotAccordingToChoices(datasets) {
```

```
var data = [];
choiceContainer.find("input:checked").each(function () {
var key = $(this).attr("name");
if (key && datasets[key]) {
data.push(datasets[key]);
}
});

if (data.length > 0) {
plot.setData(data);
            plot.setupGrid();
            plot.draw()
}
}


var plot = $.plot("#placeholder", data, options);

fetchDataForCheckboxes();

setInterval(fetchData, 1000);
fetchData();

//Tooltip
$("<div id='tooltip'></div>").css({
position: "absolute",
display: "none",
border: "1px solid #fdd",
padding: "2px",
"background-color": "#fee",
opacity: 0.80
}).appendTo("body");

$("#placeholder").bind("plothover", function (event, pos, item) {
if (item) {
var x = item.datapoint[0].toFixed(2),
y = item.datapoint[1].toFixed(2);

$("#tooltip").html(item.series.label + " vid t = " + x
+ " s & " + y + " " + item.series.unit)
.css({top: item.pageY+5, left: item.pageX+5})
.fadeIn(200);
} else {
$("#tooltip").hide();
}
})
});
```

```
</script>
</head>
<body>

<div id="content">

<div class="demo-container">
<div id="placeholder" class="demo-placeholder"></div>
<p id="choices" style="float:right; width:135px;"></p>
</div>

<div id="container"></div>
<td><br><br>
<input id = "nrOfValues" type = "text" style="width:200px";
placeholder="nr of values to display" value = "">
</td>
</div>

</body>
</html>
```

Bilaga H

PHP-kod för konfigurationssidan

```
<html>
<body>
<table>
<form method="post" action="">
<?php


include_once("db.php");

$setInputs = array();
$setOutputs = array();
$setDatasheets = readDbAssoc($queryReadDatasheetForOutput, NULL, NULL);

if(isset($_POST['pinNr'])){ //All requires sampletime & pinnr

//Get post from datasheet
$pinNr = $_POST['pinNr']; // FÖr att ta bort tex ''
$name = $_POST['name']; //Ta bort allt som inte är a-z
$analog = $_POST['analog'];
$sampleTime = $_POST['sampleTime'];
$setAsInput = $_POST['io']; //Check values with index 0-19
$input = $_POST['input'];
$startTime = $_POST['startTime'];
$runTime = $_POST['runTime'];
$digiDelay = $_POST['digiDelay'];
$sp = $_POST['sp'];
$kP = $_POST['kP'];
$kD = $_POST['kD'];
$kI = $_POST['kI'];
$sensor = $_POST['sensor'];
$deleteRowPost = $_POST['deleteRow'];
```

```
$numSetAsInputs = count($setAsInput);

$j = 0;
for($i = 0; $i < $numPins; $i++){
//Check if input
if(($pinNr[$setAsInput[$j]] == $pinNr[$i]) && ($j < $numSetAsInputs)){
if(isset($_POST['Delete']) && in_array($i, $deleteRowPost)){
deleteFromDb((int) $pinNr[$i], $queryDeleteFromInput);
}else{
if(isset($analog) && in_array($i, $analog)){ //Check if analog
$setAnalog = 1;
}else{
$setAnalog = 0;
$sensor[$i] = NULL;
}
$setInputs[] = array("pinNr" => (int) $pinNr[$i], "io" => 1, "name"
=> $name[$i], "analog" => (int) $setAnalog,
"sampleTime" => (int) $sampleTime[$i], "sensor" => $sensor[$i]);
}
$j++;
}
elseif($pinNr[$i] != NULL){ //Output
if(isset($_POST['Delete']) && in_array($i, $deleteRowPost)){
deleteFromDb((int) $pinNr[$i], $queryDeleteFromOutput);
}else{
if(isset($analog) && in_array($i, $analog)){ //Check if analog
$sensorToCheck = readDbAssoc($queryReadDatasheetBasedOnInput,
":pinNr", $input[$i]);
$convertSp = convertData($sp[$i], $sensorToCheck[0]['sensor'], NULL);

$setSp = round((int) ($convertSp / 4));
if($setSp > 255){// round(1023 / 4) = 256, 9-bits!
$setSp = 255;
}

$setAnalog = 1;
$setKP = ((int) $kP[$i]) * 10;
$setKD = ((int) $kD[$i]) * 10;
$setKI = ((int) $kI[$i]) * 10;
$setDigiDelay = 0;
}else{
if($input[$i] == NULL){
$input[$i] = 50; //No input, needed for MCU
}
$setAnalog = 0;
$setSp = 0;
$setKP = 0;
```

```
$setKD = 0;
$setKI = 0;
$setDigiDelay = (int) $digiDelay[$i];
}

$setOutputs[] = array("pinNr" => (int) $pinNr[$i], "io" => 0,
"name" => $name[$i],
"analog" => (int) $setAnalog, "sampleTime" => (int) $sampleTime[$i],
"digiDelay" => $setDigiDelay, "sp" => $setSp, "kP" => $setKP,
"kI" => $setKI,
"kD" => $setKD, "input" => (int) $input[$i], "startTime" =>
(int) $startTime[$i], "runTime" => (int) $runTime[$i]);

}
}
}

$validateInputs = validate($setInputs);
$validateOutputs = validate($setOutputs);
if(isset($setInputs[0]['pinNr'])){
if((strlen($validateInputs) == 0) && isset($_POST['Change'])){
addToDb($setInputs, $queryAddToInput);
}else{
echo $validateInputs;
}
}


$inputDoesExist = 0;
if(isset($setOutputs[0]['pinNr'])){
if((strlen($validateOutputs) == 0) && isset($_POST['Change'])){
//Check if output has input that does exist
foreach($setOutputs as $output){
foreach($setInputs as $input){
if(($output['input'] == $input['pinNr']) || ($output['input'] == 50)){
$inputDoesExist++;
}
}
}

if($inputDoesExist == count($setOutputs)){
addToDb($setOutputs, $queryAddToOutput);
}else{
echo "Output needs input that does exist!";
}
}else{
echo $validateOutputs;
```

```
}
}
}

if(isset($setOutputs[0]['pinNr']) || isset($setInputs[0]['pinNr'])){
restartMCU();
}


$setIO = readDbAssoc($queryReadSetupDb, NULL, NULL);

$nameDatasheetCols = readColNameDb($queryReadDatasheetForOutput);
$nameSetupCols = readColNameDb($queryReadSetupDb);


//Print column names
echo "<tr>";
foreach($nameSetupCols as $name){
echo "<th scope=\"col\">" . $name . "</th>";
}
echo "</tr>";

$numSetIO = count($setIO);

//Print form data
for ($i = 0; $i < $numPins; $i++){
echo "<tr>";
foreach($nameSetupCols as $name){
if(($name == "pinNr") || ($name == "input")){ //Drop down lists
echo "<td><select name=\"" . $name . "[]\">";
echo "<option value =\"\">";
foreach ($IOPins as $key => $pin) {
echo "<option ";
if(($setIO[$i][$name] == $key) && ($i < $numSetIO) && ($pin != NULL)){
echo "selected ";
}
echo "value=\"" . $key . "\">" . $pin;
if((in_array($key, $PWM)) && ($name == "pinNr")){
echo " (PWM)";
}
}
echo "</option>";
echo "</td>";
}//Text fields
elseif(($name == "name") || ($name == "sampleTime") ||
($name == "startTime") || ($name == "runTime") ||
($name == "digiDelay" ) || ($name == "sp") || ($name == "kP") ||
```

```
($name == "kD") || ($name == "kI")){
echo "<td><input type =\"text\" style=\"width:80px;\" name=\"".
$name . "[]\" value=\"";
if(($name == "sp") && ($setIO[$i][$name] != 0)){
$sensorToCheck = readDbAssoc($queryReadDatasheetBasedOnInput,
":pinNr", $setIO[$i]['input']);
$convertSp = convertData(($setIO[$i][$name] * 4),
$sensorToCheck[0]['sensor'], TRUE);
echo round($convertSp, 1);
}elseif((($name == "kP") || ($name == "kD") || ($name == "kI")) &&
($setIO[$i]['analog'] == "1") && ($setIO[$i]['sensor'] == NULL)){
echo ($setIO[$i][$name] / 10);
}elseif(($name == "digiDelay") && ($setIO[$i]['analog'] != "1")){
echo $setIO[$i][$name];
}elseif(($name == "name") || ($name == "sampleTime") ||
($name == "startTime") || ($name == "runTime")){
echo $setIO[$i][$name];
}
echo "\"></td>";
} //Checkboxes
elseif(($name == "analog") || ($name == "io")){
echo "<td><input type=\"checkbox\" name=\"";
echo $name;
echo "[]\" value=\"";
echo $i;
echo "\"";
if(($setIO[$i][$name] == "1") && ($i < $numSetIO)){
echo " checked>";
}
echo "</td>";
}
elseif($name == "sensor"){//Dropdown list
echo "<td><select name=\"" . $name . "[]\">";
echo "<option value =\"\">";
foreach($setDatasheets as $index => $label){
echo "<option ";
if(($setIO[$i][$name] == $label['sensor']) && ($i < $numSetIO)){
echo "selected ";
}
echo "value=\"" . $label['sensor'] . "\">" . $label['sensor'];
}
echo "</option>";
echo "</td>";
}
}
echo "<td><input type=\"checkbox\" name=\"deleteRow[]\" value=\"";
echo $i;
```

```
echo "\">";
echo "</td>";
}
?>
<tr>
<td><input type="submit" name="Change" value="Change"></td>
<td><input type="submit" name="Delete" value="Delete"></td>
</tr>
<tr>
<td>
<a href="dsconf.php">Edit datasheets</a>
</td>
</tr>
</form>
</table>
</body>
</html>
```

Bilaga I

PHP-kod för mejlfunktion

```
<html>
<body>
<table>
<form method="post" action="">
<?php

include_once("db.php");


if(isset($_POST['mailaddr'])){
$mailAddrPost = $_POST['mailaddr'];
$intervalPost = $_POST['interval'];
$namePost = $_POST['name'];
$maxlevelPost = $_POST['maxLevel'];
$minlevelPost = $_POST['minLevel'];

$numPost = count($_POST['name']);

for($i = 0; $i < $numPost; $i++) {
$inPin = readDbAssoc($queryReadSetupForAlarm, ":name",
$namePost[$i]);
$sensorToCheck = readDbAssoc($queryReadDatasheetBasedOnInput,
":pinNr", $inPin[0]['pinNr']);
if($sensorToCheck[0]['sensor'] != NULL){
if($maxlevelPost[$i] != NULL){
$convertMax = convertData($maxlevelPost[$i],
$sensorToCheck[0]['sensor'], NULL);
$convertMax = round($convertMax);
}else{
$convertMax = NULL;
}
if($minlevelPost[$i] != NULL){
```

```
$convertMin = (float) convertData($minlevelPost[$i],
$sensorToCheck[0]['sensor'], NULL);
$convertMin = round($convertMin);
}else{
$convertMin = NULL;
}
}else{
if($maxlevelPost[$i]  != NULL){
$convertMax = (float) $maxlevelPost[$i];
}
if($minlevelPost[$i]  != NULL){
$convertMin = (float) $minlevelPost[$i];
}
}
$addAlarm[] = array('pinNr' => $inPin[0]['pinNr'],
'mailaddr' => $mailAddrPost, 'interval' => (int)
$intervalPost, 'maxLevel' => $convertMax, 'minLevel' =>
$convertMin);
}

$validateAlarm = validate($addAlarm);

if(strlen($validateAlarm) == 0){
addToDb($addAlarm, $queryAddToRepAndAlarm);
}else{
echo $validateAlarm;
}
}


$reportAndAlarm = array();

$allRepAndAlarmCols = readColNameDb($queryReadRepAndAlarm);
$reportAndAlarm = readDbAssoc($queryReadRepAndAlarm, NULL, NULL);


$nrOfRows = count($reportAndAlarm);

//Print column names
echo "<tr>";
foreach($allRepAndAlarmCols as $name){
echo "<th scope=\"col\">" . $name . "</th>";
}
echo "</tr>";

$firstRun = true;
// Create table for existing pins, their max- and min values and
```

```
// the mailaddr + interval displayed only once
for ($i=0; $i < $nrOfRows; $i++) {
echo "<tr>";

foreach ($reportAndAlarm[$i] as $colName => $colValue) {
if($firstRun == true){
echo "<td><input type =\"text\" style=\"width:80px;\" name=\"";
echo $colName;
if(($colName == 'maxLevel') || ($colName == 'minLevel') ||
($colName == 'name')){
echo "[]";
}
echo "\" value=\""; //@todo gray out name
if(($colName == 'maxLevel') || ($colName == 'minLevel')){
$inPin = readDbAssoc($queryReadSetupForAlarm, ":name",
$reportAndAlarm[$i]['name']);
if($colValue != NULL){
$sensorToCheck = readDbAssoc($queryReadDatasheetBasedOnInput,
":pinNr", $inPin[0]['pinNr']);
$convertValue = convertData($colValue, $sensorToCheck[0]['sensor'],
TRUE);
echo $convertValue . "\"";
}else{
echo $colValue . "\"";
}

}else{
echo $colValue . "\" ";
if($colName == 'name'){
echo "readonly";
}
echo "=\"true\"";
}
echo "></td>";
}else{
if(($firstRun == false) && (($colName == 'mailaddr') ||
($colName == 'interval'))){
echo "<td></td>";
}else{
echo "<td><input type =\"text\" style=\"width:80px;\" name=\"";
echo $colName;
echo "[]\" value=\"";
if(($colName == 'maxLevel') || ($colName == 'minLevel')){
$inPin = readDbAssoc($queryReadSetupForAlarm, ":name",
$reportAndAlarm[$i]['name']);
if($colValue != NULL){
$sensorToCheck = readDbAssoc($queryReadDatasheetBasedOnInput,
```

```
":pinNr", $inPin[0]['pinNr']);
$convertValue = convertData($colValue, $sensorToCheck[0]['sensor'],
TRUE);
echo round($convertValue, 1) . "\"";
}else{
echo $colValue . "\"";
}

}else{
echo $colValue;
echo "\"";
}
}

if($colName == 'name'){
echo " readonly=\"true\"";
}
echo "</td>";
}

}

echo "</tr>";
$firstRun = false;

}

?>

<tr>
<td><input type="submit" name="Apply" value="Apply"></td>
</tr>
</form>
</table>
</body>
</html>
```

Bilaga J

PHP-kod för datablad

```
<html>
<body>
<table>
<form method="post" action="">
<?php
include_once("db.php");

$outDataPost = array();
$inDataPost = array();
$dbAddData = array();

if(isset($_POST['Delete']) && isset($_POST['sensorDef'])){
$sensorDefPost = $_POST['sensorDef'];
if($_POST['deleteRow']){
$inDataPost = $_POST['inData'];
$deleteRowPost = $_POST['deleteRow'];
foreach($deleteRowPost as $index) {
$deleteDatasheetPost = array('sensor' => $_POST['sensor'],
'inData' => $inDataPost[$index]);
deleteFromDb($deleteDatasheetPost, $queryDeleteDatasheetPost);
}
reConvPv($_POST['sensor']);
}else{
$deleteDatasheet['sensor'] = $sensorDefPost;
deleteFromDb($deleteDatasheet, $queryDeleteDatasheet);
reConvPv($sensorDefPost);
}

}elseif(isset($_POST['sensorDef']) && isset($_POST['Change'])){
$sensorDefPost = $_POST['sensorDef'];
$unitPost = $_POST['unit'];
$outDataPost = $_POST['outData'];
```

```
$inDataPost = $_POST['inData'];
$rowsAddPost = (int) $_POST['addRows'];
if((isset($_POST['sensor']) && ($rowsAddPost >= 0)) ||
(($sensorDefPost == "add") && ($rowsAddPost >= 0))){
$sensorPost = $_POST['sensor'];
$numRows = count($outDataPost);

for($i = 0; $i < $numRows; $i++) {
if($inDataPost[$i] != NULL){
$addDatasheets[] = array('sensor' => $sensorPost,
'unit' => $unitPost,
'inData' => (int) $inDataPost[$i], 'outData' =>
(float) $outDataPost[$i]);
}
}
if(isset($addDatasheets[0]['sensor'])){
$validateDatasheets = validate($addDatasheets);

if((strlen($validateDatasheets) == 0) && isset($_POST['Change'])){
addToDb($addDatasheets, $queryAddToDatasheet);
reConvPv($addDatasheets[0]['sensor']);
}else{
echo $validateDatasheets;
}
}
}
}else{
$sensorDefPost = NULL;
$rowsAddPost = 0;
}


$allSetDatasheets = readDbAssoc($queryReadAllSetDatasheets, NULL,
NULL);
$nameDatasheetCols = readColNameDb($queryReadAllSetDatasheets);
$setDatasheetsTypes = readDbAssoc($queryReadDatasheetForOutput,
NULL, NULL);


//Print column names
echo "<tr>";
foreach($nameDatasheetCols as $name){
echo "<th scope=\"col\">" . $name . "</th>";
}
echo "<th scope=\"col\">Delete</th>";
echo "</tr>";
```

```
$numSetDatasheets = count($allSetDatasheets);


$firstRun = true;

$j = 0;
/*Create table for existing datasheets, sensor and unit displayed only
once*/
for($i = 0; $i < $numSetDatasheets; $i++){
if($allSetDatasheets[$i]['sensor'] == $sensorDefPost){
echo "<tr>";
foreach($allSetDatasheets[$i] as $key => $sheet){
if((($key == 'unit') || ($key == 'sensor')) && ($firstRun == true)){
echo "<td><input type =\"text\" style=\"width:80px;\" name=\"" . $key .
"\" value=\"";
echo $sheet;
echo "\"></td>";
}elseif(($key == 'outData') || ($key == 'inData')){
if(($firstRun == false) && ($key == 'inData')){
echo "<td></td>";
echo "<td></td>";

}
echo "<td><input type =\"text\" style=\"width:80px;\" name=\"". $key .
"[]\" value=\"";
echo $sheet;
echo "\"></td>";
}
}
echo "<td><input type=\"checkbox\" name=\"deleteRow[]\" value=\"";
echo $j;
echo "\">";
echo "</td>";
$firstRun = false;
echo "</tr>";
$j++;
}
}

/*Create for new datasheets or added rows, sensor and unit displayed
only once or not if added rows only*/
if(($rowsAddPost > 0) && ($sensorDefPost != NULL)){
$firstRun = true;
$j = 0;
while($j < $rowsAddPost){
echo "<tr>";
foreach($nameDatasheetCols as $name){
```

```
if($sensorDefPost == "add"){
if(($name == 'sensor') && ($firstRun == true)){
echo "<td><input type =\"text\" style=\"width:80px;\" name=\"" .
$name . "\" value=\"\"></td>";
}elseif(($name == 'unit') && ($firstRun == true)){
echo "<td><input type =\"text\" style=\"width:80px;\" name=\"" .
$name . "\" value=\"\"></td>";
}
}elseif(($sensorDefPost != "add") && ($name == 'sensor') &&
($firstRun == true)){
echo "<td></td><td></td>";
}
if(($name == 'outData') || ($name == 'inData')){
if(($firstRun == false) && ($name == 'inData')){
echo "<td></td>";
echo "<td></td>";

}
echo "<td><input type =\"text\" style=\"width:80px;\" name=\"". $name .
"[]\" value=\"\"></td>";
}
}
if($firstRun == true){
$firstRun = false;
}
echo "</tr>";
$j++;
}
$firstRun = false;
}

echo "<tr><td>";
echo "<select name=\"sensorDef\">";
echo "<option value =\"\">";
foreach($setDatasheetsTypes as $index => $row){
echo "<option ";
if($row['sensor'] == $sensorDefPost){
echo "selected ";
}
echo "value=\"" . $row['sensor'] . "\">" . $row['sensor'];
}
echo "<option value =\"add\"";
if("add" == $sensorDefPost){
echo "selected ";
}
echo ">Add new";
echo "</option></td>";
```

```
?>
<td><input type="text" style="width:100px;" name="addRows" value=""
placeholder="# of rows to add"></td>
<td><input type="submit" name="Change" value="Change"></td>
<?php  //Delete sensor
if(($sensorDefPost != NULL) && ($sensorDefPost != "add")){
echo "<td><input type=\"submit\" name=\"Delete\" value=\"Delete\">
</td>";
}
?>
</tr>
</form>
</table>
</body>
</html>
```

Bilaga K

Triggers

```
CREATE TRIGGER InsertToReportAndAlarm
AFTER INSERT ON Input

BEGIN
INSERT INTO ReportAndAlarm (pinNr) VALUES (NEW.pinNr);
END;

CREATE TRIGGER DeleteFromReportAndAlarm
BEFORE DELETE ON Setup

BEGIN
DELETE FROM ReportAndAlarm WHERE pinNr = OLD.pinNr;
END;


CREATE TRIGGER PolationOfInData
AFTER INSERT ON DataIn
BEGIN

INSERT INTO _Variables (pinNr)
SELECT pinNr
FROM Setup
WHERE analog = 1
AND pinNr = NEW.pinNr;

-- Get corresponding sensor
UPDATE _Variables
SET
currentSensor = (
SELECT sensor
FROM Input
WHERE pinNr = NEW.pinNr
```

```
)
WHERE pinNr = NEW.pinNr;

-- Get lowest inData from the corresponding sensor
UPDATE _Variables
SET
minInData = (
SELECT MIN(inData)
FROM Datasheet
WHERE sensor = currentSensor
),

-- Get highest inData from the corresponding sensor
maxInData = (
SELECT MAX(inData)
FROM Datasheet
WHERE sensor = currentSensor
)
WHERE pinNr = NEW.pinNr;

UPDATE _Variables
SET
secMinInData = (
SELECT MIN(inData)
FROM (
SELECT inData
FROM Datasheet
WHERE sensor = currentSensor
AND inData > minInData
)
),

minOutData = (
SELECT outData
FROM Datasheet
WHERE sensor = currentSensor
AND inData = minInData
),

closestLowInData = (
SELECT MAX(inData)
FROM (
SELECT inData
FROM Datasheet
WHERE sensor = currentSensor
AND inData <= NEW.pv
)
```

```
),

closestHiInData = (
SELECT MIN(inData)
FROM (
SELECT inData
FROM Datasheet
WHERE sensor = currentSensor
AND inData > NEW.pv
)
)

WHERE pinNr = NEW.pinNr;

UPDATE _Variables
SET
secMinOutData = (
SELECT outData
FROM Datasheet
WHERE sensor = currentSensor
AND inData = secMinInData
),

secMaxInData = (
SELECT MAX(inData)
FROM (
SELECT inData
FROM Datasheet
WHERE sensor = currentSensor
AND inData < maxInData
)
),

maxOutData = (
SELECT outData
FROM Datasheet
WHERE sensor = currentSensor
AND inData = maxInData
)

WHERE pinNr = NEW.pinNr;

UPDATE _Variables
SET
secMaxOutData = (
SELECT MAX(outData)
FROM (
```

```
SELECT outData
FROM Datasheet
WHERE sensor = currentSensor
AND inData = secMaxInData
)
),

closestLowOutData = (
SELECT outData
FROM Datasheet
WHERE sensor = currentSensor
AND inData = closestLowInData
),

closestHiOutData = (
SELECT outData
FROM Datasheet
WHERE sensor = currentSensor
AND inData = closestHiInData
)
WHERE pinNr = NEW.pinNr;


UPDATE DataIn
SET convPv = (
SELECT CASE
WHEN ( NEW.pv <= minInData )
THEN ( minOutData + (secMinOutData - minOutData)
* (NEW.pv - minInData) / (secMinInData - minInData) )
WHEN ( NEW.pv >= maxInData )
THEN ( secMaxOutData + (maxOutData - secMaxOutData)
* (NEW.pv - secMaxInData) / (maxInData - secMaxInData) )
ELSE
( closestLowOutData + (closestHiOutData - closestLowOutData)
* (NEW.pv - closestLowInData) /
(closestHiInData - closestLowInData) )
END
FROM _Variables
)
WHERE DataIn.pinNr = NEW.pinNr
AND DataIn.time = NEW.time;

-- PV 50 for pinNr 18 (A0) should be converted to 15.15 with
-- the current datasheet

DELETE FROM _Variables WHERE pinNr = NEW.pinNr;
END; -- End of trigger
```