MANCHESTER 1824
The University of Manchester

An Autonomous Mechatronic Bass Guitar for the Manchester Robot Orchestra

Third Year Individual Project – Final Report

03 May 2017

Denise D'Souza

9133500

Supervisor: Professor Danielle George

School of Electrical and Electronic Engineering

Contents

Abstract

The project is the development of a mechatronic bass guitar to perform autonomously as part of the Manchester Robot Orchestra. This project is part of an outreach program by the Manchester Robot Orchestra to encourage young people and involve them in science and engineering. The orchestra consists of robotic instruments that are synchronised to perform by a robotic conductor. The project consists of two stages; the picking mechanism and the fretting mechanism with the latter as an aspirational endeavour.

The report discusses designing, assembling, implementing, and testing the various stages that involve the picking mechanism producing an autonomous working bass guitar. It also discusses the design of the fretting mechanism. The final project produces a working picking mechanism for the autonomous bass guitar.

1. Introduction

The purpose of this outreach project, which was funded by the Manchester Robot Orchestra, is to design and build an autonomous mechatronics instrument. The Manchester Robot Orchestra [1] is a University of Manchester initiative in association with The European City of Science and supported by the EPSRC, Siemens, The Granada Foundation and The University of Manchester Science and Engineering Research and Innovation. The Manchester Robot Orchestra is a collaborative and creative project for young people to get into science, engineering and music created by Professor Danielle George and Dr Erinma Ochu from the University of Manchester. They were inspired by people making instruments from recycled materials, and by The Robot Orchestra that was featured as part of Professor George's Royal Institution Christmas Lectures [1].

Each instrument from the Manchester Robot Orchestra is wired into a robotic conductor that triggers the individual instruments to play to the corresponding music.

Robotic instruments have been designed for decades by innovators from eclectic disciplines ranging from the academic to entertainment fields, implementing relevant algorithms [2]. These instruments create music autonomously using mechanical parts such as motors, solenoids and gears to produce sound-making devices by programming microcontrollers with real-time system code [2]. This interdisciplinary form of engineering applies creativity and innovation to the knowledge acquired from mechatronics, embedded systems, programming and circuit design.

Although the chosen bass guitar deviates from the quintessential guitar in terms of the number of strings, the bass guitar serves as an ideal instrument for this project. The bass guitar is a lower pitch than other guitars and consists of four strings; E, A, D and G strings. Furthermore, the bass guitar's design focuses on only the picking of the strings rather than picking and strumming, which is seen on a typical guitar. The bass guitar was designed and constructed in two stages: the picking mechanism and the fretting mechanism. The picking mechanism was the main part of the project, making it a fundamental stage. The fretting

mechanism was to be started once the picking mechanism was completed and so it was set as an aspirational part of the project due to its scope. The picking mechanism has four stepper motors mounted vertically above each string with a plectrum attached at the end of each shaft. The fretting mechanism is mounted on the neck of the guitar and pivots a finger to clamp onto the string using a solenoid. The program for this instrument takes MIDI values that correspond to musical notes on the instrument to control the mechanisms. The instrument was designed as a stand-alone project as well as part of the Manchester Robot Orchestra. It was also designed to be triggered by the robotic conductor along with the other instruments.

The motivations behind this project are for the outreach aspect as well as for the creation of an innovative robotic instrument. The involvement with the Manchester Robot Orchestra through this project has provided a platform to engage with young people helping to provide a wider reception among the youth towards science. The application of concepts covered in the mechatronics course has enabled a practical development in mechanical design, electronic design and programming.

2. Aim and Objectives

2.1. Aim

The aim of this project is to produce a working mechatronics system that is capable of autonomously playing a bass guitar for the Manchester Robot Orchestra.

2.2. Objectives

The objectives of this project are to produce a working mechatronics bass guitar. The objectives have been divided up into two stages. The first stage, which is the main part of the project, is the picking mechanism. Once the first stage is completed to an acceptable standard, the second stage objectives of the project, which is the fretting mechanism, will commence. The fretting mechanism objectives are dependent on the completion of the picking mechanism, making it an ambitious endeavour. It is essential for the project that the picking mechanism

is fully functional and optimised before any work is done on the fretting mechanism.

The objectives for the first stage of the project, the picking mechanism are as follows:

- Design and implement the stepper motor circuit for four stepper motors.
- Design and implement the mechanical frame for the picking mechanism.
- Mount the picking mechanism on the bass guitar.
- Program and test the Arduino board to accept MIDI values and control each stepper motors independently.
- Program and test the mechanism to be triggered autonomously or by an input from the robotic conductor.

Following the completion of these objectives, the secondary stage objectives of the project are as follows:

- Design and test the circuit of the solenoid for the fretting mechanism.
- Design a fretting mechanism to be mounted to the neck of the guitar.
- Design of the 'finger mechanism for fretting the notes.

3. Literature Review

3.1. Existing literature on robotic instruments

Four instruments were initially considered for the project: the piano, xylophone, flute and bass guitar. Several ideas to design these instruments were considered from research done on existing literature. The following are some of the research considered for each instrument.

a) Development of a mini-humanoid pianist.
   The research paper was based on a system that would enable a small humanoid to play a simple two-finger piano piece in response to an acoustic input [3].

b) Gesture-based human-robot Jazz improvisation.
   The project entails the design of an interactive robotic marimba player, which improvises based on a gesture framework [4]. The robot performs in real-time building up on its human-like performance [4].

c) Development of Waseda flutist robot WF-4RIV.
   The development of an anthropomorphic flautist robot to further improve its control system, enabling the robot to autonomously improve during the performance [5].

d) MechBass: A Systems Overview of a New Four-Stringed Robotic Bass Guitar.
   The construction of a four-stringed modular robotic bass guitar focusing on the fretting, picking, actuation, control electronics and control software [6].

From these four instruments considered for the project, the bass guitar was selected. In the case of the keyboard and xylophone, the Manchester Robot Orchestra was in possession of similar instruments, not making it a viable option for the project because of the similar instruments. The flute, a wind instrument, would not be feasible because the time it would take to be implemented would extend beyond the time frame of the project. The choice of the bass guitar was found to be the most pragmatic option as this allowed for the use of a 4-stringed guitar as opposed to the typical 6-stringed guitar. Additionally, the bass guitar provides lower frequency notes, which are absent in most of the instruments in the Manchester Robot Orchestra. Furthermore, the bass guitar makes for an ideal choice due to the fact that they require only picking of the strings while other guitars require picking and strumming of the strings.

3.2. Existing solutions of Robotic Guitars

Once the instrument was selected, the literature on robotic guitars was extensively researched. The literature found was analysed and used to help decide or exclude design theories for the project.

3.2.1. Strumbot and Mechbass

Research done on the existing solution of guitars was reviewed and two projects from the Victoria University of Wellington were selected. The projects developed were a standalone six-string unit robotic guitar [7] and a four-stringed modular robotic bass guitar [6]. Both projects used stepper motors to move the attached plectrums to pluck each string unit. The idea to use stepper motors originated from these existing solutions, however, this was applied to a full-size guitar as opposed to an individual string unit. Both instrument designs can be seen Figure 3.1.

Figure 3.1: Strumbot [7] (left) and CAD design of Mechbass [6] (right).

3.2.2. University of Leeds Robotic Bass Guitar

The robotic bass guitar developed by Andrew Jackson from the University of Leeds is an autonomous bass guitar loaned to the Manchester Robot Orchestra[8]. The myRio controls the robot with pneumatic values picking the strings along with a range of mechanically designed fingers that are pivoted onto the strings by solenoids. The complete instrument can be seen in Figure 3.2.

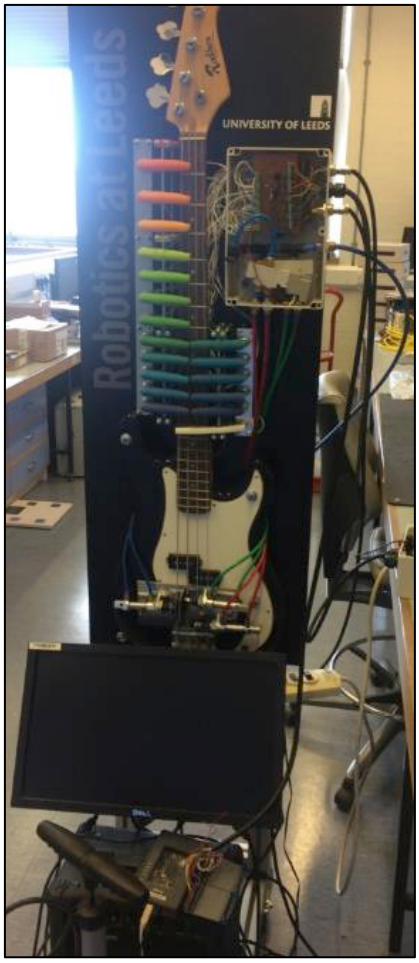


Figure 3.2: The University of Leeds robotic bass guitar.

The current idea for the fretting mechanism is based on the similar design of individual 3-D printed fingers over different strings on different frets.

3.3. Glock-o-bot

Glock-o-bot is one of the Manchester Robot Orchestra's instruments. It is an Arduino controlled robotic Glockenspiel that uses solenoids to strike the keys. The documentation for this instrument has helped in aspects of the code for this project. The instrument is shown in Figure 3.3.

Figure 3.3: Glock-o-bot [9].

3.3.1. Timer code

Glock-o-bot's documentation [10] included a timer function that configures and starts the internal timer to count for one quantised beat time. This function accepts a beat time value, which is used to set the interrupt timer and act as the main time reference for the tune. The beat time value sets the duration of the quantised step in microseconds and is derived from the Tempo.

To calculate the beat time value in the code, the tempo and beats per bar must be known. The beat time equation for 1/8th of a beat time is applied below.

$$Beat_{Time} = \frac{60}{T} \times \frac{1}{8} \quad \text{.....(1)}$$

Where: T is the tempo.

3.3.2. Conversion of MIDI files to MIDI values

The Glock-o-bot documentation [10] also discusses how to convert MIDI files into MIDI values that can be input into its program code. This is the same way MIDI values are obtained and used in the bass guitar code to make the stepper motors pick the correct strings when performing. Anvil Studios [11], a free software that is designed for MIDI equipment is used to convert the MIDI files into MIDI note number. As a few robotic instruments from the orchestra use these converted MIDI note numbers, the MIDI numbers from these converted files were used in the bass guitar code.

3.4. Control of Robotic fingers

The initial design idea considered for the fretting mechanism involved reviewing the literature about robotic fingers with two Degrees of Freedom (DOF). A research paper on the force capability of underactuated fingers discusses the use of springs and mechanical limits to adjust it to any irregularly shaped objects. This avoids the need for complex control strategies and sensors [12].

The decision to move to the current fretting mechanism was because of the constraints on the time frame of the project. This was due to the amount of time needed to review the literature in order to complete the fretting mechanism.

3.5. Guitar Theory

The National Guitar Academy [13] discusses the basic requirements and background to play the bass. There are four strings on a bass guitar, each string with a name and number. The thickest string is the fourth string often referred to as the low E string. The third string is tuned to A, and the second string is tuned to D. The thinnest string is the first string and it is tuned to G. These strings are the same as a standard guitar but an octave lower making them sound deeper[13].

Each string goes down the 12 fret forming an octave starting at E, A, D and G respectively. The C on the A string, which is on the third fret is Middle C and is the same as the C on the E string, which is on the eighth fret. In the same way, the C on the D string, which is on the tenth fret, is the same as the C on the G string, which is on the fifth fret but an octave higher to that of Middle C. This can be seen on the bass guitar notes in Figure 3.4 that show which fret must be held to play the equivalent note.

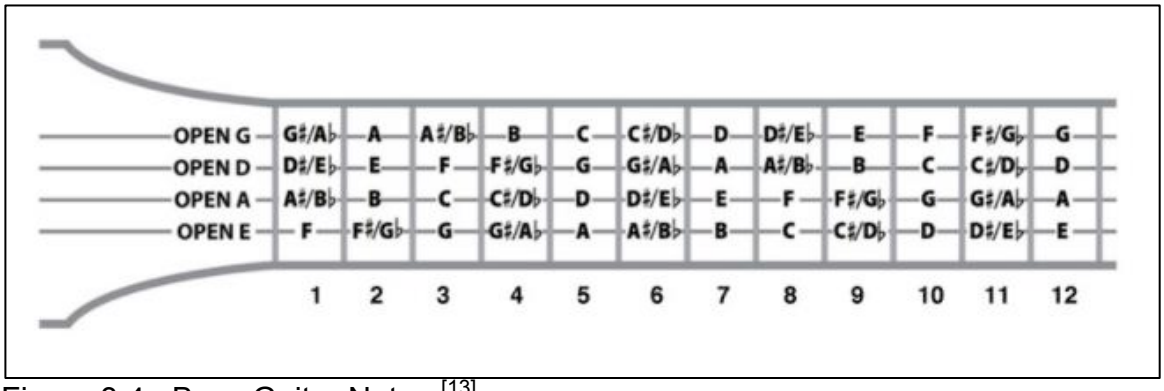


Figure 3.4: Bass Guitar Notes [13].

3.6. MIDI

MIDI [14], also known as the Musical Instrument Digital Interface is an interface for musical instruments. It was originally conceived as a way for a musician to layer or double the voices from different synthesisers in real time. It was also used to provide a common timing source for synchronising sequences and drum machines. Nowadays, MIDI can be used to stream digital data, taking advantage of the speed and processing capabilities of a computer. MIDI software allows the user to record and edit sequencer tracks, store and edit synthesisers, score arrangements and print out resulting notations [14].

MIDI files can be converted to generate MIDI numbers, which are used to put into the code for the mechatronic bass guitar. The values from the table below are used to correspond to a note on an octave, which is plugged into an array in the Arduino code.

Octave	Note Numbers											
	C	C#	D	D#	E	F	F#	G	G#	A	A#	B
0	0	1	2	3	4	5	6	7	8	9	10	11
1	12	13	14	15	16	17	18	19	20	21	22	23
2	24	25	26	27	28	29	30	31	32	33	34	35
3	36	37	38	39	40	41	42	43	44	45	46	47
4	48	49	50	51	52	53	54	55	56	57	58	59
5	60	61	62	63	64	65	66	67	68	69	70	71
6	72	73	74	75	76	77	78	79	80	81	82	83
7	84	85	86	87	88	89	90	91	92	93	94	95
8	96	97	98	99	100	101	102	103	104	105	106	107
9	108	109	110	111	112	113	114	115	116	117	118	119
10	120	121	122	123	124	125	126	127				

Figure 3.5: MIDI Note Numbers [15].

MIDI controllers can have up to 128 distinct pitches or notes and are assigned a unique number key to each note [15]. From Figure 3.5, Octave 0 is described as being the lowest octave of the MIDI note range with middle C's note name usually being C5. The lowest note name is then C0 (note number 0), and the highest possible note name is G10 (note number 127) [15].

The bass guitar notes begin from the Open E string, the note name being E2. This is because some MIDI controllers consider the first 2 octaves as -2 and -1 and so, middle C's note name is C3 [15]. This is important as MIDI values for the bass guitar are in the range of 35-50.

3.7. Arduino tutorials

The online tutorials available on the Arduino website [16], as well as the example codes for stepper motors, solenoids and push buttons, were used to help test out the initial circuits. One of the primary decisions for choosing this controller was due to the user-friendly libraries and tutorials. The code for the stepper motors and solenoid can be found further in the report while the push button example can be found in Appendix 5.

4. Proposed system

The overview of the proposed system for this project is the picking mechanism, the fretting mechanism, their integration with each other and the integration of the instrument with the Manchester Robot Orchestra. The proposed system is based on the completion of the picking mechanism to an acceptable standard before any work is done on the fretting mechanism. This is due to the importance of the picking mechanism as without the instrument working properly, the aim of the project would not be met.

4.1. Picking mechanism

The proposed design for the picking mechanism is four stepper motors with plectrums attached to each shaft, which will pick the strings of the bass guitar when rotated. The mechanism is designed to hold four motors vertically over each string and a slot at the end of the extended shaft clamps hold of the plectrum.

4.2. Fretting mechanism

The proposed design for the fretting mechanism consists of 3-D designed fingers slotted into a base mechanism that is fitted around the neck of the guitar. The fingers are pivoted by a solenoid on the opposite end which when pressed the finger holds down on the string.

4.3. Integration of mechanism

The integration of both mechanisms involves the use of the Arduino Mega [17], which has additional features to the Arduino Uno[18] that enable it to support both mechanisms to work simultaneously. The Arduino Mega supports fifty-four digital inputs/outputs pins compared to the Arduino Uno's fourteen digital input/output pins. As such, it can support the solenoid pins and stepper motor pins on the same controller board, so they function simultaneously. This will enable both mechanisms to have plenty of pins to be able to be controlled by the Arduino Mega.

4.4. Integration with the Robot Orchestra

The integration with the orchestra requires the whole mechanism to be triggered by the robotic conductor when connected to the circuit. The instrument will have an external push button circuit to trigger the mechanism as a stand-alone device. When the push button circuit is disconnected from the main circuit, the relay connections from the robotic conductor can replace this circuit avoiding any changes in the code.

5. Theoretical Development

The selection of motors and controller board were essential to the project and each decision has been carefully thought out and compared to ensure that the selection is beneficial to the project.

5.1. Hybrid Stepper motor

The hybrid stepper motors were chosen for the project as they have a small step length, which can produce a high resolution angular positioning of the plectrum [19]. They do not require any feedback mechanisms and can be controlled to a high degree [20]. Stepper motors generate precisely defined incremental changes or steps from the changes in the stator windings electrical pulses [19]. The shaft of a stepper motor is mounted with a series of magnets and is controlled by a series of electromagnetic coils that are charged positively and negatively in a specific sequence, thereby moving the shaft forward or backwards in small precise steps [20]. The excitation current is important in a hybrid stepper motor

and therefore a bipolar drive circuit or a unipolar drive with a bifilar winding configuration is required [19]. Using the bifilar winding design simplifies the driving circuit requirements in the stepper motors and so unipolar was chosen over bipolar. The unipolar and bipolar wirings are shown in Figure 5.1.

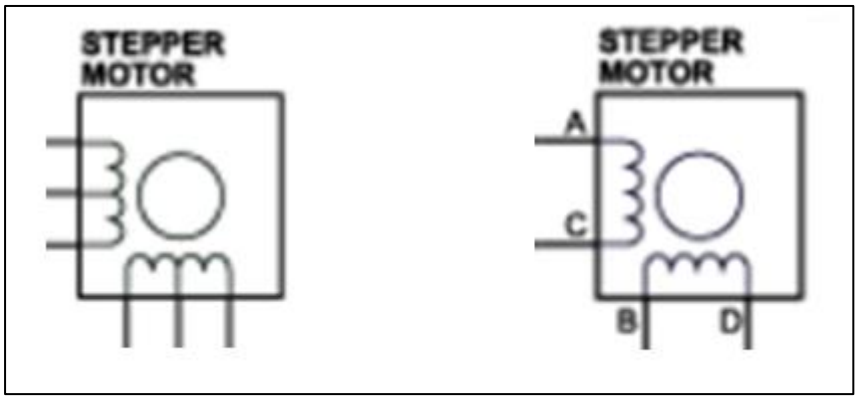


Figure 5.1: Unipolar (left) and bipolar (right) wirings [21].

5.1.1. Motor selection

The motors that were available to test the bipolar drive circuit were the MY5602 Stepper Motor and the SP2575M0206-A Stepper Motor. The datasheets for both motors, found in Appendix 4, indicate the wiring connections; technical information and product dimensions. The two stepper motors have very different maximum torque values; MY5602 Stepper Motor has a torque maximum of 8N-cm [22] while the SP2575M0206-A Stepper Motor has a torque maximum of 1.6N-cm [23]. From the stepper motors datasheets [22][23], the step angle for the MY5602 Stepper Motor is 1.8 degrees per step while the SP2575M0206-A Stepper Motor has a degree of 7.5 per step. Thus, there are 200 steps per 360-degree rotation and 48 steps per 360-degree rotation respectively. The rotation along with the 8N/cm torque produced by the MY5602 Stepper Motor made it the ideal choice to rotate plectrums attached to its shaft.

5.2. Controller selection

Two different microcontrollers: The Arduino Uno and PIC Microchip 18F8722 (PIC) were used to investigate the control of stepper motors. The decision to use the Arduino Uno was due to its user-friendly library functions and examples. The in depth knowledge attained from the School of Electrical and Electronics Engineering's PIC controller is from the Microcontroller Engineering II module. The Arduino controller's online tutorials explain the circuit and theory to control these motors along with the pre-existing library files to control its rotation. This concept was applied to the circuit and code on the PIC board using the experience from programming module.

5.2.1. Arduino controller board

The Arduino Tutorial - 'Stepper One Revolution' [20], details the wiring and Arduino code to program the motor's shaft to rotate a full rotation clockwise and anti-clockwise. Each stepper motor is connected as shown in the schematic circuit diagram in Figure 5.2 and wired onto a breadboard for testing.

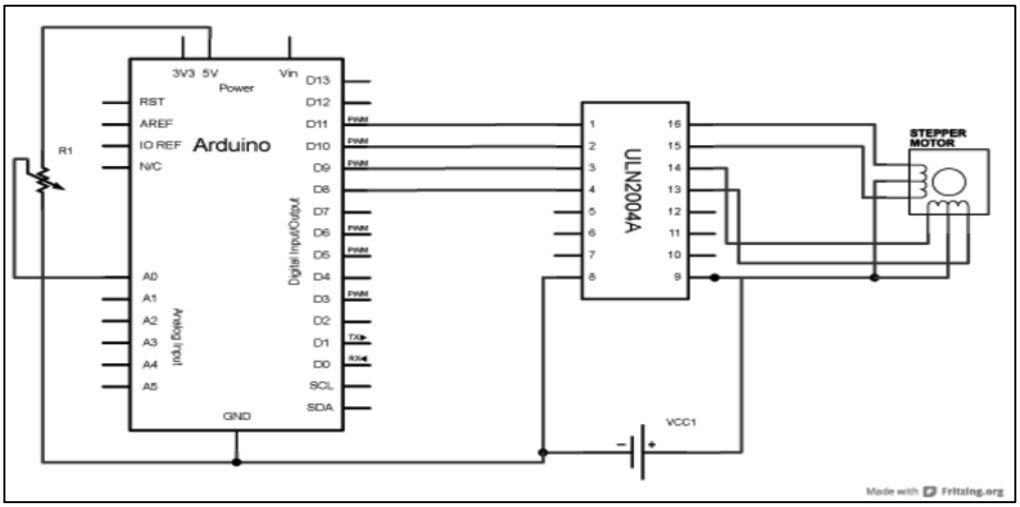


Figure 5.2: Schematic Diagram for unipolar stepper motor [20]

The components used to test this circuit are as follows:

- Arduino Uno Board
- MY5602 Stepper Motor
- ULN2003A Bipolar Transistor Array
- 10K Potentiometer

- 12V Power supply
- Jumper cables
- Breadboard

The four pulse-width modulation (PWM) outputs on the Arduino are wired to the base input pins on the bipolar transistor to control the stepper motor. The bipolar transistor array acts like an electric switch to each motor wire, which is connected to the collector output pins, allowing current to sink through the wire enabling it as high. When the switch is off, there is no current going through the wire enabling it as low. This allows for a specific stepping sequence control for each of the coils, as shown in Figure 5.3 when a voltage is applied to it.

Step	wire 1	wire 2	wire 3	wire 4
1	High	low	high	low
2	low	high	high	low
3	low	high	low	high
4	high	low	low	high

Figure 5.3: Stepping sequence for four wires [21].

From the tutorial [20], the Arduino code initialises the stepper library function and controls the speed in revolutions per minute (rpm). Using this library (Stepper.h), the ‘myStepper.step’ function sets the steps to 200 for one full rotation and runs the motor clockwise (‘myStepper.step (stepsPerRevolution’) and anticlockwise (‘myStepper.step(-stepsPerRevolution’) as seen in the code in Figure 5.4.

```
/*
 Stepper Motor Control - one revolution

 This program drives a unipolar or bipolar stepper motor.
 The motor is attached to digital pins 8 - 11 of the Arduino.

 The motor should revolve one revolution in one direction, then
 one revolution in the other direction.


 Created 11 Mar. 2007
 Modified 30 Nov. 2009
 by Tom Igoe

 */

#include <Stepper.h>

const int stepsPerRevolution = 200;  // change this to fit the number of steps per revolution
// for your motor

// initialize the stepper library on pins 8 through 11:
Stepper myStepper(stepsPerRevolution, 8, 9, 10, 11);

void setup() {
  // set the speed at 60 rpm:
  myStepper.setSpeed(60);
  // initialize the serial port:
  Serial.begin(9600);
}

void loop() {
  // step one revolution  in one direction:
  Serial.println("clockwise");
  myStepper.step(stepsPerRevolution);
  delay(500);

  // step one revolution in the other direction:
  Serial.println("counterclockwise");
  myStepper.step(-stepsPerRevolution);
  delay(500);
}
```

Figure 5.4: Code written to run stepper motor [20].

5.2.2. PIC controller board

The theory behind the PIC controller to rotate the shaft clockwise and anticlockwise uses knowledge from the Microcontroller Engineering II module and the Arduino tutorial. Each stepper motor is connected to the transistor as shown in the schematic diagram of the circuit in Figure 5.5.

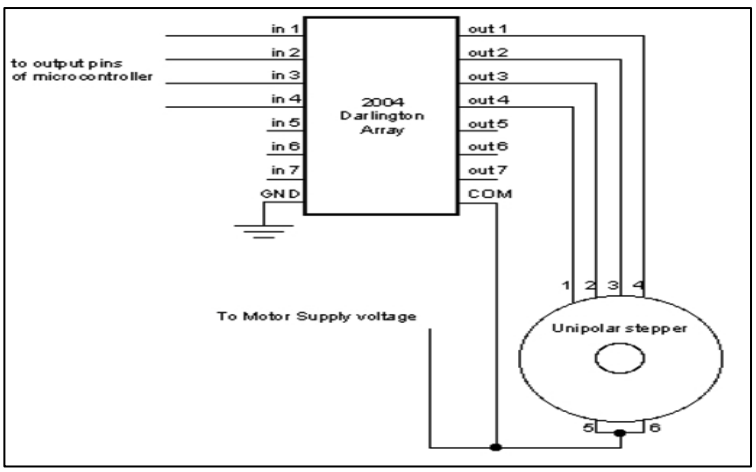


Figure 5.5: Schematics diagram for the PIC controller [24].

The components used to test the circuit were

- PIC18F8722 Microcontroller
- MY5602 Stepper Motor
- ULN2003A Bipolar Transistor Array
- 2x Pushbuttons
- 1N4004 diode
- 7.5V Power supply
- Jumper cables
- Breadboard

The program rotates the shaft clockwise and anticlockwise by pressing two push buttons using the wiring from the datasheet. Similar to triggering the stepping sequence in the Arduino, the output pins of PORTB were connected to the base input of the transistor acting like a switch to each stepper motor wire connected to the collector output. The pushbuttons were wired to the PORTB input pins and on pushing each button the motors ran clockwise or anti-clockwise. The code for the PIC can be found in Appendix 6.

Unfortunately, the test for the motors was not accurate and the complexity of using four stepper motors proved to be tricky. As the design for the PIC would require more time than assigned, there were no further tests carried out.

5.2.3. Comparison

The Arduino setup was straightforward and allowed for additional motors to be connected in parallel with ease. The testing of the circuit proved more accurate than the PIC and the Arduino's prebuilt library files make coding the subsequent motors efficient. The Arduino circuit was accurate each test time and the prebuilt library files with the code make it efficient to control more steppers. The use of the Arduino also meant easier interfacing between the two different mechanisms.

6. Initial Development of Picking Mechanism

The methodology of this process was designed and implemented on one stepper motor first and then applied to four stepper motors, modifying and improving the designs where needed.

6.1. Initial Mechanical Design and Limitations

The circuit as discussed in Section 4 was used to test the MY5602 Stepper Motor. The Arduino library and code that was used rotated the motors 100 steps or half a revolution forward and backwards to simulate the movement that would be used in the final mechanism.

Initially, the mechanical frame was designed with the idea to position the stepper motors at the end of each string. As the size of the motors was slightly larger than the strings, two stepper motors could not fit side-by-side. Instead, the stepper motors would be positioned to stagger alternately. A long tubular shaft extension to hold the plectrum was initially designed to test the ability of the motor to spin when positioned at the base of the guitar. This extended shaft was designed with six slits at the ends to hold the plectrums. When the motor spins the shafts picks the strings with six plectrums thereby increasing the speed of picking. The design for the shaft extension is shown in Figure 6.1.

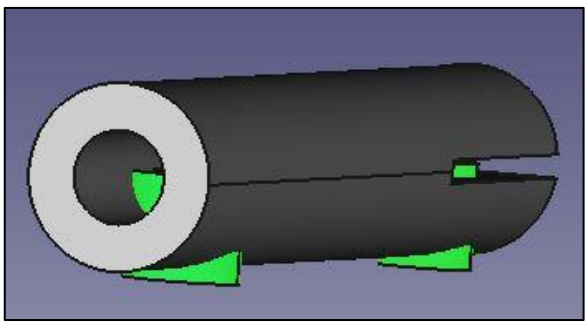


Figure 6.1: Extended tubular shaft.

When tested with more than one stepper motor, due to the large radius required for the shafts to hold the plectrums, motor's shafts would not align above each string. The solution to this would be to have alternate motors angled in order to avoid the plectrums at the end of the shafts from colliding. As this complicated the intended simple mechanical frame, an alternative method was considered.

The motors would be placed vertically above each string with the plectrums attached perpendicularly.

As with the previous design, the new mechanical design would not fit over the strings if more than one plectrum was used. Therefore, the design of the tubular shaft was modified to contain a single slit to hold the plectrums.

6.2. Initial Circuit Design and Limitations

Once the decision was taken to use the Arduino Uno to program the stepper motors, the circuit design as described in Section 5.2 was implemented for four stepper motors. However, the circuit design was found to be limited to only three stepper motors. This was due to the Arduino Uno only having fourteen digital input/output pins. The circuit design for four stepper motors would require twenty-four digital input/output pins, as this was a four-wire controller circuit. The design of the new stepper motor circuit required a two-wire controlled stepper motor to comply with the specifications of the Arduino Uno.

7. Final Picking Mechanism

The development of the picking mechanism consisted of several stages: mechanical design for the motors; circuit design of the mechanism; implementing the mechanism; programming the motors to be controlled by MIDI values and triggering the mechanism independently. The finished mechanism is shown in Figure 7.1.

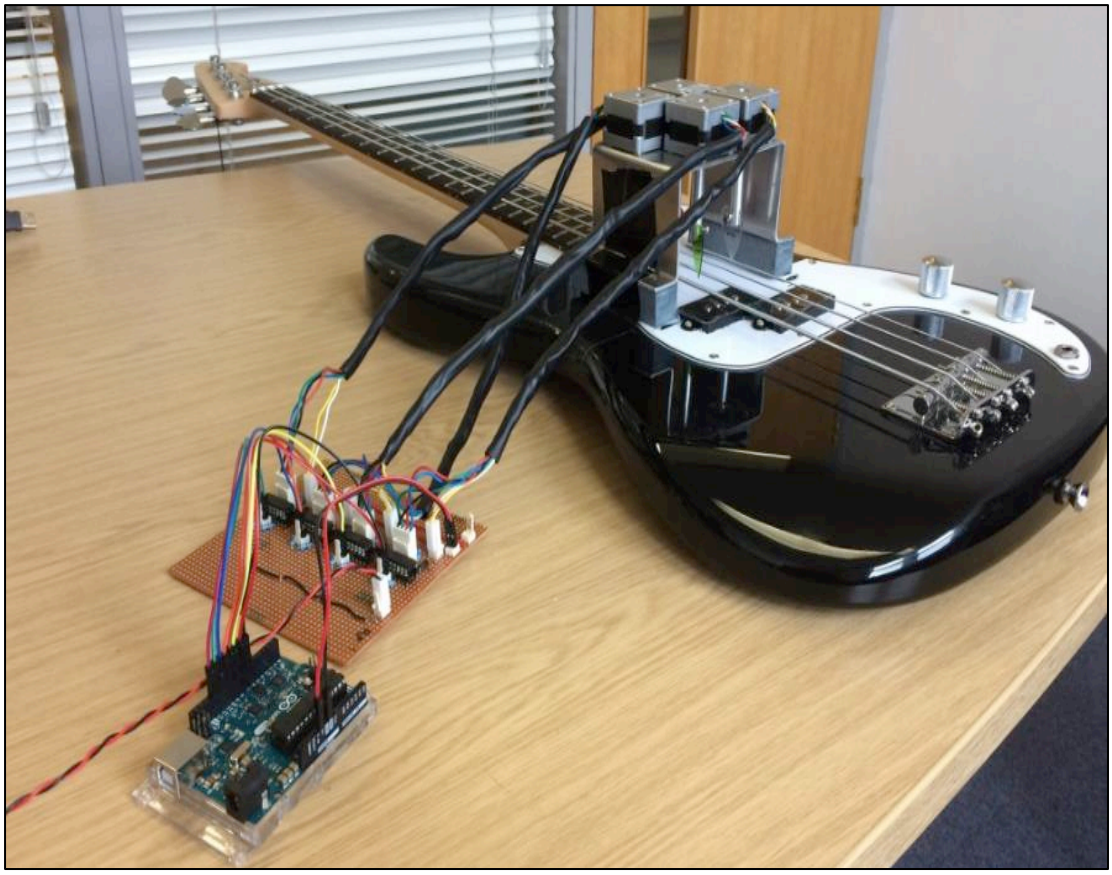


Figure 7.1: The finished design for the picking mechanism.

7.1. Mechanical design

The mechanical design of the picking mechanism consists of the motor frame and the extended motor shafts. The motor frame was designed to hold four stepper motors vertically staggered over each string. The frame design is a metal plate folded twice on each side to create a stand to sit over the strings. The motors are screwed into the plate to position them securely. The measurements for the designs can be found in Appendix 7 while the mechanical design of the motor frame is shown in Figure 7.2

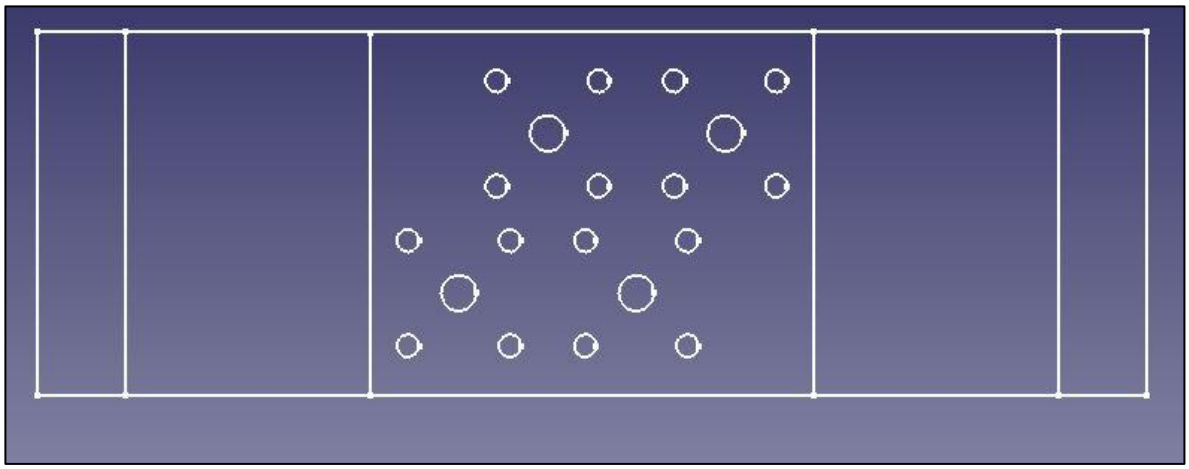


Figure 7.2: The mechanical design of the motor frame.

The extended shafts are designed with screw holes at the two ends, one to secure on to the stepper motor shaft and the other to secure the plectrum in the slit. The length of the extended shaft was measured to be 45mm long and the radius of the hole drilled in the middle is slightly larger than the motor shaft's radius, which is 2.5mm wide. The mechanical design of the shaft is shown in Figure 7.3

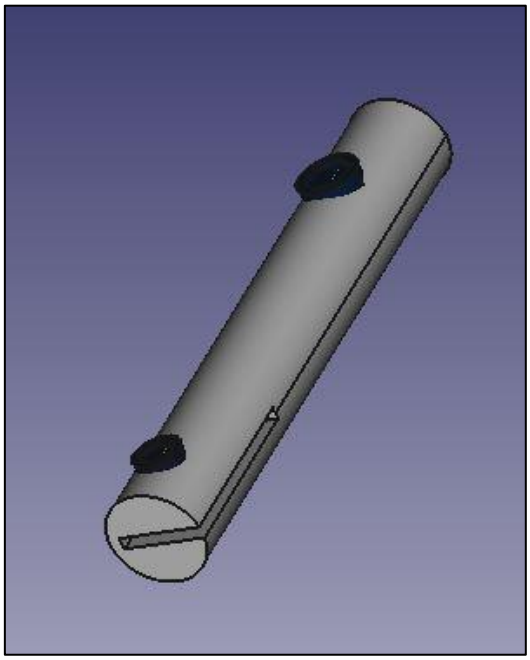


Figure 7.3: Extended shaft with a plectrum slot.

7.2. Circuit design

The circuit design for the stepper motor was taken from an Arduino example tutorial. The circuit diagram was used and tested to ensure that the circuit was properly functioning. It discusses the use of two-wire controlled circuits using the ULN2003a, 4001 diode and 1k resistors. The wiring diagram as seen in Figure 7.4 shows the circuit from the tutorial. The push button circuit was also taken from the Arduino tutorial.

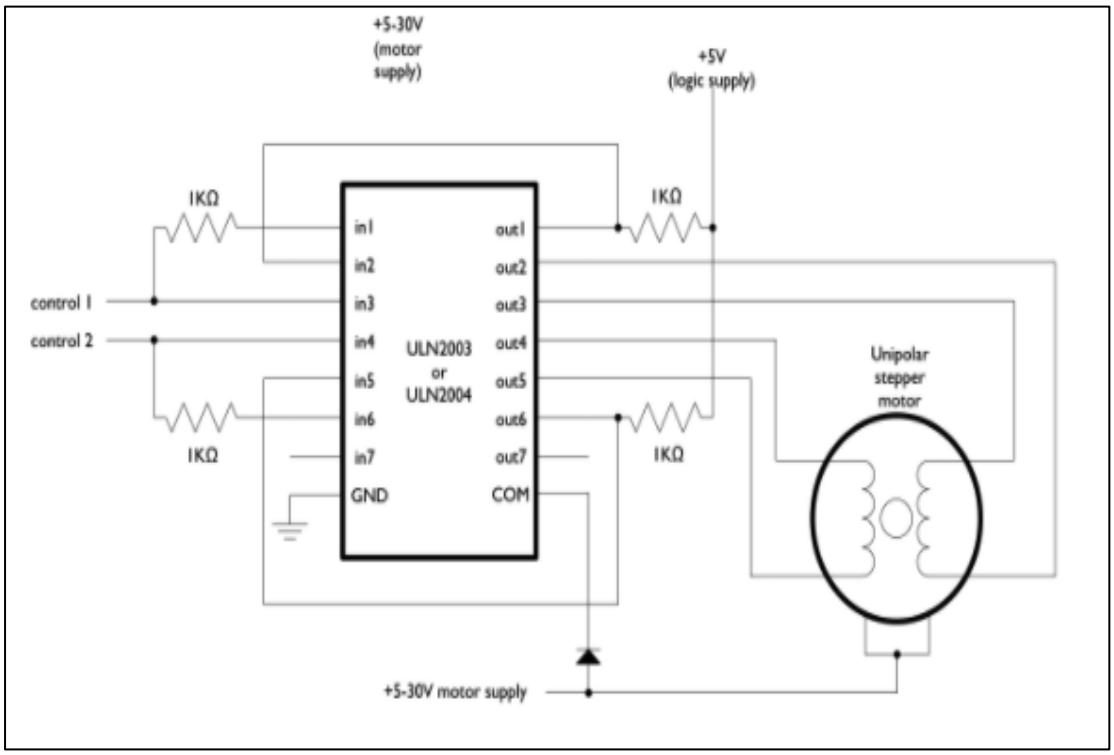


Figure 7.4: Two-wire control wiring diagram [21].

7.2.1. Bill of Materials

The following components were used in the final circuit design.

Bill of Materials	
Quantity	Part Type
Arduino Uno	1
Strip-board	1

MY5602 Stepper Motor	4
ULN2003A Bipolar Transistor Array	4
4001 Rectifier Diode	4
1kΩ Resistors	16
10kΩ Resistor	1
Jumper cables	1
Push button	1
Molex wire-board connector (4 pin)	5
Molex wire-board connector (2pin)	11
Header wire-board connector (4 pin)	5
Header wire-board connector (2pin)	11
DIP socket (16 contacts)	4
5V 3A Power supply unit	1
5V Arduino Power supply unit	1

7.2.2. Stepper motors circuit

The two pulse-width modulation (PWM) outputs on the Arduino are wired to the base input pins on the bipolar transistor to control the stepper motor. The bipolar transistor array acts like an electric switch to each motor wire, which when connected to the collector output pins, allows current to sink through the wire enabling it as high. This allows for a specific stepping sequence control for each of the coils. The schematic diagram of the circuit is shown in Figure 7.5.

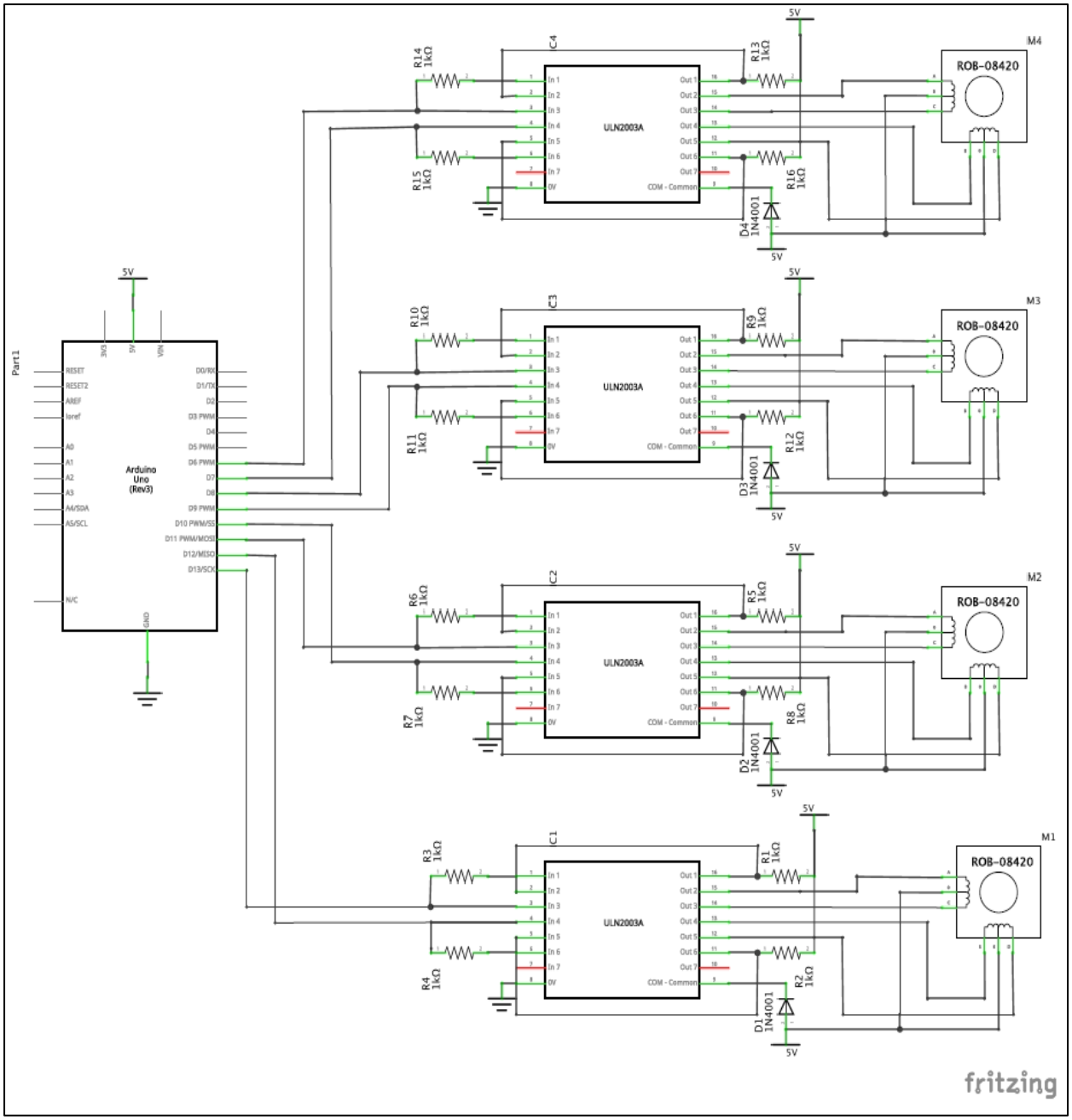


Figure 7.5: Schematic diagram for the stepper motors.

Once the wiring diagram was tested for one of the stepper motors to ensure it was functioning as needed, the circuit was implemented for four stepper motors as shown in the circuit diagram in Figure 7.5. After the circuit was successfully tested on the breadboard, the circuit was mounted onto the stripboard and neatly soldered into place.

7.2.3. Push button circuit

The push button circuit is an additional circuit taken from the Arduino tutorial [25], which triggers the stepper motors when pressed. It is separate from the main circuit as it can be interchanged with the relay connection for the robotic conductor when part of the orchestra. The push button is grounded with a 1k resistor and pulses a signal to pin 2 of the Arduino when pressed. The circuit diagram of the circuit is shown in Figure 7.6.

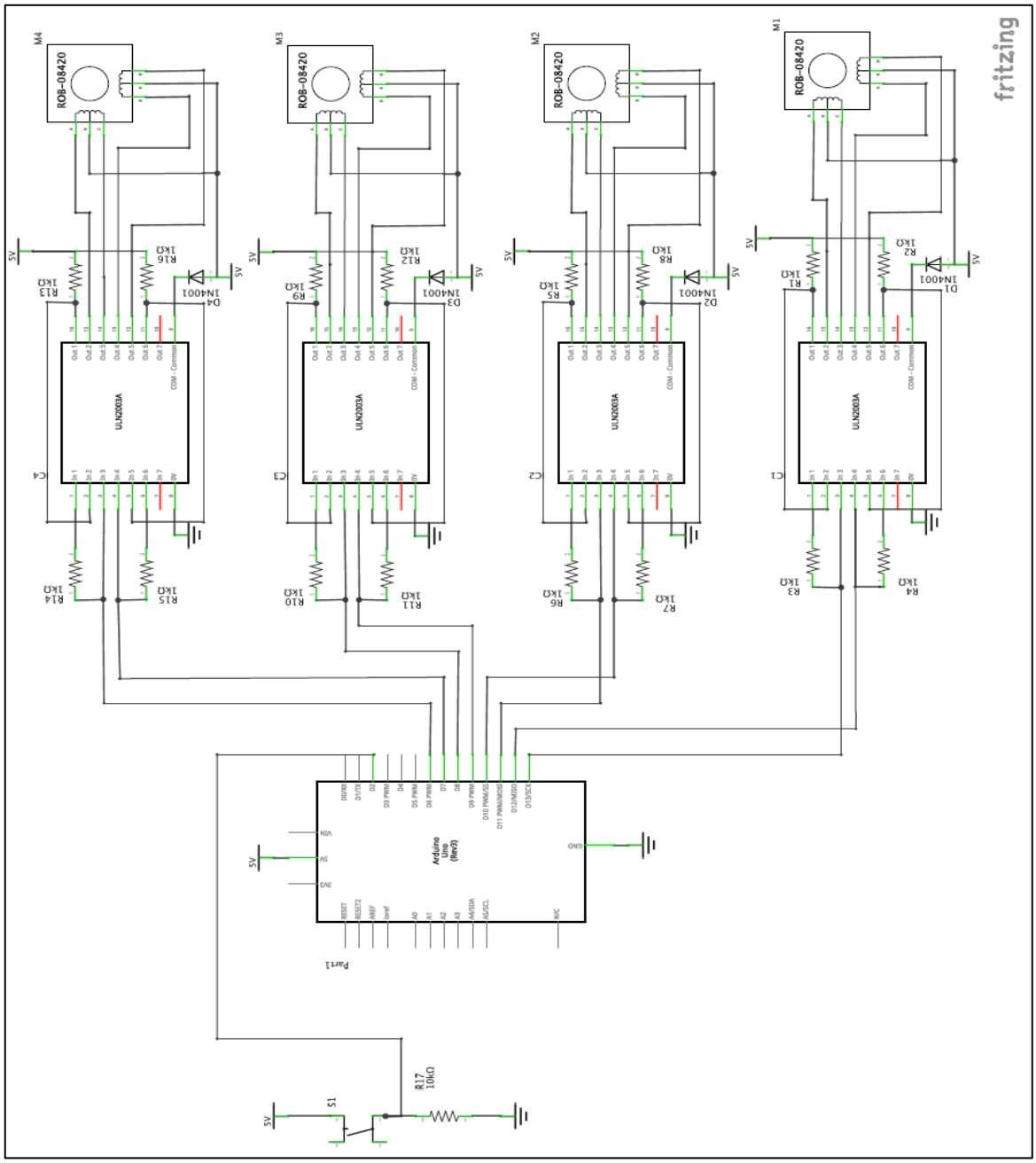


Figure 7.6: Schematic diagram of the complete circuit.

7.3. Implementation of designs

Once the stepper motor frame and shafts were designed and produced, the integration of the mechanical design and circuits were done. The mechanical frame was secured onto the guitar with sticky tape so as not to drill through the guitar's internal circuitry. A more permanent fastening of the frame will be done after the project as an item for future work. The plectrums were screwed into the shafts above each string so that they would pick only the string below them. The circuit board was then connected to the stepper motors and Arduino with the whole mechanism looking like the Figure 7.7 below.

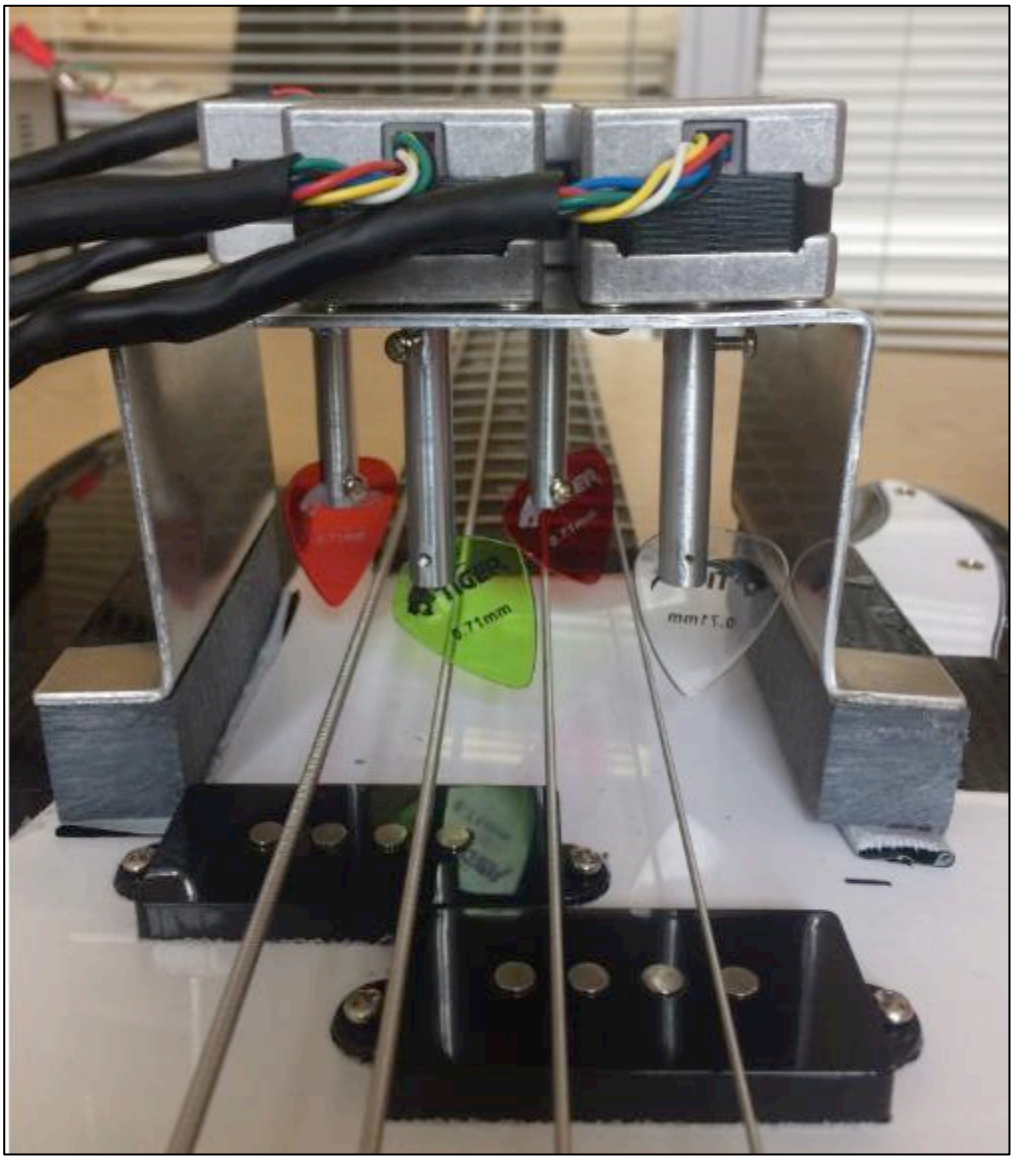


Figure 7.7: Mechanical design mounted on to the guitar.

7.4. Mechanism tests

The mechanism tests for this project were the different programs written to perform various functions. Three main tests were conducted for the working of the mechanism and its parts. The initial program was to rotate the stepper motors clockwise and anti-clockwise. This was followed by implementing MIDI numbers and lastly modified to include being triggered by a push button. All the tests for this project were recorded and uploaded to Vidme[26], a website to upload original video. This documentation of the project can be found at the link below. https://vid.me/Mechatronic_Bass_Guitar.

7.4.1. Stepper motor control

The first test conducted was to ensure that all the stepper motors rotated clockwise and anticlockwise as expected and repeatedly. This was first tested as stand-alone stepper motors before being tested when the motors were attached to the mechanical frame. The code for this test uses a variation of the Arduino Stepper Motor one-revolution example as seen in Section 5.2.2. Instead of four-wire control, only two-wire controls are initialised due to the fact that the Arduino only have fourteen digital input/output pins. The motor is programmed to rotate a whole rotation clockwise and then anticlockwise. Once this test was completed satisfactorily and recorded, the motors were then attached to the mechanical frame and then tested to see whether they would pick the strings accurately every time. When initially tested, the plectrums did not pick the strings every time and so they were adjusted and securely tighten before testing again. When the testing was successful, it was documented and recorded.

7.4.2. Motor control with MIDI numbers

The Arduino code is programmed to rotate the stepper motors depending on the MIDI note numbers. It is implemented with the Timer function from the Glock-o-bot instrument. The code takes an array of MIDI note numbers and each number corresponds to a string. With the help of the Timer function, the code loops for every MIDI note number resulting in its corresponding motor rotating. The code description follows the algorithm from the flow chart in Figure 7.8.

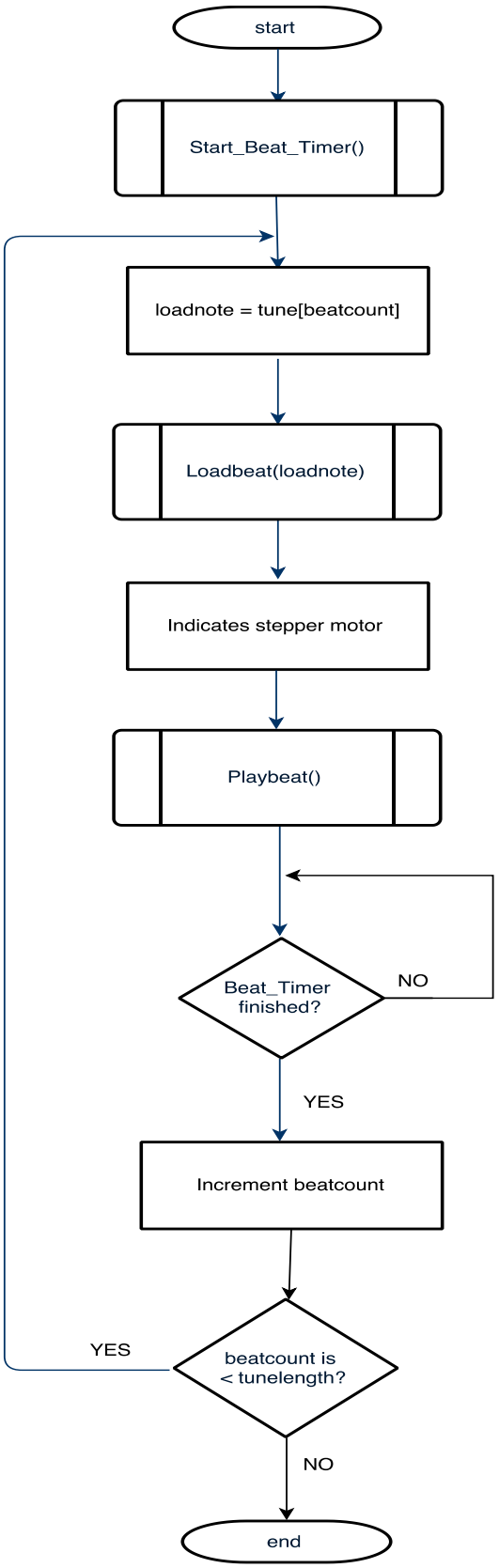


Figure 7.8: Flowchart of the program to accept MIDI numbers.

Key Variable Descriptions

tunelength	Total number of quantised steps in the tune
tempo	Number of beats per minute
beat_time	This is the duration of the quantised steps in microseconds and is derived from tempo. This is the value used to set the interrupt timer and acts as the main time reference for the tune.
string_num	Indicates the string number the motor picks
next_beat	This is used as a flag monitor if the interrupt timer has triggered to indicate the start of the next time step

Global Function Description

Loadbeat(unsigned char Note)	Loads the assigned motor pin number for the corresponding MIDI value
Playbeat()	Rotates the motor corresponding to the string number
Start_Beat_Timer()	Configures and starts the internal Timer to count for one quantised beat time
ISR(TIMER1_OVF_vect)	Interrupt subroutine called when Timer1 expires

The code was then tested with the MIDI note numbers for the song ‘Hall of the Mountain King’. Due to the absence of the fretting mechanism the tune produced is not easily recognisable and instead, the strings are open picked according to which string would be held. As discussed in Section 3.5, bass guitars have the option of playing multiple versions of one note. For the project, the notes were selected so that each of the strings would play at least once to show the

functioning of the mechanism. The MIDI number values range from 35-50 and the music notes that correspond to it were arbitrarily decided following the theory discussed in section 3.5. Table 7.1 shows the MIDI numbers and the notes and string that correspond to them.

Table 7.1: Midi numbers corresponding to bass guitar string.

MIDI number	Note	String	MIDI number	Note	String
35	B	A	43	G	E
36	C	A	44	G#	E
37	C#	A	45	A	E
38	D	A	46	A#	A
40	E	E	47	B	G
41	F	E	49	C#	G
42	F#	E	50	D	D

7.4.3. Push button implementation

The addition of the push button to the circuit is for the project to stand-alone as well as to integrate with the orchestra when needed. The code uses the same algorithms as discussed in Section 7.3.3. The code constantly checks for the push button state to go high before triggering the main code. Once the push button is high, code steps all the way through until it reaches the end. Once completing an entire iteration, the code checks for the push button state and does not do anything until it is pressed again. The code description follows the flowchart in Figure 7.9 and the entire code for the picking mechanism can be found in Appendix 8.

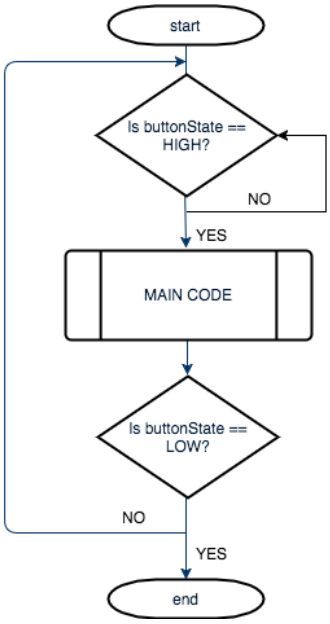


Figure 7.9: Flow chart of the program with a push button.

8. Practical Development of Fretting Mechanism

The completed sections of the fretting mechanism have been detailed and discussed in this section. As stated in Section 3, Aims and Objectives, this section of the project was aspirational and only began after the picking mechanism was complete. Due to this reason, the progress of the mechanism consisted of the mechanical design of the mechanism, the circuit design of the solenoid and the program to control the solenoid. The testing of this mechanism has not been done, however, details of how to implement and test along with the integration of both mechanisms can be found in Section 12, Future works.

8.1. Circuit Design and Testing of Solenoid

The circuit design for the solenoid was found from an online tutorial [27]. The schematic diagram in Figure 8.1 was used to test the functionality of the solenoid. The program written to control the solenoid is a modified version of the Arduino's Blink example program. The code tested for this circuit can be found in Figure 8.2.

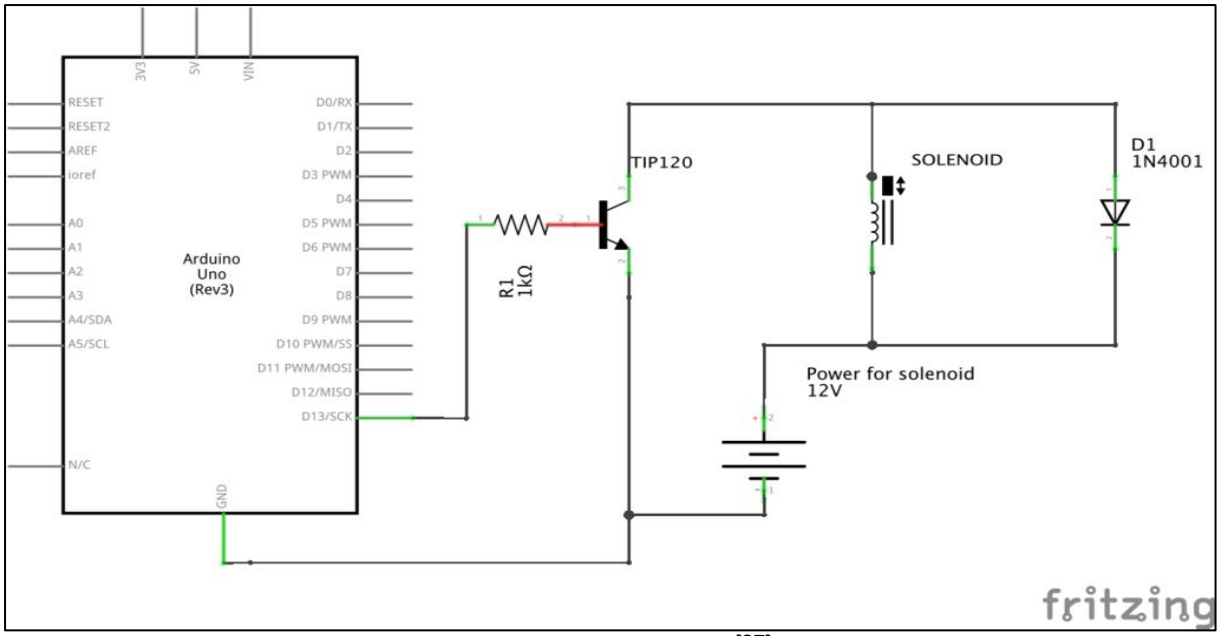


Figure 8.1: Schematic design of the solenoid circuit [27].

```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.
  This example code is in the public domain.
  modified 8 May 2014
  by Scott Fitzgerald
  modified 2 Sep 2016
  by Arturo Guadalupi
*/

/* The blink code is modified to push and pull the solenoid
   by initialising it high and then low respectively
*/

int solenoidPin = 4;    //This is the output pin on the Arduino we are using
int resetPin = 2;
void setup() {
  // put your setup code here, to run once:
  pinMode(solenoidPin, OUTPUT);           //Sets the pin as an output
  digitalWrite (resetPin, HIGH);
  pinMode(resetPin, OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
 if(resetPin == HIGH){
  digitalWrite(solenoidPin, HIGH);    //Switch Solenoid ON
  delay(5000);                        //Wait 1 Second
  digitalWrite(solenoidPin, LOW);     //Switch Solenoid OFF
  delay(1000);                        //Wait 1 Second
 }
 else if(resetPin == LOW){
  digitalWrite(solenoidPin, LOW);
 }
}
```

Figure 8.2: Code for the solenoid [28].

8.2. Design of fretting mechanism

The fretting mechanism is designed so that the finger can be used to exert a large force over a small distance at one end of the lever by exerting a small force over a greater distance at the other end [29].

The load force required for the finger to push down on to the string can be calculated knowing the tension of the string, and the distances between the solenoid and the fulcrum and the end of the finger and fulcrum [29].

The calculations for the force of the solenoid required for the mechanical design was not calculated due to discrepancies in acquiring the tensions of the string. Conversely, the general design of the fretting was done using CAD software. The base of the mechanism and the finger are shown in Figure 8.3 and 8.4 respectively.

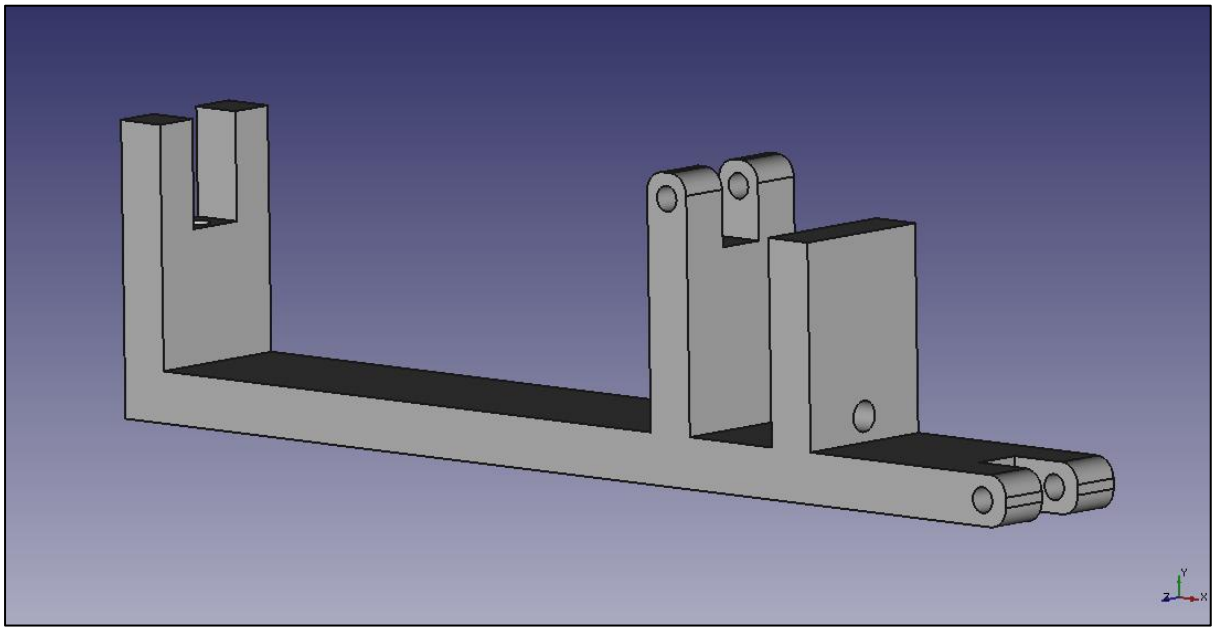


Figure 8.3: Mechanical design for the base of the fretting mechanism.

Figure 8.4: Mechanical design of the finger to fit across the base.

9. Integration with the Robot Orchestra

The integration of the mechanism with the orchestra involves replacing the push button for the robotic conductor relay. The two pins of the relay are connected to the Arduino +5V and Pin 2, thereby interchanging with the push button. The push button is designed as a separate circuit that connects on to the main circuit. This is so that when needed for the Manchester Robot Orchestra, the mechatronic bass guitar can be connected in the place of the push button circuit. The complete circuit for the integration can be seen in Figure 9.1.

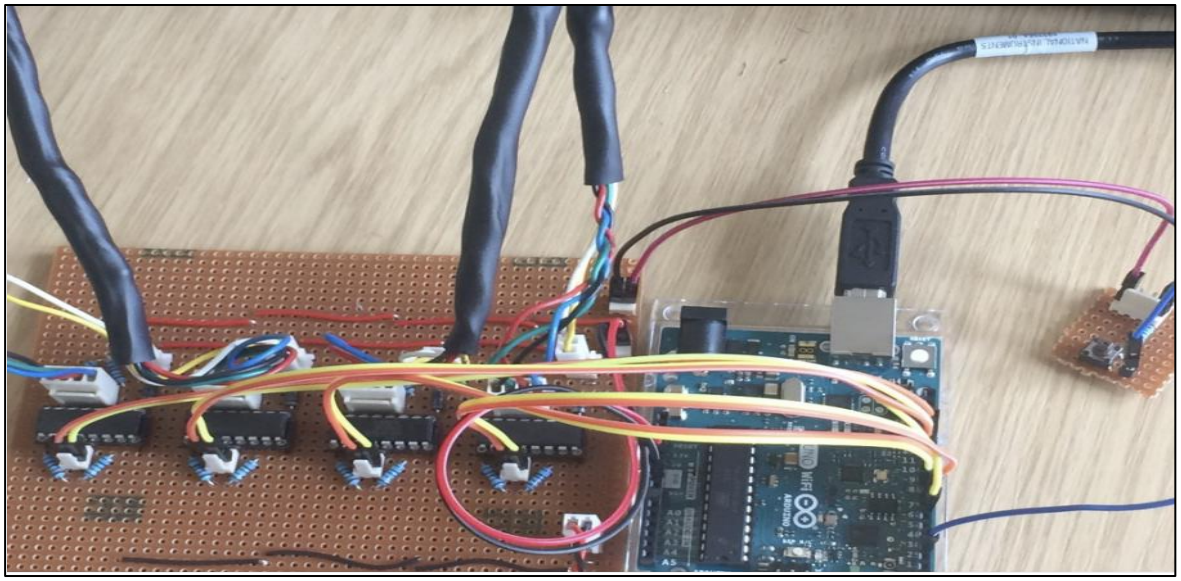


Figure 9.1: Complete circuit with indicators of relays connections of the robotic conductor.

10. Engineering Project Management

10.1. Project Plan

The project was carried out for the duration of an academic year, with weekly meetings held to discuss the progress. During the first half of the project, a Gantt chart was created to set out a definitive plan for the completion of this report. Although the project was completed to a reasonable level, unforeseen circumstances which resulted in delays for the project needing to be addressed. The initial Gantt chart took these circumstances into consideration by assigning time for any contingency. Due to these considerations the overall project was not affected. However, the Gantt chart did consider the ambition of the fretting

mechanism and so an updated Gantt chart as a result of the changes in the project can be found in Appendix 2. This is in contrast to the original Gantt chart, which can be found in Appendix A1 of the Progress report in Appendix 1.

10.2. Health and Safety Risk Assessment

The health and safety risk assessment has been maintained throughout the project. As the project involved soldering, precautions were taken to ensure there were no accidents. Soldering took place in the laboratory, which had fumes extractors to avoid inhalation of toxic fumes. The majority of the project was carried out in an office environment. Cables were secured to the floor and put away after use. When working on the computer, a screen distance of 40cm was kept and frequent breaks were taken to avoid damaging the eyes. The Risk assessment can be found in Appendix 3.

11. Conclusion

The project was split into two stages: the picking mechanism and the fretting mechanism. The objectives of the project were defined in specific stages to ensure that in the case of any unforeseen circumstances the project will still produce a working instrument. The picking mechanism was defined as the main part of the project with the fretting mechanism being reliant on the completion of the former. The final working of the project showed an autonomous bass guitar as an independent instrument as well as its capacity to be integrated into the Manchester Robot Orchestra.

The report discusses the existing literature and solutions on various musical instruments as well as background information on guitar theory and MIDI. The theoretical development of the project aided in the selection of stepper motors and controller board that is used during the entire project. The choice of Arduino Uno over the PIC was a critical decision as it saved a lot of time in the programming aspects of the project.

Throughout the project, a number of design attempts for the picking mechanism were made and improved due to the size of the stepper motors and width of the extended shaft. The final mechanical design proved to be successful in holding

stepper motors over the strings. The circuit design for the stepper motors controlled the motors with two-wired outputs. The design was tested on a breadboard before being assembled on the strip board. The additional push button circuit to trigger the stepper motors autonomously was designed so that it could replace the robotic conductor's relay without any changes to the program. The tests for the stepper motor proved challenging as fault-finding and code debugging had to be done several times before the instrument was successful.

As the fretting mechanism was an aspirational stage of the project, the functionality test of the solenoid and the mechanical design of the fretting mechanism were achieved. The mechanical design of the fretting mechanism proved to be a challenge as the design idea complex in terms of implementation. Nevertheless, a single fretting base unit fixed to the neck of the guitar was designed with a 3-D printed finger to hold down the string when pivoted by a solenoid from the opposite end. The time restriction of the project meant that only the picking mechanism was completed, leaving the fretting mechanism to be continued as future work.

The initial Gantt chart was modified along the way, with respect to the progress and the final Gantt chart was found to only vary slightly as the scope of the project was initially underestimated.

The skills applied to this project were gained from three years of extensive teaching modules which formed part of the mechatronics course. The modules that were necessary for the development and completion of the project are listed in Table 11.1.

Although this project was challenging and creative based, the skills applied and gained enable a wider practical development of all the concepts as well as support future learning and work prospects.

Table 11.1: Skills acquired from module from the course.

Modules	Skills acquired
Electronics Project	Soldering, Fault-finding
Circuit Analysis	Basic circuit theory
Electronic Circuit Design	Concepts on diodes, transistors, amplifiers, switch theory
Java Programming	Foundation, problem-solving, data structures, algorithms and good program design in Java
Applied Mechanics and Industrial Robotics	Introduction to industrial robotics and materials and structures used in mechanical designs
Microcontroller Engineering II	Operation, programming and application of the standard peripheral interfaces of modern microcontrollers, in particular, PIC
Embedded systems project	Motor characterisation, PWM and PIC programming, Engineering drawing and project management
Mechatronics Design and Analysis	Understanding of actuators and stepper motors in particular.
High speed digital and mixed signal design	Application of appropriate techniques for the design of electronic hardware.
Mobile Robots and Autonomous Systems	Robotics characterisation of actuators

12. Future Works

12.1. Picking mechanism

Although the picking mechanism produced a working autonomous bass guitar, the mounting of the mechanical frame was only temporary. A more permanent solution is needed to mount and dismount the mechanism. The mechanism can be fastened to the sides of the bass guitar using screws. However, care must be taken to avoid damaging the internal circuitry of the guitar.

12.2. Fretting mechanism

The fretting mechanism, due to the time restraints, was not assembled and tested. This can be continued, however, as a future work. The future works will consist of the assembled and tested circuit and the calculations and assembling of the mechanical design. The integration of the picking and fretting mechanism will need to be done once the fretting mechanism is completed. The mechanical design and solenoid testing for the fretting mechanism have been completed and can be continued to improve the current mechatronic bass guitar.

12.3. Integration of the mechanism

An idea for the integration of both mechanisms is the use of the Arduino Mega. The Arduino mega has fifty-four digital input/output pins to control the four stepper motors, up to forty-five individually controlled solenoid as well as connect to the push button circuit. There is barely any difference between the Arduino Uno and Arduino Mega so transferring the picking mechanism to the Mega controller should not affect the mechanism.

13. References

[1] Anon, (n.d.). The Manchester Robot Orchestra. [online] Available at: 1. http://www.robotorchestra.co.uk/ [Accessed 2 May 2017].

[2] Kapur, Ajay. "A history of robotic musical Instruments". N.p., 2005. http://www.mistic.ece.uvic.ca/publications/2005_icmc_robot.pdf. 7 [Last Accessed Apr. 2017].

[3] Batula, A.M. Kim, Y.E. "Development of a mini-humanoid pianist," 2010, 10th IEEE-RAS International Conference on Humanoid Robots, Nashville, TN, 2010, pp. 192-197. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5686330&isnumber=5686265

[4] Hoffman, G. Weinberg, G, "Gesture-based human-robot Jazz improvisation," Robotics and Automation (ICRA), 2010, IEEE International Conference on, Anchorage, AK, 2010, pp. 582-587. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5509182&isnumber=5509124

[5] J. Solis, Koichi Taniguchi, Takeshi Ninomiya, Tetsuro Yamamoto and Atsuo Takanishi, "Development of Waseda flutist robot WF-4RIV: Implementation of auditory feedback system," Robotics and Automation, 2008, ICRA 2008. IEEE International Conference on, Pasadena, CA, 2008, pp. 3654-3659. URL:http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4543771&isnumber =4543169

[6] R J. McVay, D.A. Carnegie, “MechBass: A Systems Overview of a New Four-Stringed Robotic Bass Guitar”, URL: https://hackadaycom.files.wordpress.com/2012/11/james-mcvay-mechbass-final.pdf

[7] Vindriis, R. and Carnegie, D. (2016). StrumBot – An Overview of a Strumming Guitar Robot. [online] Available at: http://www.nime.org/proceedings/2016/nime2016_paper0030.pdf [Accessed 2 May 2017].

[8] Jackson, A. (2013). University of Leeds Controls Electric Guitar with NI myRIO. [online] YouTube. Available at: https://www.youtube.com/watch?v=ZpsrZv2wUF4 [Accessed 2 May 2017].

[9] Hallé Orchestra. (2017). Robot Orchestra Performance as Part of ESOF | Hallé Orchestra. [online] Available at: http://www.halle.co.uk/robot-orchestra-performance-as-part-of-esof/ [Accessed 2 May 2017].

[10] Mathers, D. 2016, Glock-o-bot Design Guide [unpublished documentation]

[11] Anvilstudio.com. (n.d.). Anvil Studio | Free music composition, notation & MIDI-creation software. [online] Available at: http://www.anvilstudio.com/ [Accessed 2 May 2017].

[12] L. Birglen and C. M. Gosselin, "On the force capability of underactuated fingers," Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on, 2003, pp. 1139-1145 vol.1. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1241746&isnumber=27829 11

[13] "How To Play Bass Guitar". National Guitar Academy. Web. 2 May 2017.Available: https://nationalguitaracademy.com/how-to-play-bass-guitar/

[14] Casabona, Helen, David Frederick, and Brent Hurtig. What Is MIDI?. 1st ed. Cupertino, Calif.: GPI Publications, 1988. Print.

[15] Electronics.dit.ie. (2017). MIDI Note Numbers for Different Octaves. [online] Available at: http://www.electronics.dit.ie/staff/tscarff/Music_technology/midi/midi_note_numbers_for_octaves.htm [Accessed 2 May 2017].

[16] Arduino.cc. (2017). Arduino - Home. [online] Available at: https://www.arduino.cc/ [Accessed 2 May 2017].

[17]" Arduino.cc. (2017). Arduino - ArduinoBoardMega2560. [online] Available at: https://www.arduino.cc/en/Main/arduinoBoardMega2560 [Accessed 2 May 2017].

[18] Arduino.cc. (2017). Arduino - ArduinoBoardUno. [online] Available at: https://www.arduino.cc/en/main/arduinoBoardUno [Accessed 2 May 2017].

[19] Durovic, S. 2016. Actuators. EEEN30054. Mechatronics Analysis & Design. The University of Manchester

[20] Arduino tut –S. M, "Arduino - StepperOneRevolution", Arduino.cc, 2015. [Online]. Available: https://www.arduino.cc/en/Tutorial/StepperOneRevolution. [Accessed: 05- Nov- 2016].

[21] Tigoe.net. (n.d.). Stepper Motors | code, circuits, & construction. [online] Available at: http://www.tigoe.net/pcomp/code/circuits/motors/stepper-motors/ [Accessed 6 Nov. 2016].

[22] Astrosyn, “Stepper Motor”, MY5602, http://www.farnell.com/datasheets/30484.pdf?_ga=1.217808470.1355527149.14758 47027.

[23] Nanotech, “Stepper Motor”, SP2575M0206-A , http://www.farnell.com/datasheets/1702882.pdf?_ga=1.38410339.1355527149.1475 847027, March 2007.

[24] "Arduino - Stepperunipolarcircuit". Arduino.cc. https://www.arduino.cc/en/Reference/StepperUnipolarCircuit. [Last Accessed: 7 Nov. 2016.]

[25] Arduino.cc. (2017). Arduino - Button. [online] Available at: https://www.arduino.cc/en/tutorial/button [Accessed 3 May 2017].

[26] Vidme. (2017). Vidme - discover and upload original ad-free videos. [online] Available at: https://vid.me/ [Accessed 2 May 2017].

[27] Instructables.com. (n.d.). [online] Available at: http://www.instructables.com/id/Controlling-solenoids-with-arduino/ [Accessed 2 May 2017].

[28] Fitzgerald, S. and Guadalupi, A. (2014). Arduino - Blink. [online] Arduino.cc. Available at: https://www.arduino.cc/en/tutorial/blink [Accessed 2 May 2017].

[29] Engineeringtoolbox.com. (n.d.). Levers. [online] Available at: http://www.engineeringtoolbox.com/levers-d_1304.html [Accessed 2 May 2017].

Appendix 1 – Progress Report

An Autonomous Mechatronic Bass Guitar for the Manchester Robot Orchestra

Third Year Individual Project – Progress Report

Nov 2016

Denise D'Souza

9133500

Supervisor: Professor Danielle George

School of Electrical and Electronic Engineering

Contents

1. Introduction

This is an outreach project funded by the Manchester Robot Orchestra to design an autonomous mechatronics instrument. The scope of this project will involve linking mechatronics, embedded systems, programming, circuit design, sensing and actuation with a creative and innovative instrument design.

A paper on the history of robotic instruments by Ajay Kumar talks about how innovators in academics, entertainment and art have been designing musical robots using algorithms and design schemes. He defines a robotic musical instrument as a sound-making device that creates music with the use of mechanical parts, such as motors, solenoids, and gears. Crucial skills for this interdisciplinary art form include the knowledge of acoustics, electrical engineering, computer science, mechanical engineering, and machining. [1]

The instrument used for this project is a bass guitar which consists of four strings; E, A, D, and G strings. Each string when held on a fret and simultaneously picked or strummed will produce a musical note. The design of this system will be constructed in two stages: the picking mechanism and the fretting mechanism. The picking mechanism developed will consist of four stepper motors, one for each string, with a plectrum attached to it. The fretting mechanism will develop on existing robotic arm control and be replicated with four robotic fingers to press down on each individual string. The program for this instrument will accept and convert MIDI files into relevant MIDI values to control when each mechanism will play.

Aside from being designed as a stand-alone autonomous mechatronic bass guitar, the finished project will also be a part of the Manchester Robot Orchestra, where an external robotic conductor will trigger the instrument along with several other robotic instruments.

The motivation behind this project is based on the outreach aspect as well as the mechatronics aspects that involve mechanical design, electronic design, and programming. This helps in enabling a wider practical development of the concepts covered in the mechatronics degree module as well as supporting future learning and work prospects.

2. Project Aims and Objectives

2.1. Aim

The aim of this project is to produce a working mechatronics system that will autonomously play the bass guitar.

2.2. Objectives

The objectives of this project are to complete the tasks for the first stage of the project, the picking mechanism, before moving on to the second stage, the fretting mechanism.

The objectives for the picking mechanism are as follows:

- Design and test the functioning of one stepper motor for a single string.
- Design and test the functioning of two stepper motors for two strings.
- Program the Arduino board to accept MIDI values and control the two stepper motors independently.
- Design and test the functioning of four stepper motors for each string.
- Program the Arduino board to accept MIDI values and control the four stepper motors independently.
- Program the code to be triggered by an input from the robotic conductor.

The objectives for the fretting mechanism are as follows:

- Study existing robotic hand control suitable to hold down strings.
- Modify the design to control the fretting mechanism.
- Program the fretting mechanism along into the code from the picking mechanism to accept MIDI values.
- Program the code to be triggered by an input from the robotic conductor.

3. Consideration of existing literature/solutions

3.1. Existing literature on other instruments

Prior to selecting the bass guitar as the instrument to play autonomously, research on various other instruments was done to come to a clear conclusion about what

instrument was the best to use. Ideas for an instrument were taken from research done on existing literature. The following are some of the developed projects that were studied:

- Development of a mini-humanoid pianist: This was a prototyping platform to develop algorithms in a small-scale environment before moving to full-size humanoid. [2]
- Gesture-based human-robot Jazz improvisation: The project involves an interactive robotic marimba player that improvises in real-time while listening to and building upon a human performance. [3]
- Research on anthropomorphic Waseda flutist robot: The proposed system was composed of a music expressive generator, feed-forward air pressure control system with a pitch evaluation module.[4]

3.2. Existing literature on Bass Guitar

When reviewing the literature on robotic instruments, papers specifically on guitars and bass guitars were extensively looked into. These papers helped in the design phase of the project.

3.2.1. MechBass and StrumBot

MechBass and StrumBot are robotic guitars designed at the Victoria University of Wellington. The aim of the projects was to develop a standalone six string robotic guitar, consisting of mechanisms designed to enable musical expressivity and minimise acoustic noise[5] and construct a four-stringed modular robotic bass guitar[]respectively. The projects use stepper motors to hold plectrums and pluck single string units, which are electronically independent of the others [6]. These projects have helped in the design phase of the project to individually pick the different strings on the guitar with stepper motors.

3.3. Existing literature on robotic fingers

The initial review of existing literature on robotic figures proved to be very informative. The decision of the fingers being more robotic than humanoid means the fingers can have fewer degrees of freedom (DOF). A paper on the force capability of under actuated fingers states that the concept of under actuation in robotic fingers allows the hand to adjust itself to an irregularly shaped object without complex control

strategy and numerous sensors. This is through the use of springs and mechanical limits, resulting in fewer actuators than DOF.[7]

Further research into robotic fingers will be done subsequent to the completion of the prior objectives.

4. Theoretical Development

The theoretical development of the project covers research in the selection of the instrument, investigation into choosing the suitable controller board to control stepper motors, and research in the robotic hand. The progress of this development has taken into consideration and put into a Gantt chart, which can be found in Appendix A2.

4.1. Achieved Development

4.1.1. Instrument Selection

The type of instrument was first considered before the project was defined. A number of instruments were considered to be autonomously designed to play instruments such as the keyboard, xylophone, and a few woodwind instruments as well as the bass guitar. The drawbacks in the keyboard and xylophone were that the robot Orchestra was in possession of similar instruments. Woodwind instruments were not achievable due to the time constraint of the project. The research done on these instruments led to the decision of developing a mechatronic bass guitar.

4.1.2. Background working of stepper motors

Once the instrument was identified, a more detailed and definitive plan was formulated keeping in mind the time constraints and resources available.

From the Arduino Tutorials, stepper motors can be controlled to a high degree of accuracy without any feedback mechanisms. The shaft of a stepper motor is mounted with a series of magnets and is controlled by a series of electromagnetic coils that are charged positively and negatively in a specific sequence, thereby moving it forwards or backwards in small precise steps.[8]

From Northwestern University's mechatronic lab script, unipolar stepping motors

contain 5 or 6 wires and are usually wired with a centre tap on each of two windings. Bipolar stepping motors are constructed with the exact same mechanism as unipolar motors, but the two windings are wired more simply with no centre taps (Appendix A3).This makes the motor itself simpler, but the drive circuitry needed to reverse the polarity of each pair of motor poles is more complex [9]. For this reason, the unipolar mode was considered over bipolar mode.

The motors available for testing were the MY5602 Stepper Motor and the SP2575M0206-A Stepper Motor. The datasheets for both motors indicate the wiring connections; technical information and product dimensions, which help when wiring the circuits and in the designs for the mechanical structures that mount the motors and plectrums. The two stepper motors have very different maximum torque values; MY5602 Stepper Motor has a torque maximum of 8N-cm[10] while the SP2575M0206-A Stepper Motor has a torque maximum of 1.6N-cm [11]. Once the design for the plectrum mounts is completed and tested the load torque for the motor will be calculated, and the appropriate stepper motor will be selected.

According to a stepper motor website, the position of the stepper depends on how many degrees per step[12]. From the datasheets[10][11], both steppers have a 1.8-degree stepper. Therefore for a complete 360 degree rotation, 200 steps are required (1.8 x 200 degrees = 360 degrees).[12]

4.1.3. Research into controller choice

The theoretical side of the project investigated the use of two different microcontrollers: The University's microcontroller which uses the Microchip 18F8722 (PIC) and the Arduino Uno to control stepper motors.

The circuit and theory behind the control of stepper motor with the Arduino is from the Arduino tutorial. The tutorial details the circuit wiring and code to rotate the shaft in both directions. The same concept is used to test the working of the stepper motor with the PIC board, with background knowledge gained from modules from previous years.

4.1.4. Research on Robotic fingers

The review of existing literature has generated the idea of designing 2-DOF for each robotic finger. The 2-DOF will have two revolute joints moving it upwards or downwards on to the fret and left or right along the fret. More research on the fretting mechanism will be undertaken once the set objectives for the picking mechanism are complete.

4.2. Planned Development

4.2.1. Picking Mechanism

After the testing of the ongoing development (mentioned in the next section) is complete, the load torque of the plectrum holder for the stepper motor will be calculated to choose the suitable motor.

4.2.2. Fretting Mechanism

Once the basics of the picking mechanism are completed, research on the fretting mechanism will become the main focus of the project. The research on robotic hand control that has been done so far is in the early stages due to the completion of current objectives. There will be more detailed objective for the fretting mechanism in the coming weeks.

5. Practical Development

The practical development of this project discussed the progress made till date; the ongoing progress as well as the future progress of the project. All the progress discussed in this section has been included in the Gantt chart of the project, which can be found in Appendix A2. Due to the practical nature of this project, a Health and Safety Risk Assessment has been filled out and is included in Appendix A1.

5.1. Achieved Development

5.1.1. Arduino testing for each stepper motor

Using the Arduino Tutorial ‘Stepper One Revolution’ [8], the ‘MY5602 stepper motor and the SP2575M0206- A stepper motor were wired and programmed to rotate the shaft a full rotation clockwise and then a full rotation anti-clockwise.

The components required were:

- Arduino Uno Board
- Stepper Motor
- ULN2003A Bipolar Transistor Array
- 10K Potentiometer
- 12V Power supply
- Jumper cables
- Breadboard

Each stepper motor was connected as shown in the circuit diagram in Figure 5.1.The schematic diagram of the circuit can be found in Appendix A4. The datasheets for each stepper motor were used to identify the correct wiring.

The stepper motor wires are connected to the output of the transistor and the output pins 8 through 11 on the Arduino are connected to the transistor's input pins. The 10k potentiometer is connected to power and ground with its wiper outputting to analogue pin 0 on the Arduino.

The code for the Arduino initialises the stepper library on pins 8 through 11, and by increasing the speed in terms of radians per minute (rpm), the shaft will spin faster. Using the existing Arduino library files written for stepper motors (Stepper.h), the 'myStepper.step' function sets 200 times to complete a full revolution. The Arduino code can be found in Appendix A4.

The conclusion drawn after both motors were connected and tested is that the Arduino code to control the stepper motor is efficient and accurate. This allows for quick programming and setting up of multiple stepper motors, which makes it an ideal choice of controller for this project.

5.1.2. PIC testing control for each stepper motor

When testing the stepper motors with the PIC controller, the theory for the PIC control researched was applied to wiring the circuit and writing up the program to rotate shaft a full rotation clockwise and then a full rotation anti-clockwise on the pressing of two pushbuttons respectively.

The components required were:

- Microchip 18F8722
- Stepper Motor
- ULN2003A Bipolar Transistor Array
- 2x Pushbuttons
- 1N4004 diode
- 7.5V Power supply
- Jumper cables
- Breadboard

Each stepper motor was connected as shown in the circuit diagram.The schematic diagram of the circuit can be found in Appendix A5. The datasheets [13] for each stepper motor were used to identify the correct wiring connections as shown in Figure Appendix A3.

Similar to the Arduino circuit, the motor wires are connected as seen in the schematic diagram in Appendix A4 to the output of the transistor while the input is connected to output pins of Port B of the PIC, using a breakout board. The push buttons are connected to input pins on Port B with PB1 rotating the motor clockwise and PB2 rotating anticlockwise.

The code written for the PIC sets four PORTB pins as outputs and uses the stepping sequence from Appendix A3 to control the motor polarity. On pushing either pushbutton, the stepping sequence is triggered and the motor runs either in clockwise or anti-clockwise. The code for the PIC can be found in Appendix A5.

The tests for the motors on the PIC were not as expected as the motors did not run. This could be due to a possible error in the program. Further changes to the code and circuit were attempted. However, the results were not as required.

5.1.3. Comparison of controllers

Subsequent to the testing of both controller boards the following decisions were made:

- The Arduino circuit was accurate each test time and the prebuilt library files and code make it efficient to control more steppers.
- If only the picking mechanism was being developed, the use of the PIC would be preferred. However, due to the time constraint it was decided that the Arduino would be best to use.

5.2. Ongoing Development

As of the completion of the progress report, there are two on-going developments taking place.

5.2.1. Testing with two stepper motors

The testing of two stepper motors to work simultaneously involves the duplication of the Arduino circuit tested for one stepper motor. The code will be similar to that of the single motor. However, it will include four additional output pins for the second motor.

5.2.2. Mechanical structure

The design for the stepper motor is in progress. The layout will have the motors sitting at the end of the body of the guitar but staggered, so each motor can be fitted above each string. The shafts of the motors will be extended over the centre of the body of the guitar with a plectrum mounted onto the end.

5.3. Planned Development

5.3.1. Picking Mechanism

Once two stepper motors are working simultaneously, the next step in the project is to write the code to control the motors with MIDI values. Then focus will shift to triggering the circuit with an external input, such as Graphene. Once satisfied with the results, the circuit and code will be modified to work with four stepper motors.

5.3.2. Fretting Mechanism

Once the picking mechanism is fully functioning, the robotic finger will be designed. A

control system will be implemented so that each finger will hold and release a string on the fret when driven. Following this, the merging of the picking and fretting mechanism to be controlled simultaneously with the implementation of both mechanisms to be triggered on its own or by an external source like Graphene.

5.4. Technical risks

The technical risks that would affect this project and the actions taken to ensure that would not happen are as follows:

- Using not well supported software tools have been avoided by using the Arduino software as it is user-friendly and provides plenty of tutorials to aid in its use.
- The detail of the project is well defined, and timing contingencies have been considered, which are included in the Gantt chart.
- The ambition of this project is realistic as it has been divided into stages where each stage produces a working result.

6. Conclusion

The main tasks of this project are the designing and testing of the two stages: the picking mechanism and the fretting mechanism. The final working of the instrument will have these two mechanisms working together as a stand-alone instrument as well as part of the Manchester Robot Orchestra.

To ensure that the project will produce a working mechatronic bass guitar, the objectives are defined in specific stages so that regardless of any unforeseen complications there will still be working result.

The progress of the report discusses the various existing literature that has been considered as well as the existing solutions of the musical instrument. It details the theoretical and practical research and planning that has been put into this project so far as well as future planning and research.

All future progress will follow the plan as stated in the Gantt chart and will be consistently recorded to be put into the final report. On completion, the mechatronic bass guitar will play as a standalone instrument as well as be a part of the

Manchester Robot Orchestra where an external robotic conductor will trigger the instrument along with several other robotics instruments.

7. References

[1] Kapur, Ajay. "A history of robotic musical Instruments". N.p., 2005. http://www.mistic.ece.uvic.ca/publications/2005_icmc_robot.pdf. 7 [Last Accessed Nov. 2016].

[2] Batula, A.M. Kim, Y.E. "Development of a mini-humanoid pianist," *2010, 10th IEEE-RAS International Conference on Humanoid Robots*, Nashville, TN, 2010, pp. 192-197.
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5686330&isnumber=5686265

[3] Hoffman, G. Weinberg, G, "Gesture-based human-robot Jazz improvisation," *Robotics and Automation (ICRA), 2010, IEEE International Conference on*, Anchorage, AK, 2010, pp. 582-587.
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5509182&isnumber=5509124

[4] J. Solis, Koichi Taniguchi, Takeshi Ninomiya, Tetsuro Yamamoto and Atsuo Takanishi, "Development of Waseda flutist robot WF-4RIV: Implementation of auditory feedback system," *Robotics and Automation, 2008, ICRA 2008. IEEE International Conference on*, Pasadena, CA, 2008, pp. 3654-3659.
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4543771&isnumber=4543169

[5] R.G Vindriis and D.A Carnegie et al, “A Comparison of Pic -Based Strategies for Robotic Bass Playing”, 2011 , URL: http://machinehead.yolasite.com/resources/first_paper.pdf

[6] R J. McVay, D.A. Carnegie, “MechBass: A Systems Overview of a New Four-Stringed Robotic Bass Guitar”, URL: https://hackadaycom.files.wordpress.com/2012/11/james-mcvay-**mechbass**-final.pdf

[7] L. Birglen and C. M. Gosselin, "On the force capability of underactuated fingers," *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, 2003, pp. 1139-1145 vol.1.
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1241746&isnumber=27829

[8] S. M, "Arduino - StepperOneRevolution", *Arduino.cc*, 2015. [Online]. Available: https://www.arduino.cc/en/Tutorial/StepperOneRevolution. [Accessed: 05- Nov- 2016].

[9] "Introduction to Stepper Motors", *Mechatronics.mech.northwestern.edu*. [Online]. Available: http://mechatronics.mech.northwestern.edu/design_ref/actuators/stepper_intro.html. [Accessed: 05- Nov- 2016].

[10] Astrosyn, “Stepper Motor”, MY5602, http://www.farnell.com/datasheets/30484.pdf?_ga=1.217808470.1355527149.1475847027.

[11]Nanotech, “Stepper Motor”, SP2575M0206-A , http://www.farnell.com/datasheets/1702882.pdf?_ga=1.38410339.1355527149.1475847027, March 2007.

[12] Tigoe.net. (n.d.). *Stepper Motors | code, circuits, & construction*. [online] Available at: http://www.tigoe.net/pcomp/code/circuits/motors/stepper-motors/ [Accessed 6 Nov. 2016].

[13]"PIC18F8722 Family Data Sheet". *Microchip*. N.p., 2008. http://ww1.microchip.com/downloads/en/DeviceDoc/39646c.pdf. [Last Accessed: 7 Nov. 2016].

[14] "Arduino - Stepperunipolarcircuit". *Arduino.cc*. https://www.arduino.cc/en/Reference/StepperUnipolarCircuit. [Last Accessed: 7 Nov. 2016.]

8. Appendices

8.1. Appendix A1 – Risk Assessment

SCHOOL OF E&EE RISK ASSESSMENT

LOCATION/ACTIVITY: C34, Sackville Street Building

WORK ACTIVITY/ WORKPLACE (WHAT PART OF THE ACTIVITY POSES RISK OF INJURY OR ILLNESS)	HAZARD (S) (SOMETHING THAT COULD CAUSE HARM, ILLNESS OR INJURY)	LIKELY CONSEQUENCES (WHAT WOULD BE THE RESULT OF THE HAZARD)	WHO OR WHAT IS AT RISK (INCLUDE NUMBERS AND GROUPS)	EXISTING CONTROL MEASURES IN USE (WHAT PROTECTS PEOPLE FROM THESE HAZARDS)	WITH EXISTING CONTROLS				MEASURE REQUIRED TO PREVENT OR REDUCE RISK (WHAT NEEDS TO BE DONE TO MAKE THE ACTIVITY AS SAFE AS POSSIBLE)	PERSON RESPONSIBLE FOR ACTIONS AND AGREED TIMESCALES TO ACHIEVE THEM	WITH NEW CONTROLS			
					SEVERITY	LIKELIHOOD	RISK RATING	RISK ACCEPTABLE			SEVERITY	LIKELIHOOD	RISK RATING	RISK ACCEPTABLE
Soldering	Heat burns	Tissue damage	Students	Care should be taken when handling hot irons Iron kept in holder when not in use. Cold water applied to burns immediately.	3	1	3	Yes						
Soldering	Solder fume inhalation	Respiratory problems	Students	Use fume extraction while soldering.	3	1	3	Yes						
Handling Solder Paste	Absorption through skin	Lead Poisoning	Students	Wear gloves while handling material.	3	1	3	Yes						
Mechanical Fabrication	Injury due to drilling tools	Moderate Injury	Students	Safety bars on drilling equipment	3	2	6	Yes						
Operating electrical equipment from 240V AC mains supply	Electrical shock	Tissue damage Heart fibrillation	Students	All mains powered equipment must have a Portable Appliance Test (PAT) label in accordance with the School Safety Policy	5	1	5	Yes						

RISK ASSESSOR	NAME: Denise D'Souza	SIGNED:	Date: 7/11/2016	THIS RISK ASSESSMENT WILL BE SUBJECT TO A REVIEW NO LATER THAN: (12 months)

SCHOOL OF E&EE RISK ASSESSMENT

ACTIVITY/LOCATION: E13, Sackville Street Building

WORK ACTIVITY/ WORKPLACE (WHAT PART OF THE ACTIVITY POSES RISK OF INJURY OR ILLNESS)	HAZARD (S) (SOMETHING THAT COULD CAUSE HARM, ILLNESS OR INJURY)	LIKELY CONSEQUENCES (WHAT WOULD BE THE RESULT OF THE HAZARD)	WHO OR WHAT IS AT RISK (INCLUDE NUMBERS AND GROUPS)	EXISTING CONTROL MEASURES IN USE (WHAT PROTECTS PEOPLE FROM THESE HAZARDS)	WITH EXISTING CONTROLS				MEASURE REQUIRED TO PREVENT OR REDUCE RISK (WHAT NEEDS TO BE DONE TO MAKE THE ACTIVITY AS SAFE AS POSSIBLE)	PERSON RESPONSIBLE FOR ACTIONS AND AGREED TIMESCALES TO ACHIEVE THEM	WITH NEW CONTROLS			
					SEVERITY	LIKELIHOOD	RISK RATING	RISK ACCEPTABLE			SEVERITY	LIKELIHOOD	RISK RATING	RISK ACCEPTABLE
Office Environment	Tripping on trailing leads	Minor injury	Student and Staff	Cables secured to the floor using cable protector or adhesive tape. Cables stored away after use.	3	2	6	Yes						
Working at a PC	Prolonged use of PC	Repetitive Strain Injury (RSI) and Work Related Upper Limb Disorder (WRULD)	Student	Approved seating, desks and lighting are provided. Relevant University documents provided. Student instructedtake frequent breaks	3	2	6	Yes						
Working at a PC	Prolonged use of PC Incorrect monitor placement	Affected eye-sight posture	Student	Student instructed to make sure that the screen is kept at a distance of at least 40cm away. Student instructed to take frequent brakes.	3	2	6	Yes						
Working at PC	Mains Electrical Shock	Electrocution	Student and Staff	All mains powered equipment must have a Portable Appliance Test (PAT) label in accordance with the School Safety Policy.	2	2	4	Yes						

RISK ASSESSOR	NAME: Denise D'Souza	SIGNED:	DATE: 7/11/2016	THIS RISK ASSESSMENT WILL BE SUBJECT TO A REVIEW NO LATER THAN: (12 months)

SEVERITY VALUE = Potential consequence of an incident/injury given current level of controls.

5 Very High Death / permanent incapacity / widespread loss
4 High Major Injury (Reportable Category) / Severe Incapacity / Serious Loss
3 Moderate Injury / illness of 3 days or more absence (reportable category) / Moderate
2 Slight Minor injury / illness - immediate 1st Aid only / slight loss
1 Negligible No injury or trivial injury / illness / loss

LIKELIHOOD = what is the potential of an incident or injury occurring given the current level of control

5 Almost certain to occur
4 Likely to occur
3 Quite possible to occur
2 Possible in current situation
1 Not likely to occur

The intersection of the chosen column with the chosen row is the Risk classification.

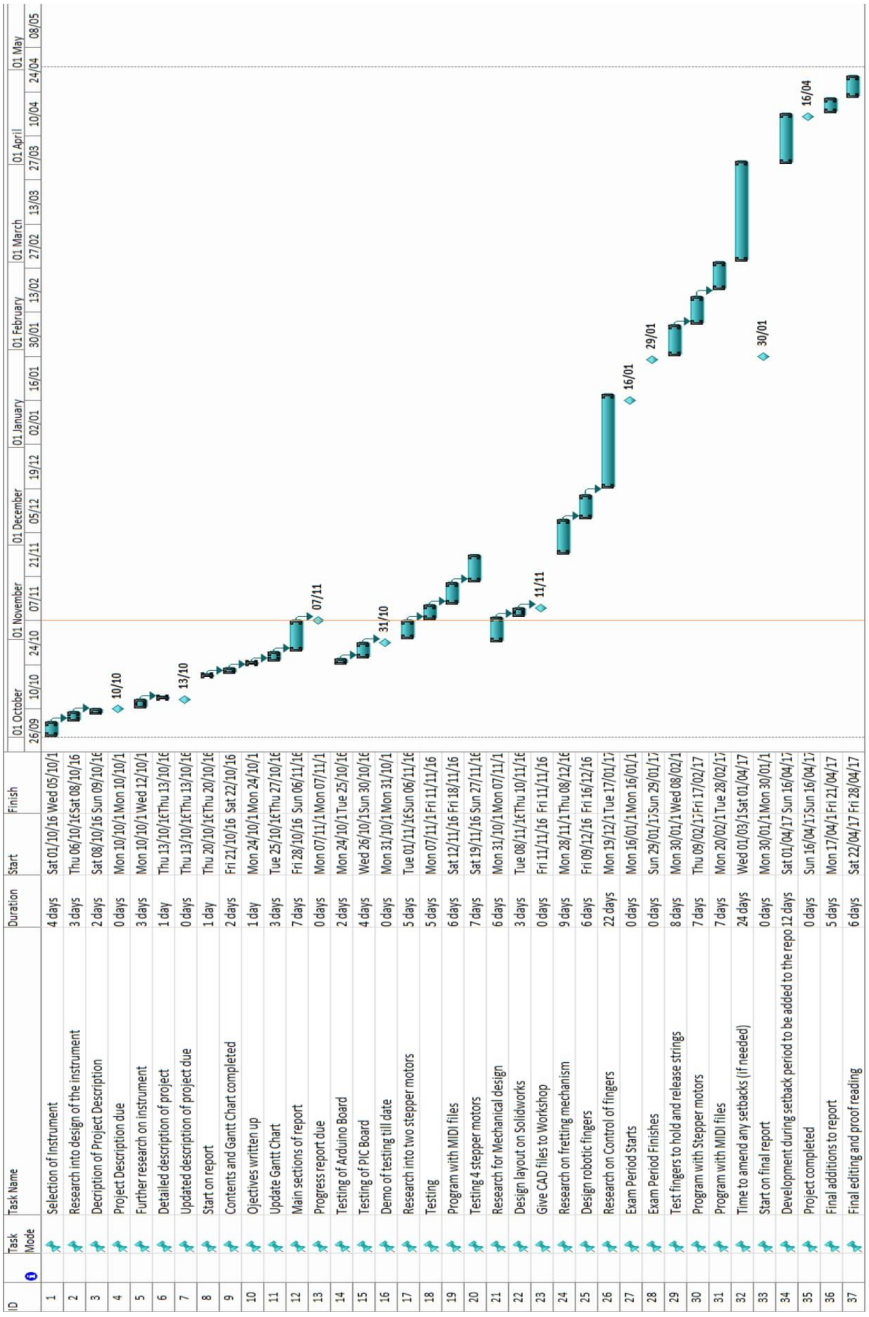
ISK SCORE Key:

1 - 5 LOW - Tolerable - Monitor and Manage

6 - 10 MEDIUM - Review and introduce additional controls to mitigate to ALARP

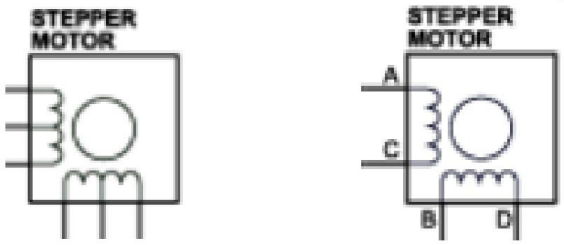
12 - 25 HIGH - Intolerable Stop Work and immediately introduce further control measures

8.2. Appendix A2 – Project Plan



ID	Task Mode	Task Name	Duration	Start	Finish
1		Selection of Instrument	4 days	Sat 01/10/16	Wed 05/10/1
2		Research into design of the instrument	3 days	Thu 06/10/16	Sat 08/10/16
3		Decription of Project Description	2 days	Sat 08/10/16	Sun 09/10/16
4		Project Description due	0 days	Mon 10/10/1	Mon 10/10/1
5		Further research on instrument	3 days	Mon 10/10/1	Wed 12/10/1
6		Detailed description of project	1 day	Thu 13/10/1	Thu 13/10/16
7		Updated description of project due	0 days	Thu 13/10/1	Thu 13/10/16
8		Start on report	1 day	Thu 20/10/1	Thu 20/10/16
9		Contents and Gantt Chart completed	2 days	Fri 21/10/16	Sat 22/10/16
10		Ojectives written up	1 day	Mon 24/10/1	Mon 24/10/1
11		Update Gantt Chart	3 days	Tue 25/10/1	Thu 27/10/16
12		Main sections of report	7 days	Fri 28/10/16	Sun 06/11/16
13		Progress report due	0 days	Mon 07/11/1	Mon 07/11/1
14		Testing of Arduino Board	2 days	Mon 24/10/1	Tue 25/10/16
15		Testing of PIC Board	4 days	Wed 26/10/1	Sun 30/10/16
16		Demo of testing till date	0 days	Mon 31/10/1	Mon 31/10/1
17		Research into two stepper motors	5 days	Tue 01/11/1	Sun 06/11/16
18		Testing	5 days	Mon 07/11/1	Fri 11/11/16
19		Program with MIDI files	6 days	Sat 12/11/16	Fri 18/11/16
20		Testing 4 stepper motors	7 days	Sat 19/11/16	Sun 27/11/16
21		Research for Mechanical design	6 days	Mon 31/10/1	Mon 07/11/1
22		Design layout on Solidworks	3 days	Tue 08/11/1	Thu 10/11/16
23		Give CAD files to Workshop	0 days	Fri 11/11/16	Fri 11/11/16
24		Research on fretting mechanism	9 days	Mon 28/11/1	Thu 08/12/16
25		Design robotic fingers	6 days	Fri 09/12/16	Fri 16/12/16
26		Research on Control of fingers	22 days	Mon 19/12/1	Tue 17/01/17
27		Exam Period Starts	0 days	Mon 16/01/1	Mon 16/01/1
28		Exam Period Finishes	0 days	Sun 29/01/1	Sun 29/01/17
29		Test fingers to hold and release strings	8 days	Mon 30/01/1	Wed 08/02/1
30		Program with Stepper motors	7 days	Thu 09/02/1	Fri 17/02/17
31		Program with MIDI files	7 days	Mon 20/02/1	Tue 28/02/17
32		Time to amend any setbacks (if needed)	24 days	Wed 01/03/1	Sat 01/04/17
33		Start on final report	0 days	Mon 30/01/1	Mon 30/01/1
34		Development during setback period to be added to the repo	12 days	Sat 01/04/17	Sun 16/04/17
35		Project completed	0 days	Sun 16/04/1	Sun 16/04/17
36		Final additions to report	5 days	Mon 17/04/1	Fri 21/04/17
37		Final editing and proof reading	6 days	Sat 22/04/17	Fri 28/04/17

8.3. Appendix A3 – Theoretical Development Tables and Figures



(a) (b)

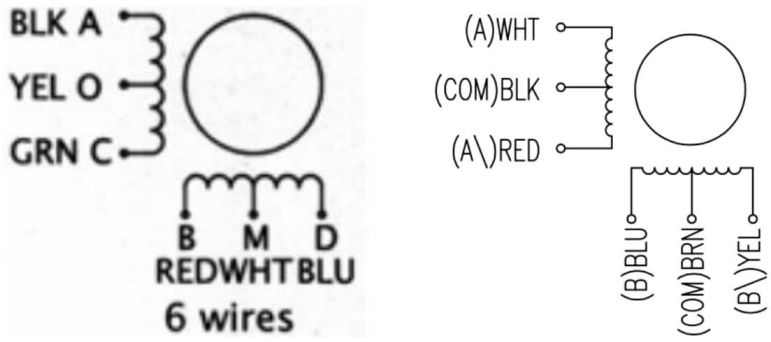
Figure 1: (a) Uniplor wiring, (b) Bipolar wiring[12]

Step	wire 1	wire 2	wire 3	wire 4
1	High	low	high	low
2	low	high	high	low
3	low	high	low	high
4	high	low	low	high

Table 1: Stepping sequence for four wires [12]

8.4. Appendix A4 – Practical Development

8.4.1. Figures



(a) (b)

Figure 1: (a) MY5602 Stepper Motor wiring [10]
(b) SP2575M0206-A Stepper Motor wiring [11]

8.4.2. Wiring diagram, schematic diagram and code for the Arduino

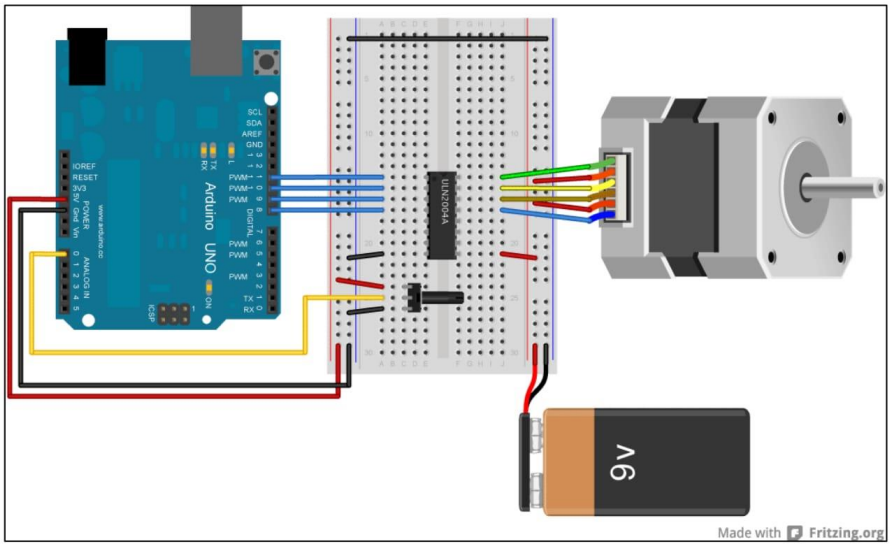


Figure 2: Wiring diagram for unipolar stepper motor [8]

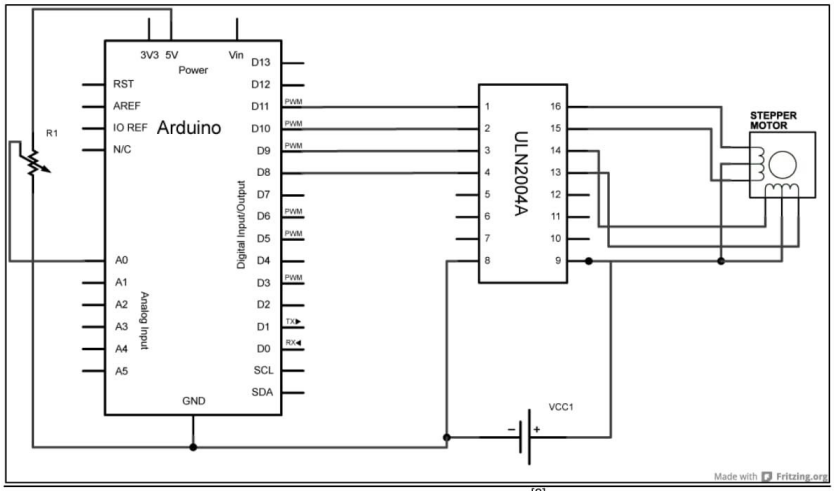


Figure 3: Schematic Diagram for unipolar stepper motor[8]

```
/*
 Stepper Motor Control - one revolution

 This program drives a unipolar or bipolar stepper motor.
 The motor is attached to digital pins 8 - 11 of the Arduino.

 The motor should revolve one revolution in one direction, then
 one revolution in the other direction.


 Created 11 Mar. 2007
 Modified 30 Nov. 2009
 by Tom Igoe

 */

#include <Stepper.h>

const int stepsPerRevolution = 200;  // change this to fit the number of steps per revolution
// for your motor

// initialize the stepper library on pins 8 through 11:
Stepper myStepper(stepsPerRevolution, 8, 9, 10, 11);

void setup() {
  // set the speed at 60 rpm:
  myStepper.setSpeed(60);
  // initialize the serial port:
  Serial.begin(9600);
}

void loop() {
  // step one revolution  in one direction:
  Serial.println("clockwise");
  myStepper.step(stepsPerRevolution);
  delay(500);

  // step one revolution in the other direction:
  Serial.println("counterclockwise");
  myStepper.step(-stepsPerRevolution);
  delay(500);
}
```

Figure 4: Code written to run stepper motor [8]

8.4.3. Schematic diagram and code for the PIC

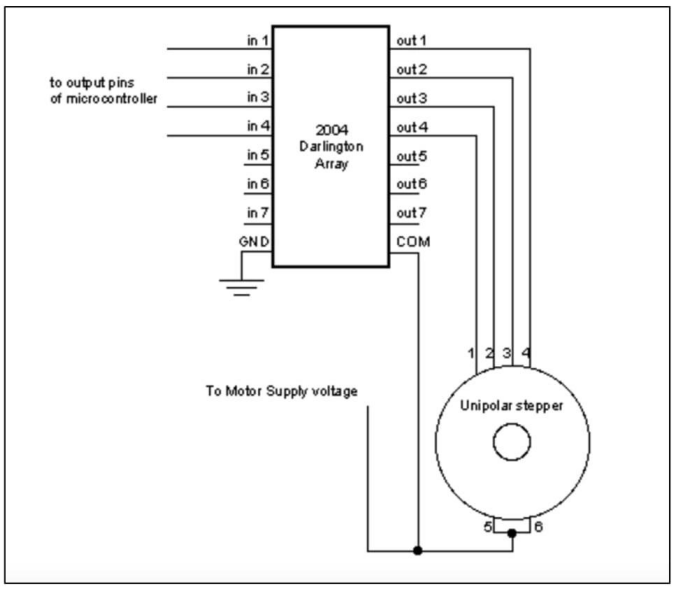


Figure 1: Schematic diagram for unipolar stepper motor[14]

Code for Unipolar stepper motor on PIC board

```
/* Program to rotate clockwise on the pushing of PB1 and to rotate anticlockwise on
the pushing of PB2 */

#include "xc.h"

void turn_clockwise (unsigned char step);

void turn_anticlockwise (unsigned char step);

void turn_off (void);

void main(void)

{

   TRISBbit.RB0 = 0;

   TRISBbit.RB1 = 0;

   TRISBbit.RB2 = 0;
```

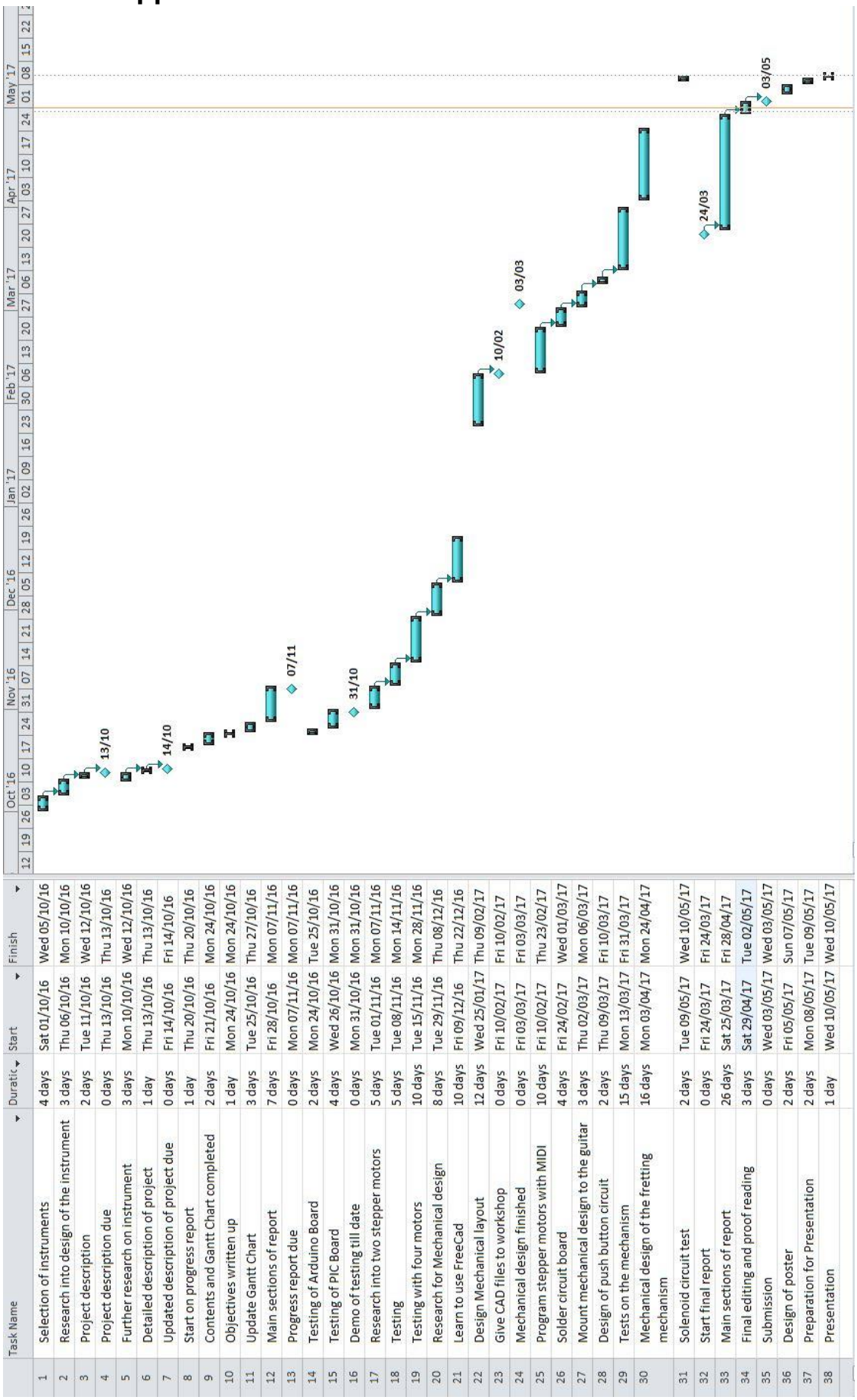
```
 TRISBbit.RB3 = 0;
 TRISBbit.RB4 = 1;
 TRISBbit.RB5 = 1;
unsigned char count = 0;
while (1)
 {
    while (PORTBbits.RB4 == 1) {       //While PB1 is pressed
       turn_clockwise (count);
      __delay_ms (10);
       count ++;
      if (count == 4)
           count = 0;
    }
    while (PORTBbits.RB5 == 1) {       //While PB2 is pressed
       turn_anticlockwise (count);
      __delay_ms (10);
       count ++;
      if (count == 4)
           count = 0;
    }
    if ((PORTBbits.RB4 == 0) && (PORTBbits.RB5 == 0))
       turn_off();
```

```
    }
      }
void turn_anticlockwise (unsigned char step) {
   switch (step) {
        case 0: LATBbits.LATB0 = 1;
              LATBbits.LATB1 = 0;
              LATBbits.LATB2 = 0;
              LATBbits.LATB3 = 0;
              break;
        case 1: LATBbits.LATB0 = 0;
              LATBbits.LATB1 = 1;
              LATBbits.LATB2 = 0;
              LATBbits.LATB3 = 0;
              break;
        case 2: LATBbits.LATB0 = 0;
              LATBbits.LATB1 = 0;
              LATBbits.LATB2 = 1;
              LATBbits.LATB3 = 0;
              break;
        case 3: LATBbits.LATB0 = 0;
              LATBbits.LATB1 = 0;
              LATBbits.LATB2 = 0;
```

```
                LATBbits.LATB3 = 1;
                break;
            default: break;
        }
}
void turn_clockwise (unsigned char step) {
    switch (step) {
            case 0: LATBbits.LATB0 = 1;
                LATBbits.LATB1 = 0;
                LATBbits.LATB2 = 0;
                LATBbits.LATB3 = 0;
                break;
            case 1: LATBbits.LATB0 = 0;
                LATBbits.LATB1 = 0;
                LATBbits.LATB2 = 0;
                LATBbits.LATB3 = 1;
                break;
            case 2: LATBbits.LATB0 = 0;
                LATBbits.LATB1 = 0;
                LATBbits.LATB2 = 1;
                LATBbits.LATB3 = 0;
                break;
```

```
            case 3: LATBbits.LATB0 = 0;
                    LATBbits.LATB1 = 1;
                    LATBbits.LATB2 = 0;
                    LATBbits.LATB3 = 0;
                    break;
            default: break;
        }
}
void turn_off (void) {
    LATBbits.LATB0 = 0;
    LATBbits.LATB1 = 0;
    LATBbits.LATB2 = 0;
    LATBbits.LATB3 = 0;
}
// End of File
```

Appendix 2 – Gantt chart



	Task Name	Duration	Start	Finish
1	Selection of instruments	4 days	Sat 01/10/16	Wed 05/10/16
2	Research into design of the instrument	3 days	Thu 06/10/16	Mon 10/10/16
3	Project description	2 days	Tue 11/10/16	Wed 12/10/16
4	Project description due	0 days	Thu 13/10/16	Thu 13/10/16
5	Further research on instrument	3 days	Mon 10/10/16	Wed 12/10/16
6	Detailed description of project	1 day	Thu 13/10/16	Thu 13/10/16
7	Updated description of project due	0 days	Fri 14/10/16	Fri 14/10/16
8	Start on progress report	1 day	Thu 20/10/16	Thu 20/10/16
9	Contents and Gantt Chart completed	2 days	Fri 21/10/16	Mon 24/10/16
10	Objectives written up	1 day	Mon 24/10/16	Mon 24/10/16
11	Update Gantt Chart	3 days	Tue 25/10/16	Thu 27/10/16
12	Main sections of report	7 days	Fri 28/10/16	Mon 07/11/16
13	Progress report due	0 days	Mon 07/11/16	Mon 07/11/16
14	Testing of Arduino Board	2 days	Mon 24/10/16	Tue 25/10/16
15	Testing of PIC Board	4 days	Wed 26/10/16	Mon 31/10/16
16	Demo of testing till date	0 days	Mon 31/10/16	Mon 31/10/16
17	Research into two stepper motors	5 days	Tue 01/11/16	Mon 07/11/16
18	Testing	5 days	Tue 08/11/16	Mon 14/11/16
19	Testing with four motors	10 days	Tue 15/11/16	Mon 28/11/16
20	Research for Mechanical design	8 days	Tue 29/11/16	Thu 08/12/16
21	Learn to use FreeCad	10 days	Fri 09/12/16	Thu 22/12/16
22	Design Mechanical layout	12 days	Wed 25/01/17	Thu 09/02/17
23	Give CAD files to workshop	0 days	Fri 10/02/17	Fri 10/02/17
24	Mechanical design finished	0 days	Fri 03/03/17	Fri 03/03/17
25	Program stepper motors with MIDI	10 days	Fri 10/02/17	Thu 23/02/17
26	Solder circuit board	4 days	Fri 24/02/17	Wed 01/03/17
27	Mount mechanical design to the guitar	3 days	Thu 02/03/17	Mon 06/03/17
28	Design of push button circuit	2 days	Thu 09/03/17	Fri 10/03/17
29	Tests on the mechanism	15 days	Mon 13/03/17	Fri 31/03/17
30	Mechanical design of the fretting mechanism	16 days	Mon 03/04/17	Mon 24/04/17
31	Solenoid circuit test	2 days	Tue 09/05/17	Wed 10/05/17
32	Start final report	0 days	Fri 24/03/17	Fri 24/03/17
33	Main sections of report	26 days	Sat 25/03/17	Fri 28/04/17
34	Final editing and proof reading	3 days	Sat 29/04/17	Tue 02/05/17
35	Submission	0 days	Wed 03/05/17	Wed 03/05/17
36	Design of poster	2 days	Fri 05/05/17	Sun 07/05/17
37	Preparation for Presentation	2 days	Mon 08/05/17	Tue 09/05/17
38	Presentation	1 day	Wed 10/05/17	Wed 10/05/17

Appendix 3 – Risk Assessment

SCHOOL OF E&EE RISK ASSESSMENT

LOCATION/ACTIVITY: C34, Sackville Street Building

WORK ACTIVITY/ WORKPLACE (WHAT PART OF THE ACTIVITY POSES RISK OF INJURY OR ILLNESS)	HAZARD (S) (SOMETHING THAT COULD CAUSE HARM, ILLNESS OR INJURY)	LIKELY CONSEQUENCES (WHAT WOULD BE THE RESULT OF THE HAZARD)	WHO OR WHAT IS AT RISK (INCLUDE NUMBERS AND GROUP S)	EXISTING CONTROL MEASURES IN USE (WHAT PROTECTS PEOPLE FROM THESE HAZARDS)	WITH EXISTING CONTROLS				MEASURE REQUIRED TO PREVENT OR REDUCE RISK (WHAT NEEDS TO BE DONE TO MAKE THE ACTIVITY AS SAFE AS POSSIBLE)	PERSON RESPONSIBLE FOR ACTIONS AND AGREED TIMESCALES TO ACHIEVE THEM	WITH NEW CONTROLS			
					SEVERITY	LIKELIHOOD	RISK RATING	RISK ACCEPTABLE			SEVERITY	LIKELIHOOD	RISK RATING	RISK ACCEPTABLE
Soldering	Heat burns	Tissue damage	Students	Care should be taken when handling hot irons. Iron kept in holder when not in use. Colder water applied to burns immediately.	3	1	3	Yes						
Soldering	Solder fumes	Respiratory problems	Students	Use fume extraction while soldering.	3	1	3	Yes						
Handling solder paste	Absorption through skin	Lead poisoning	Students	Wear gloves while handling material.	3	1	3	Yes						
Mechanical fabrication	Injury due to drilling tools	Moderate injury	Students	Safety bars in drilling equipment.	3	2	6	Yes						
Operating electrical equipment from 240V AC mains supple	Electrical shock	Tissue damage Heart fibrilation	Students	All mains powered equipment must have a Portable Appliance Test (PAT) label in accordance with the School Safety Policy	5	1	5	Yes						

RISK ASSESSOR	NAME: Denise D'Souza	SIGNED:	Date: 03/05/2017	THIS RISK ASSESSMENT WILL BE SUBJECT TO A REVIEW NO LATER THAN: (12 months)

SEVERITY VALUE = Potential consequence of an incident/injury given current level of controls.

5 Very High Death / permanent incapacity / widespread loss
4 High Major Injury (Reportable Category) / Severe Incapacity / Serious Loss
3 Moderate Injury / illness of 3 days or more absence (reportable category) / Moderate
2 Slight Minor injury / illness - immediate 1st Aid only / slight loss
1 Negligible No injury or trivial injury / illness / loss

LIKELIHOOD = what is the potential of an incident or injury occurring given the current level of control

5 Almost certain to occur
4 Likely to occur
3 Quite possible to occur
2 Possible in current situation
1 Not likely to occur

The intersection of the chosen column with the chosen row is the Risk classification.

RISK SCORE Key:

1- 5 LOW - Tolerable - Monitor and Manage

6 - 10 MEDIUM - Review and introduce additional controls to mitigate to ALARP

12 - 25 HIGH - Intolerable Stop Work and immediately introduce further control measures

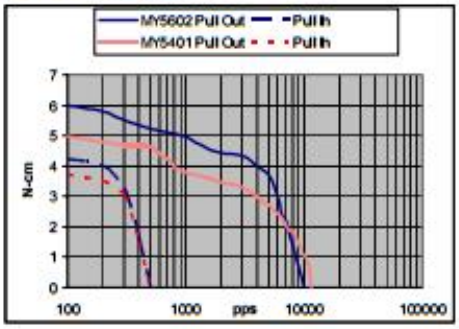
Appendix 4 – Stepper motor datasheets

MY5602 Stepper Motor

ASTROSYN
ADVANCING THE DRIVE TOWARDS MOTOR EFFICIENCY

Size 14 Mini Hybrid Stepper Motors

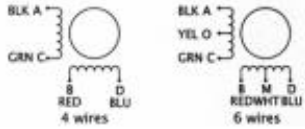
- High accuracy
- Low noise
- Low inertia



General Specifications

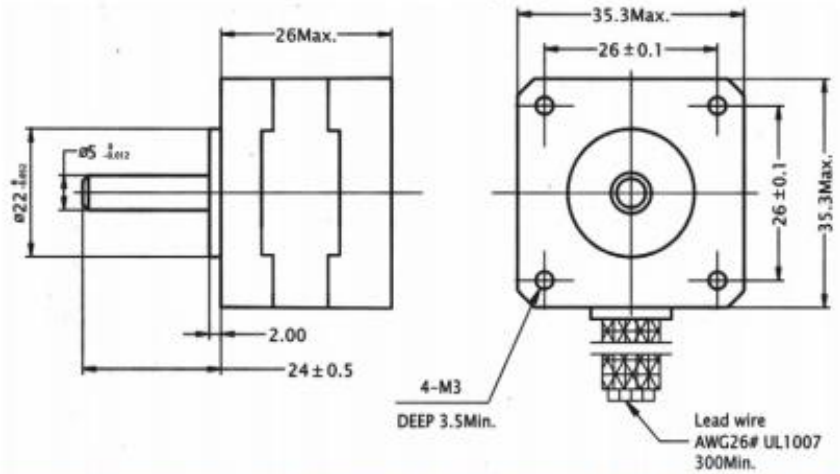
Step Accuracy:	Maximum angular deviation is ±5% of one step
Radial Play:	Maximum 0.02mm at 450g
End Play:	Maximum 0.08mm at 450g
Insulation Resistance:	Minimum 100MΩ at 500V dc
Insulation Class:	B
Temperature Rise:	Maximum 80°C

Wiring Diagram



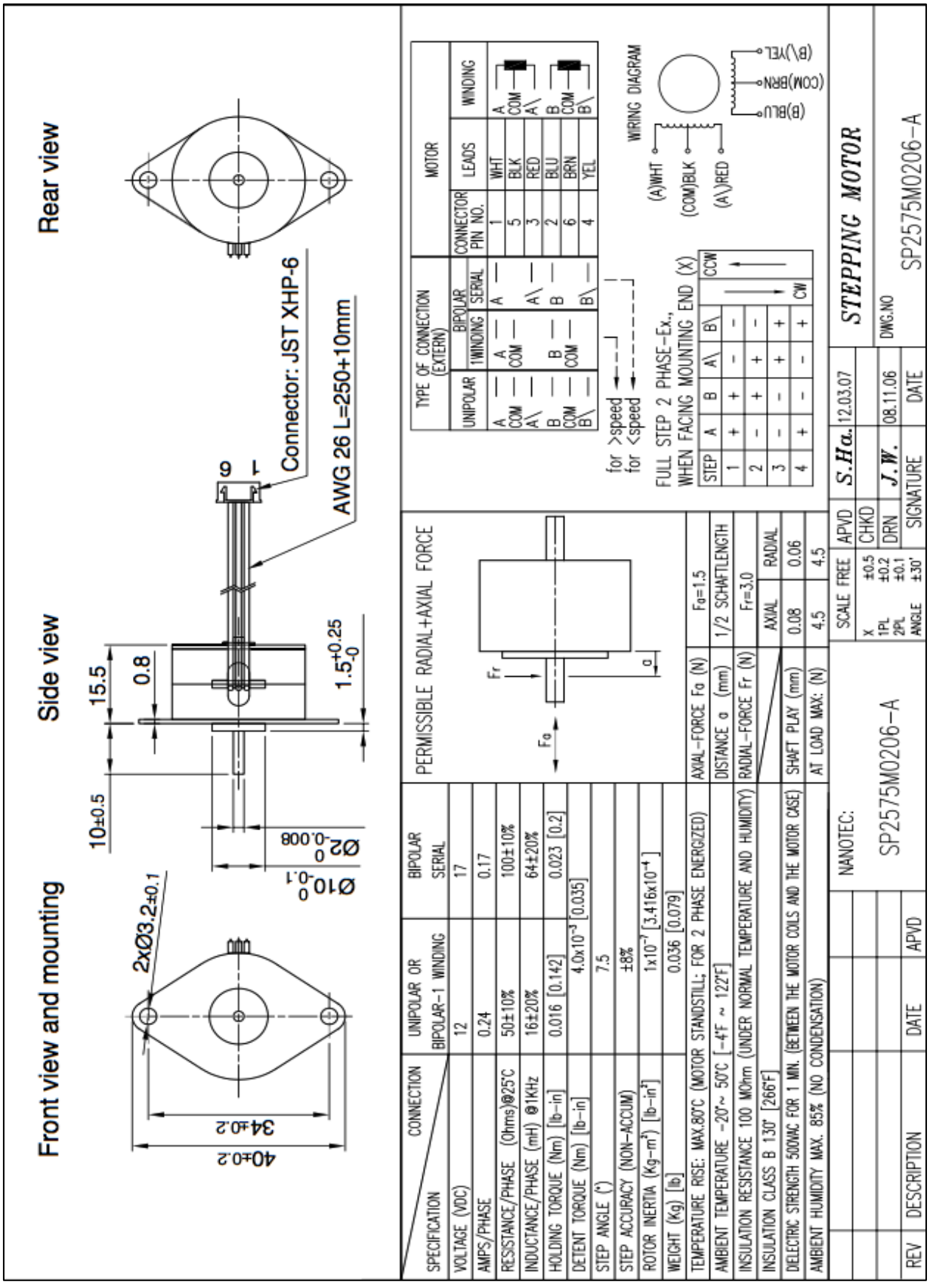
Model	Step Angle	Holding Torque	No. of Leads	Phase Current	Phase Resistance	Phase Inductance	Rotor Inertia	Detent Torque	Mass	Body Length (L)
	Deg	N-cm		A	Ω	mH	g-cm²	g-cm	g	mm
MY5401	1.8	7	4	0.75	4.3	4	12	100	150	26
MY5602	1.8	8	6	0.76	10.5	4.8	12	100	150	26

Dimensions (mm)



Astrosyn International Technology Ltd, The Old Courthouse, New Rd Ave, Chatham, Kent ME4 6BE, England
Tel: +44(0)1634 815175 Fax: +44(0)1634 826552

SP2575M0206-A Stepper Motor



Appendix 5 – Arduino Push button code

```
/*
  Button
 Turns on and off a light emitting diode(LED) connected to digital
 pin 13, when pressing a pushbutton attached to pin 2.
 The circuit:
 * LED attached from pin 13 to ground
 * pushbutton attached to pin 2 from +5V
 * 10K resistor attached to pin 2 from ground
 * Note: on most Arduinos there is already an LED on the board
 attached to pin 13.
 created 2005
 by DojoDave <http://www.0j0.org>
 modified 30 Aug 2011
 by Tom Igoe
 This example code is in the public domain.
 http://www.arduino.cc/en/Tutorial/Button
 */

// constants won't change. They're used here to
// set pin numbers:
const int buttonPin = 2;     // the number of the pushbutton pin
const int ledPin =  13;      // the number of the LED pin

// variables will change:
int buttonState = 0;         // variable for reading the pushbutton status

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
}

void loop() {
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed.
  // if it is, the buttonState is HIGH:
  if (buttonState == HIGH) {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
  } else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }
}
```

Appendix 6 – PIC code for Stepper motor

```
/* Program to rotate clockwise on the pushing of PB1 and to rotate anticlockwise
on the pushing of PB2 */

#include "xc.h"
void turn_clockwise (unsigned char step);
void turn_anticlockwise (unsigned char step);
void turn_off (void);
void main(void)
{
    TRISBbit.RB0 = 0;
    TRISBbit.RB1 = 0;
    TRISBbit.RB2 = 0;
    TRISBbit.RB3 = 0;
    TRISBbit.RB4 = 1;
    TRISBbit.RB5 = 1;
   unsigned char count = 0;
   while (1)
    {
        while (PORTBbits.RB4 == 1) {       //While PB1 is pressed
            turn_clockwise (count);
            delay_ms (10);
            count ++;
if (count == 4)
                count = 0;
        }
        while (PORTBbits.RB5 == 1) {        //While PB2 is pressed
            turn_anticlockwise (count);
             delay_ms (10);
            count ++;
if (count == 4)
                count = 0;
        }
        if ((PORTBbits.RB4 == 0) && (PORTBbits.RB5 == 0))
            turn_off();
   }
        }
void turn_anticlockwise (unsigned char step) {
    switch (step) {
            case 0: LATBbits.LATB0 = 1;
                    LATBbits.LATB1 = 0;
                    LATBbits.LATB2 = 0;
                    LATBbits.LATB3 = 0;
                    break;
            case 1: LATBbits.LATB0 = 0;
                    LATBbits.LATB1 = 1;
                    LATBbits.LATB2 = 0;
                    LATBbits.LATB3 = 0;
```

```
                break;
        case 2: LATBbits.LATB0 = 0;
                LATBbits.LATB1 = 0;
                LATBbits.LATB2 = 1;
                LATBbits.LATB3 = 0;
                break;
        case 3: LATBbits.LATB0 = 0;
                LATBbits.LATB1 = 0;
                LATBbits.LATB2 = 0;
                LATBbits.LATB3 = 1;
                break;
        default: break;
    }
}
void turn_clockwise (unsigned char step) {
   switch (step) {
        case 0: LATBbits.LATB0 = 1;
                LATBbits.LATB1 = 0;
                LATBbits.LATB2 = 0;
                LATBbits.LATB3 = 0;
                break;
        case 1: LATBbits.LATB0 = 0;
                LATBbits.LATB1 = 0;
                LATBbits.LATB2 = 0;
                LATBbits.LATB3 = 1;
                break;
        case 2: LATBbits.LATB0 = 0;
                LATBbits.LATB1 = 0;
                LATBbits.LATB2 = 1;
                LATBbits.LATB3 = 0;
                break;
        case 3: LATBbits.LATB0 = 0;
                LATBbits.LATB1 = 1;
                LATBbits.LATB2 = 0;
                LATBbits.LATB3 = 0;
                break;
        default: break;
    }
}
void turn_off (void) {
   LATBbits.LATB0 = 0;
   LATBbits.LATB1 = 0;
   LATBbits.LATB2 = 0;
   LATBbits.LATB3 = 0;
}
// End of File
```

Appendix 7: Mechanical design for the picking mechanism

The mechanical designs for the motor frame of picking mechanism along with measurements are shown in Figures A7.1, A7.2 and A7.3.

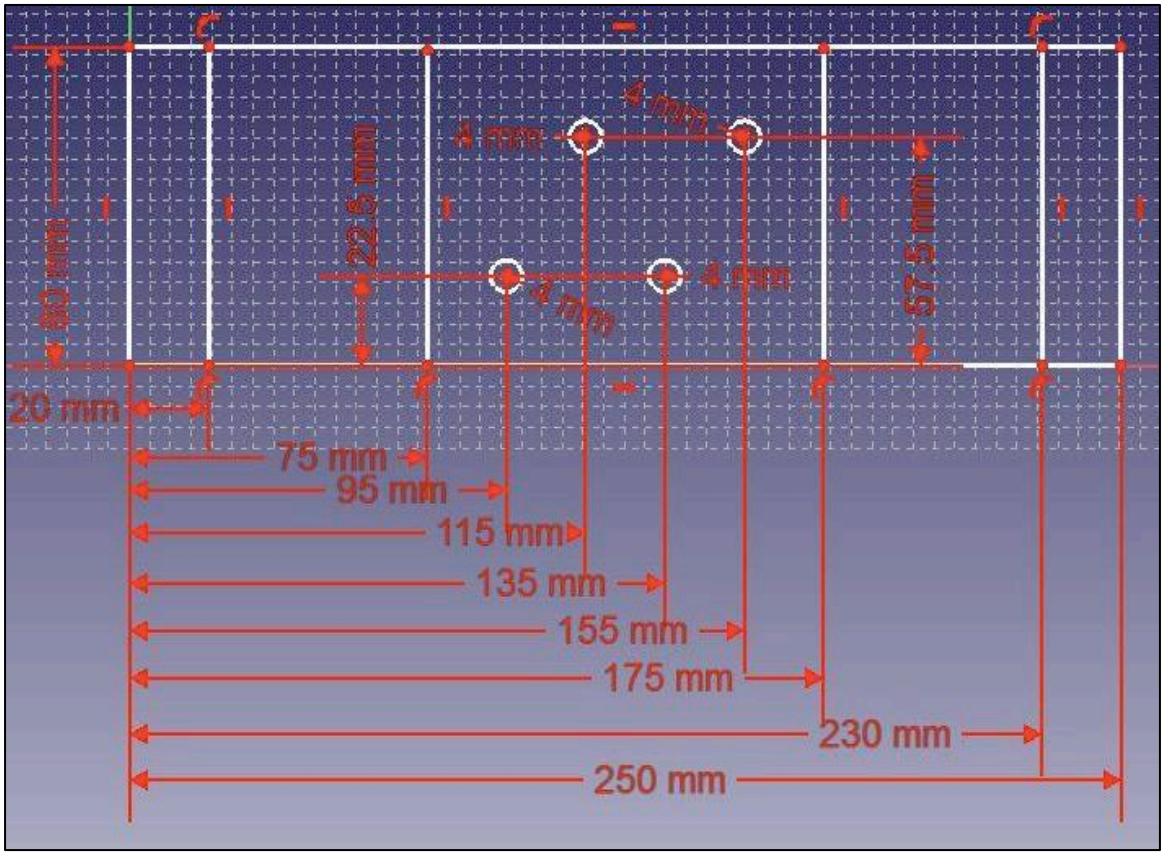


Figure A7.1: Measurement of the motor frame.

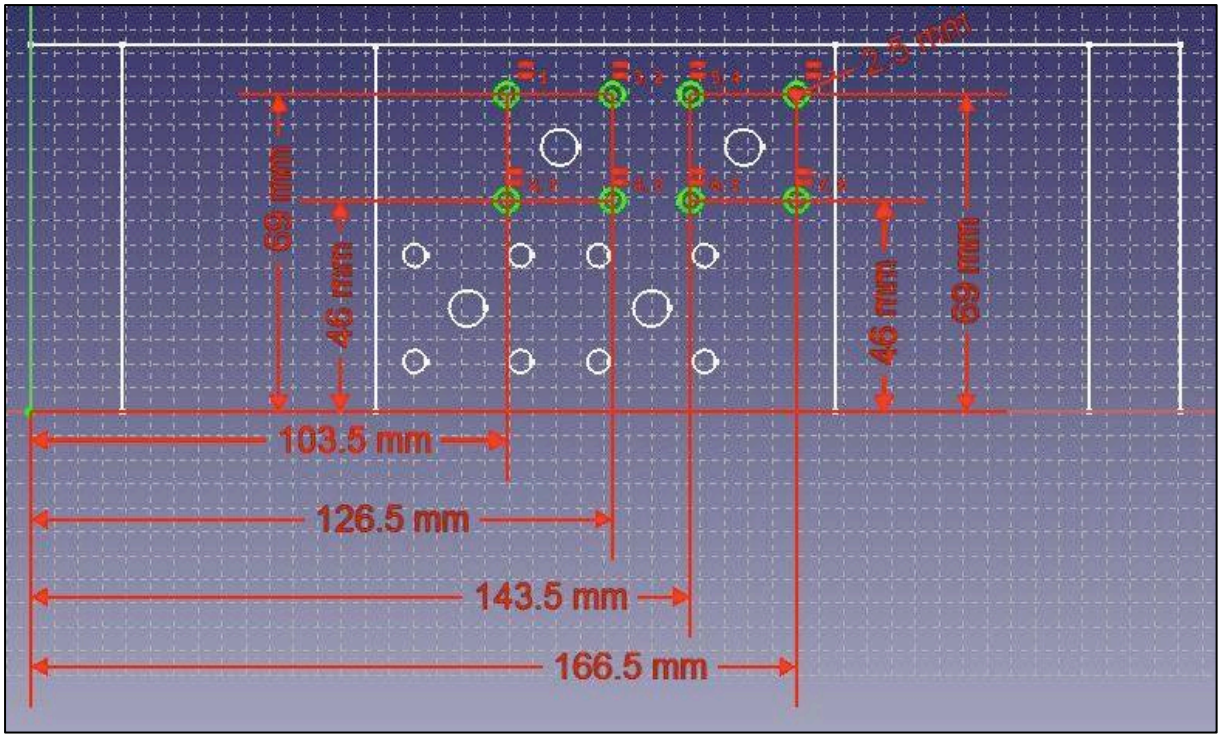


Figure A7.2: Measurement of the drill holes for the stepper motors.

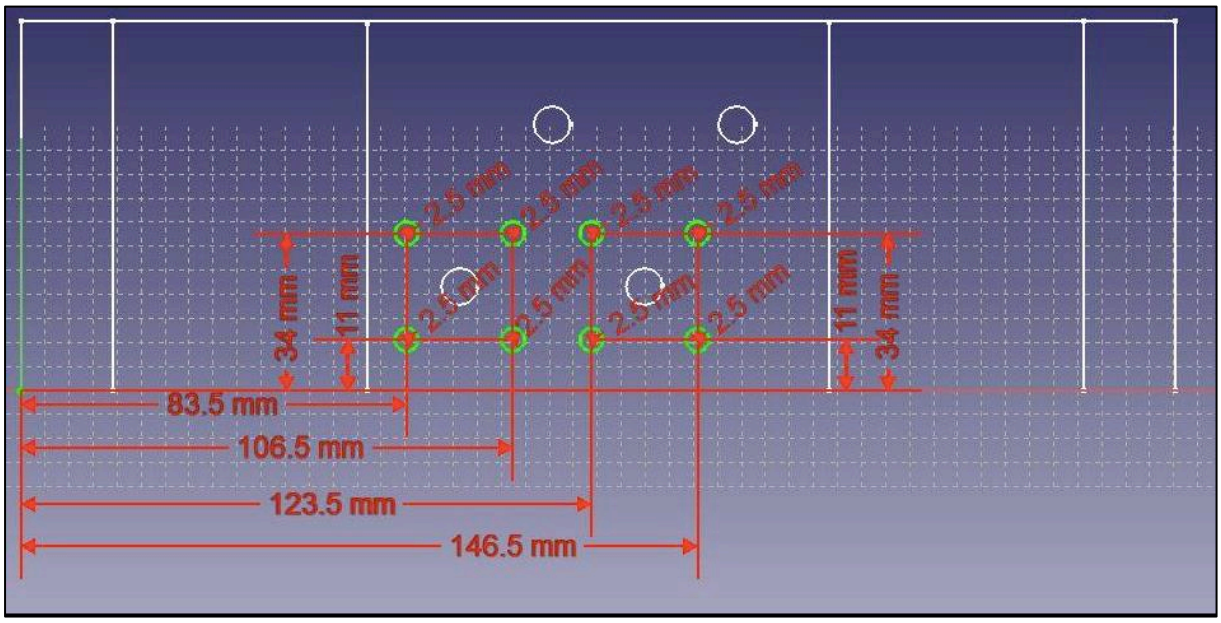


Figure A7.3: Measurement of the drill holes for the stepper motors.

Appendix 8: Final code for the picking mechanism

```
#include <Stepper.h> // Stepper

#define motorSteps 200     // Number of steps per revloution for the motors

//Pins for Stepper 1-4
#define motor1Pin1 13
#define motor1Pin2 12
#define motor2Pin1 11
#define motor2Pin2 10
#define motor3Pin1 9
#define motor3Pin2 8
#define motor4Pin1 7
#define motor4Pin2 6

//Length of MIDI values
#define tunelength 400

//Beats per minute
#define tempo 80

// 1/8th of the beat time in microseconds
const long beattime = 187500;

//PROGMEM stores the MIDI array tune is flash memory
const unsigned char tune[tunelength] PROGMEM = { 35  , 0 , 37  , 0 , 38 , 0 , 40 ,
0 , 42  , 0 , 38 , 0 , 42  , 0 , 0, 0 , 41  , 0 , 37  , 0 , 41  , 0 , 0, 0 , 40 , 0 , 36  , 0 , 40
, 0, 0 , 0 , 35 , 0 , 37 , 0, 38 , 0, 40  , 0 , 42  , 0, 38 , 0 , 42 , 0 , 47  , 0 , 45 , 0 , 42
, 0 , 38  , 0 , 42 , 0 , 45  , 0 , 0 , 0, 0 , 0 , 0, 0 , 35  , 0 , 0 , 0 , 42 , 0 , 0 , 0 , 35  , 0
, 0 , 0 , 42  , 0 , 0 , 0 , 35  , 0 , 0, 0, 42 , 0, 0 , 0 , 35 , 0 , 0, 0 , 42 , 0, 0 , 0 , 35,  0,
0, 0, 42 , 0 , 0, 0 , 35  , 0 , 0 , 0 , 42  , 0 , 0 , 0 , 38 , 0 , 0, 0, 45  , 0 , 0, 0, 38,  0 ,
0 , 0 , 45 , 0 , 0 , 0 , 42 , 0, 44  , 0 , 46,  0 , 47  , 0, 49  , 0 , 46 , 0 , 49  , 0, 0, 0 ,
50  , 0 , 46 , 0 , 50,  0, 0, 0 , 49,  0 , 46,  0 , 49 , 0 , 0 , 0 , 42  , 0 , 44,  0 , 46,  0,
47  , 0 , 49 , 0 , 46,  0, 49 , 0, 0 , 0 , 50 , 0 , 46 , 0, 50 , 0 , 0 , 0 , 49,  0 , 0, 0, 0, 0
, 0, 0 , 42 , 0 , 0 , 0 , 49 , 0 , 0 , 0 , 42  , 0 , 0 , 0 , 46 , 0, 0, 0, 42 , 0 , 0 , 0 , 46 , 0
, 0 , 0 , 42 , 0, 0, 0 , 46  , 0 , 0 , 0 , 42,  0, 0, 0 , 49 , 0, 0 , 0 , 42  , 0 , 0 , 0 , 46  , 0
, 0 , 0 , 38 , 0 , 0, 0 , 46,  0, 0, 0 , 42 , 0 , 0 , 0 , 46,  0, 0, 0 , 35  , 0 , 37 , 0 , 38 , 0
, 40 , 0 , 42  , 0 , 38 , 0 , 42 , 0 , 0 , 0 , 41  , 0 , 37,  0 , 41 , 0, 0 , 0 , 40 , 0 , 36,  0
, 40 , 0 , 0 , 0 , 35 , 0 , 37 , 0, 38 , 0 , 40 , 0, 42,  0 , 38 , 0 , 42 , 0 , 47 , 0, 45,  0,
42  , 0 , 38,  0 , 42 , 0, 45  , 0 , 0, 0, 0 , 0, 0, 0 , 35 , 0 , 0, 0 , 42 , 0, 0, 0 , 35  , 0,
0 , 0 , 42 , 0 , 0 , 0 , 35 , 0 , 0 , 0 , 42 , 0 , 0, 0 , 35  , 0, 0, 0 , 42 , 0 , 0, 0 , 35 , 0 ,
0, 0 , 47 , 0 , 0 , 0 , 45 , 0, 0, 0 , 43,  0, 0 , 0 , 42  , 0 , 0 , 0, 40 , 0, 0, 0 , 38  , 0 ,
0, 0 , 37  , 0 };

int beatcount = 0; //to index the array tune
int string_num; //to indicate which guitar string

int flagE = 0; //flag to determine rotation (E string)
int flagA = 0; //flag to determine rotation (A string)
```

```
int flagD = 0; //flag to determine rotation (D string)
int flagG = 0; //flag to determine rotation (G string)

unsigned char loadnote; //determines which motor

volatile unsigned int timer_beat_count = 0; //sets tempo in Timer1
volatile unsigned char next_beat = 0; //interrupt flag indicating next beat

int buttonPin = 2; //Signal pin for push button or robotic conductor
int buttonState = 0;

// initialize of the Stepper library:
Stepper motor1(motorSteps, motor1Pin1, motor1Pin2);
Stepper motor2(motorSteps, motor2Pin1, motor2Pin2);
Stepper motor3(motorSteps, motor3Pin1, motor3Pin2);
Stepper motor4(motorSteps, motor4Pin1, motor4Pin2);

void setup() {
  // Initialize the Serial port:
  Serial.begin(9600);

  // set the motor speed at 200 RPM:
  motor1.setSpeed(200);
  motor2.setSpeed(200);
  motor3.setSpeed(200);
  motor4.setSpeed(200);

  pinMode(buttonPin, INPUT); //sets the pin as an input

  Start_Beat_Timer(); //Start Timer1 with selected beat

  while (next_beat == 0)  //wait for beat timer
    ;
  next_beat = 0;
}

void loop() {
  // reads buttonState after every tune is played
  while ( buttonState = digitalRead(buttonPin)) {
    if (buttonState == HIGH) {
      // If high, code to control motors with MIDI values
      next_beat = 0;
      Start_Beat_Timer();

      for (int beatcount = 0; beatcount < tunelength; beatcount++) {
        loadnote = pgm_read_byte(&tune[beatcount]);
        Serial.print("Current note is ");
        Serial.println("Load_Note");

        Load_beat(loadnote);
```

```
        Playbeat();
      }

      while (next_beat == 0)
        ;
      next_beat = 0;
    }
    else {
      //If low, turn off motor pins
      pinMode(motor1Pin1, LOW);
      pinMode(motor1Pin2, LOW);
      pinMode(motor2Pin1, LOW);
      pinMode(motor2Pin2, LOW);
      pinMode(motor3Pin1, LOW);
      pinMode(motor3Pin2, LOW);
      pinMode(motor4Pin1, LOW);
      pinMode(motor4Pin2, LOW);
    }
  }
}

// Load beat load the motor pin to be played
void Load_beat (unsigned char Note) {
  switch (Note) {
    case 0: string_num = 5;
      break; //no note played
    case 35: string_num = 3;
      break;
    case 36: string_num = 3;
      break;
    case 37: string_num = 3;
      break;
    case 38: string_num = 3;
      break;
    case 40: string_num = 2;
      break;
    case 41: string_num = 2;
      break;
    case 42: string_num = 2;
      break;
    case 43: string_num = 2;
      break;
    case 44: string_num = 2;
      break;
    case 45: string_num = 2;
      break;
    case 46: string_num = 3;
      break;
    case 47: string_num = 4;
```

```
      break;
    case 49: string_num = 4;
      break;
    case 50: string_num = 1;
       default: break;  //do nothing
  }
}

void Playbeat(void) {

/*The string numbers and flags are used to determine which side the motor has
spun so that it can spin in reverse the next time
 */

  if (string_num == 1 && flagD == 0) {
    motor1.step(100);
    flagD = 1;
  }
  else if (string_num == 2 && flagE == 0) {
    motor2.step(100);
    flagE = 1;
  }
  else if (string_num == 3 && flagA == 0) {
    motor3.step(100);
    flagA = 1;
  }
  else if (string_num == 4 && flagG == 0) {
    motor4.step(100);
    flagG = 1;
  }
  else if (string_num == 1 && flagD == 1) {
    motor1.step(-100);
    flagD = 0;
  }
  else if (string_num == 2 && flagE == 1) {
    motor2.step(-100);
    flagE = 0;
  }
  else if (string_num == 3 && flagA == 1) {
    motor3.step(-100);
    flagA = 0;
  }
  else if (string_num == 4 && flagG == 1) {
    motor4.step(-100);
    flagG = 0;
  }
  else if (string_num == 5) {
    string_num = 0 ;
  }
}
```

```
/*********************************************************************************
*    Title: Glock-o-bot code
*    Author: Mathers.D
*    Date: 2016
*    Code version: 1.0
*
*********************************************************************************/
void Start_Beat_Timer(void) {

  noInterrupts();           // disable all interrupts
  TCCR1A = 0;
  TCCR1B = 0;
  TIMSK1 |= (1 << TOIE1); // enable timer overflow interrupt
  TCNT1 = timer_beat_count;   // preload timer
  TCCR1B |= (1 << CS12);    // Sets 256 prescaler 16 uSec counts
  timer_beat_count = (65536 - (beattime / 16)); // preload timer 65536-
(416666/16) = 39494
  interrupts();             // enable all interrupts

}

ISR(TIMER1_OVF_vect)        // interrupt service routine
{
  TCNT1 = timer_beat_count;   // Reload timer1 with beat count
  next_beat = 1;
}
```