CHALMERS
UNIVERSITY OF TECHNOLOGY


# Konverterare från puls till ton för dialog-telefoner

Examensarbete i Elektroteknik, vid Institutionen för Elektroteknik

TEJBIR SINGH & EMIR BASIC

Institution för Elektroteknik
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2020

Bachelor's thesis 2020

# Konverterare från puls till ton för dialogtelefoner

Examensarbete i Elektroteknik, vid Institutionen för Elektroteknik

TEJBIR SINGH & EMIR BASIC


CHALMERS
UNIVERSITY OF TECHNOLOGY

Institution för Elektroteknik
Chalmers University of Technology
Gothenburg, Sweden 2020

**Konverterare från puls till ton för dialogtelefoner**
*TEJBIR SINGH*
*EMIR BASIC*



Företaget som arbetet utförs på: **Broccoli Engineering AB** Tillgängligheten 3, 417 01 Göteborg
Handledare på företaget: **Björn Bergholm**, chef för Broccoli Engineering AB
Handledare: **Fredrik Hellström**, Institutionen för Elektroteknik
Examinator: **Thomas Eriksson**, Institutionen för Elektroteknikt
Bachlor's Thesis 2020
Institution för Elektroteknik
*Elektroteknik*
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Chalmers Reproservice
Gothenburg, Sweden 2020

# Konverterare från puls till ton för dialogtelefoner

TEJBIR SINGH & EMIR BASIC
*Institutionen för Elektroteknik*
Chalmers University of Technology

## Sammanfattning

Idag fungerar både gamla och nya telefoner i det nuvarande telefonnätet. De gamla använder sig av pulsval och de nya använder sig av tonval, men inom en snar framtid kommer pulsvalsmetoden tas bort. Därför kommer de gamla telefonerna att inte fungera längre. I detta arbete har en prototyp tagits fram som kommer göra det möjligt att använda gamla dialogtelefoner i det nya systemet.

För att göra detta möjligt har en konverterare konstruerats. Konverteraren är en prototyp som konverterar pulser från en fingerskiva till *Dual Tone Multiple-Frequency*. Den är menad att användas som en extern resurs för att kunna föra ett samtal med ens nuvarande dialogtelefon. Detta görs genom att slå in ett telefonnumer på konverterarens fingerskiva och genom en Arduino skicka ut motsvarande ton till högtalaren och låta tonerna spelas upp i dialogtelefonens mikrofon.

Detta gjordes genom att testa och undersöka olika typer av dialogtelefoner, gamla telefonnätet och det nya telefonnätet. Samtidigt så studerades olika datablad och specifikationer för telefonnätet och de nyare telefonerna. Programmering gjordes även för att kunna göra om pulser till toner och kunna få ut de specifika tonerna.

För framtida arbete kan konverteraren utökas till en komplett adapter, där den kan vara direkt kopplad mellan dialogtelefonen och telefonnätet, med fler funktioner som till exempel en display, callback-funktioner och att kunna använda hela spektrumet av DTMF-toner (A, B, C, D, * och #).

Nyckelord: Pulval, tonval, Arduino, Adapter, konverterare.

## Abstract

Today, both old and new phones work in the current telephone network. The old ones use pulse selection and the new ones use tone selection, but in the near future the pulse selection method will be removed. Therefore, the old phones will no longer work. Our goal is to provide a solution for this problem, by developing a prototype that will make it possible to use old rotary dial phones in the new system.

To make this possible, a converter has been designed. The converter is a prototype that will convert pulses from a rotary dial to *Dual Tone Multiple-Frequency*. It is intended to be used as an external resource to be able to call with your current rotary dial phone. This is done by entering a telephone number on the converter's rotary dial and sending the corresponding outputs from the Arduino to the speaker and allowing the notes to be played in the rotary dial phones microphone.

This was done by testing and investigating different types of dialogue telephones, the old telephone network and the new telephone network. At the same time, various data sheets and specifications for the telephone network and the newer phones were studied. Programming was also done to be able to convert pulses to tones and to get the specific tones.

For future work, this converter can be extended to a complete adapter, where it can be directly connected between the rotary dial phones and the telephone network, as well as having more features such as a display, callback functions and being able to use the full range of DTMF tones (A, B, C, D, * and #).

Keywords: Pulse dialing, tone dialing, Arduino, adapter, converter.

# Förord

Detta examensarbete utfördes hos Broccoli engineering AB i Göteborg för institutionen inom Elektroteknik vid Chalmers Tekniska Högskola. Examensarbetet motsvarar 15Hp utav 180Hp för högskoleingengörer och är ett utav de sista momenten som utförs innan högskoleingenjör examen.

Vi skulle vilja tacka medarbetarna hos Broccoli engineering AB samt handledarna Björn Bergholm, Broccoli och Fredrik Hellström, Chalmers. Vi skulle även vilja ge ett stort tack till vår examinator Thomas Eriksson.

Tejbir Singh, Göteborg, Juni 2019
Emir Basic, Göteborg, Juni 2019

# Innehåll

# 1
# Introduktion

I denna del kommer projektets bakgrund, syfte, avgränsningar och mål samt frågeställningar presenteras.

## 1.1 Bakgrund

Målet med detta projekt är att skapa en konverterare, som gör om pulser till toner. Denna produkt är skapad för att gamla telefoner som använder sig utav pulser ska kunna återanvändas i dagens utvecklade telefonnät. Delmålen med detta arbetet är att förstå sig på tekniken, konstruktionen och utvecklingen av dialogtelefoner, för att skapa möjligheten att återanvända dessa i dagens telefonsystem som använder sig av Dual Tone Multiple-Frequency *(DTMF)*.

Fingerskivan är en amerikansk uppfinning som skapades 1896. Men det dröjde ända till 1920-talet för uppfinningen att komma till Sverige. Fingerskivan möjliggjorde automatiseringen av telefontrafiken.

Fingerskivan är konstruerad på ett sådant sätt att det finns 10 hål och varje hål motsvarar en siffra från 0-9. För att slå in en siffra stoppar man in fingret i hålet och vrider tills det tar stopp (ändläget) och sen släpper skivan så den återvänder till sin ursprungliga punkt m.h.a fjädring. Telefonen genererar då *n+1* elektriska impulser som skapas genom att två kontakttungor påverkar ett kugghjul, dessa pulser skickas vidare till telefonväxeln.

Examensarbetet utfördes på Broccoli Engineering AB. Broccoli Engineering AB är ett konsultföretag som bildades av en före detta chalmersstudent, Björn Bergholm år 1993. Företaget har vuxit och är idag ett stort företag inom elektronikutveckling. De specialiserar sig inom inbyggda system, design och konstruktion och har samarbetat med

flera olika företag.

## 1.2 Syfte

Syftet med arbetet är att skapa ett hjälpmedel som gör så att dialogtelefoner ska kunna återigen användas i dagens telefonnät. Då dialogtelefoner använder sig av pulser funkar de inte i dagens telefonisystem som använder sig av ljudtoner, så kallade Dual Tone Multiple-Frequency. Huvudfunktionen är att konvertera pulserna från dess egna fingerskiva till ljudtoner, skicka ut tonerna genom en högtalare som placeras på dialogtelefons mikrofon och därefter nå personen vars nummer man slagit in.

## 1.3 Avgränsningar

Följande avgränsningar är vad som gäller för projektet.

- Konverteraren kommer vara baserad på gamla telefoners fingerskiva.
- Konverteraren har ingen direktkontakt med telefonlinjen.
- Konverteraren är en första prototyp och kommer inte komprimeras och förenklas till en mindre storlek för att inte förlänga projektets längd.
- Materialkostnad och hållbarhetperspektiv kommer inte hållas i åtanke om det påverkar användning och funktion.

## 1.4 Precisering av mål

Nedan presenteras mål för examensarbetet.

- Genom en kretskonstruktion konvertera pulser skickade från en fingerskiva till DTMF.
- Slagna nummer spelas upp efter en kort fördröjning.
- Samtal ska görs m.h.a. konverteraren.
- Konverteraren ska vara konstruerad på ett sådant sätt att inga inställningar ska behöva göras.

# 2
# Teori

Nedan kommer det presenteras teknisk bakgrund och information om de begrepp och komponenter som används under projektets gång. Denna information utgör grunden till projektet.

## 2.1 Telefonnät i Sverige

På sent 1800-tal började de första telefonerna att långsamt introduceras i Sverige. I en lång tid fanns det bara fast telefoni. Efter 100 år började telefoni att utvecklas, då ny teknik hade kommit ut. [1].

### 2.1.1 Fast telefoni

Fast telefoni betyder att flera ledningar kopplades ihop i ett nätverk till telefonväxeln. I början var telefonledningarna enkeltrådiga i järntråd eller bronstråd. Därefter gick de över till dubbeltrådiga ledningar. Senare blev det koppartrådar inneslutna i kabel. I kablarna fanns det tusentals trådar som var isolerade från varandra med gummi. Kablarna från ledningen kopplades till abonnenterna och i samhällen så grävdes kablarna ner[2].

### 2.1.2 Mobiltelefoni

Mobiltelefoni kom vid 1950-talet och har använts sen dess. Med radioteknologi kan bärbara enheter vara i kontakt med varandra genom olika stationer och sedan kopplas vidare till det fasta telefonnätet[3].

### 2.1.3 IP-telefoni

Internetprotokoll-telefoni innebär att samtalet överförs med nätverk baserat på internetprotokollet. Informationen överförs i datapaket, alltså via internet. För att bevara säkerheten på samtalen så kan de kodas enligt standardmetoderna. Fördelen med IP-telefoni är att med

en stabil uppkoppling fungerar de lika bra som de traditionella metoderna, och samtidigt nästan gratis om användaren har internet. Nackdelen är att säkerheten inte är den bästa och att samtalskvaliteten kan vara dålig om för många använder internet och använder upp all bandbredd. IP-telefoni arbetar med principen "best effort", som betyder att all information ska behandlas lika och att meddelanden ska försöka levereras så snabbt som möjligt. Därför finns det ingen garanti att kvaliteten är bra eller att det kommer fram[4].

## 2.2 Telefoner

För att kommunicera med telefonväxeln om vilket nummer användaren försöker ringa så finns det två sätt. Det gamla sättet fungerade med hjälp av pulser medan det nya sättet fungerar med hjälp av toner. Idag finns det telefoner där båda sätten fungerar, men det är en tidsfråga innan pulsvalsmetoden försvinner.

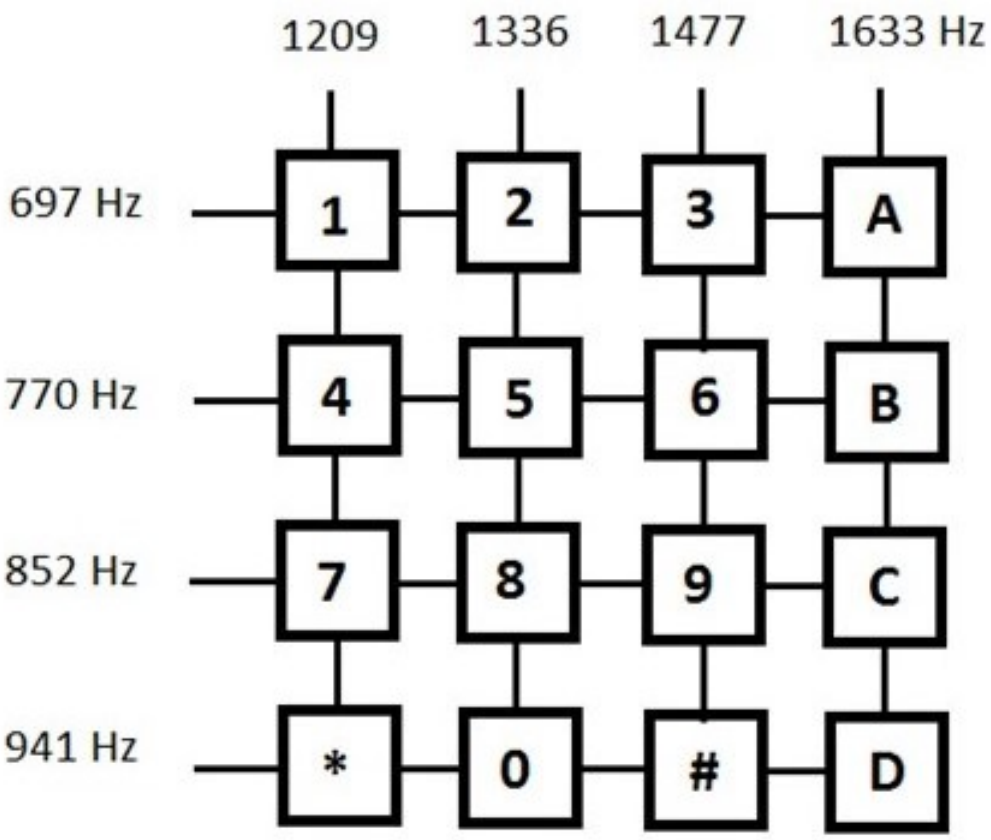
### 2.2.1 Pulsval

Gamla telefoner med fingerskiva är baserade på pulsval. För att skicka en siffra till teleledningen så snurras en fingerskiva med önskad siffra, då genereras *n+1* antal pulser som motsvarar den önskade siffran. På teleledningen är det en spänning och när fingerskivan snurrar tillbaka från ändläget är det en switch som bryter spänningen vilket genererar pulser, detta kallas *loop disconnect signaling*. På det sättet vet telefonväxeln vilken siffra som slagits[5].

### 2.2.2 DTMF - Tonval

DTMF står för *Dual Tone Multiple Frequency*, en teknik som används både inom telefoni och radio. Genom att kombinera två sinustoner med olika frekvens så genereras en siffra, bokstav eller tecken som telenätet kan uppfatta. Totalt 8 frekvenser används, 4 st inom den låga frekvensgruppen och 4st inom den höga frekvensgruppen. Ljudfrekvenserna i kombination bildar 16 par, där varje par motsvarar en ljudsignal, vilket kan ses i figur 2.1. Fördelen med tonval över pulsval är att det är ett effektivare och snabbare sätt att slå in nummer samt att kan göras semi-automatiskt med telefonväxlarna. Anledningen till

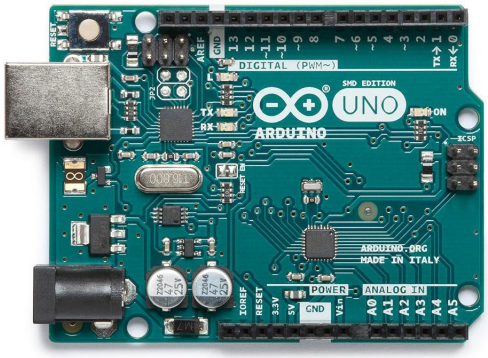
att det används två toner istället för en ton är att det ökar störtåligheten. Det gör även att samtal över längre distanser kan göras lättare med hjälp av relästationer, vilka tar emot information och sedan skickar vidare den och förstärkar signalen, vilket tar bort behovet av mellanoperatörer[6].




**Figur 2.1:** Frekvenskombinationerna för de 16 olika paren, där frekvenserna mellan 697 Hz till 941Hz motsvarar den låga frekvensgruppen och frekvenserna 1209Hz till 1633Hz motsvarar den höga frekvensgruppen

## 2.3 Arduino Uno

Arduino är en mikrokontroller som är baserad på öppen källkod. En mikrokontroller är en liten programmerbar dator, vilket betyder att den har en CPU, arbetsminne och programminne. Det finns även hjälpfunktioner och olika I/O-delar. En Arduino Uno består av en ATmega328P, och har 14 digitala I/O pinnar, 6 analoga input-pinnar, en 16 MHz kvartskristall, en Usb A till B koppling och ett uttag för ström. Den programmeras i sitt egna IDE, vilket är en enklare form av C. Fördelen med en Arduino är att den är billig och flexibel att jobba med, då det finns möjlighet att koppla upp olika komponenter för olika uppgifter (t.ex sensorer, räkna pulser etc.) samtidigt som data kan visas i realtid på dataskärmen. Som nämnt tidigare, är det baserat på öppen källkod, vilket innebär att det finns färdiga bibliotek och program att ladda ner[7].

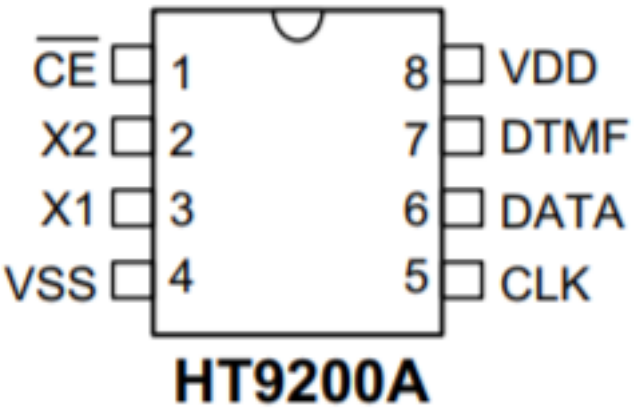



**Figur 2.2:** Bild över hur en Arduino Uno ser ut[7]

## 2.4 Komponenter

### 2.4.1 HT9200A - DTMF-generator

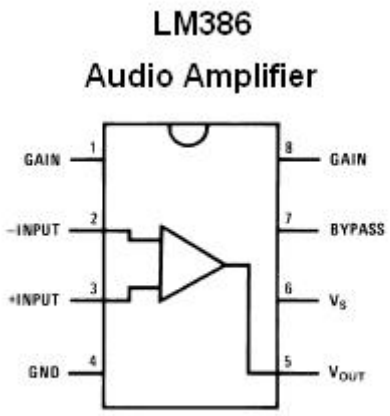
HT9200A[8] används för att genera DTMF-toner som sedan ska skickas till telefonlinjen. Den kan även generera 8 singeltoner. Den är gjord för att kunna kommunicera med en mikrokontroller, i detta fall en Arduino. Genom att skicka information i form av 5-bits data bestäms vilken ton som ska genereras. Matningsspänningen ligger mellan 2,5V till 5,5V. Högre spänning kan skada komponenten.




**Figur 2.3:** Pinlayouten för HT9200A-chippet[8]

### 2.4.2 Ljudförstärkare

För att förstärka ljudet användes LM386-N1[9], en kraftig förstärkare som är gjord för kretsar med låga spänningar. Rekommenderad spänning för LM386-N1 är mellan 4V och 12V. Förstärkarens standardinställning är att förstärka ljudet med 20 gånger och max förstärkning är upp till 200 gånger med en extern resistans eller kondensator mellan pinne 1 och 8, se figur 2.4.




**Figur 2.4:** Pinlayouten för LM386-N1-chippet[9]

### 2.4.3 Oscillator

Oscillatorer är elektriska komponenter som vibrerar i en konstant frekvens, se figur 2.5. När ström flödar igenom börjar de svänga. Eftersom svängningarna är konstanta, gör det att oscillatorer är användbara för noggrann tidsmätning.

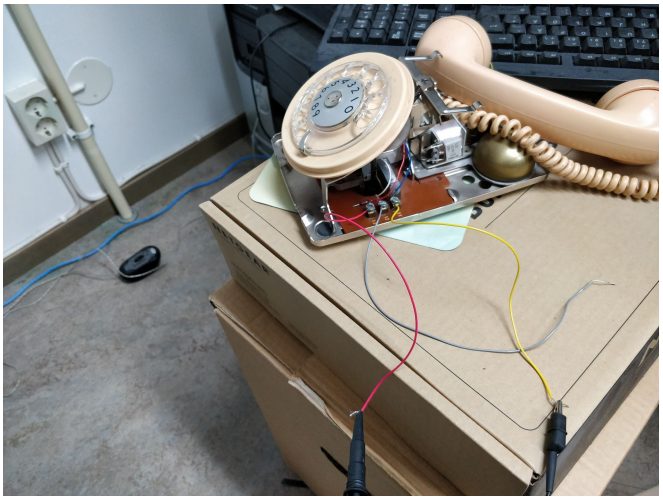
**Figur 2.5:** Figur över hur en keramisk oscillator ser ut.

# 3
# Metod

I denna del kommer projektets genomförande beskrivas.

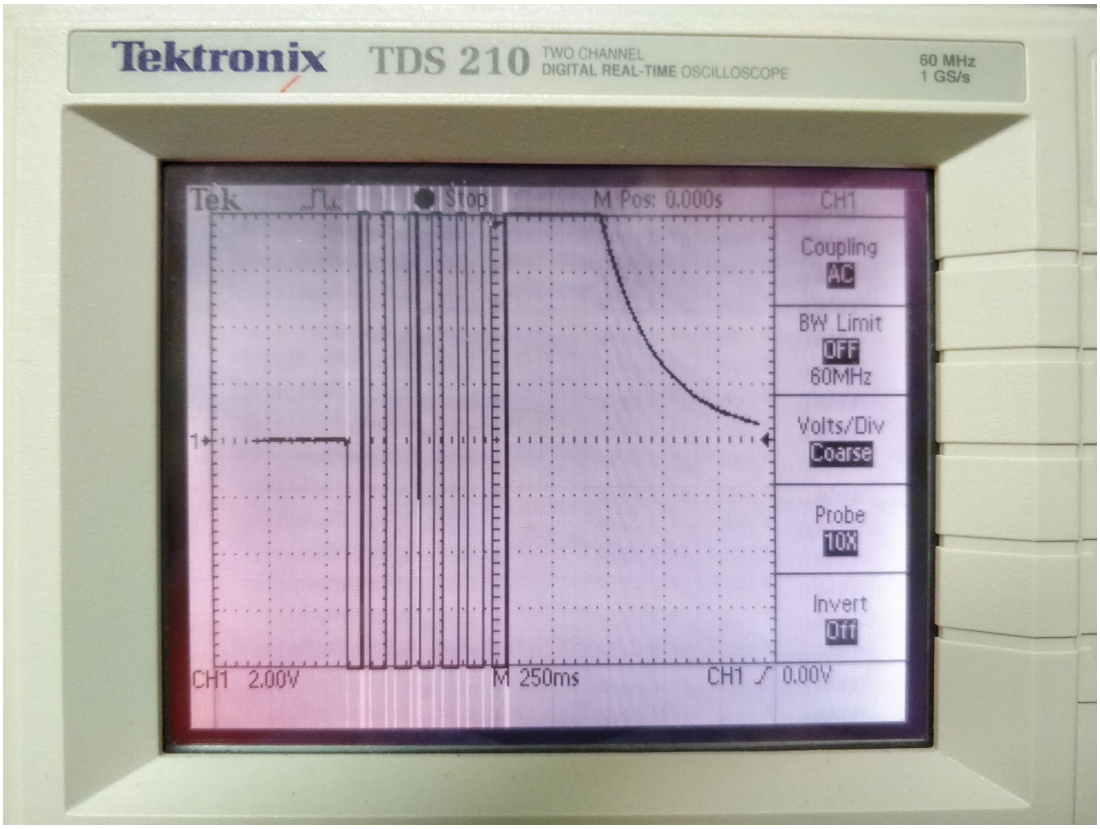
## 3.1 Sampling av pulser

Första steget var att förstå sig på hur telefonens pulsfunktion fungerar, så telefonen öppnades och utforskades. Pulserna skapas när sifferskivan roterar tillbaka från ändläget. Fingerskivan fungerar som en switch som bryter kopplingen och då genererar en puls. Beroende på vilken siffra som slås så bryter switchen *n+1* gånger. Sifferskivan kopplades upp till ett oscilloskop för tydligare bild och bekräftelse. Se figur 3.1 för att se hur kopplingen mellan fingerskivan och oscilloskopet genomfördes.



**Figur 3.1:** Fingerskivan på telefonen kopplades till ett oscilloskop för att mäta och analysera hur pulser skapas respektive pulslängden.

Med enkla justeringar på oscilloskopet så kunde man se hur många pulser som skapades vid specifik siffra, se figur 3.2.

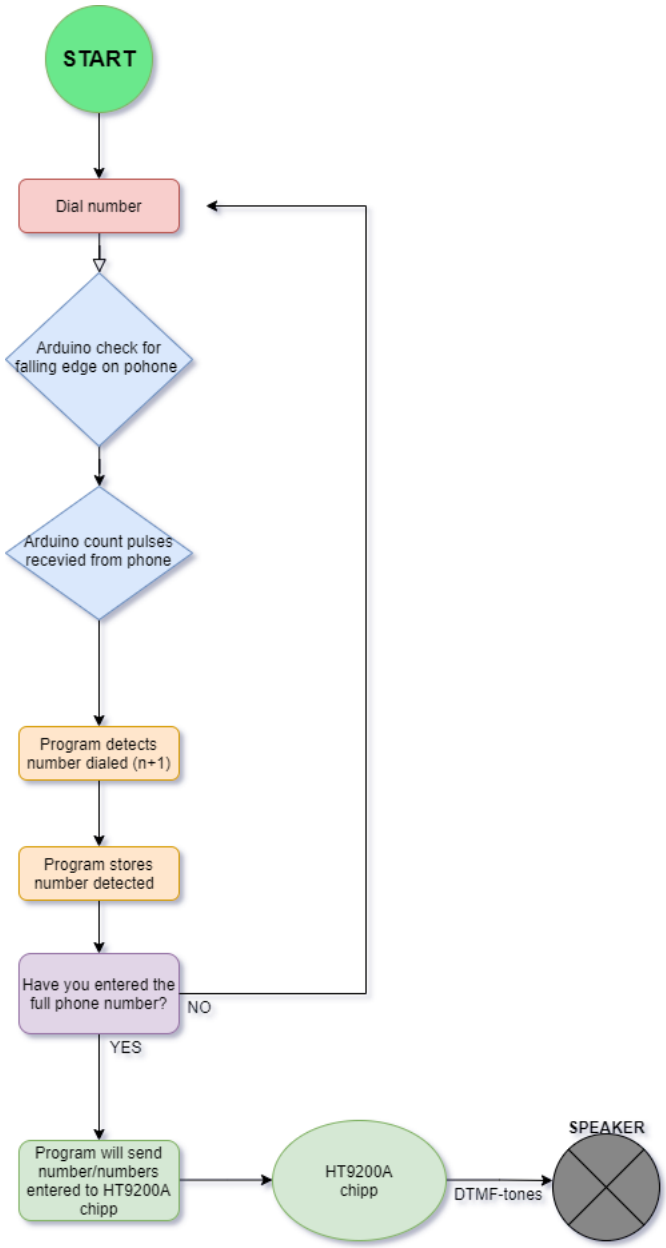



**Figur 3.2:** Bild på när siffran 6 slogs in i telefonen, då genererades 7 pulser som visas på oscilloskopet.

Som nämnt tidigare kan man se på figur 3.2 att pulserna genereras genom att switchen bryter kopplingen, alltså att spänningen går från *HIGH* till *LOW*. När sifferskivan snurrat tillbaka till dess ursprungsposition så blir spänningen *HIGH* igen. Mätningar gjordes även på när telefonen var i dess On-Hook position och Off-Hook position, detta ledde till att telefonen matas med ca 49 V när den är i dess On-Hook possition och mellan 7 och 10 V i dess Off-Hook position.

## 3.2 Flödesdiagram

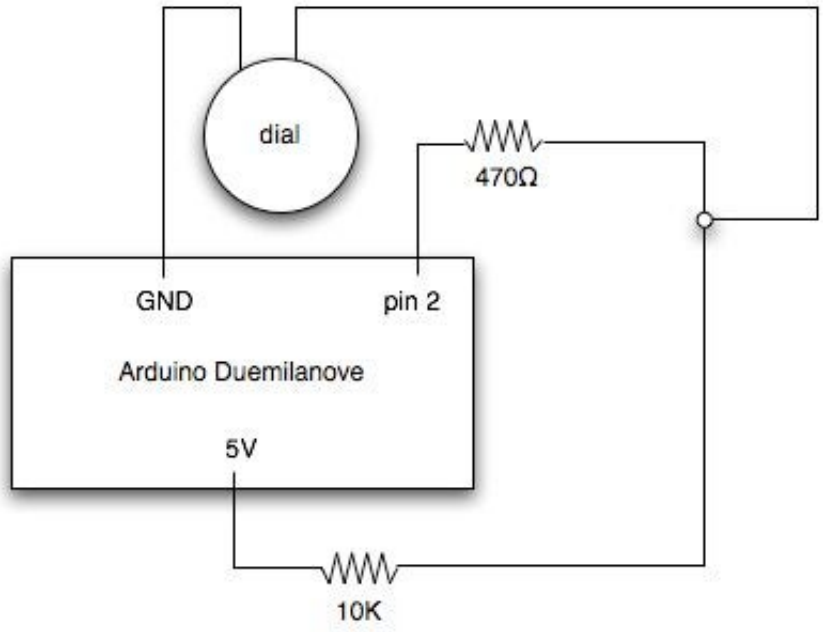
Ett flödesdiagram gjordes för den önskade processen för att kunna se steg för steg vad som behövdes göras. När telefonen inte används så ger telefonen ut en konstant signal på ca 49 V. När telefonluren lyfts sjunker signalen till ca 7-10 V, detta för att antingen ringa eller ta emot ett samtal. Denna stora spänningsskillnad indikerar för ledningen att telefonen är upptagen. Inga telefonsamtal kan då tas emot. När man ska ringa ett telefonsamtal så slår man in siffrorna och telefonen skapar pulser som motsvarar siffran. Pulserna görs genom att gå från hög till låg. Mellan varje plus så är det 10ms, detta mättes genom att pausa oscilloskop och använda dess tidsmätning funktion för att se skillnaden mellan varje puls. Samtidigt så sågs att en extra puls skapades oberoende av siffra, detta för att telefonen skapar n+1. Medans siffrorna slås så sparas de och när sista siffran har slagits så skickas alla siffror ut till HT9200A-chippet för att skapa tonerna. Därefter spelas tonerna på högtalaren, se figur 3.3.




**Figur 3.3:** Flödesschema över hur processen kommer gå till. Det kommer börja med att slå in ett number på konveterarer därefter detekteras om signalen går från hög till låg på arduinon, detta indikerar att en siffra är slagen. Sedan räknas hur många pulser som har slagits och utifrån det tolka vilken siffra som har slagits. Sedan sparas siffran som var salgen och stegen upprepas tills ingen siffra är detekterad, genom en timer. När alla siffrorna är slagna så skickas de till HT9200A-chippet, vars utgång kopplas till ljudförstärkaren som vår vidare till högtalaren.

## 3.3 Pulser till arduino

För att kunna detektera pulserna från fingerskivan till Arduinon, sattes 2 sladdar på baksidan av fingerskivan. De två sladdarna kopplades vidare till kopplingsplattan med följande krets, se figur 3.4. En pinne från skivan går till jord och den andra går parallellt till 2 resistorer som går till pinne 2 på Arduino och 5 V på kopplingsplattan.

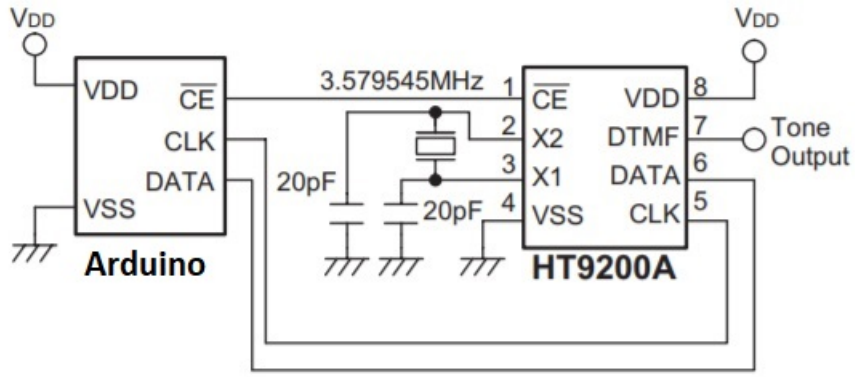



**Figur 3.4:** Telefonens skiva kopplades m.h.a två sladdar för att föra vidare signalen till Arduinon m.h.a resistorer. Pinne 2 används på Arduinon för att detektera när pulserna var slagna.

För att räkna pulserna med Arduinon så användes en timer, som konstant kollar om pinne 2 är låg eller hög. Genom att räkna hur många förändringar som skett kunde Arduinon veta hur många pulser som skickats. Om det inte skedde en förändring på mer än 100 ms, betyder det att fingerskivan är klar med att skicka pulser och siffran är slagen. Genom att kolla hur många pulser som kom in under den tiden kunde man se vilken siffra som var slagen. Siffrorna sparas sedan i en array och skickas vidare till DTMF:en. Se figur A.1 i bilagorna för fullständig kod.

## 3.4 DTMF-kretsen

DTMF-tonerna genereras av HT9200A-chippet. HT9200A-chippet tar emot data seriellt i samband med klockpulserna från Arduinon med hjälp av indata från telefonen. Chippet opererar på en 3.579545 MHz frekvens och matas med 5 V, se figur 3.5. För att aktivera chippet skickas en 0 till pinne $\overline{CE}$, se figur 3.9. Klockan reagerar på negativ flank för att ta emot data och spara bitarna som skickas in, minst signifikanta bit till mest signifikanta bit. Se figur 3.9. In till pinne 2 och 3 användes en keramisk oscillator och två kondensatorer på 20 pF. Se figur 3.5.




**Figur 3.5:** Översikt på hur Arduinon och HT9200A-chippet är kopplade med varandra. Enheterna är sammanlänkade med en keramisk oscillator (3.58 MHz) och två kondensatorer (20 $\rho F$)[8].

Data som skickas in till chippet är binärt. Beroende på vilken siffra som skickas in finns det motsvarande par av 2 frekvenser som bildar en ton, se tabell 3.1 för för detaljer.

**Tabell 3.1:** Tabell över hur HT9200A-chippet får indata, där första binära tal som kommer in är *most significant bit*, dvs D0 och sista är *least significant bit*, dvs D4. Beroende på vilken siffra som skickas kommer motsvarande 2 frekvenser ut på pinne 7. Om man ska stänga av HT9200A-chippet så skickas siffran 15, binär 1111 och då stängs HT9200A-chippet av.

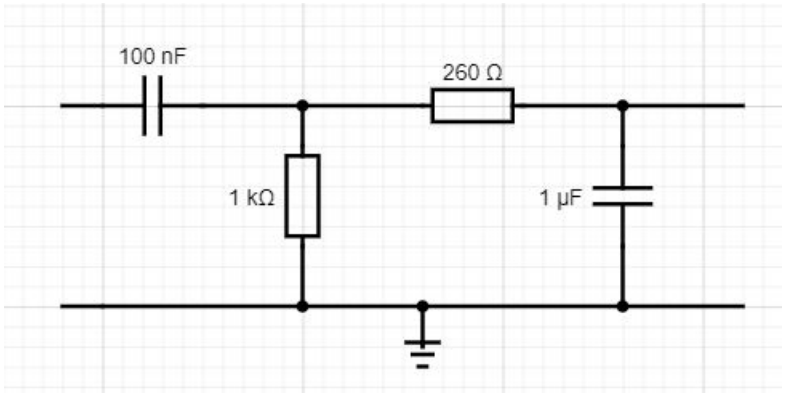
| Digit | D4 | D3 | D2 | D1 | D0 | Tone Output Frequency [Hz] |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 941+1336 |
| 1 | 0 | 0 | 0 | 0 | 1 | 697+1209 |
| 2 | 0 | 0 | 0 | 1 | 0 | 697+1336 |
| 3 | 0 | 0 | 0 | 1 | 1 | 697+1477 |
| 4 | 0 | 0 | 1 | 0 | 0 | 770+1209 |
| 5 | 0 | 0 | 1 | 0 | 1 | 770+1336 |
| 6 | 0 | 0 | 1 | 1 | 0 | 770+1477 |
| 7 | 0 | 0 | 1 | 1 | 1 | 852+1209 |
| 8 | 0 | 1 | 0 | 0 | 0 | 852+1336 |
| 9 | 0 | 1 | 0 | 0 | 1 | 852+1477 |
| *DTMF OFF* | 1 | 1 | 1 | 1 | 1 | — |

## 3.5 Bandpassfilter

För att inte få oönskade frekvenser konstruerades ett bandpassfilter. Filtret består av ett högpassfilter och ett lågpassfilter och har därför en övre och en undre gräns för vilka frekvenser ska gå igenom och vidare till högtalaren. De önskade frekvenserna för DTMF (exklusive tecken) är 1335 Hz till 697 Hz. Filtret som användes hade en undre gräns på 612 Hz och övre gräns på 1591 Hz. Bandpassfiltret som konstruerades består av resistorer och kondensatorer. Varje del består av en resistor och en kondensator. Ljudet kommer ut från HT900A-chippets DTMF-pinne (se figur 3.5) och går igenom ett högpassfilterstadie och därefter ett lågpassfilterstadie. Värdet på komponenterna bestämmer frekvensgränsen. Den önskade frekvensen för ett filter räknas ut med ekvationen:

$$f_{Hz} = \frac{1}{2\pi RC}$$

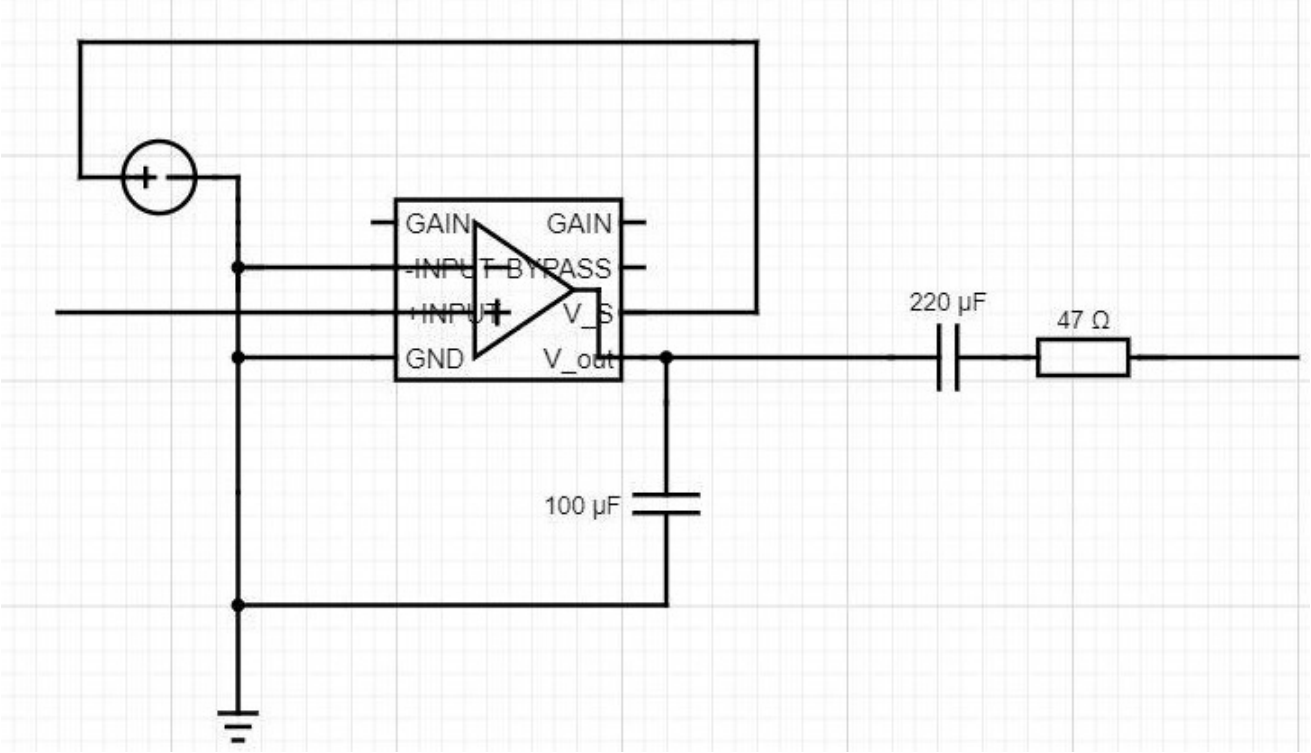
värdena på komponenterna som användes för högpassfiltret var $1k\Omega$ och $1\eta F$, detta motsvara 1591 Hz för den höga frekvensgränsen. För den låga frekvensgränsen användes komponenter med värdena $260\Omega$ och $1\mu F$, vilket motsvarar 612 Hz. Se figur 3.6 för översikt över bandpassfiltrets koppling.




**Figur 3.6:** Kretsschema över bandpassfiltret. Från höger sida kommer DTMF-signalen in från HT9200A-chippet, pinne 7. Signalen går först igenom högpassfilter och tar bort frekvenser över 1591 Hz sedan vidare igenom lågpassfilter, som tar bort frekvenser under 612 Hz, därefter lämnar signalen på vänstersida vidare till förstärkaren[10].

## 3.6 Förstärkarkrets

Eftersom ljudet ut från bandpassfiltret är lågt behövs en förstärkare konstrueras för att förstärka ljudet. För att konstruera förstärkaren användes LM386-N1[9], ett välkänt chipp som ofta används inom små elektriska projekt. För att göra ljudet tillräckligt hörbart konstruerades en krets, se figur 3.7. Kretsen består av chippet LM386-N1, en resistans på 47 Ω och två kondensatorer på 100 $\mu F$ respektive 220 $\mu F$. På pinne 3 (+INPUT) av LM386-N1-chippet kom ljudet in och på pinne 5 (Vout) kom det 20 ggr förstärkta ljudet ut. På pinne 6 (Vs) kördes 5 V in och pinne 2 och 4 (-INPUT och GND) kopplades till jord, se figur 3.7 för översikt.



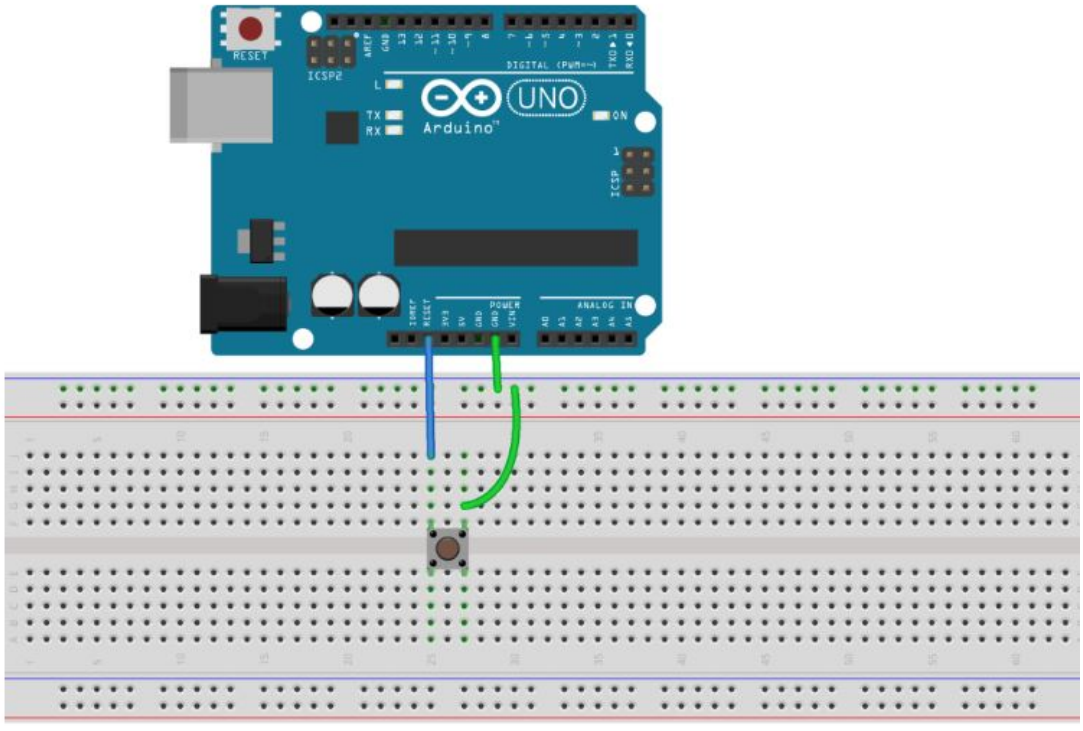

**Figur 3.7:** LM386-N1 är en välkänd förstärkare använd inom lågspänningsprojekt. LM386-N1 förstärker ljudet 20 ggr utan någon kondensator/resistor på pinne 1 och 8. Vid användning av pinne 1 och 8 kan man nå en maximal förstärkning på 200 ggr.

## 3.7 Uppkoppling till telefonnätet

Uppkoppling av samtalet gjordes på ett sådant sätt att man använder fingerskivan på konverteraren för att slå telefonnumret och HT9200A-chippet skickar ut motsvarande ljudton (2 frekvenser) av siffran som var slagen när hela telefonnumret är slaget. Ljudet går igenom band-passfiltret och sedan förstärks ljudet genom förstärkarkretsen ut till en högtalare. Högtalaren placeras intill telefonens mikrofon för att kunna skicka ut det slagna telefonnumret till telefonnätet.

## 3.8 Reset

Arduinos resetpinne kopplades till en tryckknapp, se figur 3.8. När knappen trycks så skickas en lågsignal till Arduinon och hela systemet startar om. Detta var konstruerat för att användas om fel nummer slås in.



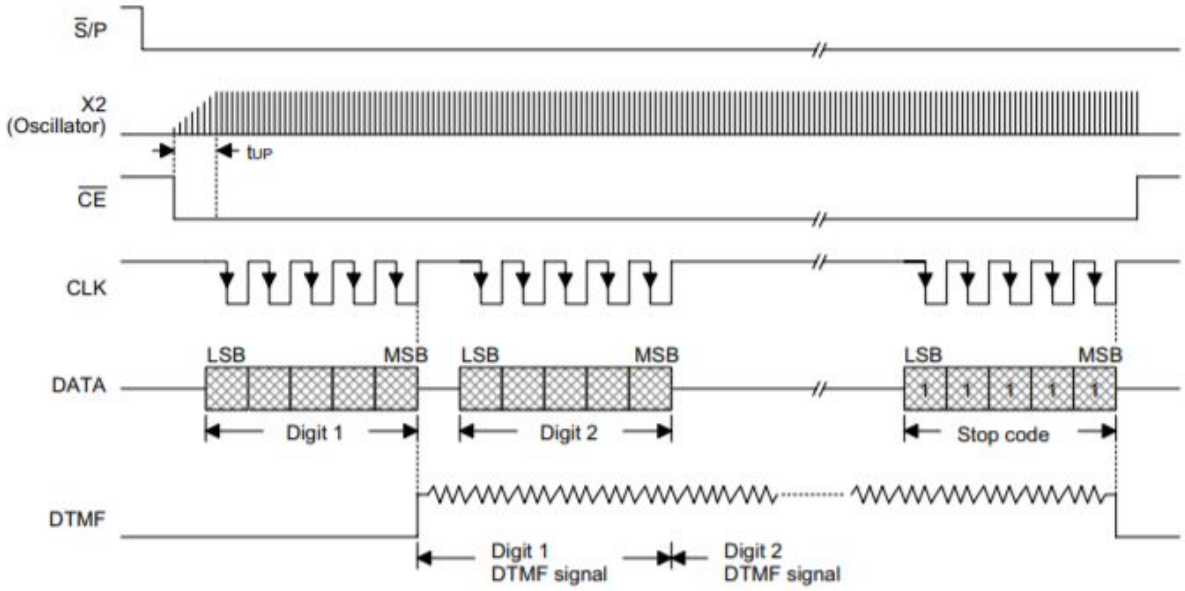

**Figur 3.8:** Koppling över hur resetknappen kopplades till tryckknappen, där en sladd går till Arduinos resetpinne och en annan till jord [11].

## 3.9 Programmering

I den här sektionen kommer koden för respektive del att förklaras. För fullständig kod, se appendix.

### 3.9.1 HT9200A-chipp

Eftersom HT9200A-chippet endast är kapabelt att skicka ut toner som motsvarar indata så krävdes en hel del programmering för att få igång denna process. Det som ska programmeras är enligt figur 3.9.




**Figur 3.9:** Figure över hur HT9200A-chippet exekverar kommandon. Den kan endast exekvera kommandon när *CE* sätts till *LOW*. När detta görs så tar det ca 10 ms för oscillatorn att börja oscillera, även kallad ramptid. Därefter så början klockan gå från *HIGH* till *LOW*. Om en siffra är slagen så omvandlas den till binär form och tas emot av HT9200A-chippet varje gång klockan går *LOW*. Efter klockan har gått *LOW* 5 gånger så spelas motsvarande *DTMF*ton upp. För att stänga av HT9200A-chippet skickas binär 11111 och då sätts *CE* till *HIGH* [8].

Eftersom chippet inte ska vara aktivt hela tiden så skapades en funktion som enbart ska aktivera chippet när indata ska skickas, se appendix A.4 för fullständig kod. Det som denna funktion gör är att aktivera chippet, därefter aktiveras klockan efter att oscillatorn har hunnit sättas på. Därefter väntar den in siffror. Efteråt inaktiveras chippet.

För att se hur siffrorna som slogs på telefonen togs emot och sparades, se Appendix B.

För att skicka de sparade siffrorna till HT9200A-chippet skapades en funktion Dialer. Det den funktionen gör är att identifiera vilka siffror som är sparade i telefonnumret, gör om dem till binär form och skickar dem vidare till nästa funktionen DTMF Out, innan de till sist går ut till chippet. Se appendix A.6 för fullständig kod.

För att skicka ut siffrorna till HT9200A-chippet användes funktionen DTMF Out. För att DTMF Out-funktionen ska skicka ut toner till HT9200A-chippet behöver tre argument anges, se appendix A.5. De tre argumenten är siffran som ska skickas in, hur lång tid tonen ska spelas och hur lång tid det ska ta för nästa ton att spelas. För att skicka ut siffrorna används en loop där byte skickar in ett femsiffrigt binärt tal som motsvarar siffran.. Medans detta skickas in så sätts CLOCK low, en bit DATA skickas in (LSB till MSB) och sedan sätts CLOCK hög igen. På detta sätt skickas alla siffror som var slagna in tills hela telefonnumret är klart, se appendix A.5 för fullständig kod.

### 3.9.2 Dialogtelefoner

För att sampla pulserna så kopplades 2 sladdar till baksidan på fingerskivan enligt figur 3.4. Fingerskivan har 10 hål som representerar varsin siffra (0 till 9), och för att slå en specifik siffra, så snurrar man hålet till ändläget. När man släpper fingerskivan så genereras *n+1* pulser. Varje gång en plus genereras så går signalen från HIGH till LOW sen tillbaka till HIGH. För att fånga detta så skapades en funktion för att kunna se vilken siffra som var slagen och kunna spara undan, se appendix B för fullständig kod.

För att läsa av signalen från fingerskivan användes pinne 2. En timer på 10 ms användes för att kunna vänta in så att signalen skulle hinna stabiliseras. Därefter kollas om sigalen är låg eller hög, detta för att kolla fingerskivans stadie. Om stadiet är lågt betyder det att pulser genereras och funktionen räknar hur många gånger den går från låg till hög, se appendix B.2.

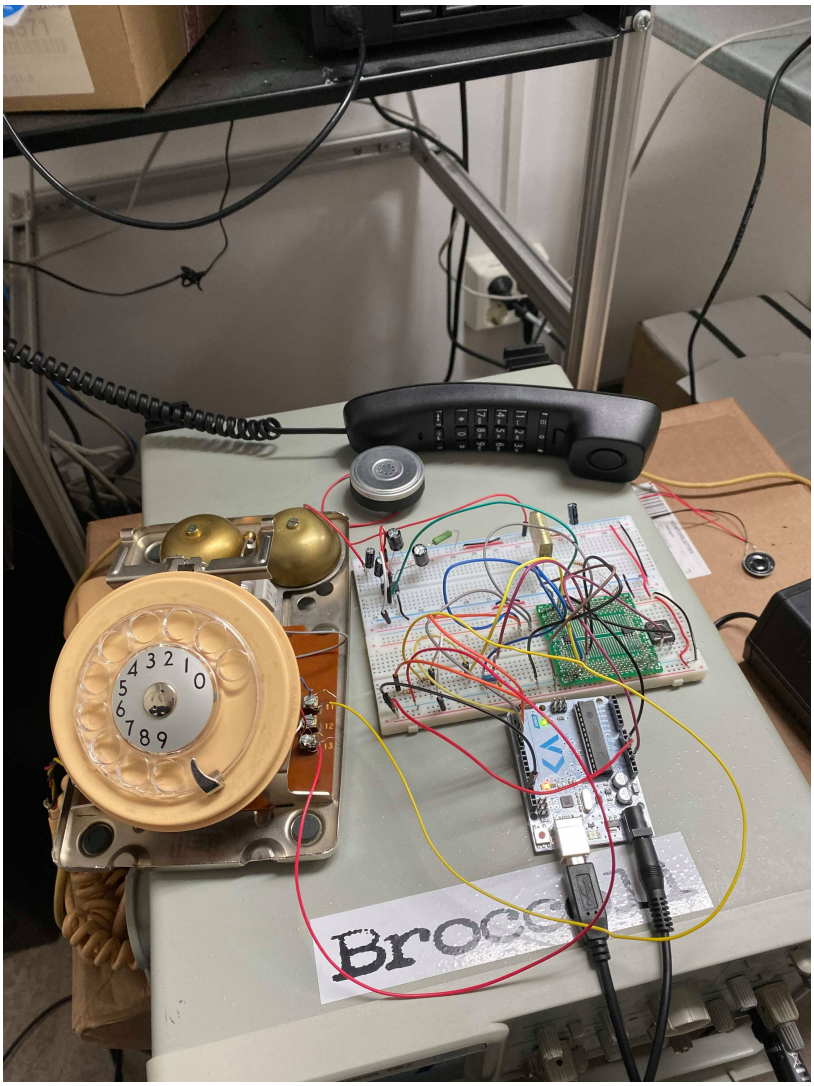
När fingerskivan har varit hög i mer än 100 ms utan att någon puls har kommit betyder det att fingerskivan har nått sitt ursprungs läge. När detta händer så kollar funktionen hur många pulser som kom in under den tiden och tar det minus ett för att identifiera siffran och sparar undan den, se appendix B.1.

För att skicka ut telefonnumret till Dialer-funktionen krävs det att ingen signalförändring sker på 5 sekunder. Detta indikerar att hela telefonnumret är slaget. Därefter skickas telefonnumret ut till Dialer-funktionen, se appendix B.3.

# 4

# Resultat

I denna del så beskrivs resultaten av de utförda testerna. Den färdiga konverteraren kan ses i figur 4.1.




**Figur 4.1:** Bild över hela systemet. Fingerskivan där telefonnumret slås in. De inslagna siffrorna plockas upp av Arduinon. Arduinon tolkar och skickar de inslagna siffrona vidare till HT9200A-chippet. Chippet går vidare till bandpassfilltret och till sist till förstärkarkretsen som är kopplad till högtalaren, som spelar upp tonerna.

**Tabell 4.1:** Tabell över hur varje frekvens procentuellt avviker utifrån standard, värdena är tagna från HT9200a-chippets datablad[8].

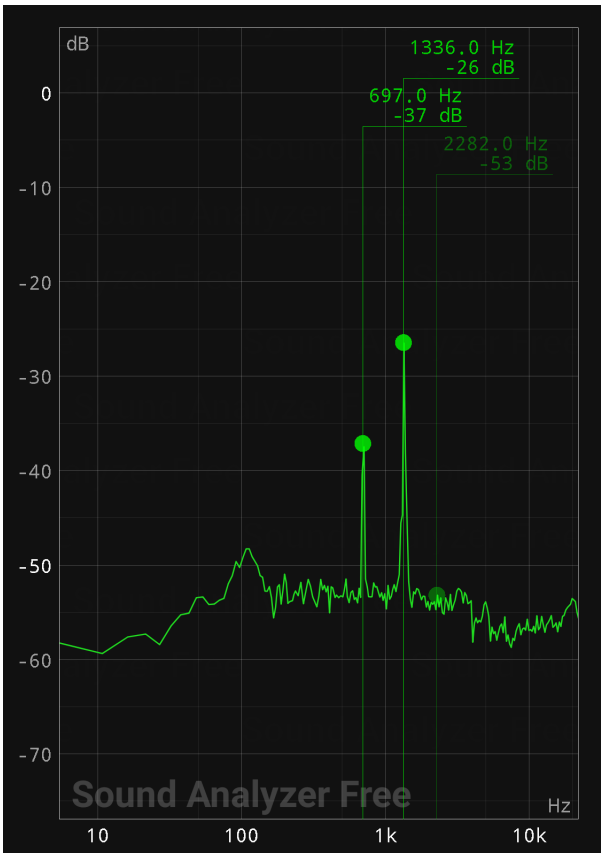
| Utgående frekvens (Hz) | | Avvikelse |
|---|---|---|
| Specificerad | Verklig | % |
| 697 | 699 | +0.29% |
| 770 | 766 | +0.52% |
| 852 | 847 | +0.59% |
| 941 | 948 | +0.74% |
| 1209 | 1215 | +0.50% |
| 1336 | 1332 | +0.30% |
| 1477 | 1472 | +0.34 |

## 4.1 DTMF

Resultaten som uppnåddes för Dual Tone Multiple-Frequency var att rätt ton erhölls när respektive siffra slogs. Tonen erhålls genom att lägga samman 2 frekvenser, se tabell 4.1. Dock så har varje frekvens en avvikelse som gör att DTMF-tonerna inte är exakta, se tabell 4.1. Detta skapade dock inga problem, då gränsen för låga och höga frekvensavvikelser ligger inom +/- 1.5% [12]. Så det uppstod inga problem för telefonnätet att identifiera vilken siffra som var slagen.

Detta testades genom att spela in tonerna som kom ut från konverteraren och jämföra dem mot en DTMF generator online för att höra om tonerna låter likadant. Detta test upprepades flera gånger för att försäkra att tonerna verkligen var densamma.

Ännu ett test som gjordes var att med hjälp av frekvensanalyseringsprogram fånga upp tonerna för att se om de rätta frekvenserna är uppnådda. Se bild 4.2 för resultat när siffran två var slagen.




**Figur 4.2:** Bild på frekvensupptagningen från högtalaren då siffran två är slagen. Siffrorna har olika toner och därför olika frekvenser.

## 4.2 Uppkoppling till telefonnätetet

Uppkoppling gjordes genom att använda dialogtelefonens mikrofon. Numret slogs genom fingerskivan på konverteraren. Konverteraren omvandlade pulserna till toner som spelades upp på högtalaren. Högtalaren placerades på mikrofonen till telefonen som skulle användas till samtalet. Telefonledningen kunde uppfatta vilket nummer som var slaget, och kopplade då vidare samtalet.

# 5
# Slutsats och diskussion

## 5.1 Slutprodukt och framtiden

I det här projektet har vi konstruerat en konverterare, som gör om pulser från en fingerskiva till *Dual Tone Multiple-Frequency*, som används inom modern telekommunikation. Konverterarens mål är att man ska kunna återanvända gamla dialogtelefoner i det moderna telefonnätet. Den fungerar genom att slå in telefonnumret man vill nå genom fingerskivan och gör om den till toner, sedan placeras högtalaren på dialogtelefonens mikrofon och ett samtal rings. När tonerna spelas upp i mikrofonen plockas de upp av teleledningen som för vidare samtalet.

Vid projektets början var målet att skapa en fullt fungerande adapter, där det skulle gå att koppla dialogtelefonens jack till adaptern och sedan vidare till telefonnätet. Adaptern skulle omvandla pulser från dialogtelefonen till toner, kunna ta emot/föra ett samtal och kunna överföra rösterna från varsin ände. Men detta fick begränsas på grund av tidsbrist till en puls-till-tonkonverterare, men detta kan göras i ett framtida projekt. Konverteraren är konstruerad för att konvertera pulser till motsvarande toner och sedan förstärka och spela upp toner via en högtalare till dialogtelefonens mikrofon. Pulsdetekteringen fungerar på samma sätt som den hade gjort för den menade adaptern. Detta har uppnåtts genom att Arduinon detekterar höga och låga tillstånd från fingerskivan och på så sätt räkna pulser. Baserat på pulserna räknas det ut vilka siffror som slagits, och siffrorna sparas i en sträng. Strängen är hela det slagna numret som ska ringas. Telefonnumret görs om till binär form och skickas vidare till DTMF-chippet, som genererar tonerna för respektive siffra. För att minimera störningar och få en så ren ton som möjligt, skickas tonerna genom ett bandpassfilter, som enbart tillåter specifika frekvenser att gå igenom. Då DTMF-toner går

från 697 Hz till 1633 Hz, blev gränserna 612 Hz och 1591 Hz. Exakta gränser uppnåddes ej på grund av brist på specifika värden på kondensatorer och resistorer. När tonerna gått genom filtret, skickades tonerna till förstärkaren, där tonerna förstärktes 20 gånger. Sedan går tonerna till högtalaren och därefter vidare till dialogtelefonens mikrofon.

Projektet är en temporär lösning för att kunna återanvända dialogtelefoner men då allt fler övergår till IP-telefoni, kan framtida utveckling vara att konstruera en adapter för det mediumet. Det finns redan idag ganska många adaptrar som behandlar det problemet, men ett mål kan vara att förenkla och minimera dem, då de idag är kanske klumpiga och komplicerade.

## 5.2 Fortsatt utveckling

Exempel på sätt som produkten kan förbättras på och funktioner som kan tilläggas;

- Koppla en LCD som visar vilka nummer som är slagna.
- Kunna spara nummer, ”speeddial” etc.
- Kunna slå in A, B, C. D, * och # genom långvariga tryck.
- Se senast uppringda nummer.

# Litteraturförteckning


[1] "Telefonnätets historia." https://www.tekniskamuseet.se/lar-dig-mer/100-innovationer/telefonen/, 2019.

[2] "Fast telefoni." https://web.archive.org/web/20160325145112/http://www.tkhv.se/artikel.pdf, 2016.

[3] "Mobiltelefoni." https://www.soluno.se/ordlista/mobiltelefoni/.

[4] "Ip-telefoni." https://www.cellip.com/vad-ar-ip-telefoni/.

[5] "Pulsval." https://en.wikipedia.org/wiki/Pulse_dialing, 2013.

[6] "Tonval." https://en.wikipedia.org/wiki/Dual-tone_multi-frequency_signaling, 2019.

[7] "Arduino." https://www.arduino.cc/en/Guide/Introduction.

[8] "Dtmf generators." http://www.farnell.com/datasheets/79214.pdf, 2019.

[9] "Lm386-n1." http://www.ti.com/lit/ds/symlink/lm386.pdf, 2017.

[10] "Bandpass filter." https://www.electronics-tutorials.ws/filter/filter_4.html.

[11] "Resetbutton." https://www.programmingelectronics.com/how-to-use-an-external-reset-button-with-arduino/, 2019.

[12] "Analog telephony compliance requirements overview." https://www.hermonlabs.com/Products/innerData/pdf/Analog%20Telephony%20Overview.pdf.

[13] "Arduino and ht9200a." https://forum.arduino.cc/index.php/topic,14687.0.html?fbclid=IwAR1XucSzUS5Syk5gsomqC4A1zU2YpOiuLE32QiTvpHgHE7m6_A2KzkW8P5w, 2009.

# A

# Appendix 1

```
/*
                                                 __________
      Chip On is low, Chip Off is high   CE-|1         8|- VDD   2.5V -5.5V
      Oscillator 3.59MHZ                OSC-|2 HT9200A 7|- DTMF  DTMF Signal Output 0,45V-0,75V
      Oscillator 3.59MHZ                OSC-|3         6|- DATA  Bits D0,D1,D2,D3,D4 bits
      GND                               VSS-|4_________5|- CLK   Data synchronous clock
*/

#define DTMF_OFF  24  // SB 31 but next cmd will be lost - a not used cmnd# works fine
#define STAR      11
#define POUND     12
#define A_KEY     13
#define B_KEY     14
#define C_KEY     15
#define D_KEY      0

// pure frequencies that can be made by the chip . . .
#define HZ_697    16
#define HZ_770    17
#define HZ_852    18
#define HZ_941    19
#define HZ_1209   20
#define HZ_1336   21
#define HZ_1477   22
#define HZ_1633   23
```

**Figur A.1:** kod för alla frekvens frekvensdefinitioner.

```
// PINS
#define IN          2                    // Data in
#define DATA_PIN    5                   // Data (serial)
#define CLOCK_PIN   6                  // Clock (serial)
#define CE_PIN      4                 // Chip Enable pin

int reading = digitalRead(IN);
int needToPrint = 0;
int SendDialer =0;
int count=0;
int lastState = LOW;
int trueState = LOW;
long lastStateChangeTime = 0;
int cleared = 0;
int n;
static int len= 0;
char PhoneNum[16];
// constants

int dialHasFinishedRotatingAfterMs = 100;
int debounceDelay = 10;
int TimeOut = 5000;
```

**Figur A.2:** Kod för alla in/ut-portar samt variabler och timers.

```
void setup() {
 Serial.begin(9600);
 pinMode(IN, INPUT);
 pinMode(DATA_PIN, OUTPUT);
 pinMode(CLOCK_PIN, OUTPUT);
 pinMode(CE_PIN, OUTPUT);
  Serial.print('\n');
  Serial.print("Enter phone number:");
  Serial.print('\n');
  Init_HT9200();                        // init the chip
 // ROTARY_PHONE();                      // Enter number thow phone


}

void loop() {

    ROTARY_PHONE();                     //  Enter number thow phone
}
```

**Figur A.3:** Kod för setup funktionen samt void loop, där in/ut-portar sattes och meddelanden som ska visas i serial monitor.

```
void Init_HT9200 (){                    // Activas chip
 digitalWrite(CE_PIN, HIGH);            // start with chip disabled (else you go nuts)
 digitalWrite(CLOCK_PIN, HIGH);         // start with clock pin high
 digitalWrite(CE_PIN, LOW);             // now enable the chip
 delay(10);                             // delay 10ms to ramp up the ocillator
 DTMF_Out (DTMF_OFF,1,0);               // turn off ant tone from previous run
}
```

**Figur A.4:** Kod för hur HT9200A-chippet aktiveras.

```
void DTMF_Out (byte digit, long duration, long pause){
 if (digit == 0) digit = 10;           // take care of 0 here
 for (byte i=0; i<5; i++){
   digitalWrite(CLOCK_PIN, HIGH);      // clock high while setting data

   digitalWrite(DATA_PIN, bitRead(digit,i)); // set data LSB→MSB
   delayMicroseconds(5);               // 1/2 of 100K Clock speed
   digitalWrite(CLOCK_PIN, LOW);       // clock low to latch data
   delayMicroseconds(5);               // 1/2 of 100K Clock speed
 }
 delay(700);                       // how long tone will play   duration!
 if (pause != 0){                        // tone sounds continuously if zero
   for (byte i=0; i<5; i++){           // same as above
     digitalWrite(CLOCK_PIN, HIGH);
     digitalWrite(DATA_PIN, bitRead(DTMF_OFF,i));
     delayMicroseconds(5);
     digitalWrite(CLOCK_PIN, LOW);
     delayMicroseconds(5);
   }
   delay(150);                         // how long pause between tones   pause!
 }
}
```

**Figur A.5:** Kod för hur siffror skickas till HT9200A-chippet.

```
void Dialer(){

  // cycles through tones to numbers
  Serial.print("Length of Phone number:");
  Serial.print(len);
  int char_len = len;
 for (byte i=0; i<char_len; i++){
     DTMF_Out(PhoneNum[i]-'0',500,100);  // 1/2 sec tome with 1/10 pause
    }
}
```

**Figur A.6:** Kod för hur siffror skickas till DTMF OUT funktionen.

# B
# Appendix 2

```
void ROTARY_PHONE()
{
int reading = digitalRead(IN);

 if ((millis() - lastStateChangeTime) > dialHasFinishedRotatingAfterMs) {
// the dial isn't being dialed, or has just finished being dialed.
    if (needToPrint) {
// if it's only just finished being dialed, we need to send the number down the serial
// line and reset the count. We mod the count by 10 because '0' will send 10 pulses.
       Serial.print(count-1 % 10, DEC);
       PhoneNum[len] = (char)(count-1);
       len++;
       needToPrint = 0;
       count = 0;
                }
 }
 if (reading != lastState) {
 lastStateChangeTime = millis();
 }
```

**Figur B.1:** Kod för hur de slagna siffrorna från fingerskivan sparas.

```
 if ((millis() - lastStateChangeTime) > debounceDelay) {
// debounce - this happens once it's stablized

    if (reading ≠ trueState) {
// this means that the switch has either just gone from closed→open or vice versa.
        trueState = reading;
        if (trueState == HIGH) {
// increment the count of pulses if it's gone high.
            count↔;
            needToPrint = 1; // we'll need to print this number (once the dial has finished rotating)
            }
       }
 }
lastState = reading;
```

**Figur B.2:** Kod för hur siffror identifieras när de slås in genom fingerskivan.

```
if (((millis() - lastStateChangeTime) > TimeOut)&& (n == 1)) {
  Serial.println('\n');
  Serial.println("Entering Dialer:..");
  for (int q = 0; q < 1; q++){
  Dialer();
  n=1;
   }
 }
}
```

**Figur B.3:** Kod för hur inslagna telefonnumret skickas till Dialer funktionen.