En växtautomat

Sammanfattning

Den här rapporten redogör ett projektarbete som utfördes under läsperiod 4 2023, i kursen EITA 15. Projektet genomfördes av en grupp om 5 personer som självmant fick skapa en produkt samt bestämma dess funktionalitet.

Grupp 4 valde att skapa en apparat som automatiserar ett klimat. Genom att automatisera den anpassningsbara temperaturen, luftfuktigheten, vattenanvändning och även ljus till olika växters behov.

Gruppen fördjupade sitt kunnande kring komponenterna med hjälp av datablad, vilket följdes av att ett kretsschema kunde ritas upp som användes som översikt vid lödning/virning av komponenterna.

Projektet resulterade i en automatiserings-apparat som kan mäta temperatur, luftfuktighet och jordfuktighet. Med dessa värden samt ett optimalt värde reagerar på förändringar av mätvärdena. Apparaten kan kopplas så att den justerar temperatur, luftfuktighet och jordfuktighet genom att slå av och på utrustning som exempelvis lampor, element och vattenkran.

Nyckelord

1. Växt-automation
2. C
3. Automatisering
4. Temperatur
5. Fuktighet

Abstract

This report describes a project work that was carried out during study period 4 in 2023, in the course EITA 15. The project was conducted by a group of 5 individuals who voluntarily created a product and determined its functionality.

Group 4 chose to create a device that automates the climate. By automating the adjustable temperature, humidity, water usage, and even light according to one of two climate zone settings.

The group deepened their knowledge of the components by using datasheets, which was followed by drawing up a circuit diagram that was used as an overview during the soldering/wiring of the components.

The project resulted in an automation device that can measure temperature, humidity, and soil moisture. Based on these values and an optimal value, it can adjust the temperature, humidity, and soil moisture by turning on and off equipment such as lamps, heaters, and a water source.

Innehållsförteckning

1. Inledning

Under Läsperiod 4 i kursen Digitala System (EITA 15) innefattade ett projektarbete, som i sig innefattade både mjukvara och hårdvara. Där syftet var att använda sig av kursens innehåll för att skapa en produkt gruppvis med fria händer att välja dess funktioner.

1.1 Bakgrund

Projektet delades ut till eleverna av kursansvariga, Bertil Lindvall, och även hans kollega Lars-Göran Larsson. Detta gjordes för att ge eleverna en chans för att testa sina kunskaper och för att utmana sig själva.

1.2 Syfte

Arbetet ska kunna förbättra och förhoppningsvis öka grönskan och hjälpa med att bidra till den globala tillväxten av gröna zoner.

Detta system ska förenkla odling av olika växter och självmant anpassa sig efter det som odlas. Dessutom ska detta ge alla en chans att kunna bidra utan att kostnaderna för ett sådant system ska bli för höga.

1.3 Problemformulering

Konstruera en fungerande växtautomat med följande utrustning; Mcu Atmega 1284; skärm, GDK 1602; sensorer, DHT22 och jordfuktighetsmätare Moisture Sensor 1.3; knappar och dioder; samt annat material för att koppla ihop komponenterna.

1.4 Motivering av projektarbetet

Motiveringen för påbörjandet av projektet var att förbättra odlings-förmågan under vintern och även möjliggöra det i hemmet. Projektet inleddes på grund av en önskan att lösa problemet med brist på grönsaker under vintermånaderna och att öka självförsörjningen på mat. Genom att kunna odla grönsaker hemma året runt skulle man också minska koldioxidutsläppen från transport av mat och bidra till en mer hållbar livsstil. Dessutom skulle en ökad självförsörjning på mat också kunna ge ekonomiska fördelar genom att minska matkostnaderna och skapa nya försäljningsmöjligheter för lokala producenter av odlingsmaterial.

1.5 Avgränsningar

Det som inte ingår är lampa, element eller bevattningssystem då vi inte fick använda stark ström.

2. Teknisk bakgrund

2.1 komponenter

2.1.1 CPU

ATmega1284 (datablad)

Atmega1284 är en 8-bitars mikrokontroller AVR processor på upp till 20Mhz och ett 128 kb flashminne. Kontrollern har stöd för upp till 32 digitala IO portar och även stöd för realräkning, samt stöd för JTAG för debugging och felsökning.

2.1.2 Display

GDM1602K (datablad)

GDM1602K är en typ av alfanumerisk LCD-skärm med 16 kolumner och 4 rader. Den används för att visa text och enklare grafik i inbyggda system och elektronikprojekt. Skärmen styrs vanligtvis via en parallell gränssnittsbuss och har en bakgrundsbelysning för att vara läsbar i olika ljusförhållanden.

2.1.3 Jordfuktighetssensor

Moisture Sensor 1.3 (SKU:SEN0114)

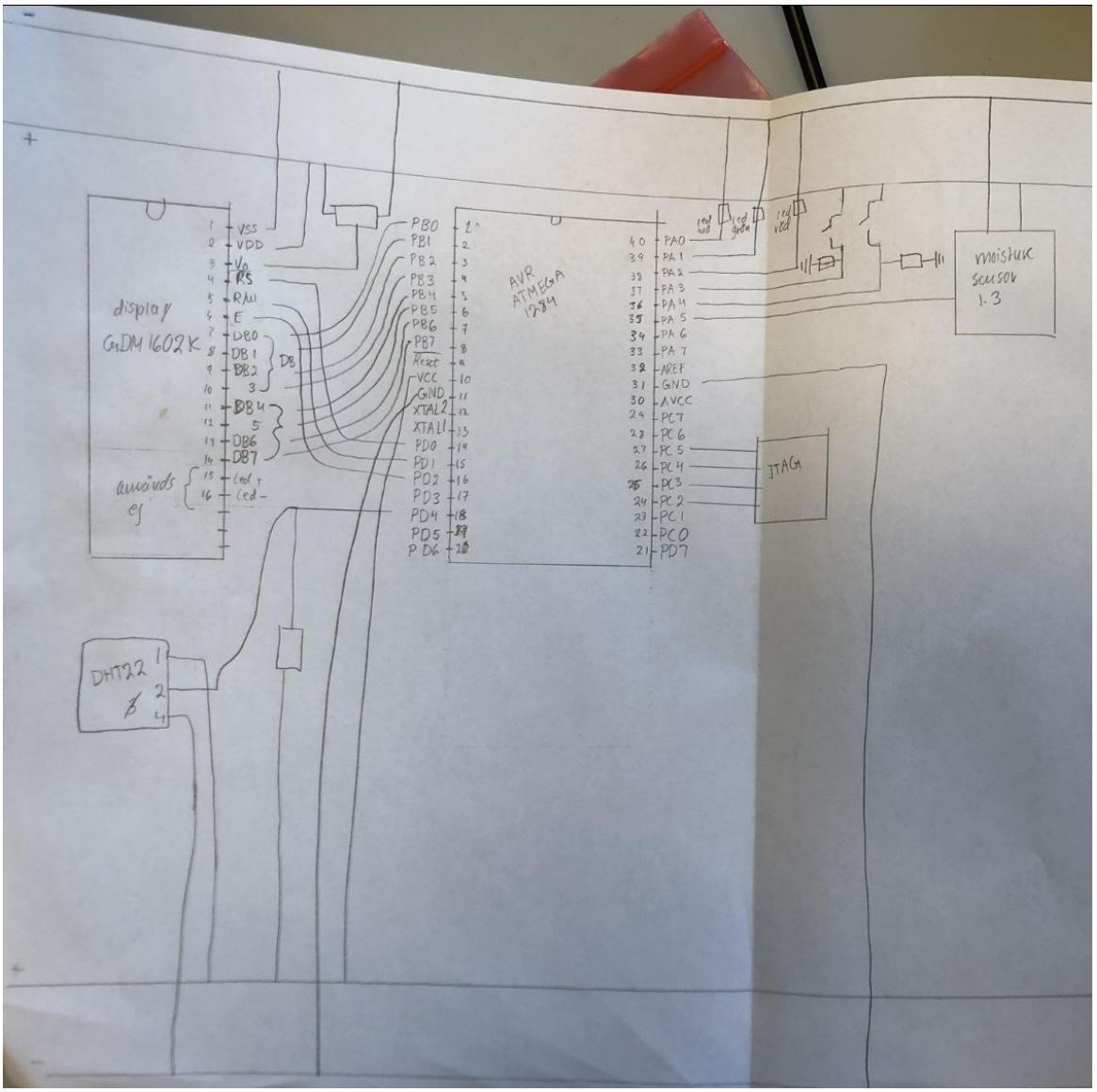
Moisture Sensor 1.3, är en elektronisk sensor som mäter fuktighetsnivåer under jorden med hjälp av två pinnar. Sensorn använder en resistiv fuktighetssensor och ger en analog spänningssignal. Den kräver en mikrokontroller eller annan enhet för att konvertera signalen till en läsbar fuktighetsnivå. Komponenten är enkel att använda och kan ha en justerbar potentiometer för kalibrering.

2.1.4 Luftfuktighet- och temperatursensor

DHT22 Sensor (DHT22.pdf)

DHT-sensorn är en fuktighetssensor som använder sig av seriell kommunikation. Den kan mäta fuktighet och temperatur och kommunicerar med MCU på en av dess digitala i/o anslutningar. Sensorn ger digitala utdata och använder en termistor för temperaturmätning och en fuktkänslig resistiv sensor för fuktighetsmätning. Avläsningarna kan vara upp till 2 sekunder gamla eftersom sensorn endast kan utsända data varannan sekund.

2.2 Kopplingsschema



Figur 1. Kopplingsschema för växtlådan.

2.3 mjukvara

C

Skapades och användes genom Microchip studio

DHT Sensor

DHT22 är koden som används för luftfuktighets- och temperatursensorn.

Webbdesign

Skriven i HTML, CSS och även Javascript för att göra sidan mer iögonfallande.

HTML för att konstruera sidan.

CSS för att ge texten en design och även tillsätta elementen sin plats.

Javascript för att göra designen av sidans innehåll bättre.

---

3. Metod

Projektet genomfördes i flera steg. I första steget planerades kravspecifikation där produktens funktioner bestämdes och komponenterna som skulle användas valdes ut med rekommendation av en kursledare. I andra steget ritades kopplingsschemat innehållande alla komponenterna för att göra det enklare att se hur de ska samverka med kretssystemet. Därefter byggdes hårdvaran efter att kretsschemat blivit godkänt. Steget därpå utgick på att skriva koden med hjälp av komponenternas datablad i C språket. Sista steget förutsatte att skapa en hemsida, gällande projektet skrivet i HTML, Javascript och dessutom CSS.

Projektet genomfördes i flera steg med noggrann planering och korrekt utförande för att säkerställa framgång och funktionalitet. I planeringsstadiet fastställdes projektets krav och önskade funktioner. Komponenterna valdes noggrant ut med hjälp av kursledarens rekommendationer för att säkerställa deras lämplighet för projektet.

Efter kravspecifikationen ritades ett kopplingsschema som visar hur komponenterna skulle kopplas samman och fungera inom kretssystemet. Noggrannheten vid ritningen av kopplingsschemat var avgörande för att undvika felaktiga anslutningar eller fel.

När kopplingsschemat godkändes, fortsatte planeringens gång med att bygga hårdvaran enligt schemat. Detta inkluderade att fysiskt koppla samman komponenterna enligt kretsschemat, med noggrannhet och precision för att undvika eventuella fel eller skador.

Efter att hårdvaran byggts var det dags att skriva den nödvändiga koden för att styra komponenterna. Datablad för komponenterna användes för att förstå deras funktionalitet och skriva korrekt kod. C-språket har använts för kodningen, med fokus på testning och felavhjälpning för att säkerställa en stabil funktion av projektet.

Slutligen skapades en webbsida med HTML, Javascript och CSS för att presentera projektet på ett tydligt och användarvänligt sätt. Hemsidan visade projektets information, med fokus på design och layout för att skapa ett attraktivt och användarvänligt gränssnitt.

Genom att följa dessa steg kunde projektet genomföras framgångsrikt och nå önskat resultat. Noggrann planering, korrekt koppling, funktionsrik kodning och en tilltalande webbsida var nyckelfaktorer för projektets framgång.

3.1 Källkritik

De enda källorna som användes under projektets gång är komponenternas datablad. De tycks vara trovärdiga eftersom de kom direkt från tillverkarna av komponenterna. En stor del av kunskaperna som användes för att genomföra projektet kommer från tidigare erfarenheter från de olika laborationerna som gjordes under kursens gång. Dessutom utnyttjades kunskaperna hos experter inom området.

4. Analys

De tre största problemen i projektet har varit DHT22 sensorn, skärmen (GDK 1602) och sedan följt av jordfuktighetsmätaren. På grund av databladens ibland svårtolkade tekniska språk, nybörjar-förkunskaper och databitarnas förglömliga utseende har problemen uppstått.

DHT22 sensorn
Den kommunicerar med AVR med seriell kommunikation i tre steg. Först sker en handshake mellan sensor och cpu och sedan skickar DHT22an 40 bitar i rad. Dessa tre steg, som vi kommer att kalla för startsignal från cpu, startsignal från dht och informationen, kommuniceras med hjälp av hög och låg signal i olika tidsintervaller. För att sensorn ska vakna måste AVR först skicka hög signal i x antal sekunder och sedan låg signal y antal sekunder. Sedan säger DHT att den är vaken till AVR på liknande vis. Sedan skickas 40 bitar där ‘1’ motsvaras av en tidskombination, och ‘0’ motsvaras av en annan, dessa sparas sedan ner i variabler i programmet (se DHT22.pdf för specifika tider)[1]. Dessa 40 bitar motsvaras av 5 bytes. Där byte 1 och 2 är temperaturen, byte 3 och 4 är fuktigheten och byte 5 är en checksumma. Efter att informationen sparats ner, kunde de skickas till displayen för att visa temperatur och luftfuktighet i rummet.

Svårigheten i denna komponent har legat dels i databladets något röriga utformning, samt komponentens krävande på precision vad gäller tidsintervallen. Databladet har förklarande bilder, men de bilderna har också pilar som pekar ibland på kinesiska ord och ibland på engelska. Det var minst sagt tidskrävande att avkoda databladet. Men med hjälp av tålamod och andra förklarande källor samt kodexempel blev koden till sist fungerande. Se 2.3 för kod-källa.

Skärmen GDK 1602
Skärmen hade i huvudsak två bussar. En databuss, db0-db7. Och en kontrollbuss, rs, rw och E. Med hjälp av att sätta kontrollbussens bitar till ‘1’ eller ‘0’ kunde man välja huruvida man ville justera muspekarens position respektive skicka bitar som skulle motsvara bokstäver på skärmen. För att uppdatera, eller skicka ny instruktion till skärmen så *flippade* man på E, dvs satte den från ‘1’ till ‘0’.

Skärmen gick egentligen enkelt att installera med välbehövlig hjälp från labbhandledare. Däremot uppstod oförklarliga buggar i systemet efter implementeringen av andra komponenter som också använde sig av samma register på AVR. Eftersom om RS respektive RW blev ändrade skickade man inte längre instruktioner om vart muspekaren ska gå, utan det kunde ske oförklarliga saker så som att hela displayens information blev skiftad

[1] https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf

ett steg åt vänster. Dessa buggar utforskades tyvärr inte längre än till dess att vi kunde gå runt dem.

Jordfuktighetsmätaren
Den var relativt enkel att installera. Kommunicerar sin analoga signal som ges av ett elektriskt fält mellan gafflarna, mellan 0 till 5v ger värden mellan 0-950. För att läsa signalen krävdes ADC-funktionen i atmegan, och kunde installeras med hjälp av kod från tidigare gjord labb (2).

Ett fjärde något oförklarligt problem
Våra dioder vägrade att lysa i sin fulla prakt, och vi vet än idag inte varför. Tre dioder skulle användas för att representera olika av och på, eftersom syftet med apparaten var att kunna simulera ett klimat ville vi att apparaten skulle slå av och på en ljuskälla, element, vattentillgång samt luftfuktare. Dessa dioder skulle då representera 3 av dessa apparater. Och vid påsättning av en diod fungerade ljusstyrkan utmärkt, men med tre dioder igång samtidigt uppstod problem med styrkan. Ett första försök till lösning var att ge samtliga dioder varsin resistor eftersom om de alla gick till samma ben kunde det tolkas som en parallellkoppling, vilket skulle ge strömdelning. Problemet kvarstår dock med varsitt resistor ben ner till jorden.

5. Resultat

Resultatet blev en fungerade klimat-simulator, eller en tomat-automat. En sådan cola-automat, fast med grönsaker i. Sensorerna ger data, som skickas till en display för enkel avläsning, dioder lyser för att simulera ljuskälla, vattentillgång, element eller luftfuktare är på respektive av. Detta beroende på mätvärdens relation till de *optimala* värdena. Med hjälp av apparatens inställningar kan användaren alltså välja klimatzon, och därefter finns förprogrammerade värden som apparaten strävar efter att upprätthålla. Exempelvis kan användaren välja klimatzon “medelhavet” för att få bra klimat för exempelvis tomater, apelsiner, citroner, oliver och så vidare.

Vad gäller det tekniska resultatet så fungerar apparaten. Den gör det den utlovar. Däremot finns det ett antal förbättringar som kan göras för en bättre slutprodukt..

Byta delay mot interrupt
Koden använder utslutande delayer för att exempelvis låta skärmen göra färdigt sina instruktioner, och med detta tillvägagångssätt använder vi CPUn på ett suboptimalt sätt. För att sensorn DHT22 ska fungera används även där delayer, om detta hade kunnat bytas ut mot interrupt, är ett outforskat ämne för rapportförfattarna.

För att simulera dag, resp. natt (lampa av och på), har en *sekundräknare* implementerats. Även denna med delayer istället för interrupt, vilket resulterar i en oprecis tidräknare.

Det finns dock ett problem med att implementera interrupter i koden då timingen på DHT22an skulle kunna råka bli avbruten mitt i sin process vilket skulle leda till att systemet

kraschar, eller fastnar då koden för DHT22 inte har någon felhantering. Skulle timingen komma av sig kan MCUn mycket väl fastna i en while-loop.

Starkström
För att ta steget från simulering till en faktiskt apparat skulle vi behöva kommunicera med starkström, enklast väg hade nog varit att koppla om de kanaler vi tänder och släcker ljuskällorna med till en apparat som drar upp spänningen till 230v och på så sätt direkt sätter på den *riktiga* belysningen, stänger av och på vattenkranar och så vidare. Men ett annat sätt att kommunicera med 230V hade varit att installera PWM-styrda dc-motorer som skulle kunna slå av och på knapparna på den utrustning som står externt från våran apparat.

Kodstruktur
Ett vidare steg hade också kunnat göras genom att skriva om koden i återanvändningsbara funktioner. Mycket av koden är skriven i statiska funktioner. Där en funktion kan heta writeTemperature(); som egentligen bara använder en funktion write(); upprepade gånger som tar en det värdet vi vill skicka till display. Otroligt ineffektiv kod. Hade kunnat skriva en mer generell kod writeToDisplay(); som tog in strängar som direkt översattes till displayen utan att behöva leta i en ascii-tabell för varje gång vi skrev ut något. Detta hade vi kunnat göra genom att definiera alla symboler skärmen hade med dess hexadecimala värde i en hjälpfil (hjälp.h fil). Detta är endast ett exempel bland många.

6. Slutsats

En fungerande apparat med hög takhöjd för utveckling. Möjligheterna för att automatisera odling är många, och en apparat lik våran skulle kunna kompletteras med exempelvis ljussensorer för att även kunna bli användbara i samverkan med solen istället för att helt använda sig av lampor. Datorn hade även kunna ha robotarmar som plockar frukterna när de är mogna. Jorden hade kunnat ha automatiserad näringsanalys och bli gödslad automatiskt.