Datainsamling med mikrokontroller

På väg mot Internet of Things

Information gathering with microcontrollers
Towards Internet of Things

Johansson, Per
Reinholdsson, Christofer

Fakulteten för hälsa, natur- och teknikvetenskap

Datavetenskap

C-uppsats 15 hp

Handledare: Kerstin Andersson

Examinator: Donald F. Ross

Oppositionsdatum: 20140602

Löpnummer

Datainsamling med mikrokontroller - på väg mot Internet of Things

Information gathering with microcontrollers - towards Internet of Things

Johansson, Per
Reinholdsson, Christofer

Denna uppsats är skriven som en del av det arbete som krävs för att erhålla en högskoleingenjörsexamen i datavetenskap. Allt material i denna rapport, vilket inte är vårt eget, har blivit tydligt identifierat och inget material är inkluderat som tidigare använts för erhållande av annan examen.

Johansson, Per

Reinholdsson, Christofer

Godkänd, Date of defense

Opponent: Joakim Wahman, Erik Larsson

Handledare: Kerstin Andersson

Examinator: Donald F. Ross

Sammanfattning

Internet of Things är ett begrepp för att beskriva ett samhälle där alla objekt, såväl som subjekt, är uppkopplade och unikt identifierbara på Internet. Mikrokontroller, som kan ses som kompakta datorer, kan komma att spela en viktig roll i detta samhälle då de, trots sina förhållandevis små dimensioner, erbjuder en hel del beträffande prestanda.

Målet med denna uppsats är att utvärdera tre olika mikrokontroller; Arduino Nano, BeagleBone Black samt Netduino. Dels för att identifiera deras styrkor respektive svagheter, dels för att enklare kunna bestämma vilken mikrokontroller som skall väljas till vilken uppgift.

Utvärderingen har utveckling av programvara till mikrokontrollerna som huvudfokus. Mikrokontrollernas plattform, utvecklingsverktyg samt programmeringsspråk är de grundläggande utvärderingspunkterna. Popularitet för mikrokontrollerna har utvärderats, då den kan spela en avgörande roll för hur mycket hjälp och stöd som finns att tillgå, både från tillverkaren och från entusiaster.

Denna studie har visat att Arduino Nano, trots att det är mikrokontrollern med den populäraste plattformen av de tre, är den som lider av flest begränsningar. Detta gäller bland annat felsökningsmöjligheter, möjlighet att använda flera sensorer och kommunikation med flera enheter seriellt.

Abstract

Internet of Things is a concept used to describe a society in which all objects, as well as subjects, are connected and uniquely identifiable on the Internet. Microcontrollers, which can be viewed as compact computers, may play an important role in this society since they, despite their relatively small size, offer a lot regarding performance.

The goal with this thesis is to evaluate three different microcontrollers; Arduino Nano, BeagleBone Black and Netduino. Partly to identify their strengths and weaknesses, partly to more easily determine which microcontroller to be selected for which task.

The main focus of the evaluation is development of software to microcontrollers. The microcontrollers platform, development tools and programming languages are the main topics. The popularity of the microcontrollers has also been evaluated, as the popularity can play an important role in the available help and support, both from the manufacturer and from enthusiasts.

This study has shown that Arduino Nano, despite beeing built upon the most popular platform, suffers from most limitations. These limitations include debugging, the possibility to use multiple sensors and serial communication with several devices.

Vi vill tacka vår handledare på Karlstads Universitet, Kerstin Andersson, för korrekturläsning och värdefulla synpunkter på uppsatsen. Vi vill även tacka Mats Persson som varit vår handledare på Sogeti och som givit oss råd och tips kring arbetet. Slutligen vill vi tacka Sogeti för att vi fick möjligheten att göra detta examensarbete hos dem.

Innehåll

Figurer

Tabeller

Kapitel 1

Introduktion

Detta arbete syftar till att utvärdera tre stycken mikrokontroller; Arduino Nano, Netduino samt BeagleBone Black. De tre huvudparametrar som valts att utvärdera ligger främst i en utvecklares perspektiv.

Den första parametern är plattform. Plattformen syftar på mikrokontrollerns hårdvarufamilj och den grundläggande mjukvaran, som eventuellt delas mellan flera mikrokontroller. Här undersöks vilket stöd plattformen har från producenten, men även dokumentation över exempelvis portar på den specifika mikrokontrollern och möjligheten det finns att uppdatera programvaran på den i drift. Utöver detta har utvärdering även gjorts beträffande de första- och tredjepartsbibliotek som finns till plattformen.

Den andra parametern är utvecklingsverktyg. Här undersöks vilka utvecklingsverktyg tillverkaren av mikrokontrollern har utvecklat eller rekommenderar, men även vilket stöd det finns för att utveckla mjukvara till den i Visual Studio.

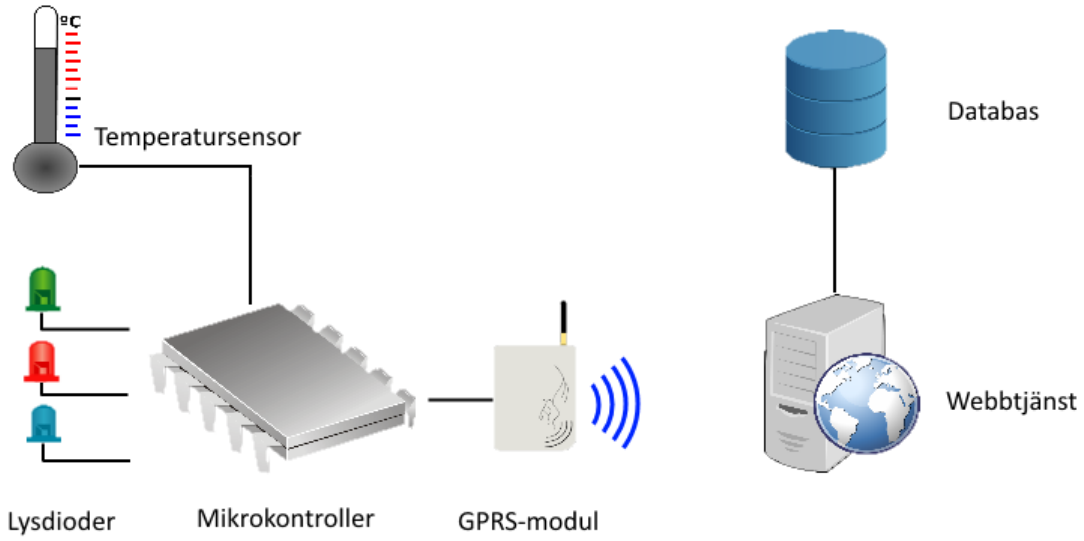
Den tredje parametern är programmeringsspråk. Här undersöks i vilka programmeringsspråk man kan utveckla mjukvara till mikrokontrollern.

Utvärderingen skall, tillsammans med utvecklingen av ett praktiskt jämförelsefall, försöka ge svar på svårigheten att utveckla mjukvara till de tre mikrokontrollerna.

Jämförelsefallet, illustrerat i Figur 1.1, går ut på att varje mikrokontroller skall kunna

läsa av en temperatur. Om temperaturen är inom ett givet intervall skall mikrokontrollern tända en grön lysdiod, annars en röd. Det avlästa värdet skall sedan, med en modul som kommunicerar via General Packet Radio Service (GPRS), skickas till en webbtjänst. Webbtjänsten skall i sin tur spara mätvärdet i en databas. Utöver programvaran på varje mikrokontroller skall även nämnda webbtjänst och databas designas och implementeras.

Detta arbete görs för att uppdragsgivaren, Sogeti, lättare skall kunna besluta sig för vilken mikrokontroller de skall använda sig av vid olika tillämpningar. Denna studie ska ge Sogeti underlag för att kunna presentera den bäst anpassade lösningen för en kund.



Figur 1.1: Översikt över jämförelsefallet.

Disposition

Kapitel 2 ger läsaren en överblick av syftet med studien. Kapitlet introducerar de mikrokontroller som skall jämföras och ger läsaren en övergripande förståelse för de verktyg och uttryck som kommer att användas i studien.

Kapitel 3 kommer att presentera designen av webbtjänsten, samt de tekniker som kommer att användas. Förutom detta kommer resultatet av utvärderingen för varje mikrokontroller med avseende på plattform, programmeringsspråk och utvecklingsverktyg att redovisas.

Kapitel 4 kommer att presentera resultatet av studien. De tre mikrokontrollerna kommer att ställas emot varandra och deras för- och nackdelar kommer att diskuteras med avseende på plattform, utvecklingsmiljö, programmeringsspråk och tiden det tog att implementera jämförelsefallet.

Kapitel 5 kommer att sammanfatta projektet. Utvecklingen av det praktiska jämförelsefallet kommer att diskuteras. Eventuella framtida vidareutvecklingar och andra användningsområden kommer att tas upp.

Kapitel 2

Bakgrund

2.1 Introduktion

Denna studie skall utvärdera tre stycken mikrokontroller. De tre mikrokontrollerna är Arduino Nano, BeagleBone Black samt Netduino. Resultatet av denna studie är av intresse för uppdragsgivaren, Sogeti, om de i framtiden skall erbjuda lösningar baserade på mikrokontroller. Introduktionen till avsnitt 2.2 beskriver allmänt hur man kan jämföra implementationer, medan avsnitt 2.2.1 beskriver det praktiska jämförelsefallet om datainsamling med hjälp av mikrokontroller som har använts i studien. I avsnitt 2.3 beskrivs varför mätdata är av intresse och vad den kan användas till. En mer utförlig beskrivning av mikrokontrollerna och deras prestanda följer i avsnitt 2.4. I avsnitt 2.5 förklaras fenomenet Internet of Things och vilka krav det ställer, både vad gäller teknik och vilka påföljder det kan leda till gällande integritet. I avsnitt 2.6 beskrivs liknande system som de utvärderade.

2.2 Implementationsjämförelse

En av de största anledningarna med denna studie är att utvärdera hur effektiva de olika plattformerna är att utveckla på. Kostnad är alltid en viktig faktor vid anskaffande av

utrustning och it-system. Detta gäller både hårdvara och mjukvara. Givet en fast investeringssumma ställer billigare hårdvara högre krav på utvecklare genom att de i regel har mindre resurser jämfört med den genomsnittliga persondatorn, som enligt enligt Steams hårdvaruenkät januari 2014 hade ca 2,3 GHz i klockfrekvens på två kärnor, 8 GB arbetsminne samt 250 GB hårddisk [1]. Dyrare hårdvara leder till att en lägre pengasumma finns att lägga på utveckling av programvaran. En symbios av precis tillräcklig hårdvara i kombination med effektiv mjukvara är därför önskvärd.

Fokus i denna studie kommer att ligga på kodeffektivitet, utvecklingsverktyg, svårighetsgrad samt tidsåtgång för utveckling av programvaran. Utöver detta kommer även hjälpresurser i form av dokumentation, manualer, guider samt diskussiongrupper och liknande där utvecklare kan söka mer information om de olika systemen att utvärderas.

Lagringsutrymmet på en mikrokontroller kan vara kraftigt begränsat, exempelvis har Arduino Nano endast 32 kB. Förutom detta har de även mindre mängd arbetsminne och lägre klockfrekvens, jämfört med en persondator. Detta ställer högre krav på utvecklaren att utveckla program som är effektiva. Detta för att alla onödiga instruktioner tar plats av kodminnet och klockcykler som istället kan användas för att utföra nyttiga uppgifter. I slutändan skulle det kunna betyda att funktionalitet måste prioriteras bort för att det inte finns resurser. Det är dock inte endast utvecklarens kod som tar utrymme. En del av utrymmet är reserverat för instruktioner som krävs för att systemet skall kunna köras överhuvudtaget, till exempel bootloadern som har till uppgift att ladda kod från minnet till ramminnet så att mikrokontrollern kan exekvera [2]. I andra fall kan minne även vara reserverat åt olika ramverk, som .NET Micro Framework i Netduinos fall [3].

Utvecklingsverktygen är viktiga hjälpmedel för utvecklaren, och ett bra verktyg underlättar och effektiviserar utvecklingen på plattformen och kan därmed bidra med både effektivitet vad gäller kodprestanda och tidsåtgång. Dåliga utvecklingsverktyg gör istället plattformen komplicerad och tidskrävande att utveckla för, vilket gör kostnaden högre. Ett bra utvecklingsverktyg kan till exempel snabbt ge feedback om syntaxfel, felstavning

av variabler och så kallad ”kodkomplettering”, som innebär att utvecklaren får förslag på instruktioner, variabelnamn och datatyper som liknar det han eller hon håller på att skriva [4] [5]. Många utvecklingsmiljöer underlättar även byggprocessen av programvaran samt innehåller verktyg som underlättar vid felsökning.

Svårighetsgraden för de olika plattformarna kan bedömas genom att undersöka hur mycket hjälpresurser det finns till de olika plattformarna, till exempel dokumentation och guider, men även genom att undersöka hur stort intresset är för plattformen på diskussionsforum och liknande resurser dit man kan vända sig för att få hjälp. Svårighetsgraden kan även mätas i hur mycket tid det går åt för att utveckla programvara till plattformen och hur många tredjepartsbibliotek det finns att tillgå.

2.2.1 En praktisk jämförelse

Idén är att utveckla samma program till de olika mikrokontrollerna för att kunna utvärdera dem på ett rättvist sätt. Programmet ska läsa av ett mätvärde från en temperaturgivare. Om mätvärdet är inom ett fördefinierat intervall skall en grön lampa lysa. Om värdet är utanför intervallet skall den gröna lampan släckas och en röd lampa tändas. Det inlästa värdet skickas, via General Packet Radio Service (GPRS), till en webbtjänst som sparar värdet i en databas tillsammans med en tidsstämpel [6]. Tidsstämpeln som lagras, tillsammans med mätvärdet, anges i Coordinated Universal Time (UTC) [7]. En lampa som indikerar anslutning till webbtjänsten skall lysa så länge anslutningen fortgår, och släckas om anslutningen bryts. Data som samlas in vid bruten kontakt med webbtjänsten kommer inte att sparas i databasen utan kastas.

2.2.1.1 Externa komponenter

Den temperatursensor som skall användas i denna studie är en OneWire-sensor av typen DS18B20 [8]. OneWire är en teknik där en anslutning skall kunna användas till att både strömförsörja en enhet, läsa från den och skriva till den. Temperaturen kommer alltså att

läsas som en digital signal från sensorn.

Den GPRS-modul som kommer att användas för att kommunicera med webbtjänsten är tillverkad av Cooking Hacks [9]. Denna är byggd för att kunna användas av Raspberry Pi och Arduino Uno, men går att använda till flera olika mikrokontroller. Kommunikation med GPRS-modulen kommer att ske seriellt, vilket innebär att databitar sänds en och en [10]. Med GPRS-moduler och annan telefoniutrustning är det vanligt att man kommunicerar med Hayes-kommandon, även kallat AT-kommandon [11]. Hayes-kommandon är små textsträngar som utgör olika kommandon och som kan kombineras för att utföra olika uppgifter som till exempel att ställa in kopplingsparametrar eller initiera telefonsamtal. Den GPRS-modul som används i denna studie har utöver standardkommandon även egendefinierade kommandon som kan användas. Ett exempel på detta är `AT+CHTTPACT` som kan användas för att sätta upp en HTTP-förbindelse med en server, och på så sätt förenkla HTTP-kommunikation genom att användaren slipper sätta upp TCP-förbindelsen manuellt.

2.3 Varför man är intresserad av mätdata

Det finns många system som genererar en hel del mätdata. Exempel på detta är kontrollsystem för industriproduktion som mäter av miljövariabler som temperatur, luftfuktighet och andra parametrar som är viktiga för att resultatet av en tillverkning skall bli så bra som möjligt. Ett annat exempel är övervakningssystem inom flygindustrin, som registrerar hastighet, roderutslag och liknande.

Vad kan man då använda all denna data till? I exemplet med kontrollsystem i industriproduktion kan den användas som indikatorer som styr, till exempel, värmekällor eller avfuktare. Det kan även användas som viktigt hjälpmedel för att analysera varför en viss serie producerad vid en tidpunkt höll mycket sämre kvalitet än vanligt. Liknande det sistnämnda använder man insamlad data inom flygindustrin, där dessa används för att

analysera vad som orsakade ett haveri. Ett bra exempel på detta är den i folkmun kallade svarta lådan, färdregistreraren [12].

Med tanke på de användningsområden mycket mätdata har är det viktigt att denna är korrekt. Om mätinstrumenten man använder har inbyggda felaktigheter vet man inte vilka mätvärden som är korrekta. Insamlad data, oavsett mängd, blir därför värdelös.

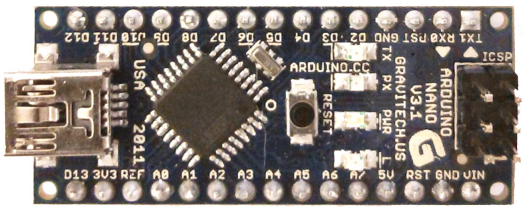
Det är inte ovanligt att onödigt mycket data samlas in och sparas. Data som man vet att man aldrig kommer att använda. Ett exempel på onödigt mycket insamlad data kan vara att man mäter av mätvärden i högre takt än de kan ändras. Men i och med att den genereras sparas den för ett eventuellt behov. På grund av den högre mängden data ställer det högre krav på hur insamlad data lagras i form av exempelvis databaser. Men det ställer även högre krav på att man skall kunna filtrera ut den data som man faktiskt har användning av.

2.4 Evaluerade mikrokontroller

I denna studie kommer tre stycken mikrokontroller att utvärderas. De tre mikrokontroller som ska utvärderas är Arduino Nano, BeagleBone Black och Netduino.

2.4.1 Arduino Nano v3

Arduino Nano är den minsta enheten i studien, endast 1s,9x4,3 cm^2 (se Figur 2.1). Det är också den minst kraftfulla enheten av de tre, sett till beräkningskraft, med en AVR-processor som har en klockfrekvens på endast 16 MHz [13][14]. Arduino Nano har ett flashminne på 32 kB, där 2 kB är reserverat för bootloadern. Det återstår då alltså endast 30 kB att utnyttja för att lagra programkod. Dess ramminne uppgår till 2 kB, och dess elektroniskt raderbara minne (EEPROM) 1 kB. Det betyder att den klassiska datorn från 1982, Commodore 64, med sina 64 kB Random Access Memory (RAM) och 20 kB Read-only Memory (ROM) hade högre lagringskapacitet för ramminnet [15].



Figur 2.1: Arduino Nano version 3

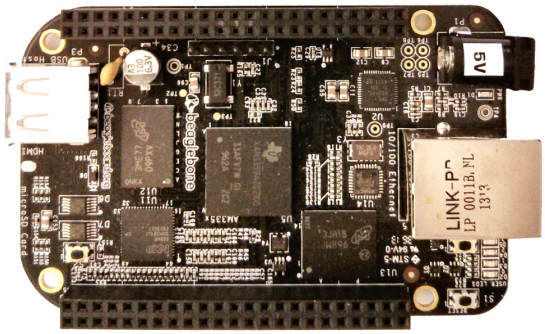
2.4.2 BeagleBone Black

BeagleBone Black är med sina 8,6x5,3 cm^2 den största, och även den kraftfullaste av de tre mikrokontroller som utvärderas (se Figur 2.2). Till skillnad från Arduino Nano och Netduino har BeagleBone Black ett fullstort operativsystem. Där Arduino Nano har 32 kB minne för att lagra programkod har BeagleBone Black 2 GB minne [16]. Dock måste man komma ihåg att operativsystemet installeras på detta utrymme, och kommer ta upp en hel del plats. Det finns även möjlighet att utöka detta med ett Micro Secure Digital-kort (MicroSD). BeagleBone Black levereras med operativsystemet Ångström, vilket är en Linuxdistribution speciellt framtagen för inbyggda system [17]. BeagleBone Black har 512 MB ramminne och en ARM-processor med en klockfrekvens på 1000 MHz [18].

Där Arduino Nano endast har ett program som körs på mikrokontrollern kommer BeagleBone Black ha ett helt operativsystem som körs. Detta gör att det går att använda mikrokontrollern som, till exempel, en webbserver, filserver och Home Theater Personal Computer (HTPC) samtidigt vilket gör BeagleBone Black till en betydligt mer mångsidig enhet än exempelvis Arduino Nano [19].

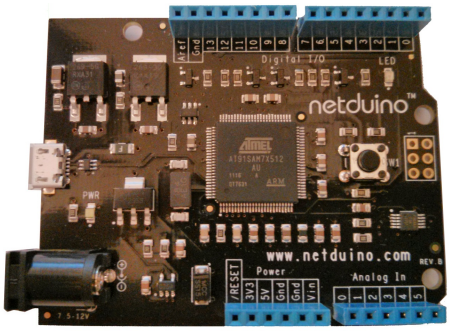
2.4.3 Netduino

Netduino (se Figur 2.3) är en enhet som prestandamässigt, och storleksmässigt med måtten 7,1x5,3 cm^2, hamnar mellan Arduino Nano och BeagleBone Black [20]. Netduino har, med sina 128 kB, mer utrymme för programkod än Arduino Nano, men betydligt mindre än BeagleBone Black. Netduino har 60 kB ramminne och en ARM-processor med en



Figur 2.2: BeagleBone Black

klockfrekvens på 48 MHz.



Figur 2.3: Netduino

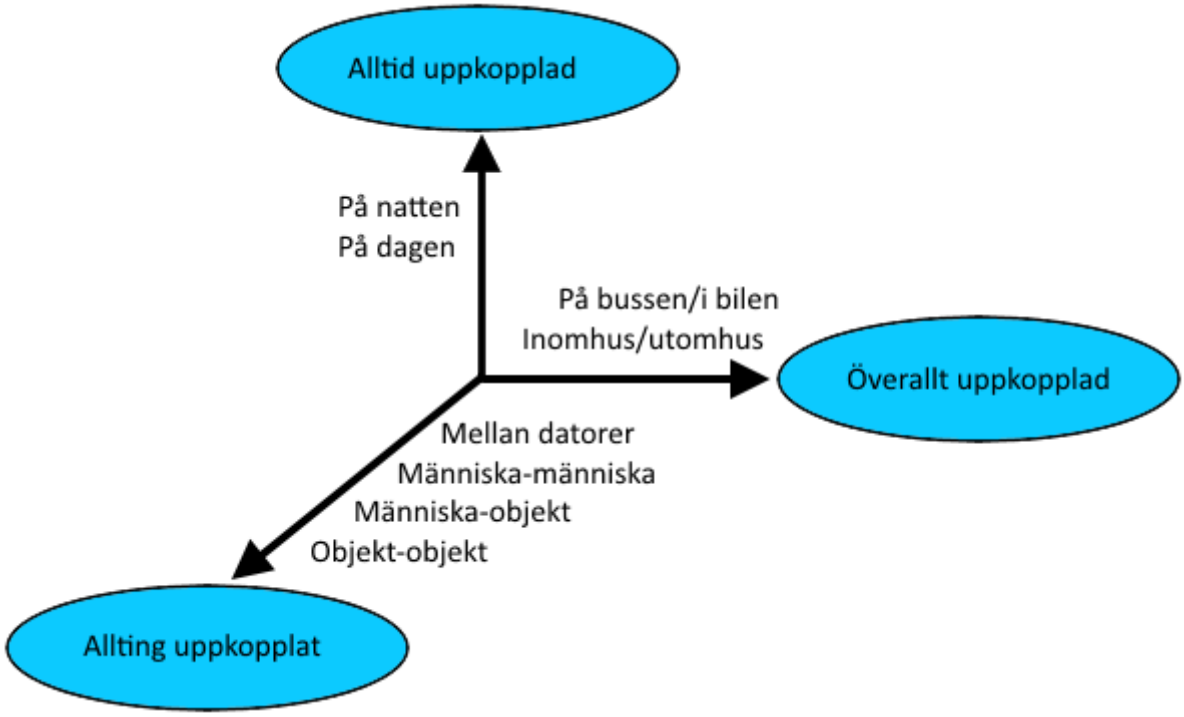
2.5 Internet of Things

Internet startade som ett nätverk av uppkopplade datorer för att kunna dela resurser och beräkningskraft med andra datorer. Med tiden har Internet kommit att utvecklas till mer än detta, det har blivit ett sätt för människor att kommunicera med andra människor. Ett exempel på detta är de senaste årens explosiva utveckling av sociala tjänster som Facebook, Twitter och Skype [21].

Internet of Things (IoT) kan ses som ytterligare en utveckling av detta. IoT är en vision om att alla – både objekt och subjekt är uppkopplade mot Internet och unikt identifierbara. En fördel med detta är att mer automatisering skulle kunna ske. Lager skulle aldrig mer

bli tomma, då man lätt skulle kunna hålla ordning på lagersaldo i realtid. Interaktioner mellan system och människa skulle kunna skräddarsys till enskilda individer och deras rättigheter. Ett mer konkret exempel på ett användningsområde är det säkerhetssystem som presenteras i artikeln ”Smart Community: An Internet of Things application”. I denna beskrivs ett säkerhetssystem som består av ett antal sammankopplade hem som bildar en virtuell miljö [22].

International Telecommunication Union (ITU) beskriver IoT som en tredje kommunikationsaxel, där de två andra axlarna är *alltid* och *överallt* (se Figur 2.4) [23]. *Alltid* realiseras genom att uppkopplingen alltid finns där, båda dag som natt. *Överallt* genom att det numera går att konsumera Internet nästan överallt genom trådlösa uppkopplingar i mobiltelefoner, surfplattor etc. Den tredje axeln blir *Allting*, när alla objekt är uppkopplade till Internet.



Figur 2.4: De tre kommunikationsaxlarna.

2.5.1 Förutsättningar

Som tidigare beskrivits innebär Internet of Things att flera miljarder enheter är uppkopplade och unikt identifierbara samtidigt. Detta ställer en hel del krav på tekniken som används för att identifiera dessa. I dagsläget är Internet Protocol (IP) det totalt dominerande protokollet för detta. För närvarande används fjärde versionen av protokollet, IPv4, för att identifiera datorer och servrar på Internet. IoT kommer inte att vara genomförbart så länge som IPv4 är den använda standarden; antalet IPv4-adresser är redan nu kraftigt begränsat. IPv4-adresserna kommer med nuvarande allokeringstakt att vara helt upptagna inom ett år [24]. De 32 bitar långa IPv4-adresserna tillåter endast adressering av 4.3 miljarder enheter [25]. Men med utrullningen av nästa version av protokollet, IPv6, med 128 bitar långa adresser kommer det gå att realisera IoT, då ca $3,4 * 10^{38}$ adresser ryms i adressutrymmet [26]. För att bättre få en förståelse för storleken på detta tal finns det möjlighet att tilldela mer än en biljard (10^{15}) adresser varannan minut i en triljon (10^{18}) år. Med andra ord kommer adressutrymmet att räcka långt in i framtiden.

Internet Society påstår att en av anledningarna till att övergången från IPv4 till IPv6 går långsamt är att det inte funnits något akut behov att av genomföra denna förändring. Man jämför med de snabba uppgraderingar, som gjordes innan år 2000, av system för att säkra dem mot millenium-buggen, där man förutspådde stora problem för system som använde tvåsiffrig representation för årtal. En annan anledning anses vara att det är en investering som både kostar personalresurser i form av utbildning, men även mycket kapital i form av nödvändiga uppgraderingar av hårdvara som routrar [27].

Att uppgradera till IPv6 är alltså en investering som kan tyckas inte ge några avkastningar på kort sikt, då olika tekniker så som Network Address Translation (NAT) har utökat adressutrymmet något genom att privata nätverk har interna IP-adresser, som sedan översätts av en gateway till en gemensam exponerad nätverksadress [28].

2.5.2 Inte utan nackdelar

När alla objekt och subjekt blir unikt identifierbara samtidigt går det att i realtid att spåra alla människor. Att intresse för detta finns både från myndigheter och företag visas av att det under 2013-2014 var vanligt med inslag i nyhetsflöden som handlade om de gigantiska övervakningsoperationerna som amerikanska National Security Agency (NSA) har organiserat tillsammans med ett antal olika länder under kodnamnet Prism, där syftet var att övervaka kommunikationen på Internet på en global nivå, dels genom att lyssna av kommunikationslänkar, dels genom att tvinga innehållsleverantörer som exempelvis Google och Facebook att dela med sig av information om användare [29].

Med IoT realiserat kommer denna information att kunna vara mycket mer omfattande, genom att allting är uppkopplat och unikt identifierbart. Detta ställer höga krav på lagar kring integritet och anonymisering. Företag vill hävda att nuvarande lagstiftning räcker, medan intresseorganisationer anser att det krävs starkare skydd för att garantera individers integritet [30].

2.6 Jämförbara system

Inom, exempelvis, pappersindustrin används ofta dyr utrustning vilken är specialtillverkad för ett specifikt ändamål [31]. Ett mål med detta projekt är att undersöka möjligheten att använda mikrokontroller, tillsammans med givare, för att uppnå samma tillförlitlighet och prestanda. Den ekonomiska faktorn är även betydande i detta fallet. Utrustning som är specifikt ändamålsutvecklad kan inte prismässigt rättvist jämföras mot prislappen på en mikrokontroller och givare. Detta gör det ytterst intressant att studera om utrustning för ungefär 1000 svenska kronor kan konkurrera med specialbyggd utrustning i en helt annan division gällande priset.

2.6.1 Raspberry Pi

En vanlig mikrokontroller på marknaden är Raspberry Pi [32]. En anledning till att den inte är med i studien är att det redan finns flertalet liknande, redan utvecklade, implementationer. Raspberry Pi kan användas till allt från HTPC, exempelvis genom att använda Xbian, till temperaturavläsning för hemmabryggning av öl [33] [34]. Det är just denna mängd av redan färdiga projekt som gör Raspberry Pi mindre intressant att studera då den får anses som välutforskad.

Av de mikrokontroller som ingår i studien är Raspberry Pi jämförbar med BeagleBone Black. De har båda betydligt kraftigare processor och mer ramminne jämfört med de mindre korten, Netduino och Arduino Nano.

2.6.2 Teensy 2.0

Ett alternativ till Arduino Nano skulle kunna vara mikrokontrollern Teensy 2.0 [35]. Både Arduino Nano och Teensy 2.0 har en klockfrekvens på 16 MHz och ansluts till en dator med hjälp av en Mini-USB-kontakt. Teensy 2.0 har med sina 25 digitala input/output-portar (I/O), 7 portar med pulsbreddsmodulering samt 12 analoga anslutningar något fler portar än Arduino Nano [36]. Arduino Nano har 14 digitala I/O-portar, där 7 av dem har pulsbreddsmodulering, och 8 analoga anslutningar [37].

2.7 Sammanfattning

I detta kapitel har bakgrunden till projektet diskuterats där nyttan av att utvärdera flera mikrokontroller tagits upp. Intressant för uppdragsgivaren är huruvida de olika enheterna skiljer sig i implementationssvårighet och tidsåtgång. Intressant är även möjligheten till användandet av tredjepartsbibliotek, för exempelvis inläsning av digitala värden från temperaturgivaren.

Detta kapitel tar upp vad mätdata kan vara, vad det kan användas till och problem

som kan uppstå när mätdata samlas i mängder men inte kan analyseras på ett fördelaktigt sätt.

De, för detta projekt aktuella, mikrokontroller har beskrivits och jämförts med motsvarande mikrokontroller som inte kommer att användas i studien. Alternativa användningsområden för mikrokontroller har nämnts.

Uttrycket Internet of Things har diskuterats för att beskriva begreppet, men även få en inblick i problem som kan uppstå när fler och fler tekniska komponenter ansluts till Internet. Det är inte längre endast persondatorer som kan dra nytta av Internet. Mobiltelefoner, surfplattor och mikrokontroller skall alla vara anslutna och unikt identifierbara på Internet. Det kräver en unik IP-adress till varje enhet, vilket ställer till problem på grund av det begränsade protokollet IPv4.

Kapitel 3

Experiment

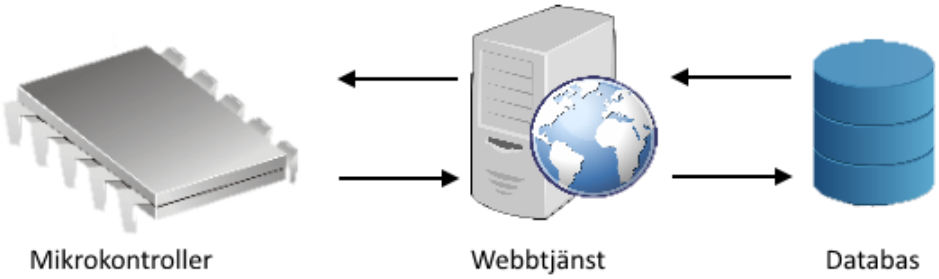
3.1 Introduktion

I detta kapitel kommer designen av webbtjänsten, samt de tekniker som används för att implementera denna presenteras. Förutom detta kommer resultatet av utvärderingen för varje mikrokontroll med avseende på plattform, programmeringsspråk och utvecklingsverktyg att redovisas.

3.2 Webbtjänst

All data insamlad från mikrokontrollerna kommer att skickas till en webbtjänst. Denna webbtjänst sparar data tillsammans med vilken mättyp det är i en databas (se Figur 3.1). För att både minska mängden data som skickas, och för att göra det möjligt att kommunicera med många olika plattformar har webbtjänsten designats utifrån Hypertext Transfer Protocol (HTTP). Denna lösning valdes för att det är en välanvänd och välkänd standard. Det finns även bibliotek för HTTP-begäranden till i princip alla plattformar. Tjänsten är baserad på Microsofts Active Server Pages Web Application Programming Interface (ASP.net Web API). Detta är en utveckling av deras Model View Controller

(MVC)-ramverk som designats för att enkelt kunna bygga HTTP-tjänster.



Figur 3.1: Översikt över systemet

3.2.1 Hypertext Transfer Protocol-meddelanden

HTTP är det protokoll som används för att transportera meddelanden på World Wide Web (WWW). HTTP-meddelanden är uppdelade i två delar, ett meddelandehuvud och en meddelandekropp. Beroende på meddelandetyp är meddelandekroppen valbar. HTTP är anpassat efter klient/server arkitekturen, där det finns en server som alltid är uppkopplad och som klienter begär resurser från. Därför är meddelanden uppdelade i begärandemeddelanden och svarsmeddelanden. För begärandemeddelanden finns det nio olika typer; GET, POST, PUT, DELETE, OPTIONS, HEAD, TRACE, CONNECT och PATCH [38, sid 35]. De mest använda är GET, som används för att begära resurser från servern och POST som används för att skicka data till servern.

Varje HTTP-svarsmeddelande är associerat till en viss tresiffrig statuskod, uppdelade i fem kategorier. Den första kategorin, med statuskod 1**, är informella meddelanden. Dessa kan till exempel handla om att servern begär att klienten skall fortsätta skicka meddelandekropp för ett meddelande den fått ett huvud för, eller att den accepterat en begäran om att byta protokollversion.

Den andra kategorin, med statuskod 2**, är meddelanden associerade med någon form av framsteg. Dessa kan till exempel vara att servern svarar med en webbsida från en

begäran, att en resurs skapats hos servern.

Den tredje kategorin, med statuskod 3**, är meddelanden som är associerade med vidarebefordring. Med dessa kan servern underrätta klienten om att resurser flyttats, både temporärt och permanent.

Den fjärde kategorin, med statuskod 4**, är meddelanden associerade med fel gjorda av klienten. Det kan till exempel vara att klienten har gjort en begäran om en resurs som inte finns (felkod 404), eller att klientens begäran inte kunde tolkas (felkod 400).

Den femte och sista kategorin, med statuskod 5**, är meddelanden associerade med fel hos servern. Några av de vanligaste är att servern har drabbats av ett internt fel (felkod 500) eller att den av klienten begärda tjänsten är otillgänglig (felkod 503).

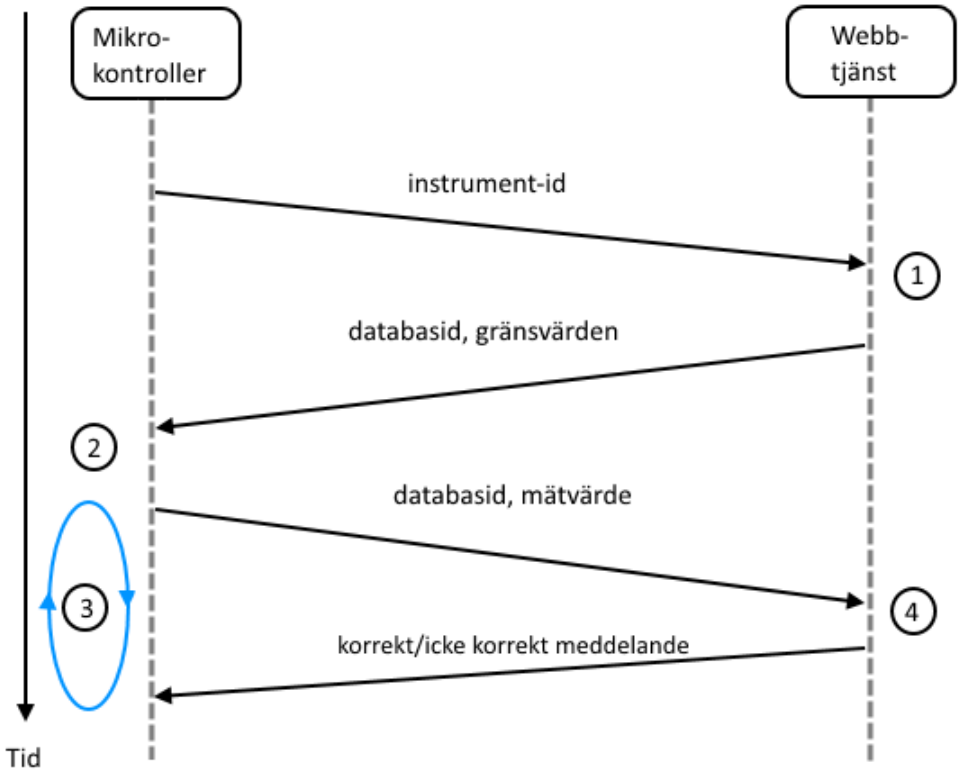
Meddelandehuvudet kan användas för att specificera meddelandet ytterligare. Exempel på detta är att klienten kan bestämma vilken meddelandetyp den förväntar sig svaret i, eller vilket språk den accepterar svar i. Servern kan använda meddelandehuvudet för att beskriva detaljer om svaret. Till exempel kan den använda det för att beskriva vilken typ av meddelande den svarar med i meddelandekroppen, eller när resursen den svarar med senast ändrades.

3.2.2 Design av kommunikationsprotokoll

Varje mikrokontroller kommer vid uppstart kontakta webbtjänsten för att hämta aktuella värdesgränser. Med denna förfrågan kommer den att skicka med ett unikt identifikations (id)-nummer för den sensor mikrokontrollern vill hämta värdesgränser för. Webbtjänsten svarar med ett id-nummer, som är sensorns databasid tillsammans med värdesgränser för denna sensor. Om sensorn tidigare inte varit registrerad hos webbtjänsten kommer id-numret för sensorn att läggas in i databasen tillsammans med förbestämda gränsvärden för nya sensorer av den typen.

Mikrokontrollern kommer sedan att med egenvalda tidsintervall skicka mätvärden tillsammans med det från webbtjänsten erhållna identifikationsnumret. Webbtjänsten kommer

att svara med ett HTTP-meddelande om inskickad data är korrekt eller inte (se Figur 3.2). Korrekt i detta fall innebär att data skall vara av förväntad datatyp för mättypen och att HTTP-meddelandet skall vara korrekt formaterat.



Figur 3.2: Kommunikationsflöde mellan mikrokontroller och webbtjänst

1. Webbtjänsten tar emot ett id från mikrokontrollern. I exemplet i bilaga A.1 är sensorns id ”netduino_temp”. Webbtjänsten undersöker om detta id sedan tidigare är registrerat genom att kolla upp det mot databasen. Är det registrerat sedan tidigare kommer den att leta upp gränsvärden i databasen för aktuell sensor och returnera dem, som ett HTTP 200 OK meddelande, där meddelandekroppen består av ett sensor-objekt i Javascript Object Notation (JSON) notation. JSON är ett sätt att representera ett objekt som en sträng, med olika tecken för att markera fält och värden [39]. Om webbtjänsten inte hittar sensorns id kommer den att skapa ett nytt i databasen, med förbestämda gränsvärden och returnera detta objekt.

2. Mikrokontrollern tar emot HTTP-meddelandet från webbtjänsten och undersöker dess innehåll. Om den mottagit ett meddelande med HTTP 200 OK kommer den att försöka återskapa ett Sensor-objekt utifrån det JSON-objekt webbtjänsten har returnerat. Om den mottar ett meddelande som den inte kan hantera kommer den att fråga webbtjänsten på nytt, tills den får ”rätt” svar. Webbtjänsten kan svara med antingen ett fullständigt svar eller ett förkortat svar som utelämnar viss information, som exempelvis namn (se bilaga A för exempel). Ett förkortat svar från webbtjänsten skulle kunna se ut så här (endast innehållet i meddelandet, för att se huvudet, se bilaga A.2)

   ```
   {''Id'':19,''SensorId'':''13",''U'':25.0,''D'':18.0}
   ```

3. Mikrokontrollern kommer sedan att med egendefinierade intervaller skicka in mätdata till webbtjänsten som HTTP POST-meddelanden. Dessa är utformade så att all data mikrokontrollern vill skicka till webbtjänsten lägger den i Uniformed Resource Locator (URL).

   Exempel: Mikrokontrollern vill skicka in mätdata 24,5 och den har erhållit id-nummer 1 från webbtjänsten. Den skickar då detta som ett HTTP POST-meddelande till webbtjänsten genom

   ```
   [webbtjänstens URL]?data=24,5&sensorid=1
   ```

   Att denna lösning valts är för att dels minska antalet byte som måste skickas för att möjliggöra rapportering över långsammare och databegränsade uppkopplingar, dels för att minska komplexiteten i de meddelanden mikrokontrollern skickar till webbtjänsten. Ett exempel på kommunikationen mellan mikrokontrollern och webbtjänsten vid rapportering av mätdata kan ses i bilaga A.3

4. Webbtjänsten tar emot HTTP POST-meddelanden med rapportering av mätdata. Webbtjänsten kontrollerar att detta meddelande är korrekt formaterat. Detta innebär att all data som är påtvingat ett mätvärde, i detta fall mätdata och sensorid, finns i meddelandet. Webbtjänsten kontrollerar även att id-numret är korrekt, det vill säga att det finns en sensor i databasen med detta id-nummer. Om inskickat meddelande är korrekt svarar webbtjänsten med HTTP 201 Created. Om det inte är korrekt svarar den istället med HTTP 400 Bad Request.

3.2.3 Implementation av webbtjänst

Webbtjänsten består av ett antal kontroller, som i Microsofts ASP.NET MVC är den komponent som står för interaktionen med användaren. Dessa definierar gränssnittet för webbtjänsten utåt. Webbtjänsten har kontroller för avläsning, enhet, sensor samt mätningstyp. Varje kontroll har ett ansvarsområde. Avläsningskontrollen är den kontroll mikrokontrollerna kommunicerar med för att rapportera mätvärden. Den har även möjlighet att returnera mätvärden som lagrats i databasen. De tre andra kontrollerna; enhet, sensor och mätningstyp kan användas för att skapa nya objekt och returnera de som finns registrerade. Detta kan ske både enskilt genom deras id-nummer samt genom att returnera alla objekt av de olika slagen. För att mappa entiteter i databasen till instanser av klasser i webbtjänsten har Microsofts Entity Framework använts som Object-Relational Mapping (ORM) [40] [41].

3.2.4 Databas

Systemet som används för att lagra data är Microsoft SQL Server [42]. Entitet-Relationsdiagram (ER-diagram) för databasen finns i bilaga B.

När en ny tupel skapas i de olika tabellerna sätts Id med hjälp av automatisk ökning. Alla fält i databasen som är primärnyckel, vilket innebär att värdet i fältet måste vara unikt, har egenskapen med automatisk ökning.

Värdet som lagras i fältet Name, som finns i tabellerna Device, SensorType och Sensor, får aldrig vara null. Detta för att tupeln senare skall kunna identifieras av användaren med något mer lättläst än det siffervärde som hittas fältet Id. Det antal tecken ett namn får ha är begränsat till 15 tecken. Begränsningen gäller i alla tabeller där Name finns och är till för att hålla nere antalet tecken som vid ett senare tillfälle kommer att sparas i ramminnet på Arduino Nano.

Att ha sensortyp och enhet i egna tabeller, istället för i sensor-tabellen, medför att det blir enklare att utöka informationen vid ett senare tillfälle. Information att utöka med skulle till exempel kunna vara mätprecision för sensortypen eller positionen på enheten.

3.2.4.1 Device

Tabellen Device består av två fält (se Tabell 3.1). Fältet Id är tabellens primärnyckel. Name är fältet där namnet på enheten sparas, exempelvis ”Arduino_03”.

	Namn	Datatyp	Null
🔑	**Id**	int	☐
	Name	nvarchar(15)	☐

Tabell 3.1: Device

3.2.4.2 SensorType

Tabellen SensorType består av två fält (se Tabell 3.2). Fältet Id är tabellens primärnyckel. Name är fältet som beskriver sensortypen, exempelvis ”Temperatur”.

	Namn	Datatyp	Null
🔑	**Id**	int	☐
	Name	nvarchar(15)	☐

Tabell 3.2: SensorType

3.2.4.3 Sensor

Tabellen Sensor består av sex fält (se Tabell 3.3). Fältet Id är tabellens primärnyckel. Name är fältet där namnet på sensorn sparas. I fälten U och D lagras de avgränsningar som en sensor kan ha. Exempelvis kan värdet på U vara 25 och värdet på D vara 18. Det bestämmer inom vilka intervall ett mätvärde ska anses som ett bra mätvärde, exempelvis för att avgöra om en temperatur är för hög, för låg eller lagom. Intervallvärdena får vara null då en sensor kanske inte behöver bägge intervallgränser. DeviceId är en främmandenyckel som kopplar samman en sensor med en enhet (se Tabell 3.1). SensorTypeId är en främmandenyckel som kopplar samman en sensor med en sensortyp (se Tabell 3.2).

Om en enhet, eller en sensortyp, som är kopplad till en sensor tas bort kommer DeviceId, eller SensorTypeId, att sättas till värdet ett. Det kännetecknar en okänd enhet eller en okänd sensortyp. Detta för att inte alla mätvärden skall försvinna om en enhet, eller sensortyp, tas bort. Att inte alla mätvärden tas bort om en enhet tas bort kan vara av intresse om man vill flytta en sensor till en ny enhet och ta bort den gamla enheten. Om man skulle råka ta bort en enhet innan man kopplat om sensorn, i databasen, skulle alla mätvärden sensorn samlat in försvinna. Det är inte önskvärt.

	Namn	Datatyp	Null
🔑	**Id**	int	☐
	Name	nvarchar(15)	☐
	U	float	☑
	D	float	☑
🔑	*DeviceId*	int	☐
🔑	*SensorTypeId*	int	☐

Tabell 3.3: Sensor

3.2.4.4 Reading

Tabellen Reading består av fyra fält (se Tabell 3.4). Fältet Id är tabellens primärnyckel. I fältet Data sparas ett mätvärde från en sensor. Värdet i data måste vara ett heltal eller flyttal. I fältet Timestamp sparas en tidstämpel för avläsningen bestående av datum och klockslag. Fältet SensorId är en främmandenyckel som kopplar samman en avläsning med en sensor (se Tabell 3.3). Inga fält i tabellen får ha värdet null, då varje avläsning måste ha någon slags mätvärde, en tidsangivelse som indikerar när mätvärdet rapporterats och en sensor som läst av värdet.

	Namn	Datatyp	Null
🔑	**Id**	int	☐
	Data	float	☐
	Timestamp	datetime2(7)	☐
🔑	*SensorId*	int	☐

Tabell 3.4: Reading

3.3 Arduino Nano

Arduino Nano har utvärderats enligt de riktlinjer som sattes upp i avsnitt 2.2.

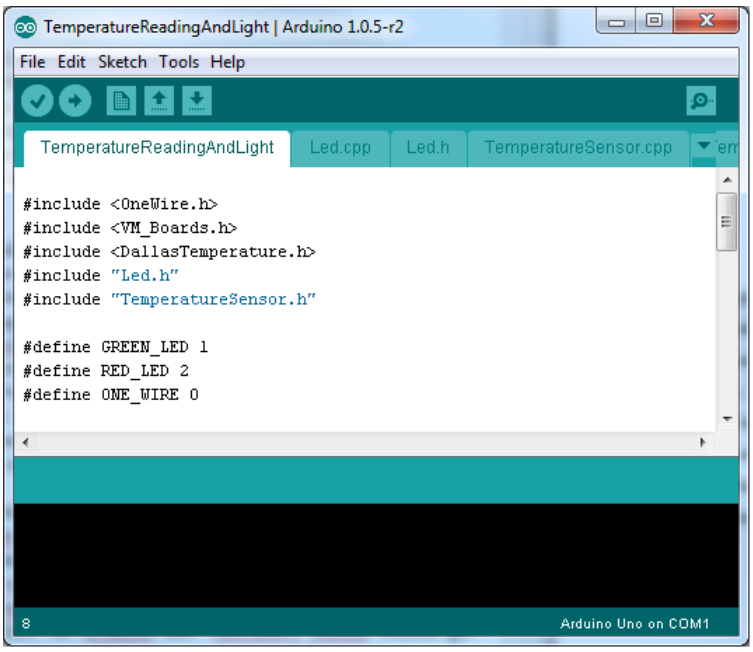
3.3.1 Utvecklingsmiljö

Till Arduino finns det en integrerad utvecklingsmiljö (IDE) [43] (se Figur 3.3). Det är en utvecklingsmiljö som är helt oberoende av vilken plattform den körs på så länge som plattformen stödjer Java som utvecklingsmiljön är skriven i.

Utvecklingsmiljön heter Arduino IDE och är baserad på utvecklingsmiljön Wiring, som i sin tur är baserad på utvecklingsmiljön för programspråket Processing [44] [45]. Processing är ett programspråk speciellt utvecklat för elektronisk konst och visuell design med syftet

att underlätta inlärning av datorprogrammering genom att göra det mer visuellt och ge icke-programmerare en möjlighet att få snabba resultat i form av visuell återkoppling.

I likhet med Arduino IDE är programmeringsspråket i Wiring C/C++. Wiring har ett medföljande programbibliotek som är till för att underlätta I/O-operationer. Ett program utvecklat i Arduino IDE, eller Wiring, kräver endast två funktioner för att fungera: setup(), som är en funktion som körs en gång i början på programmet där alla startvärden för programmet sätts, loop() som kommer att köras så länge som enheten är inkopplad till en strömförsörjningskälla. Detta betyder inte att man är begränsad till dessa två funktioner, setup() och loop(), utan man har självklart möjlighet att göra ett program med klasser, objekt och metoder.



Figur 3.3: Arduino IDE

Ett program till Arduino kallas för en Sketch. Detta är en kvarleva som följt med från Processing, till Wiring och slutligen till Arduino IDE. Arduino IDE har de flesta grundläggande funktioner som exempelvis syntaxframhävning, matchande parenteser och automatisk indentering. Arduino IDE saknar däremot avancerade felsökningsvertyg som

man kan hitta i Visual Studio. Det finns dock tillägg till Visual Studio som möjliggör att utveckla sina Arduino-program i Visual Studio istället för Arduino IDE. Visual Studio ger en mycket bättre möjlighet till överblick av ett projekt än vad Arduino IDE kan göra. Under detta projekt har Visual Studio använts för utveckling till Arduino på grund av Visual Studios inbyggda möjligheter för att versionshantera. För att få ut maximal möjlighet till felsökning vid utveckling av Arduino-program i Visual Studio krävs ytterligare tillägg. Detta tillägg tillhandahålls av Visual Micro och kostar i skrivande stund 29$ [46]. Visual Micros tillägg ger möjlighet till felsöking under drift, vilket inte är möjligt utan tillägget Visual Micro. Vanlig felsökning i Arduino-program består av meddelanden som skrivs till en seriell port vilken kan läsas av genom USB-anslutning och antingen Arduino IDE eller Visual Studio. En viktig sak att tänka på är att alla dessa felsökningsmeddelanden tar upp ramminne.

3.3.1.1 Användarvänlighet

Arduino IDE är specifikt utvecklat med nybörjare i åtanke. Det gör programmet väldigt enkelt att sätta sig in i, speciellt om man har vana av utvecklingsverktyg sedan tidigare.

Visual Studio ställer högre krav på användaren när det gäller vana av utvecklingsverktyg, men ger användaren större möjligheter.

3.3.1.2 Felsökningsmöjligheter

Möjligheten till felsökning är ganska begränsad utan programtillägg som specifikt möjliggör felsöking. Utan extra programvara är man begränsad till att skriva felsökningsmeddelanden till den seriella porten. Detta ställer till det om man har en extern modul ansluten, som i denna studie varit en GPRS-modul, som kommunicerar seriellt då alla meddelanden som skickas till felsökningsfönstret via USB-porten även skickas till den externa modulen som i sin tur försöker tolka dessa meddelanden.

Det går dock att ändra vilka portar man använder för sändning och mottagning via seri-

ell kommunikation. Ett tilläggsbibliotek, SoftwareSerial, som är inkluderat i standardutgåvan av Arduino IDE, gör det möjligt att dela kommunikationen så inte felsökningsutskrifter och instruktioner till externa moduler krockar med varandra [47]. På grund av den låga prestandan i Arduino Nano krävs att man sänker kommunikationshastigheten mellan Arduino Nano och den externa modulen om man vill använda detta tillägg. För att Arduino Nano ska klara kommunikationen genom att använda SoftwareSerial krävs att man ställer detta värde till högst 57600 baud [48].

3.3.2 Plattform

Arduino Nano saknar operativsystem. Arduino Nano har endast en bootloader, som startar det program som man kompilerat och lagrat på enheten.

3.3.2.1 Community

För att undersöka populariteten av plattformarna har en undersökning gjorts. För en utförlig redogörelse av undersökningen, se bilaga E.

Arduino har ett eget diskussionsforum där användare kan diskutera plattformen. Vid undersökningens början var antalet registrerade användare nästan 190 000. Aktiviteten på diskussionsforumet är förhållandevis hög. Undersökningen visar att det skrivs ca 1470 inlägg per dag i genomsnitt. Även medlemsantalet speglar detta, då i regel 220 nya användare registrerar sig per dag. Förutom detta diskuteras plattformen på Stack Overflow, där i snitt 9 nya inlägg görs varje dag [49].

3.3.2.2 Producentstöd

Det finns en uppsjö av hårdvara utvecklad för Arduino. Mycket av hårdvaran stöds även av andra enheter, som BeagleBone Black och Raspberry Pi. För att ansluta extern hårdvara till Arduino Nano kan man använda ett kretskort som utökar antalet I/O-portar på enheten. Det går även bra att koppla produkter direkt till enheten med hjälp av sladdar

och kontakter. På en del av Arduinos större modeller finns möjligheten att fästa extern hårdvara direkt på enheten. Till exempel går det att fästa GPRS-modulen, som används i denna studie, direkt, utan sladdar och dylikt, på en Arduino Uno. Det kan vara smidigt att slippa sladdar mellan de olika hårdvaruenheterna, men det kan samtidigt begränsa antalet I/O-portar. Om man använder GPRS-modulen som exempel så kommer den att ta upp fler I/O-portar än nödvändigt och fäster man den direkt på Arduino Uno så blockerar den även många portar. Fördelen är att man slipper fundera över var sladdarna mellan enheterna ska sitta och man slipper sladdhärvan som kan uppstå när många portar kopplas samman mellan enheterna.

3.3.2.3 Dokumentation

Det finns en hel del dokumentation till Arduino-systemen. Mycket av informationen kan man nå via den officiella hemsidan [50]. Det finns flertalet exempel där man kan hitta både kopplingsschema och programkod till de olika exemplen. Om en krets kräver fler komponenter, exempelvis elektriska motstånd, är även de angiva i kopplingschemat. Det finns också en Arduino-wiki där alla användare kan bidra med information, sprida kunskap och dela med sig av utförda projekt [51].

3.3.2.4 Möjlighet att uppdatera i drift

Arduino Nano saknar möjlighet att uppdatera under drift i standardutförande. Vid uppdatering av programvara krävs ett uppehåll vid överföring av programkod och omstart av enheten. Arduino Nano, i standardutförande, saknar även möjlighet till fjärrstyrning. Det innebär att uppdatering av programkoden på enheten kräver att man har fysisk tillgång till enheten och även att man har möjlighet att störa driften under några minuter. Kravet på tillgång till den fysiska enheten kan tänkas bli ett problem om enheten sitter på ett svårtåtkomligt ställe och rapporterar data.

Det ska dock inte ses som en omöjlighet att åtgärda problemet med fjärruppdatering.

Det går att bygga ut Arduino Nano med ett nätverksuttag, alternativt investera i en Arduino-modell som redan har nätverksuttag, exempelvis Arduino Ethernet. Det kräver även ett nytt startprogram till Arduino som stödjer fjärruppdateringar. Arduino-tftpboot är ett program som ger möjlighet att läsa in nätverksinställningar från EEPROM istället för att läsa från standardinställningarna [52]. Det kräver även en sketch där de nya inställningarna lagras. Man får även ta i beaktande att det som lagras i EEPROM kommer att skrivas över vid varje ny uppdatering av programvaran [53].

3.3.2.5 Första-/Tredjepartsbibliotek

Arduino tillhandahåller en del standardbibliotek till plattformen, som bara kräver att man inkluderar dem i projektet. Bland dessa finns till exempel SoftwareSerial för att man skall kunna använda valfria digitala portar på enheten för seriell kommunikation. Förutom detta finns bibliotek för att läsa och skriva till EEPROM, använda WiFi och för att använda olika displayer som Liquid Crystal Display (LCD) eller Thin Film Transistor (TFT) [54] [55].

Arduino har även samlat tredjepartsbibliotek på sin webbsida. Dessa är kategoriserade efter användningsområde (till exempel kommunikation och kryptografi) och är ganska många till antal, i skrivande stund ca 200 st. Många av dessa är skrivna för att förenkla kommunikation med en typ av sensor, till exempel OneWire [8]. Men det finns även andra mer generella bibliotek, som bibliotek för förenkling av användande av timers och diverse matematikbibliotek.

3.3.3 Plattformens programmeringsspråk

Arduino Nano använder en anpassad version av C/C++ speciellt designad för att minimera utrymmet som används då Arduino Nano har ett kodutrymme på endast 30 kB. Klassen String är till exempel en nedbantad version och innehåller inte alla de metoder som vanligtvis finns tillgängliga för String-klassen. Vector-klassen, för enkel implementation av listor,

saknas helt.

En annan begränsning på Arduino Nano är att möjligheten till att skriva flertrådade program saknas. Det finns dock tredjepartsbibliotek som simulerar trådning genom att schemalägga uppgifter med fasta, eller rörliga, tider [56].

3.3.3.1 Popularitet

C får anses som ett synnerligen välanvänt språk. Enligt en undersökning genomförd av Tiobe toppar C listan, för mars 2014, över mest använda programmeringsspråk med 17,5% av marknaden. C++ hamnar på fjärde plats med 6,3% av marknaden enligt Tiobes undersökning [57].

3.3.3.2 Community

Arduino tillhandahåller en egen avdelning på sitt diskussionsforum där användare kan diskutera programspråksspecifika frågor. Förutom detta kan användaren vända sig till Stack Overflow, där C++ diskuteras relativt flitigt. I genomsnitt skrivs det 302 inlägg varje dag relaterat till programspråket. För en detaljerad redogörelse av undersökningen se bilaga E.

3.4 BeagleBone Black

BeagleBone Black har utvärderats enligt de riktlinjer som sattes upp i avsnitt 2.2.

3.4.1 Plattform

BeagleBone är utöver bootloadern även utrustad med ett operativsystem. I detta fall en fullstor Linuxdistribution. För tillfället finns många av de populära distributionerna till plattformen, till exempel Ubuntu, Arch Linux, Gentoo och Fedora. Enheten levereras med distributionen Ångström, som är ämnad för användning på inbyggda system.

3.4.1.1 Communitystorlek

BeagleBoard har ett eget diskussionsforum dit användare kan vända sig för att diskutera plattformen. Aktiviteten på detta har varit varierande, men har i genomsnitt legat runt 2000 inlägg per månad. Undersökningen visar att medlemsantalet i diskussionsforumet ökar med i genomsnitt 7 per dag. Undersökningen av hur många inlägg som skrivs på Stack Overflow om plattformen visar att det i genomsnitt skrivs 0,2 nya inlägg per dag. För en fullständig redogörelse för undersökningen se bilaga E.

3.4.1.2 Producentstöd

BeagleBoard har med jämna mellanrum släppt nya revisioner av plattformen. Dessa syftade främst till att lösa buggar i tidigare utgåvor, som till exempel problem med seriell kommunikation. Andra ändringar har gjorts för att optimera plattformen [58].

Uppdateringar av mjukvaran till befintliga plattformar sköts inte av BeagleBoard själva, utan av den Linuxdistribution användaren valt att använda på plattformen.

3.4.1.3 Dokumentation

BeagleBoard har, utöver sin wiki, även försett BeagleBone Black med en referensmanual. Denna manual täcker in all hårdvara på kortet, till exempel GPIO-portar, nätverksportar och USB-portar. Förutom detta är alla komponenter beskrivna med kopplingscheman. BeagleBoard tillhandahåller även en del exempel på hur man kan programmera enklare kretsar som att tända LED, mäta värde från potentiometer med mera, men endast med deras egenutvecklade programmeringsspråk BoneScript.

3.4.1.4 Möjlighet att uppdatera i drift

BeagleBone Black kör en fullstor Linuxdistribution som operativsystem. Detta innebär att program på den körs i egna processer. Genom att program körs som enskilda processer medför det att de kan ladda ner en ny, modifierad version av sin programvara och sedan

starta om sig med den nedladdade filen. Detta tillåter att programvaran uppdateras utan att enheten behöver startas om.

3.4.1.5 Första-/Tredjepartsbibliotek

Genom att BeagleBone Black kör en fullstor Linuxkärna innebär det att alla bibliotek som kan exekveras på plattformens processorarkitektur (ARM) kan användas. BeagleBoard har skrivit en del bibliotek, fria att använda, men som är till deras egenutvecklade programmeringsspråk BoneScript.

Texas Instruments har skrivit en del moduler till Linuxkärnan för att lättare kunna arbeta med vissa sensorer, till exempel OneWire-sensorn DS18B20 som har använts för att mäta temperaturen i denna utvärdering. Det finns dock ingen ordentlig förteckning över vilka sensorer man skrivit moduler till eller var man kan finna dessa.

3.4.2 Plattformens programmeringsspråk

BeagleBoard har utvecklat ett eget programmeringsbibliotek för BeagleBone Black, kallat BoneScript. BoneScript syftar till att vara optimerat för plattformen och samtidigt använda funktionsanrop som liknar de som finns till Arduino [59]. BoneScript är en utveckling av Node.js och har stöd för ett flertal olika operationer som skrivning till och från analoga och digitala portar och skrivning till fil. BoneScript använder sig av samma syntax som Javascript, som i sin tur använder sig av samma syntax som C-familjen. I Javascript och Node.js sker de flesta anropen asynkront [60]. Detta innebär att programmeraren får lägga till anrop som skall köras när ett uttryck har exekverat klart, för att säkerställa att dessa körs sekventiellt.

Genom den fullstora Linuxkärnan är man dock inte låst till ett specifikt programmeringsspråk, utan man kan fritt välja ett programmeringsspråk som kan köras i Linuxmiljö, till exempel C/C++, Python och Ruby. Under denna studie har C++ använts som programmeringsspråk för utveckling på BeagleBone, men även BoneScript har undersökts i

viss mån. BeagleBone implementerar hela C++, med alla dess standardbibliotek. Detta innebär att flertrådade program kan skrivas.

3.4.2.1 Popularitet

BoneScript är i skrivande stund inte så populärt. På Stack Overflow finns det inget ämne associerat med biblioteket, och diskussionsämnen relaterade till BoneScript på BeagleBoards egna diskussionsforum är ganska konstant, det vill säga inga nya frågor dyker upp kring ämnet. Javascript, som programmering i BoneScript till största delen består av, är dock populärt. Det ligger på plats nio i Tiobes undersökning i mars 2014, en ökning från plats elva året innan [57].

3.4.2.2 Community

BeagleBoard tillhandahåller en underavdelning i sitt diskussionsforum där användare kan diskutera programspråksspecifika frågor. Förutom detta kan användaren vända sig till Stack Overflow, där C++ diskuteras relativt flitigt. I genomsnitt skrivs det 302 inlägg varje dag relaterat till programspråket. För en detaljerad redogörelse av undersökningen se bilaga E.

3.4.3 Utvecklingsmiljö

Förutsatt att användaren kör den till plattformen medföljande distributionen, Ångström, finns en webbaserad utvecklingsmiljö till BeagleBone. Denna utvecklingsmiljö heter Cloud9 och är främst utvecklad för att användas till Javascript och Node.js, men kan även användas till att programmera i andra programmeringsspråk, till exempel Ruby eller Hypertext Preprocessor (PHP). Cloud9 tillåter att flera utvecklare jobbar i projektet samtidigt och bistår med en del kodkomplettering för variabler och klasser.

Cloud9 innehåller en hel del felsökningsmöjligheter. Bland annat kan man sätta ut brytpunkter, titta på stacken för funktionsanrop och inspektera innehållet i variabler [61].

BeagleBone är dock inte låst till denna utvecklingsmiljö; användaren kan välja valfri utvecklingsmiljö och sedan ladda upp filerna till plattformen med File Transfer Protocol (FTP) eller Secure Copy (SCP). I denna utvärdering har Visual Studio använts för att utveckla program till plattformen. Denna utvecklingsmiljö beskrivs utförligare i avsnitt 3.5.1. Utveckling i Visual Studio kan underlättas genom att använda VisualGDB [62]. Med detta kan man utveckla, ladda upp och felsöka program på BeagleBone Black direkt i Visual Studio. Visual GDB gör det även möjligt att kompilera programvaran på utvecklingsdatorn och sedan ladda upp den till mikrokontrollern.

Vid programmering i exempelvis C eller C++, kan felsökning även göras med The GNU Project Debugger (GDB) [63]. GDB är ett populärt verktyg för felsökning som tillåter utvecklaren att sätta ut brytpunkter, stega igenom instruktioner i programkoden och undersöka innehåll i variabler.

3.5 Netduino

Netduino har utvärderats enligt de riktlinjer som sattes upp i avsnitt 2.2.

3.5.1 Utvecklingsmiljö

Netduino rekommenderar Visual Studio som utvecklingsplattform, tillsammans med deras Netduino Software Development Kit (SDK). Denna utvecklingsmiljö kräver dock Windows för att köra. Det går dock att utveckla och distribuera program i andra operativsystem som Linux eller MacOS förutsatt att man virtualiserar Windows med till exempel Parallels eller VMWare och sedan laddar upp programvaran till mikrokontrollen via seriell kommunikation. Det finns även ett öppen källkod-projekt drivet av Secret Labs som syftar till att tillåta utveckling på både Mac OS samt Linux med Mono [64]. I denna studie har den rekommenderade utvecklingsmiljön använts, vilket är Visual Studio 2010 Professional. Experimentella varianter under utveckling finns dock av Netduino SDK till Visual Studio

2012 samt Visual Studio 2013.

Visual Studio är utvecklat av Microsoft. Visual Studio finns i ett antal olika varianter, däribland en gratis variant - Express. Express är en nedbantad version av betalversionerna, men är tillräckligt funktionell för att utveckla och distribuera programvara till Netduino.

Kodredigeraren i Visual Studio är utrustad med både syntaxframhävning och kodkomplettering. Kodkompletteringen är utvecklad med IntelliSense som har möjlighet att komplettera variabler, funktioner och metoder. Men den kan även komplettera klassnamn och vissa språkkonstruktioner som iterationer.

3.5.1.1 Producentstöd

Visual Studio har ett starkt producentstöd från Microsoft, som är aktiva med att tillhandahålla uppdateringar och buggfixar.

3.5.1.2 Community

Communityn för Visual Studio är stor, dels genom att det är många som använder det, men även för att det tillåter utvecklare att skriva egna utökningar av funktionaliteten. Microsoft har även skapat en central plats, Visual Studio Gallery, där utvecklare kan dela med sig av sina utökningar med andra.

3.5.1.3 Felsökningsmöjligheter

Felsökningsmöjligheterna med Visual Studio och Netduino är många. Netduino kan ta tillvara på alla felsökningsmöjligheter som finns inbyggt i Visual Studio. Detta inkluderar både att kolla på källkoden som exekveras och att sätta ut olika brytpunkter där man vill att exekvering tillfälligt skall stoppas eller att lägga till bevakning av variabler.

3.5.2 Plattform

Netduino saknar operativsystem. Netduino har endast en bootloader, som startar det program som man kompilerat och lagrat på enheten.

3.5.2.1 Communitystorlek

Netduino har ett eget diskussionforum, dit man kan vända sig om man behöver hjälp både vad gäller plattformsspecifika frågor men även mjukvarurelaterade. Detta är tämligen välbesökt; i genomsnitt skrivs ungefär 25 nya inlägg varje dag. Antalet medlemmar ökar med i genomsnitt 19 medlemmar per dag. Förutom tillverkarens egna diskussionsforum finns även en del diskussioner på Stack Overflow om plattformen. På Stack Overflow är det dock ingen större aktivitet, antalet diskussionsämnen ökar i princip ingenting. För en fullständig redogörelse av undersökningen beträffande plattformens popularitet se bilaga E

3.5.2.2 Producentstöd

Producenten av Netduino, Secret Labs, arbetar kontinuerligt med att uppdatera sina enheter till de senaste versionerna av .NET Micro Framework. Den första generationen av företagets mikrokontroller, dit Netduino i studien tillhör, lider dock alla av för lite flashminne. Detta har inneburit att det inte funnits möjlighet att implementera nyare versioner av .NET Micro Framework än 4.2. Det verkar som om detta är den sista versionen av ramverket som de officiellt kommer att erbjuda. Företagets andra generation av mikrokontroller har dock alla uppdaterats till den senaste versionen av ramverket, version 4.3.

3.5.2.3 Dokumentation

Kvantitativt finns det inte mycket dokumentation om Netduino. Den har ett specifikationsblad som anger vilka I/O-portar som är kopplade till vad beträffande seriell kommunikation och analoga signaler [65]. Vilken spänning de digitala signalerna har för etta och nolla är

tydligt angivet, likaså hur stor ström de maximalt klarar av för att inte skadas. Utöver detta finns även ett kopplingsschema för mikrokontrollern, med alla ingående komponenter samt mikrokontrollerns layout i Board-format för att kunna läsas i CadSoft EAGLE [66]. Netduino har även en egen wiki, där både användare och deras egna medarbetare bidrar med information. Bland denna information kan man till exempel finna vilka expansionskort som testats och därmed fungerar tillsammans med Netduino.

3.5.2.4 Möjlighet att uppdatera i drift

Netduino har i standardutförandet ingen möjlighet att uppdateras i drift. Detta beror på att den endast har ett kodutrymme som bootloadern kan ladda program från vid uppstart. För att kunna uppdatera den i drift skulle det krävas två; ett som programmet laddas från och ett som det kan skriva till. Där byte av kodutrymme den laddar programmet från sker vid uppstart/omstart av enheten. Varianter av Netduino finns dock, till exempel Netduino Plus, med skrivbar minneskortplats där programmet kan spara kod som sedan kan laddas över till kodutrymmet vid en omstart.

3.5.2.5 Första-/Tredjepartsbibliotek

De förstapartsbibliotek som finns till Netduino är de som tillhandahålls av .NET micro framework. Microsoft har jobbat mycket med att lägga in stöd för hårdvara under arbetsnamnet Smart Personal Objects Technology (SPOT) [67]. Med detta har man lagt in stöd för olika sensortyper, till exempel OneWire och Inter-Integrated Circuit (I^2C), men även stöd för olika krypteringar och HTTP-bibliotek. Det finns dock ingen garanti för att Netduino har implementerat alla dessa bibliotek. På grund av begränsning i lagringsutrymmet på Netduino valde man att inte lägga med stöd för vissa sensorer, till exempel OneWire, då detta skulle minska utrymmet tillgängligt för att lagra kod.

Antalet tredjepartsbibliotek till Netduino är begränsat. Det finns förutom en lista på deras wiki ingen samlingsplats dit utvecklare kan skicka sina bibliotek så att de enkelt

kan hittas. En mindre samling av kodbibliotek finns dock på Codeplex, där användare har bidragit med drivrutiner till olika enheter, som exempelvis Liquid Crystal Display (LCD), och kontroller. Förutom drivrutiner finns där även olika hjälpbibliotek som enklare parsning av JSON och för att utföra beräkningar på vektorer [68].

3.5.3 Plattformens programmeringsspråk

Program för Netduino skrivs i programmeringsspråket C# (uttalat C-sharp) [69].

C# utvecklades kring Microsofts .NET, som är ett ramverk innehållande dels ett stort bibliotek med funktioner, men även definierar hur de olika .NET språken kan kommunicera mellan varandra [70]. På grund av det begränsade utrymmet på mikrokontrollerna kan inte hela .NET-ramverket inkluderas. Istället används ett minimerat ramverk benämnt .NET Micro Framework. Detta kräver inte ett operativsystem, utan kör istället direkt mot hårdvaran. Programbiblioteken har minskat både i storlek och i omfattning, detta kan betyda att hela klasser tagits bort, eller att vissa metoder tagits bort ur klasser. Även mer komplexa datastrukturer som vektorer i flera dimensioner har tagits bort.

.NET Micro Framework bygger dessutom på öppen källkod, vilket innebär att vem som helst kan skriva egna mikroprogram (firmware) att använda på hårdvaran. Med version 4.2 av ramverket kommer även stöd för programmeringsspråket Visual Basic .NET.

.NET Micro Framework kräver inte manuell minneshantering. Istället finns en så kallad skräphanterare som tar hand om objekt som saknar referenser från andra objekt och avallokerar minnesutrymmet de upptar. Stöd finns för att skriva flertrådade program och synkronisering av trådar kan antingen ske genom att låsa ute andra metoder från att exekvera en metod samtidigt, eller genom att låsa objekt.

3.5.3.1 Popularitet

Populariteten för C# får anses vara tämligen hög. Enligt en undersökning gjord av Tiobe, i mars 2014, ligger C# på femte plats, med ca 5,6% av marknaden [57]. Tiobe visar dock

en nedgång från föregående månad med -1,02%.

3.5.3.2 Community

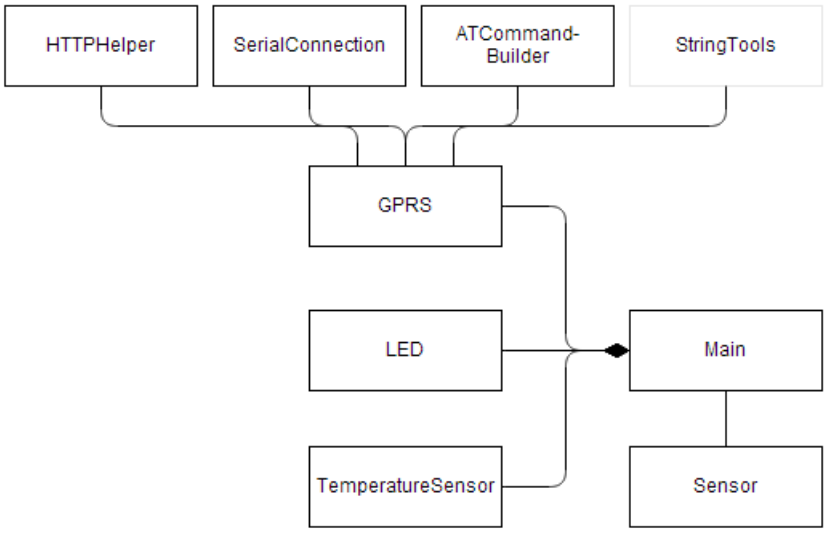
Med tanke på C#'s storlek finns det även en ganska stor community som omger språket och då .NET micro delar mycket funktionalitet med C# ger det en mer rättvis bild att utvärdera tillgängliga resurser för båda för att bilda sig en uppfattning om community-stöd. C# är väldigt flitigt diskuterat. På Stack Overflow skrivs i genomsnitt ungefär 610 nya ämnen varje dag relaterade till programspråket. För en fullständig redogörelse över undersökningens resultat beträffande popularitet se bilaga E.

3.6 Implementation

Implementationen till de tre olika mikrokontrollerna följer samma design och struktur. Detta gjordes för att underlätta jämförandet mellan plattformarnas implementation. Till Arduino krävs en del speciallösningar då standardbiblioteken till C++ saknas. Standardbiblioteken har ersatts med nedbantade bibliotek för att ta hänsyn till det begränsade lagringsutrymmet på Arduino. Den lilla mängden ramminne kräver extra noggrannhet gällande minneshantering.

3.6.1 Design

All design av mjukvara till de olika plattformarna har designats med objektorientering i åtanke. Figur 3.4 visar UML-diagrammet för den generella designen, som förklaras utförligare i detta avsnitt. Specifika UML-diagram för designen på de tre mikrokontrollerna hittas i bilaga C.



Figur 3.4: UML-diagram för den generella designen.

3.6.1.1 Main

Main initierar objekt och sätter konstanter, exempelvis domännamnet för webbtjänsten, som används av programmet. Main sköter programflödet genom att anropa metoder i de övriga klasserna.

3.6.1.2 Sensor

Denna klass sparar all nödvändig information om en sensor. Ett objekt av klassen Sensor innehåller dess gränsvärden, databasid och namn. Klassen innehåller även en klassmetod för att kunna instansiera ett Sensor-objekt utifrån en sträng som innehåller ett Json-objekt av en sensor.

3.6.1.3 TemperatureSensor

Klassen TemperatureSensor innehåller metoder som möjliggör digital avläsning av en temperatursensor. Klassen är specifikt anpassad för att läsa av temperatursensorn OneWire

DS18B20. Det är inte en underklass till klassen Sensor. Klassen innehåller metoder för att läsa av aktuell temperatur och returnera denna antingen i enheten Celsius, Fahrenheit eller milliCelsius.

3.6.1.4 Led

Klassen Led ansvarar för all styrning av lysdioder. Ett Led-objekt skapas för varje lysdiod. Objektet innehåller information om vilken GPIO-port lysdioden är kopplad till och metoder för att tända och släcka lysdioden.

3.6.1.5 GPRS

Klassen GPRS ansvarar för all kommunikation med GPRS-modulen. Objektet kommunicerar via den seriella porten med hjälp av klassen SerialConnection. Alla AT-kommandon som skickas till GPRS-modulen byggs med hjälp av klassen ATCommandBuilder. Det finns metoder för att skicka AT-kommandon, rapportera ett mätvärde, hämta intervallgränser, slå på GPRS-modulen och en metod för att rensa bort information som GPRS-modulen inkluderar i svaret från webbtjänsten.

3.6.1.6 HTTPHelper

Klassen HTTPHelper är en hjälpklass som är till för att bygga de textsträngar som används vid kommunikationen från mikrokontrollerna till webbtjänsten.

3.6.1.7 SerialConnection

Denna klass ansvarar för all kommunikation med den seriella porten på varje enhet, den bidrar därför med ett abstraktionslager mellan enheten och den fysiska kommunikationen. Klassen innehåller bland annat metoder för att läsa och skriva information från och till den seriella porten.

Vid utvecklingen av metoden för att läsa från den seriella porten har flera olika lösningar testats. Det går att läsa en hel sträng eller att läsa tecken för tecken. Genom att välja att läsa tecken för tecken har man möjlighet att avbryta inläsningen när förväntat svar lästs in. Det förväntade svaret varierar beroende på vilket AT-kommando som skickats till GPRS-modulen. Om man exempelvis skickar ”AT” till GPRS-modulen svarar GPRS-modulen ”AT OK” om allt gått väl. ”OK” är då nyckelordet som är det förväntade svaret. Att läsa in tecken för tecken, och leta efter nyckelord, är ett tillvägagångssätt som utnyttjas på BeagleBone Black och Arduino Nano trots att inläsning av den hela strängen hade blivit en enklare implementation på BeagleBone Black. På Arduino Nano var den enklaste implementationen att läsa tecken för tecken, på grund av den låga mängden ramminne, och direkt slänga information ointressant för den fortsatta programgången. På Netduino tillämpas en liknande metod, med skillnaden att innehållet i enhetens seriella buffert läses in på en gång istället för ett tecken åt gången.

Det finns även ett timeout-värde som begränsar tiden en mikrokontroller kan vänta på förväntat svar från GPRS-modulen. Om timeout-värdet nås har sannolikt något gått fel vid kommunikationen mellan mikrokontrollern och GPRS-modulen och GPRS-modulen inte svarat med det förväntade svaret eller inte svarat alls.

3.6.1.8 ATCommandBuilder

Denna klass består av metoder som innehåller och bygger alla AT-kommandon som mikrokontrollerna använder för att kommunicera med GPRS-modulen.

3.6.1.9 StringTools

Denna klass är specifik för BeagleBone och Arduino Nano då String-biblioteket i C# redan innehåller all denna funktionalitet. Den används för att ta bort överflödiga tecken, eller leta upp specifika tecken, i en sträng.

3.7 Realtidsuppdaterad webbsida

Som option till det praktiska jämförelsefallet i utvärderingen fanns ett önskemål om en webbsida med en graf över aktuell temperatur som i realtid uppdateras med nya mätningar allteftersom de anländer till webbtjänsten (se Figur 3.5). Tekniken som valdes för att åstadkomma realtidsuppdatering är SignalR. SignalR är ett relativt nytt bibliotek till ASP.NET utvecklat av Microsoft för att förenkla realtidskommunikation mellan en webbtjänst och dess klienter [71].

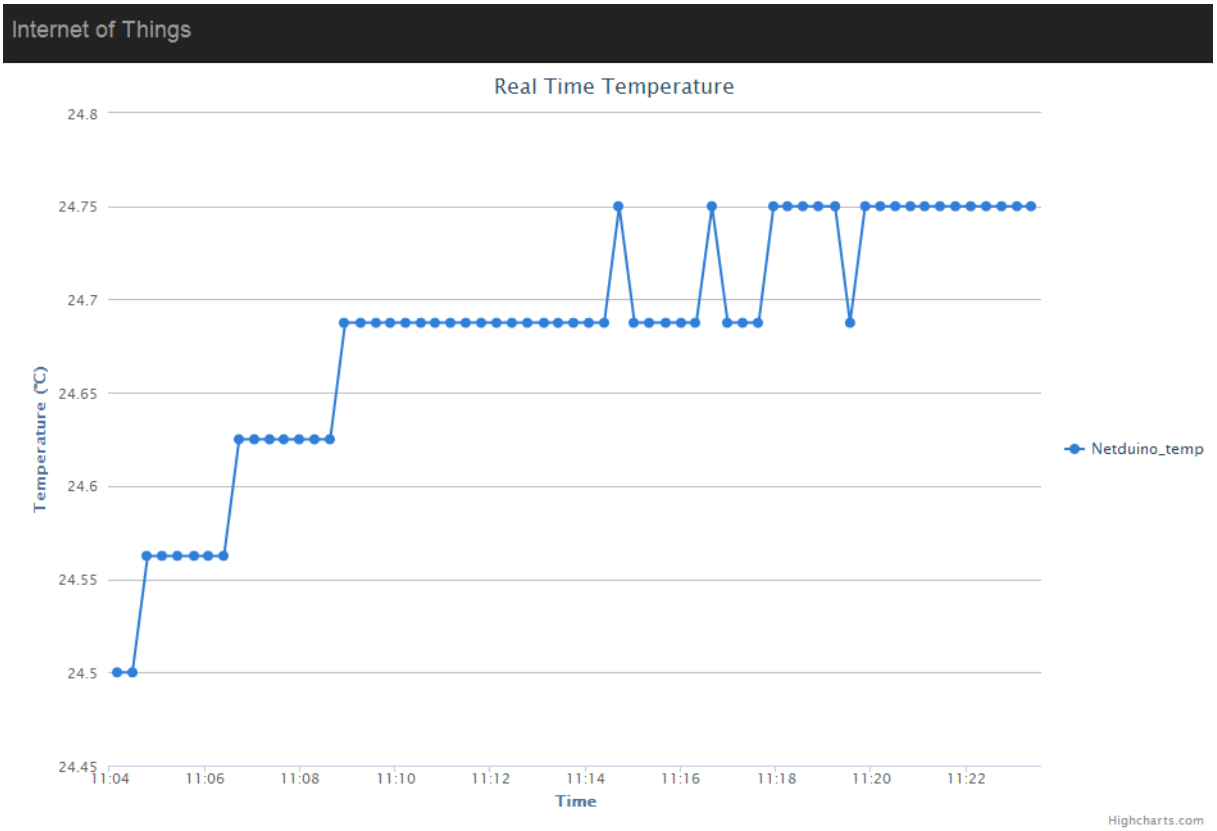
Det övergripande kommunikationförfarandet är att servern som kör webbtjänsten är den centrala kommunikationspunkten som all kommunikation går genom, även kommunikation initierad av klienter med annan klient som mål. Klienter har möjlighet att kalla på funktioner hos webbtjänsten, och webbtjänsten kan i sin tur kalla på funktioner hos klienten.

Klientfunktionerna och stödbiblioteket för SignalR hos klienten skrivs i Javascript, vilket innebär att stödet för varierande klientplattformar är stort. Kommunikationen mellan klienten och webbtjänsten sker i första hand med Hyper Text Markup Language 5 (HTML5) tekniken WebSocket [72]. För att åstadkomma kompatibilitet med klienter som saknar stöd för HTML5 finns ett antal fallback-tekniker som exempelvis EventSource eller ForeverFrame [73].

3.7.1 Implementation

För att implementera stödet för SignalR på servern skapades en Hub, som kommunikation med klienter sker via. I hubben går det sedan att implementera funktioner som kan kallas på av klienterna, något som inte använts i detta fall.

På klientsidan skapades en enkel HTML-sida som inkluderade Javascript-biblioteket Microsoft skapat. Detta bibliotek inkluderar funktioner för att sätta upp uppkopplingar till servern. En funktion definierades som kan kallas på av webbtjänsten och som tar emot



Figur 3.5: Webbsida med graf över aktuell temperatur

ett mätvärde och lägger in det i grafen. För att rendera grafen används verktyget Highcharts [74].

3.8 Sammanfattning

I detta kapitel har de tre mikrokontrollerna, Arduino Nano, BeagleBone Black och Netduino utvärderats med avseende på plattform, programmeringsspråk och utvecklingsverkyg. Designen av webbtjänsten och databasen, och en överblick av programstrukturen till rapporteringstjänsten, har presenterats. Utöver detta har även webbsidan som uppdateras i realtid med aktuella mätvärden beskrivits.

Kapitel 4

Resultat

4.1 Introduktion

I detta kapitel kommer resultatet av studien att presenteras. De tre mikrokontrollerna kommer att ställas emot varandra och deras för- och nackdelar kommer att diskuteras med avseende på plattform, utvecklingsmiljö och programmeringsspråk. Tiden det tog att utveckla jämförelsefallet på de tre mikrokontrollerna och programkodens storlek i rader och bytes kommer att jämföras.

4.2 Mikrokontroller

Detta avsnitt avser att jämföra de olika mikrokontrollerna i enlighet med den undersökning som gjorts av dem i avsnitt 3.3 för Arduino, avsnitt 3.4 för BeagleBone Black samt avsnitt 3.5 för Netduino.

4.2.1 Jämförelsetabell

Tabell 4.1 ger en överskådlig jämförelse av de olika plattformarnas egenskaper. Kodutrymme ledigt för utvecklaren beskriver den mängd utrymme utvecklaren har för att lagra sin

kod på plattformen när den befinner sig i sitt standardutförande, det vill säga utan någon användarkod lagrad. Minnesmängden ledigt för utvecklaren anger hur mycket ramminne utvecklaren har att nyttja när plattformen är uppstartad.

	Arduino	**BeagleBone Black**	**Netduino**
Kodutrymme ledigt för utvecklaren (kB)	30	253352	127,23
Minnesmängd ledigt för utvecklaren (kB)	1,79	312100	60,27
Utvecklingsmiljö	Arduino IDE, Visual Studio	Cloud9, Visual Studio, Eclipse	Visual Studio
Operativsystem	-	Linux	-
Programmeringsspråk	C/C++	Alla som går att köra under Linux	C#/Visual Basic .NET

Tabell 4.1: Jämförelsetabell

4.2.2 Utvecklingsmiljö

Alla mikrokontroller i studien har stöd för utveckling i Visual Studio, men i varierande grad. Den plattform som är bäst integrerad i Visual Studio är Netduino, som tillåter både utveckling, uppladdning och felsökning av program. Felsökningsmöjligheterna är därutöver många, och innehåller brytpunkter, variabelinspektion och villkorsbestämda brytpunkter. BeagleBone Black och Arduino Nano har i princip samma möjligheter, men kräver ytterligare investeringskostnader, i form av VisualGDB för BeagleBone Black och Visual Micro för Arduino Nano, för att det skall vara möjligt.

Vill man ladda upp programkod till BeagleBone Black utan att köpa tillägget VisualGDB får man antingen byta utvecklingsmiljö och använda Eclipse IDE, eller liknande utvecklingsmiljöer som tillåter fjärruppladdning till Linuxsystem [75]. Utan en sådan IDE kan man använda sig av en klient som använder protokollet Secure Copy (SCP) för att ladda upp programkoden till enheten och sedan, genom en anslutning med protokollet Secure

Shell (SSH), kompilera och starta programmet [76] [77].

4.2.2.1 Producentstöd

I denna studie har Visual Studio använts till alla plattformar. Producentstödet för utvecklingsmiljön får då anses likvärdigt på alla plattformar.

4.2.3 Plattform

Arduino Nano och Netduino saknar operativsystem. De har bägge en bootloader lagrat i minnet som vid uppstart startar det kompilerade programmet som finns lagrat på enheten. BeagleBone Black har ett fullstort operativsystem installerat. Det ger en möjlighet att köra flera program samtidigt, en möjlighet som inte finns på Arduino Nano eller Netduino. Användaren har även möjlighet att välja bland flera olika Linuxdistributioner på BeagleBone Black, och på så sätt skräddarsy operativsystemet efter behov. Även detta är något som inte är möjligt på Arduino Nano och Netduino.

4.2.3.1 Producentstöd

De tre plattformarna, som använts i studien, släpps i nya revisioner med jämna mellanrum. De nya revisionerna syftar till att lösa buggar i tidigare utgåvor av mikrokontrollerna, eller, som i Netduinos fall för att säkerställa att minnesutrymme finns för att kunna implementera nyare versioner av ramverket .NET Micro Framework fullständigt.

Det finns en uppsjö av hårdvara att koppla till mikrokontrollerna. Mycket av hårdvaran är anpassad för Arduino, som är den största producenten, men oftast är hårdvaran kompatibel med andra enheter också. Det kanske inte går att fästa hårdvaran direkt på mikrokontrollern, men det går alltid att använda sladdar och koppla manuellt. Det är viktigt att läsa specifikationen om hårdvaran då inte alla mikrokontroller använder samma spänning på GPIO-portarna.

4.2.3.2 Dokumentation

De tre enheterna har alla en egen wiki där både användare och medarbetare kan bidra med information. Man kan till alla tre enheter, genom deras respektive hemsidor, lätt få information beträffande hårdvaran och GPIO-portar.

När det gäller exempelkod för de tre enheterna har Arduino bra dokumentationsidor som beskriver kodbibliotek, klasser och metoder. Dokumentationssidorna kan man enkelt komma åt genom den officiella hemsidan. Till BeagleBone Black finns det ett bra utbud av exempel, men många av exemplen använder det egna programmeringsspråket BoneScript. Exempel i andra programspråk ges ej. Netduino har ett fåtal exempel att tillgå via den officiella hemsidan. Däremot finns det videogenomgångar, som tar användaren genom exemplen steg för steg, för de få exempel som finns.

Dokumentationen av hårdvaran på respektive plattform är i stort sett lika. Det finns ett kopplingsschema för alla komponenter på kortet, översikt över portar och hårdvaruinformation. Utöver detta erbjuder de alla schema-filer som man kan använda för att designa kretsar i OrCAD eller CadSoft EAGLE. Många av dokumenten, på samtliga plattformar, har dock upplevts bristande när det kommit till mer grundläggande saker. Ett exempel på detta är polariteten på laddkontakten som endast var specificerad på Netduino.

4.2.3.3 Möjlighet att uppdatera i drift

I standardutförande är det endast BeagleBone Black som kan uppdateras under drift utan omstart av enheten. Det finns även möjlighet att uppdatera programvaran på BeagleBone Black över Internet, förutsatt att Internetuppkoppling finns anslutet till enheten, genom att ansluta via SSH. Det möjliggör att man kan ladda upp ny programkod, kompilera och starta det nya programmet utan att ha fysisk tillgång till enheten. Vill man inte ha BeagleBone Black ansluten till Internet kan man genom en USB-anslutning fjärrstyra och uppdatera programvaran. Båda alternativen kräver ett litet avbrott när man byter från det gamla programmet till det nya, men ingen omstart krävs.

För att uppdatera Arduino Nano eller Netduino, i standardutförande, krävs att man har fysisk tillgång till enheten och är beredd att starta om enheten. En omstart av Arduino Nano eller Netduino går dock betydligt snabbare än en omstart av BeagleBone Black som ska ladda in ett helt operativsystem vid uppstart.

På BeagleBone Black, som kan köra flera program samtidigt, innebär det fördelar att kunna uppdatera programvara utan omstart av mikrokontrollern. Att inte behöva starta om mikrokontrollern innebär att övriga program kan fortsätta under uppdateringsprocessen. Endast det berörda programmet behöver startas om.

4.2.3.4 Första-/Tredjepartsbibliotek

Det finns flertalet förstapartsbibliotek till alla tre enheter. På Arduino kan man enkelt använda förstapartsbiblioteken i projekt genom att inkludera dem. BeagleBone, som kör fullstor Linuxkärna, har tillgång till alla förstapartsbibliotek för C/C++ och även där är det enkelt att inkludera bibliotekten. Till Netduino, som använder sig av Microsoft's .NET Micro Framework, finns också flertalet förstapartsbibliotek. Nackdelen på Netduino är dock att det inte finns någon garanti att alla dessa förstapartsbibliotek är implementerade i mjukvaran. På grund av utrymmesbristen har man på Netduino valt att inte inkludera stöd för vissa sensorer, exempelvis OneWire, direkt i mjukvaran. Det innebär att även om Microsoft har inkluderat ett stöd för en sensor i .NET Micro Framework så kanske det inte finns stöd för sensorn i mjukvaran på Netduino. Det kan betyda att man måste använda sig av tredjepartsbibliotek till Netduino trots att det egentligen ska finnas stöd i förstapartsbiblioteket.

Arduino har samlat flertalet tredjepartsbibliotek på den officiella hemsidan. De är kategoriserade efter användningsområde och är enkla att lokalisera. Till BeagleBone Black finns också ett stort utbud av tredjepartsbibliotek. BeagleBone saknar däremot den enkla strukturen, som Arduino har, för att hitta dessa bibliotek på dess officiella hemsida. Netduino har ett lite mer begränsat antal tredjepartsbibliotek. På dess wiki kan man hitta en

lista över de tredjepartsbibliotek som stöds, men det finns ingen officiell samlingsplats dit utvecklare kan skicka sina bibliotek och på så vis få en samlad samling bibliotek.

4.2.4 Programmeringsspråk

Den största skillnaden mellan de tre mikrokontrollerna, BeagleBone Black, Arduino Nano och Netduino, är att man till BeagleBone Black har möjlighet att fritt välja vilket programmeringsspråk man vill så länge det stöds i Linux. Det är möjligt då BeagleBone Black har ett fullstort Linuxoperativsystem. Vid programutveckling till Arduino Nano är man låst till C/C++ och vid programutveckling till Netduino är man låst till C# eller Visual Basic .NET. På grund av utrymmesbristen, framförallt på Arduino Nano, är man även betydligt mer begränsad till vilka bibliotek man kan inkludera på Arduino Nano och Netduino jämfört med BeagleBone Black. Både BeagleBone Black och Netduino kan exekvera flertrådade program, en möjlighet som Arduino Nano saknar. Med ett tredjepartsbibliotek till Arduino kan dock utvecklaren simulera trådning genom att schemalägga uppgifter.

4.2.5 Popularitet

Populariteten för en plattform kan vara av intresse av flera skäl. En populär plattform är det lättare att få hjälp med vid problem. Till en populär plattform finns det även större chans att hitta komponenter och sensorer. Slutligen är det större sannolikhet att utvecklare bakom plattformen lägger mer intresse vid att hålla den uppdaterad om det är många som använder den.

Den överlägset populäraste plattformen i denna utvärdering är Arduino. Detta märks på diskussionsforumet för Arduino där det skrivs sextio gånger fler inlägg, jämfört med diskussionsforumet för Netduino. Det märks även på Stack Overflow där det i genomsnitt skrivs nio gånger fler inlägg per dag om plattformen än vad det skrivs om Netduino eller BeagleBone Black (se bilaga E för undersökningen).

4.3 Implementationsjämförelse

Detta avsnitt kommer att jämföra de olika mikrokontrollerna baserat på vad som krävdes för att implementera jämförelsefallet, beskrivit i avsnitt 2.2.1, på de tre mikrokontrollerna, Arduino Nano, BeagleBone Black samt Netduino. Bilaga D innehåller en förteckning över alla tredjepartsbibliotek som har använts.

4.3.1 Externa enheter

Funktionalitet till komponenter som har implementerats i denna studie är tändning och släckning av LED, avläsning av en digital temperatursensor samt seriell kommunikation med en GPRS-modul.

4.3.1.1 LED

Både Netduino och Arduino har inbyggt stöd för kommunikation med de digitala portarna på respektive enhet. I Arduinos fall handlar det om att kalla på en externt definierad funktion, digitalWrite, och ge den argument för vilken port man vill skriva till och om man vill skriva en etta eller en nolla. I Netduinos fall måste man först skapa ett objekt; antingen av typen OutputPort för skrivning till, eller InputPort för läsning från porten. Därefter använder man objektens metod för Write, respektive Read, för att skriva till eller läsa från en port.

I BeagleBone Blacks fall måste man, om man inte vill läsa från och skriva till filträdet direkt, använda externa bibliotek för att läsa från och skriva till de digitala portarna. Att hitta biblioteket, installera och testa det tog en del tid, men ansågs motiverat genom att det underlättar framtida användning av de digitala portarna.

4.3.1.2 Temperatursensor

Ingen av mikrokontrollerna har inbyggt stöd för den temperatursensor, OneWire DS18B20, som använts i utvärderingen. Arduino krävde en nedladdning av ett tredjepartsbibliotek, Dallas Temperature, för att kunna läsa av temperaturen från sensorn. Biblioteket gjorde det sedan enkelt att läsa av temperaturen då det räckte med att initialisera ett objekt av klassen DallasTemperature och kalla på metoden getTempCByIndex för att läsa av temperaturen i Celsius. Genom att specificera index möjliggör biblioteket läsning av flera sensorer inkopplade på samma port.

I Netduinos fall krävde det ett firmwarebyte för att kunna kommunicera med OneWire-sensorn. .NET Micro 4.2 har stöd för protokollet, men av platsbrist valde Netduino att inte inkludera detta stöd. Den firmware som installerades var inte officiellt stödd, utan är skriven av en tredje part och befinner sig i en alfa-version. Med den anpassade firmware installerad ger det tillgång till klassen OneWire, som ämnar att förenkla avläsning av temperaturen. Innan avläsning av temperatur kräver den en hel del funktionsanrop som skriver olika bytevärden till den för att möjliggöra avläsning. Netduino har sagt, i augusti 2011, att stöd för OneWire skall implementeras direkt i original-firmware. Detta har, i mars 2014, ännu inte skett [78].

BeagleBone Black kräver att man installerar en särskild kärnmodul till Linuxkärnan för att man skall kunna läsa av temperaturen. Då kärnmoduler kan innebära stora säkerhetsrisker, och det gäller att lita på leverantören av dessa, är det bra att Texas Instruments, producenten bakom plattformen, skrivit kärnmodulen. Med kärnmodulen inläst måste man för varje omstart av mikrokontrollern köra ett kommando för att aktivera den, innan man kan läsa av temperaturer från sensorn. För att läsa av temperatur öppnar man sedan en fil under `/sys/bus/w1/devices/`, där ett fält i filen anger aktuell temperatur.

4.3.1.3 GPRS-modul

Alla mikrokontroller har hårdvarustöd för seriell kommunikation på en eller flera digitala portar. I Arduinos fall finns det en global instans av en klass, Serial, som används för att skriva till och läsa från denna port. Denna stödjer ett antal olika operationer för både läsning och skrivning. Bland annat kan man läsa en byte i taget, läsa en hel sträng, eller läsa ett specificerat antal bytes. Skrivning kan antingen göras bytevis eller skrivning av en sträng i en operation. Den seriella bufferten på Arduino kan lagra 64 bytes. Om man inte läser från bufferten i tillräckligt hög takt så kastas all information. Arduino har, förutom sin hårdvarustödda seriella port, ett bibliotek för att möjliggöra användning av vilka digitala portar som helst för seriell kommunikation. Detta bibliotek har dock visat sig ha en hel del brister. Bland annat klarar Arduino Nano inte av att läsa eller skriva i högre hastighet (baud rate) än 57600 när biblioteket används. Förutom detta är det inte möjligt att, med det seriella biblioteket, läsa in längre meddelanden än vad den bufferten kan lagra, oavsett om man tömmer denna allt eftersom.

Netduino är utrustad med två hårdvarustödda seriella portar. Dessa aktiverar man genom att skapa en instans av klassen SerialPort, där användaren får specificera vilken som skall användas. Läsning och skrivning kan göras antingen bytevis, eller genom att specificera det antal bytes man vill läsa eller skriva direkt. Den seriella bufferten på Netduino kan lagra 512 bytes. Till skillnad från Arduino finns det inga bibliotek till Netduino för att kunna använda valfria digitala portar för seriell kommunikation.

BeagleBone Black har sex stycken seriella portar, men en av dem kan endast användas för att skicka data. Som standard är endast en av de seriella portarna aktiverade, men att aktivera de övriga kräver inte mer arbete än att ändra i en konfigurationsfil. Seriell kommunikation via dessa portar kan ske genom att antingen skriva till och läsa från respektive fil i `/dev/ttyO` direkt, eller att använda sig av ett bibliotek, som till exempel LibSerial som abstraktionslager [79]. LibSerial gör det möjligt att kommunicera seriellt med ett antal olika operationer. Användaren kan välja att läsa och skriva bytevis eller ett antal specifi-

cerade bytes omgående. Biblioteket tillåter även användaren att ställa in baud-hastighet och stoppbitar.

4.3.2 Allmänt

De olika plattformarna skiljer sig en hel del åt, som tidigare beskrivits. Även om jämförelsefallet programmerats i C++ på både Arduino Nano och Beaglebone Black har de implementationsmässigt skiljt sig åt en hel del. Arduino Nano saknar alla C++ standardbibliotek, och har istället definierat en del egna. Detta gör att exempelvis `std::string` inte har kunnat användas, utan Arduinos egendefinierade String-klass har använts istället.

Särskild hänsyn har även varit tvungen att tas till det kraftigt begränsade arbetsminnet på Arduino Nano. Ett resultat av detta är att minneshantering har skötts manuellt, genom att allokera och avallokera objekt på heapen. Detta har framförallt gällt de olika strängar som har använts för att skapa HTTP-meddelanden och för att läsa in svar från GPRS-modulen. Minnesutrymmet är till exempel inte tillräckligt för att både hålla en HTTP-förfrågan och dess svar från webbtjänsten i minnet samtidigt. I BeagleBone Black har istället standardbiblioteket kunnat användas för att hantera strängar, och dessa har allokerats på stacken.

Netduino, som programmerats i C# har ingen minneshantering behövts att ta hänsyn till, då .NET Micro Framework har en skräphanterare som med jämna mellanrum frigör det minne som allokerats av objekt som det inte finns några referenser till.

4.3.3 Kodmängd

Tabell 4.2 visar hur mycket kod som skrivits till de tre mikrokontrollerna. Både Arduino och BeagleBone Black har skrivits i C++, som använder sig av header-filer vilket C# saknar och därav utelämnats. För att jämförelsen skall bli så rättvis som möjligt har koden formaterats på ett likvärdigt sätt. Antalet rader innefattar inte den kod som finns i externa bibliotek som använts i lösningen, utan enbart egenskriven kod.

Den andra raden i tabellen visar storleken på den kompilerade koden i kilobyte på respektive enhet. För alla enheter gäller att koden är kompilerad med standardinställningar, det vill säga kompilatorn har inte instruerats till att försöka göra ytterligare effektivisering beträffande storlek. Den kompilerade koden innefattar även de bibliotek som har kompilerats och länkats.

Den tredje raden visar hur stor del av det tillgängliga kodutrymmet, i procent, som den kompilerade koden tar upp.

I denna tabell kan man se att den kompilerade koden till BeagleBone Black tar mycket mer utrymme jämfört med Arduino Nano och Netduino, men när man ställer det i relation till hur stor andel den tar av det tillgängliga gör den minst avtryck. BeagleBone Black är den implementation som använder sig av flest tredjepartsbibliotek, vilket påverkar storleken på den kompilerade koden.

	Arduino	**BeagleBone Black**	**Netduino**
Antal rader	749	802	521
Antal rader i header-filer	183	198	-
Storlek kompilerad kod (kB)	17,2	266	9,1
Mängd av ledigt kodutrymme (%)	57,3	0,09	7,2

Tabell 4.2: Jämförelsetabell kodmängd

4.4 Tidsjämförelse

Detta avsnitt avser att jämföra tiden det tog att utveckla jämförelsefallet på de olika mikrokontrollerna. Avsnittet kan ses som en sammanfattning där alla tidigare diskuterade punkter som till exempel dokumentation och kodbibliotek vägs in.

4.4.1 Jämförelsetabell

Tabell 4.3 visar tiden det tog att utveckla jämförelsefallet beskrivet i avsnitt 2.2.1. Kolumnen test innefattar testning och felsökning av programvaran, kolumnen utveckling innefattar tiden det tog att designa och implementera lösningen på plattformen och kolumnen uppkoppling innefattar tiden det tog att fysiskt koppla in sensorer och liknande till mikrokontrollern, men även tiden det tog att leta och läsa hårdvaruspecifikationer för mikrokontrollerna.

Tiden att implementera LED på Arduino och Netduino är ganska lika. Det tog dock nästan dubbelt så lång tid i jämförelse att implementera dem på BeagleBone Black. Detta berodde på, som beskrivits i avsnitt 4.3.1.1, att ett tredjepartsbibliotek användes för att senare underlätta vid läsning från och skrivning till de digitala portarna.

Att implementera temperatursensorn tog ungefär lika lång tid på alla mikrokontroller. Ingen av dem hade stöd för den från början, utan krävde antingen ett tredjepartsbibliotek som i Arduinos fall, en kernel-modul som i BeagleBone Blacks fall eller en firmware-installation i Netduinos fall. Att installera bibliotek är klart mindre omfattande än en firmware-installation, men Arduino implementerades först, vilket innebar att kunskapen om temperatursensorn var högre när implementationen av den till Netduino och BeagleBone Black påbörjades.

GPRS-modulen är den komponent som tog mest tid att implementera. Detta berodde på att kommunikationen med GPRS-modulen är mer omfattande då meddelanden skickas, seriellt, i båda riktningar. Den mikrokontroller som implementerades först var i detta fall Netduino, vilket innebär att den tiden inte är helt sanningsenlig då lärdomar drogs från implementationen av Netduino som var användbara vid implementationen av Arduino och BeagleBone Black. Arduino är den mikrokontroller som tog överlägset mest tid att implementera fullständig GPRS-funktionalitet på. Detta beror till stor del på den låga mängd ramminne mikrokontrollern har, vilket innebar att det snabbt fylldes. När ramminnet är fullständigt allokerat leder det till oväntad exekveringsordning eller omstart av mikrokon-

trollern. Detta, i kombination med de begränsade felsökningsmöjligheterna beskrivna i avsnitt 3.3.1.2, gjorde implementationen tidsödande.

Mikrokontroller	**Tid (timmar)**			
	Test	Utveckling	Uppkoppling	$\sum$
Arduino				
- LED	1	2	1	4
- Temperatur	4	5	3	12
- GPRS	42,5	27	9	78,5
Totalt	47,5	34	13	94,5
BeagleBone Black				
- LED	3	4	1	8
- Temperatur	3	5	2	10
- GPRS	13	5,5	2,5	23,5
Totalt	19	14,5	5,5	41,5
Netduino				
- LED	1,5	2	1	4,5
- Temperatur	3,5	7	2	12,5
- GPRS	20	13	2,5	35,5
Totalt	25	22	5,5	52,5

Tabell 4.3: Jämförelsetabell tidsåtgång

4.5 Sammanfattning

I detta kapitel har en jämförelse av de olika mikrokontrollerna Arduino Nano, BeagleBone Black och Netduino enligt de punkter som sattes upp i avsnitt 2.2 gjorts, utgående från den undersökning som gjordes av mikrokontrollerna var för sig i kapitel 3. Skillnader i implementationen för det praktiska jämförelsefallet beskrivet i avsnitt 2.2.1 har diskuterats, där bland annat stödet för de olika hårdvarukomponenterna beskrivits samt hur kommunikationsförfarande med dem fungerar på de olika plattformarna.

Kapitel 5

Slutsats

5.1 Projektsammanfattning

På väg mot Internet of Things, beskrivet i avsnitt 2.5, kommer mikrokontroller troligtvis att spela en stor roll. En anledning till detta är den kapacitet de har, trots de små dimensionerna. Detta arbete syftade därför till att utvärdera tre olika mikrokontroller, främst ur ett utvecklarperspektiv. Fokus har legat på de olika mikrokontrollernas plattform, utvecklingsmiljö och programmeringsspråk. För plattformarna har operativsystem, eller motsvarande, undersökts. Förutom detta har utvärdering även gjorts över producentstödet från tillverkaren, dokumentation och popularitet.

För att enklare bilda sig en uppfattning om mikrokontrollernas styrkor respektive svagheter har ett jämförelsefall utvecklats. Detta jämförelsefall bestod av att varje mikrokontroller läser av en temperatur med en temperatursensor, om den avlästa temperaturen är inom ett visst intervall tänds en grön lysdiod. Om mätvärdet är utanför intervallet tänds istället en röd lysdiod. Den avlästa temperaturen skickas till en webbtjänst som sparar mätvärdet i en databas. Intervallet hämtas från webbtjänsten när mikrokontrollen startas upp.

5.1.1 Plattform

Den mest populära plattformen är Arduino. Vid undersökning av tillverkarnas diskussionsforum var medlemsantalet för Arduino nästan tio gånger större än för den näst mest populära, Netduino, då undersökningen påbörjades (se bilaga E.5). Att denna popularitet även visar sig i aktiviteten på diskussionsforumet visar bilaga E.4.

Dokumentationen för de olika mikrokontrollerna vad gäller hårdvara är i stort sett likvärdig, där de olika in- och utgångarna beskrivs och övrig hårdvaruinformation som exempelvis klockfrekvens och arbetsminne.

5.1.2 Utvecklingsmiljö

Vad gäller utvecklingsmiljöerna har utvärdering gjorts dels beträffande den utvecklingsmiljö som tillverkarna av mikrokontrollerna tillhandahåller eller rekommenderar, dels för stödet för utveckling i Visual Studio. Olika funktioner för att underlätta utveckling som exempelvis kodkomplettering har undersökts [4]. Förutom själva utvecklingen har även kompilering och uppladdning av programkod på mikrokontrollerna samt vilka felsökningsmöjligheter som finns utvärderats.

Inga av de övriga utvecklingsmiljöerna överträffar Visual Studio när det gäller funktionalitet. Stödet för utveckling i Visual Studio är bäst på Netduino, detta gäller även uppladdning av kompilerad kod och felsökning. Med hjälp av tredjeparts-tillägg kan dock mycket av detta stöd även uppnås på både Arduino och BeagleBone Black.

5.1.3 Programmeringsspråk

De olika mikrokontrollerna stödjer olika programmeringsspråk. Två av de utvärderade, Arduino Nano och Netduino är låsta till ett fåtal språk; C/C++ respektive C#/Visual Basic.NET, medan BeagleBone Black stödjer alla programspråk som går att exekvera i ett Linuxsystem. Det praktiska jämförelsefallet har implementerats i C/C++ på Arduino

Nano och BeagleBone Black, på Netduino har det implementerats i C#. Undersökning av de olika programspråkens popularitet visar att C# är populärare än C++ på det populära diskussionsforumet Stack Overlow (se bilaga E.1), men mindre populärt i marknadsundersökningar [49] [57]. Undersökning har även gjorts av de tredjepartsbibliotek som finns för respektive plattform, exempelvis för kommunikation med olika typer av sensorer. Denna visar att Arduino har flest, och är även den plattform var tredjepartsbibliotek är bäst organiserade.

5.2 Utveckling av det praktiska jämförelsefallet

Den övergripande designen av programvaran till de tre mikrokontrollerna är densamma, men implementationsmässigt skiljer de sig åt. Designen har gjorts med objektorientering i åtanke.

5.2.1 Mikrokontroller

Netduino har en skräphanterare som periodiskt frigör minne allokerat av objekt som det inte längre finns några referenser till. Detta gör att utvecklaren inte behöver frigöra minne manuellt, även om möjligheten finns. På BeagleBone Black, där mycket ramminne finns att tillgå, allokeras alla objekt på stacken. Detta medför att minnesutrymmet de upptar automatiskt frigörs när funktioner avslutas, varför särskild minneshantering inte behöver utföras.

Att allokera objekt på stacken var inte möjligt vid utvecklingen till Arduino Nano, på grund av den lilla mängden ramminne, då funktioner måste kunna frigöra minne under tiden de körs för att fortsätta exekvera. Minneshanteringen sker istället genom att allokera utrymme för objekt på heapen och sedan spara minnesreferensen. När objektet inte längre behövs frigörs minnet genom manuell deallokering. Manuell minneshantering på detta sätt ställer högre krav på utvecklaren, då det gäller att frigöra allt minne som allokeras. Om

utvecklaren misslyckas med att deallokera objekt när de inte längre används leder detta till minnesläckage. Minnesläckage innebär att applikationen förbrukar mer och mer minne under tiden den exekverar och kraschar när det inte längre finns minne att allokera.

5.2.2 Webbtjänst & databas

Webbtjänsten är baserad på Microsofts Active Server Pages Web Application Programming Interface (ASP.NET Web API). För kommunikation mellan mikrokontrollern och webbtjänsten används HTTP-meddelanden. Webbtjänsten sparar mätvärden och sensorer i en databas av typen SQL Server, och mappning mellan tupler i databasen och kod i webbtjänsten har gjorts med hjälp av Entity Framework. Varje mätvärde är associerad med en sensor, och varje sensor är associerad med en enhet och en sensortyp. Detta innebär att en mikrokontroller med flera olika sensorer kopplade får rapportera för varje sensor individuellt.

5.3 Vidareutveckling

I detta avsnitt kommer potentiella vidareutvecklingar av systemet att tas upp. Bland annat kommer databasstruktur och anslutning av ytterligare sensorer att diskuteras.

5.3.1 Programkod

En möjlig vidareutveckling av programkoden skulle kunna vara att skapa gränssnitt till alla hårdvaruspecifika klasser [80]. Ett gränssnitt är ett kontrakt som tvingar utvecklaren att implementera metoder och funktionalitet i de klasser där gränssnitt används.

5.3.2 Databas

I dagsläget har varje tuple i sensortabellen två fasta gränsvärden. En vidareutveckling på databastrukturen skulle kunna vara att använda sig av värde-par vilket skulle ge användaren möjlighet till att sätta hur många, eller hur få, gränsvärden användaren behöver för varje sensor. Utökningen skulle kräva en extra tabell där namn på gränsvärde, id, sensorid och gränsvärde lagras. Nackdelen med denna lösning är att den ökar komplexiteten på implementationen eller så måste man veta i vilken ordning gränserna kommer i databasen för att fördelaktigt kunna nyttja implementationen.

5.3.3 Andra sensorer

Förutom temperatur skulle det även kunna vara intressant att mäta luftfuktigheten. Detta skulle kunna göras genom att koppla in en luftfuktighetssensor och kontinuerligt rapportera dess värde. Beroende på var mikrokontrollern skall användas skulle även positionen kunna vara intressant att rapportera. Detta skulle kunna göras genom att antingen koppla en separat GPS-modul till mikrokontrollern, eller genom att använda GPRS-modulen, som även har möjlighet att bestämma GPS-position.

5.3.4 Alternativa energikällor

I denna studie har en vanlig strömadapter använts för att driva mikrokontrollerna. Vid en mobil mikrokontroller skulle man även kunna ha batteridrift. Man skulle även kunna driva mikrokontrollern med hjälp av solpaneler. Det finns flera olika batterilösningar, med eller utan solpaneler, att köpa på Internet [81].

5.3.5 Hantering av felkällor

De flesta sensorer har felmarginaler. Detta gäller inte bara precisionen med vilka de levererar värden, utan även felkällor som resulterar i felaktiga mätvärden. En sådan har erfarits

i denna utvärdering. Temperatursensorn, eller mikrokontrollern som läser av det, levererar kraftigt avvikande värden vid slumpmässiga tillfällen för att därefter leverera värden som kan betraktas vara inom det ”normala” intervallet i långa perioder däremellan. Hur man skall hantera dessa felvärden beror på var mätningen sker, och vilka mätvärden man kan förvänta sig. Ett extremvärde åt något håll skulle kunna vara ett felaktigt mätvärde i ett fall och ignoreras, medan det i ett annat fall skall betraktas som något som kräver ett agerande. I denna utvärdering har ingen undersökning gjorts för att säkerställa vilken komponent som är felkällan, men kraftigt felaktiga mätvärden har levererats både från Netduino och BeagleBone Black.

Ett sätt att hantera detta på skulle kunna vara att se hur mycket mätvärdet avviker procentuellt från det senast rapporterade mätvärdet, och förkasta det om det avviker för mycket.

5.3.6 Åtkomstkontroll och säkerhet

I dagsläget är API’et utvecklat till det praktiska jämförelsefallet öppet för alla. Detta innebär att alla som vet adressen till webbtjänsten och kommunikationsprotokollet webbtjänsten använder kan skapa sensorer och lägga in mätvärden. Likaså kan utomstående lyssna av all trafik som skickas till och från tjänsten. Detta är inte alltid önskvärt, utan någon form av åtkomstkontroll skulle kunna implementeras för att förhindra utomstående att skriva till webbtjänsten. Med någon form av kryptering av kommunikationen skulle även avlyssning kunna förhindras. Att införa denna funktionalitet i webbtjänsten innebär dock att kraven och komplexiteten på mikrokontrollerna ökas för att de skall kunna nyttja webbtjänsten.

5.4 Andra användningsområden

Det är egentligen bara fantasin som sätter gränsen för vilka användningsområden som finns för en mikrokontroller. I detta avsnitt ges några exempel där mycket av utrustningen som använts i denna studie går att återanvända.

5.4.1 Rörlighet

Med hjälp av utrustningen som används i denna studie skulle man, med en extra GPS-antenn ansluten till GPRS-modulen, kunna skapa ett system som rapporterar position för en specifik lastbil under en transport eller rapportera vilken rutt en lastbil har tagit. Analyserar man sedan den rapporterade rutten skulle man kunna optimera vägvalet och på så vis minska bränsleförbrukningen och miljöpåverkan.

5.4.2 Övervakning

Med hjälp av en rörelsedetektor och en kamera-modul skulle man, med hjälp av utrustningen som använts i denna studie, enkelt kunna bygga ett system som tar bilder när rörelsedetektorn reagerat på rörelse och ladda upp dessa bilder till en webbtjänst. På hemsidan där GPRS-modulen är beskriven finns ett exempel där man kopplar in en kameramodul till GPRS-modulen och fotograferar [9]. Ansluter man en rörelsedetektor och använder exemplets kod för att fotografera skulle man kunna skapa ett program som fotograferar när rörelsedetektorn reagerat på rörelse och därefter, med hjälp av GPRS-modulen, ladda upp de nytagna bilderna till en webbtjänst. Man har då skapat ett enkelt system för hemövervakning.

5.4.3 Hemautomatisering

Det finns många olika tänkbara användningsområden där man skulle kunna ha mikrokontroller för att automatisera hemmet. Man skulle kunna ha en mikrokontroller som styr

alla lampor i hemmet så man kan slå av, eller på, olika lampor genom ett webbgränssnitt. Man skulle kunna styra lamporna så att det är tänt mellan vissa klockslag, eller varför inte ett system som använder rörelsedetektorer som tänder lampor när rörelse uppfattas. Man skulle också kunna skapa ett system som styr värmen i huset eller i sommarstugan. Genom ett webbgränssnitt skulle man kunna sätta på element i sommarstugan så att det är varmt vid ankomst. Man skulle också kunna ha temperatursensorer som slår på, eller av, element vid en viss temperatur. Antagligen är termostaterna i dagens element mer effektiva än att slå på och av element via en mikrokontroller, men det är möjligt!

5.5 Slutsatser

Detta arbete syftade till att utvärdera tre olika mikrokontroller. Det finns inget självklart val av mikrokontroller då alla har sina fördelar och nackdelar. Vissa har kraftfullare hårdvara medan andra har bättre stöd för olika sensorer. Detta är något att överväga vid val av mikrokontroller till en specifik implementation. Programmeringsspråket skiljer sig även åt mellan de tre, där två av enheterna endast kan programmeras i ett par programmeringsspråk, medan den tredje stödjer flera. En annan aspekt att ta hänsyn till är i vilken miljö utveckling av programvaran skall ske.

5.5.1 Tillämpningar med flera sensorer

Som beskrivits tidigare, och som även visas i jämförelsetabellen 4.1 har Arduino Nano begränsat minnesutrymme. Detta kommer att ställa till problem vid användande av flera sensorer, då kommunikation med dessa i regel tar konstant ramminne. Detta är något som upplevts även i jämförelsefallet, trots den förhållandevis lilla mängd sensorer som använts. Netduino, och framförallt BeagleBone Black har mer tilltaget ramminne vilket gör det möjligt att använda fler sensorer. Antalet analoga och digitala portar på de tre mikrokontrollerna är ungefär lika, varför skillnader i begränsningar inte ligger på antalet

portar i första hand.

Förutom begränsningar i ramminne har de tre mikrokontrollerna även olika begränsningar beträffande kodutrymme. Jämförelsefallet som implementerats i utvärderingen tar upp mer än halva kodutrymmet på Arduino Nano, medan det tar upp mindre än en tiondel på Netduino och inte ens en tusendel på BeagleBone Black. Detta är ytterligare ett tecken som talar för att Arduino Nano inte lämpar sig för större tillämpningar.

5.5.2 Tillämpningar med seriell kommunikation

Vid tillämpningar med mycket seriell kommunikation har både Netduino och BeagleBone Black en klar fördel över Arduino Nano. Netduino har två seriella portar och BeagleBone Black har fem. Detta skall jämföras med Arduino som endast har en port. Utvärdering har även visat att denna port även används för all kommunikation över USB, vilket innebär att denna port inte går att använda om man vill felsöka samtidigt.

5.5.3 Utvecklingstid

Sett till utvecklingstid kan man konstatera att Arduino Nano är nästan dubbelt så besvärlig att arbeta med jämfört med Netduino, som hamnar på andra plats, och mer än dubbelt så besvärlig som BeagleBone Black som tar hem första platsen. Anledningen till att Arduino Nano sticker ut såpass mycket tidsmässigt är på grund av den lilla mängden ramminne som finns att tillgå. Det har krävt ett helt nytt tankesätt där kontroll över utnyttjad minnesmängd blivit vitalt. Bristen på felsökningsmöjligheter på Arduino Nano har också varit ett problem som påverkat utvecklingstiden då tillägget Visual Micro, som finns att köpa, inte varit aktuellt för denna utvärdering.

Skillnaden i utvecklingstid mellan Netduino och BeagleBone Black är lite missvisande då den övergripande designen av programmet förfinades och korrigerades vid utvecklingen av jämförelsefallet på Netduino. Samma design användes senare till alla plattformar.

Många problem vid implementationen av kommunikationen med GPRS-modulen löstes vid utvecklandet av Netduinos jämförelsefall och hamnar således under Netduinos timmar.

5.5.4 Avslutande tankar

Vi har under arbetets gång känt att det varit svårt att jämföra de tre mikrokontrollerna likvärdigt. Som vi tidigare redogjort skiljer sig hårdvaran åt markant. Användningsområdet för mikrokontrollerna skiljer sig därmed åt en hel del. Det praktiska jämförelsefallet var tänkt att användas för att mer konkret upptäcka vilken plattform som var smidigast att utveckla till. Risken med ett sådant jämförelsefall är att det, omedvetet, gynnar någon av mikrokontrollerna. Jämförelsefallet som använts i studien har varit ganska begränsat, sett till kraven det ställer på mikrokontrollern, därför anser vi att det ger en tillräckligt rättvis uppfattning om mikrokontrollerna.

Studien har visat att den mikrokontroller vi har upplevt som mest besvärlig att utveckla till, Arduino Nano, ändock är den mest populära av de tre. Den uppfattning vi har fått av de tre mikrokontrollerna är att de nästan uteslutande används till olika hobbyprojekt, vilka ställer andra krav än vad exempelvis industriella applikationer kräver.

Implementationen av jämförelsefallet har vi upplevt gått smidigast på Netduino, detta trots att det i flera av fallen är den mikrokontroller som implementerats först. En stor del i detta ligger i felsökningsmöjligheterna som finns. BeagleBone Black har vid ett flertal tillfällen upplevts som något krånglig, och detta kanske har sin förklaring i att den är så mångsidig.

Referenser

[1] Steam. Steam hardware & software survey. http://store.steampowered.com/hwsurvey, Januari 2014.

[2] Various. Booting. http://en.wikipedia.org/wiki/Boot_loader#Modern_boot_loaders, Februari 2014.

[3] Various. .net micro framework. http://en.wikipedia.org/wiki/.NET_Micro_Framework, Juli 2013.

[4] Various. Autocomplete. http://en.wikipedia.org/wiki/Autocomplete#In_source_code_editors, Februari 2014.

[5] Brian Alexander Lee. On the importance of development-and-build environments. http://msdn.microsoft.com/en-us/library/bb896743.aspx, Juni 2007.

[6] Various. General packet radio service. http://en.wikipedia.org/wiki/General_Packet_Radio_Service, December 2013.

[7] Various. Coordinated universal time. http://en.wikipedia.org/wiki/Coordinated_Universal_Time, Februari 2014.

[8] Various. 1-wire. http://en.wikipedia.org/wiki/Onewire, April 2014.

[9] Cooking Hacks. 3g/gprs-shield for arduino. http://www.cooking-hacks.com/3g-gprs-shield-for-arduino-3g-gps, April 2014.

[10] Various. Serial communication. http://en.wikipedia.org/wiki/Serial_communication, April 2014.

[11] Various. Hayes command set. http://en.wikipedia.org/wiki/Hayes_command_set, April 2014.

[12] Various. Flight recorder. http://en.wikipedia.org/wiki/Flight_recorder, Februari 2014.

[13] Arduino. Arduino - arduinoboardnano. http://arduino.cc/en/Main/ArduinoBoardNano, Januari 2014.

[14] Various. Atmel avr. http://en.wikipedia.org/wiki/Atmel_AVR, April 2014.

[15] Various. Commodore 64. http://en.wikipedia.org/wiki/Commodore_64, Januari 2014.

[16] Beaglebone. Beaglebone - beaglebone black. http://beagleboard.org/Products/BeagleBone%20Black, November 2013.

[17] Various. Ångström distribution. http://en.wikipedia.org/wiki/%C3%85ngstr%C3%B6m_distribution, December 2013.

[18] Various. Arm architecture. http://en.wikipedia.org/wiki/ARM_architecture, April 2014.

[19] Various. Home theater pc. http://en.wikipedia.org/wiki/Home_theater_PC, Februari 2014.

[20] Netduino. Netduino - netduino 1. http://netduino.com/hardware/, 2013.

[21] Louis Coetzee and Johan Eksteen. The internet of things - promis for the future? an introduction. Technical report, IST Africa, 2011.

[22] Xu Li, Rongxing Lu, Xiaohui Liang, Xuemin (Sherman) Shen, Jiming Chen, and Xiaodong Lin. Smart community: An internet of things application. IEEE Communications Magazine, pages 68–75, November 2011.

[23] International Telecommunication Union. Itu internet reports the internet of things. Technical report, November 2005.

[24] Internet Society. When will ipv4 addresses actually run out? http://www.isoc.org/internet/issues/ipv6_faq.shtml#q7, 2 2014.

[25] The Internet Society. Internet protocol - internet header format. http://tools.ietf.org/html/rfc791#section-3.1, September 1981.

[26] The Internet Society. Internet protocol, version 6 (ipv6). https://tools.ietf.org/html/rfc2460, Januari 2014.

[27] Internet Society. So why has it taken so long for ipv6 to be implemented? http://www.isoc.org/internet/issues/ipv6_faq.shtml#q12, Februari 2014.

[28] Various. Network address translation. http://en.wikipedia.org/wiki/Network_address_translation, Maj 2014.

[29] Mirren Gidda. Edward snowden and the nsa files – timeline. http://www.theguardian.com/world/2013/jun/23/edward-snowden-nsa-files-timeline, Juli 2013.

[30] Rolf H. Weber. Internet of things - governance quo vadis? Computer Law & Security Review, (29):341–347, 2013.

[31] SPM Instrument AB. Spm inom pappers- och massaindustrin. http://www.spminstrument.se/sv/Losningar/Industrier/Papper-och-massa/, Februari 2014.

[32] The Raspberry Pi Foundation. http://www.raspberrypi.org, Februari 2014.

[33] Xbian. http://www.xbian.org, Februari 2014.

[34] Chris Baume. Beer monitoring with my raspberry pi. http://chrisbaume.wordpress.com/2013/02/10/beer-monitoring/, Februari 2013.

[35] Paul J Stoffregen and Robin C Coon. Teensy usb development board. https://www.pjrc.com/teensy/, Februari 2014.

[36] Various. Input/output. http://en.wikipedia.org/wiki/Input/output, Januari 2014.

[37] Various. Pulse-width modulation. http://en.wikipedia.org/wiki/Pulse-width_modulation, Februari 2014.

[38] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext transfer protocol – http/1.1. https://tools.ietf.org/html/rfc2616, Juni 1999.

[39] Douglas Crockford. The application/json media type for javascript object notation (json). https://tools.ietf.org/html/rfc4627, Juli 2006.

[40] Microsoft. Entity framework. http://msdn.microsoft.com/en-us/data/ef.aspx, April 2014.

[41] Various. Object-relational mapping. http://en.wikipedia.org/wiki/Object-relational_mapping, Maj 2014.

[42] Various. Microsoft sql server. http://en.wikipedia.org/wiki/Microsoft_SQL_Server, Mars 2014.

[43] Various. Integrated development environment. http://en.wikipedia.org/wiki/Integrated_development_environment, Februari 2014.

[44] Various. Wiring (development platform). http://en.wikipedia.org/wiki/Wiring_%28development_platform%29, Augusti 2013.

[45] Various. Processing (programming language). http://en.wikipedia.org/wiki/Processing_%28programming_language%29, Februari 2014.

[46] Visual Micro. Visual micro homepage. http://www.visualmicro.com, Februari 2014.

[47] Arduino. Softwareserial library. http://www.arduino.cc/en/Reference/SoftwareSerial, Februari 2014.

[48] Various. Baud. http://en.wikipedia.org/wiki/Baud, Februari 2014.

[49] Stack Overflow. Stack overflow. http://stackoverflow.com/, Mars 2014.

[50] Arduino. Arduino - homepage. http://www.arduino.cc, Februari 2014.

[51] Arduino. The arduino playground. http://playground.arduino.cc, Februari 2014.

[52] Freetronics. arduino-tftpboot. https://github.com/freetronics/arduino-tftpboot, Februari 2014.

[53] Freetronics. How to upload a sketch to your arduino via a network. http://www.freetronics.com/pages/how-to-upload-a-sketch-to-your-arduino-via-a-network#.UwthEYWa8-U, Februari 2014.

[54] Various. Liquid-crystal display. http://en.wikipedia.org/wiki/Lcd, April 2014.

[55] Various. Thin-film-transistor liquid-crystal display. http://en.wikipedia.org/wiki/TFT_LCD, April 2014.

[56] Ivan Seidel. Arduinothread. https://github.com/ivanseidel/ArduinoThread, April 2014.

[57] Tiobe. Tiobe index for march 2014. http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html, Mars 2014.

[58] Various. Board revisions and changes. http://elinux.org/Beagleboard:BeagleBoneBlack#Board_Revisions_and_Changes, Mars 2014.

[59] Beagleboard. Beagleboard.org - bonescript library. http://beagleboard.org/Support/BoneScript, Mars 2014.

[60] Node.js. node.js. http://nodejs.org/, April 2014.

[61] Cloud 9. Cloud9 ide | your code anywhere, anytime. http://c9.io, Mars 2014.

[62] Sysprogs. Visualgdb - integrate gcc and gdb in visual studio. http://visualgdb.com/, 2014.

[63] The GNU Project. Gdb: The gnu project debugger. http://www.sourceware.org/gdb/, Mars 2014.

[64] Various. Microframeworksdk-mono. https://github.com/secretlabs/MicroFrameworkSDK-Mono, Februari 2014.

[65] Netduino. Netduino: Tech specs. http://www.netduino.com/netduino/specs.htm, Februari 2014.

[66] CadSoft EAGLE. Cadsoft eagle pcb design software. http://www.cadsoftusa.com/eagle-pcb-design-software/?language=en, Mars 2014.

[67] Microsoft. Microsoft.spot namespace. http://msdn.microsoft.com/en-us/library/microsoft.spot(v=vs.102).aspx, Februari 2014.

[68] Various. Netduino helpers. http://netduinohelpers.codeplex.com/, Februari 2014.

[69] Various. C sharp (programming language). http://en.wikipedia.org/wiki/C_Sharp_(programming_language), Februari 2014.

[70] Various. Overview of the .net framework. http://msdn.microsoft.com/en-us/library/zw4w595w.aspx, Februari 2014.

[71] Microsoft. Asp.net signalr. http://www.signalr.net, April 2014.

[72] Various. Websocket. http://en.wikipedia.org/wiki/WebSocket, April 2014.

[73] Chander Dhall. Signalr in asp.net applications. http://devproconnections.com/aspnet/signalr-aspnet-applications, April 2014.

[74] Highcharts. Highcharts - interactive javascript charts for your webpage. http://www.highcharts.com/, April 2014.

[75] Various. Eclipse (software). http://en.wikipedia.org/wiki/Eclipse_%28software%29, Mars 2014.

[76] Various. Secure copy. http://en.wikipedia.org/wiki/Secure_copy, Mars 2014.

[77] Various. Secure shell. http://en.wikipedia.org/wiki/Secure_Shell, Mars 2014.

[78] Chris Walker. Onewire alpha. http://forums.netduino.com/index.php?/topic/230-onewire-alpha/page-4#entry16691, Augusti 2011.

[79] Manish Pagey. Libserial. http://libserial.sourceforge.net/, 2004.

[80] Various. Interface (computing). http://en.wikipedia.org/wiki/Interface_%28computing%29, April 2014.

[81] Arduino. Power supplies. http://playground.arduino.cc/Main/IntWithHW-PwrSup, April 2014.

[82] Unknown. Beagleboard.org - forums. http://beagleboard.org/Community/Forums, April 2014.

Bilaga A

Kommunikationsflöden

Denna bilaga innehåller exempel på kommunikationsflödet mellan en mikrokontroller och webbtjänsten. Bilaga A.1 ger ett exempel på associering med webbtjänsten och begäran av gränsvärden. Bilaga A.2 ger ett exempel på associering med webbtjänsten och begäran av gränsvärden, men med ett förkortat svar som utelämnar viss information. Bilaga A.3 ger ett exempel på rapportering av ett mätvärde till webbtjänsten.

A.1 Begäran av mätgränser och associering med webbtjänsten

I Kodstycke A.1 och A.2 visas kommunikationsflödet mellan mikrokontrollern och webbtjänsten när mikrokontrollern associerar sig med webbtjänsten och begär mätgränser.

```
GET /api/sensor/netduino_temp/?type=json HTTP/1.1
Host: ***
User-Agent: NetduinoBrowser 0.001
```

Kodstycke A.1: Mikrokontrollerns begäran

```
HTTP/1.1 200 OK
Cache-Control: no-cache
Pragma: no-cache
Content-Type: application/json; charset=utf-8
Expires: -1
Server: Microsoft-IIS/8.0
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
Date: Fri, 14 Mar 2014 14:16:22 GMT
Content-Length: 142

{"Id":3,"SensorId":"netduino_temp","D":18.0,"U":25.0,
"Type":{"Id":2,"Type":"Temperature"},
"Device":{"Id":2,"Name":"Netduino"}}
```

Kodstycke A.2: Webbtjänstens svar

A.2 Begäran av mätgränser och associering med servern, med förkortat svar

Som ett alternativ till kommunikationsflödet i bilaga A.1 kan servern svara med endast det som är nödvändigt för mätintervallet. Detta innebär att mindre mängd data skickas från servern och underlättar behandlingen av detta för mikrokontrollern. Kodstycke A.3 och A.4 visar kommunikationsflödet med det förkortade svaret.

```
GET /api/sensorshort/netduino_temp/?type=json HTTP/1.1
Host: ***
User-Agent: NetduinoBrowser 0.001
```

Kodstycke A.3: Mikrokontrollerns begäran vid förkortat svar

```
HTTP/1.1 200 OK
Cache-Control: no-cache
Pragma: no-cache
Content-Type: application/json; charset=utf-8
Expires: -1
Server: Microsoft-IIS/8.0
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
Date: Fri, 14 Mar 2014 14:25:48 GMT
Content-Length: 69

{"Id":3,"SensorId":"netduino_temp","D":18.0,"U":25.0}
```

Kodstycke A.4: Webbtjänstens svar vid förkortat svar

A.3 Rapportering av mätvärde till webbtjänsten

Kodstycke A.5 och A.6 visar kommunikationsflödet mellan mikrokontrollern och webbtjänsten när mikrokontrollern rapporterar in ett mätvärde.

```
POST /api/reading/?data=25.5625&sensorid=6 HTTP/1.1
Host: ***
User-Agent: NetduinoBrowser 0.001
Content-Length: 0
```

Kodstycke A.5: Mikrokontrollerns begäran vid rapportering

```
http/1.1 201 created
cache-control: no-cache
pragma: no-cache
content-type: application/json; charset=utf-8
expires: -1
location: ***/api/reading/293
```

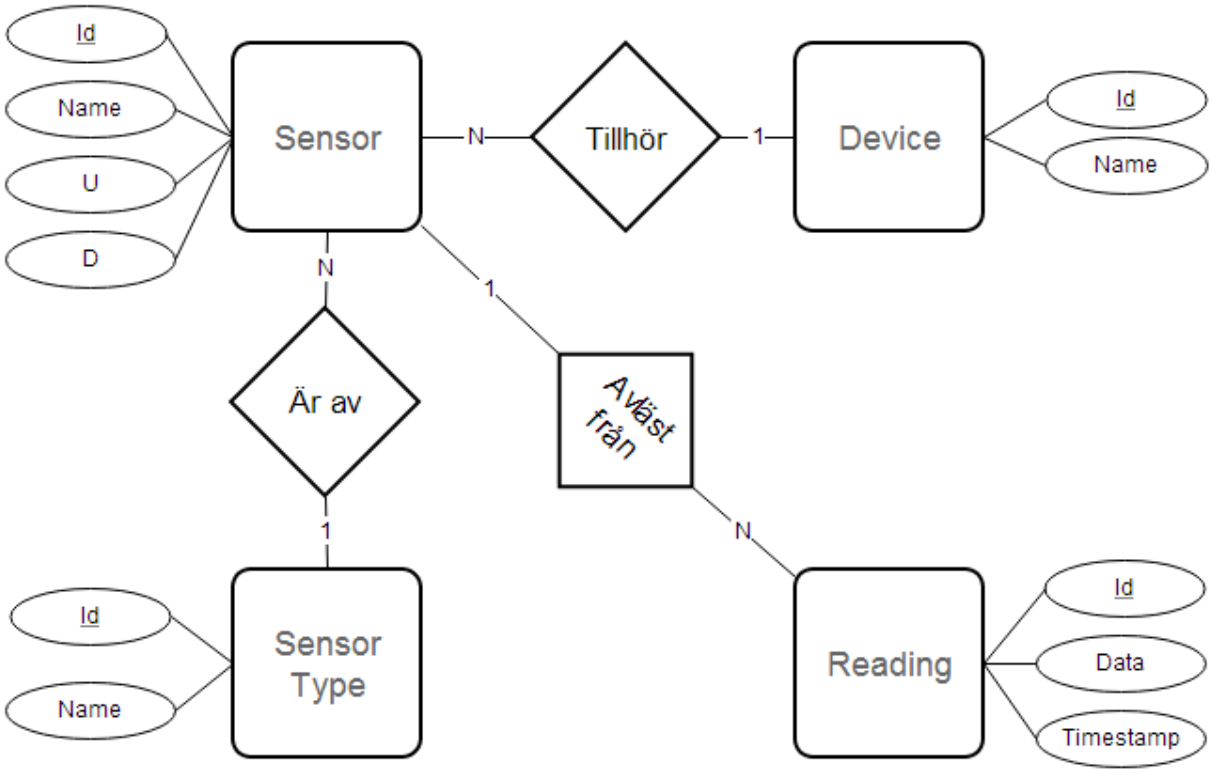
```
server: microsoft-iis/8.0
x-aspnet-version: 4.0.30319
x-powered-by: asp.net
date: tue, 25 feb 2014 10:32:44 gmt
content-length: 133

"id":293,"data":"25.5625","timestamp":"2014-02-25T10:32:44.2398105Z",
''Sensor'':Null}
```

Kodstycke A.6: Webbtjänstens svar vid rapportering

Bilaga B

ER-diagram

Figur B.1 visar entitet-relationsdiagrammet (ER-diagram, ERD) för databasen. Entiteter och relationer mellan dem är beskrivna i avsnitt 3.2.4.



ERD Peter Chen's Notation			
Cardinality		Optionality	
1	One	0	Optional
N	Many	1	Mandatory

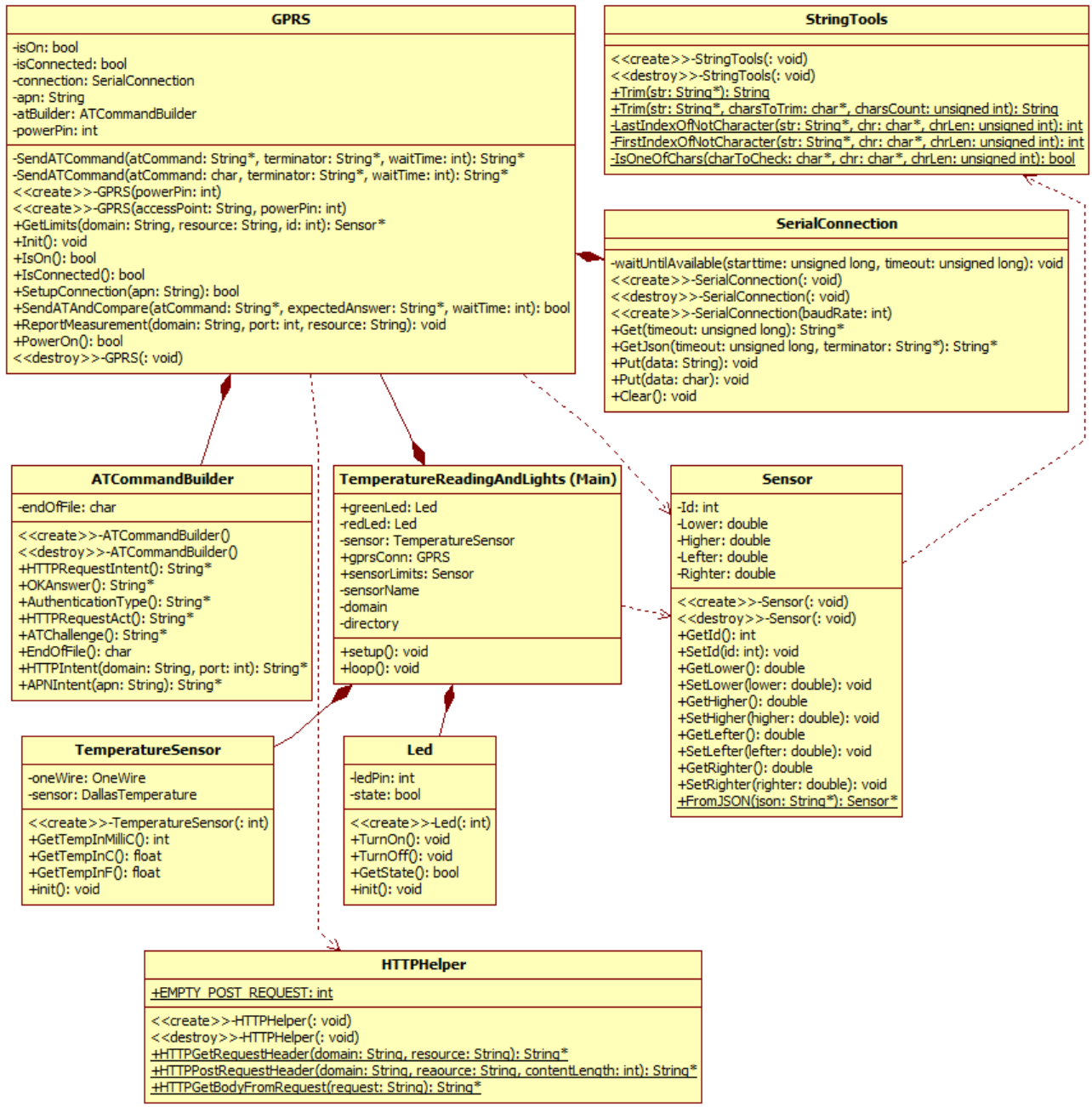
Figur B.1: ER-diagram för databasen

Bilaga C

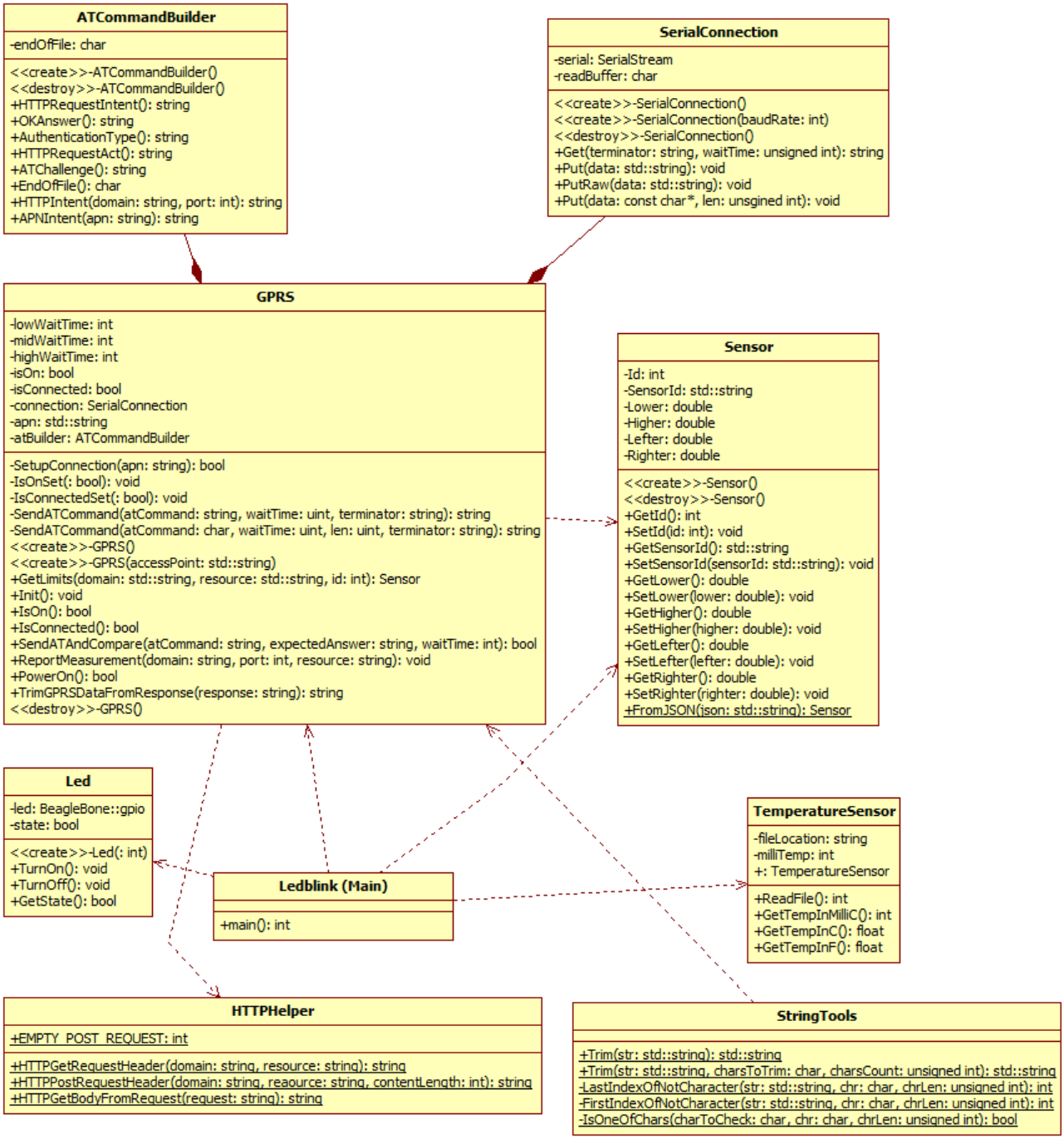
UML-diagram

Denna bilaga innehåller UML-diagram för den specifika designen av jämförelsefallet beskrivet i avsnitt 2.2.1 på de tre mikrokontrollerna. Den övergripande designen diskuteras i avsnitt 3.6. Bilaga C.3 visar designen av jämförelsefallet på Netduino, bilaga C.2 visar designen på BeagleBone Black och bilaga C.1 visar designen på Arduino.

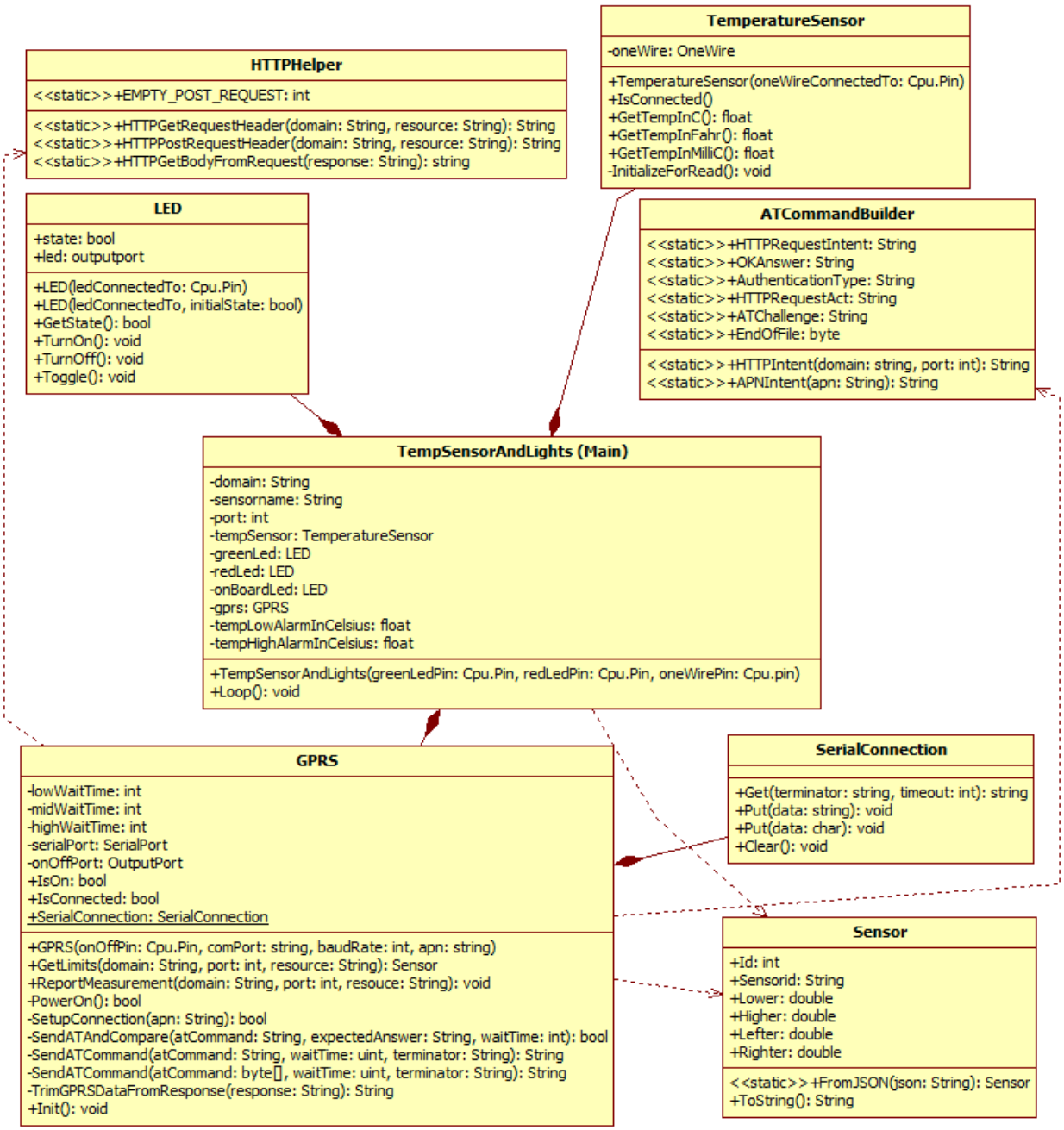
C.1 Implementationslösning Arduino



C.2 Implementationslösning BeagleBone Black



C.3 Implementationslösning Netduino



Bilaga D

Tredjepartsbibliotek

Denna bilaga sammanställer de tredjepartsbibliotek som använts för att implementera jämförelsefallet, beskrivet i avsnitt 2.2.1, på de tre mikrokontrollerna. Den övergripande designen diskuteras i avsnitt 3.6.

D.1 Arduino Nano

- *Dallas Temperature* har använts för att läsa av temperaturen från DS18B20. `https://github.com/milesburton/Arduino-Temperature-Control-Library`

D.2 Netduino

- Inga tredjepartsbibliotek har använts, men en speciell firmware installerades på enheten för att få stöd för DS18B20. `http://forums.netduino.com/index.php?/topic/230-onewire-alpha/`

D.3 BeagleBone Black

- *LibSerial* har använts för seriell kommunikation. `http://libserial.sourceforge.net/`
- *BoneLib* har använts för kommunikation med de digitala I/O-portarna. `http://sourceforge.net/projects/bonelib/`
- Kärnmodulen som använts för att läsa av temperaturen från DS18B20 finns beskriven på `http://goo.gl/pFyxyt`

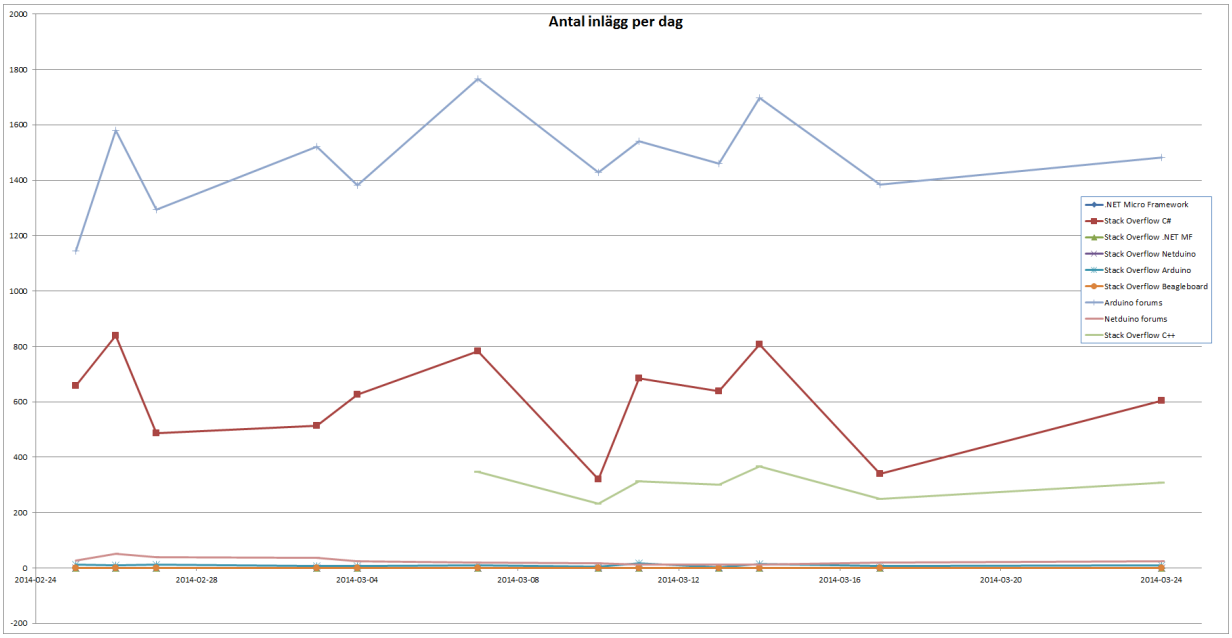
Bilaga E

Statistik över popularitet

I detta kapitel finns diagram och tabeller som redovisar resultatet av undersökningen beträffande hur populära de olika plattformarna och deras respektive programmeringsspråk är bland olika diskussionsforum på Internet.

E.1 Genomsnittligt antal inlägg per dag

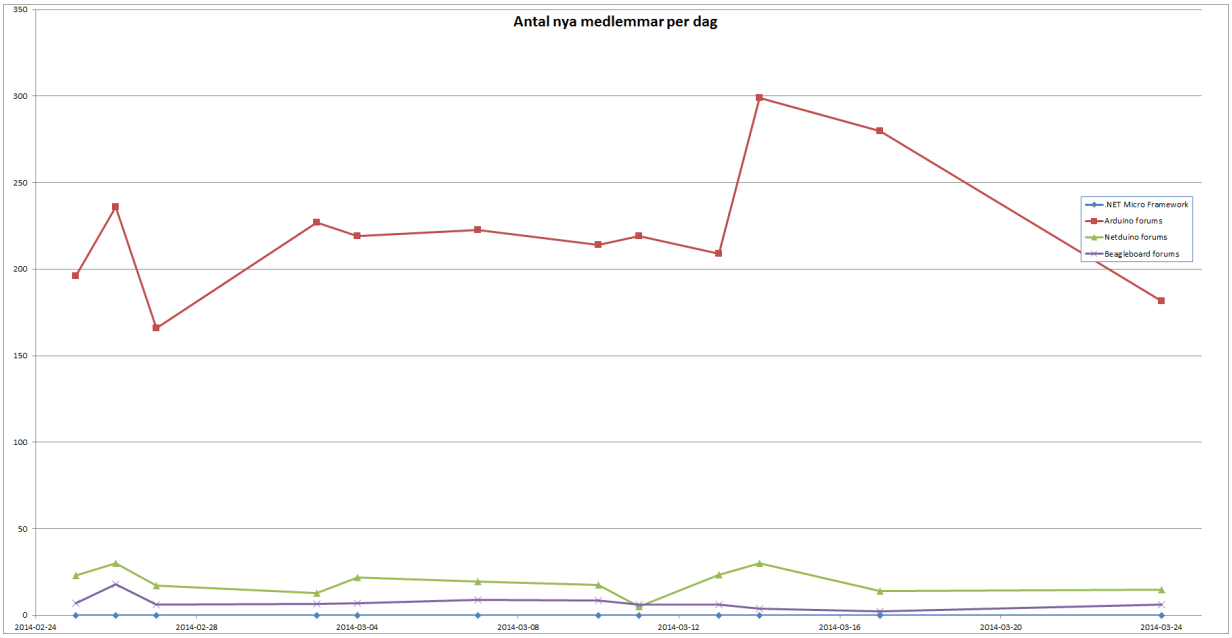
Diagrammet i Figur E.1 visar hur många inlägg som gjorts på de olika diskussionsforumen per dag. Antalet dagar mellan observationstillfällen skiljer sig åt, vilket har justerats genom att dividera skillnaden mellan två observationer med antalet dagar som skiljer dem åt. Detta resulterar i att en genomsnittlig ökning redovisas. Bland plattformarna BeagleBoard, Arduino och Netduino är det på Arduinos egna diskussionsforum överlägset flest inlägg skrivs. BeagleBoard rapporterar inte denna siffra för deras eget diskussionsforum, varför detta har utelämnats i diagrammet. Även på Stack Overflow är det om Arduino diskussioner har skett mest frekvent, även om det är i en väldigt liten grad jämfört med det egna forumet. Bland programspråken är det C# som har diskuterats mest flitigt, men intresset för C++ är högt även det. C++ kom in senare i studien, varför grafen börjar en bit in.



Figur E.1: Antalet inlägg som gjorts per dag

E.2 Genomsnittlig medlemsökning per dag

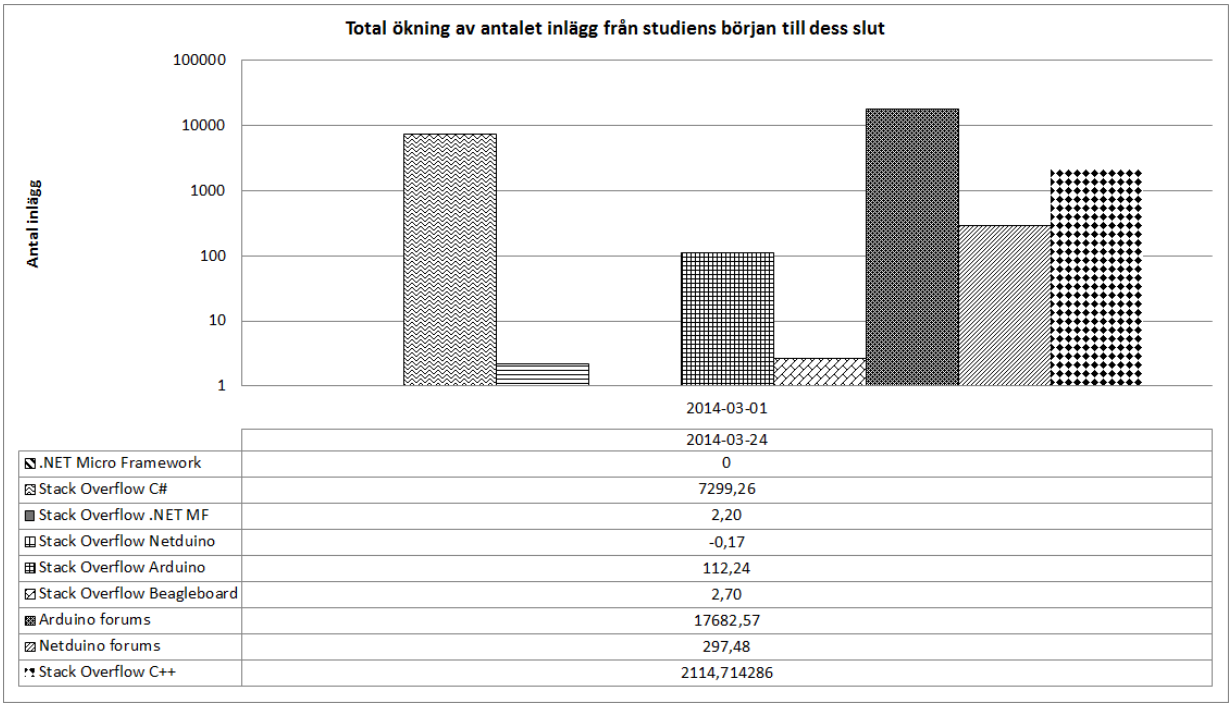
Diagrammet i Figur E.2 visar hur många medlemmar som har registrerat sig på de olika diskussionsforumen per dag. Antalet dagar mellan observationstillfällen skiljer sig åt, vilket har justerats genom att dividera skillnaden mellan två observationer med antalet dagar som skiljer dem åt. Detta resulterar i att ett genomsnittligt antal registrerade redovisas. I diagrammet finns det en plattform som sticker ut jämfört med de andra. Plattformen är Arduino. Ur diagrammet kan man dra slutsatsen att Arduinos diskussionsforum är det som har störst aktivitet sett till hur många nya medlemmar som registrerar sig varje dag.



Figur E.2: Antalet medlemmar som registrerat sig per dag

E.3 Total ökning av antalet inlägg vid undersökningens slut

Diagrammet i Figur E.3 visar den totala ökningen av antalet inlägg på de olika diskussionsforumen från undersökningens början till dess slut. I detta kan man se att bland programspråken är C# det populäraste. Ungefär 3,4 gånger fler inlägg skrivs per dag om C# än om tvåan, C++. Bland de olika mikrokontrollernas forum är det Arduino som leder överlägset. Det skrivs mer om plattformen både på deras egna diskussionsforum och Stack Overflow jämfört med de andra. Intresset för Netduino och BeagleBoard på StackOverflow är i det närmaste obefintligt. BeagleBoard redovisar inte totalt antal inlägg på sitt diskussionsforum, men enligt deras egen statistik gjordes det ca 1600 inlägg i mars månad [82]. Om denna statistik stämmer skulle det innebära att BeagleBoard är mer populär än Netduino.

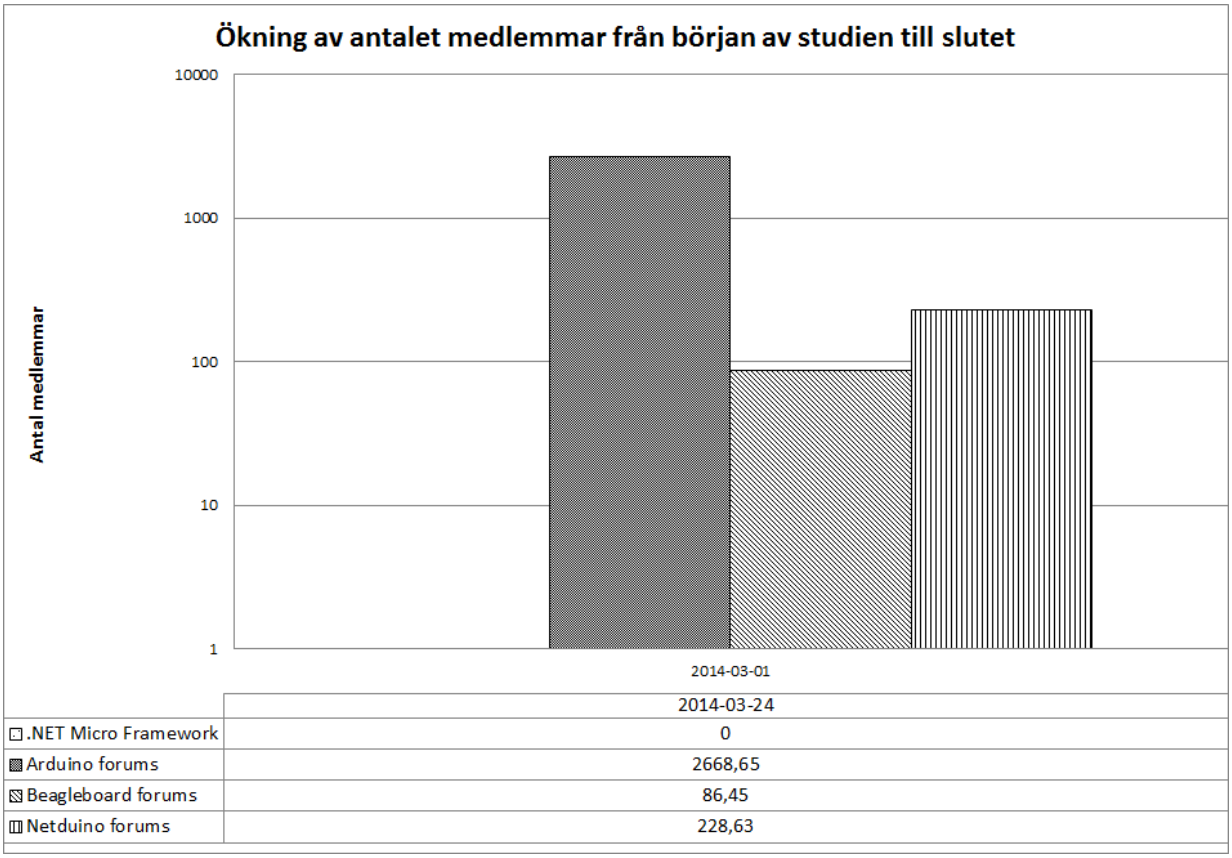


	2014-03-24
.NET Micro Framework	0
Stack Overflow C#	7299,26
Stack Overflow .NET MF	2,20
Stack Overflow Netduino	-0,17
Stack Overflow Arduino	112,24
Stack Overflow Beagleboard	2,70
Arduino forums	17682,57
Netduino forums	297,48
Stack Overflow C++	2114,714286

Figur E.3: Total ökning av antal inlägg från studiens början till dess slut.

E.4 Total ökning av antal medlemmar vid undersökningens slut

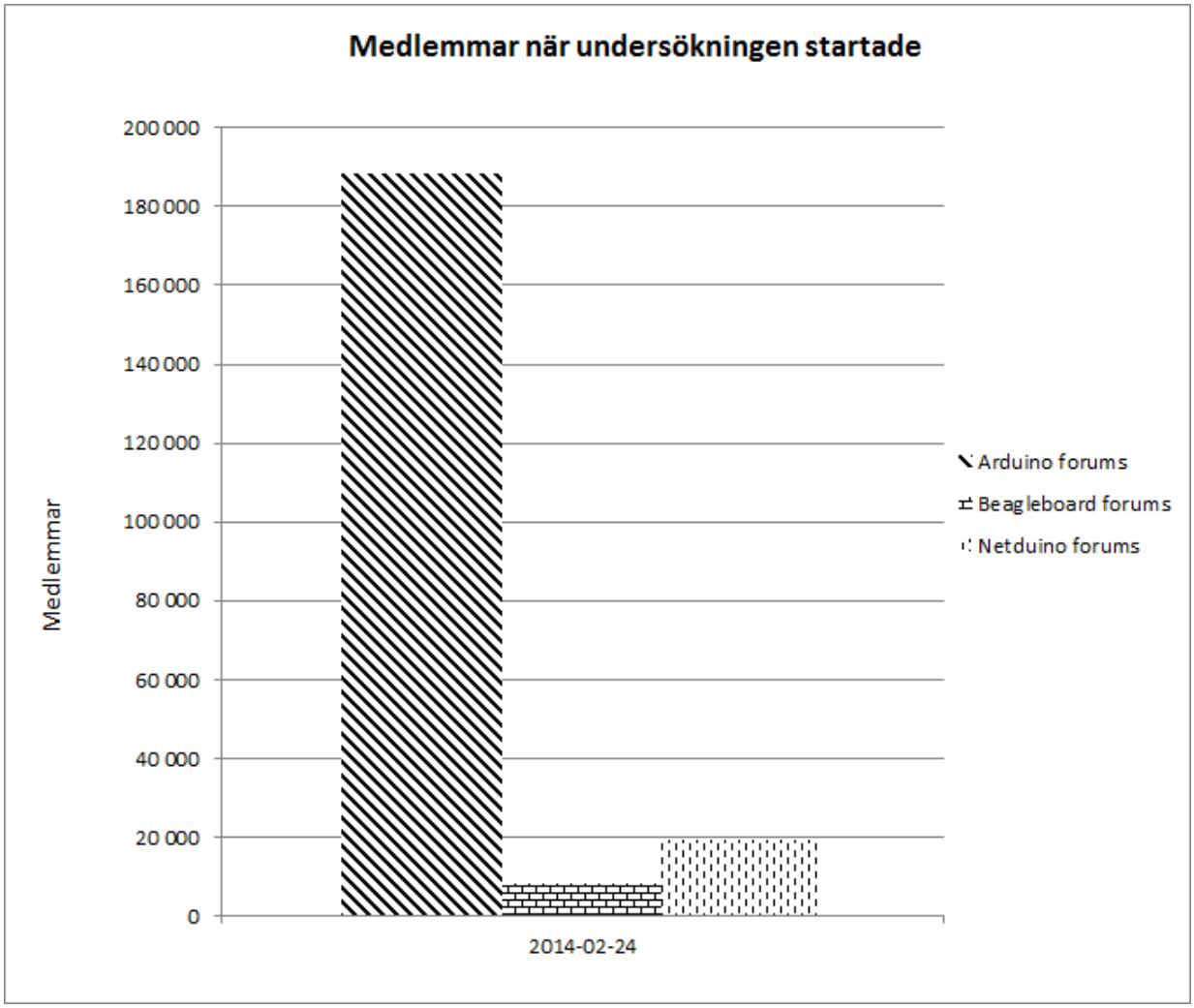
Diagrammet i Figur E.4 visar den totala ökningen av antalet medlemmar då undersökningen avslutades. I detta diagram kan man se att Arduinos diskussionsforum är det som ökar överlägset mest. Mer än tio gånger fler medlemmar har registrerat sig under perioden jämfört med, det näst största, Netduinos forum. Alla diskussionsforum som har observerats, till exempel Stack Overflow, redovisar inte antalet medlemmar. Men då dessa gäller tillverkarnas egna diskussionsforum ger det en bra bild av plattformarnas popularitet, med avseende på användarantal.



Figur E.4: Ökning av antalet medlemmar från studiens början till dess slut.

E.5 Totalt antal medlemmar vid undersökningens början

Diagrammet i Figur E.5 visar det totala antalet medlemmar på diskussionsforumen för de olika mikrokontrollerna vid tillfället då undersökningen påbörjades. I detta diagram kan man se att Arduino är den överlägset populäraste mikrokontrollern. Antalet medlemmar är nästan tio gånger fler än på, den näst mest populära mikrokontrollern, Netduinos diskussionforum.



Figur E.5: Antal medlemmar när studien påbörjades.

E.6 Rådata

Tabellen E.6 visas den rådata som tagits fram i undersökningen. Olika diskussionsforum redovisar olika parametrar, varför vissa har utlämnats. De olika datum som finns som rubrikkolumner är datum då observationer har gjorts. Dessa har skett med olika frekvens, varför antalet observerade inlägg, eller medlemmar, vid en observation har subtraherats med antalet vid föregående observation. Skillnaden har sedan dividerats med antalet dagar mellan observationerna för att få en genomsnittlig ökning per dag. Från dessa medelvärden

har sedan medelvärde, median och standardavvikelse beräknats för alla skillnader.

		2014-02-24	2014-02-25	2014-02-26	2014-02-27	2014-03-03	2014-03-04	2014-03-07	2014-03-10	2014-03-11	2014-03-13	2014-03-14	2014-03-17	2014-03-24	Medel	Median	Std. Avv.
.NET Micro Framework	Medlemmar	1325	1325	1325	1325	1325	1325	1325	1325	1325	1325	1325	1325	1325			
	Ökning		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Inlägg	887	887	887	887	887	887	887	887	887	887	887	887	887			
	Ökning		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Stack Overflow C#	Inlägg	597695	598353	599191	599678	601734	602360	604705	605665	606349	607626	608434	609456	613680			
	Ökning		658,00	838,00	487,00	514,00	626,00	781,67	320,00	684,00	638,50	808,00	340,67	603,43	608,27	632,25	168,009
Stack Overflow .NET MF	Inlägg	169	169	169	169	170	171	171	173	173	173	173	173	175			
	Ökning		0,00	0,00	0,00	0,25	1,00	0,00	0,67	0,00	0,00	0,00	0,00	0,29	0,1835	0,00	0,32813
Stack Overflow Netduino	Inlägg	51	51	50	50	50	50	50	51	51	52	52	52	52			
	Ökning		0,00	-1,00	0,00	0,00	0,00	0,00	0,33	0,00	0,50	0,00	0,00	0,00	-0,014	0,00	0,35146
Stack Overflow Arduino	Inlägg	3383	3395	3405	3416	3446	3453	3481	3496	3512	3519	3534	3553	3620			
	Ökning		12,00	10,00	11,00	7,50	7,00	9,33	5,00	16,00	3,50	15,00	6,33	9,57	9,3532	9,45	3,78456
Stack Overflow Beagleboard	Inlägg	289	289	289	289	288	289	291	292	292	294	294	293	295			
	Ökning		0,00	0,00	0,00	-0,25	1,00	0,67	0,33	0,00	1,00	0,00	-0,33	0,29	0,2252	0,00	0,44744
Arduino	Inlägg	1590484	1591629	1593209	1594504	1600592	1601973	1607274	1611560	1613100	1616018	1617715	1621871	1632249			
	Ökning		1145,00	1580,00	1295,00	1522,00	1381,00	1767,00	1428,67	1540,00	1459,00	1697,00	1385,33	1482,57	1473,5	1470,79	168,582
	Medlemmar	188590	188786	189022	189188	190095	190314	190982	191624	191843	192261	192560	193399	194670			
Beagleboard forums	Ökning		196,00	236,00	166,00	226,75	219,00	222,67	214,00	219,00	209,00	299,00	279,67	181,57	222,39	219,00	37,135
	Medlemmar	8179	8186	8204	8210	8236	8243	8270	8295	8301	8313	8317	8324	8368			
	Ökning		7,00	18,00	6,00	6,50	7,00	9,00	8,33	6,00	6,00	4,00	2,33	6,29	7,2044	6,39	3,82355
Netduino forums	Inlägg	51900	51927	51979	52018	52160	52185	52244	52296	52309	52334	52347	52405	52574			
	Ökning		27,00	52,00	39,00	35,50	25,00	19,67	17,33	13,00	12,50	13,00	19,33	24,14	24,79	21,90	12,0719
	Medlemmar	19400	19423	19453	19470	19521	19543	19601	19653	19658	19705	19735	19777	19880			
	Ökning		23,00	30,00	17,00	12,75	22,00	19,33	17,33	5,00	23,50	30,00	14,00	14,71	19,053	18,33	7,21448
Stack Overflow C++	Inlägg	0	0	0	0	0	270158	271198	271892	272204	272806	273173	273920	276074			
	Ökning							346,67	231,33	312,00	301,00	367,00	249,00	307,71	302,1	307,71	48,5738

Figur E.6: Rådata över observationer