

**Robot Orchestra**

**Interim Report**

**Group 11**

|  |  |  |
| --- | --- | --- |
| Group Members: | Simpson, Joshua | 8929879 |
|  | Chanda, Joyanto | 9015629 |
|  | Buruiana, Andrei | 9478411 |
|  | Fumagalli, Francesco | 9017237 |
|  | Dimou, Theodoros | 9126435 |
|  | Petrovs, Antons | 9474345 |

Tutors: Prof Danielle George and William McGenn

Date: 16 November 2017

1. **Table of Contents**

[1.](#_gjdgxs) Executive Summary 3

[2.](#_30j0zll) Project vision 3

[2.1.](#_1fob9te) Introduction 3

[2.1.1.](#_3znysh7) Required Skills 4

[2.1.2.](#_2et92p0) Project Roles 5

[2.2.](#_tyjcwt) Aims and Objectives 6

[2.2.1.](#_3dy6vkm) Aims 6

[2.2.2.](#_1t3h5sf) Objectives 6

[2.3.](#_4d34og8) Motivation 6

[2.4.](#_2s8eyo1) Literature Review 7

[2.4.1.](#_17dp8vu) Keyboard and Xylophone 7

[2.4.2.](#_35nkun2) String instruments 9

[2.4.3.](#_1ksv4uv) Wind instruments 9

[2.4.4.](#_44sinio) Stepper motors 9

[2.4.5.](#_2jxsxqh) Tesla Coils 10

[3.](#_z337ya) Individual Chapters 10

[3.1.](#_3j2qqm3) Embedded Systems Overview 10

[3.1.1.](#_1y810tw) Conductor 10

[3.1.2.](#_4i7ojhp) MIDI 14

[3.1.3.](#_2xcytpi) 7.2 Timers 17

[3.2.](#_1ci93xb) Xylophone 18

[3.3.](#_3whwml4) Keyboard 18

[3.4.](#_1pxezwc) Panpipes 21

[3.5.](#_49x2ik5) Stepper motors 22

[3.5.1.](#_2p2csry) Hardware 22

[3.5.2.](#_147n2zr) Software 23

[3.6.](#_3o7alnk) Tesla coils 24

[4.](#_23ckvvd) Progress and Planning 24

[4.1.](#_41mghml) Approach towards 2nd Semester 28

[4.2.](#_2grqrue) Risk Management 29

[4.3.](#_vx1227) Procurement 31

[4.4.](#_3fwokq0) Conclusion 33

[5.](#_1v1yuxt) References 33

# Executive Summary

# Project vision

## Introduction

The University of Manchester Robot Orchestra was created in order to celebrate Manchester becoming the European City of Science. Since its inception in 2016, the project has gained backing from Siemens, The Granada Foundation, EPSRC and National Instruments [1]. The Orchestra debuted at the Museum of Science and Industry in Manchester and has since gone on to perform at the opening of EuroScience Open Forum and BBC Radio 4. It has proven to be quite a popular STEM activity, promoting engineering to younger students by showing that an array of technical engineering skills can be used to produce something that is accessible to the masses. Due to its high praise, this project has been dedicated towards expanding it, giving it four new instruments and a new conductor.

### Required Skills

Looking at previous instruments that have been used in the Orchestra it can be seen that an array of skills are needed in order to make each of the instruments. For example, the Glock-O-Bot (a robot playing the Glockenspiel) required technical skills in analogue circuit design, coding in C and a deep understanding of MIDI files and text conversion. Creating the expansion for the robot orchestra will require these same skills in addition to several more, since the design is more complex and four robots are being built instead of just the one. The figure below shows how the different disciplines covered by the team help in meeting the skills requirement for this project.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Skills | Andrei Buruina | Antons Petrovs | Francesco Fumagalli | Joshua Simpson | Joyanto Chanda | Thodoris Dimou |
| **Electronic Engineering** | **Mechatronic Engineering** | | **Electrical and Electronic Engineering** | | |
| Analogue Circuit Design | X | X | X | X | X | X |
| Electronic Circuit Design | X | X | X | X | X | X |
| Coding in C | X | X | X | X | X | X |
| Experience with Arduino |  |  | X |  | X |  |
| Experience with MyRIO |  |  |  |  |  |  |
| Experience with Raspberry Pi |  |  | X |  |  |  |
| 3D Modelling |  | X |  |  | X | X |
| PCB Design | X | X | X |  |  |  |
| Basic Music Knowledge |  |  |  | X |  |  |
| Experience in LabVIEW |  |  |  |  |  |  |

The design being utilised in this project utilises three different embedded systems: the Raspberry Pi, Arduino and National Instruments’ MyRIO so previous experience working with embedded systems is crucial. As can be seen, both the Arduino and Raspberry Pi come within the scope of past experience the team has, however, no one has worked with the National Instruments MyRIO. In order to tackle this issue an application was submitted for the National Instruments Student Project Sponsorship which has several advantages. The main benefit of the scholarship is that National Instruments will provide a technical engineer to assist in the use of any National Instruments software or hardware. This project utilises both the National Instruments myRIO and is likely to utilise LabView. Table 1.1 shows that the team does not have much experience in utilising either of those so having an engineer from National Instruments will help in meeting this skills gap.

The bursary also had the added benefit of automatic entry into the National Instrument Student Design Competition. This is a competition designed for student from different universities to submit their individual project designs and have the chance to compete for a cash prize. This is also a highly technical project, so submitting the Robot Orchestra design will emphasise the technical nature of the project.

### Project Roles

In addition to the technical skills that will be required to complete the project, several soft skills will be required according to the role that each member has taken on. There are six main roles that need to be fulfilled by team members:

|  |  |
| --- | --- |
| **Project Manager**  **(Joyanto Chanda)** | Main purpose of the project manager is to ensure that the project is running on schedule and can be completed on time. This is done by defining the milestones that need to be completed in order to achieve each of the objectives and overall aim of the project. |
| **Secretary**  **(Joshua Simpson)** | Important to make sure that minutes and agendas are prepared for each team meeting. These are crucial documents since they track important decisions made by the team. |
| **Procurement Officer**  **(Andrei Buruina)** | Ensure that the team is does not exceed the allocated budget. They are also responsible for placing the orders on behalf of the team, whether that is through iProc or other means. |
| **Hardware Lead**  **(Thodoris Dimou)** | Similar to the Software Lead. Defining the team’s approach to the upcoming hardware deadlines. Important to have previous experience in hardware design (e.g. hardware lead on Buggy Project). |
| **Software Lead**  **(Francesco Fumagalli)** | Similar to the Hardware Lead. Defining the team’s approach to upcoming software deadlines. Important to have previous experience in software development (e.g. third year individual project had a significant software component). |
| **Document Control (Antons Petrovs)** | Responsible for ensuring that documents across the project are the same format and follow a similar theme. Will also be responsible for assembling the reports. |



These roles will be maintained throughout the course of the project with the exception of the Project Manager role. Multiple members of the team have expressed interest so, in order to provide a rounded learning experience, this person fulfilling this role may change partway through the project.

## Aims and Objectives

### Aims

The aim of this fourth-year project is to create 4 or more new robot music instruments and/or robots playing instruments. These robots should follow a consistent aesthetic and theme that should represent engineering and mechanics so it does not fall out of line with the existing robot orchestra. They must also be controlled by an electronic conductor who will send MIDI files to the robots playing the instruments.

These four robots need to be able to reproduce at least 2 different songs or themes, but should also be designed to be able to be programmed to play a variety of other musical pieces. Autonomy is expected from the robots; ideally the only interaction should be the transmission of notes via MIDI files between the conductor and the robots.

Apart from the flexibility in the ability to play different styles of music, this new core of instruments as well as the conductor should be mindful of future additions and should be expected to accommodate for new instruments. The robot instruments must also allow being taken apart, transported and put back together quickly and easily so that time is saved when preparing for concerts and demonstrations.

### Objectives

* Prepare the team for the project:
* Assign roles to each team member, taking into account strengths and experience.
* Do project planning for the upcoming project blocks.
* Propose designs for the four new instruments:
* Select multiple songs and assign instruments that could perform its parts.
* Select 2 suitable songs that have 4 of the same instruments.
* Divide the team into 4 groups to work on designing each instrument.
* Construct the 4 instruments:
* Design and make prototypes for each instrument.
* Tweak or alter the design as progress is made.
* Test the instruments to play tunes or songs individually
* Assembling the new core orchestra:
* Design and make the conductor.
* Interface the conductor with the instruments.
* Play all of the instruments together.

## Motivation

The work undertaken in this project is motivated by the rising interest in the university’s existing Robot Orchestra, with the orchestra having most recently been part of an engineering tour sponsored by the Royal Academy of Engineering as well as having been part of a BBC programme.

As the existing orchestra has a limited range of songs it can play, adding new instruments would allow for a wider selection of musical styles to be played and appeal to a broader audience. In addition to its outreach aspect, the project also stands to illustrate the operation of multiple embedded systems in performing a common task, making use of platforms such as Arduino and Raspberry Pi to control the performance of the orchestra.

Additionally, due to the combination of different development platforms that the orchestra uses, the project can also be used commercially by being leased to various businesses. As companies in the technology industry can be expected to be present at several trade fairs throughout the year, the Robot Orchestra offers a good opportunity for companies to showcase the applications of their products in an atypical project that has a high chance of drawing attention and attracting new customers.

Lastly, since it can be difficult to explain the benefits of pursuing one’s interest in science to an audience of different ages and backgrounds, the robot orchestra developed during this project can also serve as a good aid for the university to promote its courses.

## Literature Review

### Keyboard and Xylophone

Early on in the background we decided that using a keyboard would be a good choice for one of the instruments. There were several designs that were found online, but the two that stood out the most are shown below.

|  |  |
| --- | --- |
|  |  |

Figure 1 Two Robot Keyboard designs from picture board meetings [1] [2]

These videos showed the problem of a robot piano being approached in two different ways. The design on the right utilised ten robotic fingers split in to two ‘hands’. One of the hands remained stationery (presumably to play chords) and the other would slide up and down a rail, able to play an array of different notes. This design was shortlisted as one to base the design of the robot keyboard off because of how closely it replicated human movement, this would make it look more aesthetically pleasing when it came to the demonstration day. The design shown on the left is a much practical and easy to implement design. It utilises solenoids to push down the keys. From Figure 1 it can be seen that the solenoids seems to be staggered so that the black and the white keys can be hit.

Comparing the two designs, the one on the left is definitely going to be easier to make both in terms of the hardware and the software. Using a moving hand poses significant challenges, specifically in terms of the software, that limit the number of songs that can be played. If the hand had to move then the software would have to predict not only where the next note is based but where the next four notes are also based. If it did not do this then there is a chance that the hand would have to move for each individual note. This form of software is within the skillset of the team but due to the team constraints it was not feasible to build and construct three other instruments. The design on the left is significantly simpler, using only a series of solenoids in a fixed position to push down on the keys. The software for this style of design is a lot simpler to create as well, since for instruments are being created this was an important factor to consider. In order to progress with the design it was important to do some background reading around solenoids.

|  |  |
| --- | --- |
|  |  |

Figure 2 Pull action solenoid from Solentec Ltd. (left) corresponding force/stroke characteristic (right) [3]

Solenoids are common electronic components that come in various forms, electromechanical, pneumatic and hydraulic [4]. They consist of two main components: a coil and plunger or slug that is placed inside the coil (this is usually made out of steel) [4]. According to Ampere’s Law [5] when a current is passed through the solenoid a magnetic field is created in and around the coil which forces the plunger to be drawn in. As can be seen in Figure 2 a spring is used to hold the plunger in place, this is so that when a current is not being passed through the solenoid and no force is acting on the plunger it returns to its normal position outside of the coil, this is typical of a ‘Pull’ type solenoid. ‘Push’ type solenoids are also available where the spring holds the plunger inside the coil when the coil is not energised, when a current is passed through the plunger is forced out of the coil.

A solenoid is dictated by two parameters, the duty cycle of the solenoid and the force produced according to the duty cycle. The duty cycle is calculated by the following formula:

Equation 1 Duty Cycle of a Solenoid [6]

In this case the “On Time” refers to the amount of time that the coil is energised and the “Off Time” refers to the time that it is not. Figure 2 shows how the force that the solenoid exerts changes with the duty cycle and the length of the stroke. The stroke is the amount of the plunger that is outside of the coil when it is energised. It can be seen that as the duty cycle decreases the force that can be exerted by the solenoid increases, this means that if the solenoid is not switched on and off very quickly then more force can be expected to be produced.

### String instruments

### Wind instruments

It was decided it would be interesting to demonstrate a wind instrument so current robot wind instruments were researched using YouTube. One issue with creating a robot to play a wind instrument is that they require the air flow to be controlled in a specific way with the shape of the lips and position of the tongue for example, when playing a trumpet the lips have to vibrate [1] [2]. There is a video of a robot playing the trumpet to achieve this; the robot had artificial lungs and lips [3]. Another example of a robot with artificial lungs and lips (that can be seen) was used to play the flute [4], it was realised that designing and manufacturing a robot like this would be unrealistic due to time limit and budget.





Due to this it was decided that a wind instrument would have to be chosen in which the lip position required doesn’t change and the airflow could be constant removing the need for artificial lungs. The instruments found to meet these requirements were the recorder and panpipes. The panpipes are played by forming a small gap between the lips and blowing across the panpipe opening [5] and for the recorder there is just one tongue position which is maintained while blowing to produce the note [6]. Examples of robots playing the recorder (lost the reference) and the panpipes [7]were found on YouTube. The robot playing the recorder just uses a pump to generate air flow and doesn’t even have lips that go around the mouthpiece. One interesting thing about this instrument is that the air is produced continuously and it has a plate that moves in front of the air flow when no note is being played to deflect it away from the recorder. This removes the need for valves and the possibility of pressure building up behind the closed valve that is released when the next note is being played, which could lead to the note not being produced correctly.



The panpipe playing robot has one pipe that produces the air and that can be moved to change the angle the air is directed into the pipe this is to create half notes which correspond to sharps and flat notes. The panpipes itself is rotated under the nozzle to line up different tubes with the nozzle to produce different notes. It also has a metal plat between the nozzle and the panpipes this is to direct the air flow and simulate the lip position required. In this example the panpipes ae used to play the vocals which could be an effective way of incorporating them into the orchestra. 



Figure 3: Robot panpipe example [7]

### Stepper motors

A stepper motor is a DC motor that is able to convert digital pulses into equal steps of a rotation [1]. Generally, stepper motors are used in high-precision applications that use digital pulses as control signals rather than analogue voltage levels and by varying the frequency of the digital pulses, the mechanical shaft rotation can be modified. A high frequency of the pulses results in a continuous movement of the motor shaft. The advantage of stepper motors when compared to other type of high-precision motors such as servo motors is that it does not require a feedback mechanism as each individual pulse translates to the shaft rotating a precise angle [2].

Figure 1 presents the cross section of a permanent magnet stepper motor. As it can be observed, the permanent magnet acts as the rotor of the motor, being surrounded by the windings of the stator.

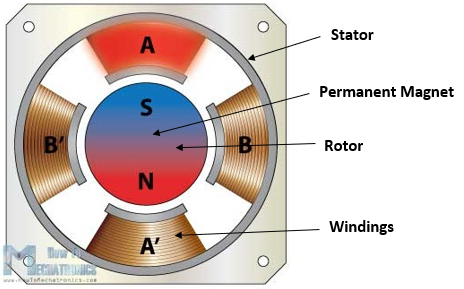


Figure 1 - Cross Section of Stepper Motor [3]

A good way of understanding how the rotational movement is achieved is to analyse the rotor and the stator of the motor individually.

The rotor comprises of 2 discs placed back to back, one acting as a magnetic north pole and the other as a magnetic south pole. As illustrated in Figure 1, the stator is placed around the rotor and is fixed. Each winding acts as an electromagnet that can be controlled independently. When activated, the stator will attract the disc of opposite polarity in the rotor, leading to the rotor’s rotational movement [4].

Depending on the level of precision required, a stepper motor can be driven in 3 different modes: full step, half step and microstep. The difference between these modes is represented by the order in which the windings in the stator are energised and the effects it has on the rotational movement. For example, in half step mode, a digital pulse would produce an angular movement that is half of the one produced by the same digital pulse when the motor is operated in full step mode.

The rotation of the motor can be controlled with a PWM signal (pulse-width modulated signal) of different frequencies and if this frequency is in the audible range (20Hz-20KHz [5]), the motor is able to produce different musical notes.

### Tesla Coils

# Individual Chapters

## Embedded Systems Overview

Embedded systems are one of the key aspects of the Robot Orchestra project. Every instrument requires a controller, and they all need to interface with the conductor. At the heart of all embedded systems lies a microcontroller so additional care needs to be taken when selecting the appropriate one. Various parameters affect the choice of microcontroller, most notably, the clock speed, number of IO pins as well as the different interfaces used to connect peripherals. Each part of the orchestra also requires certain things, for example the conductor should be able to have a graphical user interface to allow friendly user interaction. However, there is a lot of flexibility when choosing microcontrollers for the instruments and therefore choice of microcontroller for the instruments will be based on other factors discussed next.

As the project is also used to showcase our skills as well as a great way to apply them whilst learning new things, the majority of the microcontrollers and peripherals will be chosen with that in mind. Moreover, using diverse programming languages, such as C,Python and Labview will also allow us to improve on languages we have previously used and learn new ones. To conclude, apart from the conductor which has specific needs to be met, the rest of the embedded systems will be selected with the academic aspect in mind.

### Conductor

The conductor has three tasks to perform. The first is to convert MIDI files into a format which can be interpreted by the various microcontrollers used (more in section X). The second is to transmit the converted MIDI files to the corresponding instruments, and lastly to synchronise the instruments. Furthermore, to make the conductor more elegant the team has decided to user a GUI to allow the user to interface with the conductor and must be relatively easy to transport as per the project specifications. Due to these restrictions, a computer was chosen for the task. More specifically a Raspberry Pi 3 (RPi) model B single board computer was chosen. Due to the familiarity of the team with the RPi and the complexity of the conductor the RPi was considered the best choice. The RPi is a System-On-Chip which runs a Linux distribution known as Raspbian [1]. The RPi is compatible with WiFi and Bluetooth as well as 40 GPIO pins [2]. It also has various other interfaces such as SPI and I2C which are commonly used to connect to peripheral devices. Python will be exclusively used to run scripts on the Pi and PyQt, a python based GUI builder to design the user interface. Since the RPi is essentially a computer, a screen/keyboard and mouse will all be connected.

Instrument Microcontrollers

Since the team will build four different instruments, at least 2 different microcontrollers will be used. As mentioned previously, since the load on the microcontrollers will be relatively low the choice will only be based on educational factors. The team has considerd the following microcontrollers for the instruments:

ATmega328P (Arduino Uno): The ATmega328P is the microcontroller used in the Arduino Uno platform. Arduino is an open-source platform consisting of both hardware and software. The hardware is based around the ATmega 8-bit microcontroller which is flashed with the Arduino bootloader, allowing Arduino programs, known as sketches to run on the microcontroller [3]. Arduino’s can be used to program simple tasks as well as more complex ones with relative ease, thanks to the Arduino language. The Arduino language is a simplified version of C++. Arduino also has vast online support and many libraries for various peripheral devices have been written for it, allowing fast development. The ATmega328P consists of 14 digital IO pins, 6 analogue pins, SPI/I2C/USART as well as other standard peripherals such as timers and interrupts [4]. It is also interfaceble with USB/WiFi/Bluetooth and XBee to name a few. This gives the team flexibility when developing the instruments. In order to customise the Arduino board to include various other integrated circuits needed for the instruments as well as to make it as professional as possible, a breakout board for the ATmega328P was made. The Arduino bootloader was then flashed onto it to allow the team to upload Arduino sketches to it. A prototype board was first developed. The prototype includes USB to UART FTDI chip which allows the user to upload sketches directly from a PC through USB without any additional external components as well as allowing the Arduino to transmit back to the serial monitor on the PC. Moreover SPI, I2C as well as all the digital and analogue pins have been broken out. Schematic and layout diagrams can be found in APPENDIX X, pictures of the final design can be found below in figure 1.1. The prototype was designed by looking at existing ATmega328 circuits as well as reading the datasheets and application notes of the various components used [5][6][7][8]. The prototype was tested and found to be working as expected. The next step will be to expand the board by adding other integrated circuits and components needed for each instrument.

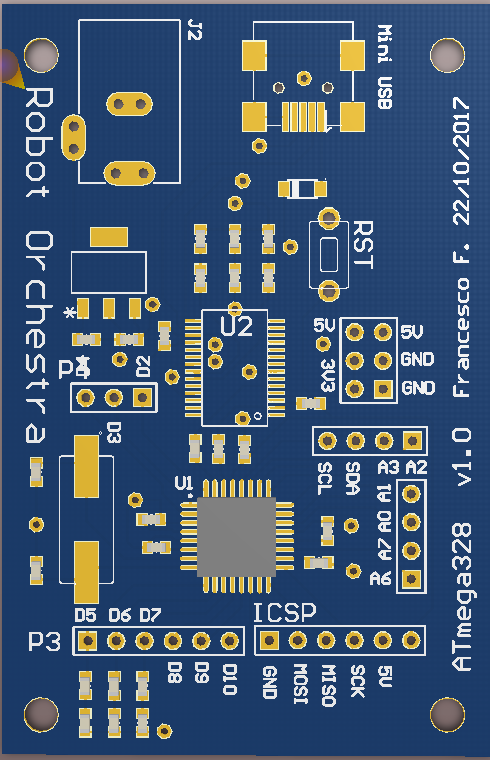
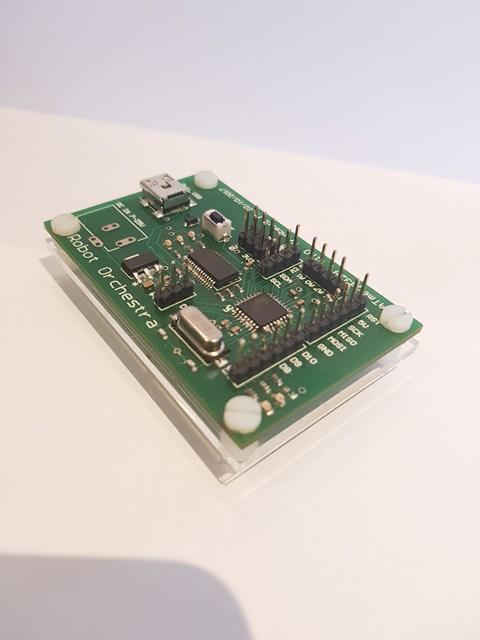


Figure 1.1: ATmega328P breakout

MyRio – The MyRio is an ‘Embedded Device’ [9] developed by National Instruments. It consists of digital IO pins as well as analogue pins. It includes a Xilinx FPGA as well as a dual core ARM A9 processor. It can be programmed using both Labview as well as C [9]. The MyRio is also compatible with WiFi and Bluetooth standards and has all the peripherals which can be found on the ATmega328. The team has decided to use the MyRio as the main driver for the xylophone. The reason for using MyRio is in our project is purely educational. As the team doesn’t have any previous experience with MyRio and only limited experience using Labview it was seen as an opportunity to familiarise ourselves with it through the project. Labview has become an industry standard used by many engineering firms and learning the language is an asset for the team. Moreover, the MyRio is extremely powerful and can easily drive any of the instruments.

NUCLEO-F303K8 – The Nucleo board developed by ST is a breakout board for the ARM Cortex-M4 microcontroller. It can be programmed in both C and C++ (mbed) [10] although the team will use C in conjunction with ST’s hardware abstraction layer (HAL) to program it. This 32-bit microcontroller has replaced the PIC18 used by the University to teach students about microcontrollers. It is an extremely powerful 32-bit microcontroller with a vast range of peripherals. ARM has become an industry standard for microcontroller and microprocessors and the main reason the team chose this microcontroller once again purely academic. In the past team members have used the Teensy microcontroller, which is an ARM Cortex-M4 based platform compatible with Arduino. However, the team has limited, to almost no experience using Nucleo boards as well as HAL. This specific Nucleo board is the smallest one made by ST and is therefore perfectly suited for our project. Moreover, the design of the board means that a PCB can be designed to fit the Nucleo board as well as any other IC or peripheral devices needed to drive the instrument.

High Level Overview



Figure 1.2 High Level System Overview

GUI → RPI → MIDI2Text→instruments:

- Channel 1: Xylophone

- Channel 2: Steppers

- Channel 3: Piano

- Channel 4: Pan Pipes

- Channel 5: Tesla

Conductor

The structure of the embedded systems has been mentioned briefly in the previous section, however this serves as a more comprehensive description. Figure X depicts a flowchart showing the hierarchy of the system. First and foremost, the Raspberry Pi conductor takes input from the user through it’s GUI. The GUI takes in a few parameters, the file location of the MIDI file that the orchestra will play, the file location which the MIDI to text output will be saved. The GUI is kept relatively simple, however if more features need to be implemented they can added in the future. Once the MIDI file is selected the song is then converted after the user clicks on the ‘Convert’ button. More on the conversion can be found in Section X. The converted file can be found in the specified location. This is useful for the user to check that the output matches with what is expected. Finally, the user clicks the ‘Play’ button causing the conductor to send the corresponding text file to each instrument. For simplicity, the instrument channels are static and have been predefined. The user is expected to edit the MIDI file prior to uploading it to the Pi to make sure that each channel corresponds to the correct instrument.

Wireless Link

The link between the conductor and the individual instruments will be wireless. The team has considered a few different options. It should be pointed out that the team is currently working on developing the instruments and therefore is not actively developing the communication between the Pi and the instruments. A preliminary research of the available options was conducted. However, an in depth look at the various options as well as testing each one will be done in semester two.

Bluetooth:

The Raspberry Pi has an inbuilt Bluetooth module capable of transmitting to other Blouetooth enabled devices [2]. All apart from the MyRio would require Bluetooth modules as they don’t have any built in. To have the Raspberry Pi transmit to multiple devices at once a piconet is set up. Up to seven devices can be connected to the piconet as slaves, with the RPi acting as the master. The group doesn’t have any experience setting up piconets, making Bluetooth rather challenging. However, prior to conducting further research into the feasibility of Bluetooth, it won’t be ruled out.

WiFi:

After researching WiFi as a transmission method the team found that it is much easier to implement than Bluetooth. The Arduino platform as well as the Nucleo board are both capable of transmitting and receiving WiFi packets using an inexpensive additional module. Since the data transmitted will be relatively simple, a User Datagram Protocol (UDP) can be used with the Pi acting as a server and the instruments as clients. The Arduino platform provides a library which converts the Arduino into a client capable of receiving UDP packets over WiFi. The Nucleo board on the other hand uses the X-CUBE-WIFI1 add-on to the STM32Cube to accelerate development of WiFi to Serial programs. The MyRio has an inbuilt WiFi card and a the MyRio toolkit makes setting up the connection straightforward. Although this hasn’t been tested yet.

Radio:

Radio is the only method which the group has experience connecting multiple devices to. The NRF24L01+ is a 2.4GHz radio module interfaceable with both Arduino and the Raspberry Pi. This specific module allows for up to 127 different modules to be connected simultaneously over different channels. Arduino and Python libraries allow easy development on either platform. The NRF2L01+ uses SPI to communicate with the microcontroller making it compatible with the MyRio, however due to the high complexity of the device a library is needed for fast development and stable operation. Another benefit of using the NRF24L01+ module is that it is much cheaper than the WiFi/Bluetooth modules.

Conclusion

The above sections serve as an overview of the main embedded systems used and how they interact with each other. Detailed information for the embedded systems used by each component of the orchestra can be found in section X.

### MIDI

Overview of MIDI Files

MIDI is an acronym which stands for Musical Instrument Digital Interface. MIDI is used to compile all the information needed to reproduce a song under a common format. MIDI files allow specification of which note to play, for how long and with what velocity the instrument key is to be pressed [1][4]. MIDI has become a standard interface used in the music industry and will be the only file format used in the project. The reason for this is simplicity. MIDI files for a vast number of songs can be easily found online allowing the team to easily choose any song available. The MIDI format is quite complex, however this brief overview will describe features.

Each MIDI file starts with a *Header Chunk*. The header chunk specifies various parameters needed to decode the rest of the file. It is constructed as follow [2][3][5]:

* The first 4 bytes of the header chunk are **‘MThd’** denoted in ASCII (4D 54 68 64) this specified that it is a header chunk.
* The next 4 bytes specify the length of the data which is to come. For header chunks that is always 6, as 6 bytes are always required
* The next 6 bytes of data specify three things:
  + The file format (ff ff).
  + The number of tracks in the MIDI file (nn nn).
  + The timing parameter (number of delta ticks per quarter note).

File Format: The file format can take one of three values and specifies the tracks in the MIDI file: [2][3][5]

* 1: One track in the MIDI file.
* 2: Multiple synchronous tracks in the MIDI file. Which means that the file contains multiple tracks which all start at the same time.
* 3: Multiple asynchronous track in the MIDI file. Which means that the file contains multiple tracks all with different starting points.

Number of tracks in the MIDI file: Is self-explanatory. It simply specifies how many tracks are in the MIDI file.

Timing parameter: The MIDI standard defines timing in its own way, which is *delta ticks per quarter note* and essentially defines the timing for the rest of the MIDI file.

The remaining MIDI file consists of the *Track Chunk* which specifies the notes, velocity and duration of each note. The format of track chunks is similar to that of the header chunk and is as follow:

* First 4 bytes specify that this is the track chunk. They are always ‘MTrk’ which in ASCII corresponds to 4D 54 72 6B in Hex.
* The next 4 bytes specify the length of the data which is to follow.
* The rest of the track chunk consist of the track event, which will be explained in detail below.

The track event can be one of three things. Midi, meta and sysex events, all of which are always preceded by the number of ticks prior to the execution of the instruction of the track event. Meta events can contain additional information such as lyrics and copyright information. For our purpose this will not be investigated further, nor will it be used. All songs are chosen to be under appropriate licenses for them to be reproduced, altered and distributed. Sysex events include information unrelated to the song or the Midi file and are therefore also omitted from the project. Midi events contain messages sent to each individual channel (instrument). They consist of three bytes. The first one byte is know as the status byte and it’s function is to specify the type of command. For example *note on*(start playing the note) and *note off* to stop playing the note. A table with all the control bits available and their functions can be found on **Appendix X**. The second byte specifies the note to which the status byte is applied to and the third byte specifies the note velocity. A table showing the musical note to which each binary value corresponds to can be found on **Appendix X**. [2][3][5]

Midi2Text

The process of converting midi files into a format which can be interpreted by the various microcontrollers is a laborious one, especially when done manually. For that reason, after having a good understanding of the midi format a Python script was written to automate this task. The Python script will run on the Raspberry Pi and will be controlled from the user through a GUI. The MIDO library contains some useful methods used to convert midi files into readable text. Figure **1.1** shows the output the Mido library has after it is given a midi file. Although this is a much clearer format it is still not in a format which microcontroller can interpret. The next step was to write some regular expressions which are capable of extracting the various information needed. Two regular expressions are created, one to match the tempo and one to match the control bits, note, velocity and timing in ticks. Once this information is available the script then converts each match to an easily interpreted format. For example, the control bits specify when note is played or released, all other control bits are ignored. The note is converted from midi format (0→127) to sheet music (A,B,C…) and the time is converted from ticks into seconds. Once the midi file is converted into this format the data is either: printed onto the serial monitor (see Fig.**1.2)** , written to a file or both. When written to a file it is in the form of three arrays. One for note status (on/off) one for the note (in sheet music notation) and one for the time in seconds prior to executing the command. These will then be sent to individual instruments. The Midi2Text script has been tested extensively and after a few revisions is currently working properly. The full code can be found in **Appendix x** [6][7].

.

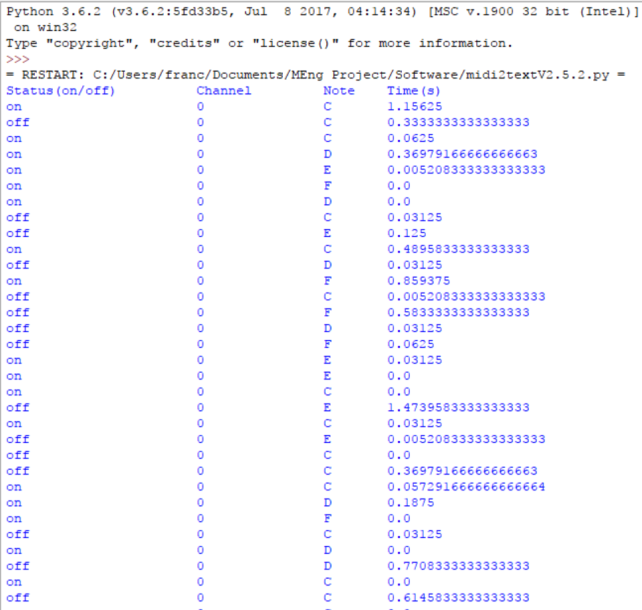


Figure 1.1 Raw Output Figure 1.2 Processed data

### 7.2 Timers

As an Arduino Uno microcontroller has been used for controlling the stepper motors, the timing functions available in the dedicated libraries were used for creating a PWM signal. By generating a square wave of different frequencies, multiple musical notes can be produced. An example for generating the note A4 is presented below.

The note A4 has a frequency of 440 Hz [1]. The period of a musical note can be calculated by using the following equation:

…………………..(1)

where T = period (in seconds)

f = frequency (in Hz)

In the case of A4, the period is approximately 2272 us. In order to generate a square wave of this frequency, the signal can be driven to a logical high for half the period, followed by a logical low for the other half of the period, obtaining thus a PWM signal with a 50% duty cycle. The pseudocode for producing this square wave is as follows:

While (True){

Drive Signal High

Wait for 1136 us

Drive Signal Low

Wait for 1136 us

}

For obtaining this result, the predefined Arduino function dealyMicroseconds() was used. This function takes one parameter representing the number of microseconds required for the delay and pauses the program for that amount of time [2].

Furthermore, playing a song requires multiple musical notes to be played for specific amounts of time. In order to control the time for which each note is playing, the predefined Arduino function millis() was used. This function returns the number of milliseconds elapsed since the running program started [3]. The Arduino Uno board has 3 timers (Timer0, Timer1 and Timer2) and millis() makes use of Timer0 to count the elapsed milliseconds.

Another possible option of generating the required PWM signal is to use the dedicated PWM pins on the Arduino board. These pins are used in conjunction with the analogWrite() function that takes the pin to write to and the duty cycle value as parameters and writes a PWM signal to the designated pin [4]. Additionally, the library in [5] allows for an easy implementation of the needed PWM period.

However, the methods presented above do have some limitations. In the case of using the analogWrite() function, this can influence the millis() function [4], posing a risk to the timing of each note. Additionally, if multiple stepper motors are to play a song simultaneously, making use of the delayMicroseconds() function might result in noticeable delay in the song as the execution of the entire code is paused when this function runs. Possible solutions to this problem are represented either by having the motors controlled by independent microcontrollers or by ensuring that at every moment, a single motor is producing a musical note.

## Xylophone

## Keyboard

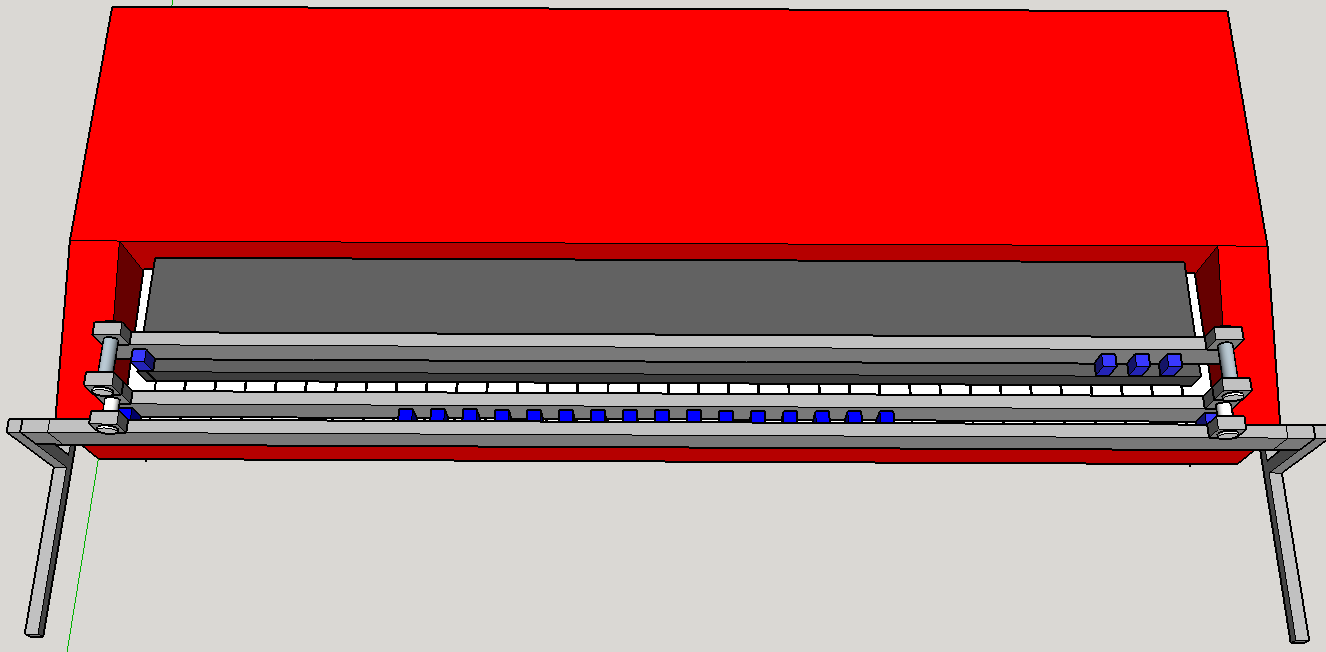
The main aim of the keyboard design was to be as simple as possible since the design for both the xylophone and panpipes were quite complicated. With this in mind the first design was created. 

Figure 3 Initial Keyboard Design shown using Google SketchUp

The first design utilised three metal panels and can be seen in Figure 3. Panels 2 and 3 were used to screw the solenoids in to. The solenoids are represented by the blue boxes in Figure 3, they would be screwed in to the panel and spaced out so the plunger of each solenoid fell in line with a key. It was important to have two separate rails of solenoids since the robot needed to be able to play both the black keys and white keys and they were two different lengths. Panel 1 acted as a support for the other two panels and was attached to the legs of the frame. This design had the advantage of being very simple in construction and not needing many steps in order to be complete.

This model was completed at the end of the first project block but at this stage the keyboard that was going to be used in the demonstration had not been ordered, this meant that the design could not be properly dimensioned. It also meant that the exact solenoid needed for the design could not be chosen since the force needed to push down the key was unknown. Since it was impossible to determine the exact solenoid that was needed some common 5V solenoids were ordered from SparkFun Electronics. These were ordered in low numbers since it was expected they would have to be changed in the second project block.

When the testing phase of the second project block began a circuit to connect the solenoids to a digital I/O pin on the Arduino was created which energise the coil and pull the plunger. This circuit was used to test whether the preliminary solenoids ordered in the first project block were strong enough to push the key down on the keyboard. It was quickly found that a 5V solenoid was simply not strong enough to push the key down, this was to be expected since they had been ordered without any measurement of the forces needed.

To measure this force some M10 nuts were weighed and then placed, one by one, on a single key until a sound was triggered. It was found that when the weight of the nuts equated too ≈70gf this was strong enough to trigger a sound from the keyboard. So a solenoid that is able to produce ≈105gf (1.5\*70gf [6]) should be used for the keyboard design. In addition to this, the maximum duty cycle had to be determined for the solenoids, this would be dependent on the songs that were being played. Using Anvil Studio the piano track for each of the songs was looked at, from this it could be determined that the song that would produce the greatest duty cycle would be *Eye of the Tiger* by *Survivor*. Since one solenoid controls one note with this design it was important to identify the note that would be need to be ‘on’ the most during the song. Through using Anvil Studio again to study the *Eye of the Tiger* MIDI file the note C# was determined to be ‘on’ the most during the song. By using Equation 1 and measuring the seconds that the C# was being played throughout the song the maximum duty cycle was determined (Equation 2) to be 30%.

Equation 2 Calculation for the maximum duty cycle

Using this and the required force a new more appropriate solenoid was selected. Initially the *MCSMO-0630S12STD* from *Multicomp* was selected since the Stroke vs Force characteristic met the requirements needed for this project (Figure 4) [7]. In order to test this solenoid a new circuit had to be created to interface the solenoid with the Arduino board since it cannot deliver enough current to meet the needs of the solenoid. At approximately 30% duty cycle the solenoid consumes 9.6W of power at 12V DC. This means that it needs 800mA to operate at this power, the Arduino is only capable of supplying 40mA from one of its digital IO pins. In order to power the solenoid a circuit was created utilising a transistor that allows the solenoid to be powered from an external power supply whilst being connected to the Arduino **(insert circuit diagram figure here).**

Using this circuit it was determined that this solenoid was strong enough to push one of the keys down on the keyboard. However, this solenoid has since been replaced with the *905-9931* from *Solantec Limited*. This is because the solenoid from multicomp did not come with a spring attached to hold the plunger in place when it was not energised. The solenoid from Solantec comes fixed with a spring and has very similar stroke vs force characteristics, using this solenoid will save significant time as individual springs will not have to be fixed on to the solenoids manually.

It was also decided that the design shown in Figure 3 would have to be modified in order to make it more adjustable to the height of the keyboard. The previous design had a fixed height for the panels, so, in order to overcome this threaded rod was used in replacement of the metal legs. These rods would be fed through a Bosch Bar (Figure 4).

|  |  |
| --- | --- |
|  | |
|  |  |

Figure 4 New robot keyboard design (top), cross-section of Bosch bar (bottom left) [8], T-Slot nut (bottom right) [9]

The reason for going with a Bosch bar is largely to do with simplicity of design. From looking at Figure 5 it can be seen that the Bosch bar being used has grooves of a specific dimension in. T-Slot Nuts can be purchased which fit in to Bosch bar and can be slid along the bar. Clamps will be designed for the solenoids which can be screwed in to individual T-Slot nuts which, depending on whether that particular solenoid will be playing a black or white key, will be a specific distance away from the Bosch bar.

## Panpipes

Out of the possible wind instruments the panpipes were chosen. The video such as the one by TeamDare 3.4[1] shows one way of making a robot playing panpipes as well as showing that automating panpipes is feasible.

The mechanism in the video shows the panpipes moving sideways where-as the nozzle that provides the airflow does not move laterally. The nozzle only moves up and down at an angle to provide a variation in the notes.

The decision for our panpipe design was to reverse the moving components; instead make the nozzle move sideways and the panpipes completely stationary. This simplified the design because the panpipes would require a larger and more sophisticated to be moved where-as the nozzle is much smaller and lighter.

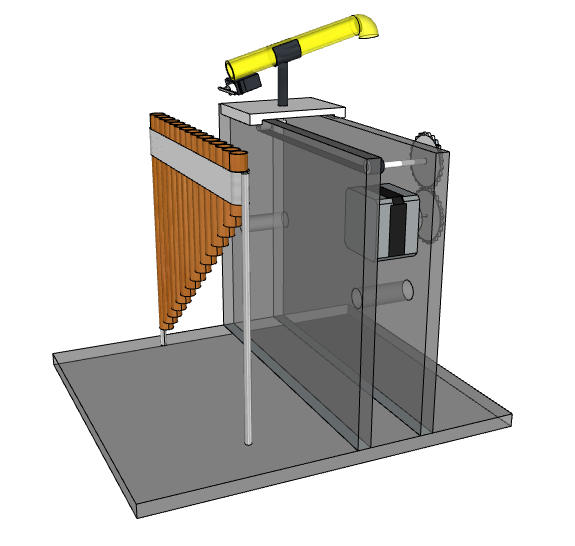


Figure 3.1 shows the initial design for the panpipe setup. The panpipes are held up and fixed and the yellow nozzle constantly blows the air at the pipes with a small servo motor at the end which blocks the airflow to the pipes when needed. The nozzle is on a platform that moves sideways using stepper motor operated track which will be controlled by a microcontroller such as an Arduino.

This setup can be broken down into a list of the main components:

* **The panpipes** are 18cm across with 15 pipes. They came tied together with string and together make a slight curve. It is best if the curve is eliminated so that there is no variation in distance between the nozzle and the individual pipes which could affect the sound. The plan to eliminate the curve is by untying the pipes and making a custom holder for the pipes.
* **The track** will resemble a treadmill and will have the platform with the nozzle attached on top of it. It must be long enough to allow the nozzle platform to go from the first to the last pipe. It was proposed that the track would be driven by a stepper motor which is controlled by an Arduino.
* **The nozzle** is going to be mounted on top of the platform which is attached to the track. The air is provided by a mattress air pump which will be constantly on. To block the air in order to hit timed notes, a servo will be attached to the end of the nozzle with a small piece of plastic on the tip, rotating 90 degrees to stop the air and -90 degrees to allow the air to pass again.
* **The frame** will be made out of 5mm thick Perspex. It will house the track and the lift up the platform with the nozzle to the required height. The stepper motor will be attached from the side.

****

From testing and video it was found that the air needs to hit the tops of the panpipes at a certain angle to provide the best sound, so the mechanical workshop made a support for the nozzle that allows the angle to be adjusted.

As this design has not been assembled yet, there will be variations in some components such as the spacing between each pipe of the panpipes, the height of the frame, the method of blocking the air could be changed to a solenoid valve instead of the servo, the distance between the panpipes and the nozzle, and the type of stepper motor that controls the track.

A change to the design was proposed where the 15 pipes would be mounted in a circle around the nozzle and the nozzle would rotate in the middle using a servo motor. This would facilitate the design by removing the moving platform setup which has a complicated design as well as the stepper motor, making the new setup smaller and requiring fewer components.

[Air pump testing]

## Stepper motors

### Hardware

For producing the required musical notes, a decision has been made to use the NEMA 17 stepper motor. This is because of its specifications which allow for the straightforward interfacing with a microcontroller and also due to the extensive online documentation regarding its interfacing and control. The motor has a rated current of 1.7A and a rated voltage of 12V. Additionally, it has a step angle of 1.8°, meaning that it requires 200 steps for a revolution.

In order to control the motor, the Allegro A4988 driver board was chosen as it was recommended by multiple sources and its datasheet confirmed it could be interfaced with the NEMA 17 motor. According to the A4988 datasheet, the driver board can supply a maximum current of 2A and has an operating voltage between 8V and 35V, meaning it could drive the motor safely [1]. For controlling the motor, an Arduino Uno microcontroller board has been used.

Figure 1 presents the wiring diagram for the stepper motor control circuit.

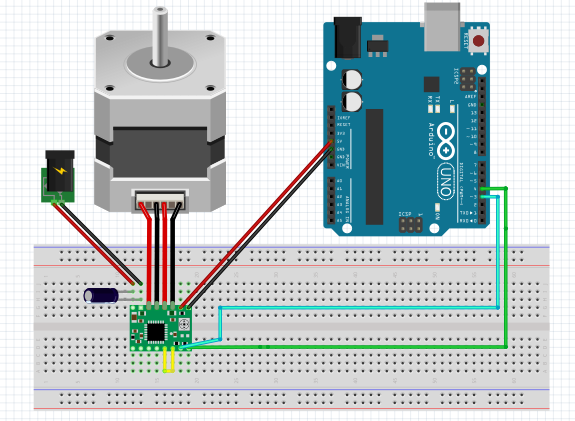


Figure 1- Stepper Motor Control Circuit

As the aim of controlling the motor was to produce various frequencies in the audible range, precise control of the rotation was not required. As a result, the pins of the driver board that control the step resolution of the motor (full step, half step, etc.) have been left disconnected, leaving the motor to operate in the full step mode. Additionally, the STEP and DIR pins of the driver board have been connected to ports 3 and 4 of the Arduino. The STEP pin is used for controlling the movement of the motor with each pulse sent to this pin being converted to a step rotation and the DIR pin is used for controlling the direction in which the motor spins. Additionally, a decoupling 100 uF capacitor has been used as indicated in [1] in order to protect the board against voltage irregularities. In setting up the control circuit, the guide presented in [2] has been used for ensuring the wiring was done properly so as not to damage the components.

### Software

The stepper motors are driven by an ATmega328P Arduino microcontroller and an A4988 stepper driver. Outputting a square wave from the Arduino to the stepper driver causes the stepper to rotate. If the frequency of the square wave is matched to that of individual notes, then the sound produced by the stepper matches that of the note. By combining various notes together at different time intervals musical pieces can be played. Three different methods have been looked at for the generation of the square wave at specific frequencies. **Timers/PWM (andrei’s bit)** , a 555 timer and a digital signal synthesizer IC.

555 Timer: The LM555 timer is an IC capable of generating specific frequencies at various duty ratios by connecting two resistors and two capacitors [1]. Multiple ICs would be used, one for each note and since the IC has an enable pin the Arduino would be able to turn it on/off with ease. This is a good solution as a lot of the software is now simplified due to the hardware used. However, more hardware is needed making it more expensive to produce. A closer investigation on the components needed to produce some of the frequencies resulted in resistor values in the microohm magnitude. This poses two issues, first, it is hard to precure micro-ohm resistors and most importantly the datasheet recommends resistor values of 1KΩ to 1MΩ [1]. This method has therefore been ruled out.

Digital Signal Synthesizer: A DSS is an IC capable of generating a vast range of frequencies. The DSS considered for this application is the AD9837. The AD9837 is capable of producing frequencies between 0-5MHz, with a 0.02 Hz resolution making them incredible accurate [2]. The IC requires no external components (apart from a decoupling capacitor) and the frequency is set by three internal registers. The IC communicated with the microcontroller via SPI [2]. This is advantageous as it can be easily reprogrammed to another frequency, allowing one IC to play multiple notes. As expected the IC is more expensive than the 555 timer. The team is currently designing a breakout PCB for it in order to test it. The AD9837 offers a promising solution, and will most probably be used.

## Tesla coils

# Progress and Planning

Since this project has a lot of different components to be successful an effective and detailed plan was crucial to be successful. The primary method for organising this project has been using a Gantt chart and conducting team progress reviews every 3 weeks.

One of the things that had to be avoided when organising the schedule for this project is creating all of the instruments and the conductors in a serial manner, i.e. finish one and then create the next. The total allocated time to the project is 17 weeks (85 days, not including weekends) so it crucial to have the instruments being built in parallel to one another. This also ensures that if there is a delay in one of the instruments (i.e. some components are taking longer than expected to be delivered) then progress can be made on a different instrument, ensuring that team members are not being underutilised.

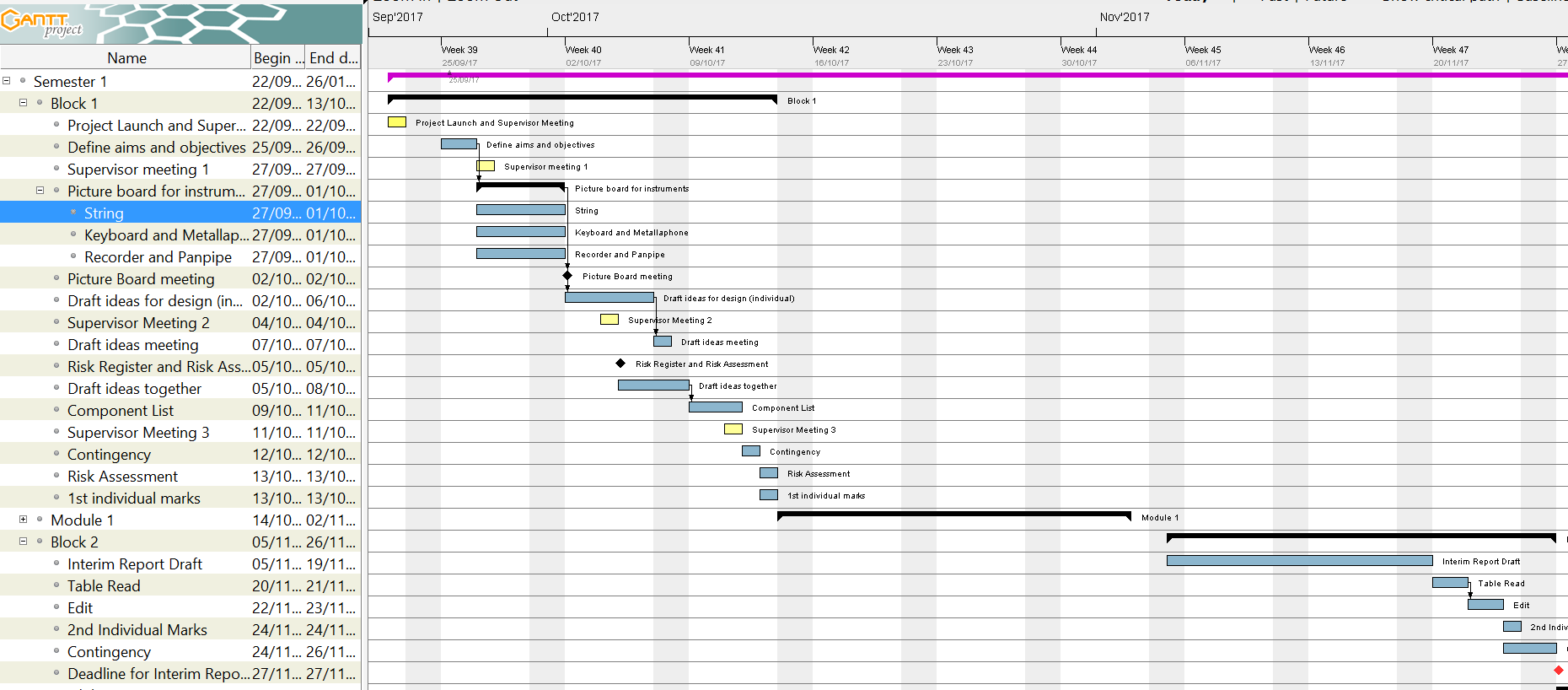
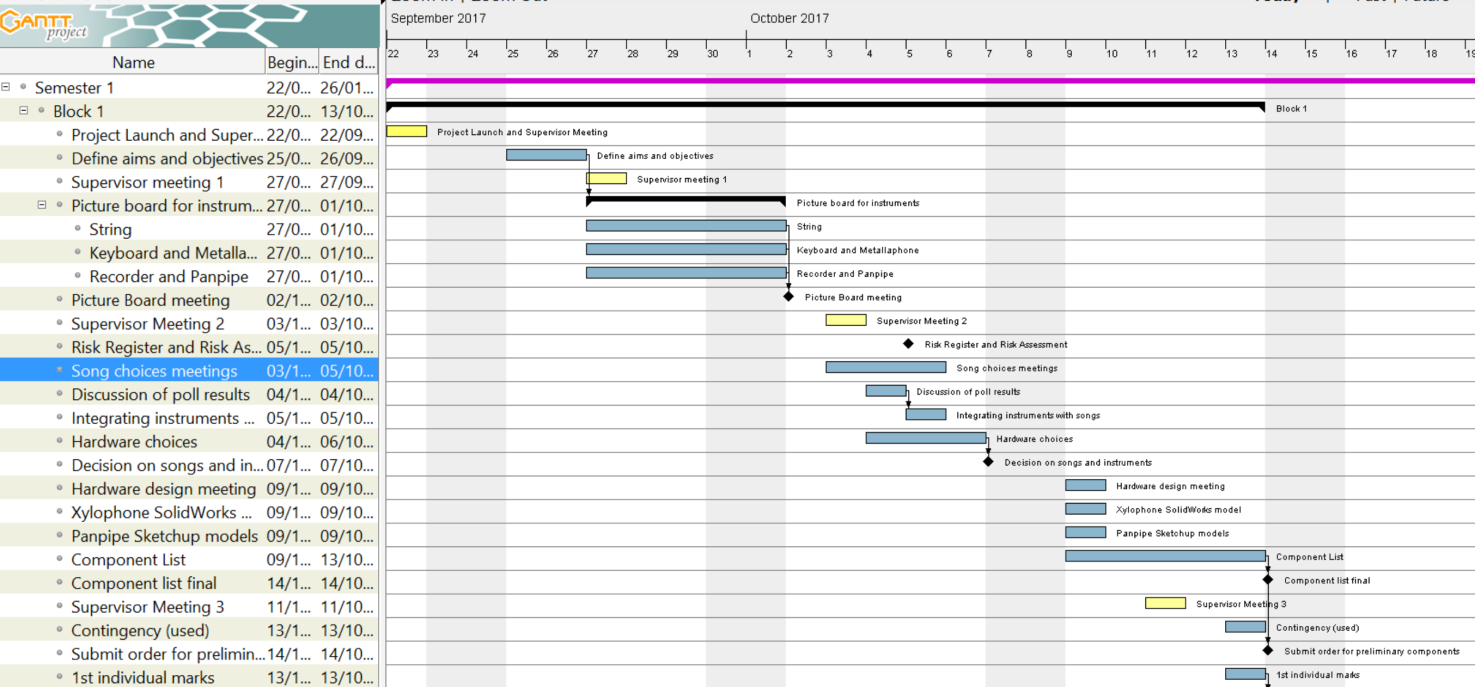


Figure 4.1 shows the first version of the Gantt chart. The initial stages of the project were left very open ended so the first tasks in the Gantt are dedicated towards narrowing down the options for the instruments. It was important that one of the very tasks was to define the aims and objectives of the project; this was followed by defining the different categories of instrument to be attempted during this project. The initial research period started on the 27/09/17 through to 01/10/17 and culminated in a picture board meeting. During the Picture Board meeting the instruments the research each individual team had found was discussed in order to find the instruments that were going to be built: panpipes, guitar, xylophone and keyboard. Deciding on the instruments to be made was a significant step in the project because it now meant more definitive milestones could be set and a more detailed plan could be created to ensure the project was a success. A milestone was created for the end of the first project block which was to have a component list ready. This meant that all necessary electronic components and hardware for the first prototypes of the instruments could be ordered to be worked on in the second project block.

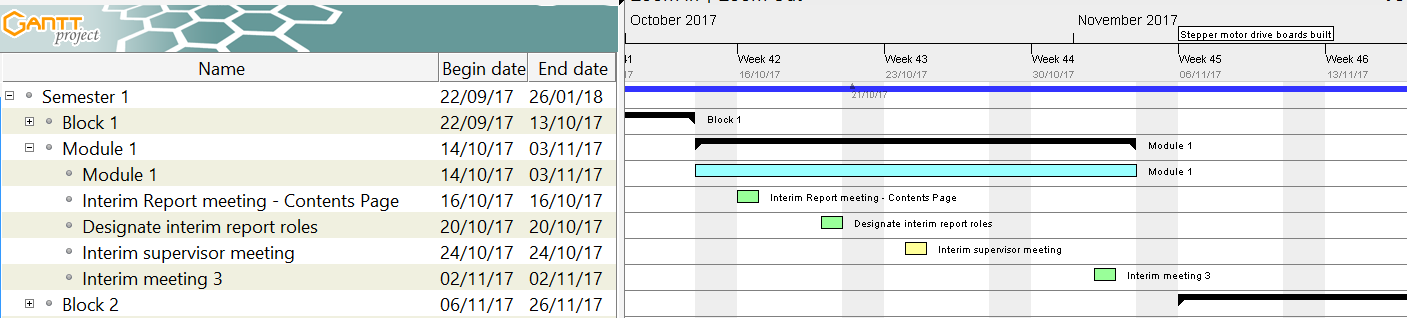
Initially, this was going to the only goal for the end of the first project block. However, specifically after *Supervisor Meeting 2*, it was realised that the instrument designs that were a product of the Picture Board meeting were far too complex to be achievable in this project. This may have been because all of the robot instruments that had been researched were individual projects so it was difficult to think of them as being quite complex designs. It was important to recognise that because the objective of this project was to build four robots the designs would have to be quite simple in order to make them achievable within the given time frame, so an adaption of each of the initial instruments was used to progress with the project. The robot guitar was also dropped in favour of a significantly simpler stepper motor design, primarily because the several iterations that were produced for the robot guitar were far too complex to be done in the time frame given.

In addition to this, one of the objectives was to have the four instruments playing two recognisable songs. It was realised that designing the hardware without thought towards the songs that they are being designed to play will make the designs more complicated. Deciding on the songs in this early stage ensures that the designs can be made as simple as possible.



As can be seen from Figure 4.2 several new tasks and milestones were added to tackle the issues that were highlighted during the second supervisor meeting. It was decided that the second week should end with both the songs and hardware choices being made meaning that the hardware could be designed with specific song choices (and therefore notes) in mind. This made the design for some of the robots significantly easier, for example the keyboard was designed to only play 19 different notes instead of the 61 it was capable of. To decide on the song choices a Facebook poll was created where individual team members could vote on different choices, since there are only 6 members it was easy to get the votes from each member and the poll only had to be live for a day.

The advantage of using a Facebook poll for this is that it allows individual members to add their own suggestions as well as easily tracking the number of votes each song achieves. The next day a meeting was held to discuss the most voted song choices and integrate them with the MIDI files of the songs using Anvil Studio. In order to make sure that the *Submit Order for Preliminary Components* milestone could still be completed on time the songs and instruments needed to be selected by the end of the second week as it would take a considerable amount of time to produce 3D models and get component lists. The milestone for the first project block was achieved and a component list was constructed and ordered which gave 3 weeks for the components to arrive, ready to be used at the start of the next project block.



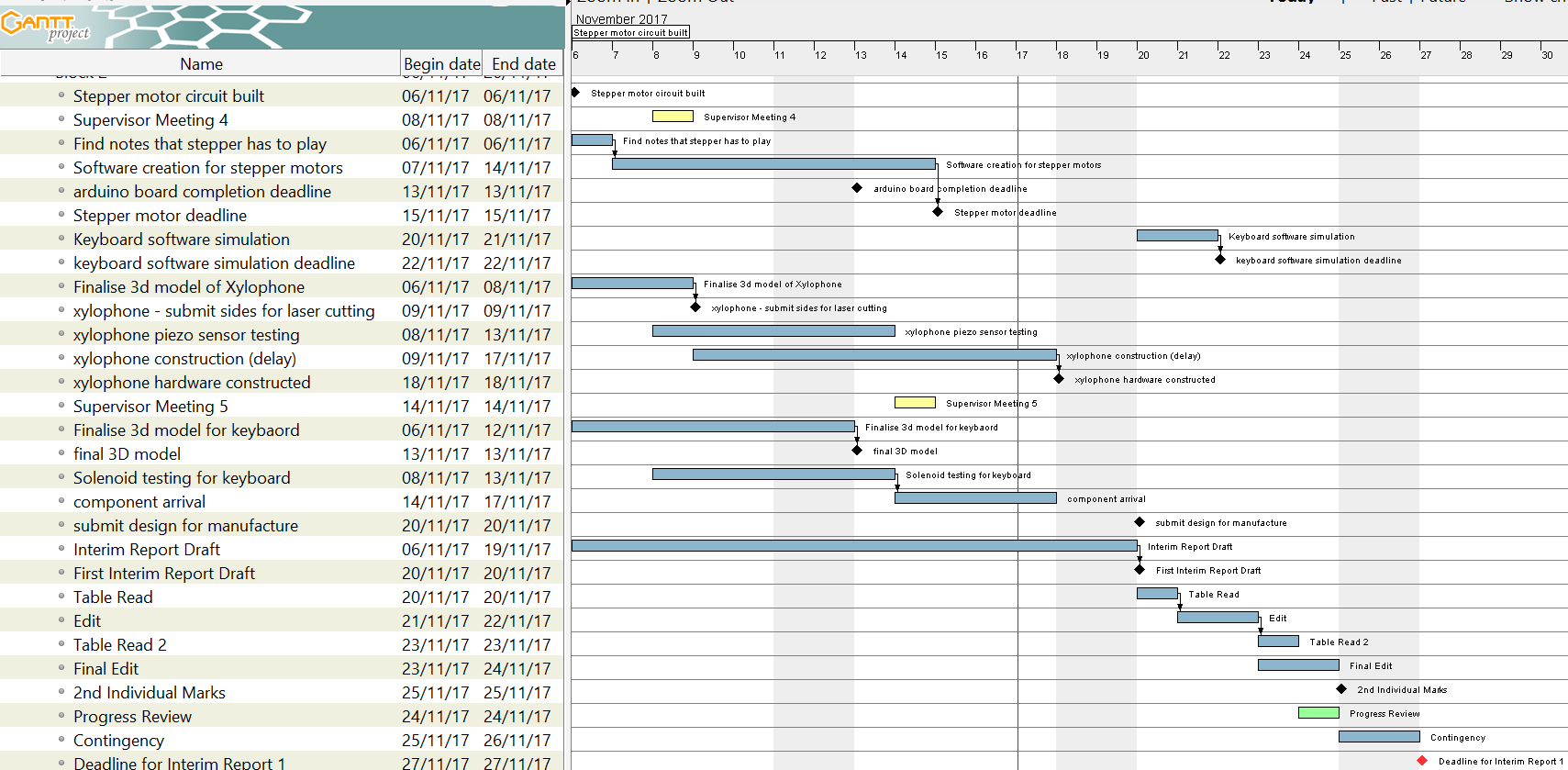
During the first module block the priorities for the group switched from making significant progress on the project to focusing on individual modules. The work done on the project during this time focused mostly on preparing for the upcoming tasks of the second project block and preparing for the first significant project deadline – *Interim Report 1.* It was clear that this milestone would require a significant amount of work would need to be done to complete this milestone well. So, in order to pre-empt the workload, and ensure that no time was wasted planning during the second project block, several meetings were held where the approach to achieving the deadline for the interim report was discussed. The first meeting (*Interim Report Meeting – Contents Page)* was used to create a contents page for the report which was used in the next meeting (*Designate Interim Report Roles*) to designate individual sections to individual team members. There was another meeting (*Interim Meeting 3)* which took place two days before the start of the second block. This was deliberately close to the start as it was used to discuss the plan and milestones for the second project block.

The second project block was significantly harder to schedule because there were more milestones and they often required the same people working on them. The milestones for the end of the second project block were:

* Software for stepper motors created
* Keyboard software simulation created
* Construct the hardware for the xylophone
* Keyboard hardware constructed
* Submit the First Interim Report

To make sure that these milestones were achieved several of the tasks were run in parallel. Also, in project block two the team was split in to a hardware team (four members) and software team (2 members). The slight bias towards the hardware team is due to the higher amount of work to be completed on the design and construction of the robot instruments. The reason for choosing the stepper motor software to be created first is because of the low hardware requirements that the stepper motors had, the circuits were quite simple and one could be connected to the Arduino with relative ease. The software development could start from early on in the project block.

Fortunately all of the components that were ordered in the first project block arrived on time so the drive boards and power supply necessary for the steppers to operate were ready to be used by the software team. Several of the components had been delivered for the keyboard and xylophone so a testing phase began. Each of the testing phases was started with a goal in mind. For example, the *Solenoid Testing for Keyboard* (Figure 4) task was created to check whether the solenoids that had been ordered in the first project block were strong enough to push down the keys, if there were not then completing this phase would ensure there was enough time to order more in and still make progress on the keyboard hardware.



Highlighted in blue in Figure 4.4 is the approach towards the interim report. A draft of the report was to be made ready by the start of the last week of the block, 20/11/17 (a week before the deadline). This ensured that there was enough time to conduct a table read and perform any edits. It was debated whether a week may be too long and that the time may be better spent working on the technical progress of the project and having more time to write the report, pushing the deadline for the draft to Wednesday (22/11/17). Due to the size of the report and the amount of time it would take to conduct a table read the initial date was settled on and remained as the deadline for the first draft. Since the edit was spaced out over a week it had the added benefit of being able to still make technical progress with the project. This approach worked well, the interim report was submitted on time and there was ample time to edit the document and have it ready. Spacing the editing out over the course of a week resulted in the team still being able to meet the deadlines for modules outside of the project.

## Approach towards 2nd Semester

At the end of the first semester there are still several tasks left to be accomplished, according to each of the robot instruments. Currently, there are four people assigned to the hardware team and two assigned to the software team. This shall be changed going into semester 2 as the requirements for the hardware side of the project begin to decrease; the teams shall be balanced out with three members in each.

Keyboard

The software simulation for the keyboard is currently being developed. It is not the complete software because the hardware for the keyboard has not been fully constructed yet. The hardware is not complete so at the start of the next project block one of the first tasks will be to integrate the software simulation with the keyboard hardware. This should not take more than 2-3 days. Once this has been completed the keyboard instrument will be completed.

Xylophone

The hardware for the Xylophone has now been delivered and constructed. One of the milestones at the end of the 2nd Project Block was to have the hardware for the Xylophone finished and constructed.

Panpipes

Not much progress has been made on the panpipes this semester, this is mostly due to the delay of the power supply to power the air pump. However, an adjustable nozzle that holds the pipe from the air pump has been created. The plug for the panpipe was delivered in the final week of the second project block so testing is underway. The tests should determine whether or not the panpipes can be used as an instrument given the time constraints of the project, if they can then a frame needs to be constructed to hold the panpipes. During the final week of the second project block another option for the panpipe design was suggested which is now being investigated. It utilises a servo motor to rotate the nozzle from the air pump in a semi-circular, the actual pipes will be placed at set positions along the semi-circle. In order to progress with this design some testing needs to be conducted on an array servo motors. Two members of the hardware team will be dedicated to this task which will run from the beginning of the third project block. It will mean that during the next module block several servo motors will have to be ordered so that they are ready to be tested in the next project block. Once this testing is complete a 3D model will be created, this will help the manufacturing workshop in creating several components of hardware. Two members of the hardware team will be fully dedicated towards the panpipe in Block 3 since this needs the most attention. At the end of the next project block the hardware of the panpipe robot should be completed. The full instruments are scheduled to be completed partway through the second week of Block 4. This gives 8 full days (excluding weekends) to create the software for the panpipes and integrate it with the panpipes. 2 members of the software team will be focused on creating the software for the panpipes during Block 4.

Stepper Motors

Tesla Coil

Conductor Design

## Risk Management

In order to identify the potential risks that project faces a risk register was established, which allowed some mitigations to be put in place. The full risk register can be seen in **Appendix 1.1.**

Risks 1, 5 and 9 are not from direct action of the team and have to be mitigated. One of the ways in doing this is to add contingency time at the end of each block. From Figure 4 and Figure 2 it can be seen that this was done, giving two days at the end of each block for tasks to overrun. During the second project block the testing phase for the panpipes was supposed to begin at the start of the second week, so that the feasibility of the panpipes could be determined. However, since the plug that was needed to power the air pump was delayed until the end of the second week the testing phase was delayed. The contingency time attached to the end of the second project block had to be used to compensate for the delay, giving two additional days to work on the testing for the panpipe. Risk 5 should not be much of an issue (shown by its *Probability being equated to 1*) since a locked cupboard has been provided, all built equipment shall be stored in there and the key is removed from the room.

Risk 2 has the potential to cause significant setbacks in the project, especially since the team has been split into hardware and software. However, the risk is somewhat reduced by having several members in each team and more than one person on each task. This ensures that more than one person is familiar with each task that is currently ongoing with the project so if someone was to become unavailable then another member will be able to pick up from the last member that left it. Also, the team has weekly meetings where the whole team is present (this is in addition to the supervisor meetings). The primary goal of these meetings is to check on progress made in the previous week but they also allow members to update each other on specific tasks. This means that if another member has to pick up the work of another there is only, at the maximum, a five day gap in the workload.

Risk 4 is reduced by using Google Drive. Since Google Drive is a completely online file storage system every file can be accessed by each individual member. This acts as a good back up to the local files stored on individual laptops.

Risk 7 is due to none of the team members having previously used a MyRIO (**figure from introduction regarding skills)**. In order to mitigate this risk two actions were taken. Firstly, the National Instruments Student Scholarship was applied for, where one of the benefits is being provided with a National Instruments Engineer. Then a training session was organised with a National Instruments Engineer to provide an introduction to the MyRIO and to demonstrate its capability. In order to further reduce this risk all members of the team will be present at the training session so that everyone can be assumed to be at the same level for the MyRIO after. This ensures that individual team members can provide support to others should there be problems using it.

Risk 8 is mitigated by constructing components lists before ordering a significant amount of components (such as at the end of the first project block). This gives the opportunity to review the items being ordered and ensure that they are within budget; it also gives time to look for alternatives to the products ordered. Some may be cheaper if factors such as delivery time are more flexible. For example, at the end of the first project block there was a three week period that the items could be delivered in so cheaper alternatives for the solenoids could be selected. These were exactly the same solenoids that were going to be ordered but because they had a longer delivery time the total cost per solenoid was cheaper.

## Minutes and agendas

A project meeting is held each week with the team and project supervisors Danielle George and William McGenn. For the meeting an agenda is produced by the Secretary and is distributed a day before the meeting. Before it is sent out it is reviewed by the team to make sure everyone’s work for that week has been included so all progress made can be presented effectively. Minutes are taken during the meeting which are typed up and uploaded to the shared drive within two days so the team can look at the actions required for that week. The minutes include an ongoing actions list, a section detailing actions from the previous week (confirming if they had been completed or not) and a section containing a summary of the topics discussed detailing the actions that the meeting produced.

## Logistics

WhatsApp was chosen as the communication platform and two groups were set up with one with the team and one with the team and supervisors. The storage platform that is to be used for sharing and saving document was chosen to be google drive as all members of the team had access to it.

The project group was split up into two with a hardware side and a software side. In the hardware group was Joyanto, Josh, Anton and Theodore and on the software side was Andrei and Francesco. Joyanto as initially in the software group but it was realised that initially more work on hardware was needed to get the robots built ready for programming. Meetings were arranged either using the WhatsApp group or at the end of a previous meeting.

## Procurement

The team has an allocated budget of £1500 and so far, £506 have been spent on musical instruments, electronic components and building materials. The full expenditure record is presented in Appendix A. Although this represents approximately a third of the budget, it is worth mentioning that a substantial amount was spent in the beginning on buying the musical instruments (keyboard, stepper motors, etc.) and the components required to set them up (ex: power sources). These expenses accounted for approximately £250 and a key aspect to be taken into account is that these were nonrecurring expenses, thus reducing the risk of a future budget exceeding.

The current financial plan leaves £200 to be used as a reserve for unexpected expenses that accounts for possible component failures or improper equipment being bought during the prototyping stages, leaving a current usable balance of approximately £800. In terms of future expenses, the biggest pressure on the budget is posed by the solenoids used for pushing the keys of the keyboard as these are expensive components and a large number is required for the project. A solenoid costs approximately £10 per piece and the team estimates that 25 will have to be bought in the next stages, thus running a cost of £250. To minimise this risk, single solenoids will be tested in advance to ensure that the large order will meet the project requirements.

Another large expected expense is that of the building materials necessary for constructing the mechanical ensemble of the robots, with the team estimating this cost at £200. Considering the available balance, the cash reserve and the future large expected expenditures, approximately £350 are left to be used for electronic components. As these have a generally low price, the team expects to finish the project within the allocated budget.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | **Quantity** | **Price/piece** | **Total Price** | **Date** | |
| Keyboard | 1 | £66 | £66 | 11 Oct 2017 | |
| Panpipes | 1 | £65 | £65 | 11 Oct 2017 | |
| Solenoids | 10 | £4.42 | £44.20 | 11 Oct 2017 | |
| Stepper Motors RS | 4 | £7.00 | £28.00 | 11 Oct 2017 | |
| Tesla Coil Kit | 3 | £9.29 | £27.87 | 11 Oct 2017 | |
| Jumper Cables set | 1 | £6.45 | £6.45 | 11 Oct 2017 | |
| 4 Stepper Driver Boards | 1 | £4.97 | £4.97 | 11 Oct 2017 | |
| 12V 2Amp Power Supply | 1 | £3.78 | £3.78 | 11 Oct 2017 | |
| HDMI Adaptor | 1 | £2.39 | £2.39 | 11 Oct 2017 | |
| Crystal SMD 16 MHz | 5 | £0.32 | £1.59 | 17 Oct 2017 | |
| 20V Diode | 10 | £0.19 | £1.91 | 17 Oct 2017 | |
| Reg Standard Lin Fix | 5 | £0.41 | £2.06 | 17 Oct 2017 | |
| Mini USB B Type | 5 | £1.38 | £6.92 | 17 Oct 2017 | |
| USB to serial TTL | 2 | £2.83 | £5.65 | 17 Oct 2017 | |
| 8bit MCY ATMEGA | 2 | £1.46 | £2.93 | 17 Oct 2017 | |
| Panpipes Customs Charges |  |  | £16.26 | 25 Oct 2017 | |
| USB to serial TTL | 2 | £2.83 | £5.65 | 30 Oct 2017 | |
| Tactile Switch | 10 | £0.30 | £3.00 | 30 Oct 2017 | |
| 20uf Capacitor | 10 | £0.16 | £1.56 | 6 Nov 2017 | |
| 2.2uF Capacitor | 10 | £0.09 | £0.86 | 6 Nov 2017 | |
| Blue LED SMD | 20 | £0.26 | £5.10 | 6 Nov 2017 | |
| Green LED SMD | 50 | £0.12 | £6.15 | 6 Nov 2017 | |
| RED LED SMD | 50 | £0.12 | £6.15 | 6 Nov 2017 | |
| 12V Solenoid | 1 | £12.42 | £12.42 | 9 Nov 2017 | |
| Aluminium Strut | 1 | £18.98 | £18.98 | 10 Nov 2017 | |
| Arduino Boards | 3 | £4 | £12 | 10 Nov 2017 | |
| Air Pump Power supply | 1 | £10.00 | £10.00 | 10 Nov 2017 | |
| Breadboard | 4 | £2.20 | £8.80 | 10 Nov 2017 | |
| STM32 Nucleo Board | 1 | £8.26 | £8.26 | 10 Nov 2017 | |
| 12V 2A Power Supply | 1 | £2.33 | £2.33 | 10 Nov 2017 | |
| Threaded Rod | 1 | £1.92 | £1.92 | 10 Nov 2017 | |
| 12V 72W Power Supply | 1 | £31.75 | £31.75 | 13 Nov 2017 | |
| Stationery |  | £2.80 | £2.80 | 14 Nov 2017 | |
| Jumper Wire set | 2 | £1.79 | £3.58 | 16 Nov 2017 | |
| Waveform Generator IC | 2 | £2.97 | £5.93 | 16 Nov 2017 | |
| 5MHz XO SMD | 1 | £1.32 | £1.32 | 16 Nov 2017 | |
| 15pF Capacitor | 25 | £0.02 | £0.58 | 16 Nov 2017 | |
| R/C servo motor | 1 | £10.41 | £10.41 | 16 Nov 2017 | |
| Cabinet Organiser | 1 | £18.96 | £18.96 | 16 Nov 2017 | |
| See-through plastic sheet | 1 |  | £28 | 17 Nov 2017 | |
| 12V 3W Solenoid | 1 | £10.40 | £10.40 | 17 Nov 2017 | |
| NPN 60V 5A Transistor | 10 | £0.33 | £3.33 | 17 Nov 2017 | |
| Total Budget Used | | | | £506 | |
|  |  |  |  |  | |
|  |  |  |  | |
|  |  |  |  | |

## Conclusion

# References

2.1[1] The University of Manchester, “European City of Science | The University of Manchester | Science and Engineering,” The University of Manchester, May 2016. [Online]. Available: http://www.se.manchester.ac.uk/about-us/ecos/. [Accessed November 2017].

3.4[1] TeamDare video

3.1.1 [1] <http://www.raspbian.org/>

[2] <https://www.raspberrypi.org/products/raspberry-pi-3-model-b/>

[3] <https://learn.sparkfun.com/tutorials/what-is-an-arduino>

[4] <https://store.arduino.cc/usa/arduino-uno-rev3>

[5] <http://www.atmel.com/images/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-328P_datasheet_Complete.pdf>

[6] <http://www.atmel.com/Images/Atmel-2521-AVR-Hardware-Design-Considerations_ApplicationNote_AVR042.pdf>

[7] <https://www.arduino.cc/en/uploads/Main/ArduinoNano30Schematic.pdf>

[8] <http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT232R.pdf>

[9] <https://www.ni.com/en-gb/shop/select/myrio-student-embedded-device>

[10] <https://os.mbed.com/questions/60867/what-are-the-programming-languages-suppo/>

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3.1.2 [1] https://www.midi.org/specifications/item/the-midi-1-0-specification

[2] http://www.petesqbsite.com/sections/express/issue18/midifilespart1.html

[3] http://www.indiana.edu/~emusic/361/midi.htm

[4] https://www.midi.org/articles/an-intro-to-midi

[5] https://www.csie.ntu.edu.tw/~r92092/ref/midi/

[6] https://mido.readthedocs.io/en/latest/

[7] <https://docs.python.org/3.6/library/re.html>

3.1.3

[1] Frequencies of Musical Notes, A4 = 440 Hz", *Pages.mtu.edu*, 2017. [Online]. Available: https://pages.mtu.edu/~suits/notefreqs.html. [Accessed: November 2017].

[2] "delayMicroseconds()", *Arduino.cc*, 2017. [Online]. Available: https://www.arduino.cc/reference/en/language/functions/time/delaymicroseconds/. [Accessed: November 2017].

[3] "millis()", *Arduino.cc*, 2017. [Online]. Available: https://www.arduino.cc/reference/en/language/functions/time/millis/. [Accessed: November 2017].

[4] "analogWrite()", *Arduino.cc*, 2017. [Online]. Available: https://www.arduino.cc/reference/en/language/functions/analog-io/analogwrite/. [Accessed: November 2017].

[5] “Timer1", *Arduino.cc*, 2017. [Online]. Available: https://playground.arduino.cc/Code/Timer1. [Accessed: November 2017].

3.5.1

[1] Allegro MicroSystems, “DMOS Microstepping Driver with Translator and Overcurrent Protection””, A4988 datasheet, 2014

[2] Stepper Motor - How It Works - HowToMechatronics", *HowToMechatronics*, 2017. [Online]. Available: http://howtomechatronics.com/how-it-works/electrical-engineering/stepper-motor/. [Accessed: November 2017].

3.5.2

[1] <http://www.ti.com/lit/ds/symlink/lm555.pdf>

[2] <http://www.analog.com/media/en/technical-documentation/data-sheets/AD9837.PDF>

3.3

[6] BICRON Electronics Company, “Standard & Customer Solenoids for OEM Application,” BICRON Electronics Company, Connecticut, 2011.

[7] Mecalectro, “Solenoid - 8.M10.02.62,” Mecaelectro, [Online]. Available: http://uk.rs-online.com/web/p/dc-d-frame-solenoid/3073326/. [Accessed October 2017].

2.4.1

[1] J. Shauw, “Dream wedding played by piano robot,” YouTube, 7 November 2010. [Online]. Available: https://www.youtube.com/watch?v=IwuiOwZrzCM. [Accessed October 2017].

[2] Teotronica, “robot playing piano,” YouTube, 14 September 2009. [Online]. Available: https://www.youtube.com/watch?v=HHcgTbs7\_Os. [Accessed October 2017].

[3] R. Components, “Pull Action DC D-Frame Solenoid, 10mm stroke, 3W, 12 V dc,” RS Components, [Online]. Available: http://uk.rs-online.com/web/p/products/9059931/?grossPrice=Y&cm\_mmc=UK-PLA-DS3A-\_-google-\_-PLA\_UK\_EN\_Automation\_And\_Control\_Gear-\_-Solenoids-\_-PRODUCT%20GROUP&matchtype=&gclid=CjwKCAiArrrQBRBbEiwAH\_6sNJzgtbESwIrdKI-Zg\_Uk7bO-MXjZdbP6-saPSDEiDa1C5HUHy4zY7Ro. [Accessed November 2017].

[4] Society of Robots, “Actuators - Solenoids,” Soceity of Robots, 9 August 2006. [Online]. Available: http://www.societyofrobots.com/actuators\_solenoids.shtml. [Accessed October 2017].

[5] H. A. Radi and J. O. Rasmussen, Principles of Physics For Scientists and Engineers, Berkley, CA, USA: Springer, Berlin, Heidelberg, 2013.

[6] BICRON Electronics Company, “Standard & Customer Solenoids for OEM Application,” BICRON Electronics Company, Connecticut, 2011.

2.4.4

[1] O. Engineering, "What is a stepper motor? - Principles, types and controllers", *Omega.co.uk*, 2017. [Online]. Available: https://www.omega.co.uk/prodinfo/stepper\_motors.html. [Accessed: November 2017].

[2] "Stepper Motors | NEMA Stepper Motors & Controllers", *Circuitspecialists.com*, 2017. [Online]. Available: https://www.circuitspecialists.com/stepper-motor. [Accessed: November2017].

[3] Stepper Motor - How It Works - HowToMechatronics", *HowToMechatronics*, 2017. [Online]. Available: http://howtomechatronics.com/how-it-works/electrical-engineering/stepper-motor/. [Accessed: November 2017].

[4] "How do stepper motors work?", *Explain that Stuff*, 2017. [Online]. Available: http://www.explainthatstuff.com/how-stepper-motors-work.html. [Accessed: November 2017].

[5] M. Pilhofer and H. Day, *Music theory for dummies*.

2.4.3

|  |  |
| --- | --- |
| [1] | Learnitonlinetoday, "How to Play Trumpet- Lesson #1 Beginner," Learnitonlinetoday, 9 January 2010. [Online]. Available: https://www.youtube.com/watch?v=2SjH0qhQpoU. [Accessed 19 November 2017]. |
| [2] | WikiHow, "wikiHow to Play the Trumpet," WikiHow, [Online]. Available: https://www.wikihow.com/Play-the-Trumpet. [Accessed 19 November 2017]. |
| [3] | A. Lim, "Toyota Trumpet Robot (Subtitles in English and Japanese)," 4 January 2010. [Online]. Available: https://www.youtube.com/watch?v=6fctULDctuA&t=107s. [Accessed 19 Novemeber 2017]. |
| [4] | BotJunkie, "Waseda Flutist Robot," 1 November 2008. [Online]. Available: https://www.youtube.com/watch?v=jx8U1FgILCE. [Accessed 19 November 2017]. |
| [5] | WikiHow, "wikiHow to Play the Panpipe or Pan Flute," [Online]. Available: https://www.wikihow.com/Play-the-Panpipe-or-Pan-Flute. [Accessed 20 November 2017]. |
| [6] | "how to play the recorder," arta recorder, [Online]. Available: http://www.arta-recorder.org/. [Accessed 20 November 2017]. |
| [7] | T. Dare, "TeamDARE's robot band plays 'Rolling in the deep'," TeamDare, 27 August 2011. [Online]. Available: https://www.youtube.com/watch?v=RKpKygGX2Nk. [Accessed 20 November 2017]. |