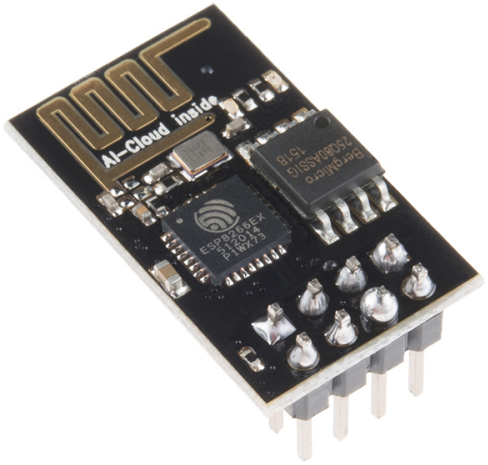
CHALMERS



Implementering av wifi med modulen ESP-01

Implementation of wifi with the module ESP-01

Examensarbete för högskoleingenjörsexamen inom elektroteknik

Henning Eggertz

Josefine Strömsten

Instutitionen för elektroteknik
Chalmers tekniska högskola
Gothenburg, Sweden 2019

Abstract

The company Trygghetsvakten manufactures dehumidifiers for creep foundations, with the purpose to reduce mold growth. The placement of the dehumidifier makes it hard to access, which make it difficult to obtain data from it. Therefore the company wants an improved product where it is easier to receive data from the dehumidifier, which is temperature, humidity and the current status. This thesis examines how data can be sent wirelessly from the dehumidifier to a server, where a user can read the information. The result of this project was a prototype where the wifi module ESP01 was implemented on the original circuit board on the dehumidifier. The data is sent in real time to the temporary server Thingspeak. This project has given an understanding in the area of microcontrollers, IoT units and programming.

Keywords: Dehumidifier, Trygghetsvakten, Arduino, ESP8266

Sammanfattning

Luftavfuktare i krypgrunder används för att motverka mögelpåväxt. Trygghetsvakten är ett företag som producerar och installerar luftavfuktare i krypgrunder för att motverka att mögel ska uppkomma. Då enheten installeras i svåråtkomliga miljöer finns det för avsikt att kunna kontrollera enheten utan att fysiskt vara nere i krypgrunden. Detta arbete grundar sig i att undersöka vilka möjligheter som finns för att skicka data trådlöst från luftavfuktaren till en server, där en användare kan läsa av i vilket tillstånd luftavfuktaren befinner sig i. Arbetet resulterade i en prototyp där wifi-modulen ESP01 implementerades på luftavfuktaren som Trygghetsvakten tillhandahåller. Datan från luftavuktaren skickades i realtid till en temporär server. Arbetet har gett en grundlig förståelse av microcontrollers, användningsområden för IoT enheter och även olika sätt att programmera olika processorer.

Nyckelord: Luftavfuktre, Trygghetsvakten, Arduino, ESP8266

Förord

Detta examensarbete är det avslutande momentet för utbildningen elektroteknik på Chalmers tekniska högskola. Arbetet omfattar 15 hp över 20 veckor, vilket har utförts på företaget Trygghetsvakten.

Vi vill ge ett stort tack till följande personer:

- Johan Östman, PhD Student inom elektroteknik
- Patrik Keussen, VD Amrox group AB
- Vilhelm Sjölander, graphic Web Designer

Vi skulle även vilja tacka Sparkfun Electroniks för tillåtelsen att använda deras bild av ESP01 till vår framsida[1].

Henning Eggertz
Josefine Strömsten
Göteborg 2019

Terminologi

A/D-omvandlare - Analog till digital-omvandlare

AT-kommandon - Används av extern utrustning för att styra modem

APL - Arduino Programming Language

Baud rate - Bitström som mäts i Bit/s

GPIO - General Purpose Input Output

IoT - Internet of Things

NTC - Negative Temperature Coefficient

TV-kort - Trygghetsvaktenkort

UART - Universal Asynchronous Receiver/Transmitter

I^2C - Inter-Integrated Circuit

Innehåll

1 Introduktion

Detta kapitel ger en introduktion till vad detta examensarbete kommer att handla om, bakgrunden till arbetet och avgränsningar för projektet.

1.1 Bakgrund

Att reglera luftfuktigheten i en krypgrund är viktigt för att hålla ett hus fritt från mögel och andra fuktrelaterade problem. Ett sätt att kontrollera luftfuktigheten är genom att installera en enhet som läser av temperatur och luftfuktighet och, utifrån dessa, styr en värmeslinga i krypgrunden. Företaget Trygghetsvakten har en sådan produkt på marknaden som har varit i drift i flera år som har visat goda resultat. Företaget vill vidareutveckla denna befintliga produkt med en trådlös överföring, för att kunna långtidslagra data från sensorerna i realtid för att göra det möjligt att föra statistik under en längre tid. Detta ska göras med en wifi-uppkoppling som används för att skicka dem mätvärden enheten har samplat till en server för presentation i ett grafiskt gränssnitt. Datan som registreras av sensorerna beslutar vilket tillstånd värmeslingan ska vara i. Den aktuella statusen visas för användaren med hjälp av tre stycken LED-indikatorer som sitter på produkten. Temperatur, luftfuktighet och status är de parametrar som ska lagras på servern.

Huvudfokus för företaget är implementeringen utav hårdvaran ESP01 på det befintliga kretskortet som heter Trygghetsvakten-kortet, även kallat TV-kort. Då produkten sitter i varierande miljöer, beroende på geografisk plats, över hela Sverige kommer produkten behöva operera över stora temperaturspann, därför behövs det omfattande temperaturtester för att försäkra korrekt funktionalitet.

1.2 Syfte

Syftet med denna förstudie är att implementera trådlös överföring på en existerande produkt för att kommunicera med en server och presentera denna i ett grafiskt gränssnitt för en användare i realtid.

1.3 Avgränsning

Då projektet är väldigt stort behöver vissa avgränsningar göras. Därför kommer det gränssnitt som skapas bara vara tillfälligt för visuell illustration för att arbetet ska bli komplett och underlätta vid tester. Även servern som kommer användas under projektets gång är en tillfällig lösning som endast kommer användas under detta arbetet. Ett krav från företaget är att lösningen bygger på wifi.

1.4 Precisering av mål

- Läsa av temperatur och luftfuktighet från sensorerna i realtid.
- Läsa av LED-indikatorerna.
- Sända data trådlöst via wifi.
- Presentera den avlästa datan i ett simpelt gränssnitt.
- Temperaturtesta dem nya komponenterna.

2 Teknisk bakgrund

I det här kapitlet introduceras de förkunskaper som krävs för att utföra projektet samt vilka förutsättningar projektet startar med.

2.1 Arduino

Genomgående för hela projektet är plattformen Arduino Nano, samt Arduino IDE med tillhörande bibliotek. Arduino Nano är en plattform som innehåller mikrokontrollern ATMEGA328P. ATmega328P är en 8-bitars microcontroller med AVR struktur[2]. Den består även av andra integrerade kretsar som gör det enkelt att koppla ihop mikrokontrollern via USB till en dator för programmering. ATMEGA328P har åtta stycken 10 bitars analog till digital omvandlare "A/D-omvandlare" som gör det möjligt att omvandla analoga signaler till digitala signaler. Dessa ingångar kan även användas som digitala in/ut gångar.

En annan funktion som finns i denna CPU är Universal Asynchronous Receiver/Transmitter "UART", som är en teknik för att överföra en byte seriellt till en annan enhet. UART är ett protokoll som har tillhörande hårdvara som finns i ATMEGA328P på två bestämda pinnar. Det är möjligt att simulera en sådan UART med mjukvara, men då fås en lägre baud rate, vilket innebär att färre bitar per sekund sänds.

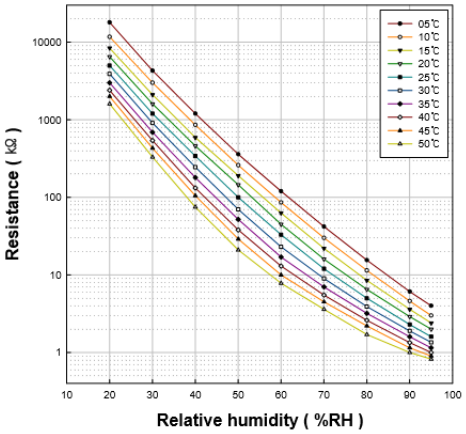
Programmeringen sker främst i Arduino IDE. Arduino har ett eget programmeringsspråk som grundar sig i C och C++, detta kallas för Arduino Programming Language (APL)[3]. Det är möjligt att programmera Arduinon i andra språk, men då den redan befintliga koden som finns för TV-kortet är skriven i APL är det även detta som kommer användas under projektet.

Arduino är en open source plattform vilket gör att det finns exempelkod i form av källkod samt färdiga bibliotek som kan användas.
ATmeaga328P har en inbyggd "Watchdog" vars funktionen är starta en reset på processorn om något går fel inom ett bestämt tidsintervall. Tidsintervallet kan programmeras mellan 16 ms till 8 s.

2.2 Orginalprodukten

Då företaget vill hålla orginalprodukten under sekretess kommer den endast beskrivas generellt, men tillräckligt väl för att kunna ge en förståelse för vilka förutsättningar projektet har.

Produkten syftar till att kunna reducera luftfuktighet i en krypgrund. Enheten är i grunden en Arduino med en kompletterande krets med en spänningsmatning på 230 V. Enheten läser av temperatur och relativ luftfuktighet. Detta görs med ett Negative Temperature Coefficient-motstånd (NTC-motstånd) vars resistans varierar beroende på temperatur och en sensor vars samband mellan resistans, relativ luftfuktighet och temperatur visas i figur 1[4].



Figur 1: Korrelationen för resistans och relativ luftfuktighet för sensorn.

Indatan från sensorerna läses in till Arduinon som baserat på dessa beslutar om värmeslingan ska vara på eller av. Om det blir för fuktigt sätts en utgång på Arduinon till hög som i sin tur drar ett relä så att värmeslingan blir spänningsatt och börjar värma. Om värmeslingan inte är varm finns det en testknapp som tvingar igång uppvärmingen utav värmeslingan, detta används för att kunna kontrollera att enhetens värmefunktion fungerar. På enheten finns det tre stycken LED-indikatorer, en röd samt två gröna. Den röda indikerar om någonting är fel med enheten, den första gröna indikerar om enheten är spänningsatt och den tredje LED-indikatorn lyser när värmeslingan är på.

Då företaget presenterar utdatan för luftfuktigheten i andra mått än relativ eller absolut luftfuktighet som vore ett naturligt val kommer datan för luftfuktigheten som skickas från luftavfuktaren benämnas som RH hädanefter. RH-värdet är relationen mellan temperaturen och den relativa luftfuktigheten.
Då huvudfokus för detta projekt är implementering av wifi lades inget arbete på att omvandla RH-värdet, då det är detta RH-värde som tidigare utlästs via USB-port.

2.3 ESP01

ESP01 är en wifi-modul som är konstruerad av företaget Ai-thinker, denna modul har en CPU ESP8266[5]. ESP01 behöver en matningspänning på 3.3 V och har en medelströmförbrukning på 80 mA, men som kan peaka upp till 170 mA. Wifi-modulen kan användas i ett temperaturspann mellan -40 till 125 grader[5]. Ai-thinker levererar wifi-modulen med en mjukvara som gör det möjligt att kommunicera med den via AT-kommandon. AT-kommandon är ett sätt att kommunicera med en annan enhet via ett terminalfönster, där det sedan innan finns implementerade kommandon. På ESP01 sitter en 2x4-polig header som inte passar i ett kopplingsdäck, därför används en adapter. På adaptern sitter även en kondensator för att filtrera ut höga oönskade frekvenser.

2.4 Temperatur

Marknaden för produkten är i Sverige, vilket betyder att TV-kortet är anpassat till dem olika klimat som finns i landet. Den kallaste temperaturen uppmätt i Sverige är -56°C och den varmaste är 38 °C[6]. Även om detta är extremvärden behöver de finnas i åtanke under projektutvecklingens gång. Dock skiljer sig ofta temperaturen i krypgrunden från utetemperaturen. Vanligtvis är krypgrunden varmare än utetemperaturen på vintern och kallare än utetemperaturen på sommaren. Detta gör att temperaturspannet inte är riktigt lika stort som dem extremvärden på -56 till 38 grader. Hur stor temperaturdifferensen är mellan krypgrund och utetemperatur beror på hur krypgrunden är konstruerad [7].

3 Metod

Metoden kommer beskriva projektets upplägg innan start, vilka förberedelser som gjordes, arbetets gång samt hur summeringen genomfördes.

3.1 Förberedelser

Mycket tid lades på att få en översikt för projektet, genom diskussion med företaget vad de ville ha ut av förstudien. När samtal hölls med företaget gavs fria händer kring projekts upplägg samt den teknik som skulle användas. Projektgruppen kom överens om en gemensam ambitionsnivå, vision om vad som skulle hinnas med och även hur projektet skulle genomföras. Det beslutades gemensamt att göra ett delbart dokument för att kunna föra en dagbok över projektets gång. Dels för att kunna dela med alla som har intresse i projektet och dels för att kunna ha som anteckningar när rapporten skulle skrivas. Det sattes av tid efter varje arbetspass för att skriva i dagboken då den skulle hålla god kvalitet och kontinuitet.

3.2 Upplägg för arbetets gång

Grundtanken under projektet var att för varje del i projektet utföra en kunskapsinsamling för vad som kommer behövas för hård/mjukvara. Efter varje delmål var genomfört skulle vad som hade utförts och vilka problem som var lösta dokumenteras.

3.3 Summering

I slutet av projektet sammanställdes den dagboken som skrivits under projektet vilket blev underlaget för rapportskrivningen.

4 Genomförande

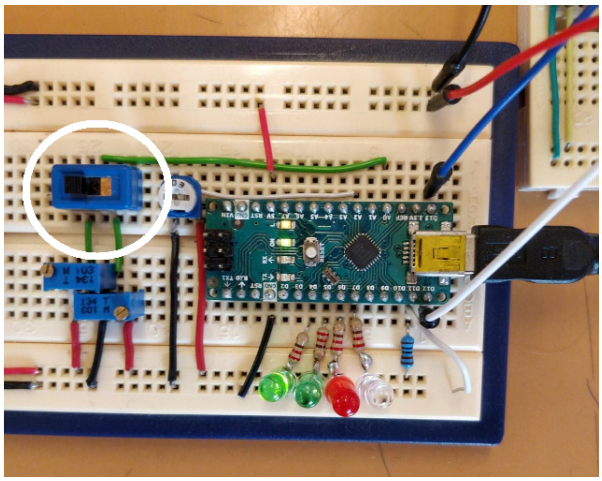
Detta kapitlet beskriver hur projektet genomfördes, resonemang, vad för kretsar som har byggts samt vilka algoritmer som implementerats.

4.1 Prototypkrets av TV-kort

För att enkelt kunna testa kod samt hårdvara under projektets gång byggdes en ny krets upp kring samma typ av Arduino som används i TV-kortet. Kretsen är förenklad så att endast det som är nödvändigt ifrån ursprungskretsen har kopplats upp.

De två gröna samt den röda LED-indikatorn är de som sitter på frontpanelen på TV-kortet. De är även med i prototypkretsen för att kunna verifiera att den fungerar likadant som originalet samt för en smidigare felsökning. Den vita LED-indikatorn motsvarar den signal som visar att värmeslingan är igång, då det är lättare att se en lampa lysa än att vänta på en värmeslinga ska bli varm eller kylas av.

För att ha ytterligare kontroll över indatan och om önskat kunna ändra spänningarna på ingången för att simulera olika temperaturer och luftfuktigheter används potentiometrar. Potetiometerna ställdes in på två olika värden som ska bestämma spänningen på ingången och därmed simulera en specifik luftfuktighet. Potentiometrarna kalibreras till två kända värden, 72 och 183 RH, som representerar att slingan är av respektive på. Switchen i figur 2 gör det möjligt att koppla mellan dem två potentiometrarna. Detta bidrar till en enklare felsökning då spänningen in på A/D-omvandlaren är känd för dem båda lägena, detta gör att sålänge inte de kända värdena ändras är den delen av kretsen rätt. Detta kom att bli användbart då en potentiometer gick sönder under projektets gång och tack vare denna metod gick felsökningen snabbt.

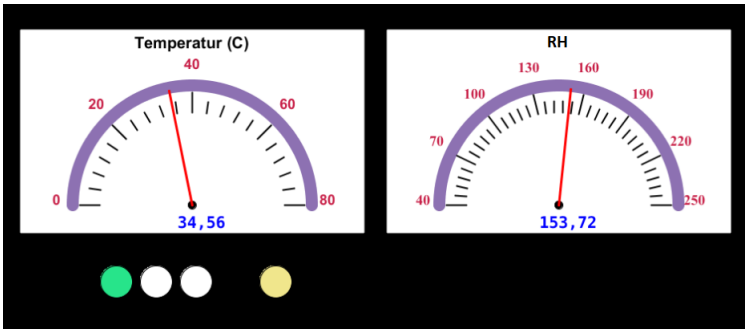


Figur 2: Switch i uppbyggd krets.

4.1.1 Mjukvara

Koden som finns till den ursprungliga produkten var ej kommenterad, därför lades en stor del av tiden på att förstå dess olika funktioner.
Det första testet som gjordes för att sända data från TV-kortet till en annan enhet gjordes med hjälp av programvaran Processing, vilket är ett verktyg för att enkelt kunna bygga upp ett grafiskt gränssnitt som kan ta emot data. Syftet var att undersöka hur de olika parametrarna från koden i TV-kortet behövde konfigureras för att på ett smidigt sätt kunna skicka data i realtid.

Första testet gjordes med den uppbyggda prototypkretsen för TV-kortet, innan Wifi-modulen var tillsatt. Datan hämtades från USB-porten i test-kretsen till datorn där det direkt visades upp i det grafiska gränssnittet. Den data som skickades är den simulerade temperaturen, RH och de fyra LED-indikatorerna. De sänds över i tre olika variabler, där LED-indikatorernas tillstånd representeras i en bitmask. Masken har fyra bitar, där 1 betyder att LED-indikatorn är hög, dvs att den lyser. I detta läget är det relativt små förändringar som är gjorda i orginalkoden för TV-kortet, det som har adderats i koden är framförallt funktionsanrop vid dem ställen där LED-indikatorerna påverkas. För att kunna ha kontroll över vilken ordning datan skickas i ligger även detta i en egen funktion, detta gör att det på mottagarsidan blir smidigare att hantera den datan och skilja på dem tre variablerna. Figur 3 nedanför visar det programmerade gränssnittet gjort i Processing som tar emot den data som skickats från TV-kortet.



Figur 3: Grafiskt gränssnitt i Processing.

Detta grafiska gränssnitt är till för att undersöka huruvida data kan skickas mellan olika enheter. Den skickade datan som visas är en simulerad temperatur på 34° C och ett RH-värde på 153. LED-indikatorerna visar att enheten är spänningssatt (grön cirkel) och att testknappen för att starta värmeslingan manuellt är nertryckt (gul cirkel). Ledmasken i koden blir då 1001, vilket blir talet 9 decimalt. Tabell 1 nedan visar hur bitmasken för LED-indikatorerna representeras i koden.

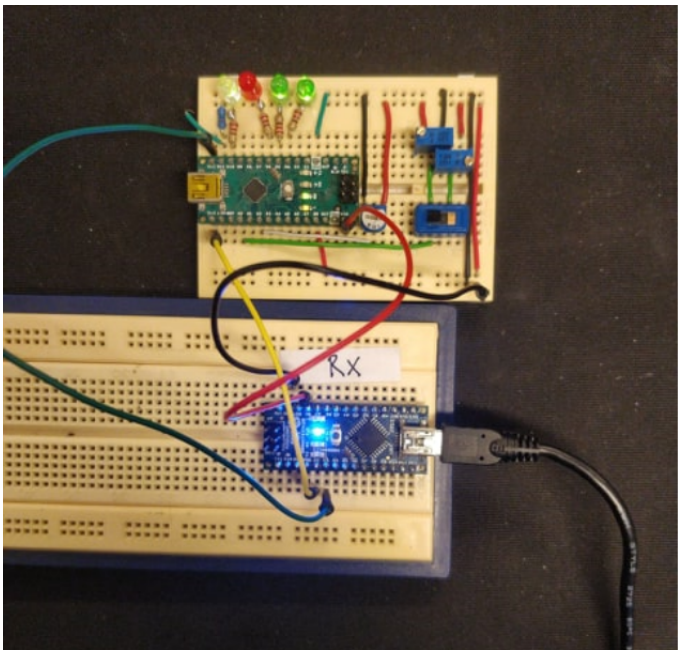
LED	Bit	Decimal
Spänningssatt	3	8
Värmeslinga	2	4
Felsignal	1	2
Testknapp	0	1

Tabell 1: LED-indikatorernas bitmask

4.1.2 Mjukvaru-UART

Universal Asynchronous Receiver/Transmitter ”UART” är ett sätt att skicka över data seriellt. Det är ett av de möjliga sätten för att kommunicera med wifi-modulen. ATmega328P har en UART-port som tyvärr är upptagen på den ursprungliga produkten. Detta kan lösas genom att skriva mjukvara som skickar ut bitarna på samma sätt som hårdvaran, dock med lägre baud rate. För att göra en mjukvaru-UART möjlig behövs två General Purpose Input Outputs (GPIO:s) för att koppla ihop dem två enheterna som ska kommunicera. TV-kortet har flera lediga GPIOs som kan användas till detta ändamål. Det finns redan färdigskriva bibliotek i Arduino IDE som simulerar en UART. Under projektet gång testades några olika sådana bibliotek samt olika GPIO. Det bibliotek som användes till slut heter ”SoftwareSerial”.

Då wifi-modulen inte har en USB-port blir det det svårare att felsöka, på grund av att det ej går att läsa ut variabelvärdena i en serielmonitor på datorn. För att kunna kontrollera att rätt data skickades över till wifi-modulen användes en Arduino i början som fick aggera som en receiver, se figur 4. Detta gjorde det möjligt att verifiera mjukvaru-UART:en fungerade som den skulle.



Figur 4: Arduino som receiver för test av mjukvaru-UART.

Vid dem första testerna av GPIO samt bibliotek fungerade inte produktens ursprungliga funktioner. För att komma vidare testades andra bibliotek för att undersöka om samma fel uppstod. Dock innehöll även de nya testade biblioteken störningar för originalkoden. Nästa metod för felsökningen var att utesluta att det inte var GPIO-pinnarna som gjorde att TV-kortets funktion ändrades, därför byttes även pinnarnas ingång. När pinnarna var bytta fungerade funktionen på TV-kortet som vanligt. Däremot fungerade inte den seriella överföringen, det skickades över en sträng men inte rätt sträng. Efter att ha konstaterat att pinnarna var en del av felet, undersöktes det om biblioteket Softwareserial gav ett bättre resultat för att ta emot strängen, vilket det gjorde. Varför det inte gick med

dem pinnarna som användes först beror på att det var analoga portar som inte hade någon digital input buffer. Då det står på Arduinos hemsida att det går att använda alla analoga inportar som GPIO togs det för givet att det även skulle gå att använda dessa till mjukvaru-UARTen[8]. Efter kontroll i databladet för ATmega328P framkom det att pinnarna A6 samt A7 inte är försedda med en digital buffer, vilket var felet för varför UART-porten inte fungerade på dessa pinnar[2]. Pinnarna A4 samt A5 som har digitala buffers används inte på den ursprungliga kretsen, när dessa portar användes fungerade det att få fram en korrekt sträng till Arduinon som agerade som receiver. Varför dessa pinnar inte valdes först är för att de har stöd för Inter-Integrated Circuit (I^2C) vilket gör det möjligt att koppla till ytterligare perferikretsar om så önskas. ESP8266 har stöd för I^2C men tyvärr är det ej åtkomliga från modulen ESP01.

4.2 Prototypkrets - Wifi-modulen ESP01

Projektgruppen har valt att använda Wifi-modulen ESP01, med den tillhörande processorn ESP8266. Den valdes på grund av att TV-kortet inte behöver konstrueras om för att kunna leverera tillräckligt med ström från transformatorn, som eventuellt krävts vid användning av en annan modul, exempelvis wifi-modulen ESP12.

Wifi-modulen ESP826 har en genomsnittlig strömförbrukning på 80 mA, men den har peakar upp till 170 mA[5]. Adderas strömförbrukningen från ATMEGA328P på 200 mA blir den maximal strömförbrukningen 370 mA. Transformatorn har en märkström på 383 mA, vilket betyder att den har tillräcklig kapacitet[9]. För att spänningsätta TV-kortet är primärsidan av transformatorn matad med 230V. På primärsidan finns en säkring på 3.15 A som varit en riktlinje för projektet, då det har varit önskat från företaget att inga befintliga komponenter ska bytas ut på TV-kortet. Då produkten är märkt med 550 W används effektformeln som visas i ekvation 1 nedan för att beräkna att enheten drar ungefär 2,4 A. Vilket lämnar utrymme för att kunna driva wifi-modulen.

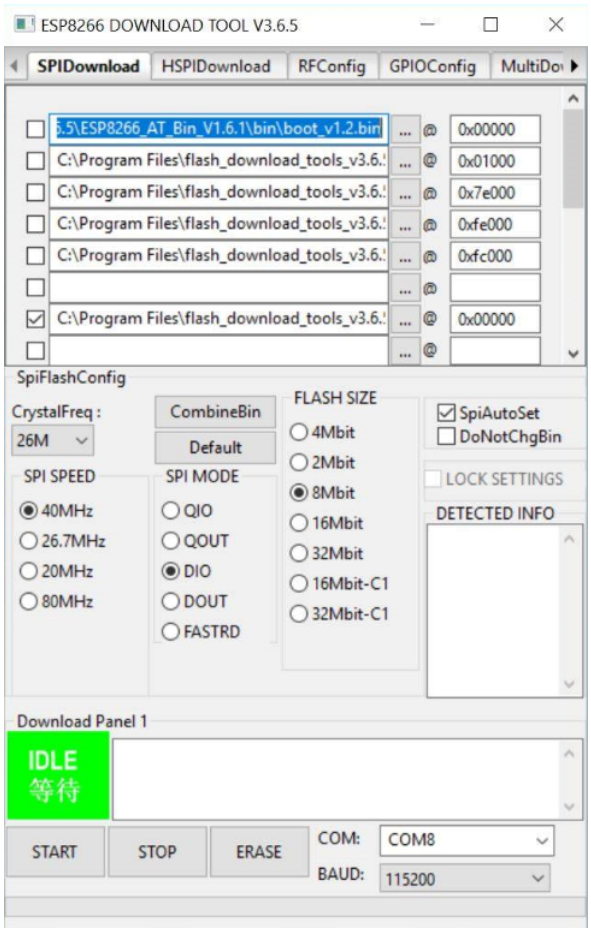
$$I = \frac{P}{U} = \frac{550}{230} \approx 2.4\ A \tag{1}$$

En anledning till att just den här modulen valdes är för att det är möjligt rent fysiskt att implementera dem komponenter från ESP01 till TV-kortet utan att behöva göra mönsterkortet större. Det är även möjligt att kunna plocka bort vissa utav dem komponenter som sitter på modulen, så som lysdioder och motstånd som ej behövs för att veta om modulen är spänningssatt.

Wifi-modulen har ingen USB-port vilket gör att det inte är möjligt att koppla den direkt till datorn och ladda över kod, däremot kan en Arduino användas för att ladda över kod till wifi-modulen istället. Detta sker med hjälp av den USB till serielomvandlare som sitter på Arduinon. Då kan koden skickas över genom seriekommunikation via RX/TX.

ESP8266 kommer med en mjukvara där kommunikation kan ske via AT-kommandon. Under projektet fanns funderingar på om kommunikationen skulle ske genom att programmera den med APL eller ej, slutgiltigt beslutades det att styra wifi-modulen genom att skicka AT-kommandon från Arduinon. För att att styra ESP01 med AT-kommandon

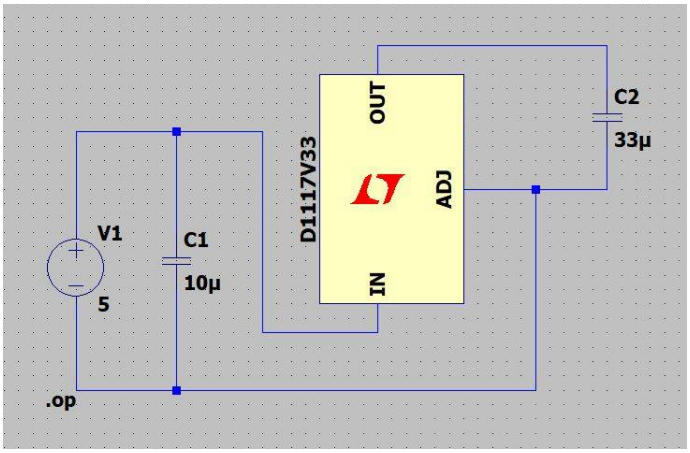
måste en mjukvara laddas över till wifi-modulen, det görs med programmet flash_download_tools_v3.6.5_0 som visas i figur 5.



Figur 5: Verktyget flash_download_tools_v3.6.5_0 för att ladda över mjukvara till ESP8266.

För att spänningsmata wifi-modulen används spänningsregulatorn D1117V33 som visas i figur 6, den används för att ESP01 ska få en konstat spänning på 3.3 V. Eftersom wifi-modulen är känslig för spänningar över 3.3 V används spänningsregulatorn under hela projektet för att säkerhetsställa att ESP01 inte går sönder. Detta gjordes på grund av att 3.3 V utgång på ATMEGA328P ej kan användas då inte tillräckligt med ström levereras. Istället kopplas en 5V utgång från ATMEGA328P som kopplas till plusingången på spänningsregulatorn, vilket löser problemet med strömförsörjningen.

Även UART-signalen från Arduinon ska regleras då den är mellan 0-5 V. En enkel spänningsdelning användas först för att få en spänning på 3.3 V, dock visade det sig att signalen inte gick fram. Motstånden togs bort, trots vetskapen om wifi-modulens känslighet. Då kommunikationen mellan Arduinon och ESP01 fungerade i detta läget beslutades det att använda denna krets även om spänningen översteg 3.3 Volt.



Figur 6: Spänningsreglering till 3.3 volt.

4.2.1 Mjukvara

Nästa steg i processen var att sända data från wifi-modulen till en server. Detta gjordes med prototypkretsen som var byggd kring wifi-modulen. Vid det första försöket för att få kontakt med servern användes terminalfönsteret för att manuellt skriva in dem AT-kommandon som behövdes för att öppna upp kommunikationen mellan wifi-modulen och servern. Efter framgångsrika försök där de två enheterna fått kontakt med varandra programmerades det istället kod i APL som laddades över till Arduinon. Här uppkom problem, för att etablera en uppkoppling krävs det ett stop i koden. Detta medför att den watchdog resetar enheten och försöket till att få åtkomst till routern avbryts. Under projektet inaktiverades denna watchdog, för att senare se hur lång tid det behövdes för att få en internetuppkoppling. Koden är skriven så att om ingen uppkoppling har etablerats inom fem sekunder resetas watchdog:n. I slutet av projektet aktiverades watchdog:n igen, men med en reset på sex sekunder istället. I koden finns det en API-nyckeln, URL, AT-kommandon och användarnamn samt lösenordet för det specifika nätverket wifi-modulen kopplades till. I detta skede är datan som sänds över satt som fasta variabler i koden. Koden som skrivits under projektets gång finns i appendix.

4.3 Server

Ursprungsplanen för att hantera den data som TV-kortet skickar var att använda en egenbyggd server. Denna plan ändrades dock längs projekts gång, vilket berodde på tid och ekonomiaspekter. Detta beslut gjorde att projektet kunde fortskrida i en snabbare takt då vi inte längre var beroende av att vänta på att en ny server skulle bli färdiggjord.

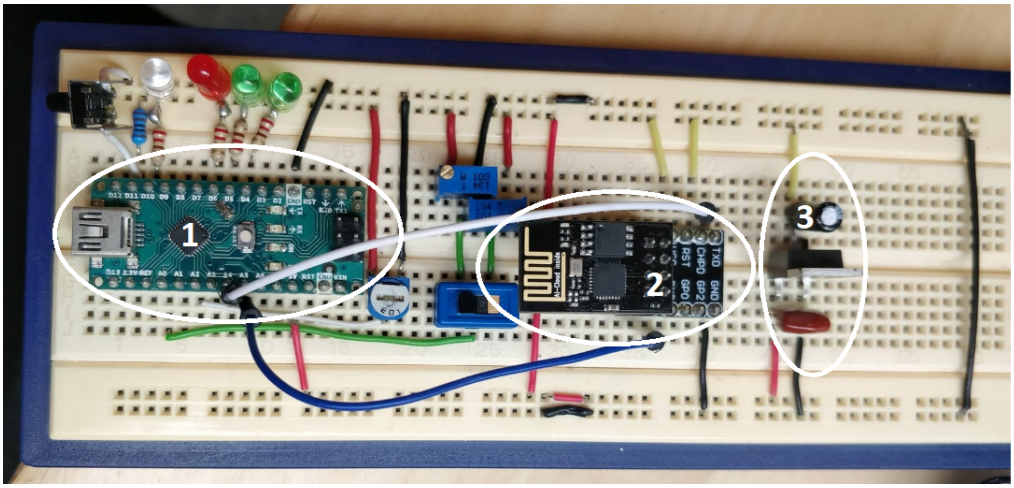
Istället för att lagra data, användes plattformen Thingspeak, vilket också är en server. För detta arbete är det denna server som kommer lagra och presentera data från TV-kortet, men i framtida utveckling kring produkten kommer Trygghetsvakten återgå till den ursprungliga planen för att uppnå optimal kundanpassning.

Thingspeak har kanaler som öppnas för att göra det möjligt att skicka sensordatan från TV-kortet upp till servern. Varje kanal genererar en API-nyckel som skrivs in i koden för Arduinon. API-nyckeln tillsammans med sensordatan och ledmasken skrivs ihop till en

URL som skickas seriellt till wifi-modulen. Då det är önskat att all data laddas upp till servern korrekt så används TCP protokollet, vilket betyder att en mottagare bekräftar att rätt data har skickats över och om ingen bekräftelse mottages kommer datan skickas om i ett nytt försök. Även IP-adressen till Thingspeaks server måste skickas till ESP8266[10].

4.4 Slutgiltig prototyp

Den sista prototypen som byggdes var en kombination av prototypkretsen av TV-kortet och wifi-modulen ESP01, som beskrivs i kapitel 4.1 respektive 4.2. Tidigare i projektet har det undersökts att båda delarna fungerar var för sig, att sensordata skickas korrekt från TV-kortet samt att wifi-modulen kan kommunicera med serven. Dessa ihopbyggda delar visas i figur 7. Ring 1 visar Arduinon från prototypkres från TV-kort se figur 2, ring 2 samt 3 visar prototypkretsen för wifi-modulen ESP01 som beskrivs i kapitel 4.2.



Figur 7: Slutgiltig prototyp

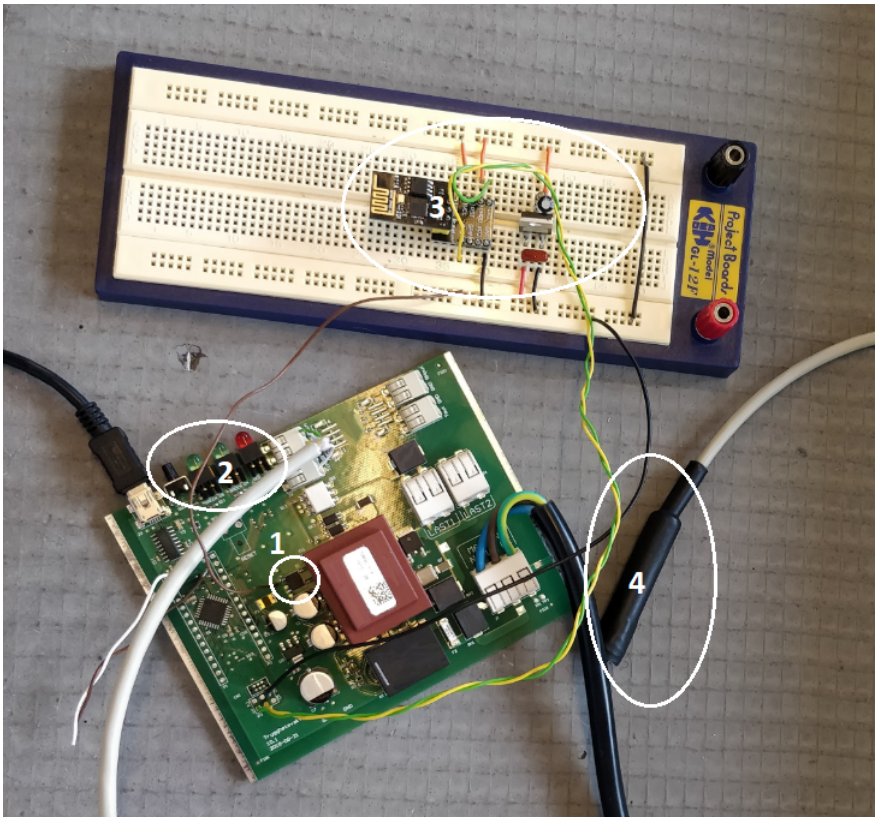
Se appendix del 2 för att hur alla kommandon skickas från arduino till ESP01.

4.5 Implementering på orginalprodukt

Det sista som genomfördes var att testa om den slutgiltiga prototypen för wifi-modulen var möjlig att implementera på orginalkortet, utan att göra några förändringar av dem komponenter som sitter på TV-kortet. Skillnaden mellan implementeringen av ESP01 på den slutgiltliga prototypen och TV-kortet var hur ESP01 spänningssattes. För att spänningsätta wifi-modulen kopplades spänningsregulatorn som visas i figur 6 till den likriktarbrygga som sitter på sekundärsidan av transformatorn på TV-kortet. I och med denna koppling behövdes hela orginalprodukten spänningsättas med 230V.

Under detta testet byttes potentiometrarna som tidigare använts för att simulera sensorvärdena ut till dem verkliga sensorerna. Sensorerna placerades i företagets testkammare som har en hög luftfuktighet, detta gör det möjligt att testa TV-kortets funktioner samt genera mätvärden som inte är simulerade med potentiometrar. Figur 8 visar implementeringen av wifi, där wifi-modulens krets är kopplad till TV-kortet. Tillåtelse gavs att använda denna bild på kretskortet där värdena på komponenterna är censurerade. Ring 1 i figur 8 är den likriktarbrygga som kopplades till spänningsregulatorn, ring 2

är LED-indikatornerna samt testkappen på orginalprodukten. Ring 3 visar ESP01 samt tillhörade spänningsregulator. Slutligen visar ring 4 den sensor som användes i stället för potentiometrarna.



Figur 8: Implementering av wifi-modulen ESP01.

4.6 Temperaturtester

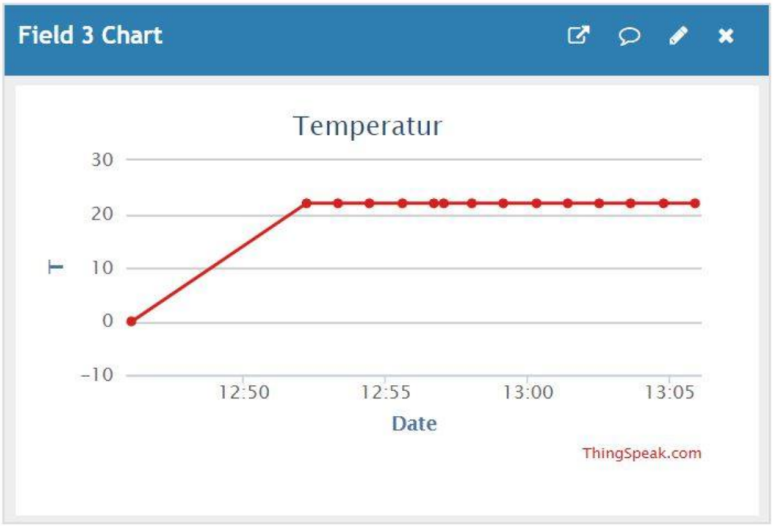
Eftersom enheten kommer vara i miljöer där temperaturen kommer ha en stor variation genomfördes det temperaturtester. Detta gjordes för att kontrollera att enhetens olika funktioner beter sig som tänkt oavsett vilken temperatur enheten utsätts för. Testerna utfördes för att kontrollera att även de nya elektriska komponenterna fungerar inom samma temperaturspann som en krypgrund kan anta. Då detta är en förstudie gjordes ej kompletta temperaturtester, istället utfördes simplare tester för att ge en indikation på hur wifi-modulen ESP01 reagerar efter att ha utsatts för varierande temperaturer.

För att undersöka om modulen klarar av dem höga temperaturer som är tänkta användes en ugn på 50 °C där wifi-modulen placerades. Det visade sig dock snabbt att antennen inte har kapaciteten att sända ut datan genom ugnen, vilket inte var ett oväntat resultat då antennen som sitter på vår valda wifi-modul inte har särskilt lång räckvidd. Utformningen på temperaturtestet modifierades därefter till att enbart ha modulen spänningsatt i ugnen utan att sända data under tiden. Wifi-modulen var placerad i ugnen i 12 timmar på 100 °C, därefter togs den ut och testades direkt med den slutgiltiga prototypkretsen för att undersöka att dess funktioner var detsamma som innan uppvärmingen. Kyltestetet utfördes på liknande vis, wifi-modulen placerades i en frys i 12 timmar på -20 °C där modulens funktionalitet sedan testades.

5 Resultat

Efter projektets sista del var utfört blev resultatet av förstudien att det är möjligt att skicka över data från kretskortet där wifi-modulen ESP01 är implementerad vidare till en server. Detta visar figurerna nedan som är den data som är insamlad från servern Thingspeak. Dock håller inte ESP01 en stabil uppkopplingen, då den har en tendens att tappa anslutningen till internet.

Figur 9 visar temperaturen som TV-kortets sensor har läst av i realtid. Det är endast prickarna som är nya samplingspunkter för temperaturen. Den börjar på 0°C då det första testet gjordes utan sensor och det är initierat att temperaturen är noll om inga nya mätvärden kan utläsas.



Figur 9: Avläst temperatur från sensor skickad till servern Thingspeak.

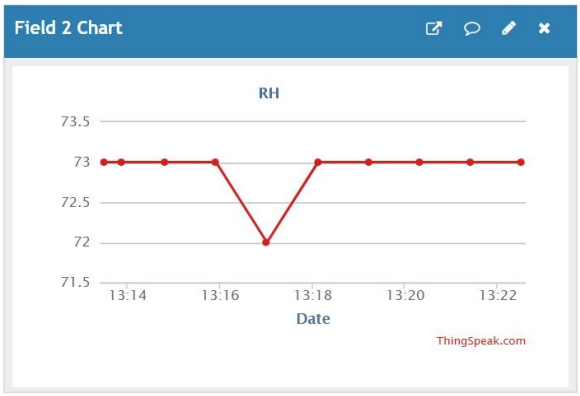
Figur 10 visar RH-värdet baserat på dem avlästa värderna från sensorn på TV-kortet som är skickad till servern Thingspeak. Även denna figur börjar på noll då testet startades utan att sensorn är kopplad till TV-kortet. Då företaget inte vill ge ut dem exakta värdena för när enheten skiftar mellan av och på-läget är y-axelns värden borttagna i figur 10.

Figur 10: Avläst luftfuktighet från sensor skickad till servern Thingspeak.

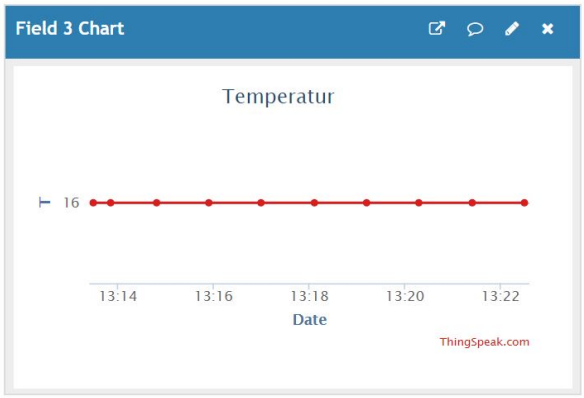
Figur 11 visar det aktuella tillståndet på led-indikatorerna som varierar beroende på indatan. I denna bilden syns en korrelation mellan RH-värdet från figur 10 och när värmeslingan startar och stängs av, vilket visar på att ledmaskan fungerar som planerat. Indatan för testknappen fungerar korrekt, detta syns då ledmasken ökar till nio i en samplingspunkt då knappen trycktes ner manuellt i kretsen. I kapitel 4.1.1 beskrivs det vad de olika värdena har för betydelse.

Figur 11: Aktuellt tillstånd för TV-kortet visat i servern Thingspeak.

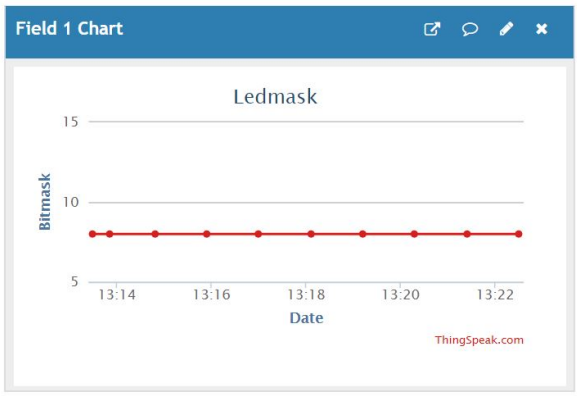
Resultatet från temperaturtesterna visade att wifi-modulen kan hantera dem temperaturerna som modulen utsattes för, 100 °C samt -20 °C i 12 timmar vardera. En kortare insamling av data visar att wifi-modulens funktioner är detsamma som innan, där RH-värde 12a, temperatur 12b och den aktuella statusen 12c visas i figur 12.



(a) RH-värde



(b) Temperatur

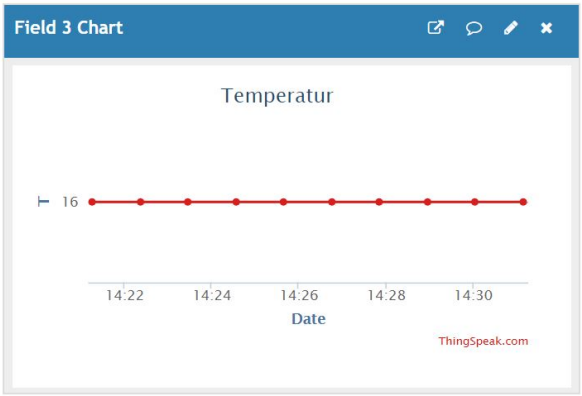


(c) Led-indikatorer

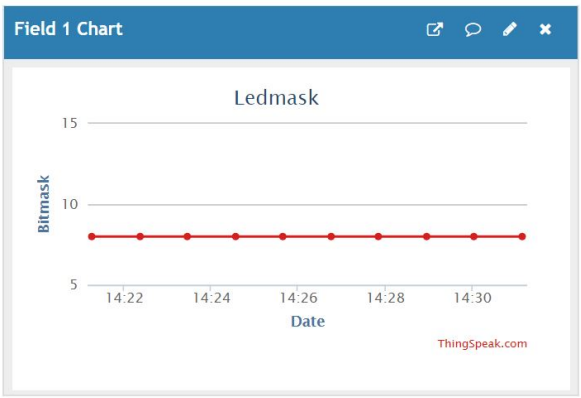
Figur 12: Temperaturtest 100 °C.

Figur 13 visar att wifi-modulen även klarade av kyltestet på -20 °C.

(a) RH-värde



(b) Temperatur



(c) Led-indikatorer

Figur 13: Temperaturtest -20 °C.

6 Diskussion

Då företaget ville ge så mycket tid som möjligt till projektet, kommer en redovisning av det arbete som utförts redovisas efter rapporten är klar. Då versionen av produkten är relativt ny ville företaget att vi skulle hålla ögonen öppna för eventuella buggar eller komponenter som skulle kunna påverka produkten negativt. En sådan redovisning kommer ske till företaget efter rapporten är klar och kommer inte behandlas i denna rapport.

De dippar som syns i figur 12a och 13a ska inte tolkas som att något drastiskt händer, då det är små skillnader i A/D-omvandlingen som gör att det kan bli sådana utslag, uppskattar därför detta ej som ett fel. Under temperaturtesterna var endast ESP01 placerad i värme och kyla och inte hårdvaran som samplat värdena som befunnit sig i rumstemperatur.

Trots att lösningen fungerade skulle vi ändå rekommendera att se om ESP32 är ett bättre alternativ med tanke på stabilitet mellan enheten och routern. Fastän detta innebär en konstruktionsändring av TV-kortet. ESP32 förbereder även för om det vore önskvärt att i framtiden kunna koppla ihop enheten via bluetooth istället för wifi, detta är dock några kronor dyrare.

De pinnar som valdes sist för användning av mjukvaru-UART:n, A4 och A5, skulle kunna göra det möjligt att kommunicera över en I^2C buss. Den möjligheten tas tyvärr bort när vi använder dessa två pinnar för mjukvaru-UART:n. Om det är önskvärt att koppla till fler enheter som har stöd för I^2C måste kretskortet återigen konstrueras om. Det är möjligt att optimera TV-kortes hårdvara för att kunna frigöra GPIO på ATMEGA328P som kan användas till mjukvaru-UART:n.

ESP01 täcker inte hela temperaturspannet om man utgår från SMHI:s köld- och värmerekord, vilket hade hade varit önskvärt. Det som vi gick efter när detta beslut togs var att vissa av dem komponenter som sitter på TV-kortet sträcker sig endast ned till -40 °C. I de datablad samt litteratur som funnits om ESP01 står det att den är väldigt känslig för spänningar över 3.3 V. Vi märkte dock att detta inte stämde när vi kollade på hur amatöranvändare kopplade ESP01 direkt till 5V. Däremot skulle det vara mer pålitligt att använda sig utav Logic Level Converter (LLC), som gör om signalen på 5 V till 3.3 V.

7 Slutsats

Vi är nöjda med resultatet för implementeringen av wifi-modulen ESP01, även om det inte är en helt stabil uppkoppling då enheten ibland tappar kommunikationen mellan routern och wifi-modulen. Vi anser att resultatet av denna förstudie visar på att det finns goda förutsättningar för att ta denna utveckling vidare.

8 Vidareutveckling

Under projektets gång har det framkommit många idéer och tankar på hur produkten skulle kunna vidareutvecklas. I det här kapitlet kommer dessa redovisas.

8.1 Konstruktion av mönsterkort

Det som krävs för att kunna slutgiltligt implementera ESP01 till orginalprodukten är en säkring till ESP01 samt ett par kondensatorer för att vara säker på att högfrekventa störningar elimineras och att spänningen ska hållas konstant. Då resten av komponentera på mönsterkortet är ytmonterade behöver det hittas ersättningskomponenter till dem komponenter som använts under projektet då de är hålmonterande. Man kommer även behöva dra nya ledningsbanor i mönsterkortet för kunna få åtkomst till dem pinnarna på processorn som ska användas till mjukvaru-UART:en.

Wifi-modulen som används under projektets gång har en antenn som sitter fastmonterat i ESP01s mönsterkoret. Kretskortet monteras i en plåtlåda innan den skickas ut till kund. Plåtlådan kommer agera som en Faradaybur vilket gör att en antenn på utsidan av lådan kommer behöva konstrueras.

8.2 Anpassad server

Som det har nämnts tidigare är den nuvarande servern Thingspeak inte den server som kommer användas i framtiden då TV-korten är utplacerade hos kunder. Thingspeak valdes som tillfällig lösning för detta arbetet då servern är användarvänlig, kostnadsfri för studenter samt att den har färdigbyggda grafiska gränssnitt som kan presentera den data som skickas.

Orsakerna till att Thingspeak inte är är optimal för företaget Amrox beror på att kostnaden skulle bli oproportionerligt hög jämfört med den summa som enheterna ökar i värde efter denna utveckling. Det är dock inte bara på grund av ekonomin som det behövs en ny egen anpassad server. Även om det grafiska gränssnittet är smidigt under utvecklingens gång är det inte tillräckligt användarvänligt för en kund. För temperaturen och luftfuktigheten är en graf ett tydligt sätt att presentera dem aktuella nivåerna på, medan LED-indikatorerna behöver ett annat tydligare gränssnitt än en graf. En tredje anledning till till att bygga upp en ny server är på grund av att varje enhet behöver ha en egen identitet som servern ska kunna urskilja. En kund behöver därför kunna använda sig av en egen nyckel för att verifiera vilket TV-kort som datan ska läsas från samt utge vilket wifi-nätverk TV-kortet ska vara kopplat till. För att uppnå detta behöver både servern samt koden i Arduinon utvecklas ytterligare.

Referenser

[1] Sparkfun Electronics. (2015). WiFi Module - ESP8266, URL: https://www.flickr.com/photos/sparkfun/19681470919/in/photostream/ (hämtad 2019-05-24).

[2] *8-bit AVR Microcontroller with 32K Bytes In-System Programmable Flash*, 2015, Atmel, 2019.

[3] Arduino. (2019). Frequently Asked Questions, URL: https://www.arduino.cc/en/main/FAQ (hämtad 2019-04-18).

[4] *HUMIDITY SENSOR*, SAMYOUNG,

[5] *ESP-01 WiFi Module*, Ver. 1.0, Ai-thinker, 2015.

[6] SMHI. (2019). Svenska temperaturrekord. April 2019, URL: https://www.smhi.se/kunskapsbanken/meteorologi/svenska-temperaturrekord-1.5792 (hämtad 2019-05-10).

[7] Polygon. (). KRYPGRUND. April 2019, URL: https://www.polygongroup.com/sv-SE/kunskapstorget/riskkonstruktioner/krypgrund/ (hämtad 2019-05-10).

[8] Arduino. (2019). Analog Input Pins. 2019, URL: https://www.arduino.cc/en/Tutorial/AnalogInputPins.

[9] *DATA SHEET 44175 EI 30x18*, Myrra, maj 2005.

[10] Mathworks. (2019). Collect Data in a New Channel, URL: https://www.mathworks.com/help/thingspeak/collect-data-in-a-new-channel.html (hämtad 2019-04-05).

Appendix

Kod

Här följer den kod som skrivits under projekts gång. Observera att koden för orginalprodukten ej finns med.

```
#include <SoftwareSerial.h>//Bibliotek för mjukvaru-UART
String myAPIkey = "XXXXXXXXXXXXX";//API-nyckel till thingspeak-servern
SoftwareSerial ESP8266(A4,A5);//Deklarerar TX och RX för Mjukvaru-UART
//Globala variabler
uint8_t LedMask = 0;
uint8_t RHStatus = 0;
int8_t TempStatus = 0;
unsigned char times_check=0;
```

```
//Finns i "SetUp" exekveras en gång
ESP8266.begin(9600);//Sätter baud rate för mjukvaru-Uart

  Serial.println("Connecting to Wifi");//Skriver ut "Connecting to Wifi" i seriell monitor
   while(check_connection==0){//När vi inte har en wifi-uppkoppling körs denna while loop
    Serial.print(".");
    ESP8266.print("AT+CWJAP=\"SSID\",\"Password\"\r\n");//Skickar AT-kommandot för att koppla upp sig mot önskat wifi
    resetWatchdog();
    ESP8266.setTimeout(5000);//Funktion som sätter att man max väntar 5 sek för att ta emot strängen "WIFI CONNECTED" från ESP01
    resetWatchdog();
    if(ESP8266.find("WIFI CONNECTED\r\n")==1){//Om ESP01 skickat tillbaka "WIFI CONNECTED" utförs if-satsen
     Serial.println("WIFI CONNECTED");//skriver ut att vi är uppkopplade
     break;//går ur while loopen
    }
   times_check++;//ökar times_check med 1
   if(times_check>3){//Tre försök att koppla upp sig till wifi görs innan "Trying to Reconnect.." skrivs ut
     times_check=0;
     Serial.println("Trying to Reconnect..");
   }
  }
```

```
void sendToESP8266() {
  /*
   *  När nya värden har samplats, testknappen trycks in eller om värmeslingans status ändras så exekveras denna funktion.
   *  Serial.print funktionerna gör det möjligt att kunna skriva ut värderna i Arduino IDE:s seriella monitor.
   *  Detta görs för att kunna kontrollera att det som skickas till Thingspeak innehåller samma värden.
   */
    Serial.print(LedMask);
    Serial.print(",");
    Serial.print(RHStatus);
    Serial.print(",");
    Serial.println(TempStatus);
    Serial.flush();

    writeThingSpeak();//skickas vidare till denna funktion
}

void writeThingSpeak(void){
  startThingSpeakCmd();// Skickas vidare till denna funktion

//För att veta vilka mätvärden som ska presenteras i vilken graf i Thingspeak så fylls olika fällt i den sträng som skapas nedan

  String getStr = "GET /update?api_key=";//Börjar på den sträng som innehåller API-nyckel och mätvärderna.
  getStr += myAPIkey;//API adressen till vår kanal på Thingspeaks server
  getStr +="&field1=";
  getStr += String(LedMask);
  getStr +="&field2=";
  getStr += String(RHStatus);
  getStr +="&field3=";
  getStr += String(TempStatus);
  getStr += "\r\n\r\n";
  GetThingspeakcmd(getStr); //Skickas vidare till funktionen med strängen vi skapat som inparameter
}
```

```
void startThingSpeakCmd(void){
  ESP8266.flush();//Flushar mjukvaru-UART:en om något skulle ligga kvar från förra gången det skickats data
  String cmd = "AT+CIPSTART=\"TCP\",\"";//Börjar på en sträng som kommer bli den sekvens som öppnar den kanal där URL:n kommer skickas upp till servern
  /*
   * AT+CIPSTART är ett AT-kommando för att öppna en kommunikationskanal till vår server
   */
  cmd += "184.106.153.149"; // api.thingspeak.com IP address
  cmd += "\",80";//Port 80 används för TCP-protokollet
  ESP8266.println(cmd);//Genom att skicka över den här strängen till ESP01 öppnas en väg för att skicka över vår mätdata
  Serial.print("Start Commands: ");//Skrivs ut i seriella monitorn för att kontrollera att det stämmer
  Serial.println(cmd);
  if(ESP8266.find("Error"))// Om wifi-modulen returnerar "Error" efter kommandot AT+CIPSTART
  {
    Serial.println("AT+CIPSTART error");//Skrivs ut i monitorn om något har gått fel
    return;
  }
}
```

```
String GetThingspeakcmd(String getStr){
  String cmd = "AT+CIPSEND=";//AT-CIPSEND är det AT-kommando som gör att ESP01 börjar sända data
  cmd += String(getStr.length());//Vi behöver veta längden av den stränga som vi har som inparameter
  ESP8266.println(cmd);//Vi skickat upp AT+CIPSEND = längden på vår sträng
  Serial.println(cmd);//Skriver även ut den på skärmen
  if(ESP8266.find(">"))//ESP01 returnerar ">" om AT+CIPSEND fungerar som den ska
  {
    ESP8266.print(getStr);//Skickar vår sträng via mjukvaru-UART:n till ESP01
    Serial.println(getStr);
    delay(500);

    String messageBody = "";
    while (ESP8266.available())//Sålänge vi tar emot data från ESP01 kommer det som står inne i while-loppen hända
    {
      String line = ESP8266.readStringUntil('\n');//Tar emot en sträng från ESP01 till ett newline
      if (line.length() == 1) //Om strängen som mottagits är en char lång
      {
        messageBody = ESP8266.readStringUntil('\n');//lägg ett newline till MessageBody
      }
    }
    Serial.print("MessageBody received: ");//skriver ut det som vi mottagit ifrån ESP01
    Serial.println(messageBody);//Här kan vi kontrollera att det vi skickat över till wifi-modulen är korrekt mottaget genom att studera returvärderna som skickas från ESP01
    return messageBody;
  }
  else//Om någonting går fel
  {
    ESP8266.println("AT+CIPCLOSE");// Skicka AT+CIPCLOSE till ESP01 för att avsluta protokollet
    Serial.println("AT+CIPCLOSE"); //skrivs ut i monitorn
  }
}
```

Strängar som skickas via UART

I denna del av appendix visas terminalfönstret för den data som skickas från kretsen till servern. Den mörkare delen av texten visar en cykel, första raden som skrivs ut är den ledmask som innehåller informationen om datan som skickas till Thingspeak.

```
10:37:45.687 -> Connecting to Wifi
10:37:45.727 -> ..WIFI CONNECTED
10:38:56.270 -> 13,184,16
10:38:56.310 -> Start Commands: AT+CIPSTART="TCP","184.106.153.149",80
10:39:01.551 -> AT+CIPSEND=71
10:39:01.671 -> GET /update?api_key=8BZ1UOKJAM8FZXB5&field1=13&field2=184&field3=16
10:39:01.671 ->
10:39:01.671 ->
10:39:02.707 -> MessageBody received: +IPD,2:17CLOSED
10:40:03.033 -> 13,184,16
10:40:03.073 -> Start Commands: AT+CIPSTART="TCP","184.106.153.149",80
10:40:08.470 -> AT+CIPSEND=71
10:40:08.590 -> GET /update?api_key=8BZ1UOKJAM8FZXB5&field1=13&field2=184&field3=16
10:40:08.590 ->
10:40:08.590 ->
10:40:09.630 -> MessageBody received: +IPD,2:18CLOSED
10:41:09.809 -> 8,73,16
10:41:09.849 -> Start Commands: AT+CIPSTART="TCP","184.106.153.149",80
10:41:15.039 -> AT+CIPSEND=69
10:41:15.159 -> GET /update?api_key=8BZ1UOKJAM8FZXB5&field1=8&field2=73&field3=16
10:41:15.159 ->
10:41:15.159 ->
10:41:16.232 -> MessageBody received: +IPD,2:19CLOSED
10:42:16.416 -> 8,73,16
10:42:16.456 -> Start Commands: AT+CIPSTART="TCP","184.106.153.149",80
10:42:21.718 -> AT+CIPSEND=69
10:42:21.798 -> GET /update?api_key=8BZ1UOKJAM8FZXB5&field1=8&field2=73&field3=16
10:42:21.838 ->
```