YRKESHÖGSKOLAN
NOVIA

Prototyp av tangentbord med programmeringsgenvägar

Mats Sandqvist

Examensarbete för ingenjörsexamen (YH)-examen

El- och automationsteknik

Vasa 2024

EXAMENSARBETE

Författare: Mats Sandqvist
Utbildning och ort: El- och Automationsteknik, YH, Yrkeshögskolan Novia i Vasa
Inriktning: Automationsteknik
Handledare: Jan Berglund

Titel: Prototyp av tangentbord med programmeringsgenvägar

Datum: 23.5.2024 Sidantal: 35 Bilagor: 1

Abstrakt

Detta examensarbete handlar om utvecklingen av en första prototyp av ett tangentbord med programmeringsgenvägar. Syftet med arbetet var att snabba upp programmering, speciellt när det är många funktioner som skall repeteras, samt att minimera skrivfel under programmering. Målet med arbetet var att ha en fungerande prototyp, vilket innebar att ha designat ett kretskort med en display som visade menyer, som i sin tur visade vad knapparna skriver ut, samt möjligheten att bläddra igenom menyer som också ändrade vad knapparna skriver ut.

Examensarbetet tar upp teori kring mikrokontroller, utvecklingsverktyg samt två kommunikationsprotokoll, USB och I^2C. Teorin går kort igenom historia och djupare igenom funktionaliteten för kommunikationsprotokollen samt mikrokontroller.

Utvecklingen av prototypen började med att välja komponenter, samt hur komponenterna skulle kopplas ihop, såsom en knappmatris, som blev implementerad i den här prototypen. Under utvecklingen av prototypen planerades också logiken för programmet, som behövdes under programmeringen så att det fanns en struktur att följa.

Tillverkningen av prototypen började med att få alla delar att fungera separat, efter att allt fungerade separat så kombinerades de. Prototypen var först byggd på kopplingsdäck, men ett kretskort blev också designat samt tillverkat, enligt hur det var uppbyggt på kopplingsdäcket.

Resultatet uppnådde inte helt målet, men som en första prototyp så gav den vad den skulle, vilket var kunskap om vad som skall förbättras.

Språk: svenska
Nyckelord: mikrokontroller, USB, I^2C, Arduino, tangentbord

OPINNÄYTETYÖ

Tekijä: Mats Sandqvist
Koulutus ja paikkakunta: Sähkö- ja automaatiotekniikka, Novia, Vaasa
Suuntautumisvaihtoehto: Automaatiotekniikka
Ohjaaja: Jan Berglund

Nimike: Ohjelmointipikanäppäimistön prototyyppi

Päivämäärä: 23.5.2024 Sivumäärä: 35 Liitteet: 1

Tiivistelmä

Tämä opinnäytetyö käsittelee ensimmäisen ohjelmointipikanäppäimistön prototyypin kehittämistä. Työn tarkoituksena oli nopeuttaa ohjelmointia, erityisesti kun toistettavia toimintoja on paljon, sekä minimoida kirjoitusvirheet ohjelmoinnin aikana. Työn tavoitteena oli saada toimiva prototyyppi, mikä tarkoitti piirilevyn suunnittelua näytöllä, joka näytti valikot, jotka puolestaan näyttivät, mitä näppäimet kirjoittavat, sekä mahdollisuuden selata valikoita, jotka myös muuttaisivat, mitä näppäimet kirjoittavat.

Opinnäytetyössä käsitellään mikrokontrollereihin, kehitystyökaluihin sekä kahteen viestintäprotokollaan, USB ja I²C, liittyvää teoriaa. Teoria käsittelee lyhyesti historiaa ja syvemmin viestintäprotokollien sekä mikrokontrollerien toiminnallisuutta.

Prototyypin kehittäminen alkoi komponenttien valinnalla ja sillä, miten komponentit liitettäisiin yhteen, kuten näppäinmatriisi, joka toteutettiin tässä prototyypissä. Prototyypin kehityksen aikana suunniteltiin myös ohjelman logiikka, jota tarvittiin ohjelmoinnin aikana, jotta olisi noudatettavissa oleva rakenne.

Prototyypin valmistus alkoi kaikkien osien erillisellä testaamisella. Kun kaikki toimivat erikseen, ne yhdistettiin. Prototyyppi rakennettiin ensin koekytkentälevylle, mutta myös piirilevy suunniteltiin ja valmistettiin koekytkentälevyn mukaisesti.

Tulokset eivät täysin saavuttaneet tavoitetta, mutta ensimmäisenä prototyyppinä se antoi tarvittavaa tietoa parannuskohteista.

Kieli: ruotsi
Avainsanat: mikrokontrolleri, USB, I²C, Arduino, näppäimistö

BACHELOR'S THESIS

Author: Mats Sandqvist
Degree Programme: Electrical and Automation Engineering, BSc, Novia, Vasa
Specialisation: Automation Technology
Supervisor(s): Jan Berglund

Title: Prototype of a Keyboard with Programming Shortcuts

Date: 23.5.2024 Number of pages: 35 Appendices: 1

Abstract

This thesis is about the development of a first prototype of a keyboard with programming shortcuts. The purpose of the work was to speed up programming, especially when many functions need to be repeated, as well as to minimize typing errors during programming. The goal of the work was to have a functioning prototype, which involved designing a circuit board with a display that showed menus, which in turn showed what the buttons output, as well as the ability to scroll through menus that also changed what the buttons output.

The thesis covers the theory around microcontrollers, development tools, and two communication protocols, USB and I²C. The theory briefly goes through the history and delves deeper into the functionality of the communication protocols and microcontrollers.

The development of the prototype began with selecting components and determining how they would be connected, such as a button matrix, which was implemented in this prototype. During the development of the prototype, the logic for the program was also planned, which was needed during programming to provide a structure to follow.

The manufacturing of the prototype began with getting all the parts to work separately. After everything worked individually, they were combined. The prototype was initially built on a breadboard, but a circuit board was also designed and manufactured based on the setup on the breadboard.

The result did not fully achieve the goal, but as a first prototype, it provided what it was supposed to, knowledge of what needs to be improved.

Language: Swedish
Key words: microcontroller, USB, I²C, Arduino, keyboard

Innehållsförteckning

1 Inledning

Detta examensarbete handlar om att bygga en prototyp av ett tangentbord med programmeringsgenvägar. Jag själv är uppdragsgivaren och Jan Berglund är min handledare och lektor vid Yrkeshögskolan Novia i Vasa. För att förverkliga projektet kommer jag att använda mig av en mikrokontroller.

I teoridelen kommer examensarbetet att ta upp mikrokontroller, kommunikation samt utvecklingsverktyg.

I utförandedelen kommer examensarbetet att ta upp hur det förverkligades, så som programmeringen samt kretskorttillverkning.

1.1 Syfte

Syftet med examensarbetet var att snabba upp programmering, speciellt när det är många funktioner som skall repeteras. Ett annat syfte var att se till så man slipper små irriterande skrivfel, som att i misstag lämna bort en karaktär, vilket man senare måste leta efter när programmet inte fungerar. Med tangentbordet är det meningen att man trycker på en tangent som skriver ut en hel funktion på en gång, vilket man i sin tur bara behöver fylla i kriterierna.

1.2 Mål

Målet med examensarbetet var att ha en fungerande prototyp, vilket innebär att ha designat ett kretskort, samt att man kan bläddra igenom menyer, en display visar vilken funktion skrivs ut med vilken knapp samt att funktionerna skrivs ut vid ett knapptryck.

1.3 Avgränsning

Examensarbetet kommer att täcka utvecklingen av en första prototyp. I första prototypen kommer Arduino IDE C++ språket att finnas, i senare prototyper är det meningen att fler programmeringsspråk skall sättas till, samt att bygga på det design val som blev bra samt ändra det design val som inte blev så bra.

2 Teori

Teoridelen täcker mikrokontrollers, utvecklingsverktyg, samt kommunikationsprotokollen USB och I²C.

2.1 Mikrokontroller

Mikrokontrollerdelen täcker historia samt funktionaliteten av mikrokontrollern, samt olika utvecklingskort.

2.1.1 Historia

mikrokontrollern uppfanns 1971 i USA, men vem som uppfann den först är lite oklart. Vissa källor säger att det var Intel med sin 4bit i4004 mikrokontroller som var först, medan vissa källor säger att det var Gary Boone och Michael Cochran från Texas instrument med sin 4bit TMS 1000 mikrokontroller som var först. (Toschiba, n.d.; Wright Jr.).

Intels i4004 mikrokontroller började sitt liv i miniräknare och sedan som en universal mikrokontroller. Efter succén av i4004 så utvecklade Intel också en 8bit mikrokontroller samt en 16bit mikrokontroller. Efter att Intel hade utvecklat flera mikrokontroller så fortsatte dom att utveckla processorer för personliga datorer. (Toschiba, n.d.).

Texas instruments TMS 1000 mikrokontroller började också sitt liv i en miniräknare, men till skillnad från Intel så användes TMS 1000 mikrokontrollern internt, eftersom Texas instruments tillverkade sin egen miniräknare. Mellan 1972 till 1974 så fortsatte dom att utveckla TMS 1000 mikrokontrollern för sin miniräknare och 1974 så började dom också att sälja dem externt. (Wright Jr.).

Under 1990-talet kom mikrokontrollers med elektriskt raderbart och programmerbart läsminne, electrically erasable and programmable read-only memory på engelska, eller EEPROM som förkortningen, vilket gjorde det enklare samt snabbare att programmera om en mikrokontroller. (Wright Jr.).

2.1.2 Mikrokontroller funktion

En mikrokontroller, MCU, är en liten och billig dator på en integrerad krets, som är designad att kontrollera olika uppgifter i ett elektroniskt system. En mikrokontroller kombinerar funktionaliteten av en central processor, CPU, med lång och kort tids minne samt ingång och utgång, allt på ett enda chip.

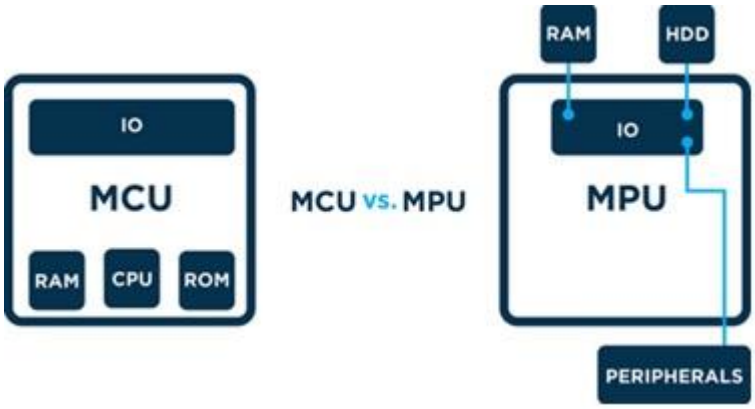
Processorn på en mikrokontroller kan man se som hjärnan i enheten, den kör programkoden och kontrollerar allt enligt instruktionerna i koden, vilket programspråk som används beror på tillverkare men två vanliga språk som används är C samt C++, som sen kompileras till ett språk som mikrokontrollern förstår. Processorn läser också och tolkar data från ingångar, som behandlas enligt instruktionerna i koden, samt skickar data på utgångar.

Minnet i mikrokontrollern används för att lagra data som processorn tar emot och används för att agera enligt det programmerade instruktionerna. Det finns två olika minnen i en mikrokontroller, programminnet samt dataminnet. Programminnet används för att lagra information över en lång tid, så som programkoden. Programminnet är ett så kallat icke-flyktigt minne, vilket betyder att det kan hålla information utan tillsatt ström. Dataminnet används för temporär lagring av data medan instruktioner körs i processorn. Dataminnet är ett så kallat flyktigt minne, vilket betyder att det kan bara hålla information så länge som det har tillsatt ström. (Geeksforgeeks, 2023; Tutorialspoint, n.d.; Lutkevich, 2019).

Ingångarna och utgångarna på mikrokontrollern är länken mellan processorn och världen. Ingångarna tar emot information och skickar det till processorn, efter att processorn har behandlat informationen skickar den instruktioner till utgångarna, som för vidare instruktionerna till externa enheter. (Lutkevich, 2019).

Microcontrollers används i många olika system, så som säkerhetssystem, robotar, hushållsapparater, miniräknare samt sjukhusutrustning. Det finns också många system där man har flera mikrokontrollers som har sina egna ansvarsområden, men samtidigt samarbetar genom att kommunicera med varandra, så som i fordon. (Geeksforgeeks, 2023; Tutorialspoint, n.d.; Lutkevich, 2019).

Skillnaden mellan en mikrokontroller och en mikroprocessor, MPU, börjar se mindre och mindre ut, men det är fortfarande en skillnad. Som det redan har nämnts så har en mikrokontroller allt från en liten dator på ett chip, som processor och minne, medan en mikroprocessor bara är en del i en dator. Mikrokontroller har inget internt minne, bara externt, men en mikroprocessor är dock mycket kraftigare och passar bättre för mer krävande uppgifter. (Geeksforgeeks, 2023; Tutorialspoint, n.d.; Lutkevich, 2019).

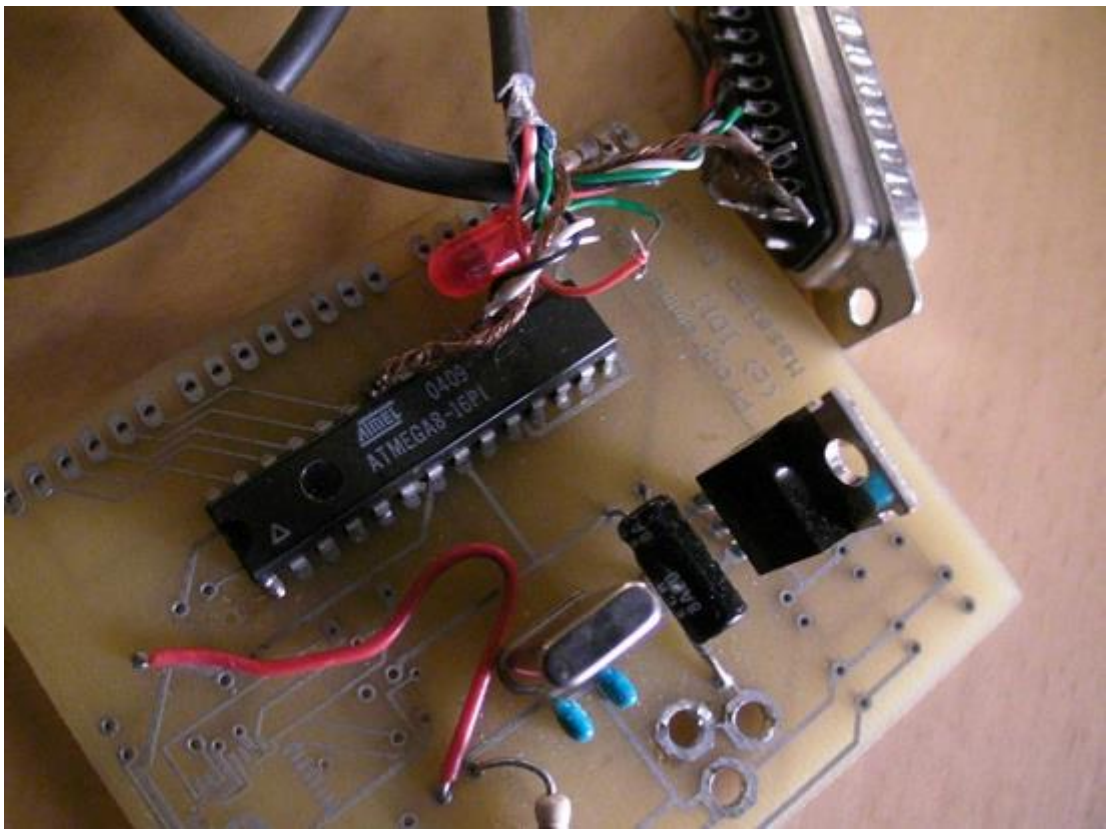


Figur 1: Skillnaden mellan en mikrokontroller och en mikroprocessor. (ST, 2022).

2.1.3 Arduino

Arduino började som ett magisteravhandling projekt vid Interaction Design Institute Ivrea, i Ivera i Italien av Hernando Barragán år 2004. Hernando Barragán skapade utvecklings plattformen Wiring, med målet att skapa ett enkelt och billigt verktyg så att icke ingenjörer skulle kunna skapa digitala projekt. Wiring plattformen bestod av ett kretskort med en ATmega mikrokontroller och en IDE baserad på bearbetning, processing på engelska, och bibliotek funktioner för att göra det lätt att programmera mikrokontrollern. (Källkritik, 2024).

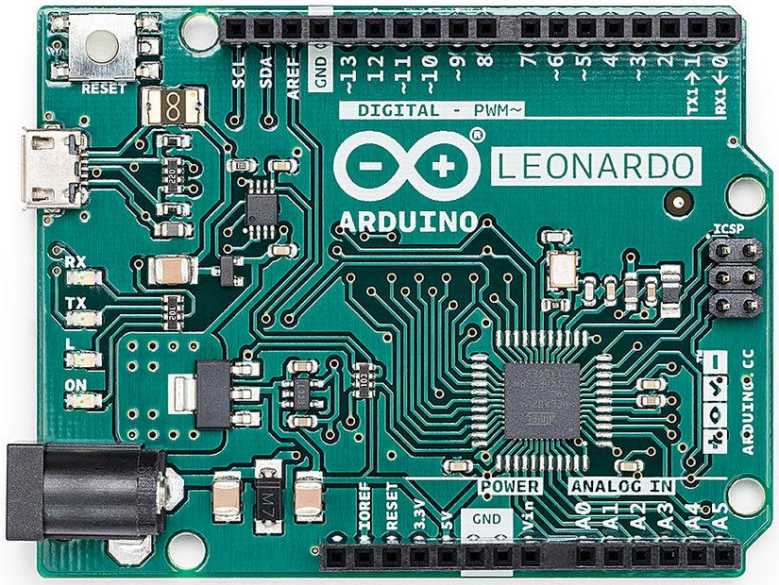
Massimo Banzi med David Mellis som också studerade vid Interaction Design Institute Ivrea, samt David Cuartielles, fortsatte år 2005 jobba på Wiring, genom att lägga till support för ATmega8, som är en billigare mikrokontroller, och det här nya projektet blev då kallat Arduino. (Källkritik, 2024).



Figur 2: Första Arduinon. (Källkritik, 2024).

Arduino är nu en lätt använd öppen källkod elektronik plattform, som över åren har använts i allt från vardagliga projekt till komplexa vetenskapliga instrument. Arduino har nu också en världsomfattande användargrupp, av allt från hobbyister, studerande, programmerare samt professionella, som har samlats runt den här öppna källkods plattformen, och med allas insatser finns en otrolig mängd kunskap som alla kan ta del av. (Arduino, 2018).

Det är på grund av all lättåtkomlig kunskap som många väljer att använda en Arduino, och en Arduino Leonardo till exempel på grund av att den använder sig av en ATmega32u4, som har en inbyggd USB-kommunikator, vilket eliminerar behovet för en till processor ifall man vill att kortet skall kommunicera över USB. (Arduino, 2018).



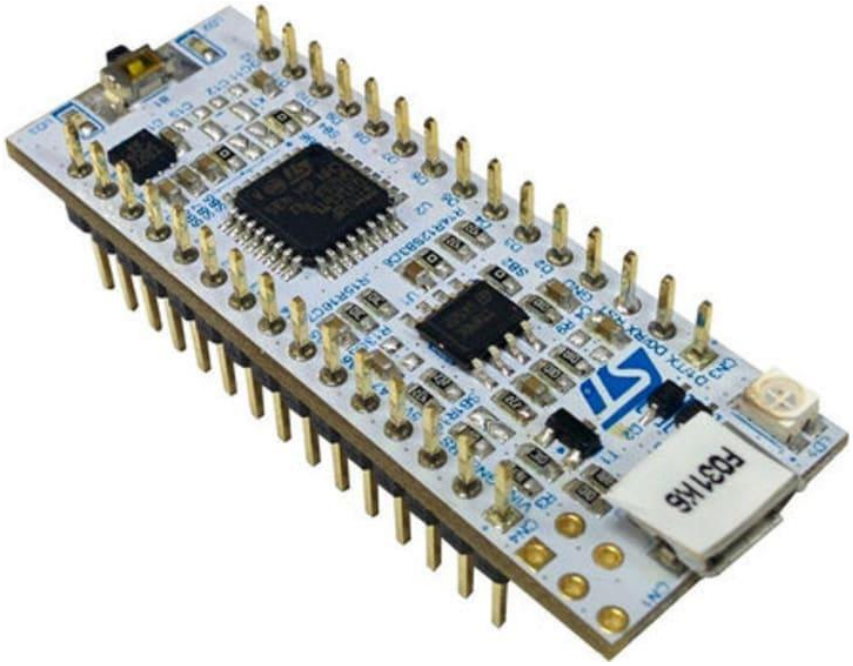
Figur 3: Arduino Leonardo utvecklingskort. (Arduino, n.d.).

Arduino har också många andra utvecklingskort man kan välja på för att hitta det som passar best för ens eget projekt, vare sig om man behöver många in och utgångar, ett fysiskt litet kort eller om man behöver trådlösa möjligheter. Arduino har nu också utvecklat en PLC med samma inställning till att dem skall vara ett billigt och lättanvänt alternativ, utan att kompromissa på prestanda. (Arduino, 2018).

2.1.4 STM32

STM32 är en familj av 32-bit mikrokontrollers av STMicroelectronics, ett utvecklingskort med STM32 skulle ha kunnat vara en alternativ för den här prototypen. STM32 är olika versioner av ARM Cortex-M kärnor, STM32 släpptes 2007 och har varit i kontinuerlig utveckling sen dess, deras senaste version släpptes 2023, det finns flera olika mjukvaror man kan använda för att programmera en STM32 och Arduino IDE är en av dem. (ST, 2022).

Största skillnaden mellan ett STM-utvecklingskort och ett Arduino utvecklingskort är att STM är mycket kraftigare och snabbare samt mera minne att lagra kod samt data.

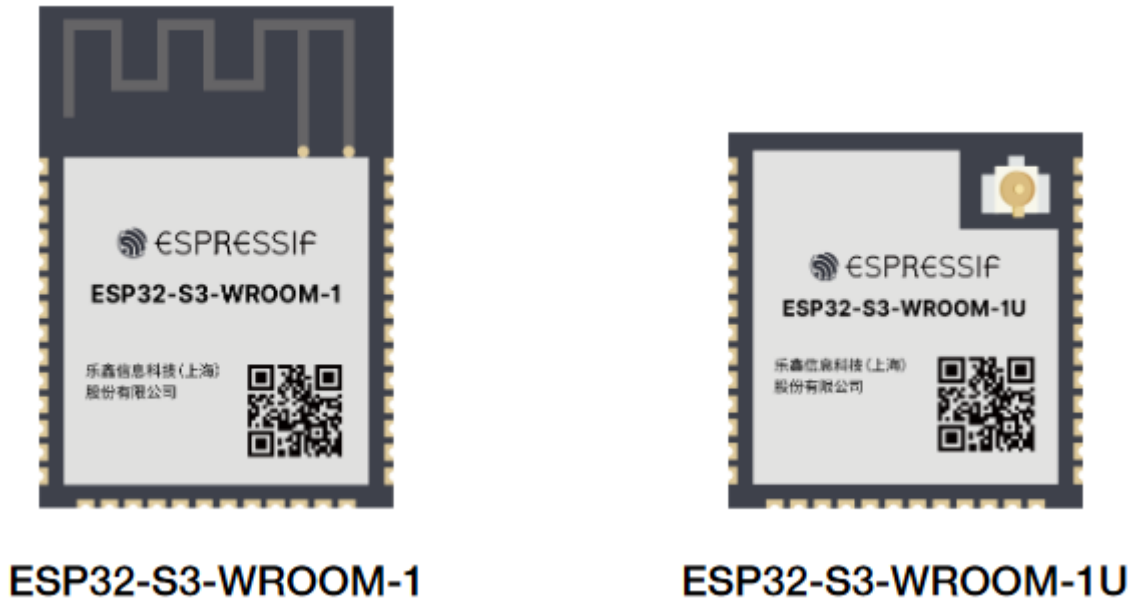


Figur 4: STMicroelectronics STM32 NUCLEO-L031K6. (RS, n.d.).

2.1.5 ESP

Espressif grundades 2008 och är företaget bakom ESP8266, samt alla varianter av ESP32 mikrokontrollers, vilket skulle ha kunnat användas som ett alternativ för den här prototypen. ESP32 är en billig och energisnål mikrokontroller med inbyggd WIFI samt Bluetooth. ESP32 använder sig antingen av Tensilica Xtensa LX6 mikroprocessor i både dubbel kärniga eller enkel kärniga varianter, ESP32 kan också använda en Xtensa LX7 dubbel kärnig mikroprocessor eller en enkel kärnig RISC-V mikroprocessor. (Espressif, n.d.).

Största skillnaden mellan ett ESP-utvecklingskort och ett Arduino utvecklingskort är att ESP är mycket kraftigare, snabbare samt mera minne att lagra kod och data samt att den har inbyggd trådlös kommunikation.



Figur 5: ESP32-S3. (Espressif).

2.2 Utvecklingsverktyg

Utvecklingsverktygs delen täcker Arduino IDE samt Kicad.

2.2.1 Arduino IDE

Arduino IDE är baserat på Processing, vilket är en utvecklings plattform som grundades 2001. Arduino IDE är lätt använt för nybörjare men samtidigt också tillräckligt flexibelt för avancerade användare, Arduino IDE är också plattformsoberoende, vilket innebär att man kan använda det på Windows, macOS samt Linux, till skillnad från många andra plattformar som är begränsade till Windows. (Arduino, 2018).

Arduino IDE är ett öppen källkod-verktyg som man kan ladda ner mera bibliotek till som andra användare har skapat för att öka funktionsmöjligheter. För dem som vill förstå det tekniska detaljerna bättre kan gå från Arduino till AVR-C språket som det är baserat på, och skriva program med att använda det i stället. Arduino IDE kan användas för att programmera mer en bara Arduino utvecklingskort, genom att ladda ner bibliotek för andra utvecklingskort som är baserat på till exempel STM eller ESP, så kan man programmera dem med. (Arduino, 2018).

IDE står för Integrated Development Environment, och är en mjukvara som har dom verktyg som behövs för att skriva kod, samt testa den. IDEn har en compiler och en debugger, compilern översätter koden till en kod som mikrokontrollern förstår och debuggern går systematisk igenom koden för att se om den hittar något fel i koden. (Söderby & De Feo, 2024).

2.2.2 Kicad

Kicad är en öppen källkod Electronic Design Automation, EDA, mjukvara. Med Kicad kan man göra el scheman samt designa kretskort, programmet fungerar på Windows, Linux samt MacOS. Kicad släpptes 1992 av Jean-Pierre Charras, och har varit i kontinuerlig utveckling sen dess, och styrs nu av kicads utvecklingsgrupp. (Kicad, n.d.).

Kicads mål är att skapa den bästa plattformsoberoende elektronikdesign mjukvara för professionella elektronikdesigners. Varje beslut som Kicad tar för vidare utveckling av mjukvaran är för professionella, men samtidigt försöker Kicad också gömma allt som verkar svårt och avancerat, så att ny användare lätt kan komma i gång och inte bli avskräckta. (Kicad, n.d.).

2.3 USB-kommunikation

USB-kommunikations delen täcker historia samt funktionaliteten av USB-kommunikation.

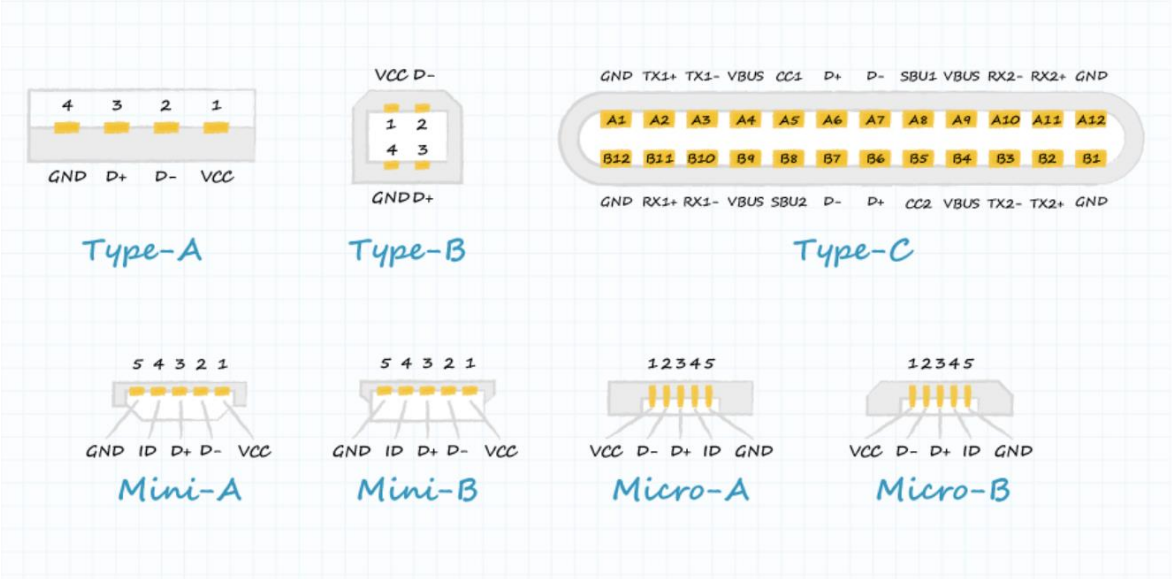
2.3.1 Historia

USB är en förkortning av namnet, Universal Serial Bus, vilket är resultatet av ett samarbete som började 1995 mellan 7 företag, Compaq, DEC, IBM, Intel, Microsoft, NEC och Nortel. Målet med USB var att göra det lättare att koppla externa enheter till ens dator samt att göra av med alla olika kontakter som fanns bak på en dator då, samtidigt också göra det lättare för företag att göra mjukvara som kunde kommunicera med olika externa enheter samt att få bättre dataöverföring. År 1996 släpptes USB med en väl kända kontakten USB-A och standarden USB 1.0. (Källkritik, 2024).

Tabell 1. USB standard, år när dem släpptes samt dataöverföring.

Standard	USB 1.0 1996	USB 1.1 1998	USB 2.0 2001	USB 3.0 2008	USB 3.1 2013	USB 3.2 2017	USB 4 2019	USB 4 V2.0 2022
Max Hastighet	1.5 Mbit/s & 12 Mbit/s		480 Mbit/s	5 Gbit/s	10 Gbit/s	20 Gbit/s	40 Gbit/s	80 Gbit/s

(Källkritik, 2024).



Figur 6. Olika typer av USB-kontakter. (Smoot, n.d.).

2.3.2 USB funktion

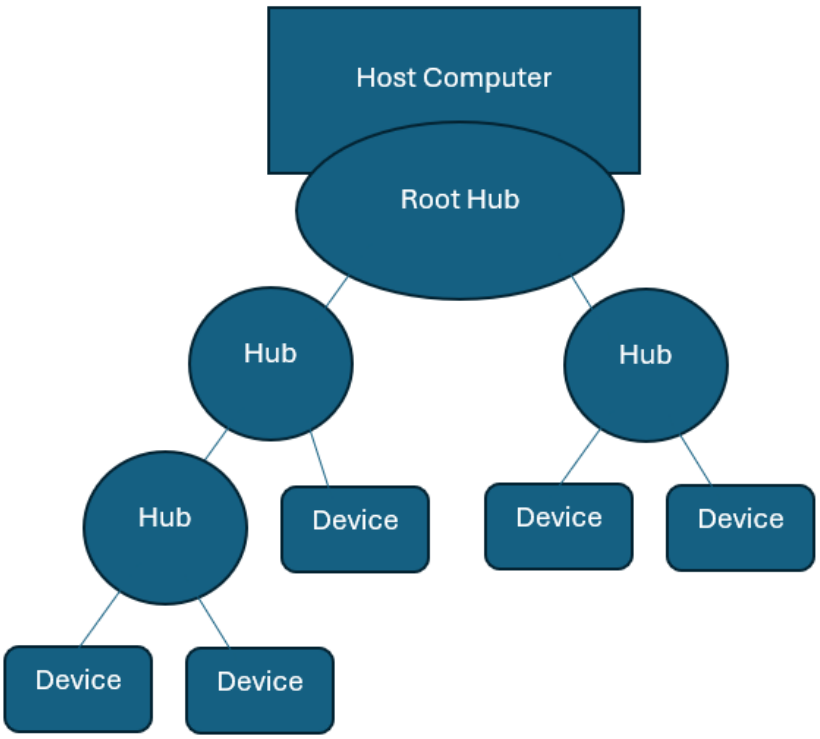
USB-protokollet gör det möjligt att kommunicera med datorn mellan flera olika enheter, så som möss, tangentbord, printer samt externa minnen för att nämna några.

USB-protokollet erbjuder också plug-N- play och hot swapping. Med plug-N- play menas att datorn automatisk upptäcker ifall en ny enhet är kopplad till datorn, och med hot swapping menas att man inte behöver stänga av eller starta om datorn för att koppla till eller koppla bort enheter, utan i stället får datorn vara i gång hela tiden.

USB fungerar med en omröstningsprincip, vilket betyder att processorn konstant kollar om tillkopplade enheter är beredda att sända data eller inte. Eftersom den tillkopplade enheten inte behöver uppdatera processorn om sitt tillstånd själv, utan processorn tar hand om det, gör det USB enklare samt billigare.

När datorn märker att en ny enhet har kopplats till, så läser datorn den tillgängliga data på den ny tillkopplade enhetens minne för att få reda på enhetens funktioner, så att datorn kan använda en lämplig driver för att kommunicera med enheten. Efteråt så ger datorn en adress till den nya enheten som sparas på enhetens egna register.

USB-arkitekturen fungerar på så vis att, när flera enheter är kopplade till datorn så blir dom strukturerad som ett träd. Varje enhet i USB strukturen gör en piont-to-point kontant för att sända data genom det seriella överföringsformatet. Enheterna i den här arkitekturen som är kopplade till datorn via USB, som kallas som en hubb. Inom arkitekturen är hubbarna kopplingspunkten mellan enheterna och datorn, rot hubben inom arkitekturen används för att koppla ihop hela strukturen till datorn. (Elprocus; Future Technology Devices International Ltd., 2009).



Figur 7: USB arkitektur. (Elprocus).

I USB kommunikationen skickas paket och paketen är uppbyggda av flera olika fält.

Varje USB-paket börjar med ett SYNC fält, det här fältet synkroniserar avsändaren och mottagaren så att paketet kan sändas korrekt. För låg och hög hastighet är SYNC fältet 8 bitar lång, och för max hastighet är fältet 32 bitar långt.

PID fältet, Packet Identifier Field, kommer direkt efter SYNC fältet och används för att identifiera vilken typ av paket som blir sänt samt paketets data format. PID-paketet är 8 bitar långt.

Adressfältet i paketet indikerar till vilken enhet paketet är adresserad till. Adressfältet är 7 bitar långt vilket tillåter 127 enheter, noll adressen är inte giltig eftersom den är ämnad som adress för enheter som ännu inte blivit tilldelad en adress ännu.

Endpoint Field är 4 bitar lång och det här fältet tillåter extra flexibilitet för adressering, vanligtvis är det här uppdelat mellan data som kommer in och data som kommer ut. Endpoint 0 är ett special fall som kallas, Control endpoint, och varje enhet har en endpoint 0. (Elprocus; Future Technology Devices International Ltd., 2009)

Datafältet har inte en exat längd, utan längden är mellan 0 till 8192 bitar, och är alltid en integral numer av bytes.

CRC fältet, Cyclic Redundancy Checks, används för att granska så det inte blev något fel i paketet. Det finns två olika CRC fält, ett 5 bitars CRC fält som används i Token paketet samt i Start of frame paketet, det finns också ett 16 bitars CRC fält som används i datapaketet.

Varje paket innehåller ett EOP fält, End of Packet, vilket är 3 bitar långt och avslutar paketet.

USB TOKEN paketet används för att få tag på rätt adress samt endpoint, och ser ut som på bilden nedan.

FIELD	SYNC	PID	ADDRESS	ENDPOINT	CRC5	EOP
#BITS	8/32	8	7	4	5	3

Figur 8: TOKEN paket. (Future Technology Devices International Ltd., 2009).

USB-datapaketet kan vara olika längder, beroende på vad som sänds, datapaketet ser ut som på bilden nedan.

FIELD	SYNC	PID	DATA	CRC16	EOP
#BITS	8/32	8	0-8192	16	3

Figur 9: Datapaket. (Future Technology Devices International Ltd., 2009).

USB Handshake paketet används för att signalera transaktionens status, om det lyckades eller misslyckades. Handshake paketet ser ut som på bilden nedan.

FIELD	SYNC	PID	EOP
#BITS	8/32	8	3

Figur 10: Handshake paket. (Future Technology Devices International Ltd., 2009).

För att slutföra ett fullständigt meddelande grupperas paketen i ramar. För att identifiera en ram behövs ett Start of Frame packet, SOP, ram nummern är 11 bitar lång. SOP-paketet ser ut som på bilden nedan.

FIELD	SYNC	PID	Frame Number	CRC5	EOP
#BITS	8/32	8	11	5	3

Figur 11: SOP-paket. (Future Technology Devices International Ltd., 2009).

2.4 I²C -kommunikation

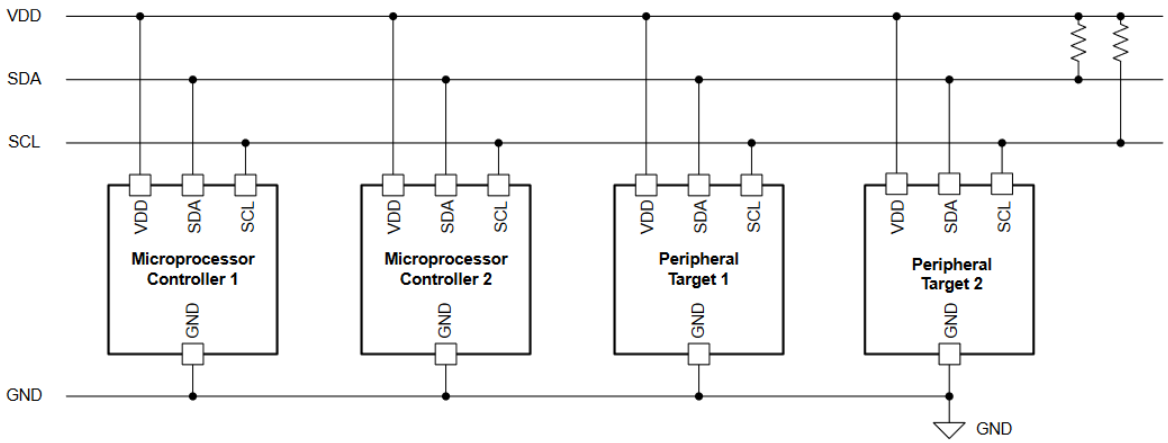
I²C -kommunikations delen täcker historia samt funktionaliteten av I²C -kommunikation.

2.4.1 Historia

I²C, som ofta kallas I two c, står för Inter-Integrated Circuit protocol. I²C utvecklades 1982 av Philips Semiconductor, som nu heter NXP Semiconductor, som ett låg hastighets kommunikationsprotokoll för controller enheter som mikrokontrollers och processorer, för att kommunicera med externa enheter. Sedan 2006 har I²C protokollet kunnat användas utan licens, och sen dess har många halvledare enhets företag introducerat I²C kompatibla enheter. (Wu, 2022).

2.4.2 I²C -funktion

I²C är ett två linje, two wire på engelska, seriekommunikations protokoll, som använder en serial data line, SDA, samt en serial clock line, SCA, för kommunikation. Med I²C -protokollet kan man koppla flera enheter till samma kommunikations bus, och samtidigt ha flera controllers som skickar och tar emot kommandon samt data. Data skickas i byte-paket med en unik adress för varje enhet. I²Cs två kommunikations bus används för halv duplexkommunikation. Båda linjerna har uppdragningsmotstånd för att hålla bussen hög, vilket indikerar att den är öppen för kommunikation. (Wu, 2022).

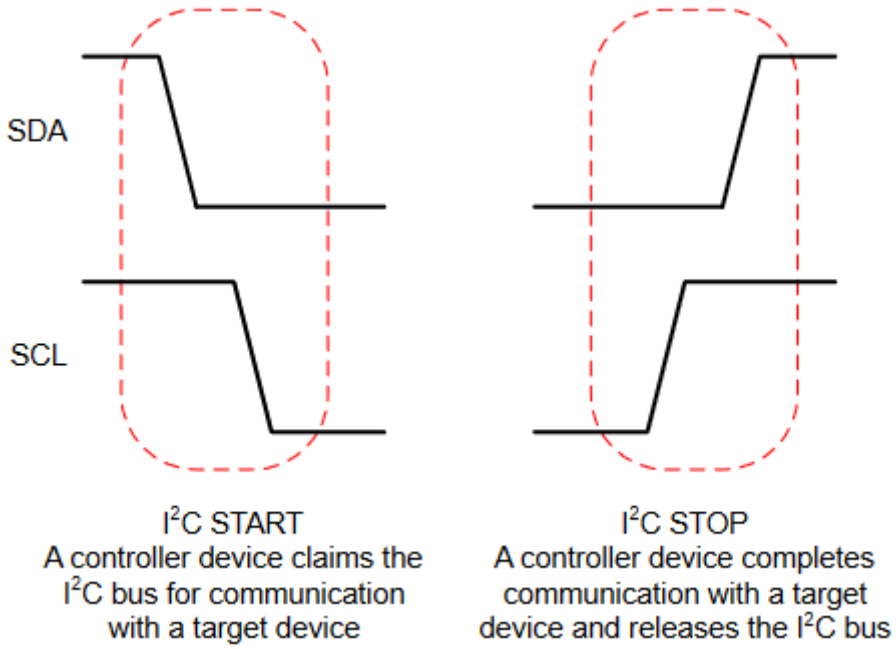


Figur 12: En implementation av I²C. (Wu, 2022).

En orsak till att I²C är populärt är på grund av att det bara använder två linjer för kommunikation. SCL -linjen kontrolleras primärt av controllerenheten, och används för att synkronisera data in eller ut ur en tillkopplad enhet. SDA -linjen används för att sända data till eller från en tillkopplad enhet.

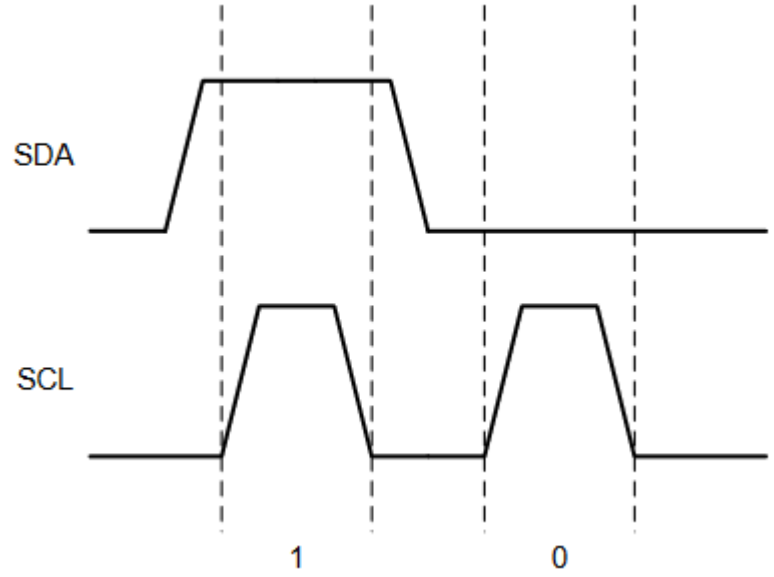
Som tidigare nämnt är I²C en halv duplexkommunikation, vilket betyder att endast en controller eller enhet kan skicka data i taget. En controller startar samt stoppar kommunikationen, vilket eliminerar busstocknings problem. Till skillnad så är SPI, serial peripheral interface, ett hel duplexkommunikations protokoll, vilket betyder att data kan skickas samt tas emot samtidigt. En annan skillnad är också att SPI använder fyra linjer för kommunikation. Två datalinjer för att sända data till och från tillkopplade enheter, en SCL -linje samt en SPI chip select line, som väljer till vilken enhet kommunikationen skall gå till.

En I²C -kommunikation börjar på så vis att, om bussen är öppen så skickar en controller ett I²C -START kommando över bussen, för att ta kontroll över den för kommunikation. För att göra det, drar controllern först SDA -linjen låg, och sen drar den SCL -linjen låg, det här indikerar att controllern har tagit kontroll över bussen, vilket tvingar andra enheter att vänta på sin tur för att skicka data. När controllern har kommunicerat klart så släpper den först SCL -linjen hög och sen släpper den SDA -linjen hög, vilket indikerar I²C -STOP. När båda linjerna är höga betyder det att bussen är öppen för andra eller samma enhet att ta kontroll och kommunicera. (Wu, 2022).



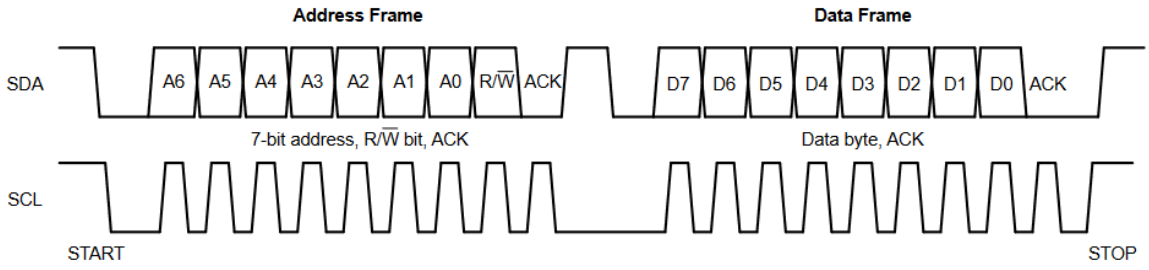
Figur 13: I²C start och stop. (Wu, 2022).

En sekvens av ettor och nollor används för I²C -kommunikation. Data-bits sänds över SDA linjen medan SCL -linjen timar bit sekvenserna. En logisk etta sänds när SDA linjen släpps hög, och en logisk nolla sänds när SDA -linjen dras låg. När SCL pulseras tas ettor och nollor emot. Signalen på SDA l-injen ändras inte medan signalen stiger eller dras ner på SCL -linjen, ifall signalen ändras på SDA -linjen medan den ändras på SCL -linjen kan det tolkas som ett start- eller stop-kommando.



Figur 14: Logisk noll och ett i I²C -kommunikation. (Wu, 2022).

I²C -kommunikation är uppdelad i ramar, som är en byte lång. Controllern börjar kommunikationen med en start, var efter en adressram skickas, efter adress ramen skickas också en eller flera data ramar. Varje ram har också en bekräftnings bit, ACK, som meddelar controllern att ramen har tagits emot av enheten som den försöker kommunicera med. (Wu, 2022).



Figur 15: I²C -kommunikations ramar. (Wu, 2022).

3 Praktiska delen

Praktiska delen täcker planering samt utförandet.

3.1 Planering

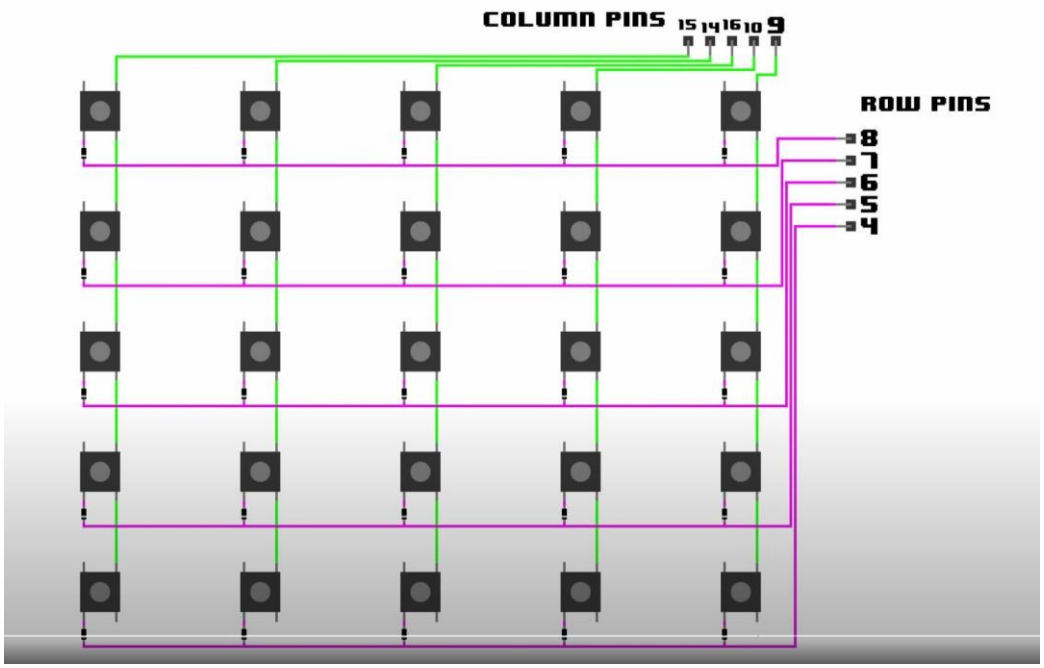
Planerings delen täcker hårdvara samt mjukvarulogik.

3.1.1 Hårdvara

Planeringen började med att välja utvecklingskort, vilket blev ett Arduino Leonardo, eftersom med den kan man direkt kommunicera över USB som ett tangentbord utan extra komponenter samt att det finns ganska mycket information om Arduino.

Som display valdes en 1,3” 128x64 OLED display med I²C -kommunikation. Orsaken till valet var att 1,3” skulle vara tillräckligt stort för en första prototyp för att få en idé om hur stor display man behöver för nästa prototyp, samt att I²C tar upp mindre in och utgångar än SPI.

För att bläddra igenom menyerna valdes en horisontell pulsgivare, rotary encoder på engelska, eftersom det ansågs som det skulle vara lämpligt. För själva tangentbordet valde jag att koppla knapparna i en knappmatris, så att det skulle använda mindre in och utgångar än att koppla en och en.



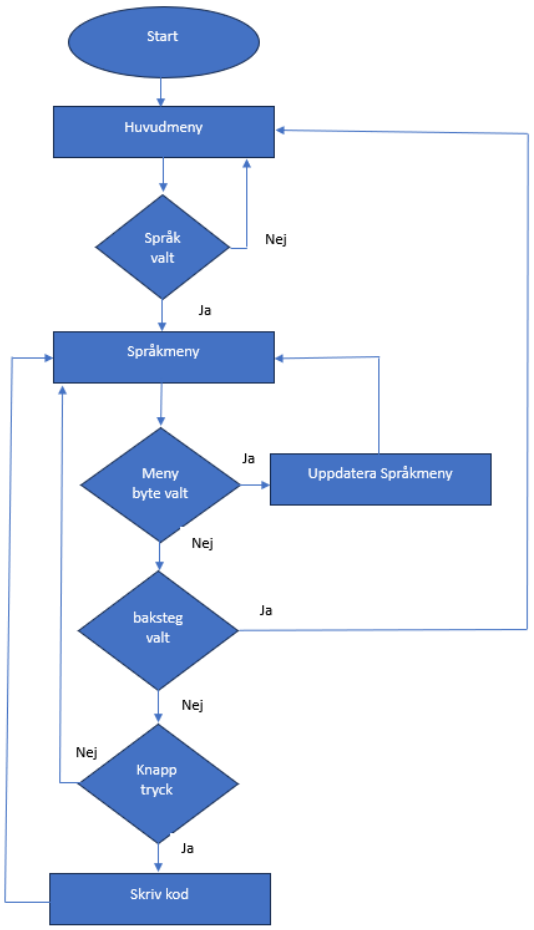
Figur 16: Exempel på en knappmatris. (RassieRaider, 2022).

3.1.2 Mjukvarulogik

I figur 17 nedan kan man se logiken jag hade planerat.

När programmet startar går det direkt till huvudmenyn där man får välja vilket programmeringsspråk man vill använda, det kontrollerar också hela tiden om man tryckt på pulsgivaren och valt ett språk.

När man valt ett språk så går programmet vidare till språkmenyn, vilket visar på displayen vad knapparna i matrisen gör om man trycker på dem. Programmet kör konstant igenom pulsgivaren samt knapparna för att se om man gör någonting. Ifall man vrider på pulsgivaren så uppdateras språkmenyn och visar nya kommandon som knapparna gör om dom blir tryckta. Om man trycker på en knapp i matrisen så skrivs koden ut på dator enligt vad som står på displayen att den knappen skall skriva ut. Om man trycker på pulsgivaren så går man tillbaka till huvudmenyn.



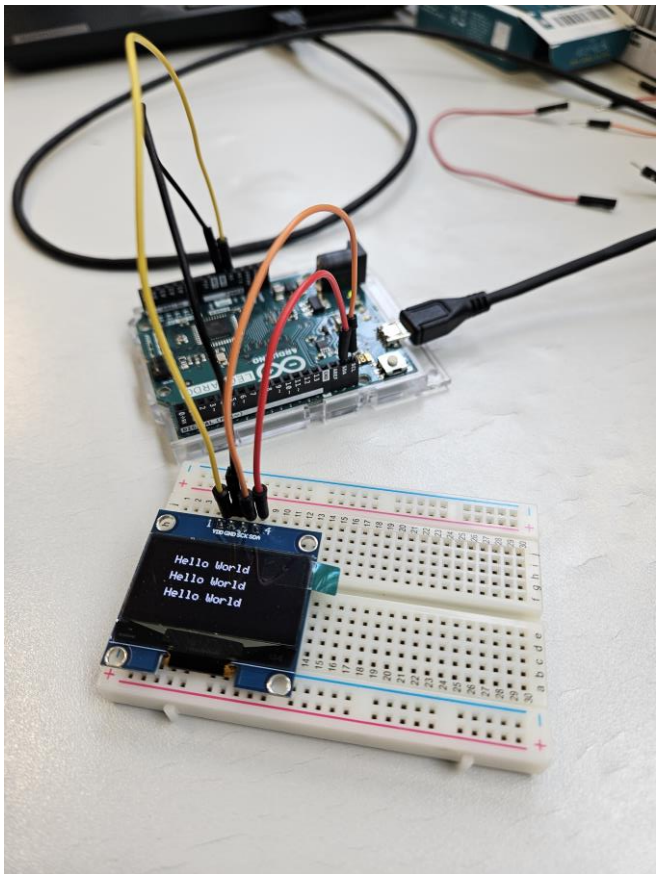
Figur 17: Mjukvarulogik.

3.2 Utförande

Utförande delen täcker första stegen samt första prototypen.

3.2.1 Första steg

Till att börja med så sågs det till att alla olika delar fungerade separat. Först började det med displayen, vilket innebar att hitta rätt bibliotek samt I²C -adress för displayen, samt att förstå sig på koden så att man kunde få displayen att visa vad man ville.



Figur 18: Display-test.

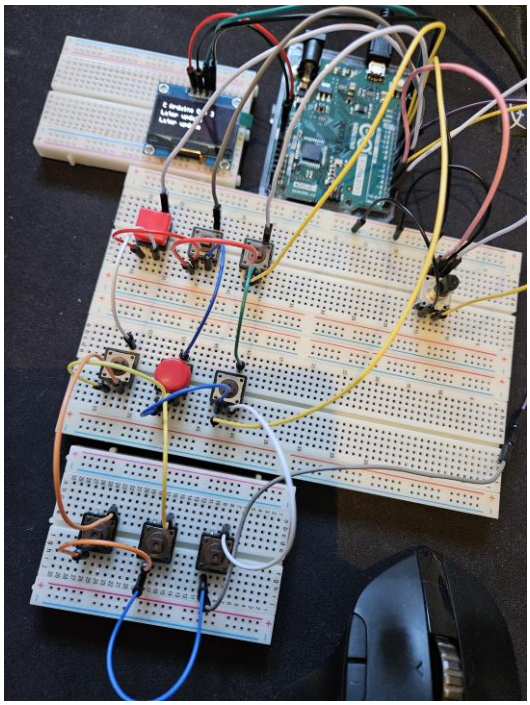
Som nästa steg blev det att få tangentbords funktionen att fungera, vilket var bara att använda tangentbordsbiblioteket och tilldela ett kommando till en knapp, sen kunde man skriva ut vad man ville på datorn. Knappmatrisen var lika lätt att få att fungera, det var bara att använda keypad-biblioteket och koppla knapparna enligt ett knappmatrisexempel. Det fans exempel av knappmatriser med både dioder samt utan dioder, för den här prototypen användes hoppkablar eftersom det inte fungerade med dioder.

Med pulsgivaren så användes en exempelkod som modifierades lite för att passa med hur den behövde fungera. Koden hade också anti hopp funktioner i sig, så ingen extern hårdvara behövdes för att få en stabil signal.

3.2.2 Första prototypen

Först började det med att bygga första prototypen på kopplingsdäck, breadboard på engelska. Efter att allt fungerade separat så sattes jag alla delar ihop, samt satt ihop dom delar av koden som behövdes från alla olika delar, samt skrev mer kod som behövdes. Det här fungerade inte, så fort man tryckte på pulsgivaren så låste sig mikrokontrollern och började printa ut fyrkanter på serial monitorn, inget hände när man tryckte på knapparna och man kunde inte ladda upp ny kod före man hade kopplat bort allt från mikrokontrollern och tryckt på reset några gånger.

När det inte fungerade att kombinera allt togs några steg tillbaka, och det började med att kombinera pulsgivaren och knappmatrisen samt skriva om koden så att den blev enklare. Efter att pulsgivaren och matrisen började att fungera tillsammans så sattes displayen till.



Figur 19: Första prototypen på kopplingsdäck.

Nedan i kod 1 är biblioteken som användes, matrisen och tangentbords funktionen behövde bara ett bibliotek var medan displayen behövde tre. Adafruit biblioteken är för själva displayen och wire är för I²C -kommunikation.

Kod 1: Bibliotek.

```
// Include the Keypad library
#include <Keypad.h>
// Include keyboard
#include "Keyboard.h"
// Include display librariy and I2C library
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SH110X.h>
```

I kod 2 definieras I²C -adressen samt display storleken i pixlar och reset kommando.

Kod 2: I²C och display.

```
/* Uncomment the initialize the I2C address , uncomment only one, If you get
a totally blank screen try the other*/
#define i2c_Address 0x3c //initialize with the I2C addr 0x3C Typically eBay
OLED's
//#define i2c_Address 0x3d //initialize with the I2C addr 0x3D Typically
Adafruit OLED's

#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels
#define OLED_RESET -1   //   QT-PY / XIAO
Adafruit_SH1106G display = Adafruit_SH1106G(SCREEN_WIDTH, SCREEN_HEIGHT,
&Wire, OLED_RESET);
```

I kod 3 skapas knappmatrisen. I koden anges hur många kolumner och rader som matrisen skall ha samt värdet för alla knappar, det anges också till vilka ingångar på Arduinon som dom går till.

Kod 3: Knappmatris.

```
// Constants for row and column sizes
const byte ROWS = 3;
const byte COLS = 3;

char hexaKeys[ROWS][COLS] = {
  {'1', '2', '3'},
  {'4', '5', '6'},
  {'7', '8', '9'}

};

byte rowPins[ROWS] = {10, 9, 8};
byte colPins[COLS] = {7, 6, 5};

// Create keypad object
Keypad customKeypad = Keypad(makeKeymap(hexaKeys), rowPins, colPins, ROWS,
COLS);
```

I kod 4 anges ingångarna på Arduinon från pulsgivaren.

Kod 4: Pulsgivare.

```
// Rotary RotaryEncoder Pins
#define ENC_A 1
#define ENC_B 2
#define ENCODER_BUTTON 12
```

I kod 5 ser man inställnings funktionen som körs först för att ställa in alla inställningar. Först startar den serial kommunikationen, efteråt startas I²C -kommunikationen och displayen, samt textstorlek samt färg väljs för displayen. Sen anges interna uppdragningsmotstånd åt pulsgivaren så att man inte behöver externa för att få en stabil signal. Till sist väljs vilken typ av tangentbord som används samt funktionen för startmenyn för displayen kallas, kod 11.

Kod 5: Inställningsfunktionen.

```
void setup() {
  // Initialize serial communication
  Serial.begin(9600);

  // Show image buffer on the display hardware.
  // Since the buffer is intialized with an Adafruit splashscreen
  // internally, this will display the splashscreen.

  delay(250); // wait for the OLED to power up
  display.begin(i2c_Address, true); // Address 0x3C default


  display.display();
  delay(2000);

  // Clear the buffer.
  display.clearDisplay();

  // text display
  display.setTextSize(1);
  display.setTextColor(SH110X_WHITE);
  display.setCursor(0, 0);

  display.display();
  delay(2000);
  display.clearDisplay();

  // Set RotaryEncoder pins as inputs
  pinMode(ENC_A, INPUT_PULLUP);
  pinMode(ENC_B, INPUT_PULLUP);

  // Set button pin as input with internal pull-up resistor
  pinMode(ENCODER_BUTTON, INPUT_PULLUP);

  // Selects keyboard
  Keyboard.begin(KeyboardLayout_sv_SE);


  // Start menu
  MENU_DISP();
}
```

I kod 6 ser vi loop-funktionen, vilket är en funktion som körs hela tiden. I början ser vi att två andra funktioner blir kallade, båda är för pulsgivaren, kod 7 samt 8. Efteråt är en if-sats som körs om man har valt Arduino som språk i startmenyn. Efteråt är en till if-sats som körs om man trycker på en knapp. När man tryckt på knappen så kallar det på en funktion från casen enligt vilken språkmeny man valt, dom olika funktionerna som kallas ger olika funktioner till knapparna, kod 9.

Kod 6: Loop-funktionen.

```
void loop() {

  RotaryEncoder();
  EncoderButton();

  if (menu_state == 1){   // Can later add more criterias and more if cases
to add more languages

    // Get key value if pressed
    customKey = customKeypad.getKey();

    if (customKey) {

      // Get key value from function
      switch (counter) {
        case 1:
        ARDU_1();
        break;
        case 2:
        ARDU_2();
        break;
        case 3:
        ARDU_3();
        break;
      }

    }
  }

}
```

I kod 7 så börjar funktionen med att läsa av tillståndet av pulsgivaren. Efteråt kommer en if-sats som ser till att counter variabeln uppdateras korrekt, if-satsen efteråt i kod 7.1 ser till att counter värdet inte blir högre än 3 eller lägre än 1, eftersom det inte finns fler menyer än tre. Den sista if-satsen ser till att rätt språkmeny syns på displayen genom att kalla på funktioner som uppdaterar displayen, kod 10.

Kod 7: Pulsgivare.

```
void RotaryEncoder() {
  // Updates counter
  // if they are valid and have rotated a full indent

  static uint8_t old_AB = 3;  // Lookup table index
  static int8_t encval = 0;   // RotaryEncoder value
  static const int8_t enc_states[]  = {0,-1,1,0,1,0,0,-1,-1,0,0,1,0,1,-1,0};
// Lookup table
  old_AB <<=2;  // Remember previous state
  if (digitalRead(ENC_A)) old_AB |= 0x02; // Add current state of pin A
  if (digitalRead(ENC_B)) old_AB |= 0x01; // Add current state of pin B

  encval += enc_states[( old_AB & 0x0f )];

  // Update counter if RotaryEncoder has rotated a full indent, that is at
least 4 steps
  if( encval > 3 ) {        // Four steps forward
    int changevalue = 1;

    _lastIncReadTime = micros();
    counter = counter + changevalue;              // Update counter
    encval = 0;
  }
  else if( encval < -3 ) {        // Four steps backward
    int changevalue = -1;

    _lastDecReadTime = micros();
    counter = counter + changevalue;              // Update counter
    encval = 0;
  }

  // Limits
  if (counter < 1){
    counter = 3;
  }
  else if (counter > 3){
    counter = 1;
  }
```

Kod 7.1 Pulsgivare.

```
// Display change
  if (Last_counter != counter){

    if (menu_state == 1){
      if (counter == 1){
        ARDISP_1();
        Last_counter = counter;
      }
      else if (counter == 2){
        ARDISP_2();
        Last_counter = counter;
      }
      else if (counter == 3){
        ARDISP_3();
        Last_counter = counter;
      }

    }

  }
}
```

I kod 8 börjar funktionen med att läsa tillståndet på pulsgivar knappen, if-satsen som följer körs ifall tillståndet har ändrats och uppdaterar menu_state variablen. Sista if-satsen körs ifall menu_state har ändrats och kallar på funktioner som uppdaterar displayen med antingen huvudmenyn eller språkmenyn, kod 10 samt 11.

Kod 8: Pulsgivarknapp.

```
void EncoderButton() {

  // Read the state of the button
  buttonState = digitalRead(ENCODER_BUTTON);

  // Check if the button state has changed
  if (buttonState != lastButtonState) {

    if (buttonState == LOW) {
      if (menu_state == 0){
        menu_state = 1;
      }
      else if (menu_state == 1){
        menu_state = 0;
      }
    }

    // Update last button state
    lastButtonState = buttonState;

    // Debounce delay to avoid multiple readings during button press
    delay(10);
  }

  // Changing menue
  if (LastMenu_State != menu_state){

    if (menu_state == 0){
      MENU_DISP();
      LastMenu_State = menu_state;
    }
    else if (menu_state == 1){
      counter = 1;
      ARDISP_1();
      LastMenu_State = menu_state;
    }
  }
}
```

I kod 9 är de tre första värdena ur första Arduno språkfunktionen. I kod 6 blir den här funktionen kallad om en knapp blir tryckt, och när funktionen körs så skriver den ut koden som motsvarar knappens värde, som hålls av customKey variabeln. Ifall knapp 1 blir tryckt skrivs en if sats ut. I programmet finns tre funktioner som i kod 9, med nio värden var.

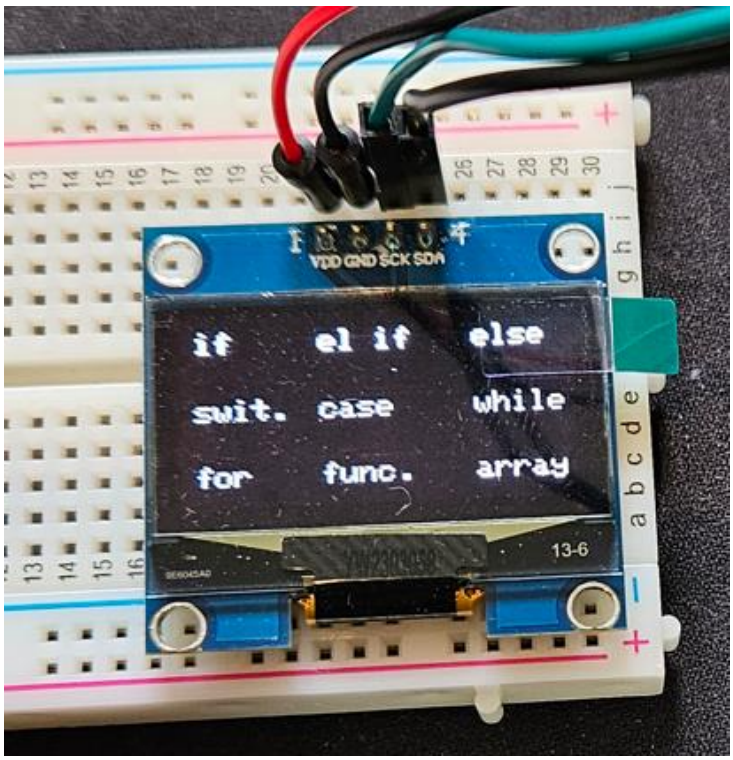
Kod 9: Exempel på funktionerna som ger tangenterna sina värden.

```
void ARDU_1(){

  if (customKey == '1'){                                          // if
    Keyboard.println("if (condition) {");
    Keyboard.println("//statement(s)");
  }
  if (customKey =='2'){                                          // else if
    Keyboard.println("else if (condition) {");
    Keyboard.println("//statement(s)");
  }
  if (customKey == '3'){
    Keyboard.println("else {");                                // else
    Keyboard.println("//statement(s)");
  }
```

I kod 10 är början på funktionen som körs när displayen uppdateras till första Arduino språkmenyn. Det finns tre funktioner som kod 10 i programmet och funktionen för att byta mellan dem ses i kod 7.1.

Kod 10: Exempel av språkmenyn för displayen.

```
void ARDISP_1(){

  display.clearDisplay();

  display.display();
  display.setCursor(5,5);
  display.print("if");

  display.display();
  display.setCursor(45,5);
  display.print("el if");

  display.display();
  display.setCursor(95,5);
  display.print("else");
```

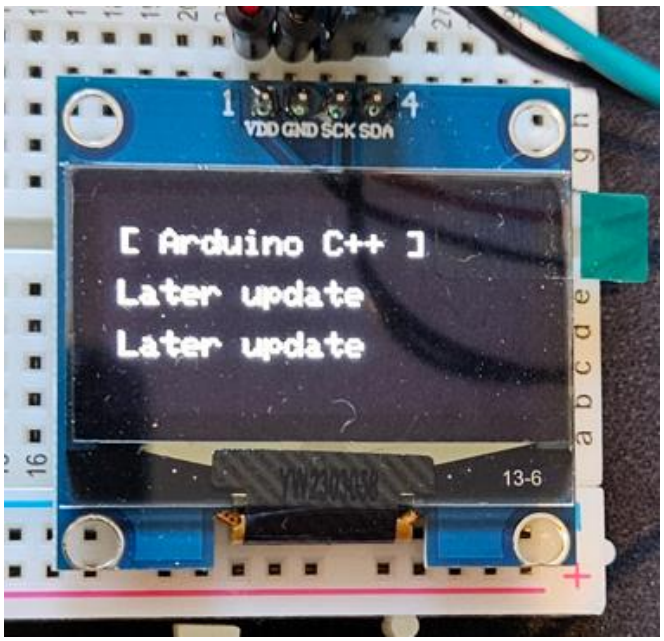


Figur 10: Arduino språkmeny 1.

I kod 11 är funktionen som visar huvudmenyn på displayen. Funktionen kallas när programmet startas, kod 5, samt när man går bakåt från en språkmeny, kod 8.

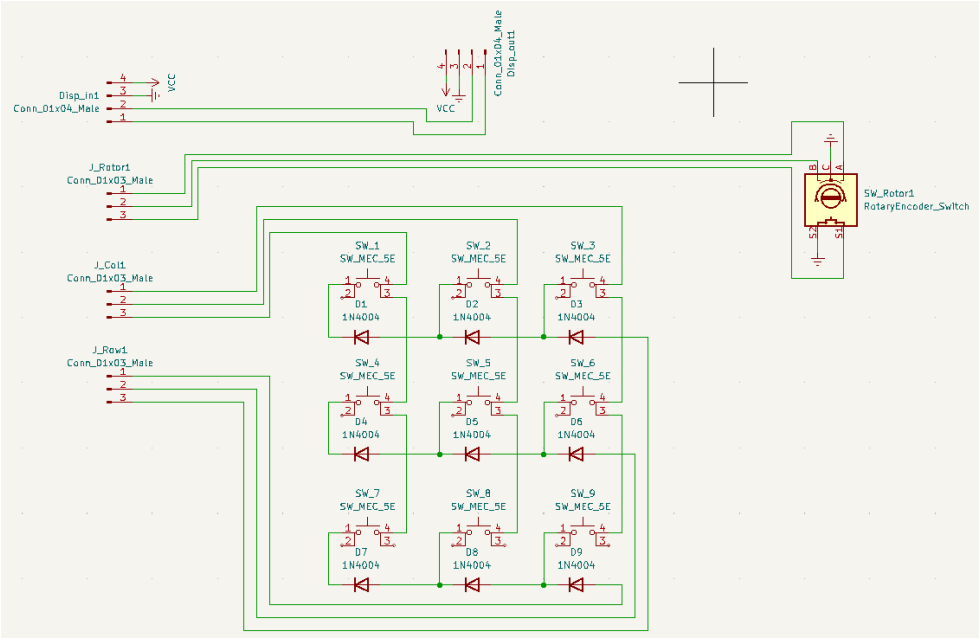
Kod 11: Huvudmenyfunktionen.

```
void MENU_DISP(){

  display.clearDisplay();

  display.display();
  display.setCursor(5,10);
  display.print("[ Arduino C++ ]");

  display.display();
  display.setCursor(5,25);
  display.print("Later update");

  display.display();
  display.setCursor(5,40);
  display.print("Later update");

  display.display();

}
```



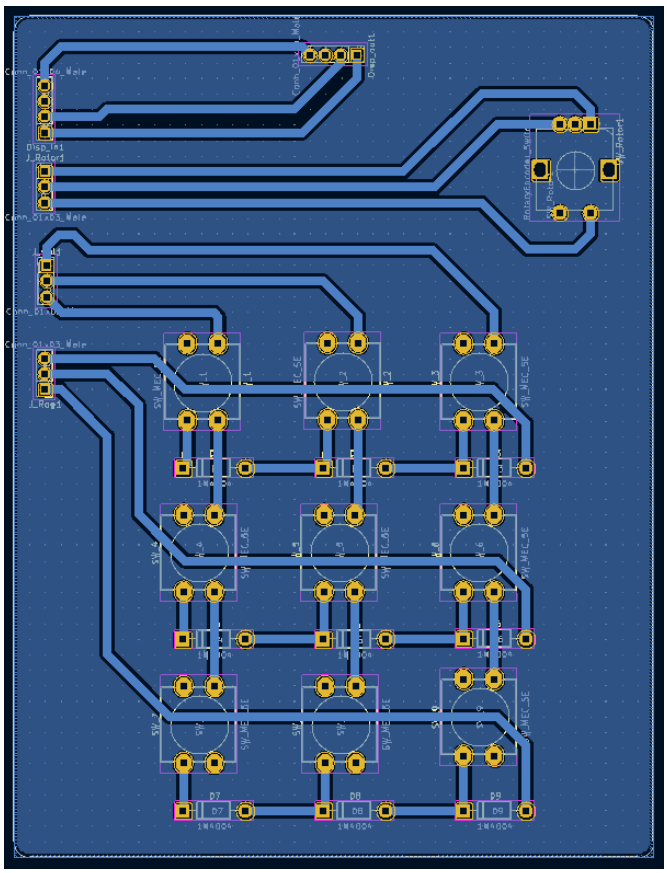
Figur 11: Huvudmenyn.

Kretskortet designades i Kicad och tillverkades i skolan. För första prototypen så sattes bara kontakter på sidan för att koppla kretskortet till utvecklingskortet för att göra det enklare. På vänstra sidan av schemat, som kan ses i figur 22, är alla kontakter som går till utvecklingskortet. Högst upp av kontakterna går till displayen, som i schemat representeras av en identisk kontakt som på sidan. Den andra kontakten går till pulsgivaren, och dom två sista kontakterna går till rad och kolumnraderna på knappmatrisen. I knappmatrisen valdes att sättas dioder som symboler eftersom då skulle det göra det enkelt under tillverkningen för hoppkablarna som sattes dit, istället för dioder.

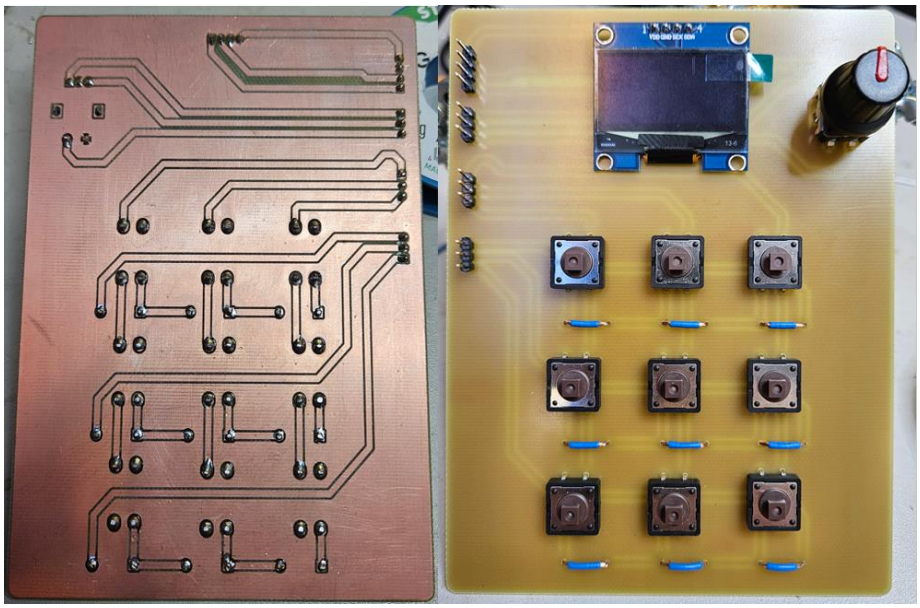


Figur 12: Kretskortschema.

Ett jordplan användes på kretskortet för att minimera arbetet under tillverkningen, så att så lite koppar som möjligt behövdes fräsas bort.



Figur 13: Kretskortdesign.



Figur 14: Kretskortet.

4 Diskussion

Prototypen fungerade inte helt som jag hade tänkt. Själva genvägstangentbordsfunktionen fungerade exakt som jag hade planerat, så att man kan byta funktioner för knapparna och skriva ut kod med ett knapp tryck, men så fort jag kopplar till displayen så slutar det att fungera.

Medan jag utvecklade koden och skrev ut värden på serial monitorn så kunde jag använda displayen, men då också ibland kunde det sluta att fungera och efter en stund började det fungera igen, utan att ändra någonting i koden.

När jag hade allt att fungera med att skriva ut värden i serial monitorn och displayen till kopplad som visade menyer som jag också kunde bläddra igenom, så bytte jag ut koden som skrev värden i serial monitorn, till kod som skrev ut som ett tangentbord på datorn så fick jag det aldrig att fungera med displayen till kopplad igen inom den utsatta tiden.

Men jag insåg också att jag inte behöver lösa det problemet, eftersom den här första prototypen redan har lärt mig vad jag behöver veta för att fortsätta med andra prototypen och implementera förbättringar där.

Den första förbättringen för nästa prototyp är en större display, 1,3” var för liten för att visa alla namn utan att förkorta dem på genvägarna för nio knappar. Jag började också tänka att det skulle kunna vara bra att ha möjlighet att se exempel på hur koden skulle kunna implementeras, och då behöver man en mycket större display. Jag tror också det är värt att tänka på om man skulle använda en touch skärm.

Den andra förbättringen är en helt annan mikrokontroller med mera minne. Endast basfunktionerna tog upp 68 % av minnet på mikrokontrollern som jag använde, och efter jag lade till genvägarna för Arduino C++, så användes redan 74 % av minnet och då hade jag bara satt dit det jag ansåg var mest nödvändigt. Ett större minne är nödvändigt om mer språk skall rymmas, samt om det skall finnas en möjlighet att implementera exempel på kod.

Den tredje förbättringen är att inte använda ett utvecklingskort alls, utan istället ha mikrokontrollen direkt på kretskortet, men det kan hända att det först implementeras på tredje prototypen.

5 Källförteckning

Arduino. (den 5 2 2018). *What is Arduino?* Hämtat från Arduino: https://www.arduino.cc/en/Guide/Introduction

Arduino. (u.d.). *Arduino Leonardo with Headers*. Hämtat från Arduino: https://store.arduino.cc/products/arduino-leonardo-with-headers

Axelson, J. (2015). *USB Complete : The Developer's Guide.* Madison: Lakeview Research.

Elprocus. (u.d.). *USB Protocol : Architecture, Working, Synchronisation, DataFormat & Its Applications.* Hämtat från Elprocus: https://www.elprocus.com/usb-protocol/

Espressif. (u.d.). *ESP32-S3-WROOM-1 ESP32-S3-WROOM-1U Datasheet.* Hämtat från Espressif: https://www.espressif.com/sites/default/files/documentation/esp32-s3-wroom-1_wroom-1u_datasheet_en.pdf

Espressif. (u.d.). *Modules*. Hämtat från Espressif: https://www.espressif.com/en/products/modules

Future Technology Devices International Ltd. (den 11 11 2009). *USB Data Packet Structure.* Hämtat från Ftdichip: https://ftdichip.com/wp-content/uploads/2020/08/TN_116_USB-Data-Structure.pdf

Geeksforgeeks. (den 5 5 2023). *Microcontroller and its Types*. Hämtat från Geeksforgeeks: https://www.geeksforgeeks.org/microcontroller-and-its-types/

Källkritik. (den 8 4 2024). *Arduino*. Hämtat från Wikipedia: https://en.wikipedia.org/wiki/Arduino

Källkritik. (den 9 4 2024). *USB*. Hämtat från Wikipedia: https://en.wikipedia.org/wiki/USB

Kicad. (u.d.). *About KiCad*. Hämtat från Kicad: https://www.kicad.org/about/kicad/

Lutkevich, B. (2019). *microcontroller (MCU)*. Hämtat från Techtarget: https://www.techtarget.com/iotagenda/definition/microcontroller

Processing. (u.d.). *Welcome to Processing!* Hämtat från Processing: https://processing.org/

RassieRaider. (2022). *Arduino Uno button matrix DCS-BIOS?* Hämtat från DCS: https://forum.dcs.world/topic/208714-arduino-uno-button-matrix-dcs-bios/

RS. (u.d.). *STMicroelectronics STM32 Nucleo-32 MCU Development Board NUCLEO-L031K6*. Hämtat från RS: https://za.rs-online.com/web/p/microcontroller-development-tools/9173781

Smoot, J. (u.d.). *The History of USB Standards from 1.0 to USB4*. Hämtat från Cuidevices: https://www.cuidevices.com/blog/the-history-of-usb-standards-from-1-to-usb4

Söderby, K., & De Feo, U. (den 17 1 2024). *Debugging with the Arduino IDE 2*. Hämtat från Arduino docs: https://docs.arduino.cc/software/ide-v2/tutorials/ide-v2-debugger/

ST. (den 21 9 2022). *STM32MCU basics*. Hämtat från Wiki.ST: https://wiki.st.com/stm32mcu/wiki/STM32StepByStep:STM32MCU_basics

Toschiba. (u.d.). *History of Microcontrollers*. Hämtat från Toshiba: https://toshiba.semicon-storage.com/ap-en/semiconductor/knowledge/e-learning/micro-intro/chapter2/history-microcontroller.html

Tutorialspoint. (u.d.). *Microcontrollers - Overview*. Hämtat från Tutorialspoint: https://www.tutorialspoint.com/microprocessor/microcontrollers_overview.htm

Wright Jr., J. R. (u.d.). *Introduction to the Microcontroller.* Hämtat från Millersville University: https://sites.millersville.edu/jwright/L1%20AENG%20467%20Intro%20to%20the%20Microcontroller%202022.pdf

Wu, J. (2022). *A Basic Guide to I2C.* Hämtat från Texas Instruments: https://www.ti.com/lit/an/sbaa565/sbaa565.pdf?ts=1710310852627&ref_url=https%253A%252F%252Fwww.google.com%252F

Bilaga: Kod