Webbaserad temperatur- och fuktighetsmätning

Kristian Granby

Examensarbete för ingenjör (YH)-examen

Utbildningsprogrammet för informationsteknik

Vasa 2014

EXAMENSARBETE

Författare:	Kristian Granby
Utbildningsprogram och ort:	Informationsteknik, Vasa
Handledare:	Mikael Jakas

Titel: *Webbaserad temperatur- och fuktighetsmätning*

Datum 15.4.2014 Sidantal 26 Bilagor 4

Abstrakt

Detta examensarbete gjordes för MKK-bolaget för att ta fram en lösning för att avläsa temperatur och fuktighet i ett torkrum. Kravet var att mätningarna skulle kunna avläsas hemifrån av användaren.

Ett Arduino Uno mikrokontrollerkort med en Ethernet-modul och en DHT22 temperatur- och fuktsensor användes som hårdvara. För att driva systemet användes kod från Arduinos utvecklingsmiljö och ett tredje parts bibliotek, vilka kompletterades för att visa mätningarna på en webbsida.

Resultatet är en flexibel lösning som kan ändras efter användarens behov.

Språk: svenska Nyckelord: Arduino, fuktighetsmätning, temperaturmätning

---

OPINNÄYTETYÖ

Tekijä: Kristian Granby

Koulutusohjelma ja paikkakunta: Tietotekniikka, Vaasa

Ohjaaja: Mikael Jakas

Nimike: *Web-pohjainen lämpötilan ja kosteuden mittaus*

Päivämäärä 15.4.2014 Sivumäärä 26 Liitteet 4

Tiivistelmä

Opinnäytetyö tehtiin MKK-yritykselle, jotta voidaan kehittää ratkaisu kuivaushuoneen lämpötilan ja kosteuden lukemiseen. Vaatimuksena oli, että käyttäjä voi lukea mittaukset kotoaan.

Laitteistona käytettiin Arduino Uno-mikrokorttia, siihen liitettyä Ethernet-moduulia sekä DHT22 lämpötilan ja kosteuden anturia. Järjestelmään käytetään Arduino-kehitysympäristökoodia ja kolmannen osapuolen kirjastoa, jotka täydennettiin mittauksien näyttämiseen verkkosivuilla.

Tuloksena on joustava ratkaisu, joka voidaan muuttaa käyttäjän mukaan.

Kieli: ruotsi Avainsanat: Arduino, kosteuden mittaus, lämpötilan mittaus

---

BACHELOR'S THESIS

Author:	Kristian Granby
Degree Programme:	Information Technology, Vaasa
Supervisor:	Mikael Jakas

Title: *Web-based temperature and humidity measurement*

Date 15.4.2014 Number of pages 26 Appendices 4

Summary

This thesis was made for MKK Company in order to develop a solution for reading the temperature and humidity in a drying room. The requirement was that the measurements could be read from home by the user.

An Arduino Uno microcontroller card with an Ethernet module and a DHT22 temperature and humidity sensor were used as hardware. To operate the system, code from the Arduino development environment was used from as well as a third-party library, which were completed to show the measurements on a website.

The result is a flexible solution that can be modified for the user's needs.

Language: Swedish Key words: Arduino, humidity measurement, temperature measurement

---

Innehållsförteckning

Ordförklaringar

RH	Relative Humidity. Relativ fuktighet anger andelen vattenånga i föhållande till den maximalt möjliga mängden vattenånga vid aktuell temperatur.
Sketch	Arduino använder namnet sketch för program i deras utvecklingsmiljö.
Sorptionstorkare	Sorptionstorkaren använder en fuktabsorbent som först torkas med varmluft och som sedan torkar luften som förs genom den. Absorbenten roterar långsamt likt en LP-skiva och man blåser fuktig luft genom en sektion av absorbenten. I en annan del av absorbenten blåser man uppvärmd luft som torkar ut materialet. Den upptagna vattenångan förs ut ur rummet som våtluft.
SPI	Serial Peripheral Interface är ett synkront seriellt data protokoll, som används av en mikrokontroller för att kommunicera med kringutrustning.
Processing	Processing är ett programmeringspråk som använder öppen källkod vars syfte var att lära ut programmering i en visuell miljö. Används i Arduino-miljön för att t.ex. rita en graf.

1 Inledning

Uppgiften för det här examensarbetet är att konstruera en anordning för att mäta temperatur och fuktighet i ett torkrum på en mink- och rävfarm. Användaren ska med en pc eller en smarttelefon kunna avläsa mätningarna. Mätningarna ska sparas i en databas för att kunna följa med variationer i temperatur och fukt. Alarm ska visas åt användaren om temperaturen eller fuktigheten över-/underskrider vissa gränsvärden.

1.1 Uppdragsgivare

MKK-bolaget i Korsnäs har som huvudsakligt verksamhetsområde att producera pälsdjur.

MKK-bolaget bildades år 2000 som en direkt följd av att produktionsinriktningen skulle breddas. Ägarna hade från tidigare i sina respektive bolag rävuppfödning som nu också skulle breddas med minkuppfödning. Under uppbyggnadsskedet av produktionsutrymmena bedrev bolaget enbart minkuppfödning för att senare sammanföra rävuppfödningen från de andra bolagen till MKK-bolaget.

Finlands Pälsdjursuppfödares Förbund r.f. införde tillsammans med det egna auktionsbolaget Saga Furs Oyj certifiering av Finlands pälsfarmer. Farmen ansökte i ett tidigt skede om certifiering. Efter auditering godkändes farmen och infördes i certifieringsregistret.

År 2012 införskaffades ett nytt farmområde för produktion av räv. Det nya området håller på att succesivt byggas ut.

Bolaget har i dagsläget möjlighet att med full kapacitet producera närmare 20 000 skinn årligen. Bolaget har inga fasta anställda förutom delägarna men tillfälliga arbetstagare utgör ca 3,0 st. på årsbasis.

Produktionen sker i tidsenliga produktionsutrymmen. All pälsning utförs på egen farm och med så långt som möjligt automatiserad utrustning.

Utfodringen sker till stor del genom individuell matning av djuren. Genom datautrustningen kan bestämmas hur mycket foder varje individ i farmen ska ha. Genom en handterminal på fodertrucken och streckkodsetiketter på burarna ger foderpumpen rätt fodermängd på varje enskild bur.

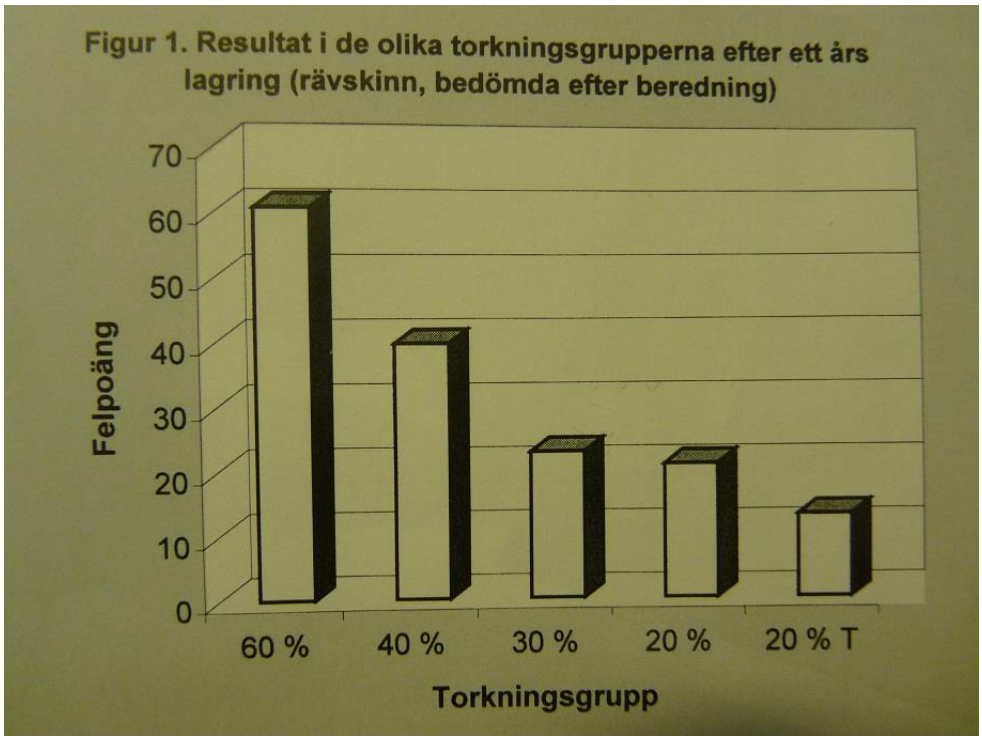
De senaste tre åren har övervakningen av farmområdet fått kameraövervakning och likaledes har övervakningen av pälsningsutrymmena försetts med kameraövervakning och uppkoppling till hemdatorn. Klimatförhållandena har inkopplats i samma system för att lättare kontrollera torkförhållandena av skinnen under pälsningen.

Följande steg är att ta i bruk ett system för den praktiska farmningen. Systemet är avsett som ett enkelt hjälpmedel i farmen. För att använda detta system klarar man sig med en vanlig smartphone och dator med internetanslutning. Med systemen kan man registrera allt möjligt avvikande som förekommer på farmen, t.ex. vattenläckage, skador på burar, skador på själva konstruktionen, insjuknade djur samt annat som bör åtgärdas. Tack vare systemet slipper man tidskrävande pappersdokumentation som krävs i certifieringskraven.

1.2 Temperatur och fuktighet

Det är mycket viktigt att hålla rätt temperatur och fuktighet i ett torkrum där man torkar mink och räv. Traditionellt har man hållit fuktigheten runt 60 % relativ fuktighet vid 18 °C, men då skinnen har blivit större har det uppkommit ett behov av effektivare torkningsmetoder. Blir fuktigheten i torkrummet för hög under torkningsprocessen leder det till att mögel uppstår och skinnen kan bli förstörda.

På farmerna har även tidigare använts kondensationstorkar som dock inte blivit särskilt populära. Turkistuottajat inledde i samarbete med ett antal farmare 1996 ett försök med sorptionstorkare. Flera försök med torkning vid olika fuktighetsprocent gjordes under flera års tid. Resultatet redovisades i Finsk Pälstidskrift 11/99. Se figur 1 nedan. Resultatet visar att felpoängen sjunker vid lägre fuktighet. T vid resultatet 20 % T betyder att det är dubbelt mera tryck i inblåsningsluften än vid de övriga mätningarna.



Figur 1 Resultat angående kvalitet vid torkning av rävskinn (1).

Eftersom MKK-bolaget har investerar i en ny avfuktare vill de gärna hålla koll på temperatur och fuktighet i torkrummet.

1.3 Uppgift

Uppgiften kan lösas med i stort sett med vilken teknik som helst som kan kopplas till ett nätverk. Den behöver dock vara pålitlig och lätt att reparera. Den ska också gå att bygga ut för mera funktionalitet. Kostnaderna får inte heller bli hur höga som helst.

2 Undersökning

Undersökningen gjordes för att hitta en enkel lösning, som skulle uppfylla kraven och samtidigt vara prisvärd. Några olika alternativ undersöktes för att hitta den bästa lösningen.

2.1 Alternativ

Följande alternativ undersöktes. De presenteras här under.

- Väderstation
- Raspberry Pi
- Arduino

- **Väderstation**

Första tanken var att använda en väderstation för projektet. Det finns hur många som helst att välja från, men det behöver vara en dyrare modell för att få en USB-anslutning och ännu dyrare för att få en nätverksanslutning. Väldigt få modeller är programmerbara. Det finns lösningar för att få dem mer anpassningsbara, men det fodrar för det mesta tilläggsutrustning.

En USB-anslutning används för att föra över mätningarna till en dator. På datorn installeras ett program där mätningarna visas. Vanligtvis finns inga färdiga funktioner för att ladda upp data till en webbsida eller skicka information vidare.

Till dyrare modeller kan finnas möjlighet att köpa tilläggsutrustning, som gör det möjligt att direkt ladda upp mätningarna till en webbsida.

Ett exempel på en produkt som kan anslutas till internet är en Vantage Pro 2. Till den kan köpas en modul som WeatherLinkIP för att ansluta till Internet. Det går åtminstone i USA att köpa en tilläggstjänst för att lagra data på en webbserver som heter WeatherLink. Från webbsidan kan skickas alarm.

Detta är dock en ganska dyr lösning jämfört med andra alternativ. Se kapitel 9 för prisinformation.

- **Raspberry Pi**

Det finns flera projekt med Raspberry Pi med temperatur och fuktsensor. De flesta verkar fungera bra. Raspberry Pi har Linux som operativsystem. Vanligtvis någon version av

Debian. Projekten är programmerade i Python och Eclipse används ofta som utvecklingsverktyg. Eftersom jag inte har någon erfarenhet av programmering i Python kommer detta att bli för tidskrävande.

- **Arduino**

De flesta projekt använder Arduino. Den har fler pinnar för in- och utmatning än Raspberry Pi. Minnet och processorn är mindre, men för detta projekt räcker det till. Det finns mest tilläggsutrustning för den. Den är lätt att bygga ut och anpassa. Den passar bra för detta projekt.

2.2 Bygga själv

Efter att ha undersökt olika alternativ har jag kommit till slutsatsen att jag kommer att bygga hela projektet på Arduino. Det kommer att ta minst tid därför att programmeringsspråket är ganska lika C och C++ som jag har erfarenhet av från tidigare. Det kommer att vara lättast och billigast att reparera eftersom det kan byggas av olika moduler som är billiga och kan bytas ut om en går sönder. Det är mycket skalbart: Behövs det mera eller annan funktionalitet är det bara att byta ut eller lägga till en modul.

3 Arduino

Beskrivning av Arduino, samt hårdvaran för Arduino-kortet. En kort introduktion till hur mjukvaran fungerar och är uppbygg. Vilket kommer att presenteras i följande underkapitel.

3.1 Vad är Arduino?

Arduino är ett mikrokontrollerkort som är öppen hårdvara. Hårdvaran består av en enkel och öppen kretsdesign med en Atmel AVR och stöd för in- och utgångar. Mjukvaran består av programmeringsspråk, kompilator och en bootloader som körs på kortet.

Hårdvaran programmeras med ett språk som liknar C++ med vissa förenklingar och med en processingbaserad integrerad utvecklingsmiljö. (2)

3.2 Hårdvaran

Ett Arduino-kort består av en 8-bitars Atmel AVR med ytterligare hårdvara för att möjliggöra programmering och inbyggnad i andra kretsar. En viktig del av Arduino är att kopplingar görs med standardiserade och enkelt tillgängliga kopplingspunkter. Det gör att utvecklare kan koppla in allehanda tredjepartsenheter och dessutom standardiserade moduler som kallas sköldar (eng. Shields).

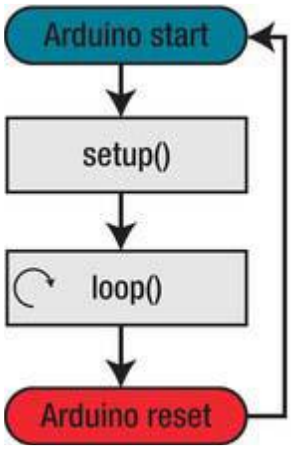
Sköldmodulerna pluggas in till Arduino-korten via kopplingspinnarna. Vissa sköldar kommunicerar direkt med Arduino-kortet medan andra ges åtkomst via seriebussen, vilket medger att många sköldar kan staplas och användas parallellt (2).

3.3 Mjukvara

Arduinos IDE är en multiplattformsapplikation skriven i Java och är ett derivat från projekten Processing och Wiring. Det är ämnat att introducera programmering till de som inte är vana vid mjukvaruutveckling.

Arduinos IDE levereras med ett C / C++ bibliotek som kallas Wiring, vilket förenklar in- och utmatningsoperationer.

Arduinoprogram skrivs i C/ C++ och det behövs bara två funktioner för att kunna köra ett program:



- Setup() – en funktion som bara körs en gång i början av programmet. Den är valfri, men kan användas för att initialisera variabler.
- Loop() – en funktion som körs ändlöst tills Arduino startas om.

Figur 2 Sketchens kretslopp (3).

Du kan också definiera dina egna metoder inuti sketchen eller länka till andra källfiler eller bibliotek. För att länka till en del av koden använder man #include, som är känt från C-utveckling. Includes placeras först i sketchen. Globala variabler placeras efter include. Kom ihåg att skriva så ”ren” kod som möjligt, vilket betyder inga dubbletter och att hålla sig till minsta möjliga datatyp, så att inte minnet tar slut (3).

De flesta guider för nybörjare börjar med att visa hur man med Arduino får kontakt och kontrollerar hårdvara. Vanligtvis genom att blinka en lysdiod, vilket också kan användas när man felsöker både hårdvara och kod.

Ett exempel på ett program som blinkar en lysdiod:

```
#define LED_PIN 13

void setup () {
    pinMode (LED_PIN, OUTPUT);        // Sätt upp stift 13 för digital
utmatning
}

void loop () {
    digitalWrite (LED_PIN, HIGH);  // Slå på lysdioden
    delay (1000);                  // Vänta 1 sekund (1000 millisekunder)
    digitalWrite (LED_PIN, LOW);   // Slå av lysdioden
    delay (1000);                  // Vänta 1 sekund (1000 millisekunder)
}
```

Arduinos fördelar:

- Prisvärd.
- Lätt att lägga till nya moduler.
- Öppen källkod och hårdvara.
- Har redan kunskap om den.

4 Vilken Arduino?

Efter en snabb sökning på Internet visar det sig att Arduino Uno borde vara det bästa valet. Den är mest dokumenterad och det finns många guider för nybörjare. De flesta återförsäljare rekommenderar Uno som det bästa kortet att börja med.

4.1 Arduino Uno

Arduino Uno är ett mikrokontrollerkort som är baserat på ATmega328. Det har 14 digitala in-/utgångar, 6 analoga ingångar. Det finns USB-anslutning, strömanslutning och en återställningsknapp. Tabell 1 visar specifikationer för Arduino Uno-kortet.

Tabell 1 Arduino Uno-specifikationer.

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPRO	1 KB (ATmega328)
Clock Speed	16 MHz

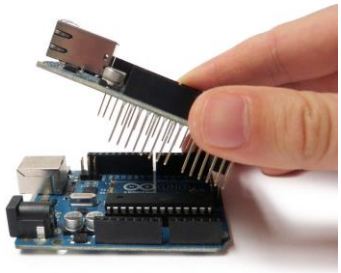
(4)

4.2 Ethernet-sköld (Ethernet Shields)

Arduino Ethernet-skölden används för att ansluta till ett nätverk. Man ansluter modulen till ett Arduino-kort, sätter i en nätverkskabel med en RJ45 kontakt och följer några enkla instruktioner för att ansluta till ett nätverk. Drivs med 5 volt som det får från Arduino-kortet. Hastighet 10/100 Mb. Ansluter till Arduino via SPI-porten.

Ethernet-skölden är baserad på Wiznet W5100 chipet. Wiznet W5100 har en nätverksstack som är kapabel för både TCP och UDP. Det stöder fyra socketanslutningar samtidigt. Ethernet-biblioteket används för att ge extra funktionalitet till sketcher (program) t.ex. för att ansluta till nätverket.

Ethernet-skölden ansluts till Arduino-kortet med långa pinnar, vilka går genom skölden. Detta gör att många sköldar kan staplas ovanpå varandra. Det gör också att pinnarnas layout bibehålls. Se figur 3 för hur det ansluts till ett Arduino Uno-kort.



Figur 3 Installation av Ethernet -köld (17).

Kortet har också plats för ett Micro-SD kort, vilket gör att det kan användas för fillagring. Den inbyggda kortläsaren görs tillgänglig via SD-biblioteket (5).

4.3 Fukt- och temperatursensor

Fukt- och temperatursensorn RHT03 är en relativt billig sensor med ett 1-wire gränssnitt. Den är färdigt kalibrerad och behöver inga extra komponenter. Se tabell 2 för fullständiga specifikationer.

Tabell 2 Fukt och temperatur specifikationer.

Model	RHT03
Power supply	3.3-6V DC
Output signal	digital signal via MaxDetect 1-wire bus
Sensing element	Polymer humidity capacitor
Operating range	humidity 0-100%RH; temperature -40~80Celsius
Accuracy	humidity +-2% RH(Max +-5%RH); temperature +-0.5 Celsius
Resolution or sensitivity	humidity 0.1% RH; temperature 0.1 Celsius
Repeatibility	humidity +-1% RH; temperature +-0.2Celsius
Humidity hysteresis	+-0.3% RH
Long-term Stability	+-0.5% RH/year
Interchangeability	fully interchangeable

(6)

Spänningen ska vara mellan 3.3-6 volt DC. När strömmen kopplas på till sensorn ska inga instruktioner skickas till sensorn inom en sekund, så att den hinner stabilisera sig. En kondensator på 100 nF kan kopplas mellan VDD och GND för filtrering.

5 Installation och testning

När alla delar som blivit beställda anlänt var det dags att plocka ihop och testa att allt fungerade. Först och främst att ingenting var sönder och att det fungerar som planerat.

5.1 Arduino Uno

Började med att ansluta Arduinon till datorn med en USB-kabel. Kontrollerade med enhetshanteraren vilken COM-port som delades ut till Arduinon. Fastän Arduino ansluts till en USB-port visas den som en enhet som är ansluten via en COM-port.

Laddade ner och installerade Arduinos IDE. Arduinos utvecklingsmiljö innehåller en textredigerare för att skriva kod, ett meddelandeområde, en textkonsol, ett verktygsfält med knappar för vanliga funktioner och menyer. Det ansluter till Arduino hårdvaran för att ladda upp program och kommunicera med dem (7).

Windows behöver drivrutiner för att kunna kommunicera med Arduino. När Arduino-kortet ansluts första gången till datorn startar en installationsprocess som misslyckas, men Arduino-programvaran innehåller en fil som heter arduino.inf. Den behöver installeras för att det ska fungerar.

Då allt är anslutet och klart är det dags att skicka över ett program (sketch) för att se att allt fungerar. Arduino kallar sina program för sketcher. I Arduino-programvaran finns många färdiga sketcher för nästan allt möjligt som man kan behöva.

Ett enkelt program som heter blink används för första testet. Programmet blinkar en LED-lampa. Lampan lyser en sekund släcks en sekund och så upprepas proceduren. Se kodexemplet i kapitel 3.3.

5.2 Ethernet-skölden

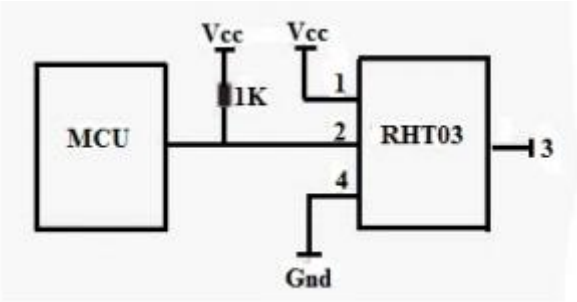
Ethernet-skölden placeras ovanpå Arduino Uno och trycks fast. Pinnarna är lite svåra att få på plats, men efter litet flyttande fram och tillbaka är den på plats. En nätverkskabel med en RJ45 kontakt ansluts.

Dags för att föra över ett program till Ethernet-skölden, men hur kan man föra över programmet? Ethernet-skölden har ingen USB- eller COM port. En snabb sökning på Internet visar att man kan använda ett USB2SERIAL-kort. Eftersom inget USB2SERIAL-kort har blivit beställt måste det finnas något annat sätt att föra över programmet.

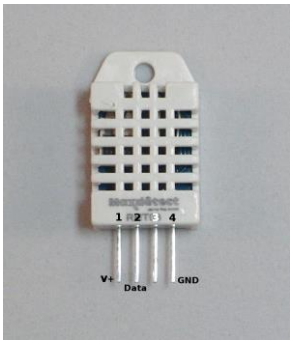
En mycket enkel lösning på problemet är att föra över programmet till Arduino Uno. Eftersom båda korten är ihopsatta fungerar de som en enhet. Testar med ett exempel som finns i Arduino IDE som heter WebServer. Det enda som behöver ändras är Mac-adressen och IP-adressen. IP-adressen måste anpassas till nätverket det ansluts till. Se bilaga 2.

5.3 Fukt- och temperatursensorn

Den sista enheten som ska anslutas är fukt- och temperatursensorn. Har läst genom databladet, men det visar inte direkt hur den ska kopplas till Arduinon. Figur 4 visar kopplingsschemat och figur 5 hur den ser ut.



Figur 4 RHT03 Kopplingsschema (6).



Figur 5 DHT22 (RHT03) (8)

Tabell 3 RHT03(DHT22) pin-layout.

Pin sequence number: 1 2 3 4 (from left to right direction).

Pin	Function
1	VDD—power supply
2	DATA–signal
3	NULL
4	GND

(6)

Förklaring på vad som kopplas in på de olika pinnarna på DHT22 (se tabell 3).

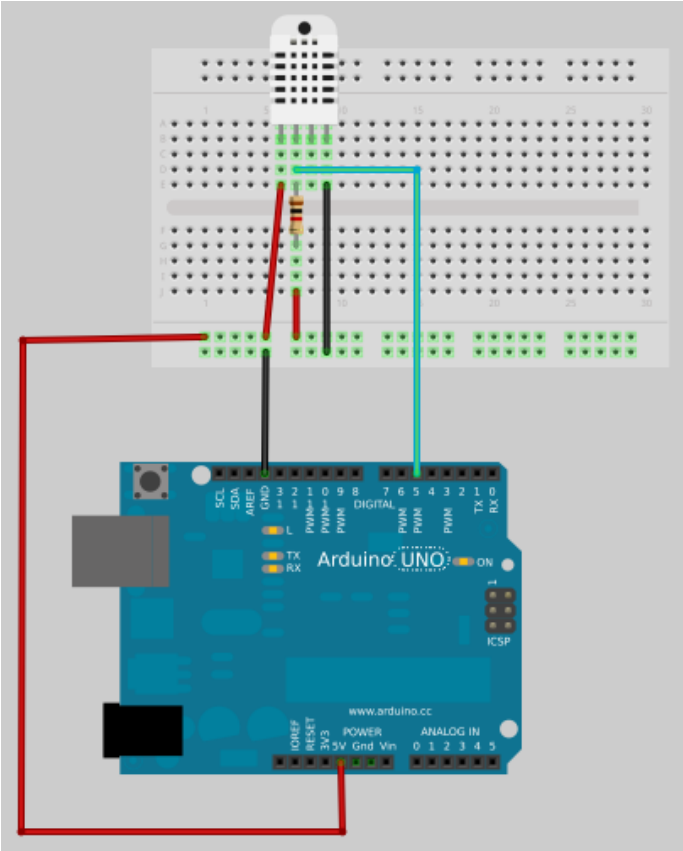
1: Ström (5V)

2: Signal

3: Inte ansluten

4: Jord

Hittade en enkel skiss som är lätt att förstå (se figur 6). Det är bara att följa skissen.



Figur 6 Kopplingsskiss (9)

Behöver ett program för att testa att sensorn fungerar som den ska. Arduino's IDE har inte något färdigt exempel att prova. Hittade dock ett som passar på Arduinos playground.

- Vad är Arduino Playground?

Det är en plats där Arduino användare kan bidra och dra nytta av varandras undersökningar och efterforskningar.

Översatt från sidan: Det här är platsen där du kan posta och dela din kod, kretsdiagram, lathundar, DIY (gör det själv) instruktioner, tips och trick, och efter allt hårt arbete visa upp dina projekt! Vem som helst kan editera och lägga till på de här sidorna (10).

Det behövs ett bibliotek för att kunna använda programmet.

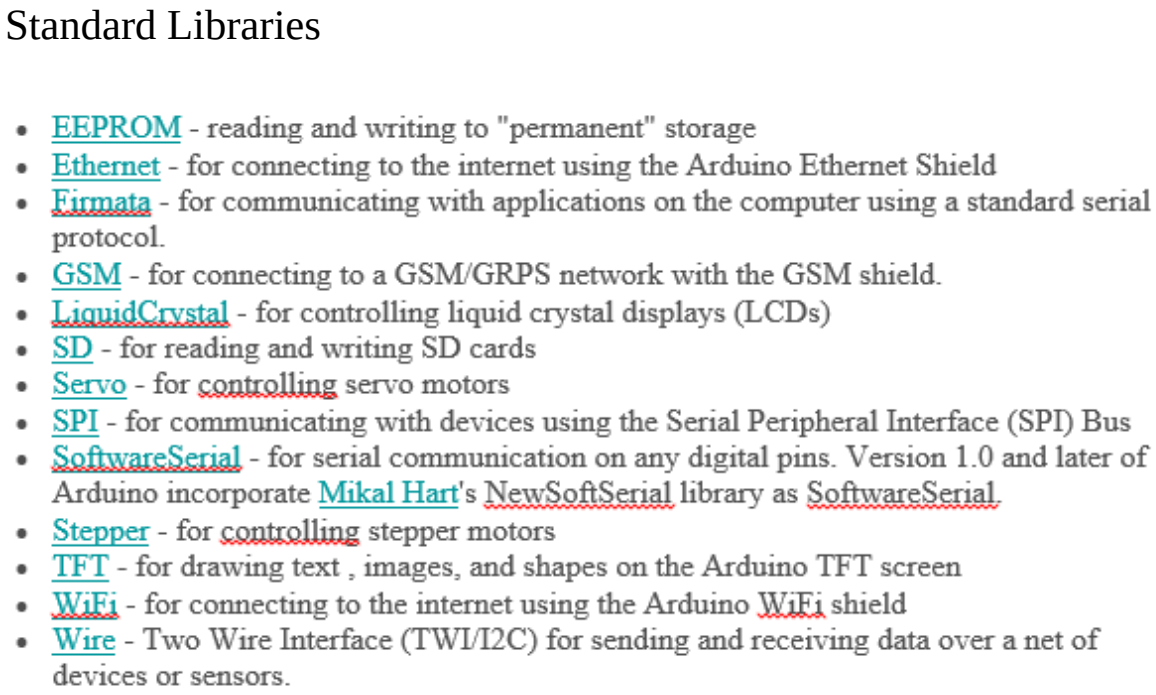
- Vad är ett bibliotek?(eng. Library)

Arduinos utvecklingsmiljö kan utvidgas genom användning av bibliotek som de flesta andra programmeringsplattformar. Bibliotek ger extra funktionalitet för användning i sketcher,

t.ex. arbete med hårdvara eller manipulering av data. Ett antal kommer med IDE:n, men de kan också laddas ner eller så kan du skapa dina egna (11).

Bibliotek är en samling kod som används för att t.ex. ansluta en sensor.

Det finns många olika bibliotek för Arduino. De vanligaste är standardbiblioteken. De kommer med Arduinos IDE. De ger tillgång till funktioner som de flesta kommer att använda någon gång. Ethernet biblioteket t.ex. har jag redan använt för att testa Ethernet kortet. Se figur 7 under på vilka bibliotek som ingår.



Standard Libraries

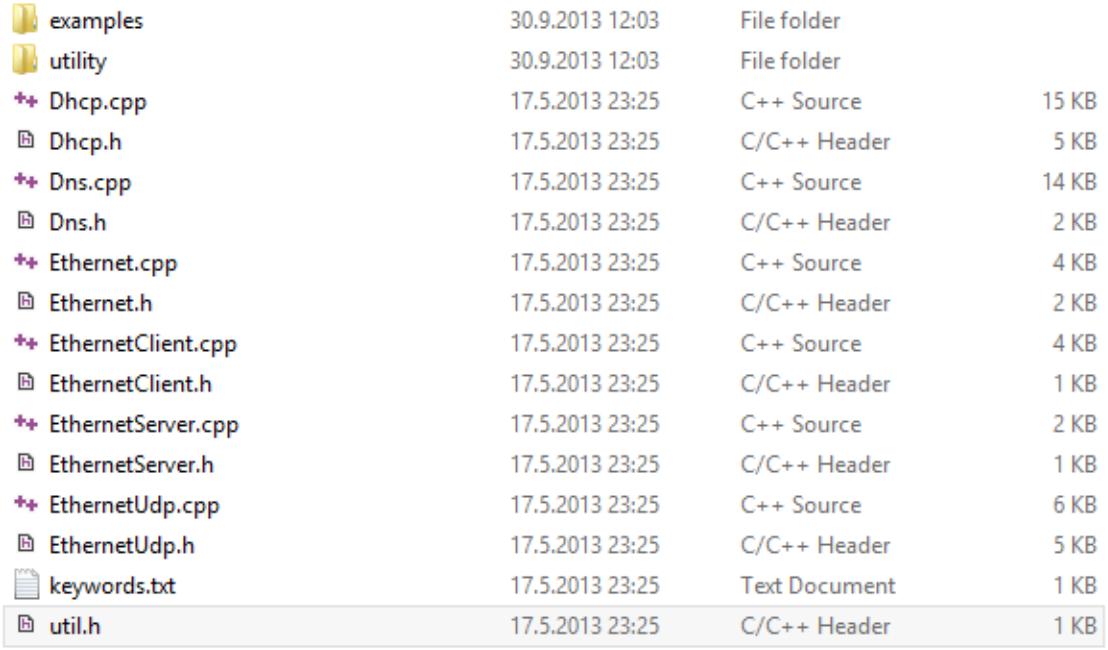
- EEPROM - reading and writing to "permanent" storage
- Ethernet - for connecting to the internet using the Arduino Ethernet Shield
- Firmata - for communicating with applications on the computer using a standard serial protocol.
- GSM - for connecting to a GSM/GRPS network with the GSM shield.
- LiquidCrystal - for controlling liquid crystal displays (LCDs)
- SD - for reading and writing SD cards
- Servo - for controlling servo motors
- SPI - for communicating with devices using the Serial Peripheral Interface (SPI) Bus
- SoftwareSerial - for serial communication on any digital pins. Version 1.0 and later of Arduino incorporate Mikal Hart's NewSoftSerial library as SoftwareSerial.
- Stepper - for controlling stepper motors
- TFT - for drawing text , images, and shapes on the Arduino TFT screen
- WiFi - for connecting to the internet using the Arduino WiFi shield
- Wire - Two Wire Interface (TWI/I2C) for sending and receiving data over a net of devices or sensors.

Figur 7 Standard Libraries (11).

Bibliotek består av två eller flera filer: en headerfil (.h) och en källkodsfil (.cpp) och eventuellt en eller flera textfiler med förklaringar eller information om licensen.

Ethernet-biblioteket består av flera filer. Se figur 8. Vilka som används beror på vilken funktionalitet olika sketcher behöver.

Ethernet Library



examples	30.9.2013 12:03	File folder	
utility	30.9.2013 12:03	File folder	
Dhcp.cpp	17.5.2013 23:25	C++ Source	15 KB
Dhcp.h	17.5.2013 23:25	C/C++ Header	5 KB
Dns.cpp	17.5.2013 23:25	C++ Source	14 KB
Dns.h	17.5.2013 23:25	C/C++ Header	2 KB
Ethernet.cpp	17.5.2013 23:25	C++ Source	4 KB
Ethernet.h	17.5.2013 23:25	C/C++ Header	2 KB
EthernetClient.cpp	17.5.2013 23:25	C++ Source	4 KB
EthernetClient.h	17.5.2013 23:25	C/C++ Header	1 KB
EthernetServer.cpp	17.5.2013 23:25	C++ Source	2 KB
EthernetServer.h	17.5.2013 23:25	C/C++ Header	1 KB
EthernetUdp.cpp	17.5.2013 23:25	C++ Source	6 KB
EthernetUdp.h	17.5.2013 23:25	C/C++ Header	5 KB
keywords.txt	17.5.2013 23:25	Text Document	1 KB
util.h	17.5.2013 23:25	C/C++ Header	1 KB

Figur 8 Filer som ingår i Ethernet-biblioteket.

Det finns även bibliotek endast för vissa Arduino-kort: t.ex. Due, Esplora, Yún.

Det finns även bibliotek som användare har lagt till (Contributed Libraries).

Hittar flera bibliotek som kan användas för test av fukt- och temperatursensorn. Det finns ett bibliotek som heter dht22, men eftersom det är två år gammalt kommer jag i stället att använda ett som heter dht_lib som är nyare. Det stöder både dht11 och dht22.

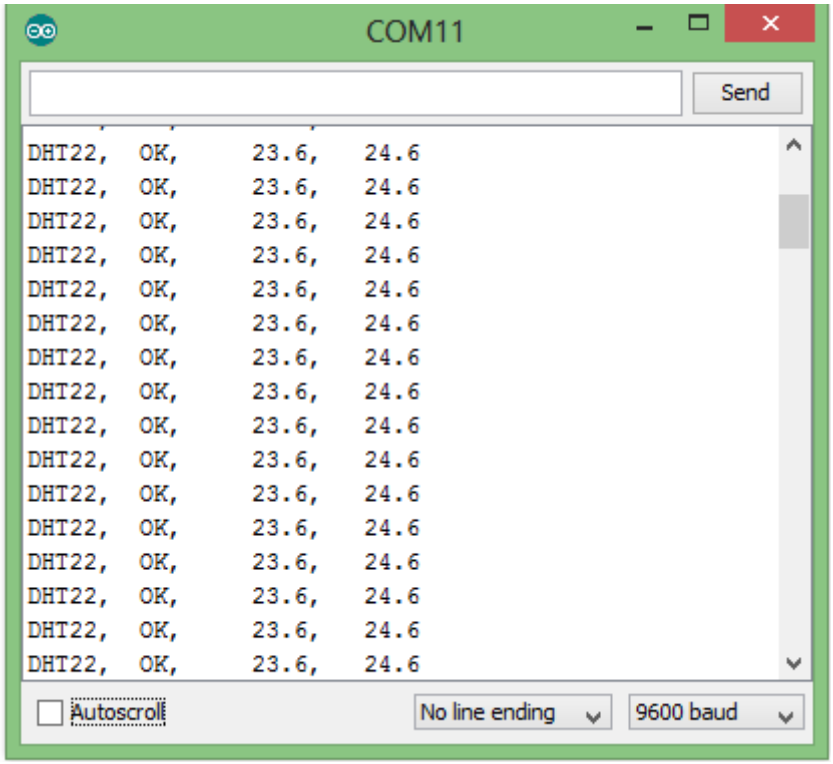
Det går att importera ytterligare bibliotek till Arduino om de är packade som zip-filer. Eftersom det här inte har någon zip-fil kommer jag att skapa en mapp DHT och i den skapa två filer dht.h och dht.cpp.

Instruktioner för att skapa ett bibliotek för DHT:

- Skapa först en mapp som heter DHT under C:\Users\Användare\Documents\Arduino\libraries
- Kopiera koden som heter dht.h till ett program som kan editera text t.ex. anteckningar. Klista in och spara som dht.h. Gör samma med dht.cpp.
- Placera dem i DHT-mappen.
- Skapa en ny sketch i Arduino IDE. Kom ihåg att inkludera <dht.h>

För att testa att biblioteket och temperatur- och fuktsensorn fungerar behövs en sketch för att testa det. Rob Tillaart som skrivit biblioteket inkluderar också en sketch för testning av biblioteket. Se bilaga 1.

Det enda som behöver göras nu är att skapa en ny sketch och klistra in koden. Eftersom mätvärdena skrivs ut med serial.print betyder det att något sorts konsolprogram behövs för att visa värdena. Lyckligtvis har Arduinos IDE ett som heter Serial Monitor. Figur 9 visar att mätvärdena skrivs ut och allt verkar vara Ok.



Figur 9 Serial Monitor.

6 Programmering

Efter all installation och testning är det dags att börja programmera. Nedan förklaras vilka metoder som använts för att få ett färdigt program.

6.1 Webbserver

Då det redan finns kod för fukt- och temperatursensorn och kod för en webbserver borde det gå att kombinera ihop dem och bara lägga till lite för att få ett färdigt program. Det går att starta från programmet som heter webbserver. Till det går det att lägga till DHT koden.

Här presenteras arbetsskedet för att kombinera ihop två sketcher:

1. Börjar med att se till att alla <include> finns i början av sketchen.
2. Lägger till DHT dht.
3. Kopplar in temperatur- och fuktsensorn på pin 6.
4. Anpassar IP-adressen för nätverket.
5. Sätter en port för webbservern.
6. Infogar kommandon för seriekommunikation för DHT.
7. Infogar koden för att läsa mätvärdena och felhantering.
8. Skriver koden för alarmfunktionen.
9. Skriver kod för att kunna visa mätningarna på webbsidan.
10. Rensar bort kod som visar information från analoga pinnarna.

Alla serial.print i DHT-koden har lämnats kvar för att lättare kunna felsöka om det skulle bli problem att ansluta via nätverket. De behövs inte för något annat. Serial,print kan läsas via en konsol som kopplas in via serieporten.

För att mätningarna ska kunna visas på en webbsida har jag kompletterar med HTML kod.

Se bilaga 3. Där finns alla punkter ovan insatta för att lättare kunna visa vad de är.

6.1.1 Funktion

När man surfar till IP-adressen på webbservern kommer webbläsaren att skicka en förfrågan. Informationen kan variera beroende på webbläsare och vilket operativsystem det skickas från. När servern har fått en förfrågan om en webbsida kommer servern att skicka ett standard HTML svar och själva webbsidan. Webbsidan kommer då att visas i webbläsaren. (12)

7 Alarmfunktionen

Alarmfunktionen konstruerades för att använda så lite minne som möjligt. Den kodades i C och en char vektor användes istället för en string. Alarmet kommer endast att visas på webbsidan. Om temperaturen överstiger 22°C visas texten Temperature HIGH och om temperaturen understiger 16°C visas Temperature LOW annars om temperaturen är mellan 16 - 22° C visas Temperature OK. Samma sak gjordes med fuktigheten, men den får variera mellan 20 - 40 %. Se figur 10 nedan.

Alarmkod

```
int tempAlarmLow = 16;
int tempAlarmHigh = 22;
char chT[6];
if (tempAlarmLow < DHT.temperature)
  strcpy(chT,"LOW");
if (tempAlarmHigh < DHT.temperature)
  strcpy(chT, "HIGH");
else
  strcpy(chT,"OK");

int humAlarmLow = 20;
int humAlarmHigh = 40;
char chH[6];
if (humAlarmLow < DHT.humidity)
  strcpy(chH,"LOW");
if (humAlarmHigh < DHT.humidity)
  strcpy(chH,"HIGH");
else
  strcpy(chH,"OK");
```

Figur 10 Alarmkod

7.1 Att välja mellan Arduinos strängar eller tecken vektorer

Arduinos inbyggda datatyp string är lättare att använda än C-teckenvektorer, men det uppnås genom komplex kod i stringbiblioteket, som gör att det ställs större krav på din Arduino och det är av sin beskaffenhet mera benägen för problem (13).

String datatypen är så flexibel därför att den använder dynamisk allokering av minnet. När du skapar eller modifierar en sträng begär Arduino ett nytt minnesblock från C-biblioteket och när du är färdig med strängen måste Arduino frigöra minnet. Det fungerar vanligtvis smidigt, men i praktiken finns det många sätt för minnesläckor att uppstå. Buggar i stringbiblioteket kan resultera i att en del eller att allt minne inte returneras. När det händer blir minnet som är tillgängligt för Arduino mindre och mindre, tills du startar om Arduino (13).

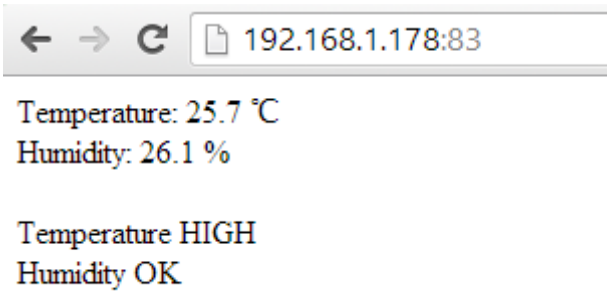
Om man använder en C-tecken vektor, så har man kontroll över minnesanvändningen, man kan allokera en bestämd statisk storlek på minnet vid kompileringen för att inte få några

minnesläckor. Arduino-programmet kommer att ha samma storlek på minnet tillgängligt så länge det körs.

Det är emellertid lättare att få ett annat problem med C-teckenvektorer: C hindrar inte att man kan modifiera minnet utanför gränsen för vektorn. Om man deklarerar en vektor myString[4] kan man ändå tilldela myString[5] = ”A” utan att någonting stoppar det. Observera att myString[3] är slutet på vektorn. För att inte få några problem med programmet måste man som programmerare själv hålla reda på slutet av vektorn.

8 Webbgränssnittet

Webbgränssnittet är mycket enkelt, men funktionellt. Se figur 11. Det visar mätvärdena för temperatur och fuktighet. Det visar också alarmfunktionen under mätningarna. Alarmfunktionen i figur 8 visar HIGH för temperatur och OK för fuktighet. Det betyder att temperaturen är över 22°C och att fuktigheten är mellan 20 - 40 %, se kapitel 7 för hur det fungerar.



Figur 11 Webbgränssnitt

Hela webbsidan uppdateras var femte sekund. Eftersom webbsidan innehåller så lite text medför det inga problem att uppdatera hela sidan. Det uppstår inget flimmer eller liknande.

Skulle webbsidan ha varit större borde t.ex. AJAX-teknik eller liknande användas för att uppdatera endast en del av sidan.

9 Pris

En viktig sak i alla projekt är hur mycket kommer det att kosta. Köper man alla komponenterna i Finland för Arduino-projektet kan det tillverkas för cirka 80 €. Priserna för

Arduino Uno, Ethernet-skölden och fukt- och temperatursensorn är från Månsteri Store, adress: http://store.mansteri.com/, som finns i Helsingfors. Resten utom laddaren är köpt från Vasa Elektronikcenter Ab, adress: http://www.vekoy.com/, som finns i Vasa. USB-laddaren är köpt från Hong Kongs affär i Vasa. Se tabell 4 härunder för mera exakta prisuppgifter. Priserna är inklusive moms 24 %.

Tabell 4 Kostnadsberäkning 1.

Kostnadsberäkning 1		
Produkt	**Antal**	**Pris**
Arduino Uno (R3)	1	24,50 €
Ethernet Shield	1	32,00 €
Humidity and Temperature Sensor - RHT03 (DHT22)	1	9,00 €
USB3-AB 1.8 m	1	3,91 €
USB laddare	1	5,95 €
Motstånd 1 kohm	1	0,20 €
Kabel CAT6 1m	1	0,45 €
Frakt		3,95 €
----------	-------	----------
Summa		79,96 €

Tillverkar man bara en enhet har priset inte så stor betydelse, men ju flera enheter man tillverkar desto större betydelse får priset. För att ha något att jämföra med har jag hämtat priser från EBay adress: ebay.com. Det skiljer över 50 € om man köper från Kina istället för Finland. Detta betyder att man kan tillverka tre enheter istället för en för samma pris. Köper man från Kina får man varorna hemskickade fraktfritt.

I tabell 5 har USB-laddaren bytts till en nätadapter laddare med en 2.1 mm plugg.

Tabell 5 Kostnadsberäkning 2.

Kostnadsberäkning 2			
Produkt	**Antal**	**Pris**	
Arduino Uno (R3) + USB kabel	1	8,66 €	Kina
Ethernet Shield	1	9,81 €	Kina
Humidity and Temperature Sensor - RHT03 (DHT22)	1	3,40 €	Kina
DC 9V/1A Power Supply	1	3,30 €	Kina
Motstånd 1 kohm	1	0,20 €	
Kabel CAT6 1m	1	0,45 €	
Frakt		0,00 €	
---	---	---	
Summa		25,82 €	

Priserna för väderstationen har hämtats från Ilkka Lilja Oy Ltd, adress http://www.iloy.fi, som finns i Jyväskylä. Kostnaderna(se tabell 6) jämfört med en Arduino köpt i Finland är tio gånger högre. Det är en skillnad i pris på 720 €.

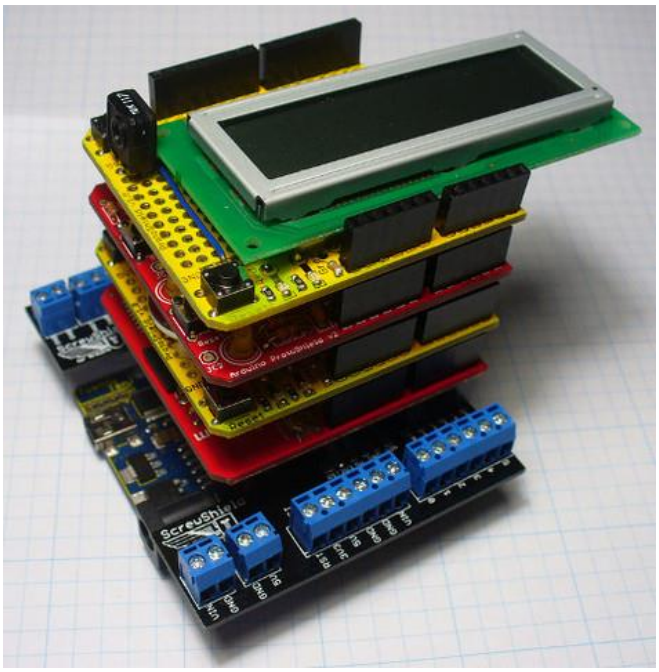
Tabell 6 Kostnadsberäkning 3.

Kostnadsberäkning 3		
Produkt	**Antal**	**Pris**
Wireless Vantage Pro2(tm)	1	535,00 €
WeatherLinkIP(tm)	1	250,00 €
Frakt	1	15,00 €
---	---	---
Summa		800,00 €

10 Vidareutveckling

Skulle det bli aktuellt med komplettering eller ändring av hårdvaran finns en lista i bilaga 4 med 317 sköldar från 125 olika tillverkare. Det betyder att det finns nästan ett oändligt antal olika användningsområden.

Det går också att stapla många olika sköldar ovanpå varandra för att kombinera funktionalitet från olika sköldar. Figur 12 visar ett exempel. Det kan jämföras med att till en dator t.ex. sätta till ett grafikkort eller liknande.

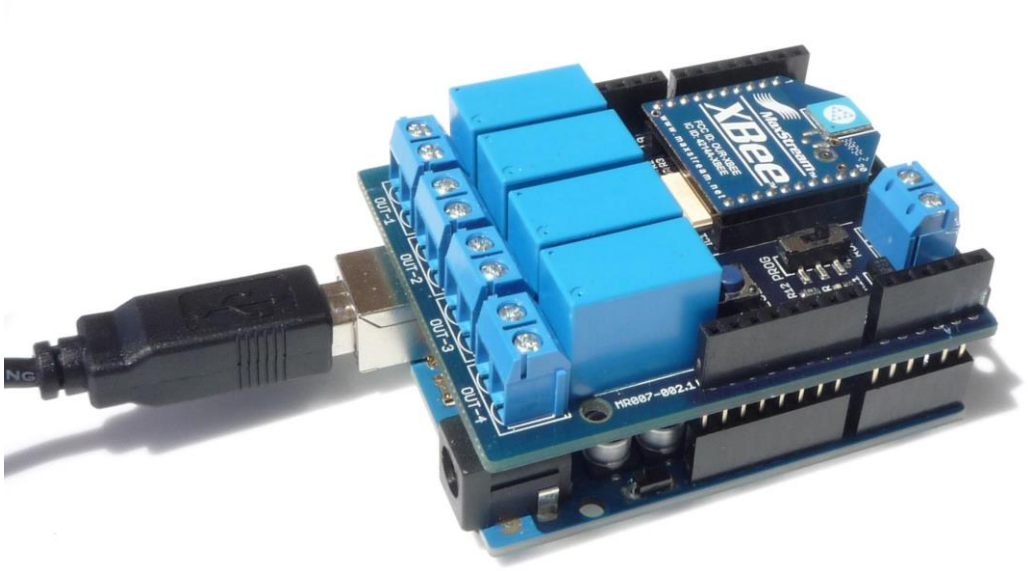


Figur 12 Exempel på många sköldar ovanpå varandra (14).

Vad som kanske är mest intressant för det här projektet är en reläsköld, se figur 13. Med den kan man kontrollera högre spänningar. Det går att bryta en spänning på 250 V. Det går att koppla in en elektrisk anordning som har en effekt på ca 1 kW på var och en av de fyra kanalerna. Se tabell 7 för specifikationer.

Tabell 7 Specifikationer MR007-002.1 (för varje kanal) (15).

Supply voltage	12VDC
Supply current	200mA (max.)
Rated voltage	30VDC, 250VAC
Rated current	5A
Electrical Life Expectancy	100000 operations
Mechanical Life Expectancy	10000000 operations
Operate time	10ms Max.
Release time	4ms Max.
Operating temperature	-30°C / +70°C
Dimensions	2.7” x 2.2” (69 x 55 mm)
Weight	1.73 oz (49 gr)



Figur 13 Relä sköld (15).

10.1 Vad man kan använda det till.

Det går ganska enkelt att konstruera en automatisering för klimatkontroll. Eftersom det går att koppla in fyra stycken värmeelement som vardera kan vara 1 kW, så är det inga problem att reglera temperaturen.

På webbsidan behöver det läggas till två komponenter där man kan välja eller skriva in vilken högsta och lägsta temperatur som man önskar. I själva sketchen behövs det en funktion för att hantera informationen från webbsidan. Det går i princip att använda koden för alarmfunktionen (se kapitel 7). Ett värde sätts när reläet ska öppna och ett annat när det ska stänga. Det går att kontrollera varje värmeelement enskilt eller alla tillsammans.

Figur 14 visar ett kodexempel som är så enkelt som möjligt. Det bygger på blink exemplet från kapitel 3.3 kombinerat med alarmfunktionen från kapitel 7. Det har inte testats med någon hårdvara, utan är bara ett exempel på hur man kan använda en reläsköld.

testRelay | Arduino 1.0.5

File Edit Sketch Tools Help

testRelay

```
#include <SPI.h>
#include <dht.h>

// Definiera variabler
int rel1 = 5; // definiera Arduino pin anslutet till relä 1
int rel2 = 6; // definiera Arduino pin anslutet till relä 2
int rel3 = 7; // definiera Arduino pin anslutet till relä 3
int rel4 = 8; // definiera Arduino pin anslutet till relä 4

int tempLow = 18;       // hårdkodad för test
int tempHigh = 20;      // hårdkodad för test

dht DHT;

//  Initsialisering
void setup()
{
  pinMode(rel1, OUTPUT); // initsialisera digital pin som output
  pinMode(rel2, OUTPUT); // initsialisera digital pin som output
  pinMode(rel3, OUTPUT); // initsialisera digital pin som output
  pinMode(rel4, OUTPUT); // initsialisera digital pin som output
}

void loop()
{
        if (DHT.temperature < tempLow)
        {
                digitalWrite(rel1, HIGH);   // På
                digitalWrite(rel2, HIGH);   // På
                digitalWrite(rel3, HIGH);   // På
                digitalWrite(rel4, HIGH);   // På
        }
        if (DHT.temperature < tempHigh)
        {
                digitalWrite(rel1, LOW);    // AV
                digitalWrite(rel2, LOW);    // AV
                digitalWrite(rel3, LOW);    // AV
                digitalWrite(rel4, LOW);    // AV
        }
}
```

Done Saving.

33 Arduino Uno on COM11

Figur 14 Exempel på sketch för test av relä.

11 Slutsats

Projektet har blivit en fungerande webbsida där temperatur och fukt kan avläsas från en pc eller smarttelefon. Inga stora problem uppstod under projektets gång. Det fanns planer från början att lagra mätningarna i en databas, men under arbetets gång framkom att det inte fanns något behov av detta. Det beslutades också att inga alarm skulle skickas vidare utan visas direkt på webbsidan.

Mitt examensarbete visar att man kan komma ganska långt med att använda kod som redan finns för Arduino. Om det inte är ett mycket speciellt projekt finns det redan någon som har gjort det förut eller något liknande. Fördelen med att använda kod som redan finns är att det går snabbt att komma igång. En annan fördel är att koden har blivit testad med hårdvaran. Det betyder att man inte behöver så mycket tid för själva testningsfasen. Det ger mera tid till att anpassa för egna behov.

12 Referenser

1. *Försöket med torkning av råskinn slutfört.* Heikki Jäykäla. 11, u.o. : Finsk Pälstidskrift, 1999. ISSN 0430-5817.

2. Arduino. *Wikipedia.* [Online] [Citat: den 23 1 2014.] http://sv.wikipedia.org/wiki/Arduino.

3. Böhmer, Mario. *Beginning Android ADK with Arduino.* u.o. : Apress, 2012. ISBN-13 (pbk) 978-1-4302-4197-3.

4. Arduino Uno. [Online] [Citat: den 24 1 2014.] http://arduino.cc/en/Main/arduinoBoardUno.

5. Arduino Ethernet Shield. *arduino.cc.* [Online] [Citat: den 25 1 2014.] http://arduino.cc/en/Main/ArduinoEthernetShield.

6. Liu, Thomas. Maxdetect. [Online] [Citat: den 26 1 2014.] http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Sensors/Weather/RHT03.pdf.

7. Arduino Development Environment. [Online] [Citat: den 28 1 2014.] http://arduino.cc/en/guide/Environment.

8. BOXTEC. *BOXTEC Playground.* [Online] [Citat: den 15 4 2014.] http://playground.boxtec.ch/doku.php/sensors/temp-hum_sensors_compared.

9. Tutorial: Humidity and Temperature Sensor with Arduino. *GarageLab.* [Online] [Citat: den 29 1 2014.] http://garagelab.com/profiles/blogs/tutorial-humidity-and-temperature-sensor-with-arduino.

10. The Arduino Playground. [Online] [Citat: den 29 3 2014.] http://playground.arduino.cc/.

11. Libraries. *Arduino.* [Online] [Citat: den 1 4 2014.] http://arduino.cc/en/Reference/Libraries.

12. Basic Arduino Web Server. [Online] [Citat: den 5 2 2014.] http://startingelectronics.com/tutorials/arduino/ethernet-shield-web-server-tutorial/basic-web-server/.

13. Margolis, Michael. *Arduino Cookbook, Second Edition.* u.o. : O'Reilly, 2012. ISBN 978-1-4493-1387-6.

14. Stacking Arduino Shields. *Freetronics.* [Online] [Citat: den 15 4 2014.] http://www.freetronics.com/pages/stacking-arduino-shields#.U008sPl_tvk.

15. Relay Shield. [Online] [Citat: den 7 4 2014.] http://www.microbot.it/en/product/62/Relay-Shield.html.

16. Arduino Shield List. *shieldlist.org.* [Online] [Citat: den 7 4 2014.] http://shieldlist.org/.

17. Arduino Ethernet Shield Tutorial. [Online] [Citat: den 15 4 2014.] http://www.airy08.com.

19. MånsteriStore. [Online] http://store.mansteri.com/index.php/fi/sensors/environment/humidity-and-temperature-sensor-rht03.html.

20. Kruger, Bertus. Protoneer. [Online] [Citat: den 15 4 2014.] http://blog.protoneer.co.nz/arduino-more-core-shield/.

Bilaga 1

```
Usage
A sketch shows how the library can be used to read the sensor.
--- Note: the max frequency the sensor can be sampled is about once per 2
seconds
//
//    FILE: dht_test.ino
//  AUTHOR: Rob Tillaart
// VERSION: 0.1.07
// PURPOSE: DHT Temperature & Humidity Sensor library for Arduino
//     URL: http://arduino.cc/playground/Main/DHTLib
//
// Released to the public domain
//

#include <dht.h>

dht DHT;

#define DHT11_PIN 4
#define DHT21_PIN 5
#define DHT22_PIN 6

void setup()
{
  Serial.begin(115200);
  Serial.println("DHT TEST PROGRAM ");
  Serial.print("LIBRARY VERSION: ");
  Serial.println(DHT_LIB_VERSION);
  Serial.println();
  Serial.println("Type,\tstatus,\tHumidity (%),\tTemperature (C)");
}

void loop()
{
  // READ DATA
  Serial.print("DHT22, \t");
  int chk = DHT.read22(DHT22_PIN);
  switch (chk)
  {
    case DHTLIB_OK:
        Serial.print("OK,\t");
        break;
    case DHTLIB_ERROR_CHECKSUM:
        Serial.print("Checksum error,\t");
        break;
    case DHTLIB_ERROR_TIMEOUT:
        Serial.print("Time out error,\t");
        break;
    default:
        Serial.print("Unknown error,\t");
        break;
  }
  // DISPLAY DATA
  Serial.print(DHT.humidity, 1);
  Serial.print(",\t");
  Serial.println(DHT.temperature, 1);

  delay(1000);
```

```
    // READ DATA
  Serial.print("DHT21, \t");
  chk = DHT.read21(DHT21_PIN);
  switch (chk)
  {
    case DHTLIB_OK:
        Serial.print("OK,\t");
        break;
    case DHTLIB_ERROR_CHECKSUM:
        Serial.print("Checksum error,\t");
        break;
    case DHTLIB_ERROR_TIMEOUT:
        Serial.print("Time out error,\t");
        break;
    default:
        Serial.print("Unknown error,\t");
        break;
  }
  // DISPLAY DATA
  Serial.print(DHT.humidity, 1);
  Serial.print(",\t");
  Serial.println(DHT.temperature, 1);

  delay(1000);

  // READ DATA
  Serial.print("DHT11, \t");
  chk = DHT.read11(DHT11_PIN);
  switch (chk)
  {
    case DHTLIB_OK:
        Serial.print("OK,\t");
        break;
    case DHTLIB_ERROR_CHECKSUM:
        Serial.print("Checksum error,\t");
        break;
    case DHTLIB_ERROR_TIMEOUT:
        Serial.print("Time out error,\t");
        break;
    default:
        Serial.print("Unknown error,\t");
        break;
  }
 // DISPLAY DATA
  Serial.print(DHT.humidity,1);
  Serial.print(",\t");
  Serial.println(DHT.temperature,1);

  delay(1000);
}
//
// END OF FILE
//
```

dht.h

```
//
//    FILE: dht.h
//  AUTHOR: Rob Tillaart
// VERSION: 0.1.09
// PURPOSE: DHT Temperature & Humidity Sensor library for Arduino
//     URL: http://arduino.cc/playground/Main/DHTLib
//
// HISTORY:
// see dht.cpp file
//

#ifndef dht_h
#define dht_h

#if ARDUINO < 100
#include <WProgram.h>
#else
#include <Arduino.h>
#endif

#define DHT_LIB_VERSION "0.1.09"

#define DHTLIB_OK                0
#define DHTLIB_ERROR_CHECKSUM   -1
#define DHTLIB_ERROR_TIMEOUT    -2
#define DHTLIB_INVALID_VALUE    -999

class dht
{
public:
    int read11(uint8_t pin);
    int read21(uint8_t pin);
    int read22(uint8_t pin);

    double humidity;
    double temperature;

private:
    uint8_t bits[5];  // buffer to receive data
    int read(uint8_t pin);
};
#endif
//
// END OF FILE
//
```

dht.cpp

```
//
//    FILE: dht.cpp
//  AUTHOR: Rob Tillaart
// VERSION: 0.1.08
// PURPOSE: DHT Temperature & Humidity Sensor library for Arduino
//     URL: http://arduino.cc/playground/Main/DHTLib
//
// HISTORY:
// 0.1.09 optimize size: timeout check + use of mask
// 0.1.08 added formula for timeout based upon clockspeed
// 0.1.07 added support for DHT21
// 0.1.06 minimize footprint (2012-12-27)
// 0.1.05 fixed negative temperature bug (thanks to Roseman)
// 0.1.04 improved readability of code using DHTLIB_OK in code
// 0.1.03 added error values for temp and humidity when read failed
// 0.1.02 added error codes
// 0.1.01 added support for Arduino 1.0, fixed typos (31/12/2011)
// 0.1.0 by Rob Tillaart (01/04/2011)
//
// inspired by DHT11 library
//
// Released to the public domain
//

#include "dht.h"

// #define TIMEOUT 10000
// uint16_t for UNO, higher CPU speeds => exceed MAXINT.
// works for DUE
// 16 MHz => 10000
// 84 MHz => 52500
// 100MHz => 62500
#define TIMEOUT (F_CPU/1600)


////////////////////////////////////////////////
//
// PUBLIC
//

// return values:
// DHTLIB_OK
// DHTLIB_ERROR_CHECKSUM
// DHTLIB_ERROR_TIMEOUT
int dht::read11(uint8_t pin)
{
    // READ VALUES
    int rv = read(pin);
    if (rv != DHTLIB_OK)
    {
        humidity    = DHTLIB_INVALID_VALUE; // invalid value, or is NaN
prefered?
        temperature = DHTLIB_INVALID_VALUE; // invalid value
        return rv;
    }

    // CONVERT AND STORE
    humidity    = bits[0];  // bits[1] == 0;
    temperature = bits[2];  // bits[3] == 0;

    // TEST CHECKSUM
```

```
    // bits[1] && bits[3] both 0
    uint8_t sum = bits[0] + bits[2];
    if (bits[4] != sum) return DHTLIB_ERROR_CHECKSUM;

    return DHTLIB_OK;
}

// return values:
// DHTLIB_OK
// DHTLIB_ERROR_CHECKSUM
// DHTLIB_ERROR_TIMEOUT
int dht::read21(uint8_t pin)
{
    return read22(pin);  // dataformat identical to DHT22
}

// return values:
// DHTLIB_OK
// DHTLIB_ERROR_CHECKSUM
// DHTLIB_ERROR_TIMEOUT
int dht::read22(uint8_t pin)
{
    // READ VALUES
    int rv = read(pin);
    if (rv != DHTLIB_OK)
    {
        humidity    = DHTLIB_INVALID_VALUE;  // invalid value, or is NaN
prefered?
        temperature = DHTLIB_INVALID_VALUE;  // invalid value
        return rv; // propagate error value
    }

    // CONVERT AND STORE
    humidity = word(bits[0], bits[1]) * 0.1;

    if (bits[2] & 0x80) // negative temperature
    {
        temperature = -0.1 * word(bits[2] & 0x7F, bits[3]);
    }
    else
    {
        temperature = 0.1 * word(bits[2], bits[3]);
    }

    // TEST CHECKSUM
    uint8_t sum = bits[0] + bits[1] + bits[2] + bits[3];
    if (bits[4] != sum) return DHTLIB_ERROR_CHECKSUM;

    return DHTLIB_OK;
}

/////////////////////////////////////////////////////
//
// PRIVATE
//

// return values:
// DHTLIB_OK
// DHTLIB_ERROR_TIMEOUT
int dht::read(uint8_t pin)
{
    // INIT BUFFERVAR TO RECEIVE DATA
```

```
    uint8_t mask = 128;
    uint8_t idx = 0;

    // EMPTY BUFFER
    for (uint8_t i=0; i< 5; i++) bits[i] = 0;

    // REQUEST SAMPLE
    pinMode(pin, OUTPUT);
    digitalWrite(pin, LOW);
    delay(20);
    digitalWrite(pin, HIGH);
    delayMicroseconds(40);
    pinMode(pin, INPUT);

    // GET ACKNOWLEDGE or TIMEOUT
    unsigned int loopCnt = TIMEOUT;
    while(digitalRead(pin) == LOW)
    if (--loopCnt == 0) return DHTLIB_ERROR_TIMEOUT;

    loopCnt = TIMEOUT;
    while(digitalRead(pin) == HIGH)
    if (--loopCnt == 0) return DHTLIB_ERROR_TIMEOUT;

    // READ THE OUTPUT - 40 BITS => 5 BYTES
    for (uint8_t i=0; i<40; i++)
    {
        loopCnt = TIMEOUT;
        while(digitalRead(pin) == LOW)
        if (--loopCnt == 0) return DHTLIB_ERROR_TIMEOUT;

        unsigned long t = micros();

        loopCnt = TIMEOUT;
        while(digitalRead(pin) == HIGH)
        if (--loopCnt == 0) return DHTLIB_ERROR_TIMEOUT;

        if ((micros() - t) > 40) bits[idx] |= mask;
        mask >>= 1;
        if (mask == 0)   // next byte?
        {
            mask = 128;
            idx++;
        }
    }

    return DHTLIB_OK;
}
//
// END OF FILE
//
```

Bilaga 2

```
/*

  Web Server

 A simple web server that shows the value of the analog input pins.
 using an Arduino Wiznet Ethernet shield.

 Circuit:
 * Ethernet shield attached to pins 10, 11, 12, 13
 * Analog inputs attached to pins A0 through A5 (optional)

 created 18 Dec 2009
 by David A. Mellis
 modified 9 Apr 2012
 by Tom Igoe

 */

#include <SPI.h>
#include <Ethernet.h>

// Enter a MAC address and IP address for your controller below.
// The IP address will be dependent on your local network:
byte mac[] = {
  0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
IPAddress ip(192,168,1,177);

// Initialize the Ethernet server library
// with the IP address and port you want to use
// (port 80 is default for HTTP):
EthernetServer server(80);

void setup() {
 // Open serial communications and wait for port to open:
  Serial.begin(9600);
   while (!Serial) {
     ; // wait for serial port to connect. Needed for Leonardo only
  }


  // start the Ethernet connection and the server:
  Ethernet.begin(mac, ip);
  server.begin();
  Serial.print("server is at ");
  Serial.println(Ethernet.localIP());
}


void loop() {
  // listen for incoming clients
  EthernetClient client = server.available();
  if (client) {
    Serial.println("new client");
    // an http request ends with a blank line
    boolean currentLineIsBlank = true;
    while (client.connected()) {
      if (client.available()) {
        char c = client.read();
```

```
        Serial.write(c);
        // if you've gotten to the end of the line (received a newline
        // character) and the line is blank, the http request has ended,
        // so you can send a reply
        if (c == '\n' && currentLineIsBlank) {
          // send a standard http response header
          client.println("HTTP/1.1 200 OK");
          client.println("Content-Type: text/html");
          client.println("Connection: close");  // the connection will be
closed after completion of the response
      client.println("Refresh: 5");  // refresh the page automatically
every 5 sec
          client.println();
          client.println("<!DOCTYPE HTML>");
          client.println("<html>");
          // output the value of each analog input pin
          for (int analogChannel = 0; analogChannel < 6; analogChannel++)
{
            int sensorReading = analogRead(analogChannel);
            client.print("analog input ");
            client.print(analogChannel);
            client.print(" is ");
            client.print(sensorReading);
            client.println("<br />");
          }
          client.println("</html>");
          break;
        }
        if (c == '\n') {
          // you're starting a new line
          currentLineIsBlank = true;
        }
        else if (c != '\r') {
          // you've gotten a character on the current line
          currentLineIsBlank = false;
        }
      }
    }
    // give the web browser time to receive the data
    delay(1);
    // close the connection:
    client.stop();
    Serial.println("client disonnected");
  }
}
```

Bilaga 3

```
// Kombination av Web Server och DHT

/*
  Web Server

 A simple web server that shows the value of the analog input pins.
 using an Arduino Wiznet Ethernet shield.

 Circuit:
 * Ethernet shield attached to pins 10, 11, 12, 13
 * Analog inputs attached to pins A0 through A5 (optional)

 created 18 Dec 2009
 by David A. Mellis
 modified 9 Apr 2012
 by Tom Igoe

 */

 //
//    FILE: dht_test.ino
//  AUTHOR: Rob Tillaart
// VERSION: 0.1.07
// PURPOSE: DHT Temperature & Humidity Sensor library for Arduino
//     URL: http://arduino.cc/playground/Main/DHTLib
//
// Released to the public domain
//
 // 1. <include>
#include <SPI.h>
#include <Ethernet.h>
#include <dht.h>
// 2.
dht DHT;
// 3. temperatur- och fuktsensorn är inkopplad på pin 6
#define DHT22_PIN 6
// Enter a MAC address and IP address for your controller below.
// The IP address will be dependent on your local network:
byte mac[] = {
  0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
// 4. IP address för nätverket
IPAddress ip(192,168,1,178);

// Initialize the Ethernet server library
// with the IP address and port you want to use
// (port 80 is default for HTTP):
// 5. Port för HTTP
EthernetServer server(83);

void setup() {
// Open serial communications and wait for port to open:
  Serial.begin(9600);
   while (!Serial) {
    ; // wait for serial port to connect. Needed for Leonardo only
   // 6. Serie kommunikation för DHT
   Serial.println("DHT TEST PROGRAM ");
   Serial.print("LIBRARY VERSION: ");
   Serial.println(DHT_LIB_VERSION);
```

```
   Serial.println();
   Serial.println("Type,\tstatus,\tHumidity (%),\tTemperature (C)");
  }
  // start the Ethernet connection and the server:
  Ethernet.begin(mac, ip);
  server.begin();
  Serial.print("server is at ");
  Serial.println(Ethernet.localIP());
}
void loop() {
  // 7. Läser in data och felhantering
  // READ DATA
  Serial.print("DHT22, \t");
  int chk = DHT.read22(DHT22_PIN);
  switch (chk)
  {
    case DHTLIB_OK:
        Serial.print("OK,\t");
        break;
    case DHTLIB_ERROR_CHECKSUM:
        Serial.print("Checksum error,\t");
        break;
    case DHTLIB_ERROR_TIMEOUT:
        Serial.print("Time out error,\t");
        break;
    default:
        Serial.print("Unknown error,\t");
        break;
  }
  // DISPLAY DATA
  Serial.print(DHT.humidity, 1);
  Serial.print(",\t");
  Serial.println(DHT.temperature, 1);

  delay(1000);

  // 8. ALARM
  int tempAlarmLow = 16;
  int tempAlarmHigh = 22;
  char chT[6];
  if (tempAlarmLow < DHT.temperature)
    strcpy(chT,"LOW");
  if (tempAlarmHigh < DHT.temperature)
    strcpy(chT, "HIGH");
  else
    strcpy(chT,"OK");

  int humAlarmLow = 20;
  int humAlarmHigh = 40;
  char chH[6];
  if (humAlarmLow < DHT.humidity)
    strcpy(chH,"LOW");
  if (humAlarmHigh < DHT.humidity)
    strcpy(chH,"HIGH");
  else
    strcpy(chH,"OK");


  // listen for incoming clients
  EthernetClient client = server.available();
  if (client) {
    Serial.println("new client");
```

```
    // an http request ends with a blank line
    boolean currentLineIsBlank = true;
    while (client.connected()) {
      if (client.available()) {
        char c = client.read();
        Serial.write(c);
        // if you've gotten to the end of the line (received a newline
        // character) and the line is blank, the http request has ended,
        // so you can send a reply
        if (c == '\n' && currentLineIsBlank) {
          // send a standard http response header
          client.println("HTTP/1.1 200 OK");
          client.println("Content-Type: text/html");
          client.println("Connection: close");  // the connection will be
closed after completion of the response
      client.println("Refresh: 5");  // refresh the page automatically
every 5 sec
          client.println();
          client.println("<!DOCTYPE HTML>");
          client.println("<html>");
          // 9. Visar mätningar en webläsare
          client.print("Temperature ");
          client.print(DHT.temperature,1);
          client.print(" &#x2103");
          client.print("<br />");
          client.print("Humidity ");
          client.print(DHT.humidity,1);
          client.print(" %");

          client.print("<br />");
          client.print("<br />");

          client.print("Temperature ");
          client.print(chT);
          client.print("<br />");
          client.print("Humidity ");
          client.print(chH);
          // 10. Kod som tas bort
          // output the value of each analog input pin
          /*
        for (int analogChannel = 0; analogChannel < 6; analogChannel++) {
            int sensorReading = analogRead(analogChannel);
            client.print("analog input ");
            client.print(analogChannel);
            client.print(" is ");
            client.print(sensorReading);
            client.println("<br />");
          */
          client.println("</html>");
          break;
        }
        if (c == '\n') {
          // you're starting a new line
          currentLineIsBlank = true;
        }
        else if (c != '\r') {
          // you've gotten a character on the current line
          currentLineIsBlank = false;
        }
      }
    }
    // give the web browser time to receive the data
```

```
    delay(1);
    // close the connection:
    client.stop();
    Serial.println("client disonnected");
  }
}
```

Bilaga 4

- 4D Systems (4)
- Adafruit Industries (9)
- AeroQuad (2)
- Andre Concalves (2)
- antrax Datentechnik (1)
- Applied Platonics (1)
- ArduCapSense (1)
- Arduino (3)
- Argent Data Systems (1)
- AsyncLabs (3)
- Batsocks (2)
- Ben Combee (1)
- Bhasha Technologies (2)
- Bliptronics (2)
- Blushing Boy (1)
- Carlos Neves (1)
- Chesters Garage (1)
- Chips To Bits (1)
- Circuit Ideas Design (1)
- Circuits At Home (2)
- CISECO (4)
- Collin Schulz (1)
- Conductive Resistance (1)
- Control Connection (1)
- Creatron Inc (1)
- Critical Velocity (5)
- Critter and Guitari (1)
- Curious Inventor (2)
- CuteDigi (8)
- Damien Good (1)
- Dexter Industries (1)
- DFRobot (16)
- Dr Michael Kroll (1)
- Dreaming Robots (1)
- DroidBuilder (1)
- DSS Circuits (1)
- Dynamic Perception (1)
- Emartee (1)
- Excamera Labs (1)
- Faz Jaxton (1)
- FlamingoEDA (1)
- Freetronics (12)
- Futura Elettronica (2)
- Galileo 7 (4)
- GeekOnFire (1)
- GfxHax (1)
- GinSing (1)
- Gravitech (1)
- Homeroasters (1)
- HW Kitchen (1)
- ITead Studio (6)
- Jee Labs (1)
- Jimmie Rodgers (1)
- John Liu (3)
- Knutsel (1)
- Lars Schumann (3)
- Libelium (6)
- LinkSprite (5)
- Liquidware (12)
- Logos Electromechanical(1)
- Low Voltage Labs (1)
- Luke Weston (1)
- macetech (3)
- Maker Shed (1)
- Mark Sproul (1)
- Max Pierson (1)
- Mayhew Labs (2)
- MCI Electronics (4)
- McLaughlin Engineering (1)
- MightyOhm (2)
- Mitek (1)
- Modern Device (1)
- Multilogica (1)
- Narbotic Instruments (1)
- Neuroelec (2)
- NKC Electronics (4)
- Nootropic Design (3)
- North And Nash (1)
- Nu Electronics (9)
- Ocean Controls (1)
- Open Electronics (1)
- PDK Solutions (1)
- Photoduino (1)
- Pololu (1)
- Practical Maker (5)
- Protuino (2)
- Ray's Hobby (1)
- Renbotics (2)
- RepRap Research Foundation (1)
- Ro-Bot-X Designs (1)
- Robot Power (1)
- RobotPirate (1)
- Rocket Scream (1)
- Rogue Robotics (1)
- Rugged Circuits (6)
- Samurai Circuits (3)
- Scattered Mind (1)
- Schmelle2 (6)
- Seeed Studio (14)
- Shieldstudio (2)
- SK Pang Electronics (3)
- Small Room Labs (1)
- Smart Energy Groups (1)
- Snootlab (6)
- Solarbotics (2)
- SonikTech (1)
- Sparkfun (26)
- SpikenzieLabs (4)
- Sunhayato (2)
- SWFLTEK (1)
- Synthetos (1)
- Unified Microsystems (1)
- Unsped (1)
- Watterott (3)
- Wavical Technologies (1)
- Wayne and Layne (1)
- Wicked Device (3)

- Eric Rogers (1)
- EtherMania (1)
- Evil Mad Science (2)
- Wingshield Industries (1)
- Wise Time (2)
- Yawp (2)
- Zach Hoeken (1)

Arduino shields (16)