



Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

Pitch Detection in the Time Domain: Designing an Automatic Guitar Tuner

by

Thorbjørn Ravnaas Kristiansen

Supervisor: Dr Stephen Mosey

Second Marker: Gareth Jones

Project submitted as part of the requirements for the

award of

BEng (Hons) Computer Systems and Electronics

May 2019

DECLARATION

I, Thorbjørn Ravnaas Kristiansen declare that I am the sole author of this Project; that all references cited have been consulted; that I have conducted all work of which this is a record, and that the finished work lies within the prescribed word limits.

This work has not previously been accepted as part of any other degree submission.

Signed:

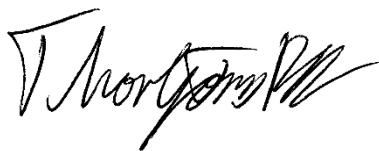
A handwritten signature in black ink, appearing to read 'T. Thorbjørn Kristiansen', with a stylized, cursive script.

Thorbjørn Ravnaas Kristiansen

FORM OF CONSENT

I, Thorbjørn Ravnaas Kristiansen hereby consent that my Project, submitted in candidature for the BEng Computer Systems and Electronics degree, if successful, may be made available for inter-library loan or photocopying (subject to the law of copyright), and that the title and abstract may be made available to outside organisations.

Signed:

A handwritten signature in black ink, appearing to read 'T. Thorbjørn Kristiansen', with a stylized, cursive script.

Thorbjørn Ravnaas Kristiansen

Abstract

A system for tuning guitar strings utilising pitch detection in the time domain was proposed. A stepper motor with a custom coupling was then implemented to turn the machine heads of the guitar automatically, depending on the current string frequency. A state-variable filter was used to acquire the fundamental frequency of the guitar string, before the signal was amplified and digitised using a 2-stage Schmitt-trigger. The user interface consisted of an LCD and 2 pushbuttons with which the user could cycle through the strings and start/stop the motor. The motor was driven by purpose-built stepper motor controllers and drivers.

Extracting the fundamental frequency using this pitch-detection technique proved successful. A drawback was the component count required for such a filter coupled with the low level of flexibility; the tuner could only tune guitar strings to a specific frequency, one string at a time. Accuracy problems were encountered when a prototype was built, mainly due to fast string vibration decay. A better solution to the problem would be techniques incorporating DSP, as this would reduce component count and offer a higher degree of flexibility.

Table of Contents

Abstract	2
Acknowledgements	8
1.0 Introduction	9
2.0 Background Research.....	10
2.1 Literature Review.....	10
2.2 The Limits of Human Hearing	11
2.3 Music Theory Glossary	12
2.5 Electric Guitar Fundamentals.....	13
2.5.1 Structural Components.....	13
2.5.2 Electronic Components	14
2.5.3 Strings	15
2.6 Pitch Detection.....	16
2.6.1 Frequency Domain	16
2.6.2 Time Domain	17
2.6.3 Autocorrelation	18
2.6.4 F/V Conversion	18
2.7 Filtering.....	19
2.7.1 Passive Filters.....	19
2.7.2 Active Filters.....	19
2.8 Existing Efforts and Designs.....	20
2.8.1 “The Mechanix” Guitar Tuner	20
2.8.2 The Roadie Tuner.....	20
2.8.3 Frequency to Voltage Conversion.....	21
2.9 Stepper Motors.....	22
2.9.1 Types of Stepper Motors.....	22
2.9.2 Unipolar Versus Bipolar Stepper Motors.....	24
2.10 Research Methods	25

3.0 Method	26
3.1 Expectations	26
3.2 Design Tools	26
3.3 Initial Experimentation.....	27
3.4 Circuit Design Requirement Analysis.....	28
3.4.1 PIC selection	28
3.4.2 The Microchip PIC16F1709	30
3.4.3 Filter Requirements.....	30
3.4.4 The State-Variable Filter.....	31
3.4.5 Power Supply Requirements	32
3.4.6 Stepper Motor Requirements	33
3.4.7 Miscellaneous Requirements	33
3.5 Circuit Description - Schematic	34
3.5.1 Schematic Design Overview	34
3.5.2 Buffer	35
3.5.3 Filter	35
3.5.4 Amplifier and Schmitt-trigger.....	38
3.5.5 Stepper Motor Circuitry	40
3.5.6 Power Supply	42
3.5.7 PIC and LCD.....	43
3.5.8 Button Debouncing Circuits.....	44
3.6 Software	45
3.6.1 The Capture/Compare/PWM Module	45
3.6.2 Boot Sequence.....	47
3.6.3 The Tuning Algorithm	48
3.6.4 Manual Motor Control	50
3.7 Circuit Description – Breadboard Prototype.....	51
3.7.1 Power Supply	52

3.7.2 Filter	53
3.7.3 Stepper Motor.....	54
3.7.4 Component list	55
4.0 Results	56
4.1 Filtering Results	56
4.2 Stepper Motor Motion.....	58
5.0 Planning and Project Management.....	59
6.0 Discussion	60
6.2 Pitch Detection	60
6.3 Motor Selection.....	60
6.4 Power Supply and Batteries	61
6.5 Reducing Pin Count	61
6.6 Feature Expansion.....	61
6.7 Miscellaneous Improvements.....	61
7.0 Conclusion.....	62
8.0 Reflection	63
9.0 References	64
10.0 Bibliography.....	67
11.0 Appendices.....	68
12.1 Appendix 1 – Stepper Motor Circuit – First Attempt	68
12.2 Appendix 2 – TL074 Op-amp characteristics	69
12.3 Appendix 3 – Stepper motor datasheet	71
12.4 Appendix 4 – CCP Setup Table	72
12.5 Appendix 5 – Gantt Chart	74
12.6 Appendix 6 – Flowcharts	75
12.7 Appendix 7 – PIC16F1709 Datasheet Key Pages.....	80

Table of Figures

Figure 1 - Overview of a Fender Stratocaster electric guitar	13
Figure 2 - Buzzing string versus normal string travel represented electrically.....	14
Figure 3 - Proteus Fourier analysis of E-string	16
Figure 4 – Raw D-string signal depicted with an oscilloscope.....	17
Figure 5 - LM2917/LM2907 output linearity plot	18
Figure 6 - The Roadie tuner	20
Figure 7 - 1980's example of analogue tuner circuit utilising frequency-to-voltage conversion techniques	21
Figure 8 - Cross section of a hybrid stepper motor illustrating the construction of the rotor and stator	22
Figure 9 – Cross-section of a variable reluctance motor illustrating the rotor's response and orientation to changing magnetic fields	23
Figure 10 - Simplified example of an H-bridge configuration	24
Figure 11 - The Microchip PIC16F1709 in plastic dual inline packaging (PDIP)	29
Figure 12 - PIC16F1709 feature set	30
Figure 13 - State-variable filter with independently settable gain and Q	31
Figure 14 - Frequency response graph of state-variable filter tuned to 82Hz, with cut-off frequencies.	31
Figure 15 - Current draw measurement on final prototype.....	32
Figure 16 - Overview of final schematic design	34
Figure 17 - Input-buffering circuit	35
Figure 18 - Frequency response plot of entire filter.....	36
Figure 19 - Detail of state-variable filter.....	37
Figure 20 - Amplifier and Schmitt-trigger circuit.....	38
Figure 21 - Irregular Schmitt-trigger output caused by negative signal components	39
Figure 22 - Stepper motor control and drive circuitry detail	40
Figure 23 - Power supply detail	42
Figure 24 - Detail of PIC and LCD schematic.....	43
Figure 25 - Button layout and debouncing circuitry	44
Figure 26 - Possible CCP division discrepancy	46
Figure 27 - Boot sequence.....	47
Figure 28 - Switch/case code excerpt.....	48
Figure 29 - Tune function overview. Refer to appendix 6 for more details.....	48

Figure 30 - Tuning function	49
Figure 31 - Altered interrupt routine for manual tuning mode	50
Figure 32 - Manual mode setup	50
Figure 33 - Prototype breadboard	52
Figure 35 - Lithium battery load test.....	52
Figure 36 - Filter section of breadboard.....	53
Figure 37 - Multiplexer truth table (functional diagram).....	53
Figure 38 - Stepper motor connection layout on breadboard.....	54
Figure 39 - Comparison between untreated guitar input waveform and filter output of D-string.....	56
Figure 40 - Comparison between input WAV file and output waveform on a simulated oscilloscope in Proteus.....	57
Figure 41 - Comparison of input waveform and final result of filtering and Schmitt triggering.....	57
Figure 42 - Early circuit development schematic	68
Figure 43 - TL074 datasheet front page	69
Figure 44 - TL074 electrical characteristics.....	70
Figure 45 - Stepper motor datasheet front page	71
Figure 46 - CCP setup table	72
Figure 47 - Microchip tips and tricks: CCP period measurement.....	73
Figure 48 - Gantt chart, full view.....	74
Figure 49 - Flowchart for main function.....	75
Figure 50 - Interrupt routine flowcharts.....	76
Figure 51 - Manual mode function flowchart	77
Figure 52 - Manual function alternative inspired by finite state machine flowchart	78
Figure 53 - Tune function flowchart	79
Figure 54 - PIC16F1709 datasheet front page	80
Figure 55 - PIC16F1709 block diagram.....	81

List of Tables

Table 1 - Pin count requirement diagram for I/O.....	28
Table 2 - Microchip parametric search entries - prototype	29
Table 3 - Filter specification overview	36
Table 4 - PIC pinout table	43
Table 5 - Component list.....	55

Acknowledgements

As this dissertation marks the end of 3 years in university, it is important to acknowledge that I did not do it without help. These 3 years have been both fun and sometimes frustrating, culminating in countless hours in the lab and library. I would like to thank my supervisor Dr Stephen Mosey for putting up with my 1 million questions on filtering and for guiding me through this endeavour despite my best efforts to procrastinate. I would like to thank Dr Gary Jones for his invaluable knowledge of CCS and Proteus and for his help with providing the specific software needed for this project. I would also like to thank Gareth Jones for helping me gain both a genuine interest and a broader understanding of embedded systems through his excellent lectures.

The foundation for this dissertation was laid by the excellent analogue electronics teachings by Mike Vanstone. I would also like to thank Dr Kelvin Sutton for his amazing maths lectures during 1st year.

I simply owe too many people my gratitude to mention them all here.

Support from my girlfriend, family and friends kept me going even when I was so stressed I doubted my decision of going to university. I want to thank my girlfriend Paige for supporting me emotionally, financially and for giving me advice on dissertation writing and university in general. You have been an important constant in my life, and I would never have been able to complete this without you. My immediate family also deserves recognition for their encouragement and support.

To my sister Dr Linda Jarrett: Thank you for inspiring me to pursue higher education. I still have a long way to go before reaching your level, and I am not sure I ever will.

-Thorbjørn Ravnaas Kristiansen

1.0 Introduction

Tuning musical instruments is one of the fundamental parts of music production, either on a hobby basis or in a professional setting. Listening to musical instruments out of tune can be perceived as uncomfortable to some listeners. The task of tuning an instrument is a trivial one to experienced musicians, especially to individuals with absolute pitch (the ability to discern the pitch of a sound without an external reference[1, p. 1]). However, the majority of people do not possess this ability, and a trained ear perceives changes in pitch more accurately than an untrained one[2]. This results in tuning becoming a time-consuming challenge for beginners, often getting in the way of practice. Professional musicians relying on tuned instruments also require efficient methods of tuning, which is the core problem this project will attempt to solve.

The aim of this project is to conceptualise and prototype a guitar tuner with repeatable, measurable accuracy. These aims will be achieved through a predetermined set of objectives:

- Conduct thorough background research to describe existing technologies and explore the different approaches
- Design a schematic capable of simulating the circuit's response to a real guitar output signal
- Design mechanical interface between tuner and guitar
- Construct a working prototype to a predetermined level of accuracy ($\pm 3\text{Hz}$).

This project relies heavily on accurate pitch detection. A fundamental requirement for the success of this project is to understand the concepts behind pitch detection techniques and how to apply them within the scope of this design.

2.0 Background Research

2.1 Literature Review

The field of musical pitch detection and tuning technology spans many disciplines, ranging from the physics of musical instruments to explaining human music perception through the field of psychoacoustics. Fletcher and Rossing provided a good research origin pertaining to the physical qualities of stringed instruments in chapter 10 of *The Nonlinear Physics of Musical Instruments*[3, p. 571]. Georgia State University has a fantastic online resource with further information on the physical movements of guitar strings, which is especially important for this application[4]. Roederer extrapolates further by investigating the psychological and biological connection to soundwaves. His paper *The Psychophysics of Musical Perception* also covers a selection of terminology relevant to this paper, like timbre and harmonics[5]. The field of acoustics is thoroughly researched, as Rossing illustrates in his review of *Springer Handbook of Acoustics*. The scope and span of the field itself is further underlined by the size of the book, at over 1000 pages written by 33 expert authors in the field[6].

A large portion of this project was dedicated to investigating pitch detection techniques. Although digital signal processing (DSP) techniques are more effective at pitch detection than the proposed solution in this article, Shanaz et. al. found that even state of the art DSP is not perfect when speech is the source data[7, pp. 333–334]. This may prove beneficial to the viability of this project, as instrument frequency signatures are less complex than human speech due to fewer variations in pitch. DSP techniques for detecting pitch in speech has existed since at least 1976, as a study of different pitch detection algorithms by Rabiner et. al. covers at least two types of autocorrelation (a software-based technique). Their findings concluded with that choice of technique was fully dependant on the application[8], which supports the previous claim that instruments require less complex systems for pitch detection. The general impression when researching pitch detection techniques was that most existing peer-reviewed research in the field pertained to vocal pitch detection.

Stepper motors and their applications are well documented through several forms of media. Chapter 10 of Hughes' *Electric Motors and Drives* provides a good foundation for research into stepper motors and their basic operation[9, pp. 314–343]. The paragraphs in this chapter ranges from basic descriptions of stepper motor functionality to descriptions of drive controllers and their different applications. For more in-depth literature regarding the electrical characteristics of different types of motor, Werninck's

Electric Motor Handbook was one of the main sources used[10, pp. 553–579]. It also offers advice and guidance on selecting the correct drive circuitry and motor, but this was mostly disregarded since the book was printed in 1978, and technological advancements in drive circuitry and motor design has made this section somewhat obsolete. Kenjo’s *Electric Motors and their Controls* provides a broader look at both DC and AC motors, covering both stepper motors and the design fundamentals for other types of motor[11]. Motor design techniques from the 1990’s are similar to the level of sophistication found in this project, making this a good source during early research. A public dissertation from MIT written by A. Bocanegra describes a system for automatically tuning guitar strings by coupling an existing digital guitar tuner to a DC motor by means of passive components and an op-amp. This provided insights into motor torque requirements, while the rest of the information was mostly discarded due to inaccuracies like claiming the high E-string has a frequency of over 1kHz[12].

2.2 The Limits of Human Hearing

The quantisation of human hearing sensitivity will determine the accuracy parameters for any tuning system. Sound and acoustics as a discipline have been thoroughly studied. For example, the Springer Handbook of Acoustics alone covers 8 separate disciplines including psychoacoustics[6][13]. As such, many efforts have been made to determine the range of human hearing, commonly accepted as approximately 20Hz-20kHz for an average adult. This range does not give any hints towards the average granularity of human hearing. Through an informal test utilising a simple in-browser signal generator, the author started noticing a difference between 2 distinct frequencies when the separation was more than 3Hz. The test was conducted blindfolded, with an assistant adjusting the frequency in 1Hz increments. The author would then signal when he heard a difference in tonality. Though informal and anecdotal, this small-scale experiment provided an absolute minimum accuracy rating, as the author’s hearing abilities are average at best. A 2007 study targeting the different levels of pitch perception accuracy between professional and amateur musicians found substantial differences in perception ability relative to musical proficiency[2].

Humans tend to prefer few tonal superpositions (finite ratios between frequencies) instead of a more analogue linear scale of preference. Roederer argues that this may be due to our sensory networks having a certain capacity limit in terms of sensory process bandwidth. Therefore, the brain prefers to work with fewer known values in order to

process them correctly[5, p. 26]. This can be indirectly compared to both how a processor has a finite amount of instructions in its register, and how the Nyquist-Shannon sampling theorem describes that the sampling rate for digitising (quantising) sound has to be at least twice the frequency of the highest expected sound frequency in the signal to prevent sampling errors like aliasing[14, p. 21][15, p. 1]. This innate preference has led to the development of a unique system for describing sound within music, basing itself on discrete levels such as octaves and cents.

2.3 Music Theory Glossary

This paper will be using various music and audio terminology. Below is a selection of relevant terms.

Fundamental Frequency

The fundamental frequency of an audio signal is the lowest frequency component of the signal. Fundamental frequency is also called pitch and is abbreviated with f_0 [16, p. 1].

Harmonics

Harmonics are overtones accompanying an audio signal, represented by integer multiples of f_0 .

Timbre

Timbre is the collective name for f_0 and all its layered harmonic components. This is what creates musical sounds, as physical instruments seldom produce a pure sine wave of a single frequency. Timbre is how the sounds of musical instruments are distinguished from another, even though they can share the same f_0 [5, p. 22].

Octave

An octave signifies a relative doubling in frequency, i.e. $E_2=82\text{Hz}$, $E_3=164\text{Hz}$ where E_2 and E_3 represents the 2nd and 3rd octave of the note E.

Sharp and Flat

When a string has too much tension and is vibrating at a frequency higher than its tuning specification, it is defined as sharp. The opposite (tension too low) is referred to as flat.

2.4 Electric Guitar Fundamentals

When a guitar string is plucked, it vibrates at its resonant frequency. The resonant frequency is dependent on the tension, mass and length of the string[3], [4]. A guitar can represent a wide range of frequencies for each string by altering the tension and the length of the string.

To aid in the understanding of this document, some basic information on guitar construction and its components is needed. A modern electric guitar is made up of several different key components, as seen in figure 1 below:

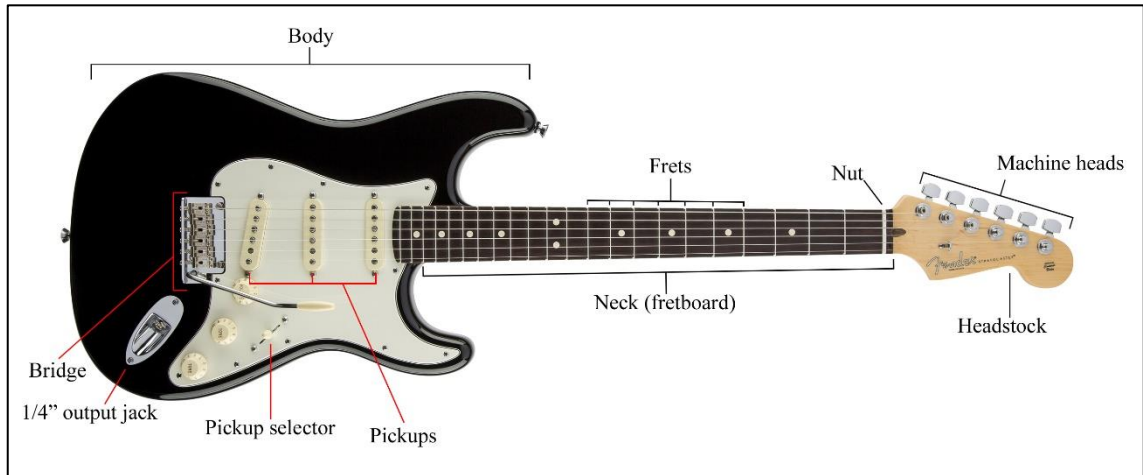


Figure 1 - Overview of a Fender Stratocaster electric guitar. Adapted from original image: Fender Musical Instruments Corporation.

2.4.1 Structural Components

Its main components are the headstock, neck and body. These components form the structure of the guitar, providing the necessary mix of rigidity and flexibility to ensure correct operation and acoustic behaviour. Rigidity is important to maintain the structural shape of the instrument under the tensile load of the strings and to ensure the guitar keeps its tuning. Attached to these components are the mechanisms and electronics necessary to accurately tension the strings and translate their mechanical movement into electrical energy. The neck is essentially a frequency divider. It consists of a structural backbone, truss rod and a fret board. The truss rod is a metal rod which can be shortened or lengthened slightly, to alter the curvature of the neck. The adjustment point (the end of the truss rod) can be seen just to the right of the nut in figure 1. A correct neck curvature is essential to prevent string “buzz”, an undesirable effect caused by too little distance between the string and fret board, causing the string to touch the frets at the end of its vibratory stroke.

As seen in the left-hand side of figure 2, a string (E-string) touching the fret board creates high frequency noise affecting the readability of the signal. On the right-hand side, the amplitude and frequency of the signal is calculated by the oscilloscope. On the left-hand side however, the signal is too noisy for the oscilloscope to perform valid calculations and output these parameters.

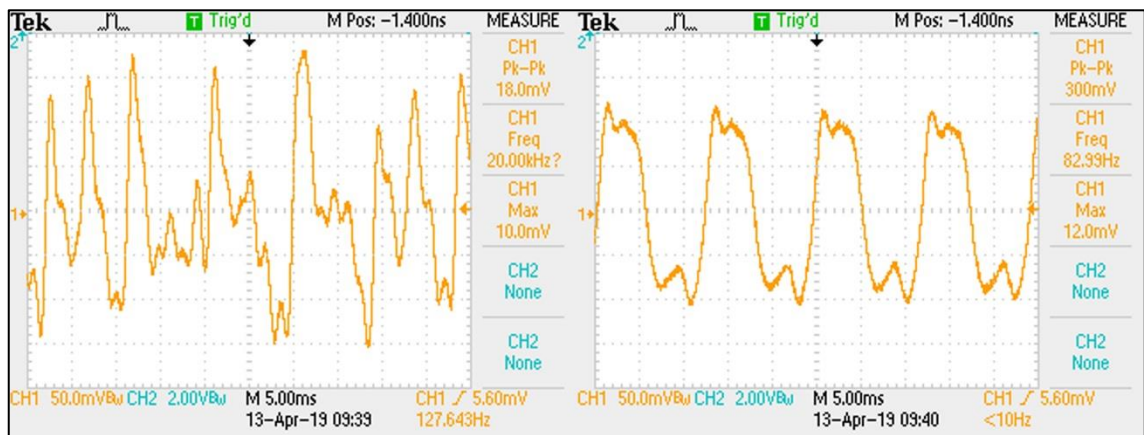


Figure 2 - Buzzing string versus normal string travel represented electrically.

Since the frequency of a string is a product of length, mass and tension, depressing a string onto one of the frets effectively shortens the string and increases its frequency. The tension (and therefore frequency) can be altered by turning the machine heads, which is the essence of this project.

2.4.2 Electronic Components

As opposed to an acoustic guitar, an electric guitar outputs AC voltage rather than acoustic soundwaves.

As seen in figure 1, the strings are tensioned over the pickups, which are magnetic coils. Much like the principle of an alternator, the metal (strings) pass over the magnets and energises the coil, creating a changing electric potential (AC). The pickups are connected to the output jack. This AC voltage is fed to various equipment, most commonly an amplifier. This allows for manipulation of the guitar signal before converting it to soundwaves through a speaker.

The output signal can be attenuated by adjusting a potentiometer, the volume knob. Some guitar models (like the one pictured) also have filtering circuits built in, allowing for passive tone control by turning the “tone” knob.

2.4.3 Strings

In scientific pitch notation, E2, A2, D3, G3, B3 and E4 labels the strings (top to bottom, figure 1), where the number after the note letter determines the octave of the note. These frequencies are 82Hz, 110Hz, 147Hz, 196Hz, 247Hz and 330Hz respectively (rounded to the closest integer)[17][18].

The strings are anchored at the bridge and attached to the machine heads on the headstock, resting on the “nut”. The purpose of the nut is to align the strings, adjust the string height (action) and ensure they are the same distance between anchor points. The bridge is the main attachment point of the strings and is also the adjustment point of the string action. Some models have floating bridges, allowing for temporarily variable pitch adjustment of the strings (vibrato) by moving the vibrato arm. The guitar pictured in figure 1 is a Fender Stratocaster with a floating bridge design.

2.5 Pitch Detection

Pitch detection is perhaps the most fundamental technical aspect of this project. The ability to accurately determine the frequency of a guitar string also dictates the accuracy of the tuning process. At the heart of this design lies a PIC microcontroller. As it is digital in nature, the analogue signal from a guitar string needs to be converted into a digital signal with good repeatability. To successfully accomplish this, the raw signal needs to be refined. After its refinement, the frequency can be accurately measured by utilising techniques in the frequency domain or the time domain.

2.5.1 Frequency Domain

In the frequency domain, mathematical operations can be performed on the waveform to obtain information about its frequency spectrum. This can provide information on which frequencies are the most common or of the highest magnitude within the signal. A pluck on each of the strings was recorded with Audacity, a free audio editing program. The recording was sourced directly from the guitar output jack and into the microphone jack of the computer. A Fourier analysis on the created audio files was then performed in Proteus, providing the frequency spectrum of the individual strings. See figure 3.

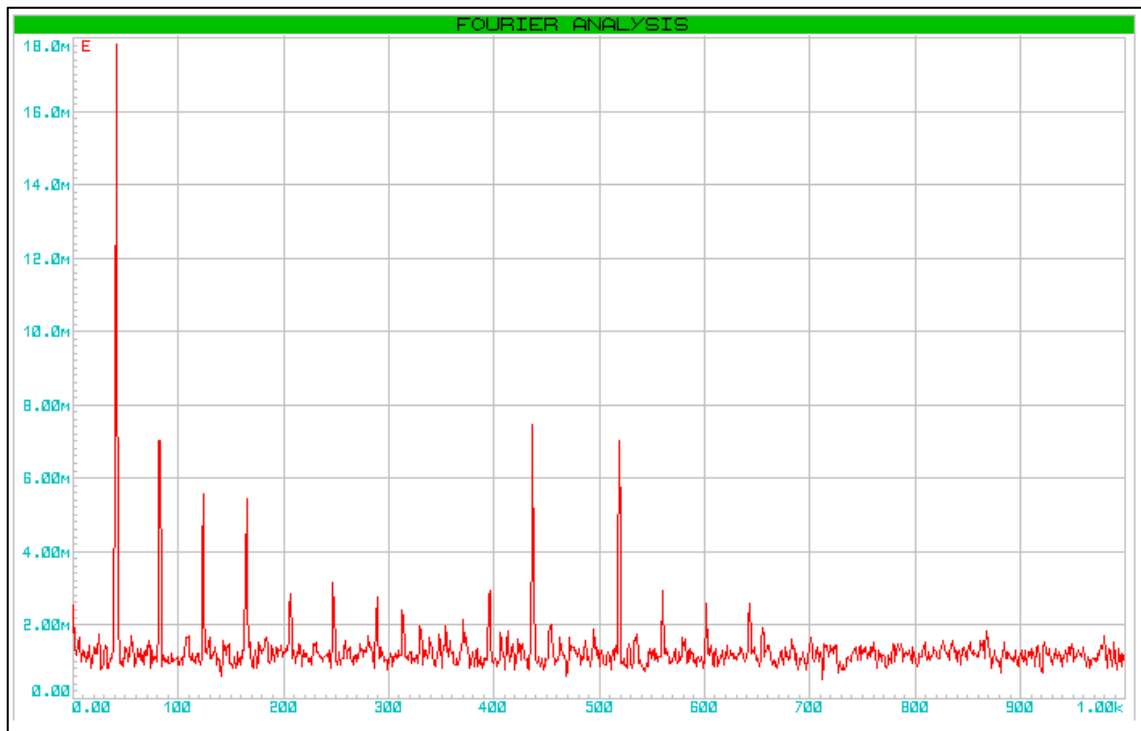


Figure 3 - Proteus Fourier analysis of E-string

This highlighted some potential problems with the frequency domain approach. The undesirable harmonics of the respective notes are prevalent on the spectrum; the desired frequency of 82Hz is of a much lower magnitude than the first harmonic of 41Hz making Fourier analysis frequency determination unsuitable as a standalone technique. These harmonics make up the timbre of the note, defined as a layer of overtones on the carrying fundamental frequency[5, p. 22].

2.5.2 Time Domain

Instead of analysing a frequency spectrum with complex mathematics, a more simplistic approach is available. Determining frequency in the time domain simply translates to counting periods over a given timeframe or recording the time of a pre-determined set of periods. For example, counting 10 periods every 100ms gives a frequency of 100Hz, given by the equation

$$F = \frac{\text{Periods}}{\text{Time in seconds}} \therefore \frac{10}{0.1s} = 100\text{Hz}$$

In figure 4, a raw guitar signal sourced from the output jack of an electric guitar is depicted through an oscilloscope (the string was slightly out of tune). The fundamental frequency (the lowest frequency) of the D-string is 147Hz, which is difficult to discern due to the many composite frequencies

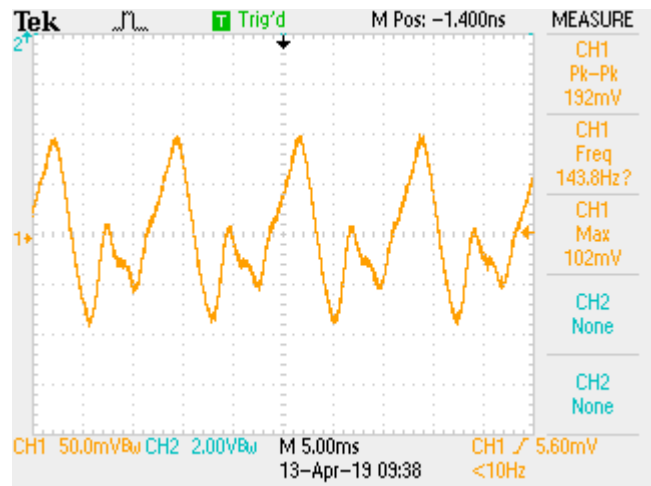


Figure 4 – Raw D-string signal depicted with an oscilloscope.

accompanying the fundamental. The amplitudes of these composites vary from miniscule to larger than the amplitude of the fundamental itself. These frequencies represent a multiple of the fundamental, called harmonics. It is therefore necessary to drastically alter the signal to successfully extract the fundamental frequency in the time domain. There are several filtering techniques available to achieve this goal, each with their strengths and weaknesses.

Time domain pitch detection has another significant advantage. Since there is no analogue to digital conversion, there will be no sampling errors or aliasing issues and the system will require significantly less processing power. The readability and validity of the end signal is wholly based on the quality of the filtering circuit alone.

2.5.3 Autocorrelation

Another time domain detection technique is known as autocorrelation. Autocorrelation algorithms compares a “snapshot” of a signal to the rest of the signal, looking for patterns. This removes the need for excessive filtering, as the inherent harmonics within the signal can be used to obtain the fundamental frequency by means of measuring repeatability. The high frequency noise components still need to be filtered out, as this adds randomness to the signal and autocorrelation’s working principle is pattern recognition. Since the harmonics are layered overtones on the fundamental frequency, the entire fundamental with all its timbre can be recorded. An AD conversion is then performed, assigning a numerical value to this pattern (a digital fingerprint). As the signal progresses, the program looks for more instances of this pattern, and records the time between each match. This then translates to the fundamental frequency of the signal.

2.5.4 F/V Conversion

Frequency-to-Voltage conversion is a pitch detection technique where a frequency is fed to a F/V IC. The F/V converter converts an input waveform to a square wave with a comparator circuit, which is then converted to a voltage by means of a charge pump[19, p. 10]. This output voltage is mostly unaffected by temperature changes, with a linearity error of only 0.4-0.5% across a wide range of temperatures as seen in figure 5. An

analogue to digital conversion can then be performed on the output voltage to digitise the value for use within the tuning algorithm. This method showed great promise and was a candidate for the choice of technique when designing the tuner, with performance equalling the chosen solution. The performance of this type of circuit when signal harmonics are present remain unknown. It can be reasonably assumed that a false reading may take place if the F/V converter treats these harmonics as part of the fundamental, “increasing” the frequency and thereby the output voltage. More research and/or a simulation is needed.

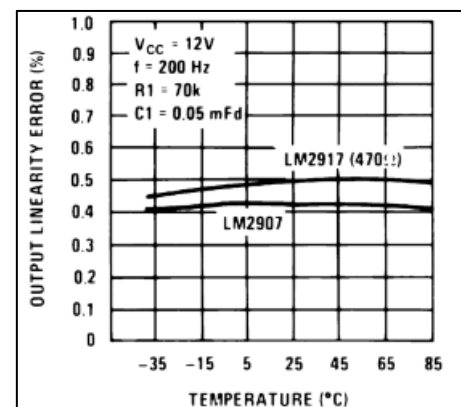


Figure 5 - LM2917/LM2907 output linearity plot. Retrieved from <http://www.ti.com/lit/ds/symlink/lm2917-n.pdf> p.7

2.6 Filtering

An audio signal can be fed through an electronic system such as a filter to isolate or exclude a specific frequency band. Horowitz and Hill defined a filter as:

“..a circuit with some deliberate passband and attenuation characteristics (both amplitude and phase) versus frequency.”[20].

Both passive and active filtering circuits exist; passive systems rely on discrete components such as capacitors, resistors and inductors, while active filter systems utilise ICs such as operational amplifiers (op-amps) for their excellent gain characteristics and high input impedances. The common denominator for all the researched pitch detection techniques was that filtering is always required to a varying degree to create a viable signal.

2.6.1 Passive Filters

The simplicity of passive filter design renders it an attractive solution. In its simplest form, a passive filter consists of a resistor and a capacitor/inductor in parallel. This topology can be altered to either function as a high pass filter (HPF) or a low pass filter (LPF), based on the position of the capacitor/inductor relative to the resistor. When combining these filters, a band pass filter is constructed, with both a high and low cut-off point.

A passive filter typically exhibits a 6dB/decade roll-off[20, p. 391] and a max Q of less than one half[21], which is too low since the guitar string frequencies are close enough to create overlap in the bandpass regions.

2.6.2 Active Filters

In order to obtain a sharper roll-off slope it is imperative to maximise the input impedance for each filter stage to prevent frequency response-affecting resistance loading. Operational amplifiers inherently possess a very high input impedance, as they are designed based on transistors rather than passive components. Where there is high impedance, there is low current which effectively renders op-amps voltage driven devices as opposed to current driven devices. This creates a buffering effect on each stage of the filter, which will improve response characteristics. A Sallen-Key filter (named after professors R.P. Sallen and E.L. Key) is one of the most popular filtering designs available since its initial description in 1955[21, p. 1].

2.7 Existing Efforts and Designs

There are many ways to design a guitar tuner, where creativity is the only limiting factor. Comparing this project to another tuning system will therefore require selective comparisons. The following examples were selected either for their likeness to this project or to illustrate how to accomplish the same results utilising different techniques and disciplines. These existing designs were found on the internet on various hobbyist and educational websites. The second example (the Roadie tuner) is incorporated to showcase the industry standard in automatic tuning. It is also the only example in this text referring to a finished, launched product.

2.7.1 “The Mechanix” Guitar Tuner

A public final project found on Cornell University’s websites revealed a project exceedingly similar to the described project in this paper[22]. Its analogue filter design is less complex where a simple Sallen-Key low-pass filter with a Butterworth response is utilised. To mitigate this, a digital Infinite Input Response (IIR) filter was designed in Matlab and implemented in an ATmega32 MCU.

The conclusion lists accuracy and speed issues as its main problems. A sampling rate of 2kHz may not be enough for accurate DSP pitch detection.

2.7.2 The Roadie Tuner

Start-up company Band Industries produces an automatic guitar tuner known as the Roadie tuner[23]. It is the main design inspiration for this project, where a design approximation of its working principle was attempted. It is a fully automatic tuner which can tune multiple stringed instruments such as guitars, basses and ukuleles among others. Little information is given about its inner workings, although it can be assumed it uses a contact microphone system since video demonstrations shows its capability of tuning in noisy environments. The tuner can be seen in figure 6.



Figure 6 - The Roadie tuner.

Retrieved: www.roadietuner.com

2.7.3 Frequency to Voltage Conversion

An article in *Electronics & Music Maker* from 1981 explains how an analogue guitar tuner circuit was designed utilising an F/V converter[24].

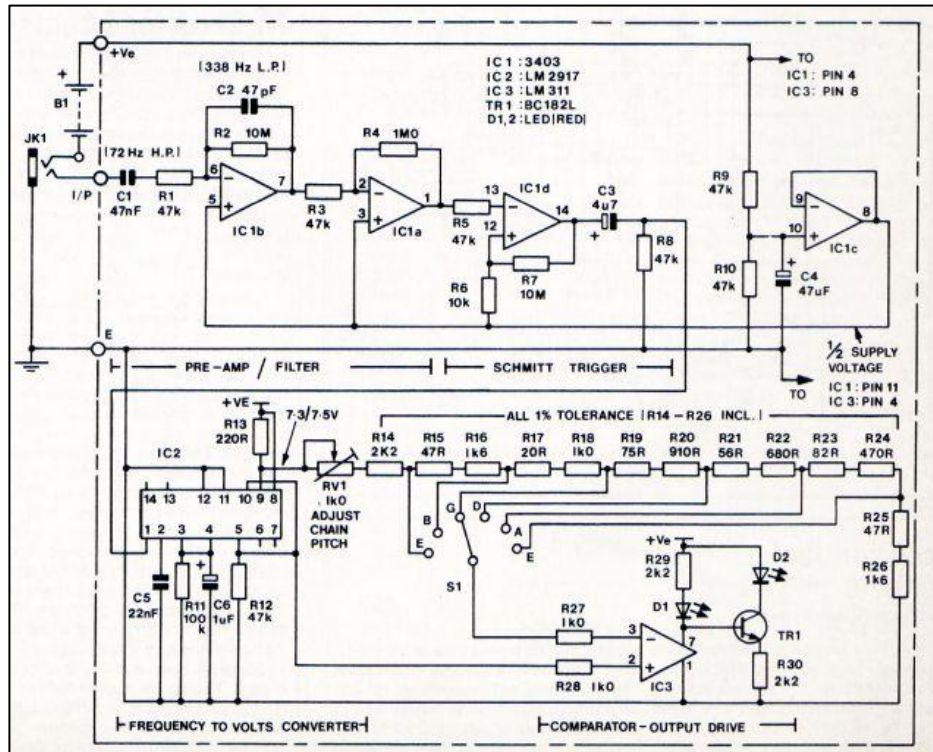


Figure 7 - 1980's example of analogue tuner circuit utilising frequency-to-voltage conversion techniques. Retrieved: <http://www.muzines.co.uk/articles/guitar-tuner/2656>

In this relatively simple circuit (figure 7), a band pass filter is implemented to isolate the 72Hz-338Hz band, resulting in a bandwidth of 266Hz. A 6-position rotary switch is used to select a specific resistance in a series of reference resistors. The voltage drop across these resistors is compared to a reference voltage generated by the F/V-chip by means of a comparator circuit. The comparator then operates 2 LEDs for indicating whether a string is sharp or flat.

The main advantage to this circuit is its analogue robustness and simplicity. It is however, limited by component tolerances. A properly designed digital systems can to some extent disregard component tolerances, as digital systems operate based on clock pulses and timing as opposed to analogue components and their tolerances. This system in conjunction with an A/D converter would be a good alternative if calibrated.

2.8 Stepper Motors

The essence of this project is to assist the user with the tuning process by automatically turning the machine heads based on a string's frequency. Stepper motors were chosen for their accuracy, holding torque and open-loop control due to their quantified operation of moving in discrete “steps”. They are also brushless (reliable) and possess good response characteristics to direction changes and starting/stopping[25, p. 1]. Holding torque is a feature found on stepper motors which electromagnetically locks the axle of the motor during standstill by energising coils without stepping to the next position[11, p. 114][9, p. 328]. This enables the user to use the unit as a “wrench”, allowing the device to turn the machine heads when not in tuning mode. Open-loop control allows for information on motor position and movement without the use of an enumerator or other servo-type feedback systems[10, p. 575][9, p. 316]. This can be useful when a certain angle of rotation is needed, i.e. when turning the machine head 45° counter-clockwise before a tune cycle.

2.8.1 Types of Stepper Motors

Stepper motors are classified as either active rotor (permanent magnet) or reactive rotor (variable reluctance)[10, p. 554]. There are 3 common types of motor available, the permanent magnet motor (PM), the variable reluctance (VR) motor and the hybrid stepper motor[25, p. 1]. The PM motor has a rotor comprised of radially opposed magnets, while the hybrid motor has a rotor of both radially and axially opposed magnets by utilising a special rotor with teeth of alternating polarity in both axes (see figure 8). Hybrid motors are also preferable when fast stopping is desirable[10, p. 577]. Due to availability and cost, the chosen solution for this application was a permanent magnet stepper motor with a gear reduction system. While this solution does provide great accuracy and torque with the added benefit of holding torque, it does not have the size/performance characteristics on the level of a hybrid motor. The same could also be accomplished by utilising a worm-gear drive (the same mechanical system found in the head stock of a guitar), which only allows for the transfer of torque in

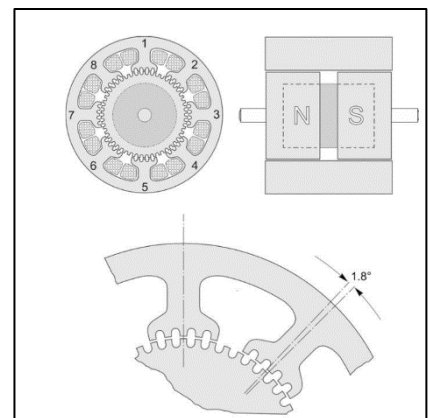


Figure 8 - Cross section of a hybrid stepper motor illustrating the construction of the rotor and stator. Image: Hughes, A. (2013). Electric motors and drives fundamentals, types and applications (4th ed.). Amsterdam: Elsevier, p.323

one direction. A permanent magnet stepper motor consists of multiple coil pairs and a rotor with multiple magnetised teeth. The magnetic orientation of these teeth alternates between north and south. When a coil is energised, the teeth are attracted to this coil and motion is created. The next coil pair is then energised, moving the rotor again. This is defined as a step. By switching alternating coils on and off in rapid succession, the speed of the rotor can be controlled with very high precision. PM motors can be made smaller than their VR counterparts due to a higher power-conversion ratio[11, p. 74]. The VR motor operates in a very similar fashion, only differentiated by its lack of permanent magnets. The rotor is instead constructed of a solid ferromagnetic material with pronounced “lobes” for interacting with the magnetic fields created by the windings in the stator. When a coil pair is energised, the lobes will orient themselves parallel to the windings to shorten the magnetic flux path[25, p. 1].

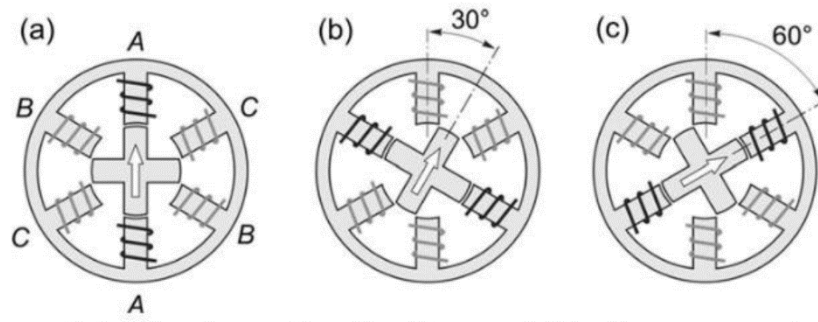


Figure 9 – Cross-section of a variable reluctance motor illustrating the rotor's response and orientation to changing magnetic fields. Image: Hughes, A. (2013). Electric motors and drives fundamentals, types and applications (4th ed.). Amsterdam: Elsevier, p.322

2.8.2 Unipolar Versus Bipolar Stepper Motors

A unipolar stepper motor has a very similar construction to a bipolar stepper motor. They are differentiated by the number of leads and their drive methodology. A typical unipolar stepper has 6 leads, 3 for each coil. Each coil is centre-tapped, allowing for directional control without the reversal of current. This allows for a less complex drive circuit, as an h-bridge is not needed. A disadvantage with unipolar steppers is their lower current-handling capability, due to thinner coil wires. A unipolar stepper motor can be operated like a bipolar motor by omitting the centre-tap leads to each coil. This effectively creates one “larger” coil. However, this doubles the stated coil resistance, as 2 coils are now being driven as one. This must be compensated for in the drive circuit, to ensure optimal efficiency, power and safety for the motor itself.

A bipolar stepper motor requires an H-bridge to be driven correctly. The motor itself requires current reversal in order to flip the magnetic field the correct way. This can be done by utilising 4 MOSFETs, arranged so that the drive current polarity changes when they are being switched correctly. An example of this can be seen in figure 10. The rotational direction of the motor can be altered by closing either SW1 and 2 or SW3 and 4. When replacing these switches with MOSFETs, the switches can be controlled electronically, allowing for very fast switching and precise control. When controlling a stepper motor, two of these circuits are needed to control each coil. This is called a dual H-bridge.

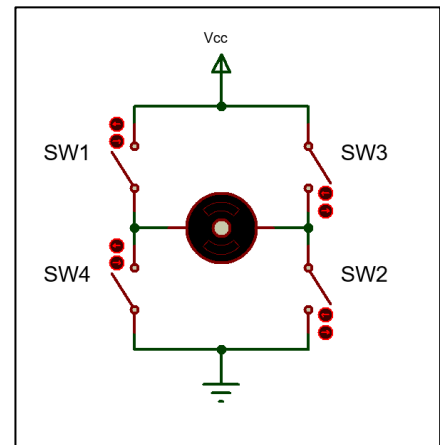


Figure 10 - Simplified example of an H-bridge configuration. The motor is in reality a single coil.

Dual H-bridges in integrated circuit form are available (stepper motor drivers). When coupled with a stepper motor controller, they provide an uncomplicated drive circuit, with digital signals controlling the speed and direction. The controller instructs the driver which coils to switch in sequence when receiving input pulses. This renders driving the motor with pulse width modulation (PWM) possible, simplifying the stepper motor drive process significantly.

2.9 Research Methods

The aims of a development project is to deliver a product (software, hardware or both) that satisfies a set of defined requirements[26, p. 6,7]. These requirements were set early on as described in the introduction through the aims and objectives.

Following the research onion model[27] by Saunders et. al, the first layer is the research philosophy method. The chosen research philosophy was the positivist approach, which is well suited for an engineering project due to the research being possible to replicate by others.

The next layer of the hypothetical onion is the approach. There are 2 main approaches, deductive approach or the inductive approach. An inductive approach is better suited for research projects with little existing information, enabling the researcher to form a hypothesis after research is concluded. Instead, a deductive approach was taken, where a hypothesis (or in this case, a goal) is set from the beginning, before culminating in the conclusion.

The third layer defines which strategy is chosen to achieve the desired outcome. This is an experimental project, which dictated the choice of an experimental strategy. This ensures the results can be replicated by others, thereby validating the conducted research done by the author.

The fourth layer decides which research methodology to use. When comparing quantitative and qualitative research, the latter is usually regarded as the most fitting for engineering projects. It focuses more on hard data, numbers and factual information, while the former is better suited for humanities-based research[28, p. 1].

The fifth layer does not apply directly to an engineering project, as it deals with either long-term or short-term gathering of data. Regardless, the most fitting option would be the longitudinal option, where research is collected and processed over an extended time period.

The sixth and final layer deals with the data collection approach, where the onion is better suited for humanities-based projects and other soft-sciences. Data collection for this and other engineering-based projects are usually done in a strict quantitative manner, where numbers and results matter more than conjecture and human psychology.

3.0 Method

3.1 Expectations

As the purpose of building this circuit is to digitise the frequency data of an analogue signal, it must be assumed that a signal conversion process is central. The goal of this conversion is to create a digitally readable signal, of which data can be processed, translated and manipulated by a microcontroller. The digital nature of microcontrollers makes it operate in the binary domain. This is fundamentally 1 and 0, voltage or no voltage, of which a square wave is the closest analogue equivalent. Intuitively, this process requires filtering to isolate the desired signal, effectively increasing the signal-to-noise ratio (SNR). The desired output is a clean square wave with the same frequency as the fundamental frequency of the input signal.

3.2 Design Tools

The circuit was exclusively designed and simulated with Labcenter Electronics Proteus 8.2 (hereby referred to as Proteus), a computer-aided design (CAD) package capable of mathematically simulating real-time circuits to a high degree of realism. The program code was written in C and compiled with the PICC compiler by Custom Computer Services (CCS).

Prototyping was done on multiple breadboards, each section separating a key circuit area for ease of troubleshooting and to simplify the manual component population of the breadboard. This was simplified because more space was created, and translating the schematic diagram required less effort when the components were placed in logical locations. The circuit was designed with discrete components and analogue integrated circuits like operational amplifiers and multiplexers. The only digital components were the LCD, stepper controller and the PIC microcontroller itself (disregarding the Schmitt-trigger).

Other equipment used were 3 standard adjustable laboratory power supplies, a function generator and basic tools (herein a Fluke 177, a soldering iron and assorted hand tools). Screengrabs of waveforms and graphs were done both through Proteus and Audacity, apart from oscilloscope pictures where the internal screengrab feature was used.

3.3 Initial Experimentation

This project was arguably initialised from the wrong end. Early design experimentation focused more on the stepper motor drive circuitry than filtering techniques and software. An advantage to this approach is the ability to declare the viability of the project early on; if the stepper motor design approach was deemed unsuitable, a direction change could be made to incorporate other solutions and allow for more time on productive progress.

Designing a stepper motor circuit in Proteus involves either manually building an h-bridge or using a purpose-made motor driver and controller chip. The latter approach offers many advantages both in time and complexity, making it a natural choice. The standard Proteus library has a limited choice of stepper motor controllers, with only an L297 controller available. This dictated the choice of the physical controller later in the build, as simulation capabilities were key in the development process. While investigating the datasheet of the STMicroelectronics L297 controller, it was discovered that it has a dedicated driver IC designed to work in conjunction with it, the L298 from the same manufacturer[29][30].

The stepper circuit was first built to perform with a randomly chosen microcontroller, as the feature requirements for running such a circuit were simplistic (refer to appendix 1 for reference to first circuit revision). It was later redesigned with the final microcontroller after a design requirement analysis was performed. The stepper circuit design remained virtually unchanged after this, with only minor changes to accommodate different sizes and types of motor. A 16x2 LCD was also incorporated in standard 7-pin mode as per the datasheet requirements. It was estimated that a 16x2 size display would suffice, as the user would only need basic instructions to operate the device. The LCD model was chosen for its low price and 4-bit connectivity[31]. Circuit details and schematic diagrams will be covered later in the report.

3.4 Circuit Design Requirement Analysis

3.4.1 PIC selection

Microchip's PIC series of microcontrollers was chosen for their flexibility, cost, availability and familiarity. They offer a wide range of functionality and has a large user base which aids with documentation, troubleshooting and advice. The PICC compiler is also designed to work specifically with Microchip products.

As this project requires simple IO and weak computation relative to ADC-systems, a PIC with an 8-bit instruction register was deemed more than sufficient.

Microchip offers a parametric search engine, where requirements are entered to produce a list of PICs with these qualities. The physical size of the PIC matters, as inputs and outputs are physical pins which occupy space. The pin count requirements can be seen in table 1:

Subsystem	IO function	Pin count requirement
Pitch detection	Signal input	1
Stepper motor control	Stepper toggle (on/off)	3
	Stepper direction	
	Stepper pulse	
LCD	LCD enable	7
	LCD register select	
	LCD read/write	
	Data pin 4	
	Data pin 5	
	Data pin 6	
Button IO	Start button	2
	Toggle button	
Multiplexers	MUX-pin A	3
	MUX-pin B	
	MUX-pin C	
Total pins (excl. power and ground)		16

Table 1 - Pin count requirement diagram for I/O

The LCD requires a total of 7 I/O pins, if connected in 4-bit mode (nybble-mode). This requires more operations per instruction to the LCD controller, but the slower operation will be unnoticeable at high refresh rates[32]. The multiplexing system of the filter and the pitch detection subsystem are separated in this table due to the multiplexing pins being configured as outputs.

The closest possible pin count to 16 in the parametric search engine is 20. 2 of these pins are reserved for power and ground, which leaves 2 pins for future upgrades.

The search was narrowed down further by specifying the attributes seen in table 2:

Parameter	Requirement	Comments
Program memory (kB)	≥ 14 kB	Prototype only
SRAM (B)	≥ 1024 B	Prototype only
Pin count	$= 20$	-
Max PWM outputs	> 0	-
UART modules (RS232)	1	For in-simulation debugging purposes

Table 2 - Microchip parametric search entries - prototype

While the pin count and PWM output requirement were final and necessary in the final product, the other requirements were exaggerated or included to aid in the design process. The program memory size of 14kB was an intentional overshoot, as no estimate of final program size existed. SRAM was also exaggerated since there were no indications on how demanding the final program would be.

Proteus contains an ability to act as a virtual terminal for software debugging. This requires a transmit and receive pin which will be occupied during virtual terminal use. The least expensive alternatives among the search results was the PIC16F1579 and the PIC16F18345 at US\$0.77. However, for a price increase of 11 cents (US\$0.88), the PIC16F1709 was available, which was the most extensively equipped PIC in the 17xx family. This provided a good prototyping platform which could develop along with the project. All prices listed are Microchip's own 5K prices. See appendix 7 for PIC info.

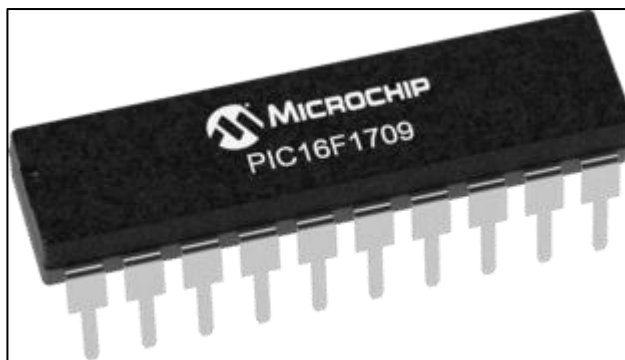


Figure 11 - The Microchip PIC16F1709 in plastic dual inline packaging (PDIP) – Image retrieved from Microchip.

3.4.2 The Microchip PIC16F1709

Being the flagship of the 16F17xx range, the PIC16F1709 is equipped with an excessive number of features relative to the needs of this project. This intentional overshoot margins offered a reliable prototyping platform and testbed for various modules of the circuit. The features are listed in figure 12. Important features listed here are the 2

Capture/Compare/PWM units which are the principal technologies behind the pitch detection technique as explained later. The 16-bit timer is also required for this purpose. No ADC is utilised in this

project, but this powerful capability could prove advantageous in the future. It also has a voltage range which allows for future changes in power supply design. The SPI and I2C peripherals could perform as a testbed for serial communications, possibly lowering the required pin count of a future circuit revision.

Parameters	
Name	Value
Program Memory Type	Flash
Program Memory Size (KB)	14
CPU Speed (MIPS/DMIPS)	8
SRAM Bytes	1,024
Digital Communication Periph...	1-UART, 1-SPI, 1-I2C
Capture/Compare/PWM Perip...	2 Input Capture, 2 CCP,
Timers	4 x 8-bit, 1 x 16-bit
ADC Input	12 ch, 10-bit
Number of Comparators	2
Temperature Range (C)	-40 to 125
Operating Voltage Range (V)	1.8 to 5.5
Pin Count	20
Low Power	Yes

Figure 12 - PIC16F1709 feature set - Retrieved from Microchip.

3.4.3 Filter Requirements

The filter design will be accommodating a low frequency range, both in bandwidth and absolute range. A high Q-factor and sharp cut-off slopes are desirable. The tuned guitar string frequencies range from 82Hz-330Hz, which gives a bandwidth of 248Hz. Early design revisions relied on a single Sallen-Key topology lowpass filter with a cut-off frequency of -3dB@600Hz. This was effective in blocking noise and narrowing the frequency band, but it was still difficult to discern the fundamental frequency because of the harmonic components. A window comparator op-amp design and a Schmitt-trigger was added to the filter output, to only capture the peak amplitudes of the signal in an unsuccessful attempt to acquire the fundamental frequency. This was largely unsuccessful due to the harmonics sometimes being of larger amplitude than the desired frequency, creating false input pulses for the PIC.

After much effort was put in to adapt and tune this filter, it was eventually abandoned as a substantially better filtering solution was discovered.

3.4.4 The State-Variable Filter

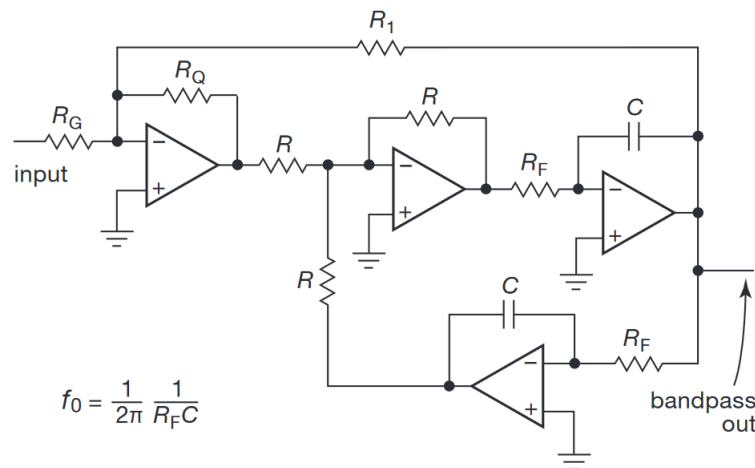


Figure 13 - State-variable filter with independently settable gain and Q. - Image retrieved from Horowitz, P., & Hill, W. (2016). *The Art of Electronics*. Cambridge University Press (3rd ed.). Cambridge University Press, p. 411.

The state-variable bandpass filter is a version of the Sallen-Key filter which offers a sharp Q-factor and steep roll-off at the expense of using 4 op-amps per filter[20, p. 411]. An early implementation utilised 6 of these filters, each filter tuned to a specific string. This was exceptionally unpractical and extremely CPU-taxing when simulating within Proteus (especially with audio samples). Although impractical, it showed great promise when a frequency analysis was performed. In figure 12, we see an exceptionally sharp frequency response, with a centre frequency of 82Hz and a bandwidth of only 48Hz. This filter still requires tuning, as the A-string sits at 110Hz. This would cause overlap, making the strings difficult to distinguish if they are very sharp or flat. Multiple of these filters were designed, each with different cut-off frequencies for the different strings and equal Q-factors and roll-offs. It became apparent that they all share all other qualities, only the fc-resistors changed. If these resistors could be switched, it would drastically reduce the size and complexity of the filter without sacrificing performance.

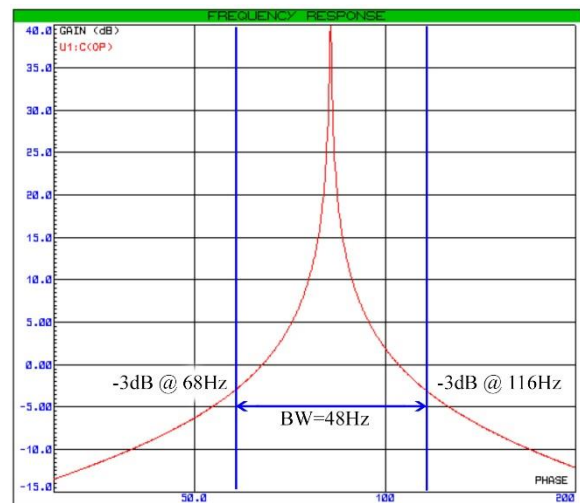


Figure 14 - Frequency response graph of state-variable filter tuned to 82Hz, with cut-off frequencies.

3.4.5 Power Supply Requirements

The circuit was initially designed without a power supply in the simulation, since lab power supplies were available upon the first prototype build. The power characteristics were then recorded using these supplies, and the requirements were calculated based on these numbers.

The unregulated 9V rail drew 100mA, while the -9V rail only drew 21mA. Since the -9V rail was mostly unused, its necessity became questionable. More details on this can be found in the Future Improvements section.

The unregulated rail for the stepper motor drew 63mA @ 16V under load, which would require a larger capacity battery pack than the PP3 batteries.

One tuning cycle was estimated to perform about 30 seconds of effective machine head rotation, based on the authors manual tuning time. Based on a halving of the voltage delta between free running and stall (42mA and 63mA), a power usage of 339mW was calculated by $P=VI$. The total power consumption for the circuit was approximately 1.5W under full load when including the dropout voltage inefficiency of the regulators.

The power supply was designed in the simulation after the lab measurements to verify the findings. This was more effective than calculating the power consumption for each individual component, since the error rate increases with the number of variables. The power supply requirements with the circuit running on battery power are documented in figure 15:

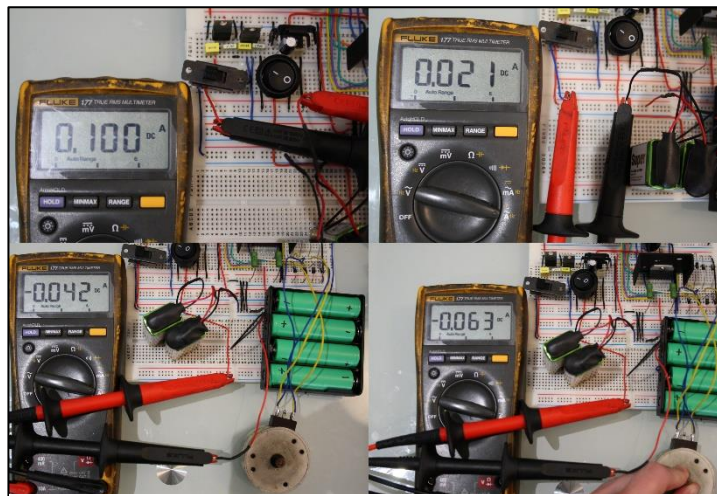


Figure 15 - Current draw measurement on final prototype. Clockwise from top left: +5V rail, -5V rail, +12V rail under motor stall, +12V rail with free running motor

3.4.6 Stepper Motor Requirements

If a stepper motor without a gear reduction is used, the motor would have to be unpractically large. A paper on designing an automatic guitar tuner from MIT specified a force of approximately 27Ncm required to turn the machine heads of an acoustic guitar[12]. This can be scaled down some, as the machine heads on electric guitars are generally easier to turn due to their smaller gear set and less resistance. The motor currently installed in the system has a force of 2.8Ncm without gearing, which renders a 10x gear reduction sufficient. Availability will dictate the choice of motor, however.

3.4.7 Miscellaneous Requirements

LCD

During simulation, it was noticed that a 16x2 LCD provided enough character space to convey the required information to the user. This includes information on tuning mode, instruction prompts and a frequency readout. Refresh rate, luminosity or colour were not critical. Price was the deciding factor upon choosing the LCD.

Switches and Buttons

Minimalistic I/O was prioritised, with fewer buttons and switches to control a limited user interface to further simplify operation of the device. This resulted with two push-button type single-pole single throw (SPST[33]) switches with debouncing circuitry and one double-pole double throw (DPDT[33]) switch to switch the 5V rails. The 12V rail was switched with its own SPST switch. This would ideally be accomplished with a triple pole configuration to reduce component count, but the design was limited by the lab inventory.

A potentiometer will be in place for setting the voltage reference level of the stepper motor driver IC[29, pp. 3–4] during the prototype stage. This would be replaced by a resistor of a fixed value when the product was complete, and a higher quality motor was acquired.

3.5 Circuit Description - Schematic

The circuit was designed multiple times in Proteus, with the progressing revisions increasing in complexity while altering some key systems like the filter. This section will cover the finished design, while any future changes will be discussed in the improvements section.

3.5.1 Schematic Design Overview

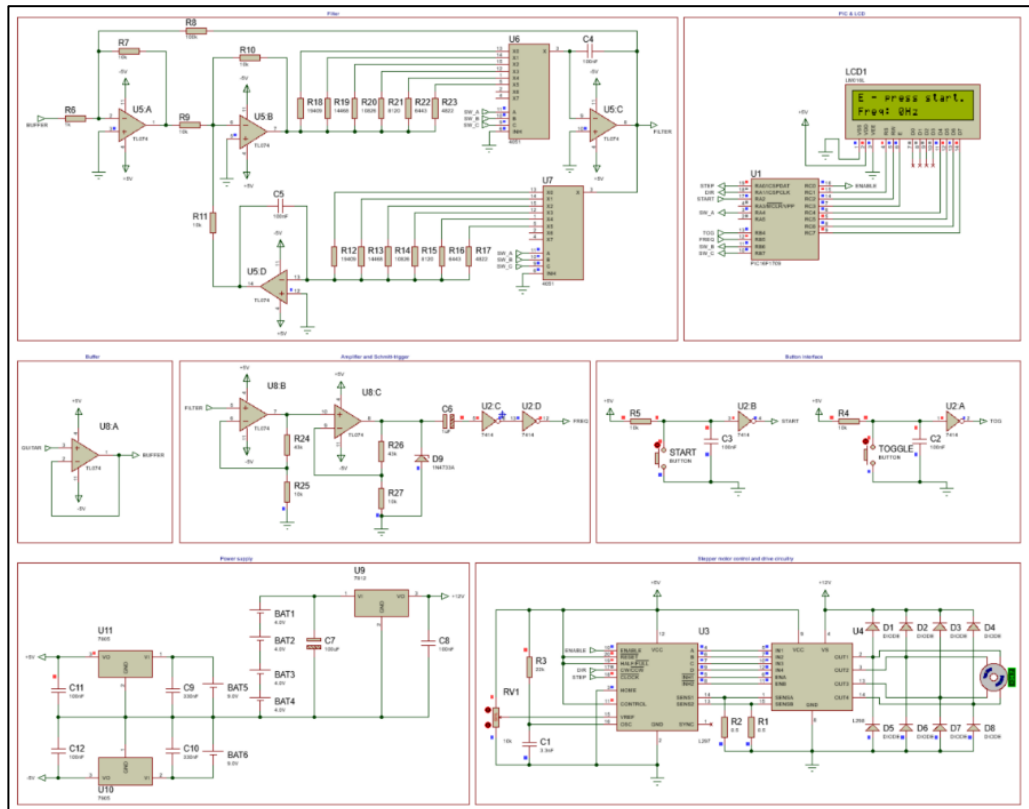


Figure 16 - Overview of final schematic design

The final circuit schematic was designed on one sheet incorporating all circuit elements. This schematic is the closest possible approximation to the physical prototype, as Proteus hides the power pins of the PIC. The noise reducing capacitors present on the prototype are not shown here, since the layout of these capacitors will differ between breadboard versions and an eventual PCB design.

3.5.2 Buffer

The main body of the filter is shown in the top left rectangle of figure 16. The buffering and amplification stages are not incorporated in the filter section, since their purpose is signal conditioning. The guitar is interfaced via a ¼” jack plug (not shown) and enters the buffer (figure 17).

The purpose of this buffer was to reduce the loading effect of the guitar on the circuit. The circuit was initially designed without input buffering, which caused a very noisy signal and “aliasing” issues when noise spikes were registered as data by the PIC. There are 2 quad op-amps in the circuit, and the buffer resides on the 1st amp of the second chip (component number U8,

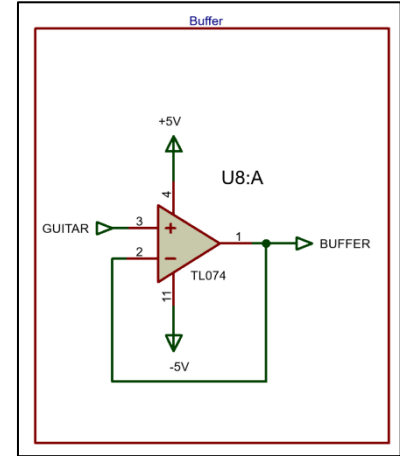


Figure 17 - Input-buffering circuit

amplifier A). Designing it this way meant the filter could have a chip to itself, since it required exactly 4 op-amps. Since the frequency range of this project spans only 248Hz and sits on a relatively low frequency level compared to modern op-amps capabilities, the choice of op-amp was not critical. The TL074CNE4 from Texas Instruments[34] was chosen partly for its availability in Proteus, but also for its price/performance characteristics. See appendix 2 for detailed overview of this IC.

3.5.3 Filter

As previously described, the chosen filter solution was a version of the state-variable filter, described by Horowitz and Hill. Provided with the filter design are the equations required to tune it[20, p. 411]. The Q-factor and gain was given by $Q = \frac{R8}{RQ}$ and $G = \frac{R8}{RG}$ where R8 is the resistor label in the schematic design. A modest gain of 6.7 was chosen by $R8=100\Omega$ and $RG=15\Omega$ to slightly amplify the 70-100mVRMS from the guitar output. The Q was first set higher than necessary, before adjusting it in-simulation to $Q=45$ for a satisfactory bandwidth. The equation for tuning the cut-off frequency was also provided:

$$f_0 = \frac{1}{2\pi RfC} \therefore Rf = \frac{1}{2\pi f_0 C} \therefore Rf = \frac{1}{2\pi \times 82Hz \times 100nF} = 19409.1394\Omega$$

In this equation, Rf is the resistor value for the 2 cut-off frequency resistors. The capacitor value was chosen for its commonality and “sensible” value. With only one unknown, it is possible to calculate Rf based on cut-off frequency and the value of the capacitor. This equation was done on all 6 string frequencies, with the capacitor value

String	f0 (Hz)	Fl (Hz)	Fh (Hz)	BW (Hz)	Rf (Ω)
E	82	74	91	17	19409
A	110	99	122	23	14468
D	147	133	163	30	10826
G	196	177	217	40	8120
B	247	223	274	51	6443
e	330	298	366	68	4822

Table 3 - Filter specification overview

remaining unchanged. Four decimal points is not necessary for this level of accuracy, all resistor values were rounded to the nearest whole number to facilitate easier potentiometer adjustment on the prototype. Table 3 provides an overview of required resistor values, cut-off frequencies and bandwidth. The average bandwidth is approximately 38Hz, which is enough to mitigate the effects of input amplitude variation, explained further down.

An interesting observation here is that the bandwidth increases as the frequency increases. This is due to the nature of the equation, where the denominator sum increases as the frequency variable increases. The only way to mitigate this is to alter capacitor values. As seen in table 3 and figure 18, this does not cause an issue since there is no overlap between the frequency bands above the -3dB point. Since the filter is acting as 6 band-pass filters simultaneously, it has limitations on how close the different bands can be. In figure 18 we see that the 4th and 5th frequency band (4th and 5th string) are almost overlapping, which means the gain would have to be decreased before the Q can be increased to close the gap on the other bands. Although the gaps are large in some places, it does not pose a problem since the spacing between the strings increase

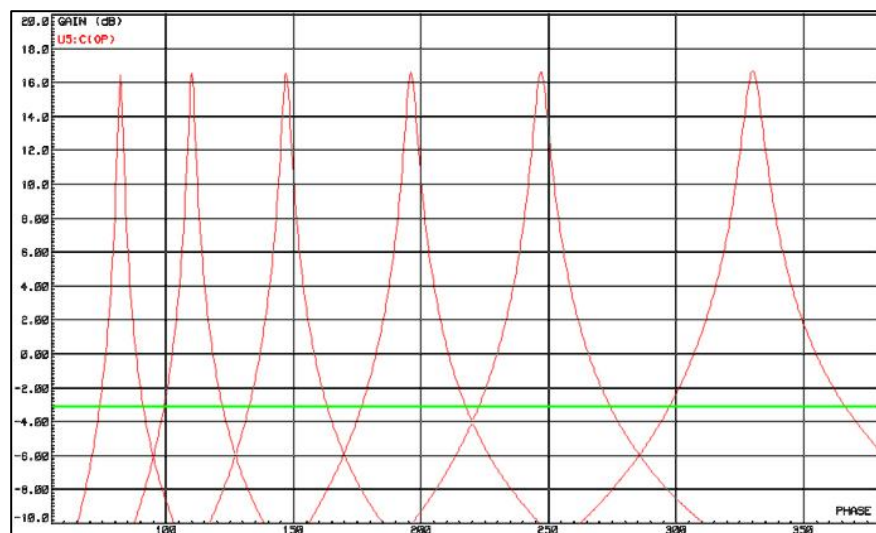


Figure 18 - Frequency response plot of entire filter. -3dB is represented by the green line. The strings are represented with increasing frequency (EADGB_e). See table 3 for numerical values not visible on the plot.

as the frequency goes up (28Hz between E and A, 83Hz between B and e in standard tuning). Since the strings do not output the same amplitude, it also allows for variations in amplitude without overlapping.

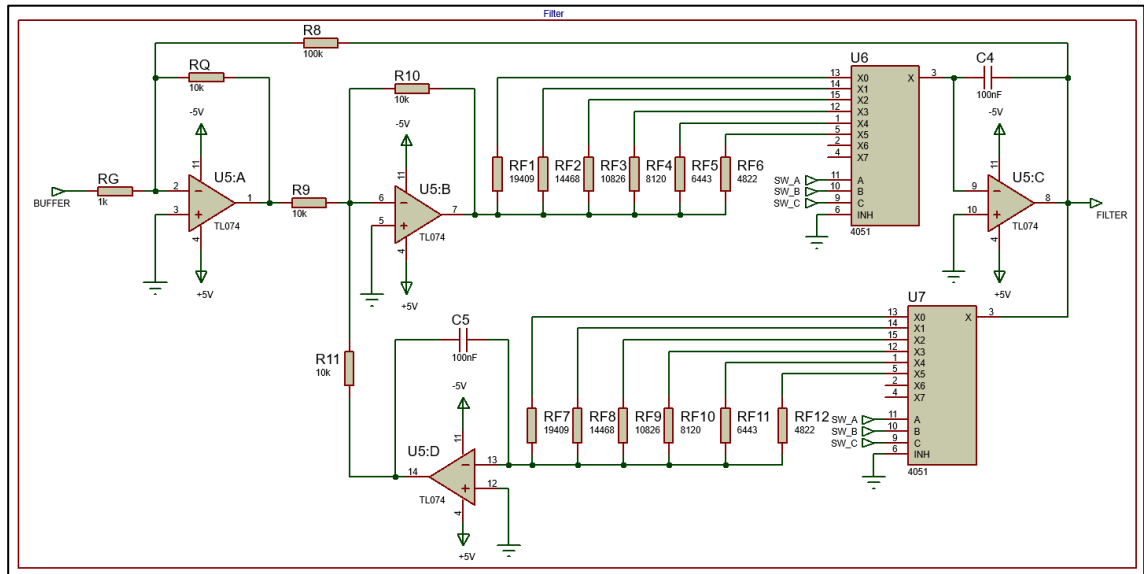


Figure 19 - Detail of state-variable filter.

In figure 19, the filter (without the buffer and amplification stage) is shown with its two multiplexers. The multiplexers and its resistor bank would normally be regarded as one resistor in series, which is where this design differs slightly from Horowitz and Hill's design. The multiplexers are essentially 3:8 analogue decoders, where a 3-bit signal from the PIC selects which connection to make internally in the chip. The user selects the channel manually, by means of cycling through a menu with a pushbutton.

3.5.4 Amplifier and Schmitt-trigger

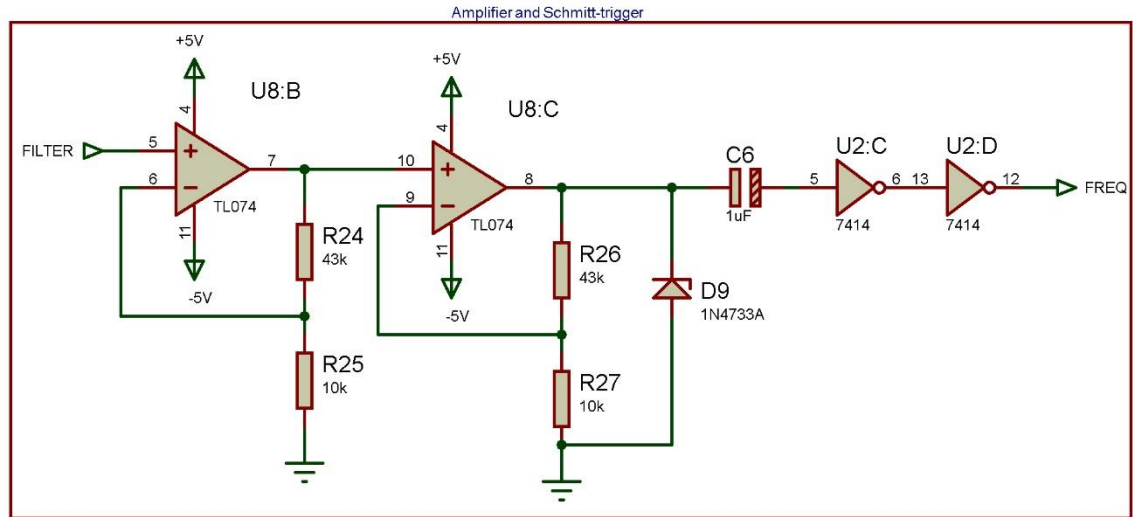


Figure 20 - Amplifier and Schmitt-trigger circuit.

In figure 20 we see that both amplifiers are on the same chip as the input buffer. This leaves an op-amp available for eventual further improvements with the same hardware. The filtered signal goes through 2 amplification stages where the total gain K_{total} was given by:

$$K = \left(\frac{R_{24}}{R_{25}} \right) + 1 \therefore K = \left(\frac{43k\Omega}{10k\Omega} \right) + 1 = 5.3 \therefore K_{total} = K^2 = 28.09$$

To calculate the amplified voltage of a guitar signal, we estimate a guitar output voltage of 150mVpkpk. In figure 4 the D-string outputs 192mVpkpk, where other strings ranges from 70mV-250mV, making 150mVpkpk an approximate median value. The first amplification stage has a gain of 6.6. Therefore, $6.6 \times 28.09 = 187.26$ giving a total gain of $187 \pm 10\%$ when allowing for component tolerances. Multiplying the 150mV output voltage by 187 gives a final output voltage of 28.05V, which is extremely high. For the purposes of this application however, it is positive. The amplifiers are intentionally overdriven to provide a near-square wave which is then cleaned up by the Schmitt-trigger. This is due to the supply voltage level of the op-amps, which ultimately limits their output voltage. Previous designs only had a single stage with a gain of 5.3. This was successful, but unstable at lower string voltages. The gain was then increased by adding the second stage after some experimentation to verify its usefulness and functionality.

After the signal has been sufficiently amplified, it passes over a 5.1V Zener diode to eliminate the negative part of the signal and protect the PIC from voltage spikes in the filter system. A DC-blocking capacitor then eliminates any DC bias while optimising

the signal for the Schmitt-trigger by allowing for cleaner transients. The decision to add a Zener diode and a series-capacitor was made after some difficulty obtaining a clean Schmitt-triggered output. In figure 21, the signal looks very noisy on the input side, having lost all its negative components (before the Zener was added). This output signal led to frequency readings of several kHz, as the PIC registered each false transition visible in figure 21. Speculation led to doubting the Schmitt-triggers capability of handling negative signals. The datasheet for the SN74ACT Schmitt-trigger was subsequently investigated, where the speculations were confirmed: Specifications stated a minimum input voltage of 0V[35, p. 2].

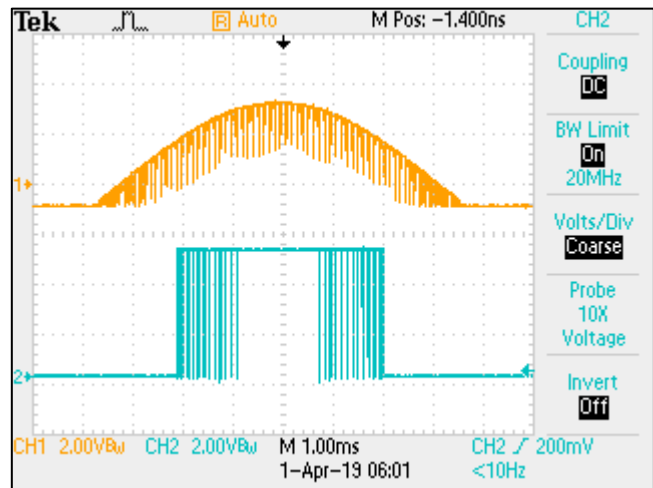


Figure 21 - Irregular Schmitt-trigger output caused by negative signal components. Input is CH1 (yellow trace).

3.5.5 Stepper Motor Circuitry

The choice of the L297 and L298 was not only dictated by their compatibility, but also for the quality of their documentation. A straightforward connection diagram was provided within the L297 datasheet, which was partially mirrored in the schematic.

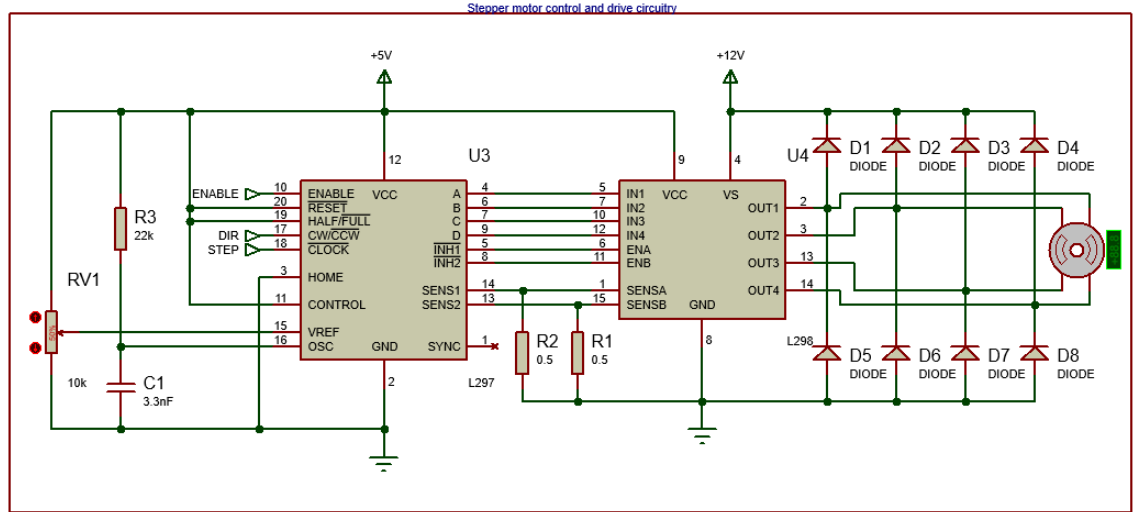


Figure 22 - Stepper motor control and drive circuitry detail

To the left in figure 22 the L297 controller chip with its connections is shown. The potentiometer RV1 exists to adjust the maximum motor current output from the L298 chip on the right. By providing a reference voltage via the potentiometer, a comparator compares the voltage over the resistors R1 and R2 to VREF. When the supply current reaches the programmed threshold, an internal SR-latch resets and disconnects the output drive until a new pulse is provided by the external oscillator circuit[29, p. 4]. The VREF voltage level is calculated with the equation $V_{ref} = \frac{I_{max}}{R_1}$ where $R_1 = R_2$ and I_{max} is the maximum current specification of the motor. Therefore:

$$V_{ref} = \frac{350mA}{0.5\Omega} = 700mV$$

The datasheet for the stepper motor specifies a maximum phase current of 175mA in unipolar mode (see appendix 3 for further info). When in bipolar mode, the current passes through 2 phase windings, doubling the resistance. Note that this motor is not the desired motor for the project, as it is estimated to be over 20 years old and lacks the desired performance ratio of modern motors. The oscillation frequency for the chopper circuit is given by the equation:

$$f = \frac{1}{0.69RC} \therefore f = \frac{1}{0.69 \times 22k\Omega \times 3.3nF} \cong 20kHz$$

The capacitor and resistor values are listed in a standard circuit example in the datasheet, and the constant 0.69 is specified in the same datasheet[29, p. 3,8]. The

purpose of the chopper circuit is to provide an oscillation for the switch-mode amplifiers within the L298, thereby decreasing power consumption[36]. This offers a high degree of efficiency, since the on-resistance is very low and the off-resistance is very high. This ripple in supply voltage is acceptable due to the high inductance of the motor itself smoothing out the current curve, which is why the torque does not vary much and the motor runs smoothly[9, p. 43].

The VREF potential divider will remain a potentiometer while the project is at the breadboard stage. The potential divider values would be calculated by the equation

$$\frac{V_{out}}{V_{in}} = \frac{R_2}{R_1 + R_2} \therefore \frac{700mV}{5V} = \frac{7\Omega}{50\Omega}$$

where R1 is given by $R_1 = (R_1 + R_2) - R_2$ [20, p. 7]. This results in resistor values of 7Ω and 43Ω , which were multiplied by 100 to lower current draw. The closest E24 resistor values are 680Ω and $4.3k\Omega$ as per IEC standard 60063:2015[37], [38].

3.5.6 Power Supply

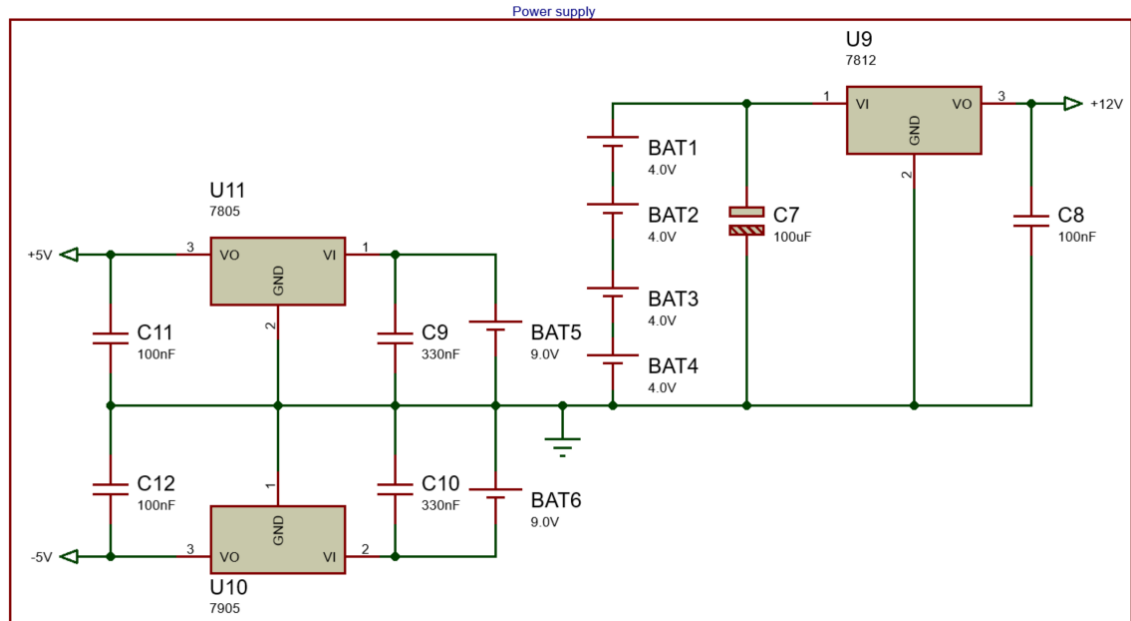


Figure 23 - Power supply detail

To accomplish the implementation of 3 different voltage rails, 3 different voltage regulators are needed. The 5V rails only draw an absolute maximum of around 150mA, while the 12V rail only draws a maximum of around 100mA with a generous over-estimation. This would dictate the choice of some very small voltage regulators, but availability once again dictated the choice of physical components. The 7805, 7905 and 7812 voltage regulators were all within specification and were used in the prototype. Both 5V regulators were installed as per the datasheet instructions, with 330nF and 100nF capacitors on the input and output respectively. Since the 7812 was driving a motor circuit, more noise and power supply ripple was expected. A 100uF electrolytic capacitor was installed to cope with this. The centre node of the two 9V batteries was the reference to ground, to be able to supply both the positive and negative regulator. The 7812-regulator supply referenced its ground from the same node. Experiments were done with isolating these two circuits completely. This did not work, as the stepper motor controller required the same ground as the driver to function because of the ground reference of the sense resistors.

3.5.7 PIC and LCD

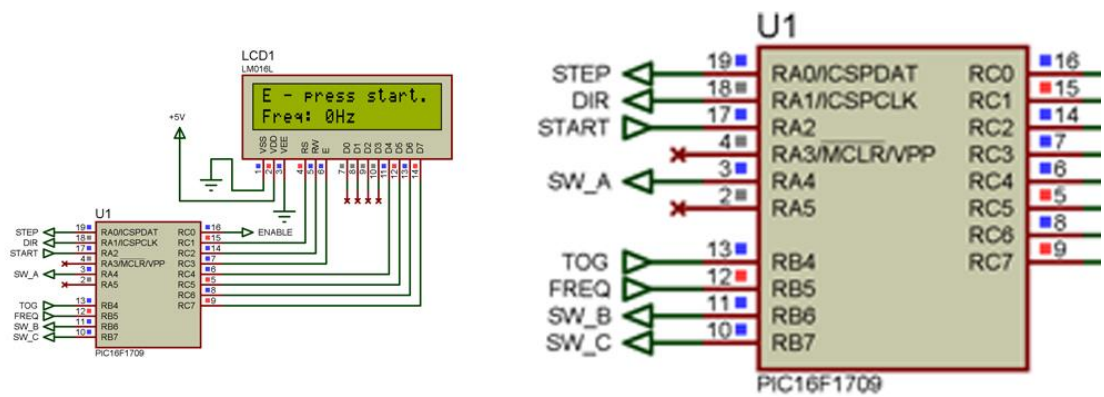


Figure 24 - Detail of PIC and LCD schematic

In figure 24, the PIC16F1709 is pictured with the connections to the circuit and to the LCD. Pin C0 (16) serves as the enable pin for the stepper motor controller (not shown in zoomed view). All connections between the different subsystems of the tuner were made using input/output connectors in Proteus to reduce the amount of wires and enhance readability. As seen, all but two pins on the PIC are utilised, which leaves room for future expansion and/or space for an RS232 port for debugging the software. The pinout table (table 4) provides more detail. Pin 1 and 20 are not visible in figure 24 due to Proteus hiding power pins by default. The LCD pins were set up on port C to keep interrupt possibilities open on port A and B for the buttons. Pins were also assigned to prevent the crossing wires in the schematic.

Pin #	Channel	Name	Function
1	-	Vdd	PIC +5V supply
2	RA5	-	NC
3	RA4	SW_A	Multiplexer bit 1 (MSB)
4	RA3	-	NC
5	RC5	D5	LCD data pin 5
6	RC4	D4	LCD data pin 4
7	RC3	E	LCD enable
8	RC6	D6	LCD data pin 6
9	RC7	D7	LCD data pin 7
10	RB7	SW_C	Multiplexer bit 3 (LSB)
11	RB6	SW_B	Multiplexer bit 2
12	RB5	FREQ	Guitar signal input
13	RB4	TOG	Toggle button input
14	RC2	RW	LCD R/W pin
15	RC1	RS	LCD RS
16	RC0	ENABLE	Stepper enable
17	RA2	START	Start button input
18	RA1	DIR	Stepper direction control
19	RA0	STEP	PWM output for stepper
20	-	Vss	PIC ground

Table 4 - PIC pinout table

3.5.8 Button Debouncing Circuits

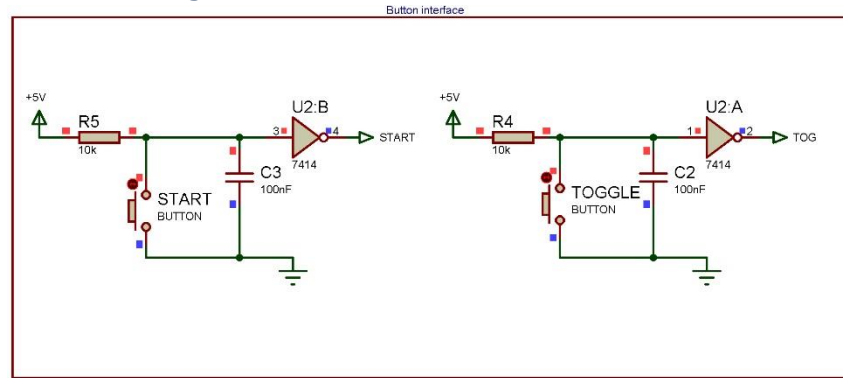


Figure 25 - Button layout and debouncing circuitry

The occurrence of switch bounce is due to the imperfect nature of switches and human movement, where a single switch press can cause the switch contacts to bounce when closing[39, p. 196]. To mitigate this, hardware or software switch-debouncing techniques can be deployed. This design is based on a combination of the two, with a Schmitt-trigger circuit removing the voltage bounce spikes and outputting a sharp transition from 0-5V. These Schmitt-triggers are also called inverters, since they have inverted outputs.

+5V was put in series with a 10k Ω resistor to limit the current to 500 μ A since the Schmitt trigger only sinks 1 μ A, while the remaining current from the supply goes to ground. A 100nF capacitor was placed after the switch in parallel with ground to block noise and create a sharp discharge curve upon button press, providing the Schmitt trigger with a clean signal. The software aspect of the debouncing simply involved incorporating a delay of 50ms to disregard any signals in the first 50ms after detecting the first edge. This proved to be a robust solution with dependable button I/O.

3.6 Software

The software running on the PIC was exclusively written in C with the CCS C compiler. This was mainly due to the authors familiarity with the language, but also because of the compiler's deep integration with Proteus, allowing compiling directly through the Proteus application framework. The main source file contains approximately 250 lines of code, excluding whitespace. As all filtering and normally CPU-intensive tasks are done by the state-variable filter, the PIC can be run at 8MHz, which is still more than enough for this application. 8MHz was also chosen due to the setup of the timers, as 8MHz provided a resolution of 4 μ s per increment and a large overflow of 262ms for the lower frequency range of input signals. A relatively slow PWM signal was needed to drive the stepper motor, and a faster clock would limit the minimum PWM frequency.

3.6.1 The Capture/Compare/PWM Module

After passing through the filter, amplifiers and Schmitt triggers, the signal has been converted to a good quality square wave for digital processing. As explained earlier, measuring frequency in the time domain is accomplished by either timing a known quantity of pulses, or measuring the time between each discrete pulse. The latter has the disadvantage of being more taxing on the system since the software must measure and process the time between each individual pulse. It also proved unstable during experiments, where the frequency readout would “jump” randomly due to the extremely high sensitivity of the 4 μ s timer. The PIC16F1709 is equipped with 2 Capture/Compare/PWM (CCP) modules, all of which are utilised in this project. The CCP1 module is used in capture mode for pitch detection purposes, while the CCP2 module is used to generate PWM pulses for the stepper motor.

In capture mode, the input waits for a predetermined set of pulses. On the last pulse, the software moves the value in the timer1 register into a variable called CCP_1. The module was first set up to capture 4 rising edges on all strings with a timer resolution of 2 μ s, as this proved stable during simulation. After building the prototype and testing it with a guitar, the E and A strings were extremely unstable, where the timer would overflow and output false values due to the lower frequency of these strings. This was mitigated by lowering the resolution of timer1 from incrementing every 2 μ s to every 4 μ s, thereby increasing the overflow period. Since the remaining strings (DGB_e) oscillate at a higher frequency, the capture module was set to capture every 16th edge of the signal, slowing the program down somewhat. This proved useful, as the program

now captures an average period time of 16 pulses which slowed the program down while avoiding triggering the interrupt too often. Averaging 16 pulses also increased accuracy. Accuracy verification was accomplished by directly feeding an 82Hz (E) square wave from a signal generator into the PIC and observing the value printed to the LCD. The input frequency was verified with an oscilloscope beforehand to exclude accuracy problems in the signal generator digital readout.

The captured timer value in CCP_1 has no real purpose if not referenced to a specific point in time. An interrupt is therefore triggered when the last pulse is detected to reset the timer, which results in CCP_1 containing the time value of X registered pulses. This interrupt also handles a small arithmetic sequence, which first divides the time of the 16/4 pulses to the time of 1 complete period before inverting the resulting value to obtain the frequency in Hertz.

To summarise, the operational sequence of the CCP module in capture mode consists of 4 steps for this pitch detection application. The first capture is to be discarded, as no time reference point is yet to be established. The signal is also somewhat erratic during the first milliseconds of string vibration before the string settles into its fundamental oscillation frequency. These 4 steps (after 16/4 captured pulses) are as follows:

1. Copy timer1 into CCP_1 variable
2. Reset timer1
3. Divide CCP_1 by 4 if selected string is D, G, B or e
4. Invert variable to obtain frequency value

Note that the inversion is obtained by $\frac{1M}{Time}$ due to the raw time value being captured in μs . A possible bug in the CCS C compiler is present due to the division ratios not conforming to logic. Represented graphically in figure 26, a division of 16 seems logical to obtain the period of a single cycle. However, this results in incorrect output readings. This also applied to the 4-pulse trigger of the E and A strings but was left unresolved as the software was functional.

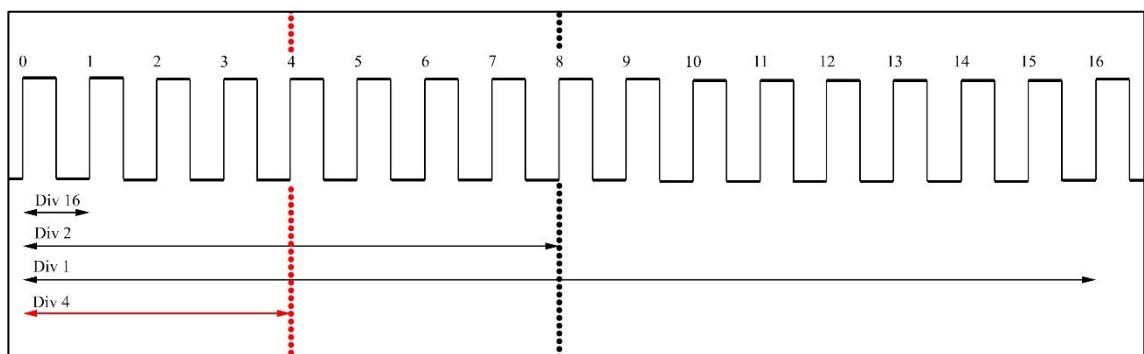


Figure 26 - Possible CCP division discrepancy

3.6.2 Boot Sequence

When the system is powered on, the PIC enables and sets up certain modules as instructed by the software. This boot sequence can be seen in figure 27 in flowchart form. The CCP module for the pitch detection is set up first, while the CCP2 module for PWM is declared in the header file. Stepper motor step frequency varies with each motor, while 300Hz was successful with all tested motors. The interrupts are set up next. This program uses a total of 3 interrupts with each button having its own interrupt function, plus a dedicated interrupt routine for the CCP1 unit. Timer1 is then started before the CCP1 variable is reset for first use. The LCD is then initialised, and the stepper motor is disabled for safety. The stepper motor is disabled at multiple locations throughout the code, even when not strictly necessary. This creates a failsafe where the user can be confident the motor will not start inadvertently. A delay of 10ms is incorporated at the end of the boot sequence to provide time for the LCD to initialise completely.

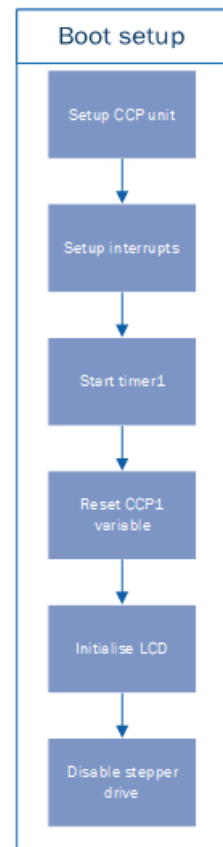


Figure 27 - Boot sequence

3.6.3 The Tuning Algorithm

After the boot sequence is completed, the program executes *main*. The only purpose of *main* is to provide an entry point for the program, and to serve as a host for the switch/case routine selecting the setup parameters for the string in question. A while-loop with the Boolean argument TRUE ensures the program goes into an infinite loop.

The switch/case routine is responsible for setting the variables needed in the tune function. The 3 multiplexer digits are selected by the pins defined as SW_A, SW_B and SW_C in code. In figure 28, channel 000 is selected to activate port X0 on the 4051 multiplexers, closing the circuit on the respective resistor connected to it. After the multiplexer port is selected, the capture module is setup with the correct parameters for each string.

The correct division ratio is then set. See appendix 4 for datasheet instructions on CCP modes.

Since the cases were handled by a single variable (*mode*) and a single button, the *modeSel* (mode select) variable was added to provide a way out of the tune cycle when the user presses the toggle button. The *modeSel* variable follows the *mode* variable, and the tuner function while loop's argument is the equality of these variables. When the *mode* variable changes by pushing the toggle button, it is not equal to *modeSel* anymore, voiding the loop arguments and exiting. *inTune* sets the desired frequency for the selected string, while the *string* character variable gets assigned the appropriate letter for use in the tune function. When setup is complete for the selected string, the program calls the tune function. The tune function is designed as a dynamic function

receiving necessary variables from other functions instead of static design which would take up a substantial amount of code space. Like the filter design, early software revisions had entire tuning functions dedicated to each string, making the program over 600 lines long. Figure 29 provides an overview of the tuning logic, where the top

```
177 while(TRUE)
178 {
179     switch (mode)
180     {
181         case 1:
182             output_low(SW_A); //Selects port X0
183             output_low(SW_B); //
184             output_low(SW_C); //
185
186             setup_ccp1(6); //Setup capture after 4 rising edges
187             divider=1; //Sets division ratio for capture
188             modeSel=1; //While-loop argument
189             inTune=82; //Sets desired string frequency
190             string='E'; //Labels string for LCD printing
191
192             tune(); //Call tune function
193
194             break;
```

Figure 28 - Switch/case code excerpt

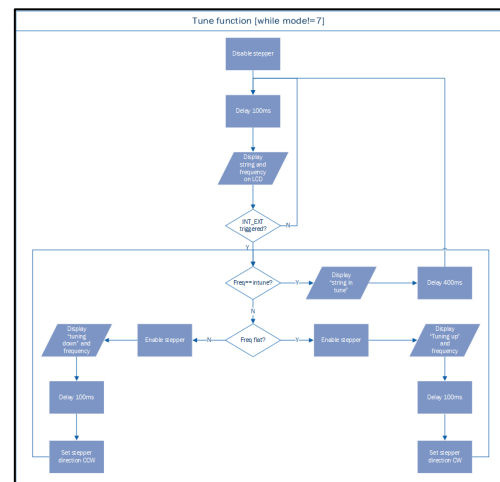


Figure 29 - Tune function overview. Refer to appendix 6 for more details.

section is the pre-tune waiting area. The two “legs” represents CCW and CW tuning modes. The system does not output the state of tuning before the start button has been depressed. Before this point only the frequency in Hertz is represented on the LCD. When the program enters the tune functions, it first disables the stepper motor. It then displays the frequency on the screen while awaiting start button input. When the button is pressed, it passes through 2 checkpoints. First it attempts to verify if the string is already in tune. If no action is needed, the LCD will refresh with the tuning info for 1 second before incrementing the *mode* variable and exiting the while loop. The program then goes to the corresponding *mode* value and the selection process starts over. If a string is out of tune, the corresponding while loop will be triggered, and the stepper motor is enabled in the appropriate direction. It will continue to rotate until the frequency number reaches the desired value, before it outputs the tuning value and continues as described above. The tuning function in its entirety can be seen in figure 30.

```

80 void tune() //Main tuning algorithm
81 {
82     toggle=0; //Synchronising toggle variable with stepper output state
83     output_low(stepEnable); //
84
85     while(toggle==0&&mode==modeSel) //Pre-tuning state. Wait for user input.
86     {
87         printf(lcd_putc, "\n%c-frequency: %ld\nPress start", string, freq);
88         delay_ms(50);
89     }
90
91     while(toggle==1&&freq==inTune&&mode==modeSel) //String in tune. Jumps to next string
92     {
93         output_low(stepEnable);
94         for(i=0;i<10;i++) //Displays the string frequency with a 10Hz refresh rate for one second before moving to next string
95         {
96             printf(lcd_putc, "\n%c-string in tune\nFrequency: %ld", string, freq);
97             delay_ms(100);
98         }
99         mode++; //Breaks the while loop and goes to next switch/case in main
100     }
101
102     while(toggle==1&&freq!=inTune&&mode==modeSel) //Tuning cycle
103     {
104         while(freq>inTune&&mode==modeSel) //Sharp
105         {
106             printf(lcd_putc, "\n%c-string sharp.\nF=%ld Tuning", string, freq);
107             output_high(stepDir); //CCW
108             output_high(stepEnable);
109             delay_ms(50);
110         }
111
112         while(freq<inTune&&mode==modeSel) //Flat
113         {
114             printf(lcd_putc, "\n%c-string flat.\nF=%ld Tuning", string, freq);
115             output_low(stepDir); //CW
116             output_high(stepEnable);
117             delay_ms(50);
118         }
119     }
120 }
121 }
122

```

Figure 30 - Tuning function

3.6.4 Manual Motor Control

Having manual control over the motors rotational direction provides better usability.

This allows the unit to be used as a pure machine head winder to facilitate rapid string changes. Mode 7 of the switch/case in *main* sets the flag

manualMode, which in turn alters the functionality of the

external interrupt routine. This allows the start button to be used

in the same manner as the toggle button, with an incrementing

variable allowing the four different manual sub-modes to be

activated. The manual function

consists of clockwise and

counter-clockwise modes with a

wait-routine before each to

inform the user about which way

the motor will rotate upon button

press. The start-button sequence is now *clockwise-off-counter-clockwise-off*. The user

can at any point in the code press the toggle button to exit the function. Likewise, the

start button can stop the motor at any point in the program.

```
257 case 7:
258
259     modeSel=7;
260     manualMode=1;
261     manual();
262     manualMode=0;
263
264 break;
```

Figure 32 - Manual mode setup

```
52 if(manualMode==1)
53 {
54     x++; //Increments X for selecting function of manual mode
55     if(x>4)
56     {
57         x=1; //Resets variable when out of bounds
58     }
59 }
```

Figure 31 - Altered interrupt routine for manual tuning mode

3.7 Circuit Description – Breadboard Prototype

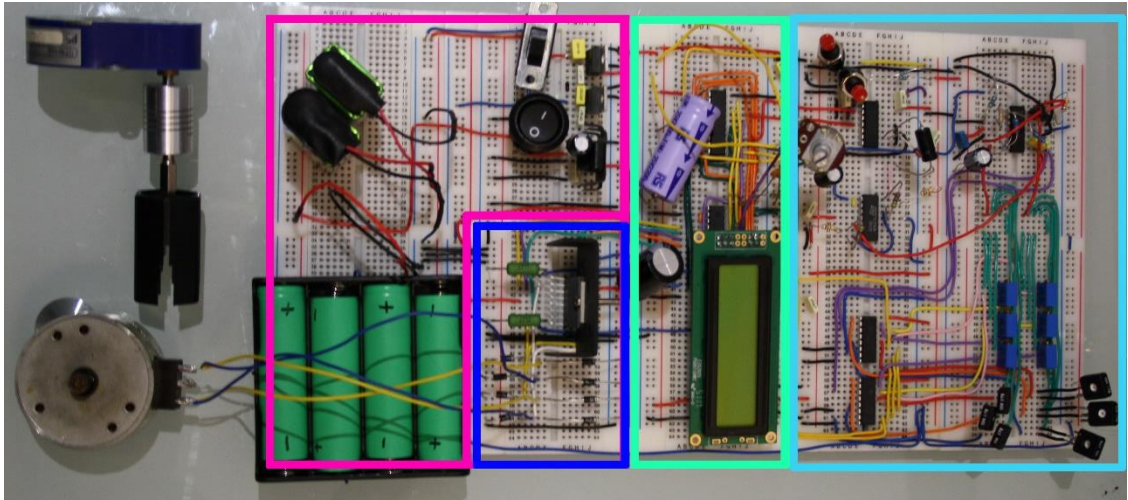


Figure 33 - Prototype breadboard. Pink section: Power supply. Blue section: Stepper motor drive circuitry. Green section: Digital components. Turquoise section: Filter.

Stepper motor, gearbox and coupling is shown on the far left.

Efforts were made to mirror the schematic and breadboard by choosing obsolete/old components in the simulation since they were in physical stock. The entire prototype was constructed on 4 breadboards, excluding 1 additional breadboard for supporting the batteries and to serve as an isolated unregulated rail. A mistake was made early on where the +5V regulated rail was connected to the unregulated +9V from the battery. The isolated breadboard was included as an attempt to prevent future errors. As seen in figure 33, the circuit can be divided into 4 main sections, each section built and tested as an individual stage on its own breadboard before being fused with the greater circuit. The specific stepper motor and gearbox visible on the far left was the only combination in stock exhibiting the needed torque to turn machine heads. The coupling system was custom designed using a drill-bit machine head winder and a flexible aluminium shaft coupling with 4 setscrews.

3.7.1 Power Supply

The circuit was operated by 5V, -5V and 12VDC. Each rail was regulated by its own voltage converter, with a lithium battery pack powering the motor and 2 9-volt (PP3) batteries powering the rest of the circuit. The lithium battery pack consisted of 4 4V cells in series, providing an unregulated voltage of 16V. The two 9V batteries were also connected in series, providing an unregulated voltage of 18V. This supply was centre tapped to provide a reference to ground for all the rails. The 16V-battery ground for the motor was connected directly to this centre-tap. According to the datasheet, the two 9V PP3 provides a continuous runtime of approximately 11 hours @ 50mA before the end-voltage of 5.4V depending on the age and temperature of the battery cells[40]. The 18650-size lithium-ion battery cells were load tested with an automatic battery tester to verify their functionality (figure 35). With a maximum measured current draw of 70mA and an approximate battery capacity of 2200mAh, the battery would keep the motor running continuously under full load for over 30 hours. This bulky battery with a large capacity is not necessary for a final product. It is also impractical having 2 different battery chemistries and form factors. This was however practical during the prototype stage, where each battery could be switched and individually monitored. The extra capacity also meant less time was spent recharging the batteries. Short wire legs were soldered directly on the battery bank switches to provide mechanical stability.

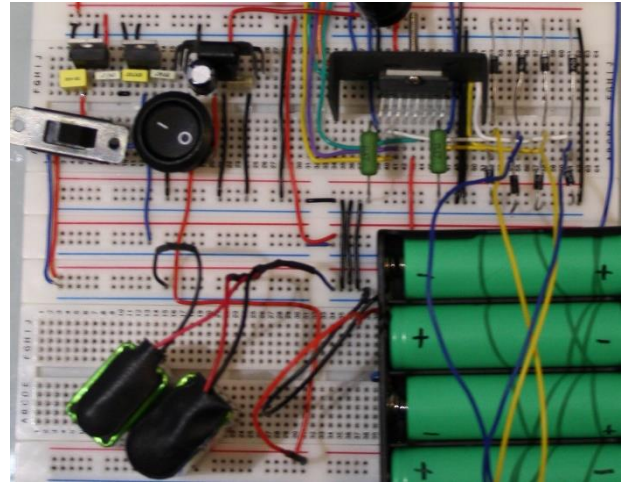


Figure 34 - Close-up of breadboard power supply. The voltage regulators and battery switches can be seen on the top left. The top right quadrant contains the stepper motor drive circuitry.



Figure 35 - Lithium battery load test

3.7.2 Filter

The main body of the filter consisted of a single quad-op-amp IC with passive components surrounding it. Resistors for setting the frequency response were laid out in 2 banks of 6 resistors, 2 for each string. The 10k Ω potentiometers were 1-turn devices. This made tuning them difficult, as the filter relied on very precise resistance values. The 20k Ω potentiometers were 10-turn devices. Since they are selected by the multiplexer system, the inputs can all be connected together, as the current follows the path of least resistance (the selected channel).

The previously described analogue multiplexer lowered the component count drastically, as only 4 op-amps were needed in the filter alone (excluding buffering and amplification) instead of the previous 24, where each string had its separate filter.

The frequency response resistors were all either 10k Ω or 20k Ω potentiometers, allowing for a finely tuned filter. Since board space is scarce and PCB “real-estate” is expensive, the potentiometer solution was less than ideal due to their bulkiness. This could potentially be solved in the future by utilising surface-mount devices (SMD) of a higher quality to decrease the overall spatial footprint. In figure 36, the multiplexer ICs are visible in the bottom right quadrant. The top half contains the filter in its entirety, with the potentiometers on the right. The bottom left quadrant houses the hex Schmitt trigger and the 2nd quad op-amp for buffering and amplification.

The multiplexers were operated directly by the PIC, with a 3-digit binary code as their output selection instruction. The functional diagram for the port selection can be seen in figure 37.

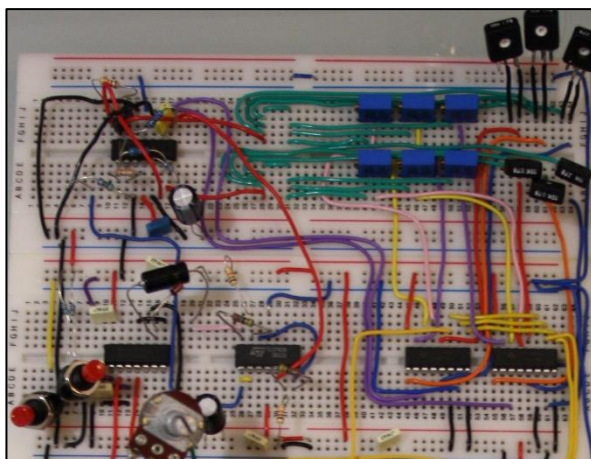


Figure 37 - Filter section of breadboard

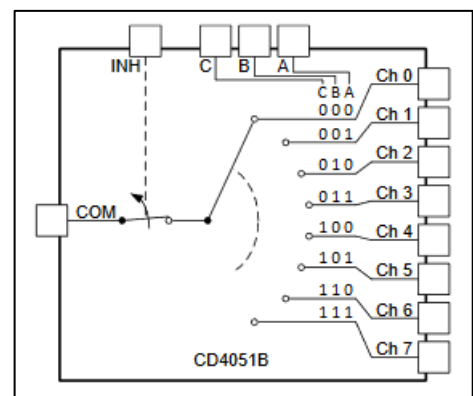


Figure 36 - Multiplexer truth table (functional diagram). Retrieved:

<https://www.ti.com/lit/ds/symlink/cd4051b.pdf>

3.7.3 Stepper Motor

The stepper motor circuit as described in the datasheet was mirrored on the breadboard and circuit diagram. In figure 38, the stepper motor driver IC and its heatsink can be seen in the centre. Below it are the two sense resistors, coupled directly to the ground rail. On the right-hand side, the 8 flyback diodes for reducing back-EMF can be seen. These diodes discharge the voltage generated from the collapsing magnetic field in the inductor (motor winding) when the inductor is de-energised[41, p. 880]. A pair of diodes is installed for each coil, as the coils polarity changes depending on rotational direction. The yellow and blue phase output wires to the motor itself can be seen between the diodes. The large capacitor on the top right mitigates noise and ripple caused by both the stepper driver and the back-EMF from the motor. On the left side the four phase connections from the stepper controller can be seen. These wires provide information on which coil winding the stepper motor should energise, similar to the H-bridge MOSFET design mentioned in chapter 2.9.2, figure 10. An attempt was made to mimic the schematic layout on the breadboard as it made component placement and troubleshooting simpler.

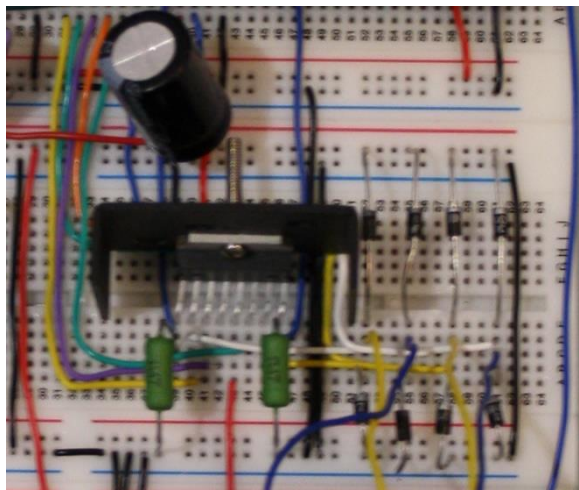


Figure 38 - Stepper motor connection layout on breadboard

The stepper motor installed in the circuit is a very old stepper motor from McLennan servo supplies. The gearbox is unmarked, so an attempt to measure the gear reduction by means of manually counting the turns was done. This concluded with an approximate 125:1 reduction. According to the datasheet, the stepper motor exhibits a torque value of 2.8Ncm, which results in an output force of 350Ncm.

3.7.4 Component list

Below is a table listing all components used on the breadboard. Standard passives like resistors and capacitors were omitted as the final choice of these components requires a finalised design, preferably a PCB design. Supporting components such as heatsinks, breadboards, PP3 batteries and wires were also omitted, along with the gearbox for lack of information. Components no longer manufactured are marked as *Obsolete*.

Component	Manufacturer	Manufacturer's part number	Supplier	Supplier's stock number	Quantity	EA Price
Stepper motor	McLennan	9904-112-31004	<i>Obsolete</i>	<i>Obsolete</i>	1	-
+5V voltage regulator	Texas Instruments	UA7805CKCS	DigiKey	296-13996-5-ND	1	£0.61
-5V voltage regulator	Texas Instruments	UA7905CKCS	DigiKey	296-17077-ND	1	£0.64
Stepper motor controller	STMicroelectronics	L297/1	DigiKey	497-1394-5-ND	1	£6.60
Stepper motor driver	STMicroelectronics	L298N	DigiKey	497-1395-5-ND	1	£3.82
Microcontroller	Microchip	PIC16F1709-I/P	Farnell	2422887	1	£1.22
LCD	Midas	MC21605A6W-SPR-V2	Farnell	2674228	1	£7.61
Op-amp	Texas Instruments	TL074CN	DigiKey	296-1777-5-ND	2	£0.51
Batteries and charger	Samsung, LiitoKala	INR18650-25R, Lii500	eBay	302628396099	1	£37.95
Shaft coupling	-	-	eBay	262814471837	1	£4.99
Peg winder	Alice	-	eBay	111708418693	1	£5.99
Schmitt-trigger	Texas Instruments	SN74ACT14N	Farnell	1749916	1	£0.34

Table 5 - Component list

4.0 Results

As the project went through 3 stages of testing, a measure of success was the correlation of results from these three stages. The circuit was first designed and simulated exclusively in SPICE software, where a prediction can be made on the real-life characteristics of the circuit. After achieving a stable simulation, the circuit was built on breadboards to record the frequency readout on the LCD. The last stage of testing was the comparison between the physical oscilloscope (as opposed to the simulated one) and the readout on the LCD. The stepper motor motion was a secondary priority, as this was seen as an uncomplex software routine where the direction of the motor depends on the state of an integer (frequency).

4.1 Filtering Results

In figure 39, the buffered input waveform (CH1, yellow trace) is compared to the output of the last amplifier stage. The strong harmonics of the D-string are clearly visible, along with some noise components most likely due to the long guitar cable acting as an aerial receiver for lab background noise. This noise was mainly 50Hz mains hum, but also contained some high frequency (>50kHz) components of unknown origin. This explains the frequency being displayed as 440Hz, which coincidentally is the standard tuning reference for tuning instruments. Note that the oscilloscope's internal filtering is enabled, blocking frequencies over 20MHz. On the right-hand side of figure 39, the measured values for both waveforms are displayed. The input signal has a base amplitude of 260mVpkpk, while the output is amplified to 5.88V. Due to the filter's high selectivity, the resulting sinewave on CH2 is very clean and is a suitable signal for the Schmitt-trigger.

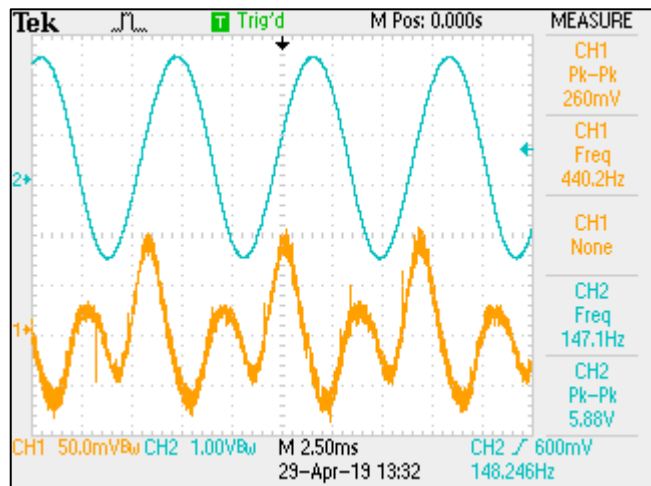


Figure 39 - Comparison between untreated guitar input waveform and filter output of D-string.

The same comparison can be seen on the software oscilloscope in figure 40. The appearance of the input waveform is heavily modified by Proteus' sampling algorithm, where curvy sinewaves are represented as triangular peaks. This audio sample was recorded with Audacity and saved as a 32bit float WAV-file. This filetype was the only accepted filetype for simulation purposes.

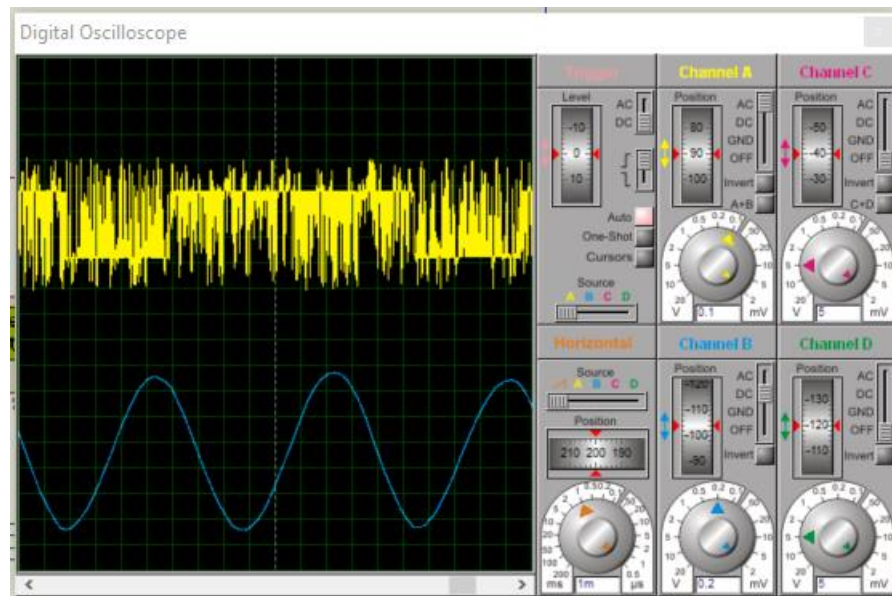


Figure 40 - Comparison between input WAV file and output waveform on a simulated oscilloscope in Proteus.

The resulting waveform must be Schmitt-triggered to create a readable signal for the PIC. A comparison between the guitar input waveform and the final, triggered output can be seen in figure 41.

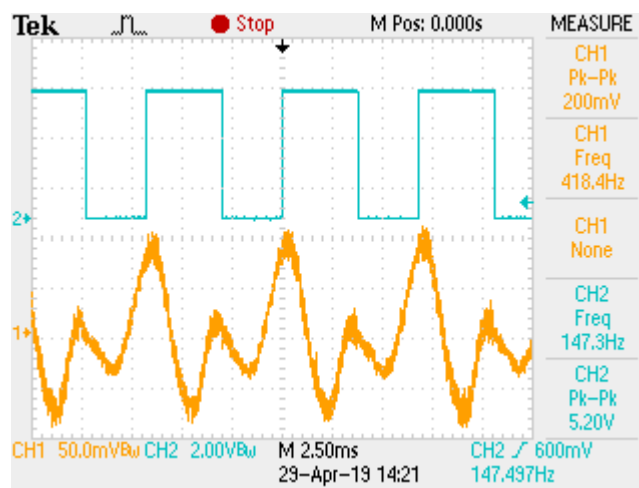


Figure 41 - Comparison of input waveform and final result of filtering and Schmitt triggering.

4.2 Stepper Motor Motion

As covered in the software section, the stepper motors rotational direction is a direct consequence of the frequency integer value. When the prototype was fully assembled, a dry test was conducted without turning the machine heads. The motor did not perform entirely as expected. Since the string vibration decays very quickly, the motor has limited time to turn the machine heads and achieve the correct frequency. The software was also flawed in that it stopped the tuning process immediately when the desired frequency was detected. Successful tests were rare. While the proof of concept is intact, the software requires much refinement before a stable system release. Thoughts and ideas on improving the stepper software can be found in the future improvements section.

5.0 Planning and Project Management

A Gantt-chart was created early on in the project process to establish a project outline and to provide an overview over task prioritisation. This plan was not followed perfectly but functioned more as a checklist and timekeeper where important deadlines could be monitored. The general order of operations was however followed accurately as the critical path had to be chronological to ensure project success.

The project's critical path consists of the main tasks necessary for completion. Each of these tasks has a set of subtasks. These subtasks are required for the critical path to progress but can be interchanged within their own group. This allowed for great flexibility, as a project this size is a substantial undertaking for one person. The project was a design exercise as much as a planning exercise, where tens of elements needed to come together to reach the desired outcome.

The main tasks (critical path) are listed below:

1. Preliminary research
2. Component specification (requirement analysis)
3. CAD design and programming
4. Testing and software debugging
5. Final build and prototype realisation (also incorporated report writing)

Milestones on the end of each tasks marked their completion to provide a general overview of project completion. Some tasks were not completed, as their complexity and/or time requirements were grossly underestimated. This most notably affected the design of the printed circuit board (PCB), which should have been started at a much earlier stage because of its complexity and manufacturing time. It was eventually dropped completely.

Prototype work testing and debugging continued far longer than planned. This was due to new ideas and solutions to problems being discovered when the report was written. The amount of research conducted for this project was substantial and when piecing it together in the report, new approaches were discovered that improved the outcome of the prototype. The Gantt-chart was a helpful tool overall but was not critical for a project this size where a simple checklist would have sufficed. It can be argued that a Gantt-chart is an utmost necessity when the project relies on the coordination of multiple people completing different tasks. See appendix 5 for full Gantt-chart.

6.0 Discussion of Future Improvements

Many discoveries were done during the course of the project, relating to both technical and practical aspects. These discoveries all pertained to things that could or should have been done differently

6.1 Pitch Detection

As seen in chapter 4, the filtering circuit and its external components created a powerful signal converter capable of detecting the fundamental frequency of extremely noisy and imperfect signals. Although the outcome was successful in terms of signal processing, hindsight revealed that a DSP technique would have been much better suited for this application. DSP was never considered upon project initiation, due to its perceived complexity and the authors unfamiliarity with its concepts. If DSP had been utilised, the component count and complexity of the filter could have been dramatically reduced. The filter itself would only attenuate the high-frequency noise present in almost any analogue signal, while a technique such as a combination of FFTs and autocorrelation would determine the fundamental frequency while relying entirely on software. Another change would be to use floats instead of integers, to tune the strings even closer to their fundamental frequency (the E-string is in reality 82.4Hz, not 82).

6.2 Motor Selection

While a stepper motor possesses a high degree of accuracy since it can move in discreet steps, it is not entirely necessary for this application. Stepper motors were primarily chosen for their holding torque. A massive disadvantage to stepper motors are their complex support circuitry, where a controller, driver and microcontroller are necessary for their operation. While stepper motors are strong, a gear reduction was required to output the necessary torque to turn the machine heads. It was later discovered that this gear reduction produced “holding torque” of its own, because of the nature of gear reduction. It was difficult enough to turn the output shaft regardless of the motors on-state. A better solution would therefore be to use a permanent magnet DC motor with a gear reduction. These can offer much higher speeds and torque parameters (depending on gearing) while not suffering from the same complexity issues as stepper motors. They are also cheaper and usually smaller than their stepper motor counterparts.

6.3 Power Supply and Batteries

While the batteries currently employed in the prototype offer versatility and control, they are cumbersome and non-ideal for this application. The system would benefit from a physically smaller Lithium-ion battery pack for its high energy density while boost converters could be used to reach the necessary voltages. A 3.3V system voltage would be desirable, as it would decrease the physical size of the battery pack substantially. This would require a complete revamp of the system architecture, as all components currently installed are designed to work with 5V supply voltages.

The necessity of the -5V rail is disputed. Although no experiments were executed to verify this, it can be argued that biasing the input signal to a DC level would reduce the overall complexity of the system by rendering the -5V regulator and circuitry redundant. Although probably unnecessary, it still provided a good filtering exercise.

6.4 Reducing Pin Count

The LCD in this system is run in 4-bit parallel mode. By selecting a different protocol (SPI, RS232, I2C), the pin count could have been reduced even further. This could have paved the way for the selection of a smaller PIC, or the same PIC could be used and its internal op-amps could be used for filtering purposes, possibly saving board space since one less dedicated op-amp was used.

6.5 Feature Expansion

The software running on the PIC could have been much improved. It could have been made more stable, and more features could have been added. Support for multiple instruments would not be possible, as it would require a redesign of the filter circuit. However, as previously mentioned, a more powerful DSP solution would have circumvented this. Sharp, flat and in-tune LEDs would have been a good addition to the system, as would a piezoelectric sounder indicating tune state.

6.6 Miscellaneous Improvements

Backlighting the LCD would have created a more attractive product. Viewing angle could also have been improved, at the expense of a more expensive LCD unit. Surface mounted components would also have been preferable if the project ever reached the PCB design stage.

7.0 Conclusion

The main design requirement for this system was to achieve an accuracy of $\pm 3\text{Hz}$ when detecting the pitch and tuning the string. Pitch-detection wise this was accomplished, as the system directly mirrored the oscilloscope frequency reading in a laboratory setting. This was accomplished through extensive background research, where insights into analogue filtering yielded the successful state-variable filter design. Existing efforts have used different techniques relying mostly on digital circuits and DSP techniques, while at least one technique utilised a fully analogue approach with frequency-to-voltage conversion.

Relying on analogue filtering for extracting the fundamental frequency of a musical instrument is viable, but impractical. If designed correctly and tuned properly, some impressive filtering characteristics can be achieved, at the disadvantage of component count and physical circuit complexity.

A stepper motor was utilised to turn the machine heads of the guitar headstock. This worked but was impractical due to the complexity of the supporting circuitry and their low RPM capabilities. A better solution would be a permanent magnet DC motor capable of higher RPMs with a reduction gearbox, since the gear reduction offers some holding power of its own.

A power supply consisting of two battery banks with differing chemistry and form factor was used during the experimental prototype stage. For future designs, it is recommended to use a smaller battery pack while lowering the overall system voltage by using low-voltage (3V3) components where possible.

The proposed design was confirmed to be working but unstable, as the string decay was too fast, and the motor was too slow to tune the string before it decayed completely.

8.0 Reflection

This project marks the highlight of my undergraduate degree, because I could finally research and build exactly what I wanted to. As this project had many different tasks needed for satisfactory completion, it quickly became evident that a form of discipline was needed to not focus too much on the sections I found the most interesting. The prototyping stage was the most intellectually stimulating, as seeing an envisioned concept take physical shape is not only motivating, but a lot of fun. This taught me a lot about handling large undertakings like dissertations and other large assignments.

The most prominent lesson is to never underestimate a time-frame. Even though lecturers, supervisors, parents, siblings and my girlfriend urged me to start early, I got off to a later start than desirable. This made the project less fun than it could have been. Another lesson was the necessity and importance of jumping in the deep end no matter how daunting it may be. This dissertation is arguably one of the biggest undertakings I have ever attempted to complete, and I was dreading starting it for weeks for the fear of not knowing where to begin. When I finally started however, it was quite fun, and I wish I started earlier. I will take these lessons with me when I continue on to do a master's degree.

As for the more academic lessons, I have learned the importance of a good reference manager and academic search engine. Finding sources for seemingly obvious facts can be incredibly tedious if these two things are missing. My main barrier when writing academic content is the general tediousness of doing research, because a vast majority of all papers and books hold very little information of actual interest and/or usability pertaining to the task at hand. After finally settling on a good reference manager, the workload seemed to vanish, as the dissertation wrote itself in some respects. If I ever write another dissertation, which I hopefully will, I will dedicate a lot more time to gather sources and data before starting the writing process. Research-as-you-go is a very tempting approach to take, despite its many pitfalls.

The final lesson I will take with me is the incredible feeling I got when finalising something as large as this assignment by writing this chapter. This will motivate me to take on even larger projects in the future.

9.0 References

- [1] R. B. C. Leite, S. A. Mota-Rolim, and C. M. T. Queiroz, “Music Proficiency and Quantification of Absolute Pitch: A Large-Scale Study among Brazilian Musicians,” *Front. Neurosci.*, vol. 10, p. 447, 2016.
- [2] A. Seither-Preisler *et al.*, “Tone Sequences With Conflicting Fundamental Pitch and Timbre Changes Are Heard Differently by Musicians and Nonmusicians,” *J. Exp. Psychol. Hum. Percept. Perform.*, vol. 33, no. 3, pp. 743–751, 2007.
- [3] N. H. Fletcher, “The nonlinear physics of musical instruments.” 1999.
- [4] R. Nave and Georgia State University, “Standing Waves on a String.” [Online]. Available: <http://hyperphysics.phy-astr.gsu.edu/hbase/Waves/string.html>. [Accessed: 19-Mar-2019].
- [5] J. G. Roederer, “The Psychophysics of Musical Perception,” *Music Educ. J.*, vol. 60, no. 6, pp. 20–30, Feb. 1974.
- [6] T. D. Rossing, *Springer handbook of acoustics*. Springer, 2007.
- [7] C. Shahnaz, W. P. Zhu, and M. O. Ahmad, “Pitch estimation based on a harmonic sinusoidal autocorrelation model and a time-domain matching scheme,” *IEEE Trans. Audio, Speech Lang. Process.*, 2012.
- [8] L. Rabiner, M. Cheng, A. Rosenberg, and C. McGonegal, “A comparative performance study of several pitch detection algorithms,” *IEEE Trans. Acoust.*, vol. 24, no. 5, pp. 399–418, Oct. 1976.
- [9] A. Hughes, *Electric motors and drives fundamentals, types and applications*, 4th ed. Amsterdam: Elsevier, 2013.
- [10] E. H. Wernick, *Electric Motor Handbook*. Maidenhead: McGraw-Hill Book Company, 1978.
- [11] T. Kenjo, *Electric Motors and their Controls*, 1st ed. Oxford: Oxford University Press, 1991.
- [12] A. Bocanegra, “Automatic Acoustic Guitar Tuner,” Massachusetts Institute of Technology, 2005.
- [13] K. Munakata and D. F. Horler, “Book reviews - Springer Handbook of Acoustics,” *J. Travel Tour. Mark.*, vol. 24, no. 4, pp. 321–325, 2008.
- [14] E. J. Candes and M. B. Wakin, “An Introduction To Compressive Sampling,” *IEEE Signal Process. Mag.*, vol. 25, no. 2, pp. 21–30, Mar. 2008.
- [15] B. A. Olshausen, “Aliasing - PSC 129 - Sensory Processes.” Redwood Center for

- Theoretical Neuroscience University of California, Berkeley, pp. 1–6, 2000.
- [16] C. Shahnaz, W.-P. Zhu, and M. O. Ahmad, “Pitch Estimation Based on a Harmonic Sinusoidal Autocorrelation Model and a Time-Domain Matching Scheme,” *IEEE Trans. Audio. Speech. Lang. Processing*, vol. 20, no. 1, pp. 322–335, Jan. 2012.
 - [17] I. Barbancho, L. J. Tardon, S. Sammartino, and A. M. Barbancho, “Inharmonicity-Based Method for the Automatic Generation of Guitar Tablature,” *IEEE Trans. Audio. Speech. Lang. Processing*, vol. 20, no. 6, pp. 1857–1868, Aug. 2012.
 - [18] M. French, “Pitch and Frequency.”
 - [19] Texas Instruments, “LM2907 / LM2917 Frequency to Voltage Converter.” [Online]. Available: <http://www.ti.com/lit/ds/symlink/lm2917-n.pdf>.
 - [20] P. Horowitz and W. Hill, *The Art Of Electronics*, 3rd ed. Cambridge University Press, 2016.
 - [21] Karki, “Learn the limitations of low-pass sallen-key filters,” *Electron. Des.*, vol. 47, no. 26, pp. 96–100, 1999.
 - [22] A. Wang, A. Nandi, and P. Liew, “‘THE MECHANIX’ GUITAR TUNER,” *Cornell university*, 2008. [Online]. Available: https://people.ece.cornell.edu/land/courses/ece4760/FinalProjects/s2008/an94_fw45_pcl25/an94_fw45_pcl25/index.htm. [Accessed: 24-Apr-2019].
 - [23] “Roadie Automatic Guitar Tuner.” [Online]. Available: <https://www.roadietuner.com/>. [Accessed: 18-Apr-2018].
 - [24] C. Button, “Guitar Tuner,” *Electronics & Music Maker*, Jun-1981.
 - [25] Dr. Douglas W. Jones, “Stepping Motors Fundamentals | Microchip Technology Inc.,” *Microchip Appl. Note*, p. 22, 2004.
 - [26] P. L. Weaver, *Success in your project : a guide to student system development projects*. Prentice Hall Financial Times, 2004.
 - [27] “Research Onion,” *University of Derby*. [Online]. Available: <https://onion.derby.ac.uk/>. [Accessed: 30-Apr-2019].
 - [28] C. Barnham, “Quantitative and Qualitative Research: Perceptual Foundations,” *Int. J. Mark. Res.*, vol. 57, no. 6, pp. 837–854, Nov. 2015.
 - [29] STMicroelectronics, “L297 stepper motor controller,” *Datasheet*, 2001. [Online]. Available: <https://www.st.com/resource/en/datasheet/l297.pdf>.
 - [30] STMicroelectronics, “L298 Dual Full Bridge Driver,” *Datasheet*, 2000. [Online].

- Available: <https://www.st.com/resource/en/datasheet/l298.pdf>.
- [31] Midas Displays, “MC21605B6W-SPR-V2 16x2 Liquid Crystal Display,” *Datasheet*, 2016. [Online]. Available: http://www.farnell.com/datasheets/2173671.pdf?_ga=2.176142177.2134104247.1555411597-1800469538.1553279453.
- [32] University of Iowa, “Alphanumeric LCD Displays,” *Embedded Systems and Systems Software lecture notes*. [Online]. Available: http://user.engineering.uiowa.edu/~ece_036/Lecture/Lecture9.pdf.
- [33] “SPST, SPDT, and DPDT Switches Demystified,” *Information Sheet*. [Online]. Available: http://musicfromouterspace.com/analogsynth_new/ELECTRONICS/pdf/switches_demystified_assembly.pdf.
- [34] Texas Instruments, “TL07xx Low-Noise JFET-Input Operational Amplifiers,” *Datasheet*, 2017. [Online]. Available: <http://www.ti.com/lit/ds/symlink/tl071a.pdf>.
- [35] Texas Instruments, “SN54ACT14, SN74ACT14 Hex Schmitt trigger inverter,” 2004. [Online]. Available: <http://www.ti.com/lit/ds/symlink/sn74act14.pdf>.
- [36] Applied Motion Products, “Introduction To Step Motors,” *Omega*. [Online]. Available: https://www.omega.co.uk/auto/pdf/REF_IntroStepMotors.pdf.
- [37] “Standard Resistor Values » Resistor Guide,” *resistorguide.com*. [Online]. Available: <http://www.resistorguide.com/resistor-values/>. [Accessed: 26-Apr-2019].
- [38] International Electrotechnical Commission, “IEC 60063:2015 | IEC Webstore,” *IEC*, 2015. [Online]. Available: <https://webstore.iec.ch/publication/22011>. [Accessed: 26-Apr-2019].
- [39] R. Toulson and T. Wilmshurst, *Fast and Effective Embedded Systems Design: Applying the ARM mbed*. Newnes, 2012.
- [40] GP Batteries, “GP1604A PP3 Battery,” *Datasheet*. [Online]. Available: <http://www.houseofbatteries.com/documents/GP1604A.pdf>.
- [41] R. J. Herrick, *DC/AC circuits and electronics : principles & applications*. Thomson/Delmar Learning, 2003.

10.0 Bibliography

- Sallen, R. P., & Key, E. L. (1955). A practical method of designing RC active filters. *IRE Transactions on Circuit Theory*, 2(1), 74–85.
<https://doi.org/10.1109/TCT.1955.6500159>
- Palmer, M. (2006). Using the CCP Module(s). Retrieved from
<http://ww1.microchip.com/downloads/en/AppNotes/00594B.PDF>
- Roy, S. K., Molla, M. K. I., Hirose, K., & Hasan, M. K. (2011). Harmonic modification and data adaptive filtering based approach to robust pitch estimation. *International Journal of Speech Technology*, 14(4), 339–349.
- Staudacher, M., Steixner, V., Griessner, A., & Zierhofer, C. (2016). Fast fundamental frequency determination via adaptive autocorrelation. *EURASIP Journal on Audio, Speech, and Music Processing*, 2016(1), 17.
- Ting, H.-W., & Wang, H.-Y. (2010). Improvement of stop-band attenuation for the Sallen-Key low-pass filter. In *2010 International Symposium on Next Generation Electronics* (pp. 158–161). IEEE.
- Marczyk, G. R., DeMatteo, D., & Festinger, D. (2005). *Essentials of research design and methodology*. John Wiley & Sons.
- Sourav, S., Murugan s, B., Ramakrishnan, M., Ramasamy, S., & Annasamy, B. (2016). Acoustic Guitar Tuner and Identification of Chords using LabVIEW. *Global Journal of Pure and Applied Mathematics*, 11, 1171–1178.
- Walker, J. (1998). Just two ICs can create an accurate guitar tuner. *Electronic Design* v.46, 21.
- Feldmann, H. (1997). History of the tuning fork. I: Invention of the tuning fork, its course in music and natural sciences. Pictures from the history of otorhinolaryngology, presented by instruments from the collection of the Ingolstadt German Medical History Museum. *Laryngo- Rhino- Otologie*, 76, 116–122.
- Lourde, M., & Saji, A. K. (n.d.). A Digital Guitar Tuner. Retrieved from
<https://arxiv.org/ftp/arxiv/papers/0912/0912.0745.pdf>
- Adams, C. (2005). Device and Method for Automatic Tuning of a String Instrument in Particular a Guitar. Retrieved from
<https://patents.google.com/patent/US20080006140A1/en>
- Lemme, H. E. W. (2009). The Secrets of Electric Guitar Pickups. Retrieved from
<http://www.jpbourgeois.org/guitar/pdf/Secrets.pdf>

12.1 Appendix 1 – Stepper Motor Circuit – First Attempt

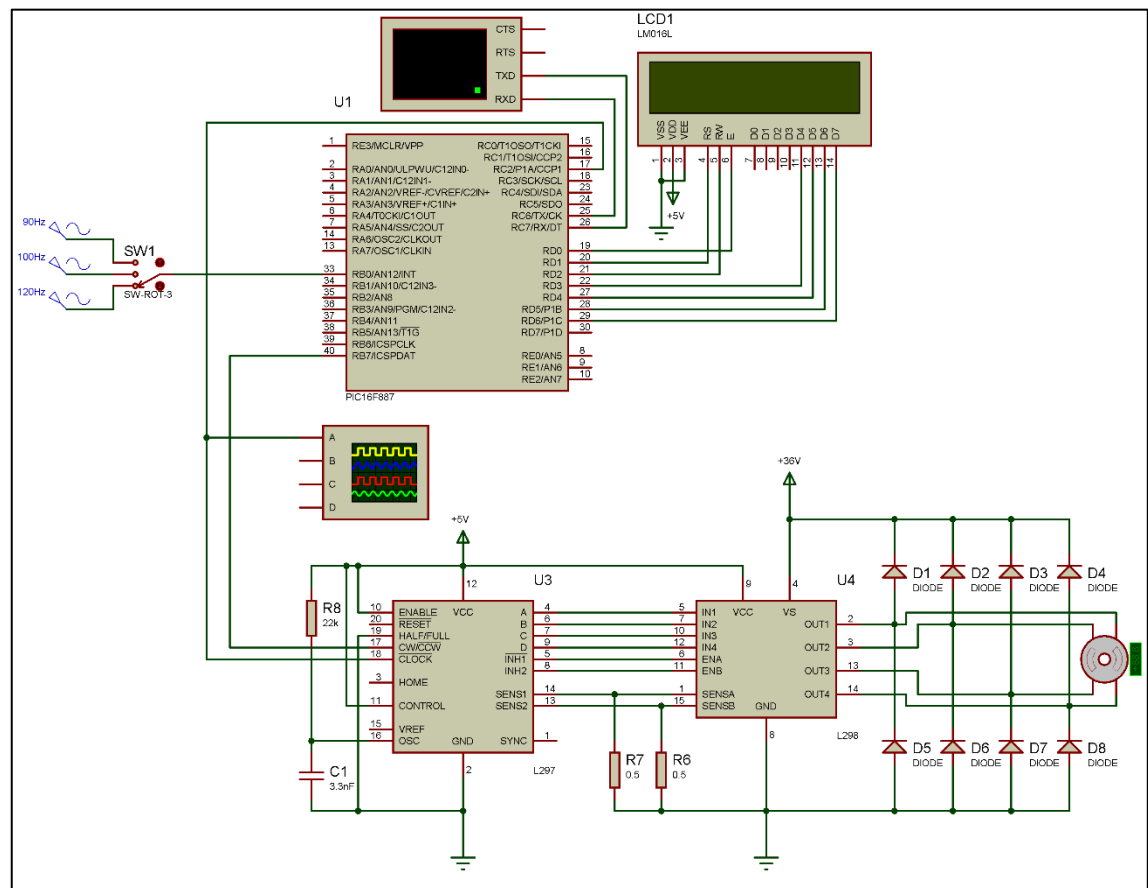


Figure 42 - Early circuit development schematic

This circuit was a testbed for designing a working stepper motor circuit and represents the first working result. The circuit is built from and directly reflects the design notes in the L297 datasheet[29]. An LCD was also incorporated with the standard connections required for driving an LCD in 7-pin mode.

A terminal and an oscilloscope were incorporated for testing purposes, and the +36V supply voltage is to be discarded as it makes no difference in this experimental application.

Microchip's PIC16F887 was chosen as its application methodology was familiar to the author. It was fed a combination of sinewaves, and later square waves to investigate its frequency measurement capabilities. The stepper motor is a generic bipolar stepper SPICE model in Proteus 8.2.

12.2 Appendix 2 – TL074 Op-amp characteristics

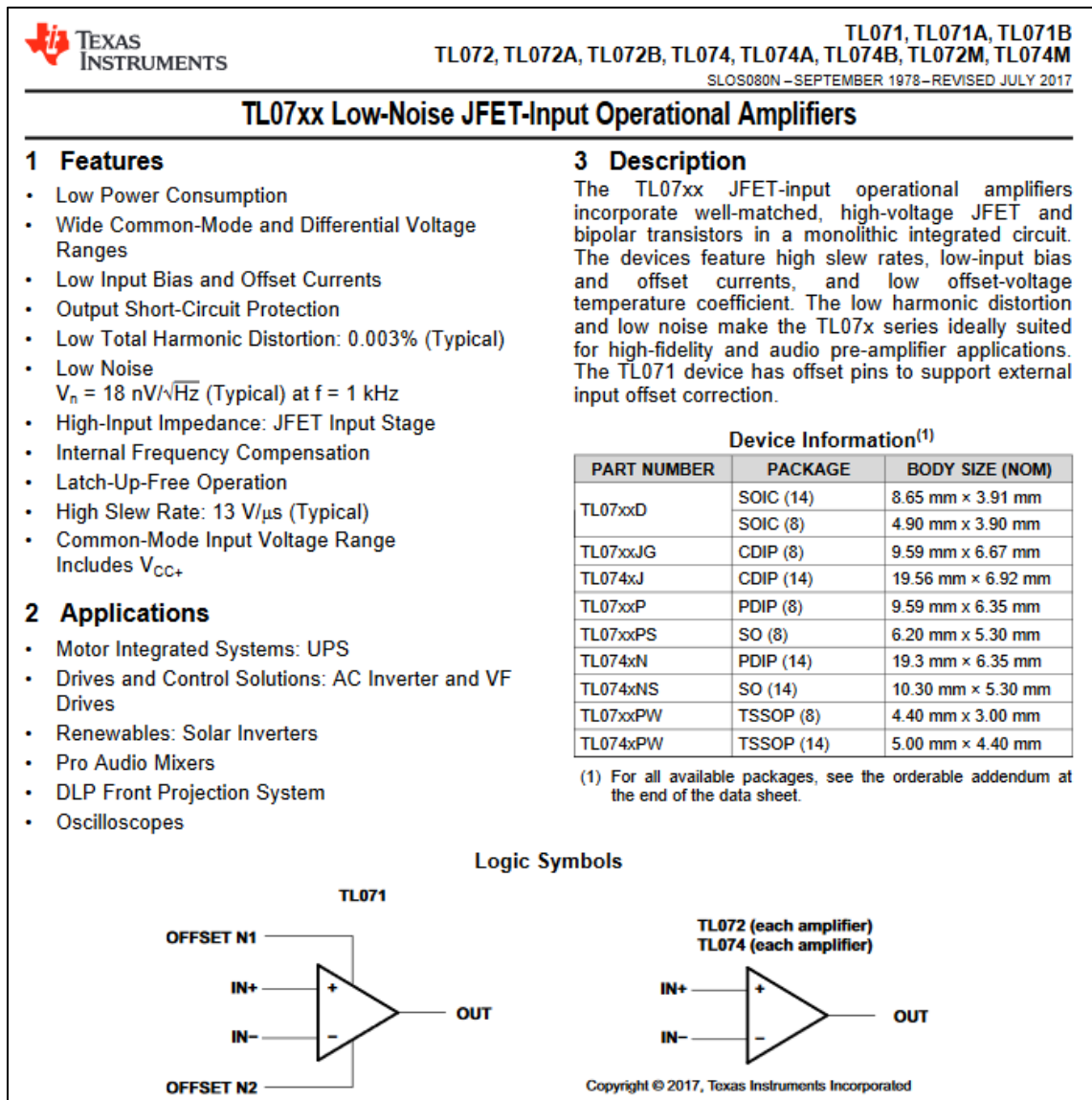


Figure 43 - TL074 datasheet front page. Retrieved: <http://www.ti.com/lit/ds/symlink/tl071a.pdf>

6.10 Electrical Characteristics: TL071C, TL072C, TL074C

 $V_{CC\pm} = \pm 15\text{ V}$ (unless otherwise noted)

PARAMETER		TEST CONDITIONS ⁽¹⁾ ⁽²⁾		MIN	TYP	MAX	UNIT
V _{IO}	Input offset voltage	V _O = 0 R _S = 50 Ω	T _A = 25°C		3	10	mV
			T _A = Full range			13	
α	Temperature coefficient of input offset voltage	V _O = 0 R _S = 50 Ω	T _A = Full range		18		μV/°C
I _{IO}	Input offset current	V _O = 0	T _A = 25°C		5	100	pA
			T _A = Full range			10	nA
I _{IB}	Input bias current ⁽³⁾	V _O = 0	T _A = 25°C		65	200	pA
			T _A = Full range			7	nA
V _{ICR}	Common-mode input voltage range	T _A = 25°C		±11	–12 to 15		V
V _{OM}	Maximum peak output voltage swing	R _L = 10 kΩ	T _A = 25°C	±12	±13.5		V
		R _L ≥ 10 kΩ	T _A = Full range	±12			
		R _L ≥ 2 kΩ		±10			
A _{VD}	Large-signal differential voltage amplification	V _O = ±10 V R _L ≥ 2 kΩ	T _A = 25°C	25	200		V/mV
			T _A = Full range	15			
B ₁	Unity-gain bandwidth	T _A = 25°C			3		MHz
r _i	Input resistance	T _A = 25°C			10 ¹²		Ω
CMRR	Common-mode rejection ratio	V _{IC} = V _{ICR(min)} V _O = 0 R _S = 50 Ω	T _A = 25°C	70	100		dB
k _{SVR}	Supply voltage rejection ratio (ΔV _{CC±} /ΔV _{IO})	V _{CC} = ±9 V to ±15 V V _O = 0 R _S = 50 Ω	T _A = 25°C	70	100		dB
I _{CC}	Supply current (each amplifier)	V _O = 0; no load	T _A = 25°C		1.4	2.5	mA
V _{O1} / V _{O2}	Crosstalk attenuation	A _{VD} = 100	T _A = 25°C		120		dB

(1) All characteristics are measured under open-loop conditions with zero common-mode voltage, unless otherwise specified.

(2) Full range is $T_A = 0^\circ\text{C to } 70^\circ\text{C}$.

(3) Input bias currents of an FET-input operational amplifier are normal junction reverse currents, which are temperature sensitive, as shown in Figure 1. Pulse techniques must be used that maintain the junction temperature as close to the ambient temperature as possible.

Figure 44 - TL074 electrical characteristics. Retrieved: <http://www.ti.com/lit/ds/symlink/tl071a.pdf>

12.3 Appendix 3 – Stepper motor datasheet

Instrument stepper motors

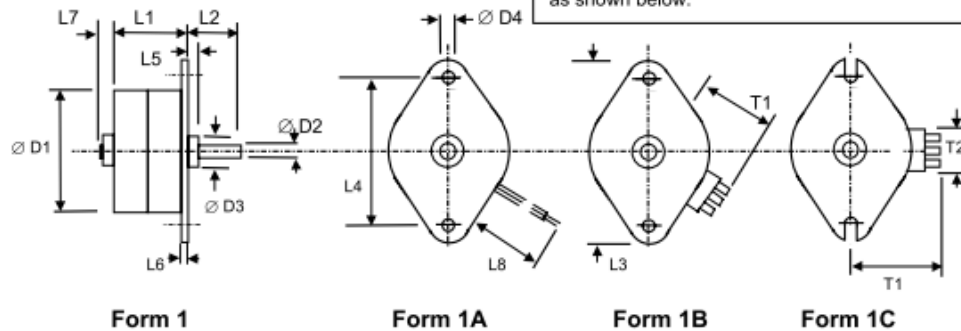
The instrument stepper motors are high quality permanent magnet types providing a 7.5 degree step angle

The motors may be specified factory fitted with a gearhead where increased torque and resolution are required at reduced operating speed. The programme has gained a world-wide reputation for high reliability and is widely used in industrial control and instrumentation drives. The miniature 1415 series, incorporates a low backlash precision gearhead to provide outstanding torque and resolution for a unit of such compact dimensions. A comprehensive programme of drives, power supplies and controllers have been developed to match the motor's characteristics to enable single or multi-axis systems to be readily constructed.



ID series stepper motor dimensions

ID series motors are either provided with leads (Form 1 A) or solder terminals (Forms 1B & 1C) as shown below:



motor type	form	dimensions mm													
		D1	D2	D3	D4	L1	L2	L3	L4	L5	L6	L7	L8	T1	T2
9904 112 series	1 A	35	2	10	3.2	21.5	8.8	50	42	1.2	0.8	2.2	175	N/A	N/A
9904 112 31001	1 A	51	3	10	3.5	25	8.2	70.5	60.2	1.5	1.0	3	175	N/A	N/A
9904 112 31101	1 A	51	3	10	3.5	25	8.2	70.5	60.2	1.5	1.0	3	175	N/A	N/A
9904 112 31004	1 B	51	3	10	3.5	25	8.2	70.5	60.2	1.5	1.0	3	N/A	38.5	18.3
9904 112 31104	1 B	51	3	10	3.5	25	8.2	70.5	60.2	1.5	1.0	3	N/A	38.5	18.3
9904 112 35 series	1 C	56.3	4	12	4.4	33.5	16	76.5	66	1.5	1.5	3.8	N/A	37.1	20

specification

motor type	step angle degrees	holding torque Ncm	rotor inertia Kgcm ²	resistance per phase ohms	current per phase amps	inductance per phase mH	number of leads or terminals	mass gms
9904 112 32001	7.5	1.0	0.0026	120	0.1	160	6	80
9904 112 32101	7.5	1.0	0.0026	21	0.22	30	6	80
9904 112 31004	7.5	2.8	0.011	65	0.175	100	6 terminals	170
9904 112 31001	7.5	3.0	0.011	62	0.190	160	8	200
9904 112 31104	7.5	3.2	0.011	11	0.4	16	6 terminals	170
9904 112 31101	7.5	3.4	0.011	17	..0.325	45	8	200
9904 112 31206 **	7.5	5.5	0.011	7	0.50	18	4	170
9904 112 35014	7.5	8.5	0.045	47	0.24	400	6 terminals	300
9904 112 35114	7.5	8.5	0.045	7.7	0.60	65	6 terminals	300
Precision Geared stepper motors								
1415 - 41:1 **	0.366	10	0.00045	12.5	0.25	5.5	4	47
1415 - 76:1 **	0.197	10	0.00045	12.5	0.25	5.5	4	50
1415 -141:1 **	0.106	10	0.00045	12.5	0.25	5.5	4	50

note** The above motors are, in general, 4 phase units suitable for use with uni-polar or bi-polar drive circuits. Motors identified by "*" however are only designed for bi-polar operation.

McLennan Servo Supplies Ltd. Tel: +44 (0)8707 700 700 www.mclennan.co.uk



Figure 45 - Stepper motor datasheet front page. Retrieved:

http://www.farnell.com/datasheets/330510.pdf?_ga=2.190572203.1083897800.1556570895-1800469538.1553279453

27.4 Register Definitions: CCP Control**REGISTER 27-1: CCPxCON: CCPx CONTROL REGISTER**

U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	DCxB<1:0>		CCPxM<3:0>			
bit 7							bit 0

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Reset

'1' = Bit is set

'0' = Bit is cleared

bit 7-6 **Unimplemented:** Read as '0'bit 5-4 **DCxB<1:0>:** PWM Duty Cycle Least Significant bitsCapture mode:

Unused

Compare mode:

Unused

PWM mode:

These bits are the two LSBs of the PWM duty cycle. The eight MSBs are found in CCPRxL.

bit 3-0 **CCPxM<3:0>:** CCPx Mode Select bits

11xx = PWM mode

1011 = Compare mode: Auto-conversion Trigger (sets CCPxIF bit), starts ADC conversion if TRIGSEL = CCPx (see [Register 20-3](#))

1010 = Compare mode: generate software interrupt only

1001 = Compare mode: clear output on compare match (set CCPxIF)

1000 = Compare mode: set output on compare match (set CCPxIF)

0111 = Capture mode: every 16th rising edge

0110 = Capture mode: every 4th rising edge

0101 = Capture mode: every rising edge

0100 = Capture mode: every falling edge

0011 = Reserved

0010 = Compare mode: toggle output on match

0001 = Reserved

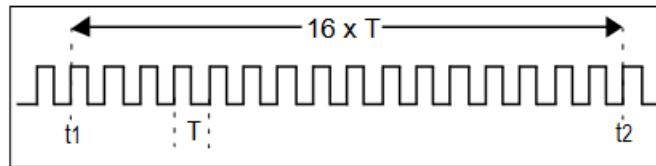
0000 = Capture/Compare/PWM off (resets CCPx module)

Figure 46 - CCP setup table. Retrieved: <http://ww1.microchip.com/downloads/en/DeviceDoc/40001729C.pdf>

Tips 'n Tricks

TIP #2 Measuring the Period of a Square Wave with Averaging

FIGURE 2-1: PERIOD MEASUREMENT



1. Configure control bits CCPxM3:CCPxM0 (CCPxCON<3:0>) to capture every 16th rising edge of the waveform.
2. Configure the Timer1 prescaler so Timer1 will run $16 T_{MAX}^{(1)}$ without overflowing.
3. Enable the CCP interrupt (CCPxIE bit).
4. When a CCP interrupt occurs:
 - a) Subtract saved captured time (t1) from captured time (t2) and store (use Timer1 interrupt flag as overflow indicator).
 - b) Save captured time (t2).
 - c) Clear Timer1 flag if set.
 - d) Shift value obtained in Step 4.a right four times to divide by 16 – this result is the period (T).

Note 1: T_{MAX} is the maximum pulse period that will occur.

Figure 47 - Microchip tips and tricks: CCP period measurement.

Retrieved:

<http://ww1.microchip.com/downloads/en/devicedoc/41214a.pdf>

12.5 Appendix 5 – Gantt Chart

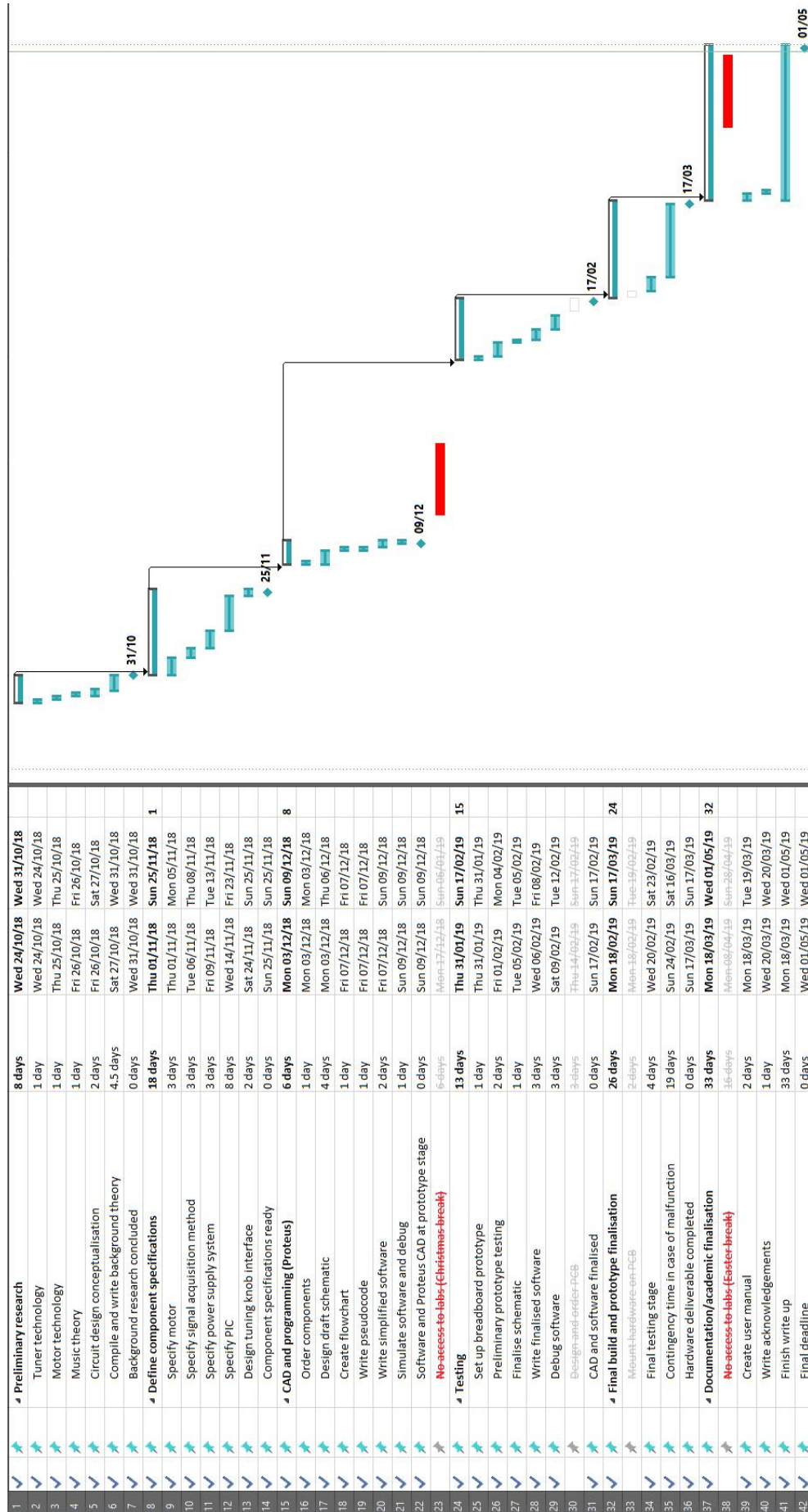


Figure 48 - Gantt chart, full view

12.6 Appendix 6 – Flowcharts

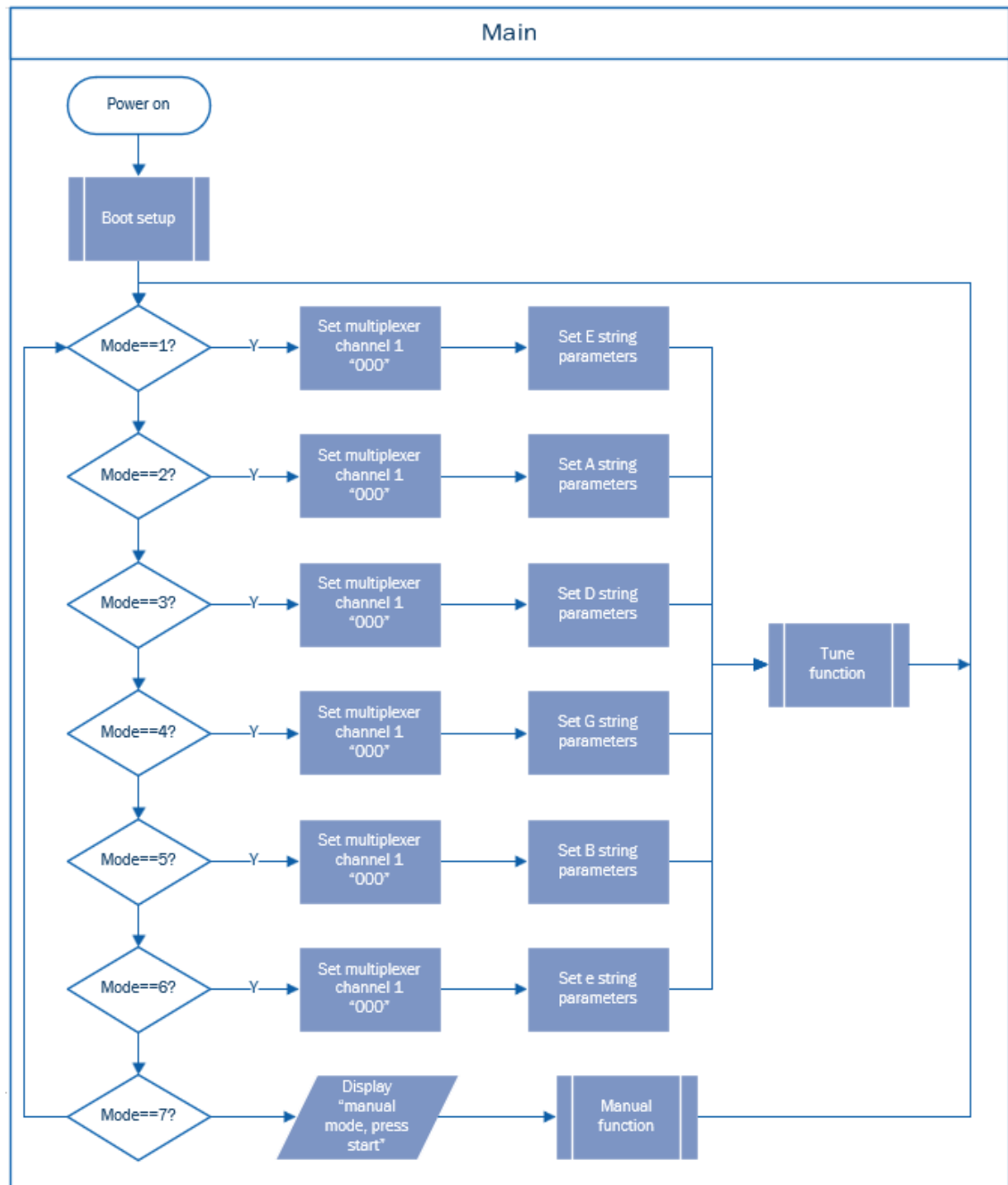


Figure 49 - Flowchart for main function

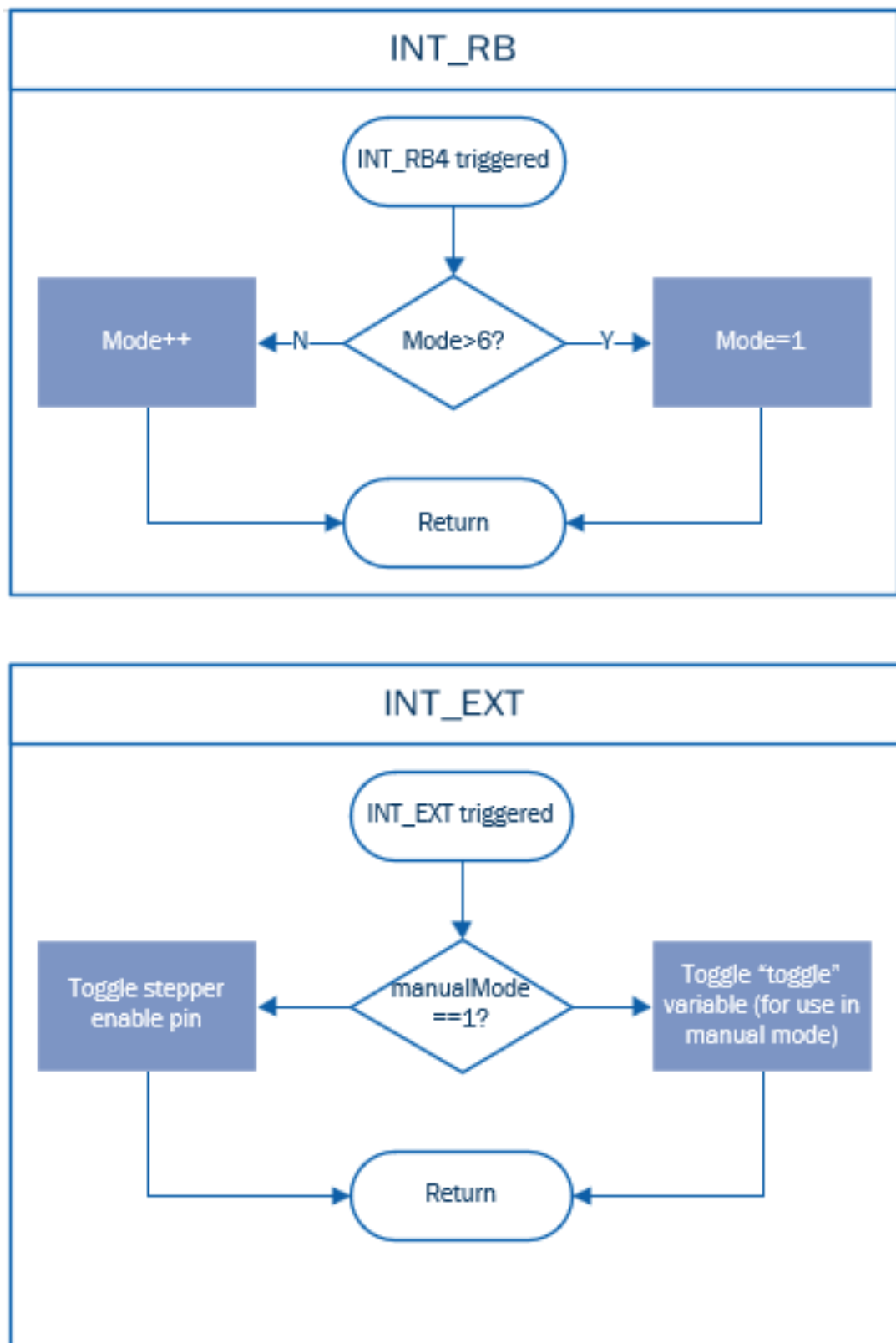


Figure 50 - Interrupt routine flowcharts

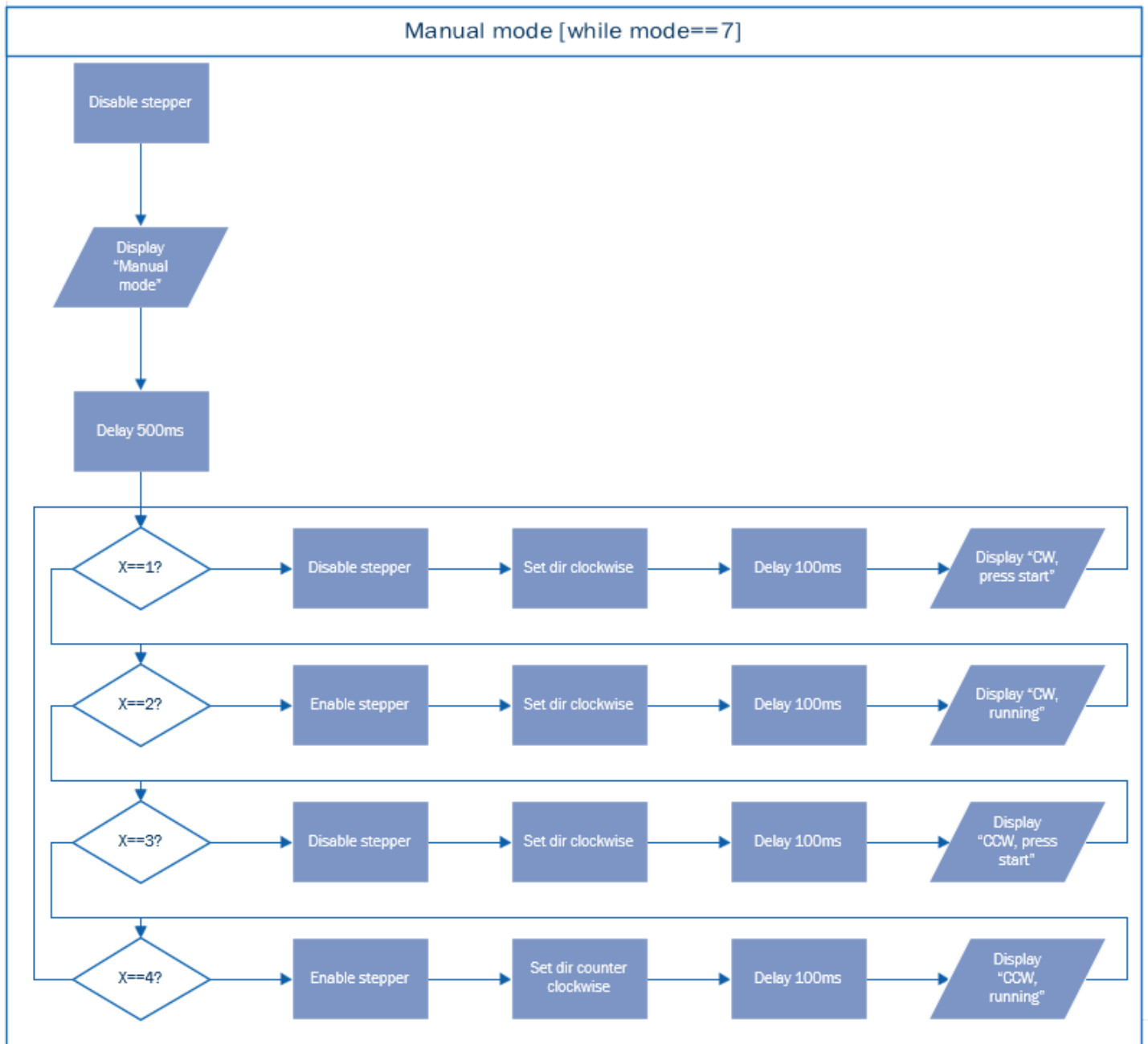


Figure 51 - Manual mode function flowchart

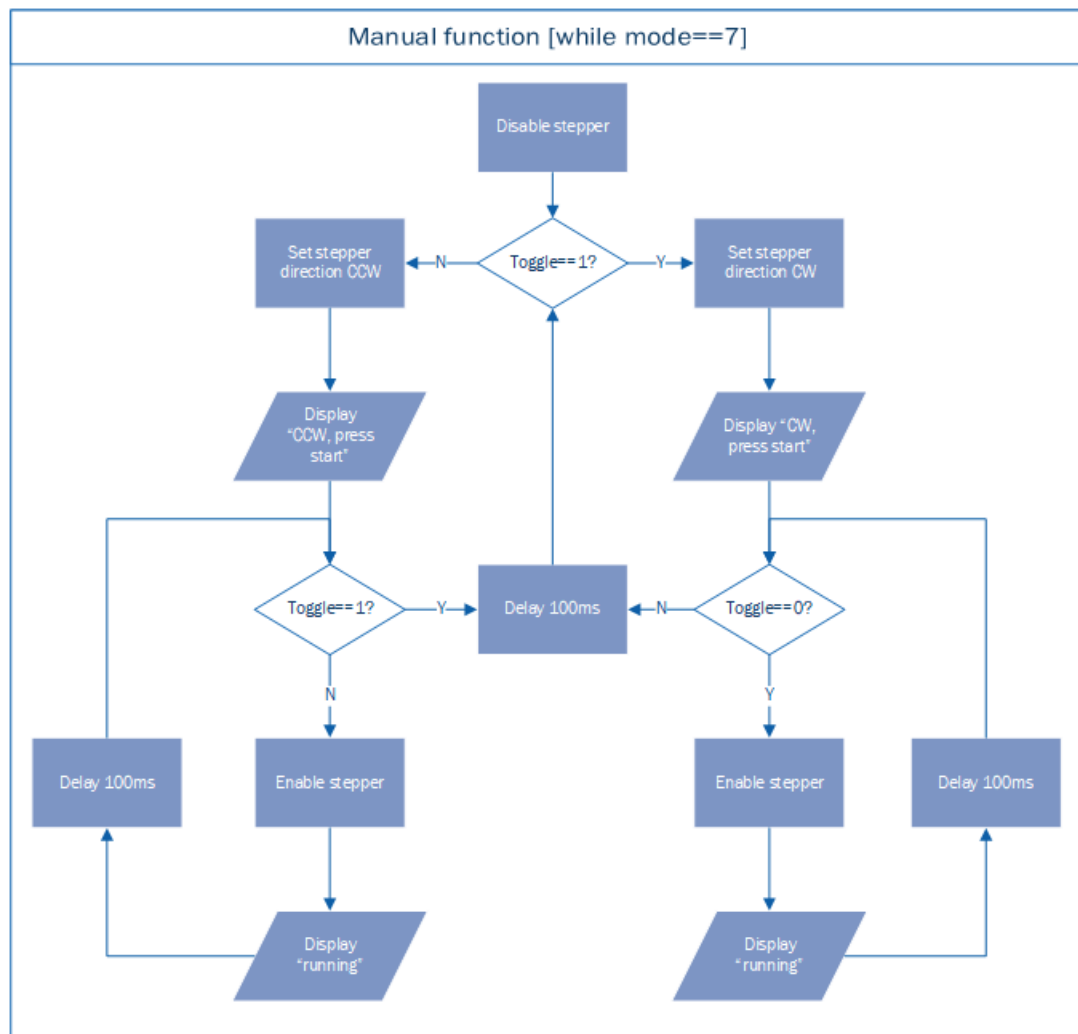


Figure 52 - Manual function alternative inspired by finite state machine flowchart. Not implemented in final design.

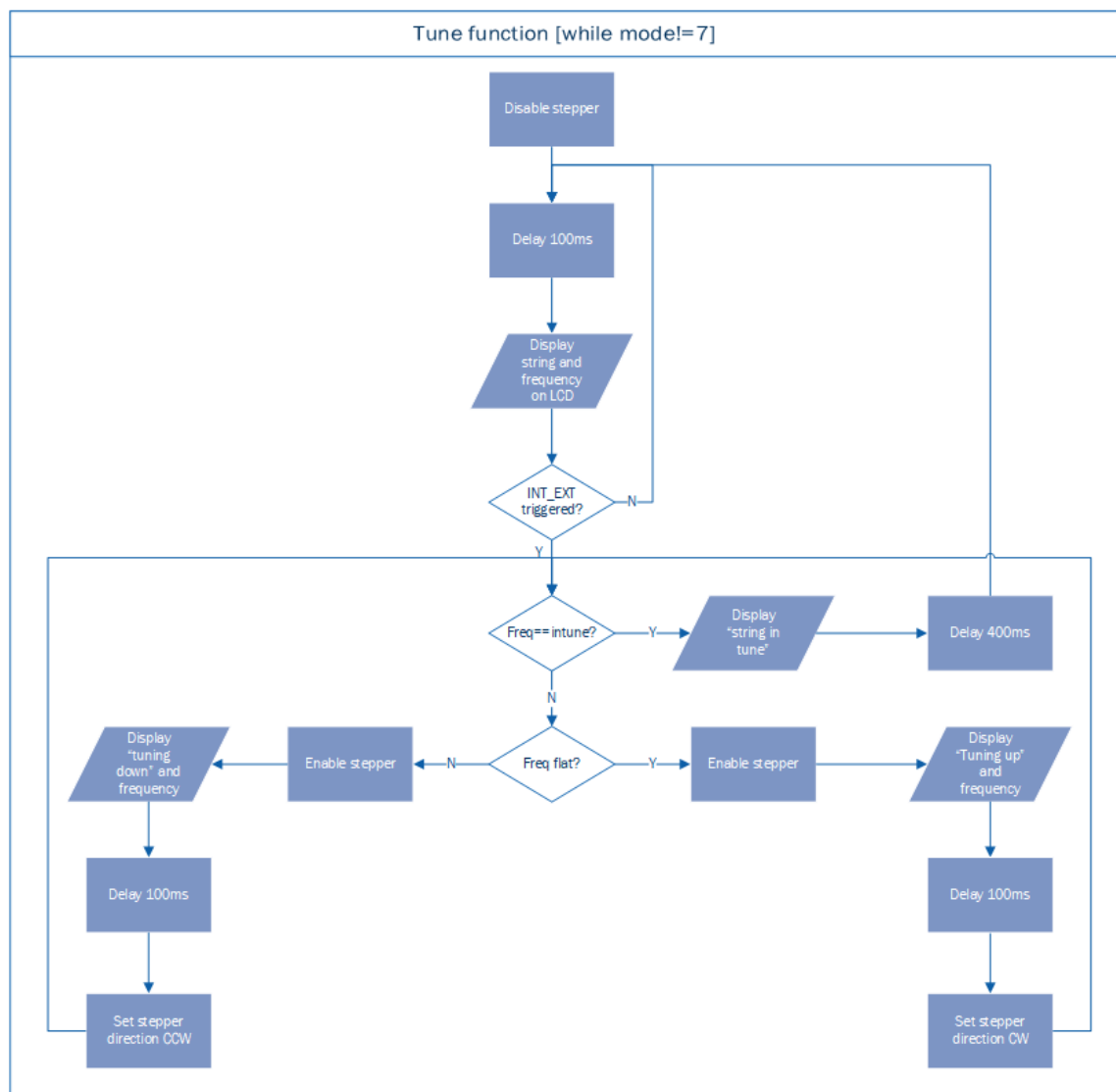


Figure 53 - Tune function flowchart



PIC16(L)F1705/9

Cost-Effective 8-Bit Intelligent Analog Flash Microcontrollers

Description

PIC16(L)F1705/9 microcontrollers combine Intelligent analog integration with low cost and extreme low-power (XLP) to suit a variety of general purpose applications. These 14- and 20-pin devices deliver on-chip op amps, core independent peripherals (CLC and COG), Peripheral Pin Select and Zero-Cross Detect, providing for increased design flexibility.

Core Features

- C Compiler Optimized RISC Architecture
- Only 49 Instructions
- Operating Speed:
 - 0-32 MHz
 - 125 ns minimum instruction cycle
- Interrupt Capability
- 16-Level Deep Hardware Stack
- Four 8-bit Timers
- One 16-bit Timer
- Power-on Reset (POR)
- Power-up Timer (PWRT)
- Low-Power Brown-Out Reset (LPBOR)
- Programmable Watchdog Timer (WDT) up to 256s
- Programmable Code Protection

Memory

- 8 Kwords Flash Program Memory
- 1024 Bytes Data SRAM Memory
- Direct, Indirect and Relative Addressing modes
- High Endurance Flash Data Memory (HEF)
 - 128 bytes of nonvolatile data storage
 - 100k erase/write cycles

Operating Characteristics

- Operating Voltage Range:
 - 1.8V to 3.6V (PIC16LF1705/9)
 - 2.3V to 5.5V (PIC16F1705/9)
- Temperature Range:
 - Industrial: -40°C to 85°C
 - Extended: -40°C to 125°C

eXtreme Low-Power (XLP) Features

- Sleep mode: 50 nA @ 1.8V, typical
- Watchdog Timer: 500 nA @ 1.8V, typical
- Secondary Oscillator: 500 nA @ 32 kHz
- Operating Current:
 - 8 uA @ 32 kHz, 1.8V, typical
 - 32 uA/MHz @ 1.8V, typical

Digital Peripherals

- Configurable Logic Cell (CLC):
 - Integrated combinational and sequential logic
- Complementary Output Generator (COG):
 - Rising/falling edge dead-band control/blanking
- Capture/Compare/PWM (CCP) module
- PWM: Two 10-bit Pulse-Width Modulators
- Serial Communications:
 - SPI, I²C, RS-232, RS-485, LIN compatible
 - Auto-Baud Detect, auto-wake-up on start
- Up to 17 I/O Pins plus One Input-Only Pin:
 - Individually programmable pull-ups
 - Slew rate control
 - Interrupt-on-change with edge-select
- Peripheral Pin Select (PPS):
 - Enables pin mapping of digital I/O
- Timer0: 8-Bit Timer/Counter with 8-Bit Prescaler
- Enhanced Timer1:
 - 16-bit timer/counter with prescaler
 - External Gate Input mode
 - Low-power 32 kHz secondary oscillator driver
- Timer2/4/6: 8-Bit Timer/Counter with 8-Bit Period Register, Prescaler and Postscaler

Intelligent Analog Peripherals

- Operational Amplifiers:
 - Two configurable rail-to-rail op amps
 - Selectable internal and external channels
 - 2 MHz gain bandwidth product
- High-Speed Comparators:
 - Two comparators
 - 50 ns response time
 - Rail-to-rail inputs
- 10-Bit Analog-to-Digital Converter (ADC):
 - Up to 12 external channels
 - Conversion available during Sleep
 - Temperature indicator

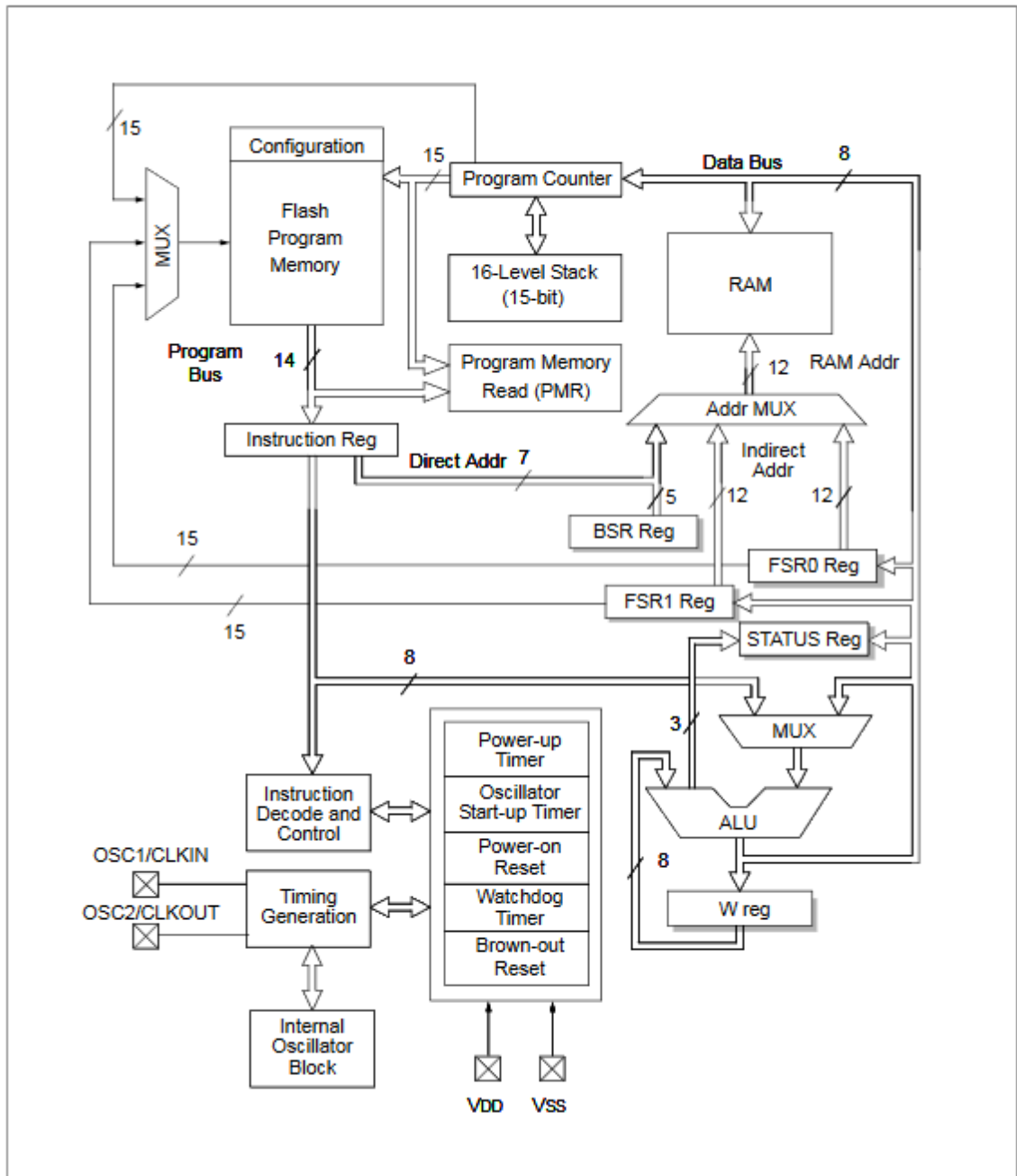


Figure 55 - PIC16F1709 block diagram. Retrieved:
<http://ww1.microchip.com/downloads/en/DeviceDoc/40001729C.pdf>