Music from Biosignals

THESIS

Adam De Pierro
FAN: depi0012
Student ID: 2151628
Academic Supervisor: Associate Professor Kenneth Pope

ACKNOWLEDGEMENTS

I, Adam De Pierro, state that all of the work presented in this paper is my own unless otherwise specified. I formally acknowledge the assistance of Associate Professor Kenneth Pope with music generation using the MATLAB software and Lucas Paix and Lucas Moss with their assistance in using the Altium Designer software for my PCB design. I also acknowledge the other students of Associate Professor Kenneth Pope for their ongoing overall support throughout this project.

ABSTRACT

The Music from Biosignals project is an exceptionally large project when considering every feature of the system involved. The system itself can be divided into four main subsections:

Phase 1 – Obtain a biosignal

Phase 2 – Transmit biosignal data

Phase 3 – Process the data

Phase 4 – Play the music

To achieve a greater understanding of the entire project, research was conducted into each of the phases. After conducting a review into the existing literature it was found that people had researched into parts of this project, but nobody had pieced them together as this project does.

Using an online set of ECG data, attention can be initially focused on Phase 4 where MATLAB is used to process the data to create music. The first sounds were basic two-tone patterns which developed into an ascending pattern. The code was refined to give these sounds musical meaning. Understanding the differences in pitches between musical notes and the separations in musical scales, equations can be written, and these sounds can be replicated in MATLAB. The code in this case is designed to monitor the mouse cursor on the screen. As the mouse travels along the screen the pitch increases by the equations used for each scale.

As the project progressed, my focus settled on Phase 2. Initial electronics were investigated to determine how the circuitry should be designed. It was through this research that the MAX30003 chip was found which reads in ECG with no additional circuitry.

With a basic understanding of how the system should work, the on-body PCB could now be designed. There were many different aspects to consider in the PCB design process, all of which I had very minimal experience in. There was an incredibly steep learning curve with this entire process and in particular navigating the Altium Designer program. It was eventually found that this process can be split into four main steps:

Step 1 - Build schematic parts

Step 2 - Design footprints for each schematic part

Step 3 - Route schematic

Step 4 - Layout PCB design

Each step took a large amount of time as I developed a better understanding of the design process and continued to rectify errors as they were encountered. After multiple interactions with Engineering Services at Flinders University to ensure the design worked, in the end there was simply not enough time to fix every issue and manufacture the board.

Although there was insufficient time to properly complete the PCB design it was still an incredibly worthwhile learning experience, and the knowledge and confidence gained can be taken and applied to future PCB designs.

Table of Contents

TABLE OF FIGURES

Table of Tables

1 LITERATURE REVIEW

1.1 BACKGROUND

1.1.1 What is Music?

The Oxford Dictionary defines “music” as “Vocal or instrumental sounds (or both) combined in such a way as to produce beauty of form, harmony, and expression of emotion” (Oxford, 2019).

This project aims to deliver a wearable electronic device which dancers can utilise to create the music that they dance to. This device will study the movements and biosignals from the body and output the appropriate sounds. During the early stages of this project, the predominant focus of “music”, with regards to its formal definition, will be *instrumental sounds producing form, harmony and expression of emotion*.

The reason why this is the key area of research is because initially instrumental sounds are a more achievable goal, but it is crucial that this music have form and sound pleasant while also maintaining an expression of emotion. An example of expression of emotion can come in the form of fast music if the dancer moves with an increased heart rate.

Music takes many different forms and can include a multitude of scales and modes to produce a combination of various sounds. There will be no specific music type or genre that will be focused on in this research as it is too broad to consider at this stage.

1.1.2 What is a Biosignal?

A biosignal is defined as “any signal in living beings that can be continually measured and monitored. The term biosignal is often used to refer to bioelectrical signals, but it may refer to both electrical and non-electrical signals” (Nait-Ali, 2009).

There are a multitude of biosignals and some of these include Electromyography (EMG) which is the biosignal produced from muscles, Electrocardiography (ECG) is the electrical signal produced from a heartbeat and Electroencephalography (EEG) which studies the electrical activity from the brain. The research presented here will have a focus on creating music from the properties of an ECG signal and understanding its key characteristics.

1.1.3 Electrocardiogram (ECG)

An ECG measures the electrical activity generated by the heart and is used for monitoring a heartbeat and detecting irregularities (Islam et al., 2012). The ECG signal itself is further characterised by its shape which is identified by a simple PQRST plot (refer to Figure 1). In practice the main focal points of this plot is the QRS section i.e. a QRS Complex.

Figure 1: The PQRST Complex for an ECG signal (Illanes, 2006)

1.1.3.1 Noise and Interference

There are many aspects to consider when studying ECG such as the fact that the signal itself is very weak. This means that amplification is required, and it is also more susceptible to noise and interference. There exists a number of different types of noise and sources of interference and they are outlined below.

1.1.3.1.1 Interference from other Electronic Devices

Often in locations where ECG is being monitored there are a number of other electronic devices present. Due to the nature of these devices they will produce their own signals. These external signals from other devices have the capacity to interfere with the weak ECG signal. It is important to note that included in this type of interference are the signals from power lines (Joshi et al., 2013).

This interference can be attenuated out of the final ECG signal using a combination of filters in hardware and algorithms in software. One common algorithm is the Hilbert Transform which is used to improve the QRS shape (Benitez et al., 2001). It is also designed to improve the R-R peak detection. The R peak is the most prominent feature of an ECG signal and is therefore crucial in monitoring a heartbeat (Pan and Tompkins, 1985).

1.1.3.1.2 Motion Artefacts

Another major source of interference on an ECG signal are motion artefacts. As the name suggests, motion artefacts are discrepancies on the plot which are caused by the movement of the person's body (Joshi et al., 2013). For example, when monitoring an ECG signal and the person moves their arm, the outcome on the ECG plot can be very significant as highlighted in Figure 2.

As previously mentioned, the ECG signal is a very weak signal and so even minute bodily movements can have an effect on the plot. There are often a number of noise-cancelling algorithms in place to attenuate these out and as the motion artefacts are large in comparison, the QRS complex can be easily isolated (Kumar et al., 2012). The goal of these types of configurations are to reduce the noise on the signal while also enhancing the signal itself for improved efficiency of heartbeat monitoring.

Figure 2: An example of a motion artefact in an ECG signal (Strasseer, 2012)

1.1.3.1.3 Baseline Drift

One of the main sources of noise on the signal is baseline drift. Baseline drift is the drift of the ECG signal about the x-axis (Chouhan and Mehta, 2007). In theory, the signal is studied as if it travels perfectly horizontal along the x-axis, but in practice real heartbeats do not operate this way. Shown in the figure below, when a dataset of 108000 samples is plotted, the signal indeed oscillates by the fact that when zoomed in on a subsection, the QRS shape appears to travel along a horizontal line but the shape changes when zoomed out on the entire plot (dataset taken from (Physionet, 2005)).

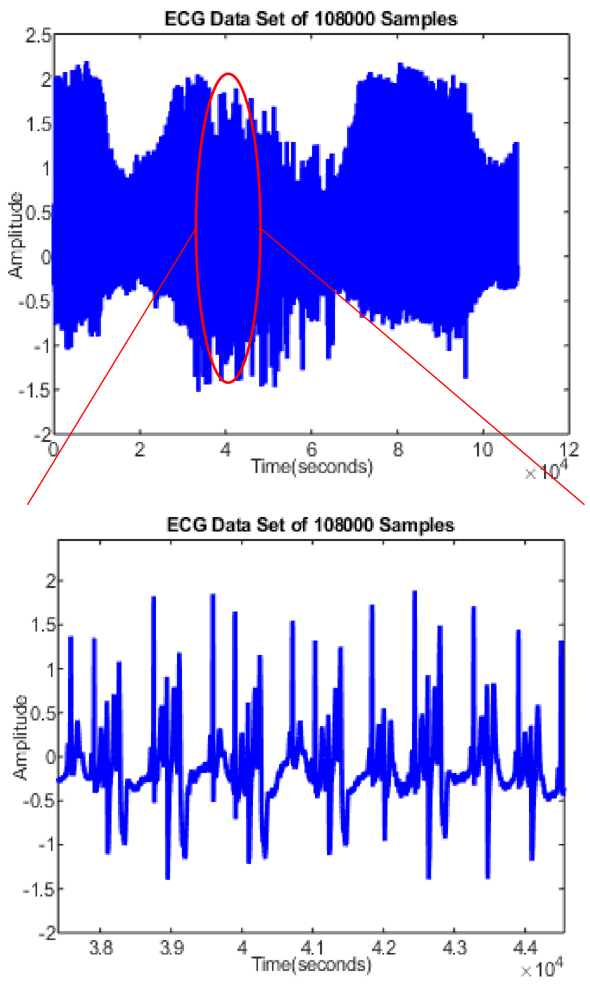


Figure 3: Baseline drift of an ECG dataset

A common software technique to resolve this issue is the implementation of a least square error correction algorithm which is designed to restrain the signal for improved readability and monitoring (Chouhan and Mehta, 2007).

1.1.4 Other Biosignals

As previously mentioned, this research focuses primarily on the ECG signal and its capacity to create sounds, however, there are a number of different biosignals that can also be considered in the creation of music. This list includes EMG i.e. biosignals from muscles, and respiratory biosignals which can be monitored by studying the changes in a respiratory band (Berry, 2016).

The benefit of studying different signals is that there will be more signal properties that can be utilised to create different music. Increasing the number of sounds that are available means an increased repertoire of music that can be created by the dancer.

1.1.5 MATLAB

The computer program MATLAB will be used for its programming language and predefined functions. It is a powerful tool containing commands that other languages do not necessarily possess or are not as easily accessible.

In the beginning of the project, in particular with music creation, MATLAB is a valuable aid and has two main built-in functions in order to play sounds; they are *sound()* and *audioplayer()*. Sound is useful for creating notes of certain pitch in a more basic manner based on a dataset and frequency whereas audioplayer is an object function that enables flexible audio functions such as pausing, resuming and defining audio call-backs (Mathworks, 2019). For this application, audioplayer appears to be the more appropriate selection as it is believed to be simpler to implement and more flexible.

1.1.6 Musical Instrument Digital Interface (MIDI)

If there is sufficient time towards the end of the project, the final stage of the music created will involve having the data sent as a MIDI stream. MIDI is a common technique used for communication between electronic devices, particularly in a musical setting (Lisle et al., 1991). The final device's ability to output MIDI is a crucial component of the system as it allows for greater flexibility of the device in the future.

There are a multitude of MIDI messages available and the table below shows the basic messages that may be of interest to this project.

Table 1: Basic MIDI Messages and their Purpose (Association, 2019)

Message	Description
Note on	The message that is sent when a note has started (depressed).
Note off	The message that is sent when a note has ended (released).
Timing Clock	When synchronisation is required, the timing clock message is sent.
Reset	Resets all receivers in the system.
Undefined (reserved)	There are multiple undefined message codes which are reserved and can be used for custom messages.

1.1.7 The System

In the beginning stages it is important to understand the key features of the project in its entirety. This will be achieved by creating a sketch of the system as shown in Figure 4.

Figure 4: The Entire System. Generic images taken from: (Duolingo, 2019), (IndiaMart, 2019) and (Shutterstock, 2019)

The above figure is a graphical demonstration of what the entire system will eventually become. The sensors (indicated by orange squares) will be placed in strategic places around the body to acquire specific biosignals. These signals will communicate with a printed circuit board (PCB) through wired connections (black straight lines) as well as via Bluetooth (blue curved lines) where appropriate.

The information will then be taken from these biosignals and sent to the off-body computer through WIFI (black curved lines). The properties sent from the PCB will be analysed by the computer to produce music. This music can be sent through a specific protocol, for example MIDI, to ensure compatibility with other musical devices.

1.2 Gaps in Literature

Humans have been able to obtain ECG signals for a long time and the concept of taking a signal and using its properties to determine an appropriate output is not necessarily a new idea. However, using this to create music is something that has not yet been done.

Additionally, the use of MIDI, or other communicative protocols, in electronics and the music industry has allowed for many advancements to ensue, specifically the simplified communication between musical devices; but this has not yet been used in conjunction with biosignals.

Essentially, the literature gap here is the fact that there has been no research into connecting the two fields. This project aims to bridge the gap between the acquisition of a biosignal, using it to create music and then having it communicate with other electronic instruments within a musical setting.

1.3 My Primary Contributions

1.3.1 Research and Development

The Music from Biosignals project is ongoing and will span many years as the technology develops. The research presented in this paper will be mostly focussed on the design of a PCB as well as using an ECG signal to create music. A key feature of this system for now and into the future will be a generic PCB that the sensors will communicate with in order to relay the information to the off-body computer.

Once it was established that this PCB design was crucial to the project, research was conducted into the field and it was found that there already exists a chip for acquiring certain biosignals. The MAX30003 chip by Maxim Integrated is a component which has been constructed for wearable applications to obtain and filter the wearer's ECG signal (Integrated, 2019b).

Further research indicated that a basic development board exists with the inclusion of the MAX30003 chip that only reads ECG data (refer to Figure 5).

This image has been removed due to copyright restrictions. Available online from [https://www.protocentral.com/4939-medium_default/protocentral-max30003-single-lead-ecg-breakout-board.jpg]

Figure 5: The MAX30003 Single-lead ECG Breakout Board (Electronics, 2019)

Another board was also found which uses this chip in conjunction with other Maxim Integrated chips to obtain the wearers ECG and heart rate, skin temperature as well as study the user's motion (refer to Figure 6).

This image has been removed due to copyright restrictions. Available online from [https://in.element14.com/productimages/large/en_GB/2668592-40.jpg]

Figure 6: The MAXREFDES100#: Health Sensor Platform (Integrated, 2019a)

The main issue with these existing boards is that there are no free I/O ports to be able to attach other sensors. This will not be an issue now as this area of research is focussing specifically on ECG, however in the future when the same analysis is conducted using other sensors and other biosignals, these will no longer be suitable.

There were a number of different routes this research could take, so a decision matrix was created to mathematically find the most appropriate solution for a PCB design.

Table 2: The Decision Matrix for PCB solution

Weights

3 = Very Important

2 = Important

1 = Must be at least considered

Factors	**Can be used now and into the future**	**Wide variety of applications (ECG, EMG, etc)**	**Cost Effective**	**Total**
Weights	3	3	1	
Option 1 – Design entirely new boards specific to my purpose with MAX30003 chip.	$1 \times 3 = 3$	$0 \times 3 = 0$	$4 \times 1 = 4$	7
Option 2 – Design entirely new generic board to communicate with the off-body computer.	$3 \times 3 = 9$	$4 \times 3 = 12$	$5 \times 1 = 5$	26
Option 3 – Use ECG Board purely for ECG .	$3 \times 3 = 9$	$1 \times 3 = 3$	$2 \times 1 = 2$	14
Option 4 – MAXREFDES100 Board which does more than ECG and can be used to communicate to an off-body computer.	$1 \times 3 = 3$	$4 \times 3 = 12$	$0 \times 1 = 0$	15

The data from Table 2 shows that the most suitable PCB solution will be the design of a new and completely generic PCB. The table shows that the most important factors in this project are the PCB's ability to communicate with a wide variety of sensors to attain different biosignals which is also closely related to the usefulness of the PCB for now and into the future.

1.3.2 My PCB

After much deliberation with Associate Professor Kenneth Pope, the functionality of the PCB was defined. This predominantly involved deciding which components to select for the design. The requirements of the board were that it must have a microprocessor with a sufficiently large number of I/O ports, Bluetooth and WIFI capability as well as an Analog-Digital Converter (ADC). Aside from these main requirements the other chip specifications did not necessarily matter. The requirements outlined above are detailed in the table below.

Table 3: PCB Requirements and Reasons

Requirement	Reason
Large number of I/O ports available from the Microprocessor	This requirement is in place because the idea of the project will be to have a number of sensors acquiring a multitude of biosignals simultaneously. Designing the board with this in mind is crucial for the long-term success of the project.
Bluetooth Capability	Bluetooth will be used to communicate the data from the compatible sensors to my PCB.
WIFI Capability	The information that the PCB receives from the biosignals will be sent from the board to an off-body computer which will receive that data and then create music from it.
Inclusion of an Analog-Digital Converter	The ADC will be responsible for the sensors which produce an analog signal. Essentially the inclusion of the ADC accounts for all types of sensors that can be considered for obtaining biosignals.

Appropriate solutions were found to each of these requirements, however, after meeting with an expert, more suitable components were identified (Cramer, 2019). Table 4 below shows the originally selected component, the new component and the reason why changes were made. It is worth noting that the originally selected components did meet the requirements stated above but the new components perform exactly the same task and are more tailored to this application.

Table 4: A study into the components on my PCB

Function	Previous Component	New Component	Reason for change
Microprocessor	ATSAM3X8CA	PIC32MX5/6/7	Although the original component met the requirement for having a sufficient number of I/O ports, it lacked Flinders University support from a debugging and operational viewpoint. The PIC microprocessors have a sufficient number of I/O ports and are compatible with the MPLAB ICD3 debugger which is readily available.
Bluetooth	CYBT-353027-02	ESP32	The only requirement of these components were to transmit data via Bluetooth and WIFI. A chip was recommended which has both transmission techniques in one package. This is a better solution and allows for a cleaner design as there are less components to consider. It also makes another I/O port available.
WIFI	ESP8266		
Analog-Digital Converter	ADS130E08	ADS1298	While both components are nearly identical, the analog front-end of the new component is built for biopotential measurements whereas the previous component is simply a generic ADC.

In terms of acquiring ECG, the MAX30003 Single-lead ECG Breakout Board shown in Figure 5 is a relatively cost-effective solution that can be connected to my PCB. This board uses the MAX30003 chip which is responsible for ECG acquisition, filtering and amplification. Therefore, it is unnecessary to include these aspects in my PCB design.

1.4 Other Applications for the Technology

There are a number of different applications for this concept. The ability to control things, or in this case create music, using biosignals can be a powerful tool.

One example would be to assist people who have limited mobility. Given that everyone is capable of producing biosignals this means that regardless of a person's physical capabilities they will still have the same biosignals as able-bodied people. If these signals are accessed properly then they can be utilised to control equipment such as wheelchairs and communication devices. This is a concept that is beyond the scope of this research and at this stage ECG might not be a feasible biosignal to control these assistive technologies.

There also exists applications of this technology in the field of rehabilitation. The biosignal produced from an elastic band which measures respiration can be used to monitor how a person is breathing. Studying the plots produced by these biosignals, the differences can be studied, and changes can be observed over time.

2 PROJECT SPECIFICATION

The system in its entirety is described in Section 1.1.7 of Chapter 1: Literature Review (refer to Figure 4 for a diagram of the entire system). Essentially the project can be divided into four main phases:

Phase 1 – Acquire a biosignal: The data collected from the signals is what will be processed to create the sounds. As previously mentioned, in order to test code and hardware that result from my contributions, I focussed my attention on the ECG signal. Note that there exists a number of different biosignals as described in Chapter 1: Literature Review.

Phase 2 – Transmit biosignal data: The collected data is read in from the sensors to an on-body PCB placed on the person. This data is then sent to an off-body computer for processing.

Phase 3 – Process the data: The data sent to the off-body computer is processed through the software which needs to be designed. This software will filter the information within the data to output appropriate music accordingly.

Phase 4 – Play the music: Based on the information given to the computer after being processed, appropriate combinations of notes can be assigned, and music can be played.

As the project progressed my contributions began to settle on Phase 2. The collected data needed to communicate with a PCB, and it eventuated that the design of that PCB will be my primary focus. In the early stages of the project, in order to gain a better understanding of the tasks required each of the four phases were studied.

3 MUSIC GENERATION AND SOFTWARE

3.1 CREATING SOUNDS

As described in Chapter 1: Literature Review, music is defined as "Vocal or instrumental sounds (or both) combined in such a way as to produce beauty of form, harmony, and expression of emotion" (Oxford, 2019).

For this project there were two main components of music composition. The initial idea behind creating sounds was the analysis of an ECG signal and the second part involved taking these sounds and putting them into a musical context.

Studying the dataset found from an online source, this data could be read into MATLAB and by simply using these values in conjunction with the inbuilt MATLAB functions, simple sounds can be created (Physionet, 2005).

The construction of the code began by reading in the data values and comparing them with a threshold value. This was an arbitrary value based primarily on the R-peak value (refer to Figure 1) within the dataset. Above that threshold a certain pitch can be outputted and below that threshold a different pitch can be outputted. This was a simple idea to create a two-tone music player based on the ECG dataset. These sounds started out by being generic sounds based on the ECG data and the threshold set by the code (refer to Appendix). With the assistance of the audio sharing website Clyp, these raw sounds were recorded through a computer and are presented here.

3.2 GIVING MUSICAL MEANING TO THESE SOUNDS

Once generic sounds were successfully created it came time to add meaning to them. This was in the form of a simple ascending pattern. It was at this point that Associate Professor Kenneth Pope assisted in the further refinement of the code. Developing this two-tone concept into a more flexible design, he suggested subdividing the dataset into blocks, processing them individually and outputting sounds based on the individual blocks. This allowed for different sounds to be created. The result of this refinement can be found through the following link.

Another additional feature that was added to the code was the loop, i.e. rather than reading through the entire dataset and then terminating the program (which only takes several seconds), the dataset is then looped. Rather than thinking of the data like one finite dataset, the data can be thought of as an infinite loop of one dataset on repeat. The main benefit to this was particularly for research and development of the code as I strove to understand what the code was doing and the effect the dataset had on it.

Once these sounds were made, it was time to start applying these in a musical context. For increased readability this section will refer to notes on a piano keyboard, specifically Middle C as shown in the diagram below. Note also that the shortest distance between two keys is referred to as a "semitone" and one more step is called a "tone".

This image has been removed due to copyright restrictions. Available online from [https://acousticnotesblog.files.wordpress.com/2017/01/middleconkeyboard.png]

Figure 7: The piano keys for reference, note the position of Middle C (Chaffman, 2017)

Physically speaking Middle C has a frequency of 261.63Hz. Understanding that the frequency difference between Middle C and one semitone higher to C# (C sharp) has a difference of 15.55Hz, mathematical equations can be written for notes in a scale. Depending on what type of scale we want our music to sound like will depend on which combinations of notes will be used in the tune.

As an example, the major scale is known to have a separation of *tone, tone, semitone, tone, tone, tone, semitone* between notes and hence this can be mathematically represented in MATLAB in terms of their respective frequencies. Similar principles can be taken for other scales, such as the minor and pentatonic scales which are based on combinations of notes and hence different equations can be written for different sounds.

Understanding these key musical concepts will assist us in taking ordinary sounds that we have created and giving them meaning. Although the ECG data was used to create musical sounds, this part primarily focussed on the mathematical models that can be used to output different scales. With Kenneth Pope's assistance, MATLAB code was developed which monitors the location of the mouse cursor on the computer screen. As the mouse travels vertically along the screen, the volume of the sound changes (up direction for an increase in volume and down direction for a decrease in volume) while horizontal travel of the mouse along the screen changes pitch of the outputted sound. Considering this concept, the computer screen can be divided into a number of columns. In each column, a certain pitch is outputted, as the mouse passes the line into another column, the pitch changes. Therefore simply defining this change in pitch as the change in frequency according to one of the scales above (major, minor or pentatonic) these scales can be easily represented as the mouse changes columns.

Note the description of the above code of scale creation in MATLAB does not actually work in this way, but for simplicity and understanding it can be thought of like this (refer to Appendix for the full code for this section).

The final sounds can be found through the following links for the major, minor and pentatonic scales. These sounds presented here are merely a proof of concept and the timing between notes and the volumes can be ignored.

It should also be reiterated that I worked in conjunction with Associate Professor Kenneth Pope on this section in order to get the programs compiling. As such each line of code will not be described in exhaustive detail as this is not the key point of this chapter but rather the thinking involved in order to create sounds and the process taken to get to that stage.

3.3 MIDI Implementation

As mentioned in Chapter 1: Literature Review, the final stage of this project is to store the processed data as a MIDI stream for outputting music. Unfortunately due to time constraints this section could not be sufficiently studied. There does exist some code online designed by (Schutt, 2012) which delves into MIDI using MATLAB programming. I imported this code into MATLAB for initial research, but the tasks involved with MIDI became too great as my priorities changed with my research beginning to develop a key focal area.

It was at this stage of the project where my focus began to settle on the electronics side of the system and more specifically the design of the on-body PCB. To properly design a PCB more research needed to be conducted into its requirements. Many different configurations were designed and tested, and the results are shown below.

4 Electronics and PCB Design

4.1 Initial Electronics

At the commencement of my project I knew I needed to acquire an ECG signal somehow. Associate Professor Kenneth Pope assisted by giving experiment documents for designing the analog front-end of an ECG circuit. The documents explained how to filter the ECG signal in conjunction with an ECG signal generator. Learning how to use a breadboard proved crucial for effectively building this circuit. The figure below shows the ECG signal generator with proper filtering and amplification achieved by the analog-front end that I built.

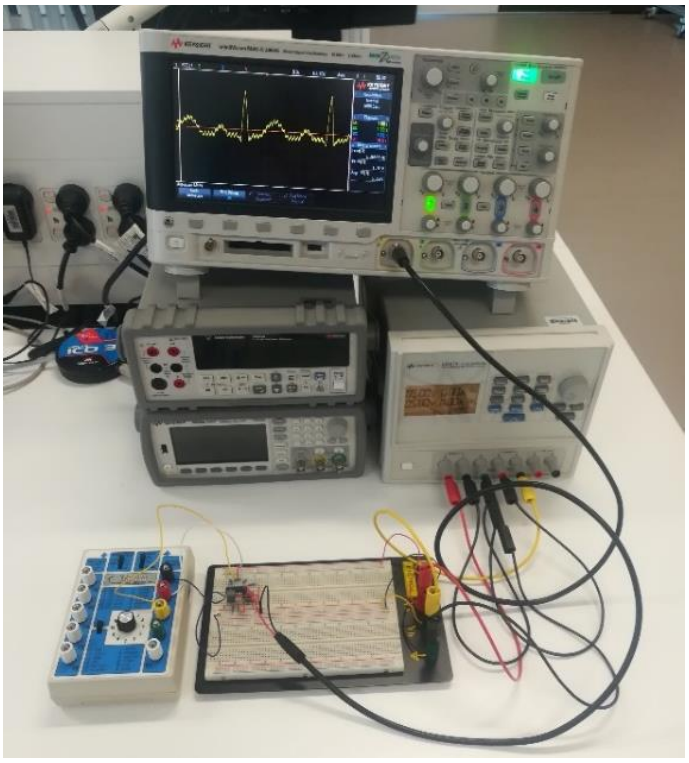


Figure 8: The waveform generator producing an ECG signal on the oscilloscope

The circuitry designed in Figure 8 is based off the circuit diagram below. Note the annotations at each operational amplifier I added as the circuit was pieced together. This assisted me in monitoring the design as it progressed. Unfortunately, I did not have time or proper safety precautions in place to connect myself to this circuit, however, the waveform generator was enough for the research required for this section.

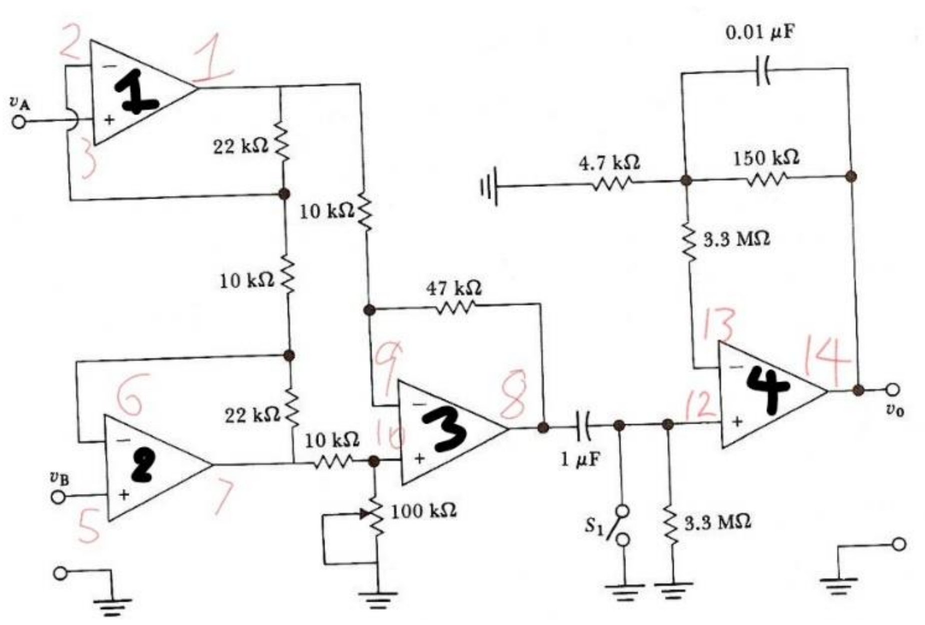


Figure 9: The circuit diagram used to design the ECG monitoring circuitry (Pope, 2019)

In my study of electronics I also had the opportunity to play with a cheap ECG sensor (the specifics of which are unknown) and the Flinbit development board provided by Flinders University as shown in the figure below. With this sensor it was a simple task to acquire an ECG signal. However, as demonstrated in the image, the sensor does not effectively filter the signal and so the result is an incredibly noisy signal.

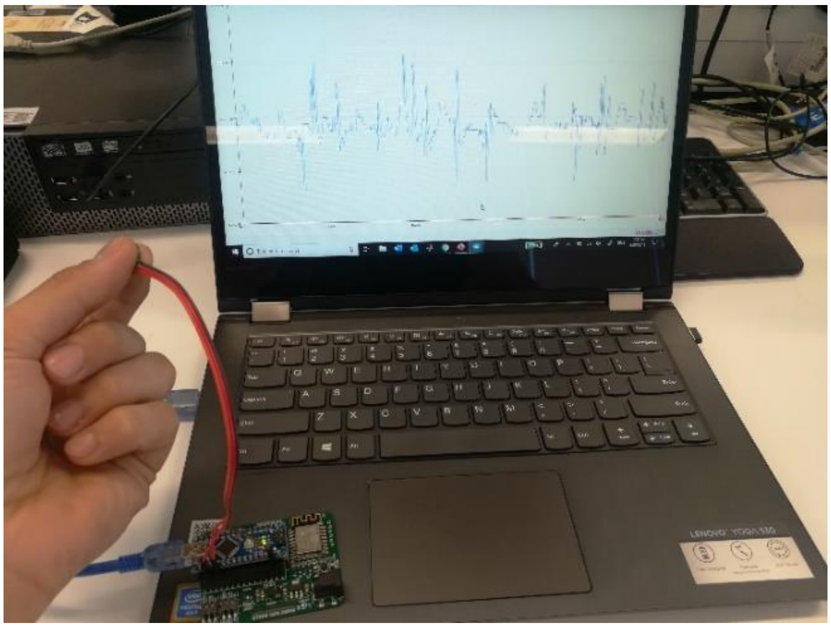


Figure 10: The Flinbit development board showing real-time ECG data

As mentioned in Chapter 1: Literature Review, I was informed that the MAX30003 chip exists which is capable of acquiring the signal and filtering it appropriately. The work I had done on the analog-front end was a good learning experience and helped develop a foundation of knowledge for my electronic applications. However, it was not essential in my final design due to the capabilities of the MAX30003 chip.

Through my research I found the ECG Breakout Board by Protocentral (refer to Figure 11) which utilises the functionality of the MAX30003 chip for wearable applications. The idea of this board is to obtain the wearers ECG signal and communicate with an external microprocessor through SPI; this information is then displayed on a Graphical User Interface (GUI). The online tutorial shows the board connected to an Arduino Uno kit but due to availability, the information will be simply conveyed through the Flinduino board designed by Flinders University. The ECG Breakout Board is approximately 3cm long and 2.5cm high which is incredibly small and gave a much greater appreciation for the size of PCB's used in wearable applications.

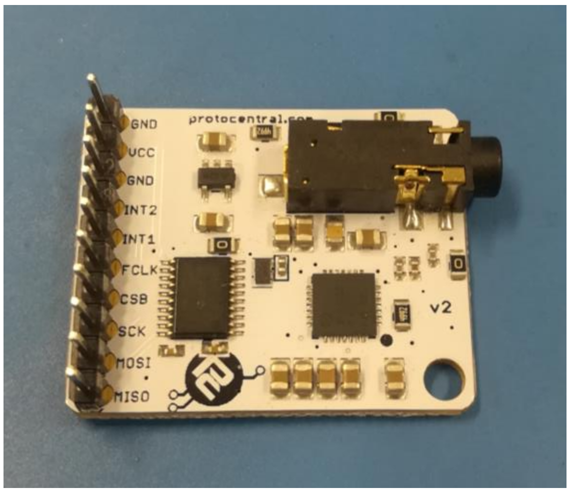


Figure 11: The ECG Breakout Board (protocentral, 2019)

Acquiring a signal from this board was no small feat as it required thorough research into the workings of SPI communication. Furthermore, the 10-Pin header came separate and needed to be soldered onto the board.

Developing a better understanding of SPI involves gaining knowledge of the different aspects of this communication method. There are four wires involved in SPI and they are as follows:

Table 5: SPI Commands and their Functions

Pin	**Description**
SCLK	The clock, sent from the master to control timing
MOSI	When data is sent from the master to the slave it is sent down the Master Out – Slave In line
MISO	When data is sent from the slave to the master it is sent down the Master In – Slave Out line
SS	SS stands for Slave Select and is used particularly when there are multiple slaves connected to a single master

Connecting the Breakout Board to the Flinduino gave me a much greater understanding of electronics and programming in a more practical application. In order to program onto the Flinduino, the Flinduino chip set had to be downloaded from the Flinduino wiki page.

Fortunately the board came with code which had already been written. Some changes did need to be made to the code to support the Flinduino in the Arduino programming environment. The main change was renaming some of the pins. For example, the slave select pin on the Arduino Uno is pin 6 whereas on the Flinduino it is pin number 11.

As previously mentioned, connecting this device gave me a deeper understanding into electronics in a more practical situation. I had plugged the wires in and could not successfully obtain my ECG signal. I took it to Engineering Services at Flinders University and it was not until the schematic was studied that it was realised the power was coming from a 5V supply rather than the 3.3V supply that the Flinduino operates at. Fortunately this error caused no damage to the components and simply swapping the wire allowed the successful reading of ECG data from my body.

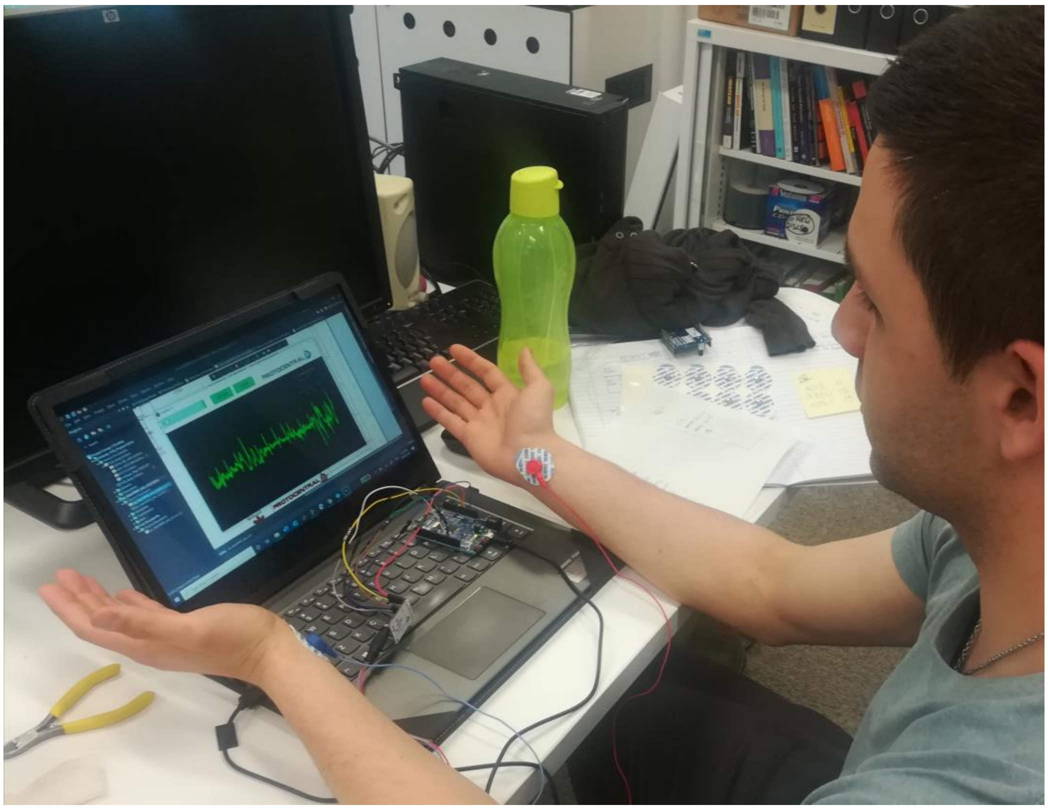


Figure 12: I connected myself to the circuitry and successfully obtained an ECG signal

Note that there was some confusion regarding SPI connections, so it is worthwhile reiterating that MISO from the master connects to MISO from the slave. It is also important to note that for my safety, my laptop had been disconnected from the mains power and was operating on its battery alone.

Now that I had a clearer understanding of the electronics involved in attaining a biosignal and more importantly, the direction of which I wanted my electronics to go, I could now begin to develop the on-body PCB.

4.2 My PCB Design

4.2.1 My Initial Block Diagram

I had never designed a PCB on my own and initially it was a daunting task. Much research was done to determine whether or not a board already exists which performs similar tasks as a starting point. Unfortunately, there was nothing in this field that was particularly useful for this application.

Everything involved in the PCB design process was a learning experience. I began with the initial block diagram physically written down on paper showing each component and how they communicate with each other (refer to Figure 13).

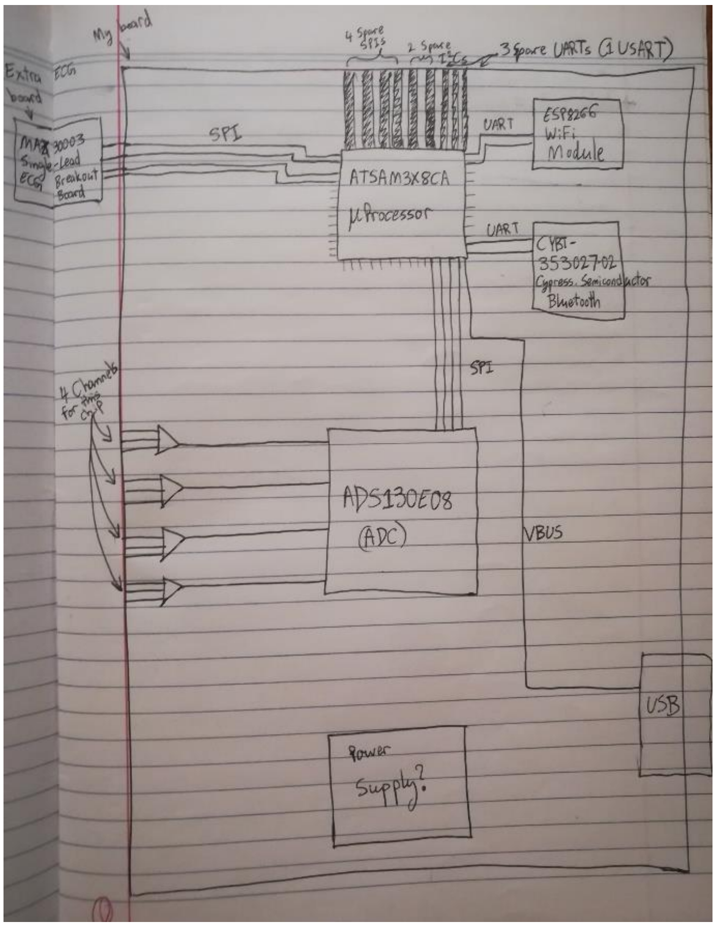


Figure 13: My initial block diagram

After meeting with the expert as explained in Chapter 1: Literature Review, a new block diagram was designed considering the new components as shown in the following figure.

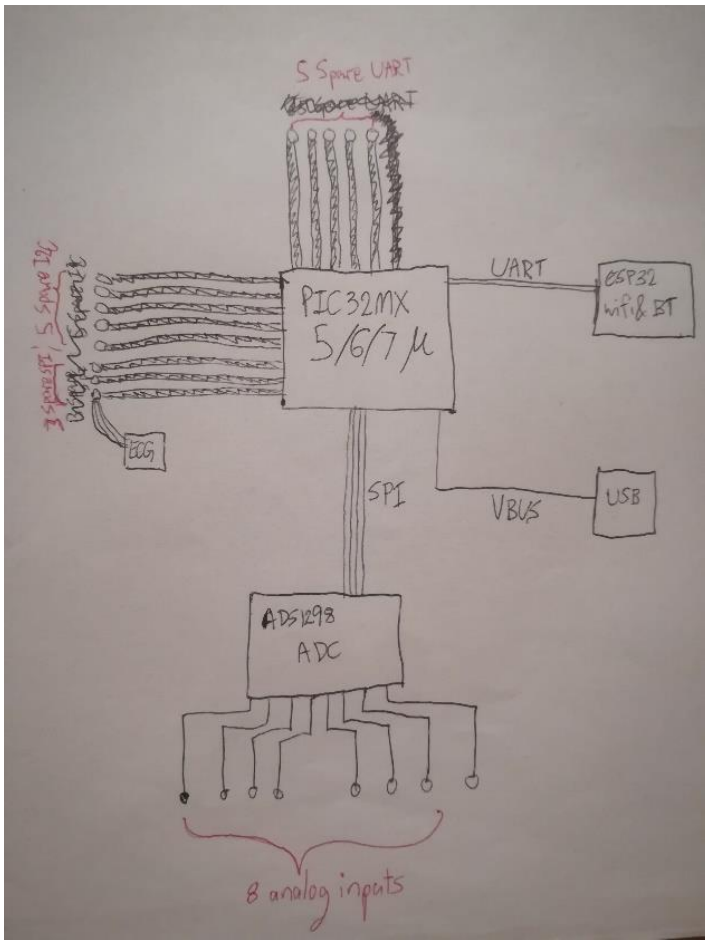


Figure 14: My second block diagram

4.2.2 Using Altium Designer

I then had to transfer this drawing onto Altium Designer. There was an incredibly steep learning curve from the very beginning. The assistance of engineers Lucas Moss, Lucas Paix and an online series of tutorials designing a PCB in Altium gave me a strong base on which I could begin my work with Altium Designer.

I began by implementing the large components such as the microprocessor, the ESP32 (Bluetooth and WIFI) module and Analog to Digital Converter. Fortunately, rather than needing to design the schematic for each of these parts I was able to find their schematics through online sources. Having these schematics readily available online proved to be an effective time saving measure as I did not need to manually draw them on Altium. The trade-off to this, however, was that I had to cross reference each schematic with their respective datasheets to ensure that each part was correct, and each pin was present and labelled correctly. Finding the component through the online supplier Digi-Key and copying the part number into Altium was a skill that was learnt through research. This part is imported into Altium which results in a blank schematic part being created. The schematic found online can be copied and pasted into this imported Digi-Key part (Feranec, 2018).

Simultaneously while I was doing this, I was also searching for 3D models and footprints for the PCB parts. My intention is not to reinvent the wheel and if a free to use footprint and 3D model is available for a component then it is a clever time-saving method. These footprint files can be easily opened in Altium through the use of the “Footprint Wizard” tool. Following the prompts in the tool, the model can be imported into Altium Designer and then added into the project.

It is worth noting at this stage that for my design I employed a USB of type micro. The reason why this was done is because it is compatible with the hardware that is available at Flinders University. The standard for USBs appears to be moving towards the Type C USB but for simplicity I decided to maintain the USB micro and further refinements can be made on this design in the future (refer to Figure 15 for a comparison of the two USB standards).

This image has been removed due to copyright restrictions. Available online from [https://asset.conrad.com/media10/isa/160267/c1/-/sv/1341725_LB_00_FB/image.jpg]

Figure 15: Type C USB (left) and Micro USB (right) (Banks, 2017)

Although the footprints are collected and designed at the same time that the schematic parts are designed, these parts are not yet ready to be placed onto the PCB. Firstly, the schematic parts must be wired together in a schematic document. For my particular design I opted for a multisheet design as it could allow for a tidier top-level schematic (refer to Figure 16). Each component piece was quite large with a vast number of pins so having them all on one page did not seem like a sensible solution.

This portion of the PCB design process assisted me in understanding how to read datasheets. Each component that was selected has an associated datasheet which clearly outlines the function of each pin as well as recommended configurations. Often in the datasheet there will also be a “Typical Application Schematic” section which outlines the additional electronics required to make the component to work. Parts like decoupling capacitors which are used to exclude noise will usually be found in this part of the datasheet. Once it is clear which extra circuitry will be required, the process of including these extra capacitors and resistors etc. is exactly the same as for the bigger components. This involves simply finding them in Digi-Key and including their schematic and footprint to go towards the final PCB design.

An error was encountered where the footprint had not been correctly assigned to the schematic component for one of the resistors in the design. There is a tool in Altium called “Footprint Manager” which, rather than looking at components individually, it opens a list of every single component and their corresponding footprint. When this list was opened, it was found that there was indeed a problem with the way in which I had initially assigned the footprint to the schematic part; where a thumbnail of the footprint should have been, instead there was a black square. This was a simple fix and all that needed to be done was to delete the current footprint and assign the footprint again in the Footprint Manager. It is unsure what caused this issue but fortunately the solution was straightforward.

Having a multisheet design makes the schematic more readable, however, research needed to be done in order to know how each sheet pieced together and it was found that Altium has tools to achieve this. Utilising the “Port” tool, creating and naming ports across two different pages is a way that connects each sheet and Altium automatically handles the resultant hierarchical structure of sheets. It is important to note that in order to properly connect these ports, on the master sheet, the function “Sheet symbols created from sheets” must be used and Altium automatically creates the ports on the sheet symbols. It is also important to understand that ports which are required to connect to each other must be called the same name otherwise

Altium will display a warning message. The following five figures are the schematics created for each major component, including the master sheet.

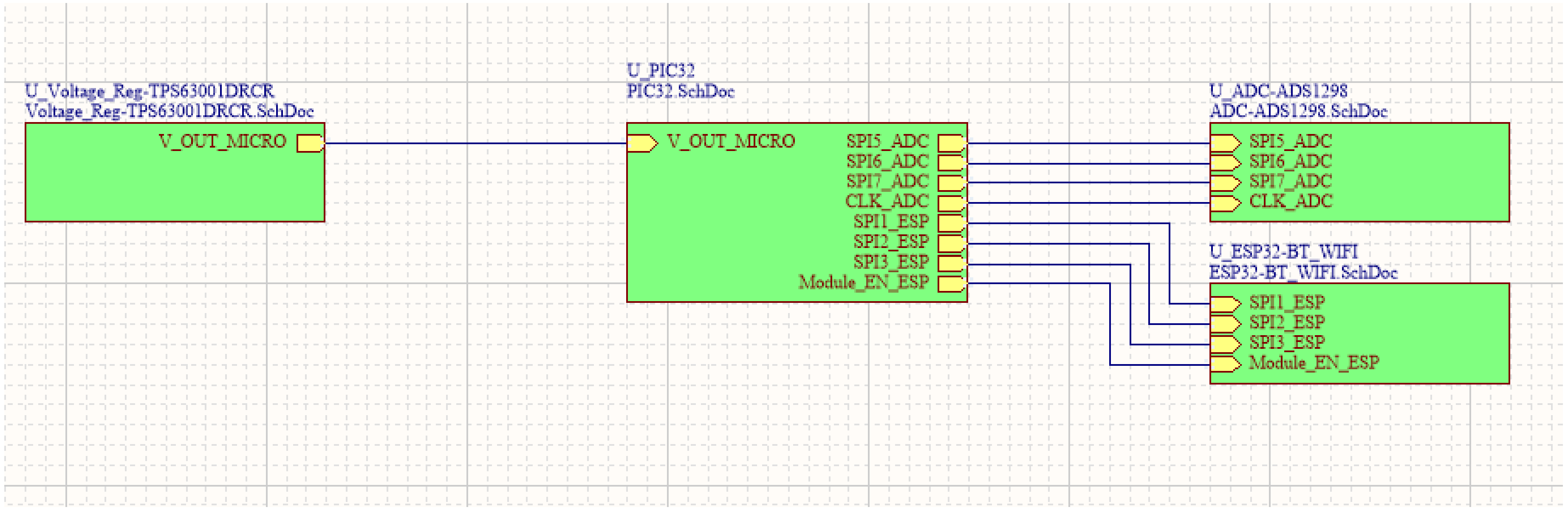


Figure 16: The master sheet of the schematic. Note the ports used to connect the sheets

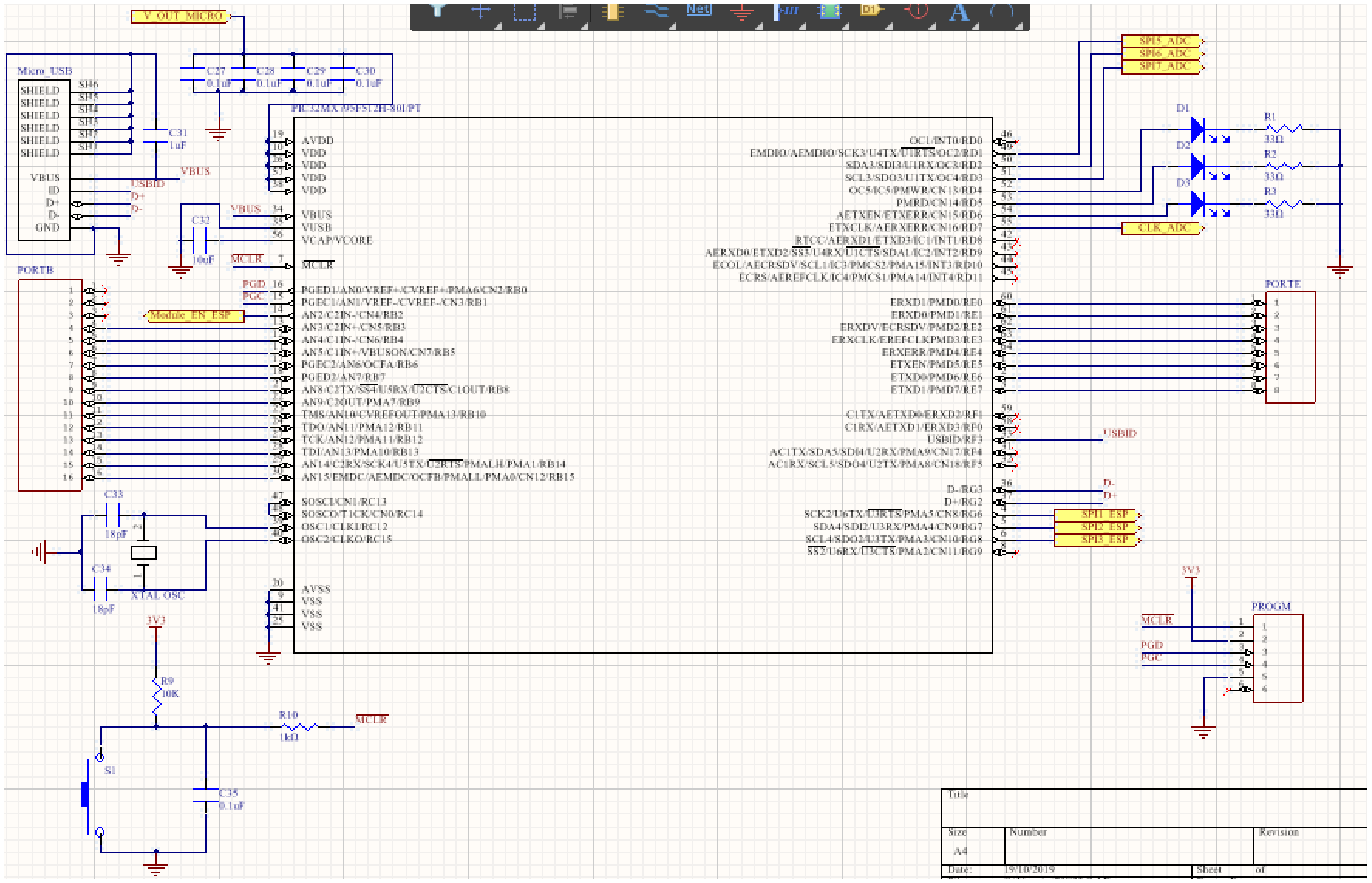


Figure 17: The schematic configuration for the PIC32MX microprocessor

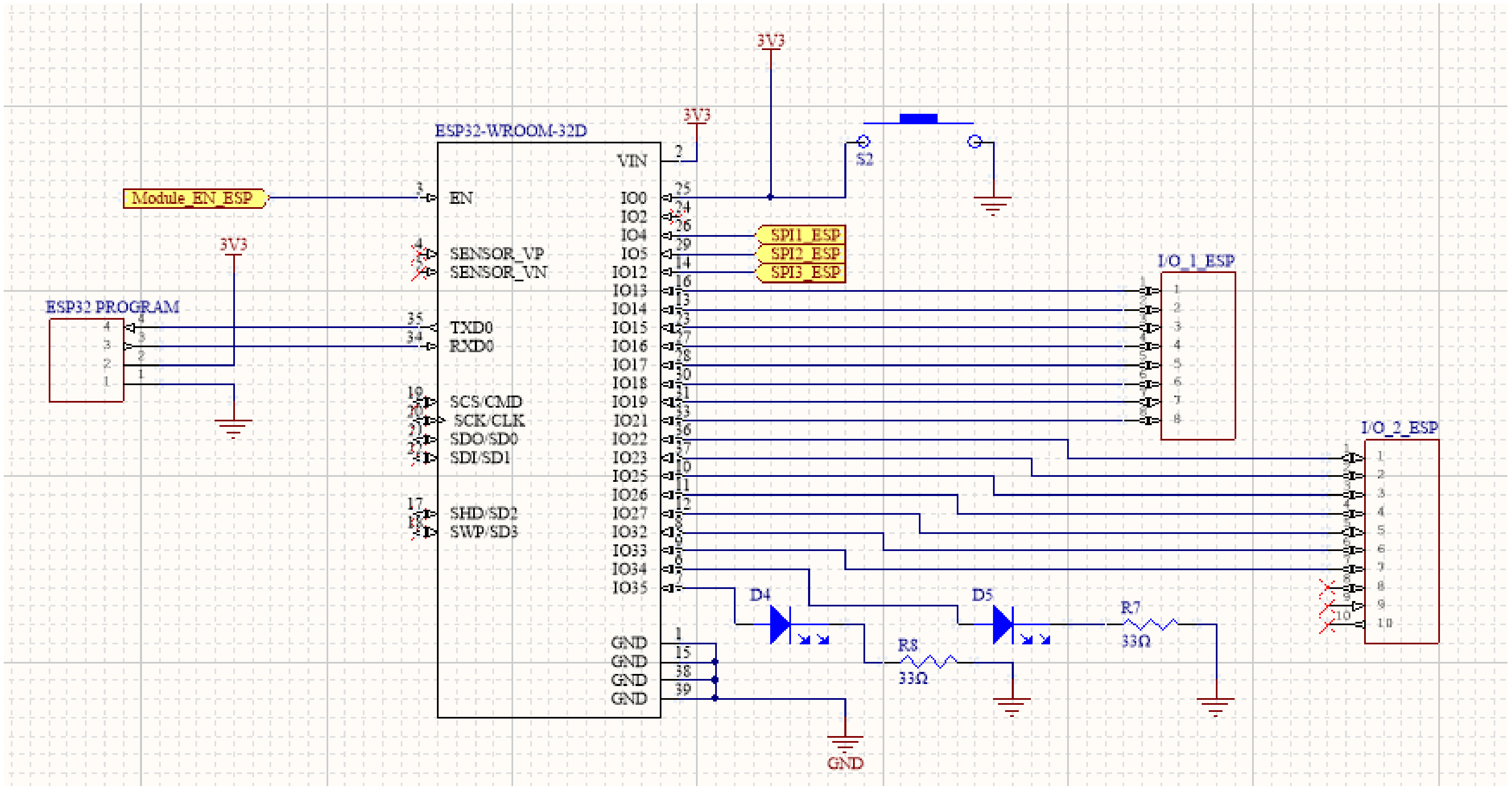


Figure 18: The ESP32 schematic

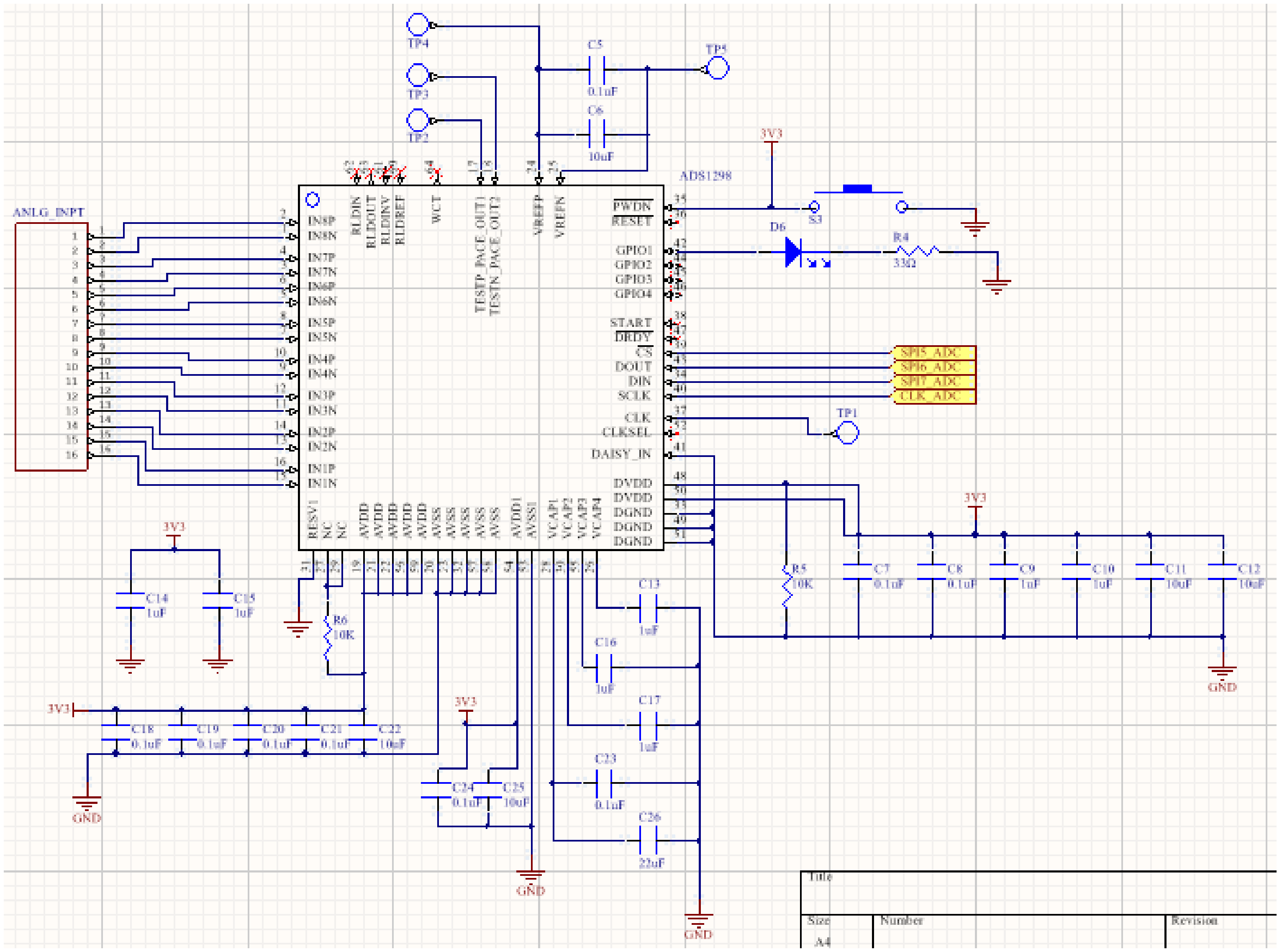


Figure 19: The schematic for the Analog to Digital Converter

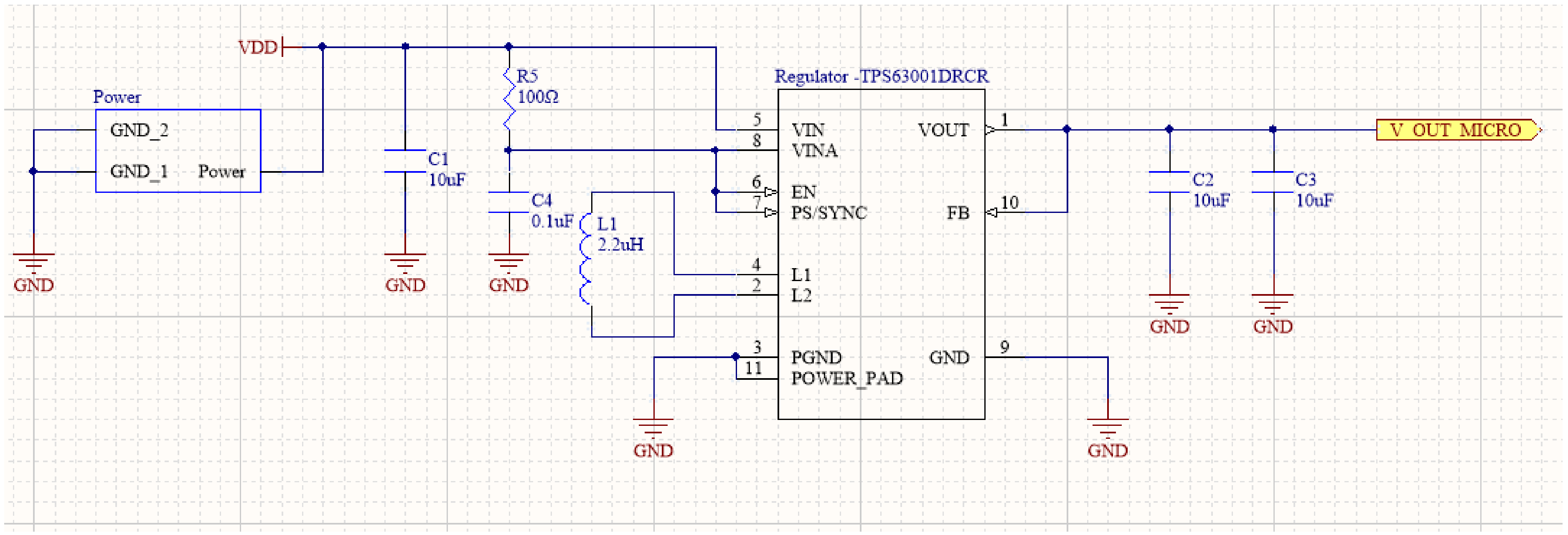


Figure 20: The schematic for the voltage regulator component

The schematic of the analog to digital converter (Figure 19) is particularly interesting because it was the first schematic I created when this process began. Once Engineering Services had reviewed it, they informed me that there was a multitude of flaws that needed to be reconsidered, particularly with some of the configurations of the pins. After gaining more experience with the other schematics, I was able to return to the ADC schematic and remedy the errors with greater ease than I had initially. This was a particularly good experience as it demonstrated an obvious growth in my own knowledge.

Another interesting point about this schematic is the pins which are not connected. The pins which are not connected are those that are responsible for the biopotential measurements. Due to time constraints these extra features for the analog to digital converter could not be taken advantage of but are still available for future students to implement.

At the end of this section, once each schematic part has been connected, the project can be compiled to determine whether or not any warnings/errors have occurred.

I encountered multiple warnings but there were two main ones. The first one was the connectivity of certain pins, for example an I/O port connected to a Power pin. As these schematics were obtained from online sources, I did my best to check and confirm each pin was correct, however, some were overlooked. I fixed these warnings by going through the datasheet of the components and identifying the function of each pin and made the necessary changes to the schematics.

The other, more difficult issue I was getting was in relation to the multisheet design. As previously mentioned, connecting pages in a multisheet design is done using ports. I was getting a warning message which explained that there were contradictions with the net names of the wires that were being used to connect ports across sheets. Initially I did not understand the nature of these warnings, but through research and patience I was able to identify that in order to properly connect ports across sheets they must have the same name. This may seem like a simple fix but pinpointing what exactly was causing the warning proved to be quite a difficult task.

4.2.3 Finishing my schematic and starting the PCB Layout

In essence the entire PCB design process began by me conducting my own research into each section and teaching myself the majority of the work. Once I had connected each component and completed the schematic to a finished state, I submitted it through to Engineering Services at Flinders University for review. Engineering Services are a team of engineers who assist students with their projects. The benefit of completing my PCB through Engineering Services is that it gives an opportunity to gain a deeper understanding into specific components and configurations.

The first time I had sent my schematic through, Engineering Services had informed me that decoupling capacitors had been forgotten as well as a reset circuit. They also suggested that I study the schematic for the Flinduino board as it uses the same PIC32MX microprocessor that is in my design.

Once these revisions were made, my schematic document was again sent to Engineering Services for another review. During this time I began to piece together the footprints of each component and begin my PCB layout.

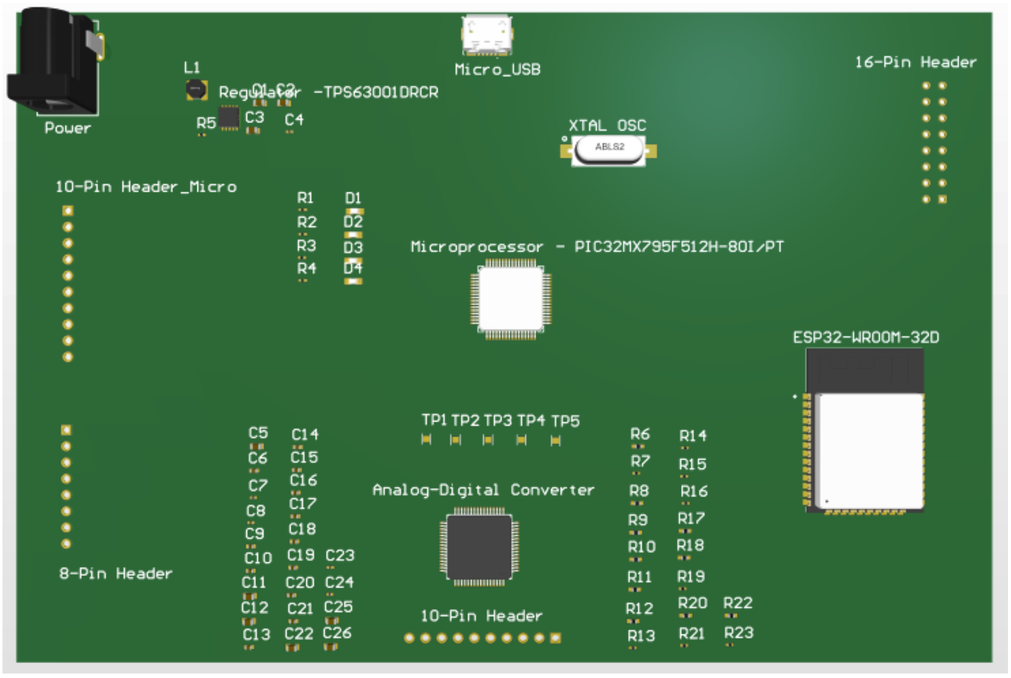


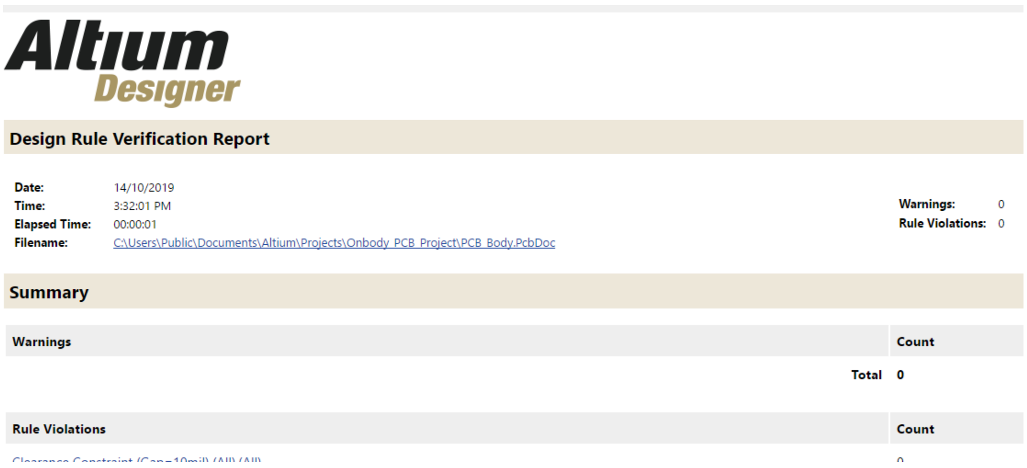
Figure 21: An early revision of my PCB design

I simply started by importing the components into the PCB document to piece together the design shown in the figure above. The next step of the process is to route the design. By routing the design, wired links are made between the connections imported from the schematic document.

This is a process which is typically done manually to optimise the board but Altium has an "auto-route" function which connects these pads automatically. This has the added benefit of decreasing the time spent on routing, but the cost is that the resultant design is of poorer quality. The auto-route function creates connections where it thinks the best paths are, however, it does not consider the reality of the connections it makes like a person would. Just to get a routed design done, I used auto-route to see how my design managed at this stage. Not all of my connections could be made, and the auto-route terminated. To see where the errors were occurring, I cleared the routes that had been made and I had to run a check of my PCB.

In order to check the PCB, Altium has a tool called "Design Rule Check" which essentially checks the PCB against a set of criteria to ensure its design before it gets physically printed. It was at this point that I ran the check to see how my design fared at this stage of development.

By default the Design Rule Check terminates once it reaches 500 errors and my design easily reached this 500-error point. An example of this check is shown below where the types of rule violations are listed and can be simply pinpointed.



Altium Designer

Design Rule Verification Report

Date:	14/10/2019		
Time:	3:32:01 PM	Warnings:	0
Elapsed Time:	00:00:01	Rule Violations:	0
Filename:	C:\Users\Public\Documents\Altium\Projects\Onbody PCB Project\PCB Body.PcbDoc		

Summary

Warnings	Count
Total	0

Rule Violations	Count
Clearance Constraint (Gap=10mil) (All),(All)	0

Figure 22: The Altium Designer Design Rule Check

It is worth reiterating that each part of the PCB design process was new to me and repairing over 500 errors seemed like an incredible task. I began simply with the first error and I found that some of the nets were not properly imported into the PCB layout. The nets are the properties that each wire has and is determined by what the wire is connected to. This error of importing the nets is also the reason why the auto-route failed.

After much research it was found that the reason these nets were not importing correctly is due to the fact that the component's pin numbers in its schematic was different to the corresponding pad number in the footprint.

As an example, the figure below shows the footprint of the $10\mu F$ capacitor with pads labelled "1" and "2".

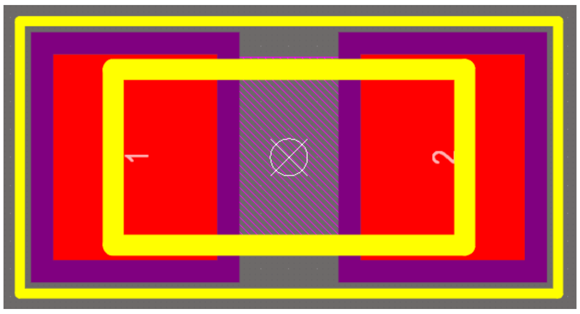


Figure 23: Footprint of the $10\mu F$ capacitor

The corresponding schematic part is shown below which clearly shows pins 1 and 2. When these schematic components were created, the default of these pin values can change and at the time it was not known that this would cause an error later in the design process.

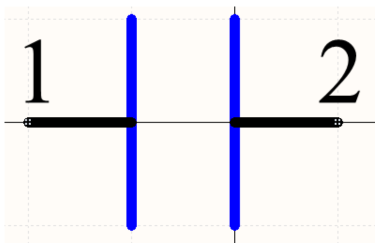


Figure 24: Schematic of the 10µF capacitor

Simply changing the labels of the pins in the schematic to match the pad numbers in the footprints, this caused a number of nets to be successfully imported into the PCB layout and hence assisted in clearing some of the errors.

The largest source of errors that the Design Rule check was presenting to me was the "Unrouted Net Constraint" Error, which was giving me over 200 sources of error. Again, this seemed like a large task to repair all of these errors. Through further research it was found that this error is related to the connections between pads in the PCB design layout. As I had only just placed the components onto the board without routing them, none of the wires had been connected in the PCB document. At this stage, having successfully imported all of the nets into the PCB document, I tried the autoroute again which produced the following PCB design. Note some changes had been made to the schematics to include additional headers.

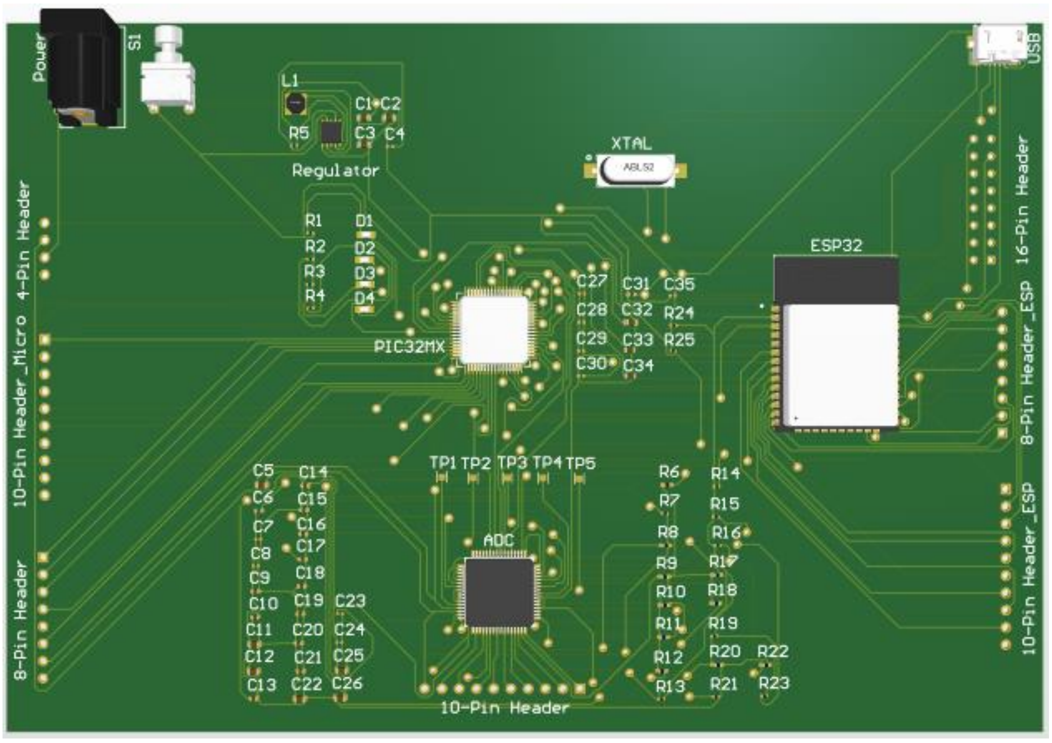


Figure 25: Updated PCB design with routing done by autoroute

After making all the refinements of my design, everything had been appropriately connected in the PCB document. The little holes scattered across the board are called "vias" and are used to swap the connection between the top and bottom layers of the PCB. Switching between layers in this way is a simple way to avoid unwanted connections between components.

The next sources of error that I was getting were the "Minimum Solder Mask Sliver", "Silk to Solder Mask" and "Clearance Constraint". These errors are related to the distances between parts. For example there is a minimum allowable distance between routing to vias, labels to components and also distance between pads on a component.

To accommodate the small sizes of my components these design rules needed to be modified. By decreasing the minimum allowable distances of the "Minimum Solder Mask Sliver" and "Silk to Solder Mask" from 10mil to 1mil many of the errors can be remedied. The "Clearance Constraint" design rule could also be cleared by unselecting the tick-box located in the design rule modification window. Although the methods outlined above seem to mask the issues rather than explicitly repair each design breach, the design rules needed to be modified to suit the size of the components. For example, the 64 pins on the microprocessor are particularly small and modifying the design rules to accommodate this was crucial.

Note that the unit “mil” is an Imperial unit commonly used in electronics. It is the name given to one-thousandth of an inch and has a metric equivalence of 0.0254mm.

Once my PCB layout was completed, I had received the second review of my schematic from Engineering Services. The first improvement to be suggested was that the UART pins from the ESP32 (Bluetooth and WIFI) were one of the methods used by the chip to communicate and hence should be brought into a header.

The next point that they had made was, again, to do with the communication pins of the microprocessor and the ESP32 module. The schematic documents had been misread and the programming clock (PGC) and data lines (PGD) on the microprocessor had not been included. The ESP32 was communicating with the microprocessor through SPI communication, but were not connected to the correct pins. The errors from above were vital repairs as they are all methods used for communication and without proper implementation the microprocessor and the ESP32 could not communicate with each other and other components on the board.

Making these adjustments and importing the new schematic design allowed for an updated PCB to be laid out. The layout was also improved significantly at this stage with the inclusion of 3D models for each of the header components. A website was found called 3D Content Central which is an online database full of three-dimensional models. The inclusion of these 3D models helped to produce the image shown in Figure 25.

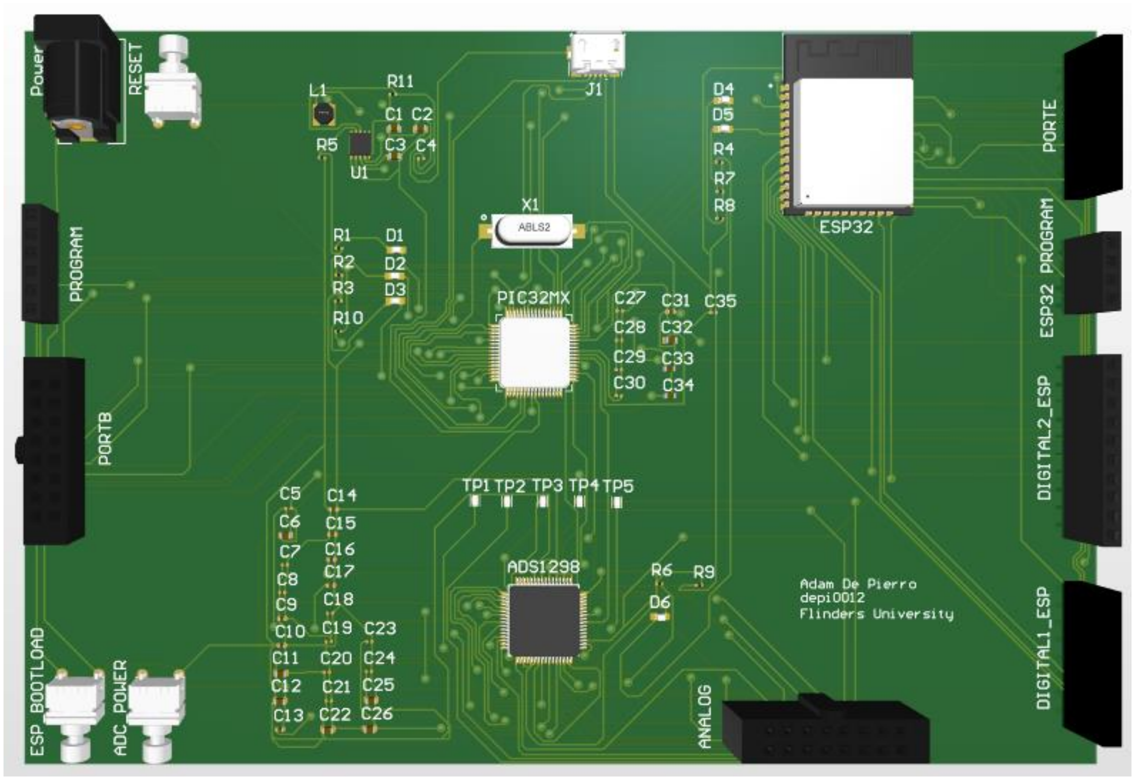


Figure 26: The final rendition of my PCB design

The above design has also been improved further in a number of other ways. The first improvement technique is to tent the vias. This essentially means that the vias will be covered with a layer of soldermask to enclose the holes in the board (Dillman, 2018). This improves safety of the vias as they are now protected, as well as the overall appearance of the board. Another simple, yet effective improvement technique was by improving the labels of the components. Appropriately naming each component and laying the names out properly is a key concept in good PCB design. It should also be noted that the ESP32 module has been shifted to the top of the board. This was done intentionally as the top of the module is an antenna and requires some separation from the other components on the board.

At Flinders University before a PCB design can be printed, Engineering Services reviews it to ensure that the design has the best chance of working once it is manufactured. This process is very much a repeat-loop method in the sense that the PCB can be reviewed multiple times with many iterations being cycled through before it is deemed ready. Unfortunately this is not a step which was allocated the time it deserved and due to time constraints my final PCB was unable to be manufactured. It was also found that there was insufficient time to properly layout the capacitors, in particular the decoupling capacitors, with respect to their corresponding chips, hence the capacitor layout in Figure 26.

4.2.4 Techniques employed to guide me through Altium Designer

There were a number of different techniques I employed in order to help me navigate Altium. Using a pen and paper to physically write down key aspects of my design really helped me understand what needed to be done. This was particularly useful when debugging the errors and warnings after compiling the project.

Another one of the main methods which really helped solidify my understanding of the work I was doing was writing down the steps to access different tools within the environment. The table below shows some of these tools just as a brief example of the thought processes involved as I underwent this portion of the project.

Table 6: A brief list of the tools I commonly used in Altium

Function	Flow
Annotating schematics	$Tools \rightarrow Annotation \rightarrow Annotate\ Schematics \rightarrow$ $Update\ Changes\ List \rightarrow OK \rightarrow$ $Accept\ Changes\ (Create\ ECO) \rightarrow Execute\ Changes$
Accessing footprint manager	$Tools \rightarrow Footprint\ Manager$
Adjusting origin position in footprint	$Edit \rightarrow Set\ reference \rightarrow Center$
Delete entire track in PCB design	$Select\ one\ segment \rightarrow Tab \rightarrow Delete$

This is not a complete list of the functions and tools that were used in Altium but explicitly writing down steps in this fashion really helped to learn the primary functions of the program without researching every time the location of a tool was forgotten.

Another technique I employed to understand the work I was doing in Altium, was storing each PCB component in a separate spreadsheet in Microsoft Excel. I stored the name of the component, the quantity and a URL address to the component's purchase location. I also compiled a folder of the 3D models of each part as they were used. Keeping lists like these helped me monitor the parts used in my design so I could keep track of what I had done and what I had yet to do for my PCB.

5 Discussion and Conclusion

There are many different aspects to this project in its entirety and is too much for one person. This project has been worked on by previous students of Flinders University, but no one has worked on the on-body PCB where my research has been primarily focussed.

Taking the initiative of dividing the project into four different phases assisted in overall understanding of the tasks which needed to be achieved, who was responsible for achieving them and how. Separating the project in this manner was the primary method used to identify my key areas of research and my role in the project long-term. Before coming to this conclusion of my place within the project, I did research into each of the phases to broaden my knowledge of the tasks that needed to be done.

Beginning the project by studying ECG data and using it to create music assisted me in the way I logically thought about code and the way to structure it. Converting the musical scales of major, minor and pentatonic from piano scales to MATLAB code taught me how to look for relationships between real world situations and the ability to model them using mathematics.

Designing the ECG circuitry using nothing but a circuit diagram (refer to Figure 9) taught me how to read circuit diagrams and make sense of them in terms of real-world electronics. Labelling the circuit diagram as I went was a useful tool to help guide my research while piecing together in this design.

There were many tasks which came with extremely steep learning curves. One of the initial roadblocks which actually seemed quite simple initially was using the ECG Breakout Board with the MAX30003 chip. It was known that this communicates with an external microprocessor through an SPI connection. Although there are numerous sources which assist beginners with communicating using SPI, translating these solutions to my setup using the Flinduino proved to be quite a difficult task.

I had connected the Breakout Board with a number of different wires, downloaded the GUI in order to see my ECG signal but nothing happened. With the assistance of Engineering Services, we were able to conclude that the incorrect voltage was being applied to the board. Swapping the wires around allowed the board to work properly and I was successfully able to monitor my ECG signal in real time.

The main factor which haltered any major progress on the project was learning, understanding, and implementing my design in Altium Designer. At the commencement of the project there was an overwhelming number of subtasks involved which I simply did not know how to do. I attended a multitude of meetings with Altium users to help guide the first few steps of the process. In addition, I found an extremely helpful series of online tutorials which at this stage are out of date but were similar enough for me to understand the starting point for each of the steps involved. To simplify the required tasks, I explicitly divided the Altium Designer process into four steps which are outlined below.

Step 1 - Build Schematic Parts

Step 2 - Design footprints for each schematic part

Step 3 - Route Schematic

Step 4 - Layout PCB design

The online tutorial really cemented the process for me and allowed me to not only know where to start, but how the design starts, progresses and where it ends. Furthermore, the assistance I received from Engineering Services at Flinders University helped broaden my knowledge of the PCB design process and gave me a much greater understanding of electronics at a deeper level. Unfortunately, the PCB design process was grossly underestimated, especially for a beginner, and as a result there was insufficient time to manufacture the board. Regardless, debugging the PCB gave the opportunity to practice a useful skill and was an incredibly worthwhile learning experience.

The research and work conducted through my contributions to the project provide a solid foundation for following students to continue where I leave it. There are opportunities to refine my design based on what I have achieved using Altium Designer or start something completely new using the research that I have done. This project was incredibly difficult at times but taught me different concepts in coding, electronics and overall thinking. This learning process really helped me overcome barriers and gave me the knowledge and confidence to implement future PCB designs.

6 REFERENCES

AFONSO, V. X., TOMPKINS, W. J., NGUYEN, T. Q. & LUO, S. J. I. T. O. B. E. 1999. ECG beat detection using filter banks. 46**,** 192-202.

ALFAOURI, M. & DAQROUQ, K. J. A. J. O. A. S. 2008. ECG signal denoising by wavelet transform thresholding. 5**,** 276-281.

ARI, S., DAS, M. K. & CHACKO, A. 2013. ECG signal enhancement using S-Transform. *Computers in Biology and Medicine,* 43**,** 649.

ASSOCIATION, M. 2019. *Summary of MIDI Messages* [Online]. Available: https://www.midi.org/specifications-old/item/table-1-summary-of-midi-message [Accessed 28/7 2019].

BANKS, L. 2017. The Difference Between Micro USB and USB type C. Complex Central.

BENITEZ, D., GAYDECKI, P., ZAIDI, A., FITZPATRICK, A. J. C. I. B. & MEDICINE 2001. The use of the Hilbert transform in ECG signal analysis. 31**,** 399-406.

BERRY, R. B. R., SCOTT; GIRDHAR, ANKUR; WAGNER, MARY H. 2016. *Use of Chest Wall Electromyography to Detect Respiratory Effort during Polysomnography* [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4990946/ [Accessed 28/7 2019].

CENTRAL, D. C. 2019. *FREE 3D and 2D CAD Models of User-Contributed and Supplier-Certified Parts, Assemblies and more* [Online]. Available: https://www.3dcontentcentral.com/ [Accessed July 30 2019].

CHAFFMAN, T. 2017. What is middle C in piano and keyboard playing? Quora.

CHOUHAN, V. & MEHTA, S. S. Total removal of baseline drift from ECG signal. 2007 International Conference on Computing: Theory and Applications (ICCTA'07), 2007. IEEE, 512-515.

CLYP 2019. THE BEST PLATFORM FOR CREATORS TO SHARE AUDIO.

CRAMER, F. 2019. Review of my PCB. *In:* DE PIERRO, A. P., KENNETH (ed.).

DEMSKI, A. J. & LLAMEDO SORIA, M. 2016. ecg-kit: a Matlab Toolbox for Cardiovascular Signal Processing.(Software metapaper)(Report). 4.

DIGI-KEY. 2019. *World's Largest Selection of Electronic Components Available for Immediate Shipment!* [Online]. Available: https://www.digikey.com.au/ [Accessed July 30 2019].

DILLMAN, P. 2018. *Via Tenting for PCB Design* [Online]. MacroFab. Available: https://macrofab.com/blog/via-tenting-for-pcb-design/ [Accessed October 19 2019].

DUOLINGO 2019. Body.

EEROLA, T. & TOIVIAINEN, P. 2004. MIDI toolbox: MATLAB tools for music research.

ELECTRONICS, P. 2019. MAX30003 Single-lead ECG Breakout Board - v2.

FERANEC, R. 2018. *Tutorial for Altium Beginners* [Online]. YouTube: Fedevel Academy. Available: https://www.youtube.com/channel/UCJQkHVpk3A8bgDmPlJlOJOA [Accessed July 30 2019].

GUPTA, R., BERA, J. & MITRA, M. J. M. 2010. Development of an embedded system and MATLAB-based GUI for online acquisition and analysis of ECG signal. 43**,** 1119-1126.

ILLANES, A. Z., QINGHUA; MEDIGUE, CLAIRE; PAPELIER, YVES 2006. Multi-lead T wave end detection based on statistical hypothesis testing. *Symposium on Modellin and Control in Biomedical Systems*.

INDIAMART 2019. Lenovo Desktop Computer.

INTEGRATED, M. 2019. MAXREFDES100#: HEALTH SENSOR PLATFORM.

INTEGRATED, M. 2019. *Ultra-Low Power, Single-Channel Integrated Biopotential (ECG, R to R Detection) AFE* [Online]. Available: https://www.maximintegrated.com/en/products/analog/data-converters/analog-front-end-ics/MAX30003.html [Accessed].

ISLAM, M., TANGIM, G., AHAMMAD, T., KHONDOKAR, M. J. I. J. O. C. & ENGINEERING, E. 2012. Study and analysis of ecg signal using matlab &labview as effective tools. 4**,** 404.

JOSHI, S. L., VATTI, R. A. & TORNEKAR, R. V. A survey on ECG signal denoising techniques. 2013 International Conference on Communication Systems and Network Technologies, 2013. IEEE, 60-64.

KOHLER, B.-U., HENNIG, C., ORGLMEISTER, R. J. I. E. I. M. & MAGAZINE, B. 2002. The principles of software QRS detection. 21**,** 42-57.

KUMAR, N., AHMAD, I. & RAI, P. 2012. Signal processing of ECG using Matlab.

LISLE, R. J., MCDONALD, B. S. & WILKES, M. D. 1991. Method and apparatus for simultaneous output of digital audio and midi synthesized music. Google Patents.

MATHWORKS. 2019. *Documentation* [Online]. Available: https://au.mathworks.com/help/matlab/ref/audioplayer.html [Accessed 28/7 2019].

MCSHARRY, P. E., CLIFFORD, G. D., TARASSENKO, L. & SMITH, L. A. J. I. T. O. B. E. 2003. A dynamical model for generating synthetic electrocardiogram signals. 50**,** 289-294.

NAIT-ALI, A. 2009. *Advanced Biosignal Processing*, Springer.

OJHA, D. K., SUBASHINI, M. J. W. A. O. S., ENGINEERING, TECHNOLOGY, I. J. O. M., HEALTH, PHARMACEUTICAL & ENGINEERING, B. 2014. Analysis of electrocardiograph (ecg) signal for the detection of abnormalities using matlab. 8**,** 114-117.

OXFORD. 2019. *Definition of music in English* [Online]. Lexico. Available: https://www.lexico.com/en/definition/music [Accessed 16/7 2019].

PAN, J. & TOMPKINS, W. J. J. I. T. B. E. 1985. A real-time QRS detection algorithm. 32**,** 230-236.

PHYSIONET. 2005. *MIT-BIH Arrhythmia Database* [Online]. Massachusetts: MIT Laboratory for Computational Physiology. Available: https://www.physionet.org/content/mitdb/1.0.0/ [Accessed March 11 2019].

POPE, K. 2019. Analog Front-End Circuit Diagram for ECG. Flinders University.

SAMENI, R., SHAMSOLLAHI, M. B., JUTTEN, C. & BABAIE-ZADE, M. Filtering noisy ECG signals using the extended kalman filter based on a modified dynamic ECG model. Computers in Cardiology, 2005, 25-28 Sept. 2005 2005. 1017-1020.

SCHUTT, K. 2012. *MATLAB and MIDI* [Online]. Available: http://kenschutte.com/midi [Accessed May 31 2019].

SHUTTERSTOCK 2019. Music Notes flowing on white Background, seamless animation.

SINGH, B. N. & TIWARI, A. K. J. D. S. P. 2006. Optimal selection of wavelet basis function applied to ECG signal denoising. 16**,** 275-287.

SNAPEDA. 2019. *Built for electronics designers & engineers* [Online]. Available: https://www.snapeda.com/?gclid=CjwKCAjwxaXtBRBbEiwAPqPxcHNxZQjYTM1AZDy86Quw8QY0YGhC3kd694kPlBrA6cHPmdmNwt989hoC_FgQAvD_BwE [Accessed July 30 2019].

STRASSEER, F. M., MICHAEL; ZOUBIR, ABDELHAK M. 2012. Motion artifact removal in ECG signals using multi-resolution thresholding. *Proceedings of the 20th European Signal Processing Conference*.

SULLIVAN, T. J., DEISS, S. R. & CAUWENBERGHS, G. A Low-Noise, Non-Contact EEG/ECG Sensor. 2007 IEEE Biomedical Circuits and Systems Conference, 27-30 Nov. 2007 2007. 154-157.

THAKOR, N. V. & ZHU, Y.-S. J. I. T. O. B. E. 1991. Applications of adaptive filtering to ECG analysis: noise cancellation and arrhythmia detection. 38**,** 785-794.

7 APPENDIX

```matlab
%% Initial stages: Creating sounds with no musical meaning %%

% File Reader %
data = importdata('ecg_datatest.txt');
x = data(:,1);
y = data(:,2);

% Sound Maker %
% definitions
Fs = 18000;        % Sampling frequency: 8kHz, 44.1kHz, 48kHz, 88.2kHz, 96kHz
i = 1;
block_time = 0.025;
amplitude_threshold = 0.4;
count_threshold = 0.1;
bandwidth = 0.01;
pulse_frequencies = [300 500];

% derived
Ts = 1 / Fs;       % Sampling period
t = 0:Ts:1;        % Time range
B = block_time * Fs;
tB = (1:B) * Ts;
Np = numel(pulse_frequencies);

% make the pulses
g = zeros(Np, B);
tb = ((1-B/2):(B/2))/Fs;
for pi = 1:Np
   g(pi,:) = gauspuls(tb,pulse_frequencies(pi),bandwidth);
end

% while there is enough data in my data vector
ptr = 0;
while numel( y) >= ptr + B
   % choose which frequency to play
   if sum( y( ptr + ( 1:B)) > amplitude_threshold) > count_threshold * B
      pulse_index = 2;
   else
      pulse_index = 1;
   end
   ptr = ptr + B;

   % play the sound
   sound( g( pulse_index, :), Fs);
   pause( block_time);
```

```
end

%% Development Stages: Creating sound of ascending pattern with no musical
meaning %%

% Sound Maker %
% how much feedback do you want?
verbose = false;  %#ok<*UNRCH>

% which source and how to process it
biosignal_source = 'ecg datafile 1';
block_time = 0.15; % Block length for analysing ECG data
amplitude_threshold = 0.4;
count_threshold = 0.1;

% fade in
fadein = 1;
fadeinframes = 100;

% Allowed notes for major scale
fm = 110;  % fundamental frequency for creating audio
Noctaves = 6;
major = reshape(repmat([0; 2; 4; 5; 7; 9; 11], 1, Noctaves) + ...
   repmat(12*(0:(Noctaves-1)), 7, 1), Noctaves*7, 1)';

% create the audio output device
Fs = 44100; % audio sampling frequency
audiodevice = audioDeviceWriter('SampleRate', Fs);

% Load in the biosignal %
switch biosignal_source
   case 'ecg datafile 1'
      data = importdata( 'ecg_datatest.txt');
      fs = 500;
      x = data(:,1);
      y = data(:,2);
      if verbose
         plot(x,y,'b-','LineWidth',2)
         title('ECG Data Set of 108000 Samples')
         xlabel('Time(samples)')
         ylabel('Amplitude')
      end
   otherwise
      error('Unknown biosignal source');
end

% Derived Variables
Ns = numel(y);
Block = round(block_time*fs); % Number of samples per block
```

```
% process a frame at a time %
% initalise for the loop
ptr = 0;
phase = 0;
note_index = 1;
% freq = fm*(1+4*abs(y));
t = (1:(block_time*Fs))/Fs; % Time range
counter = 0;
while ptr + Block <= Ns
   % determine the frequency we want to use
   freq = fm*2^(major(note_index)/12);
   ph = 2*pi*freq*t;
   ph = ph+phase;
   audiosignal = sin(ph)*fadein;

   % Update for next frame
   phase = ph(end);
   fadein = min(1,fadein+1/fadeinframes);

   % Play the sound and advance the ecg frame
   if sum(y(ptr+(1:Block)) > amplitude_threshold) > count_threshold*Block
      audiodevice(audiosignal');
      note_index = mod(note_index, numel major))+1;
   end
   ptr = ptr+Block;
end

% Tidy up
release( audiodevice)

%% Final stage: Monitoring mouse position to create sounds with musical meaning %%

% definitions
framelength = 256;
samplerate = 8000;
ecgfile = 'C:\Users\Adam\Documents\MATLAB\ecg_datatest.txt';
ecgframelength = 16;
which_notes = 'linear_spaced_pentatonic';

% create the allowed notes for pentatonic, major, minor
Noctaves = 16;
pentatonic = reshape(repmat([0; 2; 5; 7; 9], 1, Noctaves) + ...
   repmat(12*(0:(Noctaves - 1)), 5, 1), Noctaves*5, 1)';
major = reshape(repmat([0; 2; 4; 5; 7; 9; 11], 1, Noctaves) + ...
   repmat(12*(0:(Noctaves - 1)), 7, 1), Noctaves*7, 1)';
harmminor = reshape(repmat([0; 2; 3; 5; 7; 8; 11], 1, Noctaves) + ...
   repmat(12*(0:(Noctaves - 1)), 7, 1), Noctaves * 7, 1)';
```

```
% how to generate the sound
fm = 100;
t = ( 1:framelength) / samplerate;

% fade in
fadein = 0;
fadeinframes = 100;

% load the ecg data
ecg = textread( ecgfile); %#ok<DTXTRD>
Ns = numel( ecg);

% use pointer location
Ns = 1e9;
scrsz = get( 0, 'ScreenSize');

% create the audio output device
audiodevice = audioDeviceWriter( 'SampleRate', samplerate);

% process a frame at a time
ptr = 0;
phase = 0;
while ptr + ecgframelength <= Ns
  % play with pointer location
  pl = get( 0, 'PointerLocation');
  freq = fm * ( 1 + 4 * pl( 1) / scrsz( 3));

  % quantise frequency to a note in the scale
  % logarithmically space notes
  switch which_notes
    case 'log_spaced_semitones'
      ns = round( 12 / log( 2) * ( log( 2 * freq) - log( 220)));
      freq = 220 * 2 ^ ( ns / 12);
    case 'linear_spaced_semitones'
      ns = round( freq / 20);
      freq = 220 * 2 ^ ( ns / 12);
    case 'linear_spaced_pentatonic'
      ns = round( freq / 50);
      freq = 110 * 2 ^ ( pentatonic( ns + 1) / 12);
    case 'linear_spaced_major'
      ns = round( freq / 50);
      freq = 110 * 2 ^ ( major( ns + 1) / 12);
    case 'linear_spaced_minor'
      ns = round( freq / 50);
      freq = 110 * 2 ^ ( harmminor( ns + 1) / 12);
    otherwise
      error( 'Don''t know how to make the frequencies');
  end
```

```
    ph = 2 * pi * freq * t;
    ph = ph + phase;
    audiosignal = pl( 2) / scrsz( 4) * sin( ph) * fadein;

    % update for next frame
    phase = ph( end);
    fadein = min( 1, fadein + 1 / fadeinframes);

    % play the sound and advance the ecg frame
    audiodevice( audiosignal');
    ptr = ptr + ecgframelength;
end

% tidy up
release( audiodevice)
```