ENGR9700: Master's Thesis

Respiratory Rehabilitation through an Assistive Electronic Trumpet

Thomas Beltrame

Student ID: 2151250

Primary Academic Supervisor:

Dr David Hobbs

Industry Supervisor:

Dr Jordan Nguyen

Submitted to the College of Science and Engineering in partial fulfilment of the requirements for the degree of *Bachelor of Engineering (Biomedical) (Honours), Master of Engineering (Biomedical)* at Flinders University - Adelaide Australia

Submission Date: 21st October 2019

Declaration

I certify that this work does not incorporate without acknowledgment any material previously submitted for a degree or diploma in any university; and that to the best of my knowledge and belief it does not contain any material previously published or written by another person except where due reference is made in the text.

Thomas Beltrame Date: 21/10/2019

Abstract

The Incentive Spirometer (IS) is a device prescribed to patients following abdominal or thoracic surgery, to facilitate deep breathing and reduce, or prevent, Postoperative Pulmonary Complications (PPCs). The utility of the IS has been debated in recent literature, as disengagement with the IS has been reported to contribute to lowered patient compliance, hence expected respiratory benefits have not been realised (Fernandez-Bustamante, Schoen, et al., 2017). In 2011, over 1 million PPCs occurred annually in the United States, associated with 46,200 deaths and 4.8 million additional hospitalisation days (Shander et al., 2011). IS compliance has been reported to be as low as 6% (Pantel et al., 2017). To potentially improve respiratory outcomes and satisfaction during therapy, a music-focused electronic therapeutic device was modelled from the IS.

The trumpet is a musical instrument Activated by the mouth and breath, which can be naturally coupled to a breath-based therapy such as Incentive Spirometry (ISy). A more interactive and enjoyable respiratory rehabilitation program was integrated through the design of an electronic trumpet. To increase user engagement, multiple mobile applications were designed to integrate with the electronic trumpet, which provided customisable visual and graphic feedback, with the option to include tactile feedback. The combination of adaptability, trimodal feedback and customisation and have been recognised to contribute positively to user engagement (Frid, 2019).

The intricate and complex process required to play modern musical instruments often requires the musician to possess complete muscular control. Such systems impose great challenges for those with physical impairments who wish to access and play music. Music has been shown to promote cognitive development (Talamini et al., 2017; Vaughn, 2000), reduce, or prolong, the onset of dementia and cognitive impairment in the elderly (Balbag et al., 2014), and music therapy has also been shown to improve mental health (Canga et al., 2015; Ferrer, 2007).

The assistive electronic trumpet was designed to be portable, simple, and convenient to use, embedded with onboard processing capabilities and Bluetooth connectivity. In addition to acting as a therapeutic device, the electronic trumpet, also provides musical access to users who lack either the fine motor control or the respiratory capacity required to play a conventional trumpet.

Table of Contents

List of Figures

List of Tables

Acronyms

AARC	American Association for Respiratory Care
ADMI	Accessible Digital Musical Instrument
AMEB	Australian Music Examination Board
COPD	Chronic Obstructive Pulmonary Disease
CPAP	Continuous Positive Airways Pressure
FIS	Flow-Orientated Incentive Spirometers
HVH	High-Volume Hospital
I/O	Input/ Output
IPPB	Intermittent Positive Pressure Breathing
IS	Incentive Spirometer
ISy	Incentive Spirometry
LCD	Liquid Crystal Display
LVH	Low-Volume Hospital
MDT	Morrison Digital Trumpet
MIDI	Musical Instrument Digital Interface
PB	Pushbutton
PEP	Positive Expiratory Pressure
PEEP	Positive End Expiratory Pressure
PPC	Postoperative Pulmonary Complication
VIS	Volume-Orientated Incentive Spirometers
VMI	Virtual Musical Instrument

Abbreviations

App	Mobile Application
Aug	Augmented
Dim	Diminished
Maj	Major
Maj7	Major Seventh
Min	Minor
Min7	Minor Seventh
Sus2	Suspended Second
Sus4	Suspended Fourth
7	Dominant Seventh

Acknowledgements

I would like to sincerely thank my university supervisor Dr David Hobbs, for his continual support, advice, guidance, kindness, assistance and encouragement during my entire degree and throughout the duration of my master's project. David has facilitated numerous projects in the earlier stages of my degree, all crucial in the identification of my passion for rehabilitation and assistive technology.

I am most appreciative of my industry supervisor Dr Jordan Nguyen, for providing me with the opportunity to collaborate with Psykinetic, an organisation I have aspired to support since the commencement of my degree.

I would like to acknowledge James Dean, from Technical Solutions in Victoria, for his integral role in the preliminary stages of my project. James assisted in the identification of which measurement would be most suitable and provided the recommendation of an appropriate differential pressure sensor.

I would also like to express my gratitude to Associate Professor Kenneth Pope, for his critical role in the initial stages of my project. Kenneth was fundamental in the selection of an appropriate instrument to simulate and has offered valuable recommendations throughout the duration of the project.

I am grateful to my partner, Anika Talukder, for her assistance, recommendations and support throughout the duration of my thesis and degree.

Finally, I would like to acknowledge my parents, David and Marianne Beltrame, and my sister, Lara Beltrame for their positive support and continual encouragement throughout my degree.

1 Introduction

It was estimated that 321.5 million surgical procedures were performed worldwide in 2010 (Rose et al., 2015); a substantial increase of 31% from the estimate of 234.2 million surgical procedures performed in 2006 (Weiser et al., 2008). Of the 321.5 million listed procedures, 41.23 million were of respiratory, cardiovascular and digestive in nature, equating to approximately 12.8% of the total number of surgeries performed annually (Rose et al., 2015).

Postoperative Pulmonary Complications (PPCs) have an associated incidence ranging between 6% - 80%, depending on the research method, surgery type and classification of a PPC (Fernandez-Bustamante, Frendl, et al., 2017). A PPC refers to a pulmonary abnormality resulting in a disease or dysfunction combined with a negative patient outcome, usually identified as respiratory failure, pneumonia or atelectasis (Davies et al., 2017). PPCs significantly affect patient outcomes and attribute to an increased hospitalisation duration, morbidity and mortality. Strategies to reduce PPCs can occur preoperatively, perioperatively and postoperatively (Davies et al., 2017). It was found that the occurrence of a complication within the 30-day postoperative period was the strongest determining factor of mortality (Khuri et al., 2005).

The primary focus of this analysis is abdominal or thoracic surgeries, associated with the highest incidence of PPCs (Ruscic et al., 2017). The occurrence of significant PPCs was found in 20% - 40% of abdominal or thoracic surgeries (Bartlett et al., 1973). However, approximately 3% of general surgical procedures result in PPCs (Johnson et al., 2007), due to the disruption of the natural deep breathing pattern while under the influence of anaesthesia. Regular deep breathing prevents atelectasis and is therefore critical to maintaining standard lung function (Bendixen et al., 1964).

In 2011, over 1 million PPCs occurred annually in the United States, associated with 46,200 deaths and 4.8 million additional hospitalisation days (Shander et al., 2011). The annual rehospitalisation cost in the United States was estimated to be USD$835 million (Sabaté et al., 2014).

Introduction

To prevent the incidence of, or reduce the effect of a PPC, the Incentive Spirometer (IS) is commonly prescribed which facilitates voluntary deep breathing, or sustained maximal inspiration (U.S. National Library of Medicine, 2017). Following the invention of the IS by Bartlett in 1970, the IS was shown to be superior in reducing PPCs compared to alternative physical therapies recommended in the 1970s (Craven et al., 1974). Ten IS repetitions are commonly prescribed per waking hour, for the first five postoperative days (do Nascimento Junior et al., 2014), equating to approximately 160 total repetitions per day.

More recently, PPCs were reduced following the appropriate engagement with the IS (Westwood et al., 2007). However, it was reported in 2017 that low patient compliance resulted in the device being used less than the recommended amount, hence expected benefits were not achieved (Fernandez-Bustamante, Schoen, et al., 2017). An analysis of 12 studies found no statistical difference between those prescribed Incentive Spirometry (ISy) and those prescribed either no respiratory therapy, deep breathing exercises, or chest physiotherapy (do Nascimento Junior et al., 2014). Patients have been reported to perform as little as 6% of the recommended exercises (Pantel et al., 2017). It is assumed that patient compliance is a major factor in the results, however, most studies do not record compliance and therefore do not offer a mechanism to validate claims without further investigation.

Acknowledging the additional respiratory complications associated with poor IS compliance (Fernandez-Bustamante, Schoen, et al., 2017; do Nascimento Junior et al., 2014), a more engaging electronic therapeutic device is proposed. The device is intended to be incorporated into respiratory rehabilitation programs; initially through music, with possible extensions to digital gameplay-based therapy. The music produced by the device will reflect the sounds of a trumpet and require logic similar to conventional trumpet play. Unlike a conventional trumpet, however, the breathing emphasis can be placed on either inspiration or expiration, at the discretion or requirements of the user. The electronic trumpet will be designed for both music therapy purposes and as an assistive musical instrument. End-users who lack either fine motor control or the respiratory capacity required to play a conventional trumpet will be able to access music through the proposed electronic trumpet.

Auditory feedback is projected to increase compliance as musical engagement has been demonstrated to enhance physical effort and ability to complete tasks (Elliott et al., 2005).

Music has been shown to promote cognitive development through the improvement of memory (Talamini et al., 2017), mathematical capacity (Vaughn, 2000) and literacy (Department for Education, 2019). The onset of dementia and cognitive impairment was also reduced, or prolonged in the elderly following musical engagement (Balbag et al., 2014). In a medical context, music therapy has demonstrated an improvement in physiological responses such as respiration (Smith et al., 1989), heart rate (Uggla et al., 2016) and blood pressure (do Amaral et al., 2016). Mental health was also positively affected by music therapy, with a reduced incidence of depression (Canga et al., 2015) and anxiety recorded (Ferrer, 2007).

This thesis provides a review of current literature in Section 2, identifying the prevalence and consequences of PPCs, the current modalities to prevent or reduce PPCs and commercially available assistive devices, with a musical or respiratory therapy focus. Section 3 provides an overview of the project while Section 4 describes the developmental process of an assistive electronic trumpet. The hardware and software components utilised to develop the functional prototype were discussed and integrated. Section 5 presents the final outcomes of the project while Section 6 summarises the limitations and future recommendations.

2 Literature Review

2.1 Postoperative Pulmonary Complications

To understand the effects of anaesthesia on pulmonary function, Mead and Collier performed an experiment on anaesthetised dogs (Mead and Collier, 1959). The results indicated that short-term atelectasis, the collapse or partial collapse of the lung, occurred while performing regular and constant tidal ventilation (**Figure 1**) with a volume between 400mL to 500mL (Bartlett et al., 1973). Atelectasis continued to occur until the required pressure was applied to reopen the closed alveoli through a deep breath (Mead and Collier, 1959). It was found that these deep breaths, either a sigh or a yawn, generally occur 5 - 10 times per hour, are critical to maintaining standard lung function, however, do not occur under the influence of anaesthesia (Bendixen et al., 1964).

The abnormality of exclusively breathing shallow breaths, combined with the tidal volume reduction, prevented the natural clearance of secretions and resulted in hypoventilation through retained secretions, leading to atelectasis (Craven et al., 1974) after a number of hours (Bartlett et al., 1973). Atelectasis had the potential to lead to PPCs such as pneumonia (Kigin, 1981). Patients with reduced lung capacity prior to the surgery, due to chronic bronchitis, emphysema, smoking, or obesity, also had a higher chance of developing a PPC (Kigin, 1981). The review conducted by Kigin found that respiratory failure contributed to approximately 50% of postoperative deaths in the 1970s.

Figure 1 has been removed due to copyright restrictions

***Figure 1**: Lung Volumes (Carroll, 2007)*

2.1.1 Postoperative Pulmonary Complication Prevalence

Following any procedure requiring neuromuscular blocking drugs such as general anaesthesia, PPCs often occur due to the reduction in lung volume and change in respiratory and drive muscle functions (Miskovic and Lumb, 2017). Common complications include short-term atelectasis in at least 75% of patients who received a general anaesthetic (Miskovic and Lumb, 2017), pneumonia, and increased occurrences of morbidity and mortality (Sengupta, 2015). The high inflation of the alveoli over a sustained period of time is critical for preventing respiratory diseases (Bartlett et al., 1973).

PPCs after a thoracotomy vary in prevalence, depending on a range of factors relating to both the patient and the procedure (Forrest, 2016). Fisher *et al.* conducted a systematic review of blinded studies ranging from 1966 to 2001 to determine the accuracy of predicting the prevalence of a PPC based on preoperative factors. Although a statistically significant factor could not be identified across studies, it was found that the incidence of PPCs ranged from 2% – 16% (Fisher et al., 2002).

A study conducted by Arozullah *et al.* analysed the outcomes following major surgery for 160,805 patients who were all operated on in 100 Veterans Affairs Medical Centres. It was found the 1.5% of patients developed pneumonia with a 21% mortality rate following the 30-day postoperative period. The surgeries performed were abdominal aortic aneurysm repair, thoracic, upper abdominal, neck, vascular, and neurosurgery (Arozullah et al., 2001).

A later study compared the prevalence of complications following postoperative respiratory failure. Of the 180,359 patients who underwent major vascular and general surgical operations, respiratory failure occurred in 3% of patients (Johnson et al., 2007). Of the 3% of patients, 35.4% developed pneumonia (1.05% of total patients), with a 26.5% mortality rate within the respiratory failure group compared to a 1.4% mortality rate without postoperative respiratory failure, following the 30-day postoperative period (Johnson et al., 2007). The study by Johnson *et al*. comprised of 19.8% female subjects; of while 2.19% developed pneumonia, compared to 3.18% of males. Compared to the findings of Arozullah *et al.*, the number of patients who developed pneumonia was lower, however, it was recognised that only 3.2% of subjects were female. Both studies displayed a significant correlation between the identification of PPCs and mortality, with males shown to carry a greater risk (Johnson et al., 2007).

A study conducted by Dimick *et al.* investigated the likelihood of a postoperative complication occurrence following a high-risk surgery, depending on whether the procedure was performed in a high-volume hospital (HVH) or low-volume hospital (LVH). The findings identified a near 2-fold increase of PPCs when the three analysed operations were performed at a LVH compared with a HVH (Dimick et al., 2003). For oesophageal resection, pancreatic resection and abdominal aortic aneurysm repair, the HVH resulted in postoperative complication for 30%, 6% and 12%, compared with 51%, 12% and 18% for LVHs, respectively (Dimick et al., 2003). It is assumed that the higher quality hospital resources, combined with the superior experience of nursing, anaesthesia and intensive care staff led to the decrease in PPCs in HVHs (Dimick et al., 2003).

The combined PPC prevalence was recorded to be 25.9% (HVH) and 35.9% (LVH), 9.3% (HVH) and 14.7% (LVH), and 8.7% (HVH) and 10.2% (LVH), for oesophageal resection, pancreatic resection and abdominal aortic aneurysm repair, respectively (Dimick et al., 2003).

Depending on the definition followed by each researcher of a PPC, the type of surgery performed and the personal factors of the patient, the prevalence of PPCs do change, therefore making it difficult to confidently identify if PPC reduction strategies have been effective.

It was reported by Bartlett *et al.* in 1973 that between 20% and 40% of abdominal or thoracic operations were followed by pulmonary complications, including atelectasis, tachypnoea, hypoxia, fever, and pneumonia according to sources published in 1957 and 1971 (Bartlett et al., 1973). In approximately 30% of post laparotomy patients, the usual decrease in total lung capacity, functional residual capacity and residual volume were not naturally restored within seven days, and instead presented atelectasis, observable through an x-ray (Bartlett et al., 1973). A review conducted based on studies ranging from 1966 to 2001, where only nonthoracic surgeries were included, reported a PPC occurrence in 2% – 16% (Fisher et al., 2002).

2.1.2 Postoperative Pulmonary Complication Risk Factors

The risk of developing a PPC is dependent on a number of factors, of which age has the most significant influence, followed by the presence of additional medical conditions, and the nature and severity of the operation (**Table 1**) (Forrest, 2016).

***Table 1**: Risk of Postoperative Respiratory Failure (Adapted from Forrest, 2016)*

Risk Factor Multiplier	Risk Factor
6x	Age > 80
4.5x	ASA IV-V
4x	Low serum albumin thoracic surgery
2x	Upper abdominal surgery totally dependent functional state CCF
2x	ASA III neurosurgery emergency surgery COPD
1.5x	General anaesthesia
1.5x	Diabetes
1.5x	Renal failure
1.5x	Anaemia
1.5x	Blood transfusion
1.5x	CVA
1.5x	Recent respiratory infection
1.5x	Recent weight loss
1.5x	Smoking
1.5x	Regular alcohol consumption

Irrespective of the cause, prevention of PPCs is of the utmost importance as PPCs reduce patient quality of life and are associated with significant additional economic factors. PPCs are estimated to cause 4.8 million additional hospitalisation days (Shander et al., 2011) with an annual rehospitalisation cost of USD$835 million in the United States (Sabaté et al., 2014).

Pulmonary impairment was observed up to four months postoperatively following coronary artery bypass graft surgery (Westerdahl et al., 2003), confirming respiratory conditions persist significantly beyond the period of anaesthesia and effective management strategies are critical for the prevention of PPCs.

The review conducted by Fisher *et al.* incorporated more recent studies and does display a reduction in PPCs, presumably due to the incorporation of recent preoperative, perioperative and postoperative techniques implemented to reduce PPCs. However, it is acknowledged that the studies did not measure identical surgical procedures, and due to the number of studies included, the personal factors of each patient could not be localised to a singular population, which would have enabled higher confidence of progress. Due to the high prevalence of PPCs, the development of effective countermeasures is imperative.

2.2 Postoperative Interventions

The postoperative interventions explored in this review include expiratory manoeuvres, CO_2 induced hyperventilation, ISy, and physical therapy aided by music.

A study conducted in 1954 demonstrated postoperative atelectasis occurred in 42% of the control group, 27% of the postoperative deep breathing group and 12% in the preoperative and postoperative deep breathing group (Thoren, 1954). *Bartlett et al.* also reported two studies from 1966 identified sustained and maximum inspiration assisted both the postoperative hypoxemia reversal and alveolar inflation. It was reported by McConnell *et al.* that in 1974, the US spent an annual USD$400 million in the prevention of PPCs (McConnell et al., 1974).

In 1973, Bartlett *et al.* provided a critical review of the physiological effects of performing respiratory manoeuvres postoperatively, with the intention of preventing PPC's. The findings and recommendations of *Bartlett et al.* were analysed and compared against current literature findings to assess the appropriateness of each intervention method.

2.2.1 Expiratory Manoeuvres

Postoperative expiratory manoeuvres, such as coughing, breath-holding and blowing into a balloon or against resistance in post-laparotomy patients induced significant pain after recruiting expiratory muscles such as abdominal muscles (Bartlett et al., 1973). However, expiratory manoeuvres increase the pleural pressure with respect to the airway pressure, causing the alveoli to deflate, the opposite of the desired effect (Bartlett et al., 1973).

It was also recommended that postoperative patients blow on a wind instrument to prevent pulmonary complications. Similarly, blow bottles were recommended postoperatively, by which the patient would blow into a bottle, partially filled with water to create resistance.

The only recorded benefit of the expiratory manoeuvres was the associated preparatory inspiration, however the expiration that followed often resulted in unnecessary pain and effort which did not provide worthwhile benefit (Bartlett et al., 1973).

2.2.2 Intermittent Positive Pressure Breathing

Intermittent Positive Pressure Breathing (IPPB) correctly emphasised inspiration, which resulted in maximal inspiration through the inflation of the lungs by either automatically with a flow-generating pressure-controlled ventilator or manually with a gas-filled anaesthesia bag (Bartlett et al., 1973).

Although theoretically valuable, IPPB has been unsuccessful in studies prior to the 1970s, primarily due to the inflation volume being incorrectly measured. If dependant on the peak airway pressure or resistance was experienced by the patient, the supplied volume was often too small due to the reduction in Functional Residual Capacity following atelectasis, leading instead to shallow ventilation (Bartlett et al., 1973).

In 2002, Méndez-Téllez *et al.* completed a review of IPPB studies and concluded the routine use of IPPB is not supported, nor beneficial. Evidence of appropriate use of IPPB leading to successful outcomes of reducing atelectasis, prevention of respiratory failure, accelerated pneumonia recovery was not reported (Méndez-Téllez et al., 2002).

Modern techniques including a carefully measured inflating volume were shown by Moses *et al.* in 2016 to reduce postoperative pulmonary complications following blunt chest trauma. It was recognised that chest wall trauma which results in fractured ribs reduce the capacity to breathe deeply, leading to pulmonary complications such as atelectasis and pneumonia, and later respiratory failure (Moses et al., 2016). Although the study by Moses *et al.* comprised of only 30 patients, the results found a 36% reduction in pulmonary complications and demonstrated that when appropriately delivered, IPPB is safe and effective (Moses et al., 2016).

2.2.3 CO_2 Induced Hyperventilation

The method of CO_2 induced hyperventilation was also investigated as the monotonous tidal ventilation was improved, and breathing was recorded to be deeper. However, due to the increase in breath rate per minute, the sustained alveolar inflation could not be achieved for a duration with enough length to significantly reduce PPCs (Bartlett et al., 1973).

2.2.4 Positive End Expiratory Pressure

Positive End Expiratory Pressure (PEEP) describes the process of pumping air into the lungs to increase alveolar pressure if the alveoli were unable to inflate. Alveoli pressure reaches its most positive value at the end of inspiration, then decreases to its least positive value at the start of expiration (Bluth et al., 2019). PEEP prevents the pressure from dropping to the original pressure of the lungs, hence alveolar inflation is maintained, preventing the collapse of the alveoli and increasing stability. PEEP also decreases venous return and cardiac output. PEEP is implemented as a peri-operative strategy and was found to be ineffective at reducing PPCs (Bluth et al., 2019).

2.2.5 Incentive Spirometry

Sustained inspiration through deep breathing or yawning has been shown to inflate alveoli and prevent pulmonary complications, primarily atelectasis (Ward et al., 1967). Harken introduced remedial breathing exercises and physical therapy which focussed on the chest to the United States in 1945, following the proposal of post-thoracotomy sustained maximal inspiration in England by Edwards (Harkden, 1946). The technique of holding a deep breath for 3 – 5 seconds was found equivalent to a sigh or yawn in the prevention of a PPC (Bartlett et al., 1970).

The IS is a device prescribed to patients following abdominal or thoracic surgery, to facilitate deep breathing and reduce, or prevent, PPCs (U.S. National Library of Medicine, 2017). The common prescription of use is 10 deep breaths for each waking hour for the first five postoperative days, as reported by the American Association for Respiratory Care (AARC) Clinical practice guidelines (Restrepo et al., 2011), and repeated by multiple authors (Hall et al., 1996; do Nascimento Junior et al., 2014; Overend et al., 2001).

An operational objective of the IS was to reduce the demand for medical personnel (Lederer et al., 1980). The IS is ideally placed on the bedside table, to serve as both a visual reminder and remain within reach, for independent postoperative use (Hough, 1996). Once educated by medical personnel, the IS is expected to be used prior to discharge from the hospital and has been independently continued at the home for up to two months (Basoglu et al., 2005).

2.2.5.1 Device Progression

The Bartlett-Edwards Incentive Spirometer (**Figure 2**) was the first iteration of Volume-orientated Incentive Spirometers (VIS), released in 1973 (Lederer et al., 1980). VIS are designed to encourage consistently low but sustained inspiratory flow (Alaparthi et al., 2016). Detection of the inspired volume in the Bartlett-Edwards Incentive Spirometer occurred through the rise of a piston-like plate along the volumetric shaft (**Figure 3**). Once the desired volume was reached, feedback was provided through battery-powered flashing lights (Lederer et al., 1980).

Figure 2 has been removed due to copyright restrictions

***Figure 2**: Bartlett-Edwards Incentive Spirometer (Van De Water, 1980)*

Figure 3 has been removed due to copyright restrictions

***Figure 3**: Bartlett-Edwards Incentive Spirometer Components (Stock et al., 1985)*

Similarly to the Bartlett-Edwards Incentive Spirometer, the Spirocare provides confirmation of the volume through an incremental illumination of lights as the patient approaches the volume goal (Lederer et al., 1980). The Spirocare, another VIS released in 1975, displayed both the volume objective set by the physician and the number of times the goal was achieved through a visible counter. The device (**Figure 4**) is dependent on mains power and is, therefore, less portable (Lederer et al., 1980). The Spirocare utilised a bellow which would rise upon inhalation between 200mL and 2000mL; the speed at which the propeller within the mouthpiece was spinning determined the depth of inspiration (Lederer et al., 1980).

Figure 4 has been removed due to copyright restrictions

Figure 4*: Spirocare Incentive Spirometer (Frea, 2012)*

Released prior to 1980, the Triflo represents Flow-orientated Incentive Spirometers (FIS) that contain a ball in each of the three chambers positioned in series. Through inhalation, the ball rises due to the subatmospheric pressure generated above each ball, at an inspiratory flow of 600mL/s, 900mL/s and 1200mL/s for the first, second and third ball, respectively (Alaparthi et al., 2016), From **Figure 5** the leftmost ball requires the least flow to lift, compared to the rightmost ball which will require the strongest flow (Lederer et al., 1980). The Triflo was the lightest, cheapest and most portable of the three investigated devices.

The effectiveness of three devices designed to encourage deep-breathing, the Triflo, Bartlett-Edwards Incentive Spirometer, and Spirocare, were investigated and compared following upper-abdominal surgery (Lederer et al., 1980). The Triflo was used the most frequently on average, however, no patient used their respective device as frequently as recommended. Additionally, 15% - 77% of patients did not use their device at all (Lederer et al., 1980).

No statistical significance in respiratory benefit between the three devices was found, therefore providing no therapeutic incentive to support expensive and complex devices (Lederer et al., 1980). The Triflow (**Figure 5**) and Voldyne 5000 (**Figure 6**) represent modern FIS and VIS, the two primary IS categories widely available and currently used. FIS and VIS are plastic, disposable and inexpensive, which surpassed the popularity of the initial electronic models, primarily due to convenience and low-cost (Eltorai et al., 2018).

VIS such as the Voldyne 5000 (**Figure 6**), contains a slider to indicate the desired volume inspired. The main white float will reach to indicator following inspiration of the total required volume. A smaller yellow float in the left chamber was designed to indicate the inspiratory flow.

Figure 5 has been removed due to copyright restrictions

Figure 6 has been removed due to copyright restrictions

***Figure 5**: Triflow Incentive Spirometer (David Jones Pharmacy, 2019)*

***Figure 6**: Voldyne 5000 Volumetric Incentive Spirometer (Healthy Kin, 2019)*

2.2.5.2 Volume-Orientated and Flow-Orientated Incentive Spirometry

While both VIS and FIS mimic the natural process of sighing through encouraging long, slow, deep breaths, the resultant physiological effects differ between the two methods. VIS improve diaphragmatic activity, and require reduced work of breathing and respiratory activity of the intercostal and sternocleidomastoid muscles, assumed to be a result of additional abdominal compartment displacement (Paisani et al., 2013). FIS was found to improve postoperative diaphragmatic dysfunction and preserve inspiratory muscle endurance, in bariatric surgery patients (Pazzianotto-Forti et al., 2015). FIS devices do, however, increase upper chest muscular activity as a higher level of work is enforced (Paisani et al., 2013). It was found that FIS introduced thoracoabdominal asynchrony (TAA) in healthy adult subjects, attributed to an increase in respiratory loading (Paisani et al., 2013), while in the elderly both FIS and VIS lowered thoracoabdominal synchrony similarly (Lunardi et al., 2014). TAA describes nonparallel motion, and in certain cases opposition, of the abdomen and rib cage during inspiration. The asynchrony results from the uncoordinated respiratory muscles relating to lung mechanics, primarily the diaphragm and inspiratory accessory muscles (Pereira et al., 2017). Compared to FIS, a greater chest wall volume can be achieved through VIS (Paisani et al., 2013).

Diaphragm excursion and pulmonary function, assessed through forced vital capacity, were found to improve postoperatively with VIS and diaphragmatic breathing than FIS (Alaparthi et al., 2016). FIS and VIS were compared in relation to increasing the chest wall volume and were found to perform similarly in the elderly (Lunardi et al., 2014), while in adults VIS outperforms FIS (Paisani et al., 2013). FIS was also found to require greater muscle activity in the elderly whereas adults required equal inspiratory muscle activity for both FIS and VIS (Lunardi et al., 2014).

VIS has been demonstrated to provide superior respiratory recovery outcomes when compared to FIS (Paisani et al., 2013; Alaparthi et al., 2016; Pereira et al., 2017), with lowered respiratory frequency and sternocleidomastoid activity (Tomich et al., 2007). However, it was found that the resultant breathing pattern was dependent on the inspiratory flow rate and independent of device type. Low flow rates reduced chest wall movement and increased abdominal wall excursion while high flow rates increased chest wall movement (Chang et al., 2010). The importance of factoring the patient's clinical conditions, age, preference and objective of the therapy, is critical in prescribing the correct form of ISy (Lunardi et al., 2014).

2.2.5.3 *Clinical Effectiveness*

Craven *et al.* conducted a study following the invention of the IS by Bartlett in 1970 to determine the effectiveness of IS. Sustained maximal inspiration achieved with IS was compared to conventional preoperative and postoperative physiotherapy and was found to reduce the number of abnormal postoperative chest X-rays from 63% to 40%, from the physiotherapy and ISy groups, respectively (Craven et al., 1974). The primary findings support the proposal that the sustained maximal inspiration achieved with the IS was superior in reducing PPCs compared to alternative physiotherapy recommended in the 1970s (Craven et al., 1974). ISy was found to reduce PPCs and the length of hospital stay (Ali Khan et al., 2015). Craven *et al.* also found that of the 35 ISy patients, only 11 completed the prescribed number of exercises. This indicated uncooperative patients must be instructed and supervised by a nurse, physician or physiotherapist to receive the full benefit of the device (Craven et al., 1974).

In 2007, an observational study found that alongside other postoperative physiotherapy interventions, IS contributed to a decrease in the incidence of PPCs (Westwood et al., 2007). The visual feedback observed by the patients as the volume inhaled is reflected by the rising of an object was found to encourage users and provide enthusiasm and motivation to continue therapy (Hough, 1996). Thomas *et al.* also found that either ISy or deep breathing exercises provide a greater ability to prevent PPCs than no physical therapy (Thomas et al., 1994).

Do Nascimento Junior *et al.*, conducted a meta-analysis of the evidence for the benefits of IS in 2014, and concluded the evidence cannot confidently assume IS is effective in reducing PPCs following upper abdominal surgery. It was acknowledged that patient compliance was not recorded and should be considered in future studies (do Nascimento Junior et al., 2014).

Recently, the benefit of ISy according to the literature has been inclusive. A study was conducted to determine the effect of respiratory physiotherapy, inclusive of ISy, on PPC occurrence and concluded that no combination of respiratory therapy assisted in the prevention of PPCs (Pasquina et al., 2006). ISy was defended through the argument that few studies record patient compliance and those that do report low levels of engagement with the IS (Narayanan, Hamid, et al., 2016). The low ISy patient compliance was attributed to the absence of expected benefits (Fernandez-Bustamante, Schoen, et al., 2017).

Pfenninger and Roth demonstrated in 1977 that IPPB could be used to produce higher inflation volumes of the lungs than IS, however, reiterated that for IPPB to be successful, the delivery must be optimal. To detect the optimal pressure, accurate spirometry must be performed; to produce high inflation volumes, the flow rate must be as low as possible (Pfenninger and Roth, 1977). It was also speculated that if a trained physiotherapist or intensive care unit was not available to facilitate IPPB, IS will be a more effective form of respiratory rehabilitation (Pfenninger and Roth, 1977).

2.2.5.4 Patient Compliance

A study consisting of 152 patients found that on the second postoperative day, the patients slept for an average of 8.47 hours (Dolan et al., 2016), resulting in 16.87 were waking hours. If applied broadly, at least 160 repetitions of ISy should be performed on the second postoperative day. Bariatric surgery participants were found to use the IS an average of 10.4 times in the second postoperative day (Pantel et al., 2017), approximately 6% of the recommended amount.

Following instructions to perform ten breathing exercises every two hours, daily, during the postoperative period of myocardial revascularization surgery, approximately 67% of the recommended exercises were completed (Renault et al., 2009). During the trial, patients were asked to fill out an "Adhesion Diary" to document all efforts. It is acknowledged that 23% of participants did not complete the adhesion diary, it is therefore unknown if the exercises were performed as recommended as only documented efforts were included in the analysis (Renault et al., 2009), assumed to consist of the more compliant participants. Additionally, it is recognised that the prescription was halved, compared to the amount reported by the AARC, which may have contributed to higher compliance than that reported by Pantel *et al.*

Another study which measured IS compliance following abdominal surgery reported 63% of the recommended exercises were completed (Hall et al., 1996). The compliance was assessed based on a visual linear analog scale, completed by the research nurse (Hall et al., 1996), it is unclear if the nurse was present during the ISy, or if the patient completed a log, which was translated into a visual linear analog scale.

2.2.5.5 *Electronic Accessories to Monitor/Improve Compliance*

To further highlight the dangers associated with non-compliance, a study investigated the effectiveness of an ISy reminder in coronary artery bypass patients. The IS accessory, a SpiroTimer, recorded and timestamped each breath effort, and reminded patients to use the device each hour, with a bell (Eltorai et al., 2019). The experimental group received hourly reminders, while the control group did not receive any reminder. The mean number of daily breaths and recorded hours of use doubled for the experimental group, compared to the control group (Eltorai et al., 2019). As expected, additional engagement with the IS led to physiological benefits including a reduction in presence of atelectasis, postoperative length of stay, intensive care unit length of stay and mortality rate after 6 months (Eltorai et al., 2019).

An additional (unpublished) study was designed to assess the effectiveness of increasing patient compliance through a digital IS, the Smartpeakflow^{TM}. Results are yet to be published, however, the outcome of the study will provide a behavioural insight following interaction with a mobile phone application, designed to encourage, track and report use (U.S. National Library of Medicine, 2019). Similarly, a device was designed at the National University of Malaysia, to monitor ISy performance by attaching directly to the Spiro-ball incentive spirometer. Although not universally applicable to all IS models, the device did enable a quantitative method to assess patient engagement (Narayanan, Ayob, et al., 2016).

Two patents were identified that propose electronic devices designed to integrate with incentive spirometers, both aiming to increase patient compliance. Abandoned in 2013, patent US20130066225A1 describes an ISy monitoring system designed to record all engagement and encourage use through alarms (Kojouri, 2013). Patent US8262583B1 utilises a capacitive sensing circuit to determine performance and outputs an electronic human-like voice, programmed to assist, guide and prompt additional use (Bryant, 2019).

A growing number of electronic additions to the IS have been designed and commercially sold, with the common objective to increase patient compliance. With the exception of the *GroovTube* discussed in 2.2.6, no respiratory therapy device has been released with a musical focus.

In the evaluation of ISy effectiveness, it is therefore essential to record and include patient compliance as a significant contributing factor.

2.2.5.6 Financial Analysis

Despite the conflicting literature evidence of effectiveness, ISy is ubiquitously recommended postoperatively. It was reported that Lahey Hospital & Medical Center in Massachusetts, visited by 3,000 patients daily (Tufts University School of Medicine, 2018), spent USD$33,491 in 2013 on IS devices (Pantel et al., 2017). The total annual financial cost of routinely prescribing ISy after surgery in the United States is estimated to be USD$1.04 billion (Eltorai, Baird, et al., 2018). The financial impact was calculated based on the time physicians, nurses and respiratory therapists spend performing ISy related activities, including the initial education of use, following by reminders and repeated education. The per-patient cost of ISy is predicted to be between USD$65.30 to USD$240.96 (Eltorai, Baird, et al., 2018).

In 2018, the global spirometry market was identified as a USD$655 million dollar industry, and is projected to exceed USD$1 billion by 2023; the projection includes a compound annual growth rate of 9.8% (MarketsAndMarkets Research Private, 2018). The increase in demand was and is forecast to be, due to both an ageing population, and the industrialisation of emerging economies, leading to higher pollution levels and associated respiratory conditions such as COPD (MarketsAndMarkets Research Private, 2018). The growing trend of spirometer sales, which further emphasises the importance of an effective respiratory therapy modality (**Figure 7**).

Figure 7 has been removed due to copyright restrictions

***Figure 7**: Spirometer Sales Trend (MarketsAndMarkets Research Private, 2018)*

2.2.6 GroovTube

The GroovTube is a device designed to facilitate breathing control development and improve motor skills through an interactive modality. The GroovTube inputs to an iPad, with specific apps designed to encourage both inspiration and expiration. The GroovTube is intended to introduce respiratory therapy as an enjoyable and interactive exercise, with the objective of increasing patient compliance.

Eight apps were developed for the GroovTube, each differing in the modality by which the breathing exercise is delivered

The GroovTube can be purchased for €1,200, approximately AUD$1,900, directly from the official GroovTube website in the Netherlands (GroovTube, 2019).

The GroovTube App first requires the selection of an image within the iPad camera roll. An effect such as performing a swirl, pixelating, or expanding a particular section of the image for entertainment purposes occurs once the desired level of force from each breath is detected. (GroovTube, 2018). The expand and swirl effects are shown in the middle and right of **Figure 8**; the original image selected from the camera roll is shown on the left of **Figure 8**.

Figure 8 has been removed due to copyright restrictions

***Figure 8**: GroovTube Screen: Left - Original Image, Middle - Expand, Right - Swirl (GroovTube, 2018)*

2.2.6.1 Fairhammer

The Fairhammer App moves a thermometer style graphic to the top of the screen if the inhaled or exerted breath reaches the required level (**Figure 9**). The input can be toggled between inhalation and exhalation, and the breath strength required can also be modified depending on the user's abilities (GroovTube, 2018). The interface and function of Fairhammer are assumed to be modelled from the IS, with the additional function of training expiration in addition to inspiration.

Figure 9 has been removed due to copyright restrictions

***Figure 9**: Fairhammer Screen (GroovTube, 2018)*

2.2.6.2 Billiard Breath

The objective of the Billiard Balls App is to move the eight balls to the top of the screen in one of the three modes: sequence, power or duration; all three modes have multiple difficulty settings.

Sequence – Move each ball to top with a new breath above the required threshold.

Duration – Maintain a breath of the required strength for the desired period of time, the percentage of time is graphically depicted as all eight balls move towards the top of the screen.

Power – Raise all eight balls at the same time with a single powerful breath above the threshold, (**Figure 10**)

Easy to interpret graphs of progress can also be accessed following each exercise; the zoom functions can be applied to the graph and the axis can be shifted for optimised viewing (GroovTube, 2018).

Figure 10 has been removed due to copyright restrictions

Figure 10*: Billiard Breath Screen - Power Mode (GroovTube, 2018)*

2.2.6.3 Groovy the Dragon

The Groovy the Dragon App is similar in concept to the population game Flappy Bird, however instead of increasing a bird's flight position by tapping, Groovy the dragon moves up or down within the screen depending on whether the user inspires or expires, respectively (**Figure 11**). The objective of the game is to use breath control to dodge obstacles and maximise distance flown (GroovTube, 2018). To increase difficulty, gravity can be toggled, once gravity is off, either inhaling or exhaling is performed as Groovy will automatically reduce flight position, hence continual breath must be exerted for Groovy to maintain a position. Additionally, the required strength of each breath can be modified depending on the ability of the user (GroovTube, 2018).

Figure 11 has been removed due to copyright restrictions

***Figure 11**: Groovy the Dragon Screen (GroovTube, 2018)*

2.2.6.4 Breath Trainer

The Breath Trainer App controls the journey of a cyclist either along a flat surface or up a hill (**Figure 12**), while either inspiring or expiring. The objective is to cycle as far as possible by breathing for as long as possible; to increase the difficulty a minimum strength of breath can be set, or the focus can be entirely based on the length of the breath (GroovTube, 2018).

Figure 12 has been removed due to copyright restrictions

***Figure 12**: Breath Trainer Screen (GroovTube, 2018)*

2.2.6.5 BreathScore

The BreathScore App displays a music score, sheet music, on the iPad in traditional musical notation. The app can digitally play the music score, while visually indicating which note is being played, to enable the user to hear the piece as it is intended to be played (**Figure 13**). The objective of the app is to encourage breathing of varying lengths; each inspiration, or expiration, of the user, is recognised and the audio file of one music note will be played. The objective of this exercise is to encourage fast, shallow breathing as the app will stop producing sound after the duration of a single note, and will not continue until the next breath (GroovTube, 2018).

Figure 13 has been removed due to copyright restrictions

***Figure 13**: BreathScore Screen (GroovTube, 2018)*

2.2.6.6 Breath Music

High-level music creation and control can be made through the three modes, tap, song, and scale.

Tap – After selecting the desired instrument and song, inhaling or exhaling either slows down or speeds up playback of the song.

Song – After selecting the desired instrument sound and the song, the objective is to breathe for as long as possible and 'play' as much of the song as possible in one breath. The sound is not determined by the user, the action of the user blowing, or inhaling, instructs the app to play the embedded song as an audio file (GroovTube, 2018).

Scale – After selecting the desired instrument sound and the key, the eight notes within the octave are cycled through individually; a new note is played with each new breath (**Figure 14**).

Figure 14 has been removed due to copyright restrictions

***Figure 14**: Breath Music Screen (GroovTube, 2018)*

2.2.6.7 PEP

The Positive Expiratory Pressure (PEP) App is a similar version of the GroovTube App, however, designed specifically to integrate with a PEP mask and for children with profound intellectual and multiple disabilities (GroovTube, 2018). Through engagement with the app, the number of repetitions and the appropriate length of breath are configured. Following each correctly performed breath, an effect is applied to an image; **Figure 15** displays the swirl effect performed on the shape of a heart (GroovTube, 2018).

Figure 15 has been removed due to copyright restrictions

***Figure 15**: PEP Screen: Left - Original Image, Right - Swirl (GroovTube, 2018)*

2.3 Music

Music is a form of art associated with the combination of instrumental or vocal accompaniment (Epperson, 2019). Music is used to convey and express emotion, provide entertainment and is referred to as a protean art form, as it is incorporated into writing through lyric and physical movement through dance (Epperson, 2019). Music has been recorded as a part of human culture for the past 60,000 years (Black, 2013), and has been divided into six main periods, Medieval/Gothic, Renaissance, Baroque, Classical, Romantic and Modern; the style a reflection of the era (Paterson, 2019).

2.3.1 Music Education in School Systems

Music in schools has been identified to increase attendance and school activity participation, improve literacy and numeracy performance and build social inclusion (Department for Education, 2019). In both the United States and Australia, music is generally taught once or twice a week in primary school, outside of school hours during middle school and as an elective in high school (The Australian Curriculum, 2019). For the optimal benefit of musical education to occur, it must be incorporated into the curriculum with a stronger focus. Hungary, Japan and the Netherlands have integrated music into the core of their school curriculums and were ranked as the leading three nations following a scientific achievement study for 14 to 17-year-old school students (Kelstrom, 1998).

2.3.2 Effect of Music on Cognition

Bugaj and Brenner published an overview of the correlation between cognitive development and reading skills with music. It was identified that learning to read music is similar to learning to read text as both are generally read from left to right (Bugaj and Brenner, 2011). Additionally, language processing skills such as phonological distinctions, blending and sound segmentation, based on auditory analysis, are similar to the music perception skills required to identify rhythm, harmonies and melodic discrimination (Bugaj and Brenner, 2011). Music and language are temporally organised, as both are perceived as frequencies, produced from a finite set of possible sounds. This similarity in interpretation has been confirmed as music and language share processing components within the brain (Bugaj and Brenner, 2011). Studies involving primary school students strongly suggested those with musical exposure possessed significantly better mathematics, vocabulary and reading skills (Bugaj and Brenner, 2011).

Talamini *et al.*, conducted a meta-analysis to determine whether musicians were shown to have a better memory than non-musicians. The memory was separated into short-term, working and long-term memory and the three memory systems were studied against tonal, verbal and visuospatial stimuli (Talamini et al., 2017). It was concluded that musicians do possess better long-term memory than non-musicians, however, the difference between the three memory systems was not statistically significant (Talamini et al., 2017). Improved working memory for musicians was recorded to be higher for verbal stimuli and significantly higher for tonal stimulation. The multisensorial nature of music training through the association of music notation with sounds and motor actions is assumed to enhance active and controlled learning skills, and hence improve memory for visuospatial stimuli (Talamini et al., 2017).

Musicians have been shown to have a greater capacity to hear speech in challenging listening environments, attributed to the effective combinational use of working memory, stream segregation, and the identification of time-varying perceptual-based cues (Parbery-Clark et al., 2009). The positive correlation between music training and mathematics found that the voluntary study or training of music does improve mathematical achievement (Vaughn, 2000).

The active involvement of playing music through adulthood has also been identified to reduce the likelihood of dementia and cognitive impairment (Balbag et al., 2014). The study comprised of 157 twin pairs, one twin who was confirmed to have either dementia or cognitive impairment, and the other twin who did not (Balbag et al., 2014). The assessment concluded the twin with a higher musical interest possessed a reduced risk of developing dementia (Balbag et al., 2014).

Music has been strongly associated with cognitive development, relating to improved mathematical, scientific (Kelstrom, 1998), literacy and memory skills (Talamini et al., 2017) during the developmental years as a child (Bugaj and Brenner, 2011). Additionally, continual music use has been shown to reduce the likelihood of dementia and similar cognitive impairment later in life (Balbag et al., 2014).

2.4 Music Therapy

Following the positive benefits playing a musical instrument was reported to have on the general population regarding improvements to cognition (Talamini et al., 2017), further investigation into the positive outcomes of music therapy was explored. Music therapy is a recognised paramedical profession, categorised alongside occupational therapy, speech therapy, psychology and physiotherapy, delivered by music therapists (Misic et al., 2010). Music therapy can be applied to individuals across a range of ages and with a variety of conditions including physical impairments, substance abuse, ageing, communication disorders, psychiatric disorders, sensory impairments, interpersonal problems and developmental disabilities (Misic et al., 2010).

2.4.1 Effect of Music Therapy on Blood Pressure

Do Amaral *et al.* performed a meta-analysis of two studies investigating the effects of music on hypertensive patients. The first study, performed in a Chinese aged care facility, identified that listening to classical music at approximately 60 beats/minute significantly reduced systolic blood pressure (SBP) after one month of daily listening (do Amaral *et al.*, 2016). The second study was completed in Turkey and focussed specifically on the effect of listening to Turkish classical music or resting for 25 minutes; the results indicated a reduction in SBP for both groups, however, the music-listening group experienced twice the reduction in SBP benefit (Bekiroğlu et al., 2013). Therefore, music therapy presents a low-risk and simple strategy to reduce SBP in comparison to lifestyle changes and medication (do Amaral et al., 2016).

The benefits of music therapy were also recognised by Zanini *et al.*, who assessed patients over 50 with stage 1 hypertension. The patients participated in weekly musical therapy for 12 weeks, in addition to the conventional treatment, compared to a control group that only received conventional treatment. The outcome of the study showed a significant improvement in the quality of life and blood pressure in patients with stage one hypertension (Zanini et al., 2009). Between the findings of Zanini *et al.* and do Amaral *et al.* it is shown the treatment of hypertension can be positively aided by music therapy, which is capable of eliciting the physiological response of a reduction in blood pressure.

2.4.2 Effect of Music Therapy on Heart Rate

The effect of music therapy on the heart rate of severely sick children younger than 16 were explored by Uggla *et al.* (2016). As paediatric recipients of haematopoietic stem cell transplants (HSCT) possess an increased risk of post-traumatic stress disorder development, the objective of the study was to determine a potential alleviation method of stress (Uggla et al., 2016). Uggla *et al.* reported a significant reduction in heart rate, sustained for four to eight hours after the music therapy, indicating a reduction in stress levels and hence reduction in post-traumatic stress disorder likelihood. No significant changes in blood pressure or blood saturation were observed between the experimental and control group (Uggla et al., 2016).

It was found by Ferrer in 2007 that when patients undergo chemotherapy treatment, live music therapy significantly reduced fatigue and anxiety sensations, and improved relaxation and comfort. It was therefore concluded that music had the potential to improve quality of life and result in positive responses for patients undergoing chemotherapy (Ferrer, 2007). Chuang explored the effect of music therapy on cancer survivors, focussing on fatigue, relaxation and comfort. Heart rate variability was used as an indicator for sympathetic and parasympathetic neural activity. Supporting the conclusion of Ferrer, it was found that music therapy did reduce fatigue and increase relaxation, observed through the increase of parasympathetic nervous system activity in treated cancer survivors (Chuang et al., 2010).

Although focussed on chemotherapy, treated cancer survivors and stem cell transplants, Uggla *et al.* Chuang and Ferrer discovered the positive effect music therapy elicits in physiological responses. The responses were specifically related to the reduction of fatigue and anxiety, and improvement of comfort and relaxation through heart rate reduction.

2.4.3 Effect of Music Therapy on Respiration

A study conducted by Canga *et al.* explored the effect of music therapy on chronic lung diseases, specifically Chronic Obstructive Pulmonary Disease (COPD), in combination with common pulmonary rehabilitation practices. Multimodal music therapy was evaluated, which included music visualisation, therapeutic singing and live wind played (Canga *et al.*, 2015). Music as a non-invasive and non-pharmacological intervention method against breathlessness is well recognised in modern literature (Bausewein *et al.*, 2008). Easy-to-play wind-based instruments have increasingly gained research focus as instruments such as the harmonica, melodica, recorder and slide-whistle facilitate voluntary diaphragmic breathing and body posture awareness, leading to a possible expiratory airway pressure increase (Canga et al., 2015). The literature identified that wind-instrument musicians were capable of voluntary regulation of breath size, achievable through heightened awareness of lung volume changes (Smith et al., 1989).

In COPD patients, depression has an estimated prevalence of 25%, approximately twice as high as people without COPD (van Manen, 2002). Music therapy was observed to reduce symptoms of depression after six weeks of treatment, the difference between the music therapy group and control group was most apparent after an extended period of time (Canga et al., 2015). The improvement attributed to the critical therapeutic relationship achievable through music therapy in patients suffering from breathlessness, trust and empathic understanding enabled the improvement in symptoms of depression (Canga et al., 2015).

The conclusions derived by Canga *et al.* are of particular significance as COPD is one of the primary diseases in which alternative treatment modalities will be proposed.

2.4.4 Effect of Music Therapy in Children and Families

A study conducted by Rose *et al.* (2018) assessed the benefits of a child with special needs learning a musical instrument. The child began the study at the age of 8, underwent music training for one year and was then reassessed. The child was diagnosed with comorbid autism spectrum disorder, attention deficit hyperactivity disorder, sensory processing difficulties, dyslexia, and dyspraxia throughout the duration of the study (Rose et al., 2018). At the conclusion of the study, the child was reported to have improved motor skills, IQ, pattern detection, reasoning, problem-solving and fluid intelligence, and good musical progress was recorded (Rose et al., 2018). The results did indicate a decline in social-emotional functioning, however, it is common for children with multiple development disorders to experience high levels of aggression, behavioural problems and depression (Rose et al., 2018).

Family therapy is a form of psychotherapy that encourages and facilitates discussions regarding development and change within a family context (Nemesh, 2016). Nemesh conducted a study that incorporated music therapy into family therapy sessions. While family-music therapy is commonly used to focus on a child, family member with special needs or families with multiple diagnoses, the objective was to enable exploration of the family as an entity, including non-clinical families who desire therapy (Nemesh, 2016).

Following the completion of one musical intervention implementation workshop, 35 certified family therapists incorporated music therapy into their conventional family therapy sessions (Nemesh, 2016). The results included a stimulation in family dynamics, better communication between family members, more meaningful interactions, and a platform to develop positive change in a safe, non-threatening and playful environment (Nemesh, 2016). By modifying the learning environment and incorporating expressive arts into family therapy, the new behaviours were reported to be effective in promoting family and personal health and well-being (Nemesh, 2016).

Prior to the findings of Nemesh, a study by Twyford investigated the effects of music therapy in New Zealand mainstream schools for students with special needs. The objective of the program was to enable students with special educational needs to develop peer relationships. In all sessions, at least one school staff member was present in addition to the music therapist (Twyford, 2012). The results of the study were separated into five themes: new learning, wellbeing through music, relating to others, musical skills and generalising skills. The study reported consistent positive outcomes across the five themes, and the benefits of the therapy were recorded by both teachers and peers (Twyford, 2012). Benefits included improvements in the student’s confidence, appropriate behaviour and peer interaction, and enthusiasm for non-music related activities and new music skills through which a creative and expressive platform was enabled (Twyford, 2012).

Across the articles explored, music therapy has been shown to be an effective therapeutic modality in the improvement of physiological and psychological responses. The major outcomes from music therapy include improved self-esteem and learning, physical exercise support, a reduction in stress and the likeliness of dementia and seizures, and facilitation of other health-related activities (Misic et al., 2010).

2.5 Musical Instruments

2.5.1 Assistive Instruments

2.5.1.1 Magic Flute

Musician and physical therapist Ruud van der Wel from the Netherlands created The Magic Flute, an inclusive electronic musical instrument designed to enable people with disabilities to access music (My Breath My Music, 2007). The Magic Flute can be played without the use of hands, where the angle of the device is controlled by head movements to alter the note, and the volume is dependent on the velocity of the breath. The angular position is detected by an internal gyroscope and the device can swivel on a camera mount (My Breath My Music, 2007). Through the Musical Instrument Digital Interface (MIDI), the instrument itself can be toggled between 128 different instruments ranging from a flute, saxophone, or trumpet to a guitar, piano or drum kit. The scale of the music can be altered and additional MIDI interfaces can also be connected (My Breath My Music, 2007).

The Magic Flute can be purchased for USD$2,300, approximately AUD$3,000, from Touch the Future, an American-based assistive technology distributor (Touch the Future, 2019).

2.5.1.2 Virtual Musical Instrument

Social isolation, learned helplessness, disengagement and passivity are a number of the risk factors associated with physical disabilities such as cerebral palsy (Hough, 1999). These traits have the negative impact of potentially causing an emotional, medical, psychosocial, cognitive or physical secondary condition which reduces quality of life, health and wellness (Hough, 1999). As previously discussed (Section 2.4), improvisational music therapy was found to increase communicative behaviours in eleven children with autism (Edgerton, 1994).

The objective of the Virtual Musical Instrument (VMI) was to allow children with disabilities to create music based on gestures. (Ahonen-Eerikainen et al., 2008). The VMI operates through video-capture software, combined with the placement and assignment of musical instruments to blocks visible on the screen (**Figure 16**). The VMI is programmed such that if a movement is detected in the area of the drawn shape, the instrument or sound assigned to shape will play, which could be a single note or a chord (Ahonen-Eerikainen et al., 2008). The VMI converts movement in a given region of interest into sound as the user is able to see themselves interact with the shape and produce music as a result. The size of the shape can be drawn to reflect the requirements and control of the user, and the graphic assigned to the shape can be correlated to the instrument through a user-selected graphic (Ahonen-Eerikainen et al., 2008).

Figure 16 has been removed due to copyright restrictions

***Figure 16**: Virtual Musical Instrument (Ahonen-Eerikainen et al., 2008)*

The results concluded that the VMI enhanced the participant's physical, cognitive, communicative, emotional, behavioural and social functioning; the participant was also recognised to develop competency in the VMI through practice and exposure (Ahonen-Eerikainen et al., 2008). Beyond providing a platform for the child to create music, the self-image of the child was restored, auditory, kinaesthetic, and visual skills were increased, and self-awareness was developed while interacting with the VMI (Ahonen-Eerikainen et al., 2008). As a therapeutic tool, the VMI is therefore recommended for the child to develop social-communicative skills, kinaesthetic abilities, cognitive development, socio-emotional growth and motor skills. Additionally, the VMI can be utilised for the purposes of speech therapy, psychotherapy, occupational therapy and physiotherapy, depending on the specific requirements of the child (Ahonen-Eerikainen et al., 2008; Hobbs and Worthington-Eyre, 2008).

2.5.1.3 Quintet

The Quintet is a switched-based musical instrument that can replicate the sounds of any instrument through the MIDI controller (AudioRhoon, 2018). The five multi-coloured switches and control unit (**Figure 17**: Quintet (Frid, 2019)**Figure 17**) enable musical interaction through either individual or group-based play. Songs can be stored within the battery-operated device, and output through the built-in speaker (AudioRhoon, 2018). The Quintet retails for approximately AUD$3,000 and can be used to create music, and as a therapeutic aid (AudioRhoon, 2018).

Figure 17 has been removed due to copyright restrictions

***Figure 17**: Quintet (Frid, 2019)*

2.5.1.4 Human Instruments

Human Instruments (HI) is an organisation dedicated to providing musical instruments opportunities to people with physical disabilities. Rolf Gehlhaar initially developed *sound=space*, a musical instrument which responded to position movement in the human body, prior to founding the organisation with his son, Vahakn Matossian (Human Instruments, 2019).

Human Instruments designed the Hi Note, Touch Chord and Doosafon; the Doosafon is described as a mouth and head operated digital xylophone. Doosafon is a device with components from both Hi Note and Touch Chord; moving upwards or downward enables access of either sharps and flats or natural notes, respectively, while music is generated in combination with a breath and conductive paint key touch (Human Instruments, 2019).

Hi Note (Typhoon)

Hi Note is a hands-free expressive instrument designed to translate the positional movement of the head in the X and Y direction, breath and other mouth characteristics into music. The sensors are contained within a mouthpiece which also relays position information relative to the music key through haptic vibrations (Matossian and Gehlhaar, 2015). Hi Note is a hands-free dual controller of both music and a computer. The 9-dimensional wireless motion sensor combines the head position with breath pressure (**Figure 18**) to compose and perform music (**Figure 19**).

Figure 18 has been removed due to copyright restrictions

Figure 19 has been removed due to copyright restrictions

***Figure 18**: Clarence Adoo Testing Hi Note (Human Instruments, 2019)*

***Figure 19**: Hi Note Breath Component (Human Instruments, 2019)*

Touch Chord (Puffin)

Touch Chord is an instrument designed to be controlled through a combination of breath and touch. With two fingers the musician is able to play notes over three octaves and is the first breath controlled instrument to do so (Human Instruments, 2019). Previously titled Puffin, Touch Chord is polyphonic, enabling multiple notes to be played simultaneously, including access to chords. A note is generated by combining a breath with the touch of a note key. The keys are painted onto the instrument with Bare Conductive – Electric Paint, which offers a superior ability for musical expression unobtainable with a touch screen tablet (**Figure 20**) (Human Instruments, 2019). Certain keys are also modifiable and customisable to the user, hence additional functionality can be incorporated. The synthesizer also enables expression to be conveyed with breath control, ranging from subtle tone changes to soundscape reconstructions (Human Instruments, 2019).

Figure 20 has been removed due to copyright restrictions

***Figure 20**: Touch Chord (Human Instruments, 2019)*

Instrument A

Instrument A is a music composing and performing tool generated through a loop-based sequencer. Through the engagement of a switch such as a blow switch, push switch, footswitch or head switch, the automatically rotating menus can be controlled (Gehlhaar et al., 2014).

The instrument comprises of the graphical user interface and the audio driver. By rotating through the submenus the user is able to select the desired sound through controlling the pitch, volume, position in the loop (Gehlhaar et al., 2014). The device, therefore, can be mouth operated, however, the mouth operation does not create the music directly, it instead enables selection of the desired sound generated in software.

2.5.1.5 Review of Inclusive Musical Interfaces

In 2019, Frid released a review of musical interfaces in inclusive music practice, through which the critical factors of a successful Accessible Digital Musical Instrument (ADMI) was assessed. The review investigated 113 publications released from 1990 to 2018, which collectively assessed 83 instruments (Frid, 2019). The field of ADMIs has grown significantly in recent years following the open-source capabilities of MIDI, the low-cost of electronic hardware, and the rise of digital musical instruments, which offer the opportunity for customisation into an ADMI (Frid, 2019).

Organisations, initiatives and companies such as Drake Music, Open Up Music, Adaptive Use Musical Instruments Projects, One-Handed Musical Instrument Trust, and Human Instruments aim to promote and facilitate access to ADMIs. Commercial musical instruments designed for incorporation into musical therapy sessions include the Soundbeam, Beamz, Skoog, Magic Flute, Quintet, Jamboxx Pro, BioVolt, and Groovtube (Frid, 2019).

The 83 analysed ADMIs were separated into categories based on the primary mode required for sound synthesis. It was acknowledged that while the majority (30) of the ADMIs were tangible or physical, few incorporated vibrotactile feedback, indicating the lack of haptic feedback receivable by many ADMIs (Frid, 2019). Of particular significance were the mouth-operated interfaces, which included only 3 ADMIs (Frid, 2019). The low number of mouth-operated ADMIs currently available indicates a large potential for growth.

Frid acknowledged that the majority of the ADMIs were designed to suit physical disabilities and very few were suitable for those with a visual or hearing impairment. The majority of the ADMIs equally produced either unimodal or bimodal feedback, while only 4.8% presented trimodal, despite findings that multisensory environments facilitate optimal learning and music creation (Frid, 2019). The most successful ADMIs were found to be adaptable, customisable, and were developed by interdisciplinary teams, following user feedback and iterative prototyping.

In conclusion, the frequency of publications in the field of musical instruments with an assistive focus is increasing over time, indicating growth in both interest and knowledge of the ADMI field. Future instruments have the potential to become more intelligent through sensor fusion and machine-learning; it is recognised that most creators intentions are to produce a simple and affordable solution with readily available components (Frid, 2019). Frid identifies the incorporation of multisensory feedback as an area to improve the ADMI field as children with learning disabilities function optimally in multisensory environments (McCord, 1990).

Table 2 was designed to compare the IS with mouth-operated ADMIs, to determine the points of difference between the devices. It was identified that all assess devices accept an analog input, however, no other criteria are met by all four devices. The Magic Flute and Hi Note facilitate expiratory music creation and the GroovTube plays music files following inspiration or expiration. The IS is purely rehabilitative and does not contain and electronic or musical capabilities. It was identified that no current device enables music to be created through inspiration.

***Table 2**: Respiratory Devices Comparison*

Assessment Criteria	**Magic Flute (My Breath My Music, 2007)**	**Hi Note (Matossian and Gehlhaar, 2015)**	**GroovTube (Ruud van der Wel, 2018)**	**Incentive Spirometer (Healthy Kin, 2019)**
	Figures in Table 2 have been removed due to copyright restrictions			
Analog Input	Yes	Yes	Yes	Yes
Create Music Through Expiration	Yes	Yes	Instruct music file to start/stop/change speed	No
Create Music Through Inhalation	No	No	Instruct music file to start/stop/change speed	No
Control pitch with breath	No (breath controls volume, angle controls pitch	No (breath pressure controls volume, head angle selects note)	Instruct music file to start/stop/change speed	No
Designed for music	Yes	Yes	No	No
Designed for therapy	No	No	Yes	Yes
Cost (AUD)	$3,000	-	$1,900 (+ iPad)	$8 - $40

2.5.2 Trumpet

2.5.2.1 Acoustic Trumpet

Prior to 1810, the trumpet was only capable of producing the natural harmonics, and to change key a different trumpet was required. The inclusion of valves in a trumpet enabled a mechanism to decide which tube the trumpeter's breath followed. By extending the length of the tube, the notes would lower in turn, assuming a consistent lip movement and speed of breath (Yamaha Corporation, 2017).

To play the trumpet, the musician blows into the mouthpiece, the sound is expelled through the bell (**Figure 21**). To produce an appropriate sound, the lips must vibrate into the mouthpiece through a steady exhale; the speed at which the lips vibrate directly influences the pitch (Yamaha Corporation, 2017). The mouthpiece is primarily made from silver or brass, and the size and depth influence the sound produced (Yamaha Corporation, 2017). The three tubes each extend the length of the tube by a differing amount, correlating to a lowering by one tone, one semitone, and one and a half tones, for the 1st, 2nd, and 3rd, valves, respectively. Multiple valves can also be pressed simultaneously, to lower the note further (Yamaha Corporation, 2017). The valves produce only an approximation of the desired note and the fine adjustments to produce the exact note occur through mouth adjustments, the 1st slide or the 3rd slide (Yamaha Corporation, 2017).

Figure 21 has been removed due to copyright restrictions

***Figure 21**: Trumpet Components (Yamaha Corporation, 2017)*

The acoustic trumpet requires precise control of the lips, fingers and lungs. For people with cerebral palsy or a similar physical condition which affects muscle coordination and body movement, manipulating a conventional trumpet is often not possible. Similarly, COPD or a similar respiratory condition which affects the amount of breath which can be expelled, the force of that breath and the time it can be sustained, may prevent play (Walters Wright, 2019).

2.5.2.2 Digital Trumpet

Numerous electronic trumpets are available including the Morrison Digital Trumpet (MDT), designed by Steve Marshall and renowned Australian trumpet player James Morrison, available for AUD$2195 (Morrison and Marshall, 2005). The MDT offers advantages such as a MIDI interface to vary the instrument sound produced, transposition to change key without changing the style of play, and software micro-adjustment to ensure the precisely correct frequency is output. The MDT also facilities longer playtime through note control without the requirement to 'buzz' the lips, and produces professional-quality sound without requiring professional technique, both of which reduce the physical requirements necessary (Morrison and Marshall, 2005). However, the device is designed to look and feel similar to an acoustic trumpet (**Figure 22**) and is therefore inappropriate for play with the impairments listed above.

Figure 22 has been removed due to copyright restrictions

***Figure 22**: Morrison Digital Trumpet (Morrison and Marshall, 2005)*

The Yamaha EZ-TP Electronic Trumpet (**Figure 23**) was designed as a teaching tool to demonstrate the required fingering to play a song and included 21 built-in demonstration songs (Japan Trend Shop, 2019). The EZ-TP (now discontinued) is used by ‘singing’ or ‘humming’ into the mouthpiece and the microphone identifies the frequency and applies the sound to the appropriate note. Hence, it does not require traditional trumpet expertise to produce the desired sound (Japan Trend Shop, 2019). The device does appear appropriate in some contexts as an alternative modality to access music if physically impaired. Many past trumpet players who lost their teeth have reported enjoyment with the device (Amazon, 2017).

Figure 23 has been removed due to copyright restrictions

***Figure 23**: Yamaha EZ-TP Electronic Trumpet (Amazon, 2017)*

The Yamaha WX range, the WX5 (**Figure 24**), WX7 and WX11, are MIDI wind controllers based on saxophone finger positions, and due to the key layout and lip pressure sensor, must be played similar to a clarinet or saxophone (Yamaha, 2017). The WXs are capable of producing trumpet-like sounds, as well as any sound accessible by MIDI through the external sound module or tone generator. The WX range has been discontinued but is listed as costing approximately AUD$1,000 (Yamaha, 2017). The objective of the WX range is to replicate traditional music creation through an electronic modality and therefore is not suitable for users who lack the physical ability to play the acoustic equivalent.

Figure 24 has been removed due to copyright restrictions

***Figure 24**: Yamaha WX5 Wind Controller (Yamaha, 2017)*

2.6 Literature Conclusions

Humans regularly incorporate deep breaths, through either a sigh or yawn, into tidal breathing; the deep breaths reopen closed alveoli that would otherwise cause permanent atelectasis (Mead and Collier, 1959). Anaesthesia prevents the natural response of deep breathing and causes atelectasis, which can lead to PPCs such as pneumonia (Kigin, 1981). PPCs have an associated incidence ranging between 6% - 80%, depending on the research method, surgery type and classification of a PPC (Fernandez-Bustamante, Frendl, et al., 2017). It was reported that significant PPCs occurred in 20% – 40% of patients after an abdominal or thoracic operation (Bartlett et al., 1973). In 2011, over 1 million PPCs occurred annually in the United States, associated with 46,200 deaths and 4.8 million additional hospitalisation days (Shander et al., 2011).

It was found that Expiratory Manoeuvres, CO_2 Induced Hyperventilation and Intermittent Positive Pressure Breathing were not effective or appropriate for treating or preventing PPCs, and that emphasis must be placed on sustained maximal inhalation (Bartlett et al., 1973). The IS was found to be the most appropriate tool to prevent or reduce PPC onset (Craven et al., 1974). The utility of the IS has been debated in recent literature, as disengagement with the IS has been reported to contribute to lowered patient compliance, hence expected respiratory benefits have not been realised (Fernandez-Bustamante, Schoen, et al., 2017). IS compliance has been reported to be as low as 6% (Pantel et al., 2017)

Music is known to benefit the player in both a developmental and therapeutic context, at all stages of life. In children, music promotes cognitive development through the improvement of memory (Talamini et al., 2017), mathematical capacity (Vaughn, 2000) and literacy (Department for Education, 2019). In the elderly, music has also been shown to reduce, or prolong, the onset of dementia and cognitive impairment in the elderly (Balbag et al., 2014). For adults or those with a medical condition, music therapy has also been shown to improve physiological responses such as respiration (Smith et al., 1989), heart rate (Uggla et al., 2016) and blood pressure (do Amaral et al., 2016). Mental health was also positively affected by music therapy as shown by a reduced incidence of depression (Canga et al., 2015) and anxiety (Ferrer, 2007).

The acoustic trumpet requires precise control of the lips, fingers and lungs, often not possible for those with physical impairments (Walters Wright, 2019). While electronic trumpets have been commercially introduced, dependency on an acoustic equivalent technique often remains with the exception of the discontinued Yamaha EZ-TP Electronic Trumpet (Japan Trend Shop, 2019).

ADMIs were found to be increasing in both popularity and number of available devices, following the capabilities of MIDI, and music technology advancements (Frid, 2019). ADMIs promote physical, cognitive, communicative, emotional, behavioural and social functioning, and an improved self-image and feeling of environmental control (Ahonen-Eerikainen et al., 2008). Despite findings that multisensory environments facilitate optimal learning and music creation, only 4.8% of the ADMIs included in the review were found to present trimodal feedback (Frid, 2019).

Frid proposed that personal expression through music and full participation in society, including music-making, is a basic human right, however, the design of mainstream musical instruments prevents those with physical disabilities to do so. An ADMI that simultaneously facilities musical expression and creativity coupled with respiratory rehabilitation is yet to enter the commercial market and warrants further investigation (**Table 2**).

3 Project Overview

3.1 Problem Statement

Surgical procedures disrupt the natural, subconscious, deep breathing process required to maintain healthy lung function, leading to the development of PPCs in 6% - 80% of postoperative patients (Fernandez-Bustamante, Frendl, et al., 2017). The IS is prescribed to patients following surgery, to facilitate deep breathing and reduce or prevent PPCs. Disengagement with the IS has contributed to lowered patient compliance, hence expected respiratory benefits have not been realised (do Nascimento Junior et al., 2014; Fernandez-Bustamante, Schoen, et al., 2017).

3.2 Project Objectives

The primary objective of this project was to develop a cost-effective engaging respiratory therapy device, which provided health benefits equivalent to the IS. Rather than exclusively providing visual feedback, the proposed device also provided auditory feedback, in the form of music. The device was designed to replicate a trumpet in logic, where the strength of the breath was the primary input, however, was configured to accept inspiratory or expiratory breaths. To further promote engagement with the electronic trumpet, a mobile application (app) interface was incorporated into the project design, which provided three different modes and enabled user customisation. The electronic trumpet was designed to be convenient to use and was therefore battery-powered and on-board processing was enabled.

A secondary objective was to develop a breath-based assistive musical instrument for people with physical disabilities. The trumpet was designed to be played as an expressive musical instrument, or as a therapeutic tool with a musical output.

4 Methodology

4.1 Hardware Development

4.1.1 Breath Sensor

The primary objective of the device was to facilitate the control of an electronic trumpet through breathing. It was critical to enable the user to simulate ISy through inhalation, and acoustic trumpet play through expiration. To ensure users received maximum benefit from the device, it was identified that customisation and adaptation to the user's individual preferences were critical (Frid, 2019).

During an examination of pre-existing assistive technologies, sip 'n' puff devices were explored, as inhalation and exhalation could both be utilised as control methods. Upon further investigation, it was understood that the sip 'n' puff systems accepted only digital inputs (Special Needs Computer Solutions Inc., 2019). Digital inputs are optimised for activation of a system, however, are unusable as a method of selecting from 31 options in a single action.

Sensors capable of accepting analog inputs were examined and evaluated based on their ability to translate breath information into an electrical signal. It was identified that a differential pressure sensor could analyse breathing patterns. A differential pressure sensor compares the difference between two recorded pressures. The MPXV7002DP is a commercially available sensor that possesses two sensing elements (DigiKey Electronics, 2019a). In breath-based applications, a tube ending with a mouthpiece was attached to one sensing element, while the other remained unconnected, providing a constant comparison with atmospheric pressure. A review of respiratory pressures confirmed that Maximal Expiratory Pressure (MEP) is greater than maximal inspiratory pressure (Evans and Whitelaw, 2009). Based on the derived formula for adults, $MEP = 174 - (0.83 \times Age)$, it was found that the MEP occurs at age 20, at $157.4\, cm\ H_2O$ (Evans and Whitelaw, 2009), or $15.4kPa$. Trumpets and incentive spirometers are controlled through the flow rate of the breath, both include a mechanism for air to exit the device and prevent the build-up of pressure. Therefore, the maximum pressure required for detection is significantly lower than the maximum respiratory pressure a person is capable of generating. The introduction of an air-outlet valve into the electronic trumpet tube significantly reduced the recorded pressure, hence, a pressure range of $\pm 2kPa$ was deemed acceptable.

From the MPXV7002DP datasheet, the critical specifications examined were the accuracy, operating pressure and the *Auto Zero* capabilities. *Auto Zero* negates the zero pressure output reading of the device, in the event that mechanical stresses and the mounting position altered the output (DigiKey Electronics, 2019a). Through extensive research, the MPXV7002DP was selected as the primary input sensor as the requirements described in **Table 3** were met. All MPXV7002DP specifications can be found in detail in the datasheet, located in Appendix A.

Table 3*: Pressure Sensor Specifications*

Specification	Requirement	MPXV7002DP Capability	Justification
Signal Type	Analog	Analog	Detection of breath variation was required to select different notes
Accuracy	< 3%	2.5%	10-bit analog reading obtained from the sensor with 31 different notes, the note selection had a tolerance of 33 values, accounting for 3% error tolerance
Pressure Range	$\pm 2kPa$	$\pm 2kPa$	Air relief mechanism configured to release 87% of maximal expiratory pressure
Auto Zero	Yes	Yes	Account for misaligned mounting position or variation in zero-state output

At atmospheric pressure, or without the detection of a breath, the MPXV7002DP will return a value at half the maximum analog value. Based on the configuration that the tube was connected to the top sensing element (**Figure 25**), an exhale will raise the analog output towards the sensor's maximum, while an inhale will lower the analog output towards the sensor's minimum. For the purposes of simplification during prototyping, a potentiometer will initially be utilised to simulate the output of the MPXV7002DP, prior to the differential pressure sensor integration.

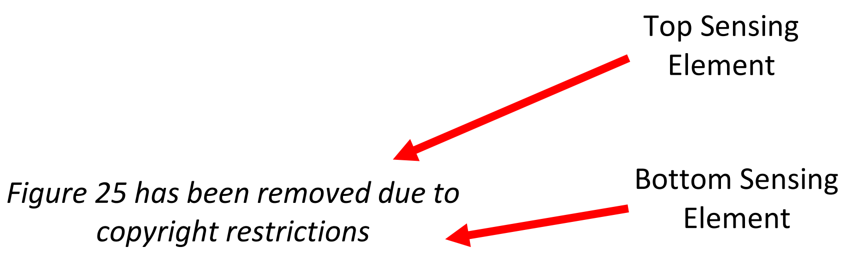


Figure 25 has been removed due to copyright restrictions

Figure 25*: MPXV7002DP Differential Pressure Sensor (DigiKey Electronics, 2019a)*

4.1.2 Arduino Uno

An Arduino Uno Development Board was utilised as the primary microcontroller during prototyping. Arduino is an open-source platform and was selected due to extensive online support and documentation (Arduino, 2017). The capabilities of music production within the Arduino was first explored through inbuilt functions, followed by more complex MIDI libraries and sensor integration.

The Arduino function `tone()` was trialled initially, which delivered a specified frequency to the designated output pin; the output pin was connected to a buzzer. The `tone()` function generates a square wave at the designated frequency and is limited to outputting a single tone, (Arduino, 2019a). To control the frequency of the buzzer, a potentiometer was implemented; the Arduino Uno contains a 10-bit analog to digital (A/D) converter and the `analogRead()` function, therefore, returned a value between 0 and 1023 (Arduino, 2019b). To regulate the output volume, a second potentiometer was introduced that dynamically reduced the voltage supplied to the buzzer.

Three pushbuttons were also implemented to replicate the effect of the three valves of a trumpet (**Figure 26**), where the pitch could be lowered by one to six semitones, depending on the combination of valves pressed. **Table 4** identifies the number of semitones the note was lowered by, depending on the combination of valves pressed; a valve status of `0` refers to the unpressed, or open, position, while a `1` refers to a valve in the pressed position.

***Table 4**: Valve Effect*

Valve Status	Semitones Lowered
0-0-0	0
0-1-0	1
1-0-0	2
0-0-1	3
1-1-0	3
0-1-1	4
1-0-1	5
1-1-1	6

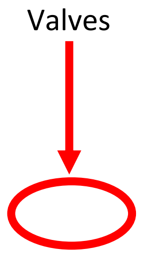


Figure 26 has been removed due to copyright restrictions

***Figure 26**: Trumpet (Yamaha Corporation, 2017)*

The desired frequency was derived from the output of the centre potentiometer, connected to analog Pin A5, depicted in orange (**Figure 27**). The frequency information was then sent to volume regulating potentiometer, connected to analog Pin A0 depicted in blue, and finally, the output frequency was supplied to the buzzer (**Figure 27**). The pushbuttons returned `HIGH` when unpressed and `LOW` when pressed, and therefore operated as active low, connected to digital Pins 8, 9 and 10, in green (**Figure 27**).

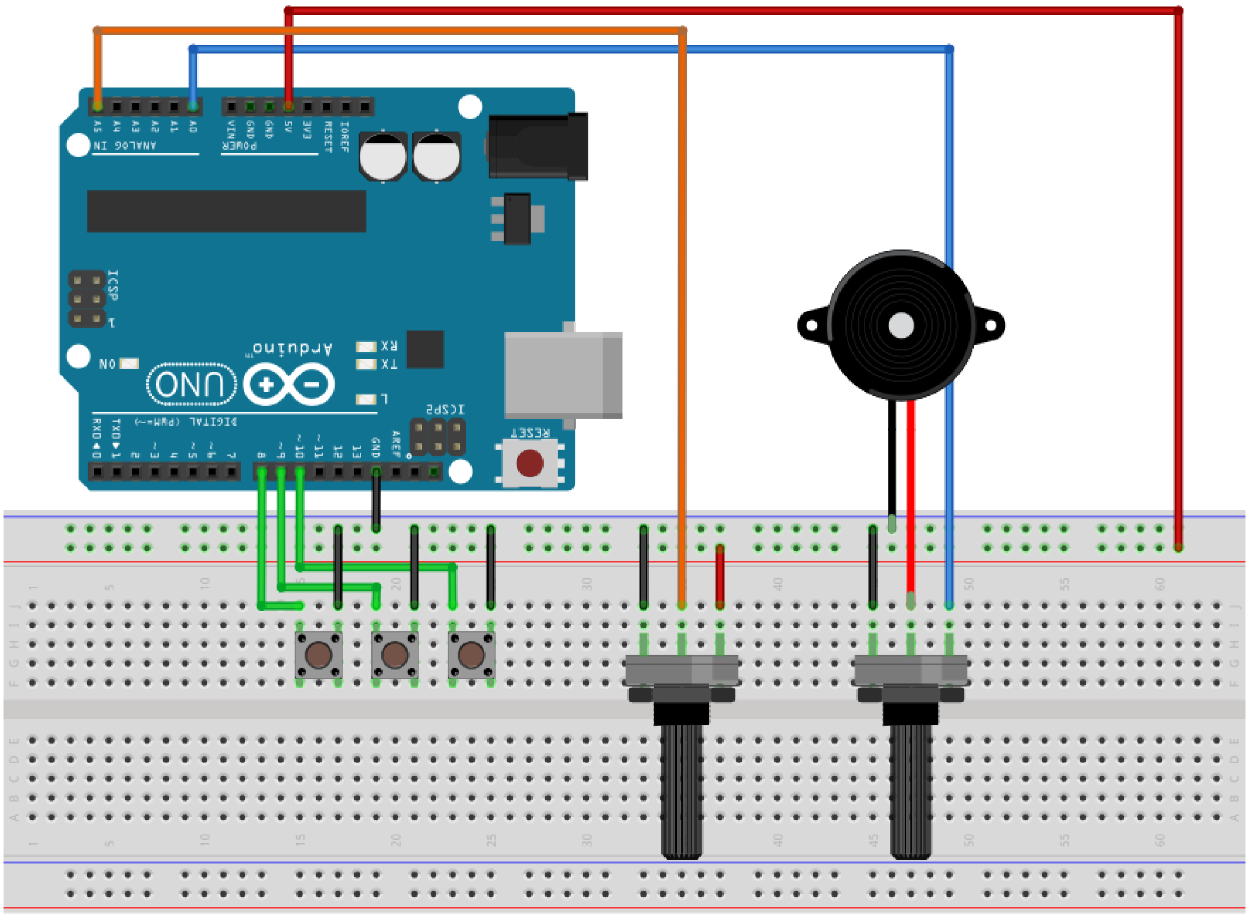


***Figure 27**: Arduino Pushbutton and Potentiometer Connections*

The output frequency did not correlate to a specific note and simply confirmed a tone could be generated. To simulate the frequencies correlating to notes on a trumpet, the 31 standard trumpet notes from E_3 - Bb_5 (Haas, 2011) were identified and assigned their respective value in Hertz, ranging from 164.81Hz - 932.33Hz (Michigan Technological University, 1998). To ensure the output frequency did not occur in-between notes, a scaling factor of 33 was applied to the potentiometer value, which then referenced a lookup table to correlate the name of each note to its frequency equivalent.

The `tone()` function was confirmed to regulate frequency based on the value returned by the potentiometer and the status of the pushbuttons, however, the sound produced did not represent any musical instrument and purely represented a buzzer. The `tone()` function was therefore not appropriate for music generation and alternative approaches were investigated.

4.1.2.1 MIDI Integration

Standardised in the 1980s, MIDI is a protocol that enables communication between computers and instruments for electronic music generation and synthesis (The MIDI Association, 2016). MIDI enables musicians to control instruments digitally, either through computer software or electronic instruments, and edit instrument parameters such as volume or add synthetic effects. Modern electronic keyboards utilise MIDI to output the sound of different instruments, such as a guitar, despite maintaining a piano interface (The MIDI Association, 2016).

Editing MIDI sequences is simplified through the ability to alter the tempo, key or instrument. The sequences can also be converted into music notation, or music notation can be played back, enabling the composition of music without the required standard prerequisite music theory knowledge (The MIDI Association, 2016). Similar to the `tone()` function, the frequency is specified, however, the status and velocity of the note are also included, enabling more sophisticated musical simulation.

The Arduino Uno is not a recognised MIDI instrument and could not directly interface with music production software. The MIDIUSB library that provided critical MIDI communication commands was therefore incompatible. (Gratton, 2011).

Hairless MIDI and loopMIDI are two software applications that when combined, facilitate MIDI message transmission from the Arduino Uno, through serial communication (Gritti, 2016). Configuration of Hairless MIDI and loopMIDI, as detailed by Gritti, enabled the Arduino Uno to interface with a music production software package. However, hardware limitations prevented access to the MIDIUSB library (Gratton, 2011).

Gritti manually created a function capable of transmitting MIDI messages without the MIDIUSB library, which was modified and applied. Ableton Live 10 Lite was then configured to receive MIDI messages from the Arduino. To send a note, the pushbutton connected to digital Pin 7 of the Arduino was pressed; the note was determined by the position of the potentiometer (**Figure 28**) The notes were successfully output as a `Silk Horn Synth Brass` instrument, the most appropriate instrument available to simulate an acoustic trumpet within Ableton Live.

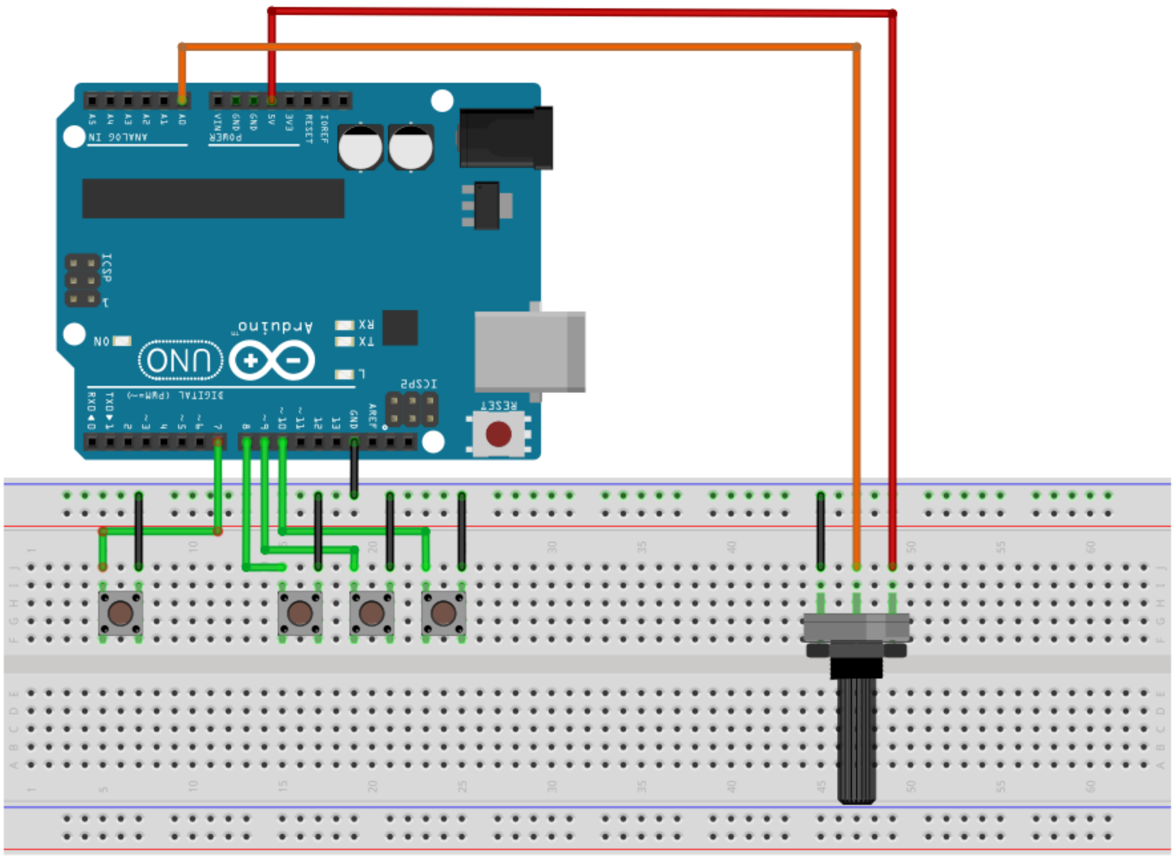


***Figure 28**: Preparation for Ableton Live Connections*

The primary disadvantage of utilising Hairless MIDI application was that any other form of communication across the serial port was prevented. While transmitting MIDI commands to Ableton Live, the Arduino could not be reprogrammed, and information could not be sent to the serial monitor. Hence, Hairless MIDI could not be connected during the debugging of the program if access to values displayed on the serial monitor was required.

The complete Hairless MIDI, loopMIDI and Ableton Live configuration process is detailed in Appendix B. Additionally, a technical description of the Arduino Uno hardware limitations, an analysis of the MIDI function written by Gritti, and the modifications applied, are also explained.

4.1.2.2 Visual Display

It was desired to display information regarding the note being played, and whether inhalation or exhalation was expected. As the serial monitor was inaccessible while providing information to Ableton Live, the 1602A LCD module was incorporated into the prototype to provide visual feedback.

The Liquid Crystal Display (LCD) module was configured in 4-bit mode, 12 connections were therefore required (de Bakker, 2019). Four orange `Data` pins, the blue `Enable` pin and the green `Register Select` pin were connected to the Arduino, while the black `Ground` and red `Power` pins, and yellow `Display Contrast Adjustment` pins were connected to the breadboard and potentiometer, respectively (**Figure 29**).

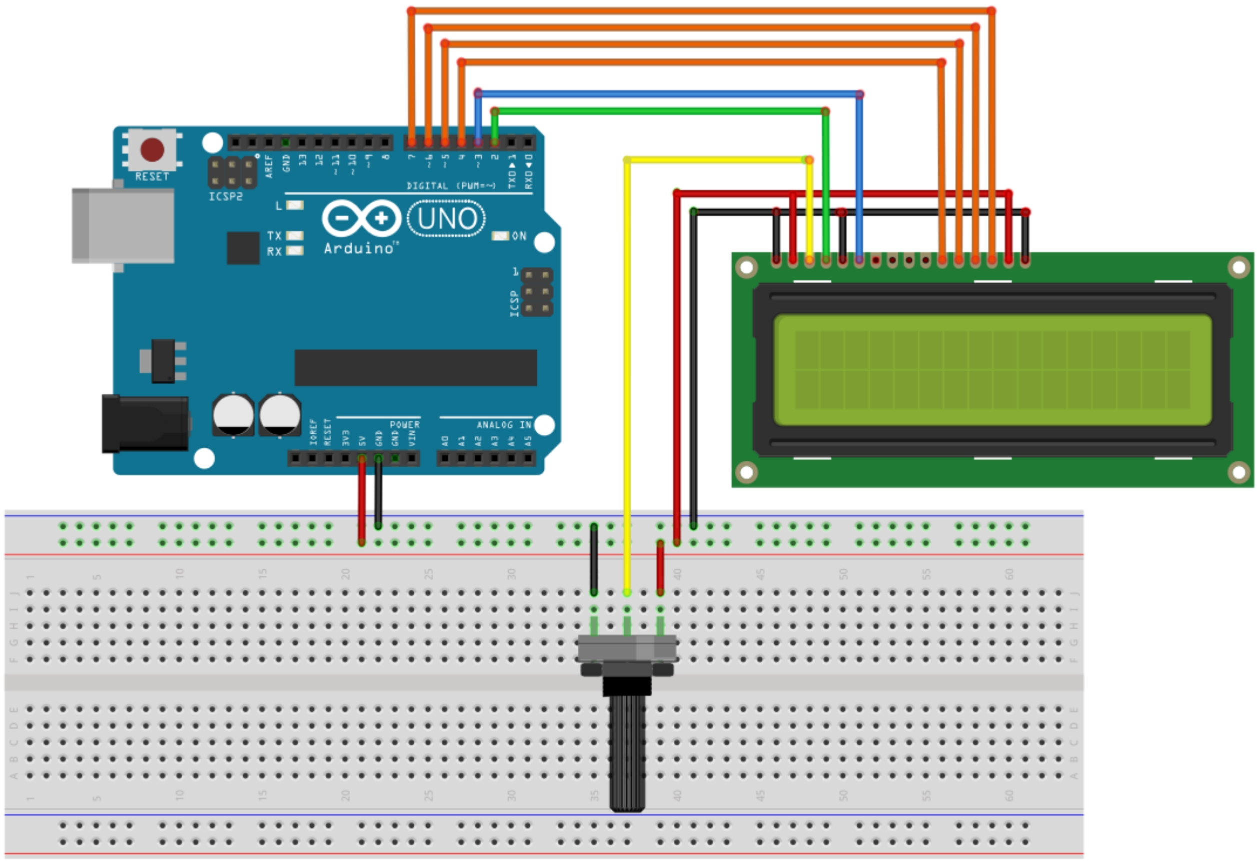


***Figure 29**: LCD to Arduino Connections*

The 1602A is a 16 pin, 16 x 2 character display LCD module, with a maximum of 32 characters displayable at once. The mode of breath, numerical index of the note and corresponding note represented as a letter were assigned the positions of the first line, bottom left and bottom right of the LCD, respectively (**Figure 30**). The potentiometer was substituted for the differential pressure to confirm the output could be controlled through inhalation or exhalation. As both sensors were read through a 10-bit A/D, the input was equivalent to the Arduino and therefore, the output produced the expected `Silk Horn Synth Brass` note.

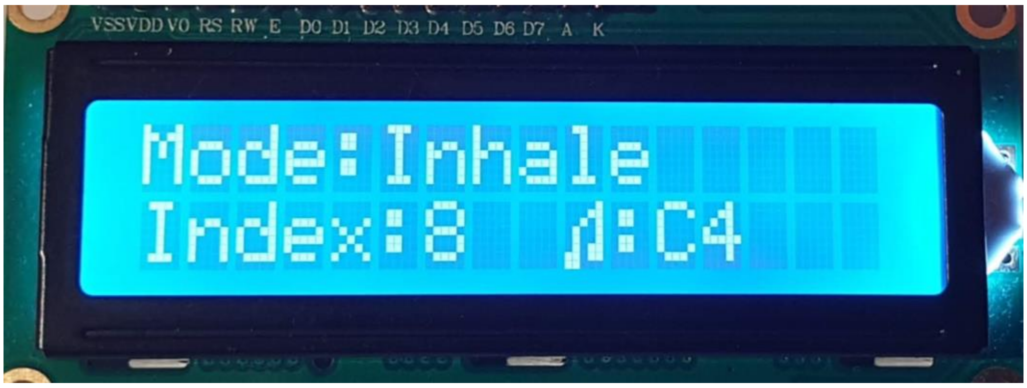


***Figure 30**: LCD Output:*
Top Left – Breath Mode, Bottom Left – Note Index, Bottom Right - Note

Pin 1 is required to be reserved for serial communication within the Arduino (Arduino, 2019c). The `Register Select` pin was initially connected to Pin 1 of the Arduino, which resulted in the improper display of characters.

Musical association to the character which represented the note was provided through a pair of quavers (**Figure 30**). Typically, only standard ASCII characters can be directly displayed on an LCD, however, the quaver pair was defined as a custom character.

The full developmental process including the consequences associated with incorrect connections, the design of the custom quaver pair character, and a description of the 1602A LCD module pinout and Arduino LCD functions are detailed in Appendix C.

The Arduino Uno coupled with Ableton Live was able to convert the flow rate of the breath into a musical note. However, the objective of increasing respiratory therapy compliance through an engaging and convenient device was not met due to the additional software interfacing required. An alternative MIDI-compliant developmental board was therefore investigated.

4.1.3 Teensy 3.6

The Teensy 3.6 USB Development Board (**Figure 31**) is a USB-based microcontroller development system, with a 32 bit 180 MHz ARM Cortex-M4 processor, full specifications are located in Appendix D. The primary reason for the selection of the Teensy board was due to the USB MIDI capabilities through configuration within the Arduino IDE (PJRC, 2017).

Figure 31 has been removed due to copyright restrictions

***Figure 31**: Teensy 3.6 (PJRC, 2017)*

Sending and receiving MIDI messages as a 'class compliant' MIDI device enabled the Teensy to communicate with music software packages directly and independently. Therefore, programs that translated serial data into MIDI data, such as Hairless MIDI and loopMIDI, were no longer required, simplicity in design and use were subsequently increased.

Standard MIDI messages to turn notes on or off could be sent by selecting the USB Type to contain MIDI within the Arduino IDE (PJRC, 2017). The Teensy 3.6 connection configuration is shown in **Figure 32**, through which the differential pressure sensor and three pushbuttons are inputs.

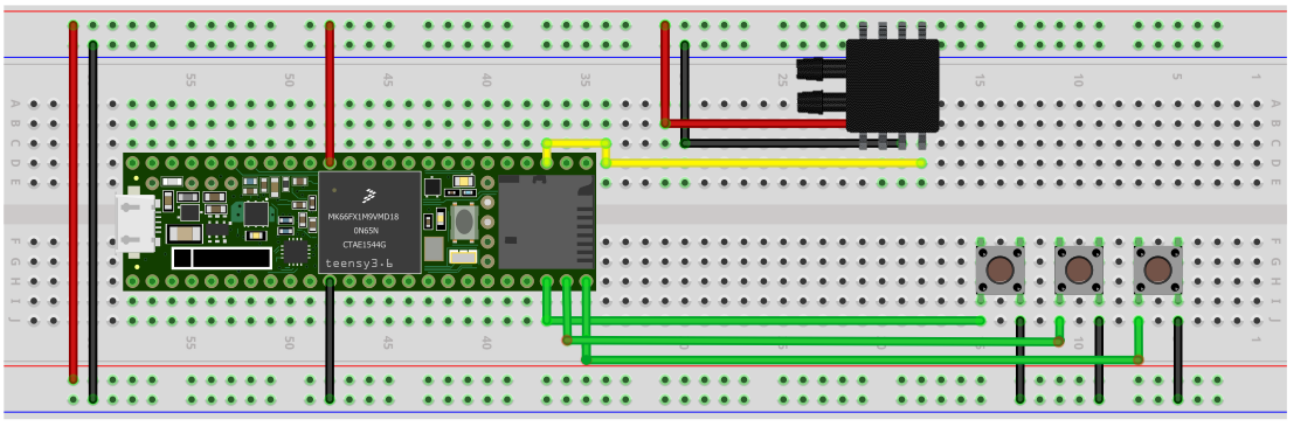


***Figure 32**: Teensy 3.6 Connections*

The Teensy 3.6 coupled with Ableton Live was able to simply convert the flow rate of the breath into a musical note. However, further convenience was desired through the removal of music production software dependency. A method to achieve on-board musical processing was therefore investigated.

The primary user interface was a plastic tube connected to the differential pressure sensor. The plastic tube was designed to represent the mouthpiece and flexible tube, or lead pipe, of either the incentive spirometer or trumpet, respectively. The output was controlled by a continual breath and a modular relief valve was attached to the mouthpiece to facilitate continual air movement. The valve could be opened fully to enable a user with exceptional lung capacity to breath maximally or partially opened for users with reduced lung capacity. The principle of the valve was based on regulating the volume, with the air released through an inline tap connected to the breathing tube **Figure 33**.

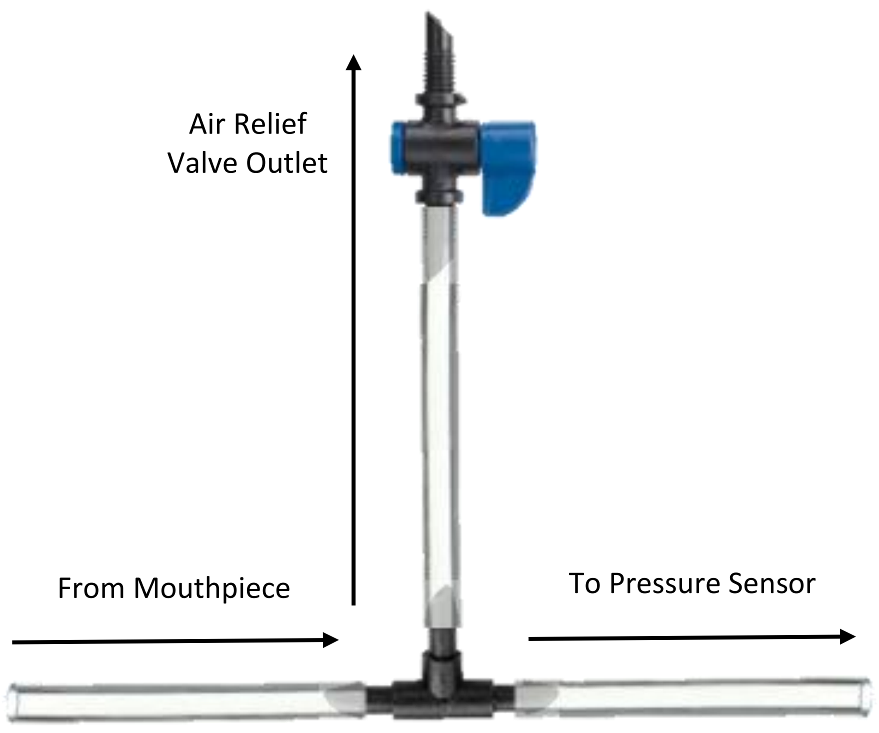


Figure 33: *Modular Air Relief Valve*

4.1.4 Teensy Audio Adaptor Board

The Teensy Audio Adaptor Board (**Figure 34**) enables the incorporation of CD-quality audio into Teensy-based projects. The board supports mono microphone and stereo line-level inputs, and stereo headphone and stereo line-level outputs (PJRC, 2019a). The board integrates with the Teensy Audio Library, a purpose-built library which provides access to audio processing objects. The library enables simultaneous play of the input and output, multiple audio file playback, synthesized waveform creation, the application of effects, and real-time fast-Fourier transform analysis, to create sound-reactive projects (PJRC, 2019a). Most importantly, the board enables capabilities to output audio without the requirement to be connected to a computer.

Figure 34 has been removed due to copyright restrictions

Figure 34*: Teensy Audio Adapter Board (PJRC, 2019a).*

The pin layout of Teensy Audio Adaptor Board (Rev C) was designed correlate to the pin layout of the Teensy 3.6 modules, therefore enabling direct mounting in the form of a shield (PJRC, 2019a).

While attempting to remove dependence on computer software, 87 MP3 samples of each note of the trumpet, ranging from the A0 to Gb7, were located (VexFlow, 2014). Although each note is unique in frequency, the duration of the sample and audio characteristics displayed equivalence across each sample. **Figure 35** displays the audio profile of the A3 MP3 sample, representative of the collection of samples. The duration of each audio file is 2.456 seconds; however the files do not contain a consistent sound; a 0.055-second absence of audio output can be observed at the beginning of each file, followed by a temporary increase in amplitude before stabilisation occurs.

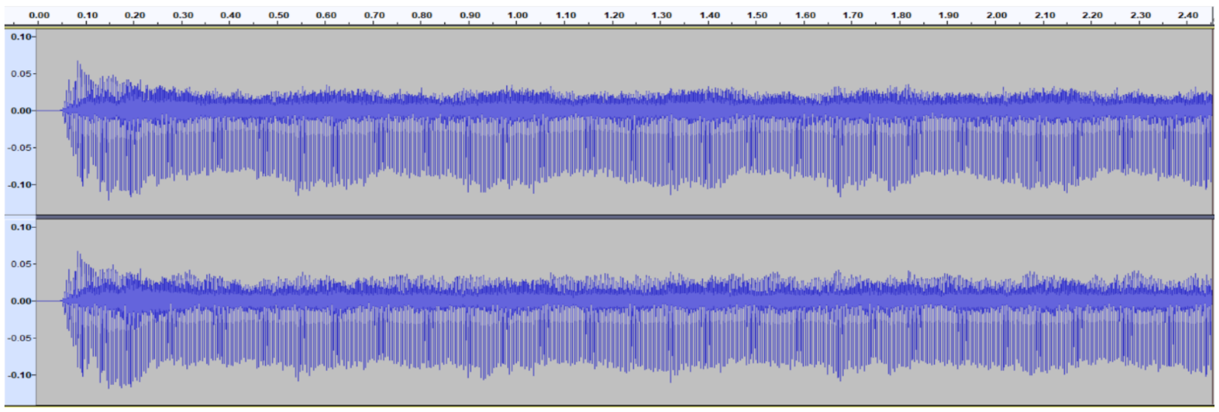


***Figure 35**: A3 Musical Note Audio Profile*

The Teensy Audio Adapter Board cannot playback MP3 files, the files were, therefore, converted into WAV files and saved onto the SD card of the Teensy Audio Board. The A3 note was looped during playback with a `playSdWav` object. The output produced an audible trumpet note, however, the initial lack of audio prevented immediate playback and the length of the note could not be effectively controlled. When looped, the overlaying between the ending and beginning of the track was distinctly noticeable and did not simulate the true sustain of a note.

To reduce the obviousness of the loop, a second `playSdWav` object was created and played at an offset to the first object. The second file was played 0.06 seconds prior to the completion of the first file; audio was, therefore, output during all instances of the desired playback (**Figure 36**). The additional 0.005 seconds was programmed to enable crossfading; the volume of the second file was progressively decreased as the first file was increased, in an attempt to create a smooth transition between audio tracks. Tuning was implemented through proportional gain control during the crossfade, however, despite all efforts, the looping of each track remained detectable.

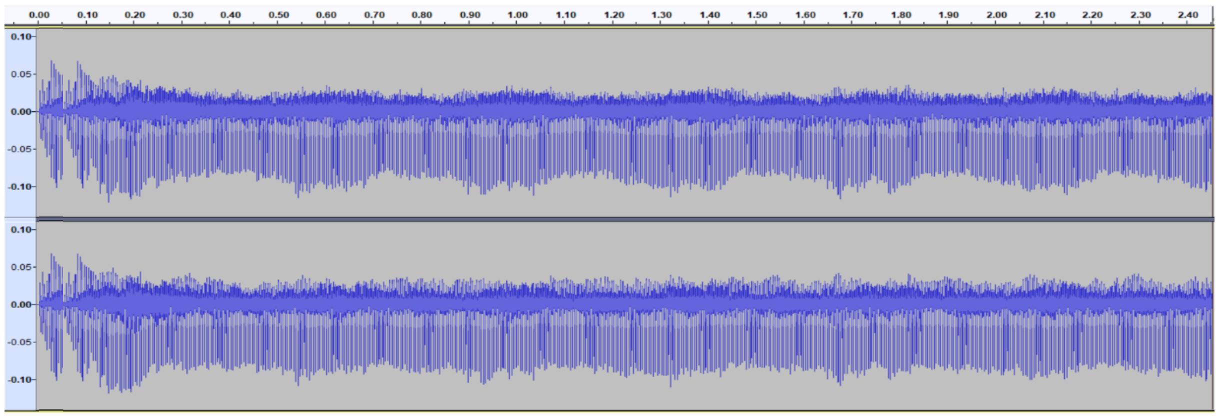


***Figure 36**: Crossfaded A3 Audio Profile*

In addition to the audible loop, alternative strategies were investigated as playing an audio file did not utilise the advanced synthesis capabilities of the Teensy Audio library and restricted the user to an unmodifiable output.

Within the Arduino IDE, the `simpleWavetable` example was loaded to understand the capabilities of a wavetable object. The example enabled playback and control of a flute sample by referencing the C++ class `Flute_100kbyte_samples`. The corresponding header file which contained the class functions and definitions and cpp file which contained the implementation of the class were contained within the example folder (Microsoft, 2018).

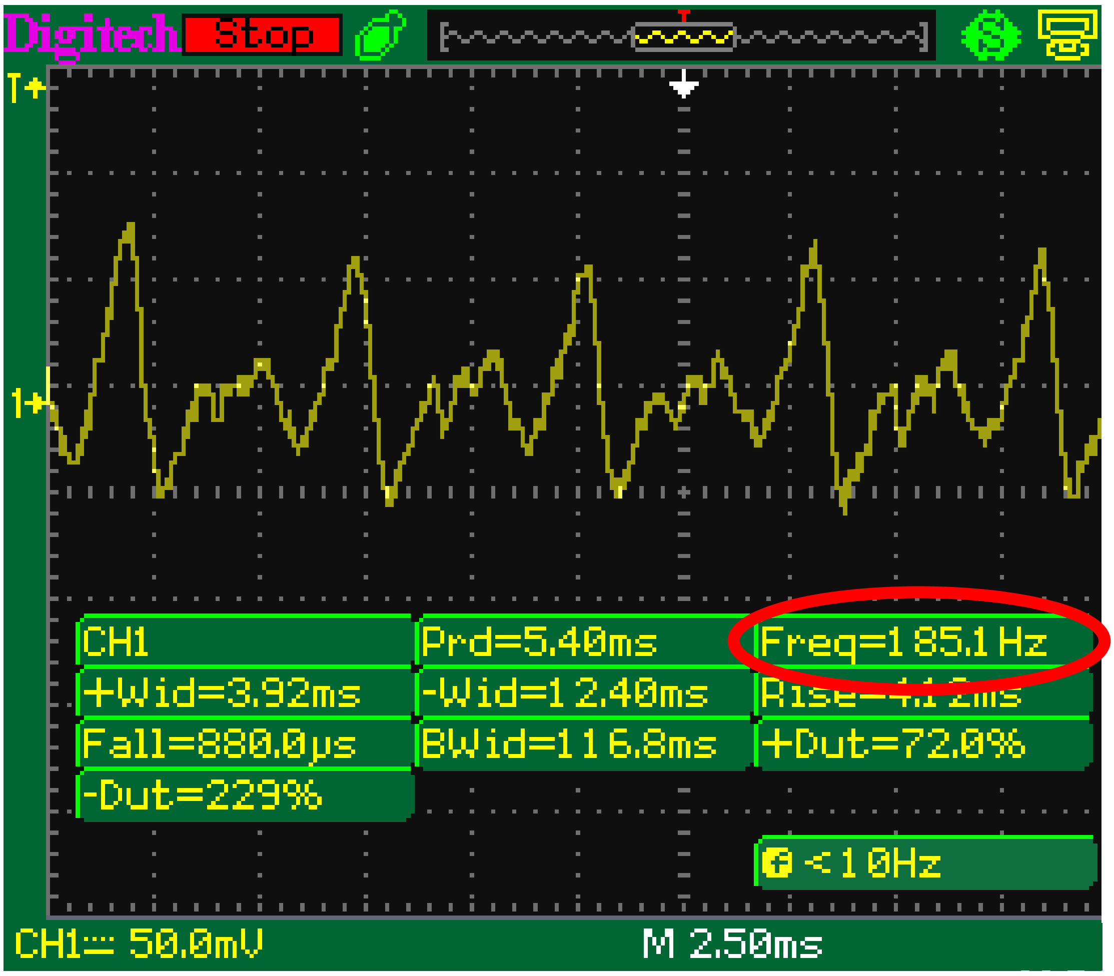
The wavetable object was stored within the flash memory of the Teensy Audio Adaptor (PJRC, 2019a). The USB cable was unplugged from the PC and supplied powered only to the Teensy from an AC outlet, therefore removing dependency to the Arduino IDE and any computer applications; the audio output remained unchanged.

By modulating the frequency, different notes were achieved, however, the output was not constrained to the frequency of musical notes and usually produced frequencies in-between notes. To ensure an exact musical note was produced, a lookup table was constructed such that the 10-bit analog input was converted to one of the 31 standard notes of a trumpet.

The files contained within the `MidiSynthLarge` example provided the audio array of a trumpet with five note samples included titled `trumpet_samples`.

Through replication of the naming conventions and references required to select `Flute_100kbyte` as the wavetable instrument, `trumpet` was assigned the designated instrument. To do so, the `trumpet_samples` header and cpp files were obtained from the `MidiSynthLarge` example and inserted into the `simpleWavetable` folder. Without further modification, the gain and frequency of the trumpet sample were modulated, confirming the functions could be applied to any compatible data array.

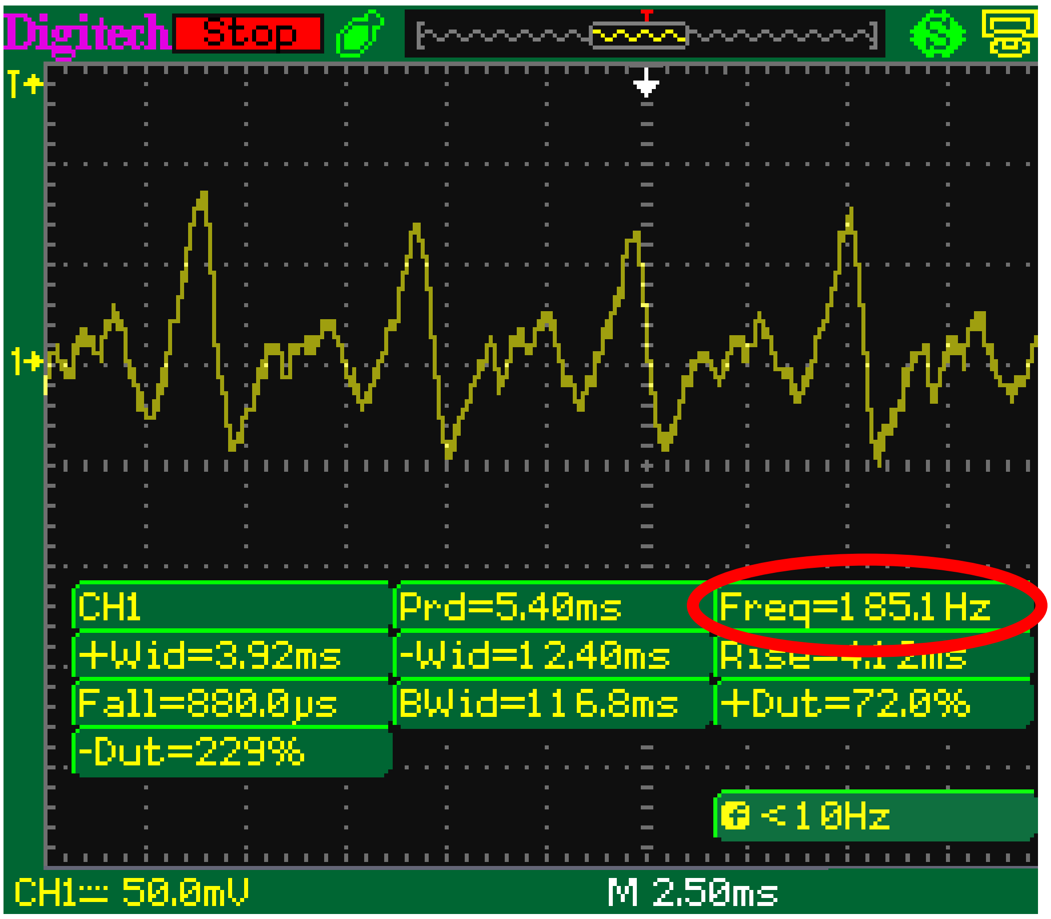
To confirm the frequency passed through the `wavetable.playFrequency()` function was output correctly, the line out pin of the Audio Adapter Board was connected to an oscilloscope. A frequency of 185.00Hz was passed to represent the note F#3 (Inspired Acoustics, 2019). As denoted by the red ellipse in **Figure 37**, a frequency of 185.1Hz was recorded, therefore assumed equivalent to the specified frequency due to a negligible error of 0.05%.



```
CH1              Prd=5.40ms       Freq=185.1Hz
+Wid=3.92ms      -Wid=12.40ms     Rise=4.12ms
Fall=880.0μs     BWid=116.8ms     +Dut=72.0%
-Dut=229%
                                  <10Hz
CH1 50.0mV                 M 2.50ms
```

***Figure 37**: Oscilloscope wavetable.playFrequency(185) Output*

The lookup table produced the desired output by correlating the 10-bit analog value to an integer between 0 and 30 representing a note and finally to a frequency. MIDI communication typically required an index number and not a precise frequency, for simplicity, additional wavetable functionality was investigated. Through analysis of the `AudioSynthWavetable.cpp` file, the function `playNote()` was discovered which accepted an integer. Although unspecified, it was assumed that the integer represented the MIDI note number. To verify, the MIDI note number representation of the note F#3, found to be 54, (Inspired Acoustics, 2019) was passed through the `wavetable.playNote()` function. As denoted by the red ellipse in **Figure 38** a frequency of 185.1Hz was again recorded, therefore confirming the `wavetable.playNote()` function expected to receive the MIDI Note Number.



```
CH1              Prd=5.40ms       Freq=185.1Hz
+Wid=3.92ms      -Wid=12.40ms     Rise=4.12ms
Fall=880.0µs     BWid=116.8ms     +Dut=72.0%
-Dut=229%
                                  <10Hz
CH1 50.0mV                  M 2.50ms
```

Figure 38: *Oscilloscope wavetable.playNote (54) Output*

In previous examples, multiple samples of an instrument were contained within the cpp file, enabling the entire range of notes to be extrapolated. The minimum number of samples required to extrapolate additional notes was undefined; the capabilities of the wavetable object were further tested through assigning an instrument with a single sample to the wavetable.

The `MidiSynthLarge` example contained 24 unique audio sounds with varying sample numbers. The French horn contained one sample and though assignment as a wavetable instrument, was tested through the range of the 31 notes applied to the trumpet. The test was successful as each note was derived from the original sample, therefore confirming the wavetable was capable of mapping a range of notes from a single note sample.

In the event that the user desired an instrument which was not available within the examples, methods to input external instruments and sounds were investigated. To play MIDI files, SoundFont (sf2) files were utilised; the Teensy SoundFont Decoder accepts and converts SoundFont files into data arrays for interpretation by the Teensy Audio Adapter (Mellmer, 2017).

Trumpet samples were located from a `Synth Brass 2` sf2 file obtained from Musical Artifacts (Strix SoundFont Team, 2018). The trumpet samples were successfully converted from an sf2 file to a header file and cpp file, and imported into the Arduino sketch (Mellmer, 2017). Upon assignment of the imported trumpet to the wavetable instrument, compilation errors presented. The Teensy SoundFont decoder was released in 2017; it is assumed that current Teensy libraries have been updated such that compatibility with the decoder is not directly achievable. The following modifications made to the header file and cpp file, shown in **Table 5** and **Table 6**, respectively, enabled the trumpet object to be assigned as a wavetable instrument. Although a trumpet file within a Teensy example could be utilised, the process represents the ability for the user to output any sound with a corresponding sf2 file, including non-traditional instruments, for future development and additional customisation.

***Table 5**: Header File Modifications*

Original	Modified
#include <AudioStream.h>	<Audio.h>
#include <AudioSynthWavetable.h>	(removed)
sample_data TRUMPET_samples[12]	AudioSynthWavetable:: sample_data TRUMPET_samples[12]
instrument_data TRUMPET = {12, TRUMPET_ranges, TRUMPET_samples };	AudioSynthWavetable:: instrument_data TRUMPET = {12, TRUMPET_ranges, TRUMPET_samples };

***Table 6**: Cpp File Modifications*

Original	Modified
#include <AudioStream.h>	<Audio.h>
#include <AudioSynthWavetable.h>	(removed)
sample_data TRUMPET_samples[12] = {	AudioSynthWavetable:: sample_data TRUMPET_samples[12] = {
SAMPLES_PER_MSEC	AudioSynthWavetable:: SAMPLES_PER_MSEC
LFO_PERIOD	AudioSynthWavetable:: LFO_PERIOD
UNITY_GAIN	AudioSynthWavetable:: UNITY_GAIN
DECIBEL_SHIFT	WAVETABLE_DECIBEL_SHIFT
CENTS_SHIFT	WAVETABLE_CENTS_SHIFT

The Teensy Audio Board Adaptor Board enabled access to the Audio System Design Tool, an online block-based interface for audio system configuration. The Audio System Design Tool contains blocks from the categories: input, output, mixer, play, record, synth, effect, filter, analyse and control (PJRC, 2019b).

One wavetable block enables the output of one note, of a particular instrument. To enable up to four notes, or instruments, to be output simultaneously, four wavetable objects were included as inputs to the audio system. The four wavetable outputs were then connected to a mixer block; the mixer output provided a single stereo line that combined the four wavetables. The Teensy and Audio Board communicate through the I^2S protocol, utilised specifically to connect digital audio devices through the transfer of audio data (PJRC, 2019b). The mixer output was therefore connected to the i2s block, the output of the audio system.

The SGTL500 is a low-power stereo codec capable of controlling of the audio board. The SGTL5000 block does not require inputs or outputs, however, configures I^2S communications and was required to be included in the audio system.

Upon completion of the Audio System (**Figure 39**), the code was generated online, in the Audio System Design Tool, for direct inclusion into the Teensy Code, detailed in Appendix E.

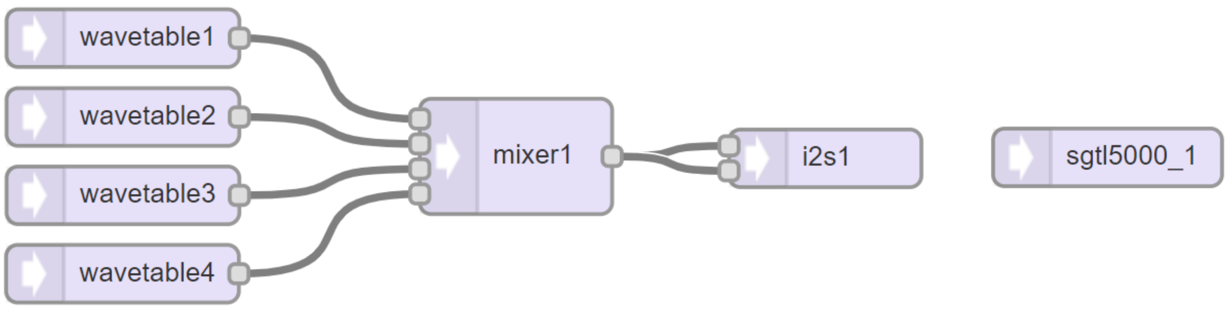


***Figure 39**: Audio System Design Tool*

4.1.5 Cost Evaluation

For the proposed assistive electronic trumpet to substitute current incentive spirometers, the price must be considered. Australian incentive spirometers have been found to retail between AUD$8 (Opentip, 2019) and AUD$40 (Stark Medical Pty Ltd, 2019), on average. The individual component cost required to build the electronic trumpet totalled AUD$122.53 (**Table 7**). It is acknowledged that the assembly labour and a casing solution have not been factored into this analysis and the total prototype price will exceed AUD$122.53.

***Table 7**: Cost Evaluation of Final Solution*

Specification	**Teensy 3.6 (Little Bird Electronics, 2019)**	**Audio Adaptor Board (Core Electronics, 2019a)**	**HC-06 Bluetooth Module (Core Electronics, 2019b)**	**MPXV7002DP Pressure Sensor (Mouser Electronics, 2019)**	**Pushbutton (DigiKey Electronics, 2019b)**	**Jumper wire Bundle (Element14, 2019)**
Quantity	1	1	1	1	3	1
Cost (AUD)	$45.00	$30.30	$12.00	$25.84	$1.83	$7.56
Total (AUD)	$122.53					

The 67% - 93% cost increase is expected as the prototype contains multiple complex electronic components. Additionally, the proposed device is designed for longevity and continual use following recovery, conversely, modern incentive spirometers are designed to be cost-effective and disposed of immediately following recovery (Eltorai et al., 2018).

4.1.6 System Block Diagram

The block diagram (**Figure 40**) denotes the pure inputs (green), output (orange), processors (blue) and element which acts as both an input and an output (black). Following a breath from the user into the mouthpiece, the differential pressure sensor provides the Teensy with a digital representation of the breath strength. User input from the app is provided to the Teensy via the HC-06, similarly, visual information from the Teensy is output by the app via the HC-06. Audio information is then provided from the Teensy to a loudspeaker from the Teensy auxiliary output.

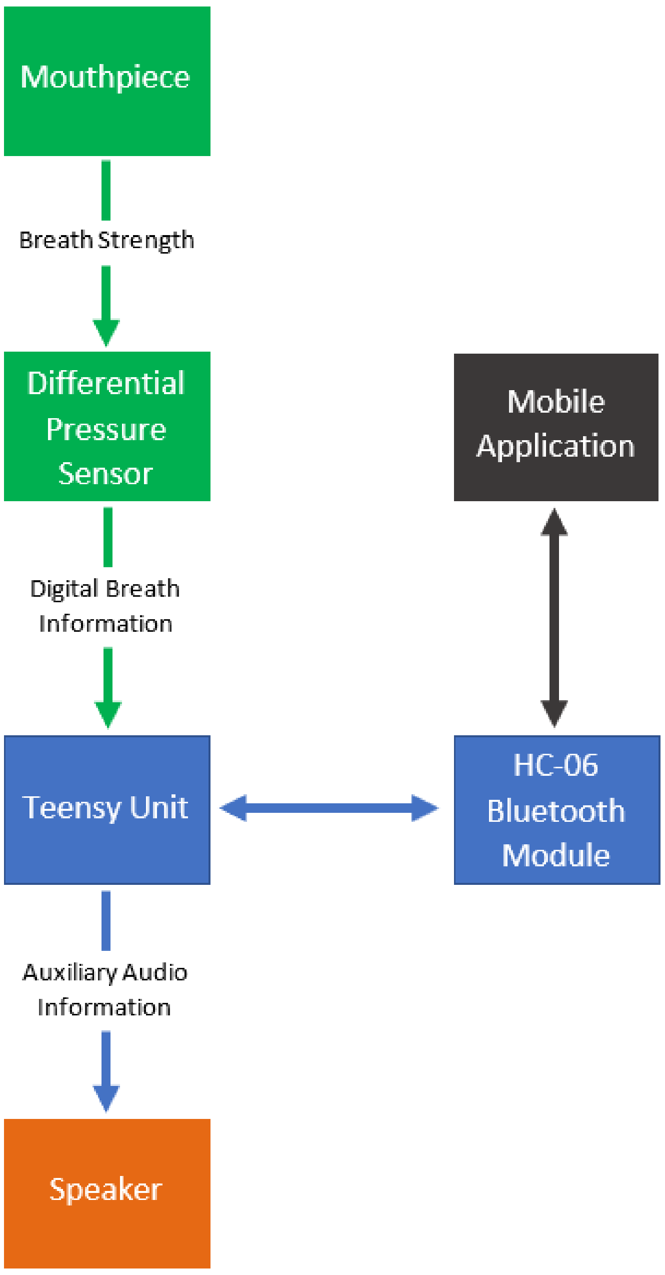


***Figure 40**: System Block Diagram*

4.2 Software Development

4.2.1 Music Development

4.2.1.1 Chords

The ability to output single notes had been achieved, however, additional musical expression was facilitated through the more complex note combination, chords.

A chord is officially defined as two or more notes played simultaneously, however; most chords consist of at least three notes (MacFarlane, 2016). The naming convention of a chord provides two pieces of information; the root note of the chord and the chord type. For example, a C Major identifies the root note of the chord to be a C, while the chord type belongs to the Major type (MacFarlane, 2016).

The root note determines the note the chord is built upon, while the intervals between each note within the chord determine the characteristics and identifiable sounds produced, the chord type. Each chord type can be attributed to particular feelings or emotions. The emotions include happiness, sadness, strength or suspense, elicited from major (maj), minor (min), dominant seventh (7) or augmented (aug) chords, respectively, described for each chord type (**Table 8**) (MacFarlane, 2016).

Each chord type can be built based on the interval between each note within the chord and root note (MacFarlane, 2016). To program each chord, the interval pattern described in **Table 8** was applied to a function within the Arduino code. For example, to produce a major chord, an array with three elements was initialised, the first, second and third elements were assigned the numerical value of the root note, the root note + 4 and the root note + 7. A second function was then called which played the three-note array simultaneously with three consecutive `usbMIDI.sendNoteOn` commands, followed by a delay dependant on the duration specified, which concluded the chord with three `usbMIDI.sendNoteOff` commands. A similar secondary function was designed for the dominant seventh and suspended chords, which contain four notes as opposed to three.

***Table 8**: Chord Information (MacFarlane, 2016)*

Chord Type	Interval Pattern	Emotion Association
Major (maj)	0 – 4 – 7	Happy and Simple
Minor (min)	0 – 3 – 7	Sad and Serious
Dominant Seventh (7)	0 – 4 – 7 – 10	Strong and Relentless
Suspended Second (sus2)	0 – 2 – 7	Bright and Nervous
Suspended Fourth (sus4)	0 – 5 – 7	Bright and Nervous
Major Seventh (maj7)	0 – 4 – 7 – 11	Thoughtful and Soft
Minor Seventh (min7)	0 – 3 – 7 – 10	Moody and Contemplative
Augmented (aug)	0 – 4 – 8	Anxious and Suspenseful
Diminished (dim)	0 – 3 – 6	Tense and Unpleasant

The chords programmed into the electronic trumpet were based on the chords commonly utilised by professional musicians and are therefore present in popular music. Based on an audio chord recognition analysis of 1100 unique songs obtained from the Billboard 'Hot 100' chart in the United States between 1958 and 1991, the most popular chord types were realised (Burgoyne and Wild, 2011). The major (maj), minor (min), seventh (7), minor seventh (min7) and major seventh (maj7) chords were the five most popular chord types, with a prevalence of approximately 52%, 13%, 11%, 8%, and 5%, respectively (Burgoyne and Wild, 2011). All subsequent chords displayed an occurrence of less than 5% and with the omission of chords with less than three notes or more than four notes (Burgoyne and Wild, 2011), the suspended fourth (sus4) and suspended second (sus2) were the most recognisable chord types and were also included.

Extended chords are not listed in **Table 8** and will not be included in the electronic trumpet, however, an extended chord consists of notes beyond the octave of the root note, of which there are many variations including ninth, eleventh and thirteenth chords (MacFarlane, 2016).

4.2.1.2 Scales

In addition to playing chords during music creation, it was desired to produce a musical output during ISy. As the task required the user to sustain a consistent flow, it was initially thought that a single sustained note would be appropriate. However, the objective of the device was to encourage use and increase user compliance, meaning that strategies to excite and entice the user to maintain therapy engagement were pursued. Through investigating the addictive factor of electronic gambling machines, it was discovered that receiving visual and auditory cues associated with previous wins increased the desire to continue gambling (Cherkasova et al., 2018). It was therefore assumed that providing musical variance through positive reinforcement of correctly performed therapy could improve focus, attention and motivation. The musical variance was enabled through the output of a music scale, which played a series of ascending and descending notes in sequence, usually with equivalence.

The Australian Music Examination Board (AMEB) is the primary musical accreditation and certification body in Australia utilised to assess students from a preliminary to diploma level. To ensure the music theory introduced within the program was recognised to be useful in musical skill development and training, the AMEB examination syllabus was analysed (AMEB, 2018). AMEB begins at Preliminary Level, followed by Grade 1 and continues to Grade 8, after which a Certificate of Performance can be achieved, followed by associates and licentiates (AMEB, 2019). It was discovered that major scales and natural minor scales were introduced in Grade 1, harmonic minor scales in Grade 2 and melodic minor scales in Grade 5 (AMEB, 2018); the four scales were then enabled for playback within the final application.

Typically played from low to high, a musical scale consists of an organised sequence of tones and semitones usually arranged as octave repeating notes (Simplifying Theory, 2014). As the most simple and fundamental scales, a function was written to output, the two natural, or diatonic scales, the major and minor scale. A major scale consists of a tone, tone semitone, tone, tone, tone, semitone, with pattern 0 – 2 – 4 – 5 – 7 – 9 – 11 – 12, while the natural minor scale consists of the pattern tone, semitone, tone, tone, semitone, tone, tone (Simplifying Theory, 2014).

Similarly to the chord function, a scale function was designed to turn on and off multiple MIDI note commands, however, a delay was set in between the activation of each note, therefore staggering the output of the scale to replicate incremental rather than simultaneous play.

In addition to the two natural scales, the harmonic minor and melodic minor were also programmed into the scale function. The harmonic minor scale is equivalent to the natural minor; however, the seventh step is raised by half a step. The melodic minor raises both the sixth and seventh note by half a step while ascending unlike the previous scales, however, the melodic minor scale descends in the natural minor scale (Hollis, 2017). All four scale intervals are described in **Table 9**.

Table 9*: Scale Information (Simplifying Theory, 2014; Hollis, 2017)*

Scale Type	Interval Pattern
Major Ascending	0 – 2 – 4 – 5 – 7 – 9 – 11 – 12
Major Descending	12 – 11 – 9 – 7 – 5 – 4 – 2 – 0
Natural Minor Ascending	0 – 2 – 3 – 5 – 7 – 8 – 10 – 12
Natural Minor Descending	12 – 10 – 8 – 7 – 5 – 3 – 2 – 0
Harmonic Minor Ascending	0 – 2 – 3 – 5 – 7 – 8 – 11 – 12
Harmonic Minor Descending	12 – 11 – 8 – 7 – 5 – 3 – 2 – 0
Melodic Minor Ascending	0 – 2 – 3 – 5 – 7 – 9 – 11 – 12
Melodic Minor Descending	12 – 10 – 8 – 7 – 5 – 3 – 2 – 0

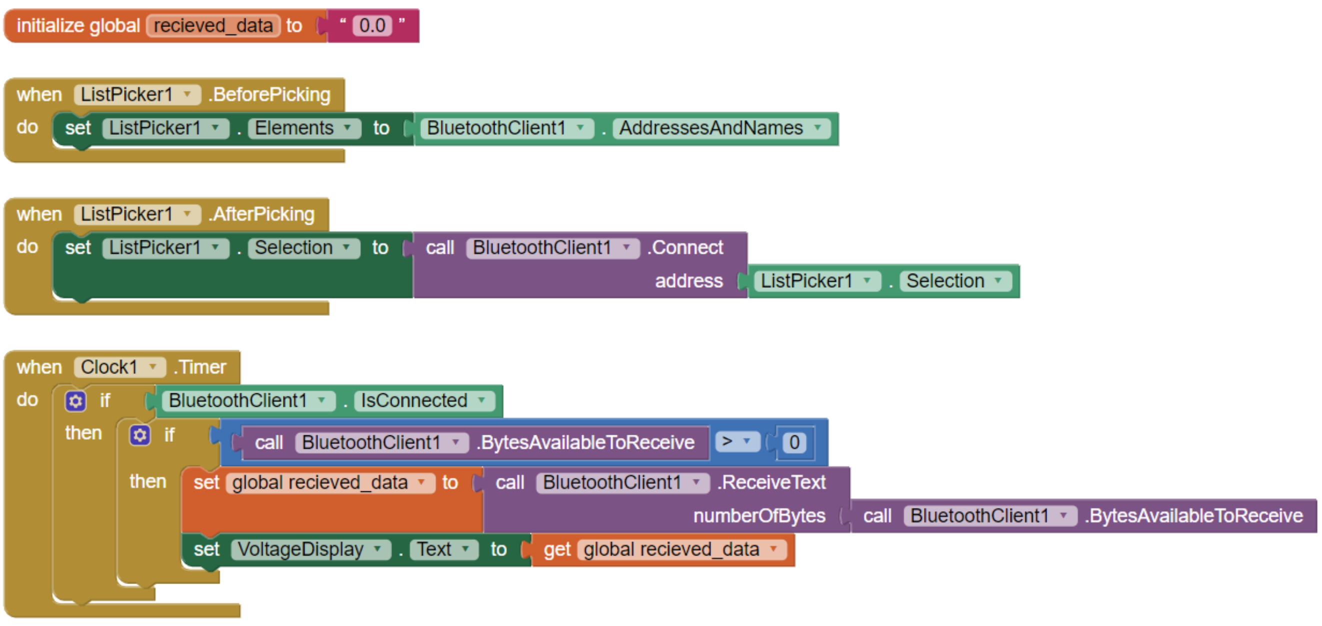
4.2.2 App Development

To provide additional feedback and engagement mechanisms, an app was designed in MIT App Inventor 2, an online platform that facilitates android app development through visual block-based programming (MIT App Inventor, 2019). Trimodal feedback was recognised to facilitate optimal learning and music creation (Frid, 2019), therefore visual and haptic feedback were incorporated into the different app modes.

Three different app modes, or interfaces, were created, each enabling the user to select the preferred mode of breath detection, either inhalation or expiration, and whether haptic feedback was desirable. Two of the three app interfaces were designed to provide the benefits of either FIS or VIS, depending on the recommendations prescribed by the physician and the preferences of the user. The final app interface was designated to music creation, controlled with FIS techniques.

To enable communication between the app and the Teensy 3.6, a low-cost Bluetooth Module was incorporated into the project. The HC-06 was selected due to simplicity in connectivity through serial communication with the designated microcontroller, and the ability to define the baud rate through ATtention (AT) commands (Core Electronics, 2019b).

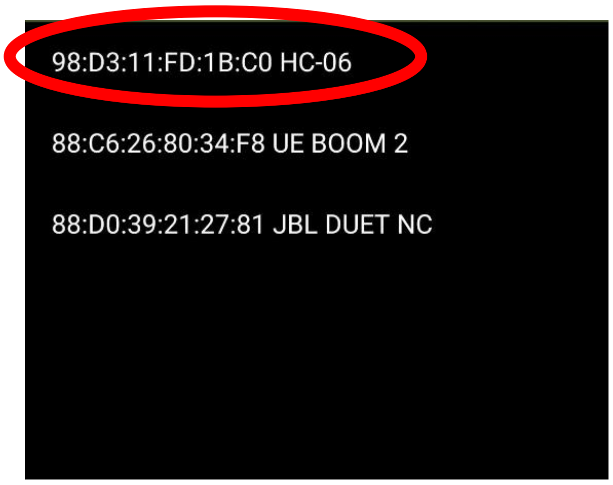
Due to the large availability of online support for the Arduino Uno compared to the Teensy 3.6, communication was first achieved between the Arduino and the app, prior to interfacing with the Teensy 3.6. To confirm the Bluetooth module did not possess any manufacturing faults, the output following trial inputs were confirmed against known outputs provided by Anghel, before formal testing was commenced. Anghel detailed how to connect the HC-06 to the Arduino, which assigned the HC-06 transmitter and receiver pins to pin 0 and pin 1 of the Arduino, respectively (Anghel, 2017). A potentiometer was then connected to the Arduino, the analog value was scaled between zero and five to represent the voltage and was printed serially. As pin 0 and pin 1 of the Arduino directly stream data to the serial monitor, the data was transmitted via the HC-06 (Anghel, 2017). For the app to receive data, a `ListPicker` block was placed on the app screen (**Figure 41**), the `ListPicker` is set to the Bluetooth Client.



```
initialize global recieved_data to " 0.0 "

when ListPicker1 .BeforePicking
do  set ListPicker1 . Elements to  BluetoothClient1 . AddressesAndNames

when ListPicker1 .AfterPicking
do  set ListPicker1 . Selection to  call BluetoothClient1 .Connect
                                                        address  ListPicker1 . Selection

when Clock1 .Timer
do  if  BluetoothClient1 . IsConnected
    then  if  call BluetoothClient1 .BytesAvailableToReceive > 0
          then  set global recieved_data to  call BluetoothClient1 .ReceiveText
                                                        numberOfBytes  call BluetoothClient1 .BytesAvailableToReceive
                set VoltageDisplay . Text to  get global recieved_data
```

Figure 41: Receive and Display Bluetooth Data Code - Test

The `ListPicker` enabled selection of the intended Bluetooth device (**Figure 42**), all paired Bluetooth devices were displayed, and the HC-06, circled in red, was selected.



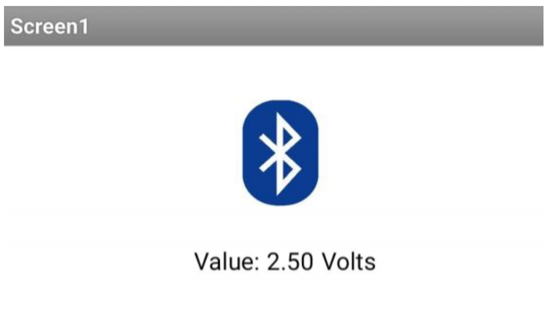
98:D3:11:FD:1B:C0 HC-06

88:C6:26:80:34:F8 UE BOOM 2

88:D0:39:21:27:81 JBL DUET NC

Figure 42: Bluetooth List Selection

To open the list displayed in **Figure 42**, the Bluetooth symbol (**Figure 43**) is pressed. The app was then confirmed to produce a value of 2.50V as the potentiometer was positioned in the centre of its rotational range (**Figure 43**).



Value: 2.50 Volts

***Figure 43**: Bluetooth Voltage Output Test*

A baud rate of 9600 was set by default for the HC-06, however, for communication with the MIDI software, a baud rate of 115200 was required. To modify the settings of the HC-06, serial communication was required to access the AT commands. Prior to reconfiguring parameters, it was required to confirm communication between the HC-06 and the serial monitor of the Arduino by typing "OK" into the serial monitor receiving "OK" as a reply (Rahman, 2018). Based on the original pinout configuration, a reply could not be generated, attributed to pin 0 and pin 1 exclusively being required for serial communication (Arduino, 2019c).

A second tutorial was located which presented an alternative connection method to any two digital pins, therefore removing the requirement to disconnect the module for each new code upload and enabling serial communication. The method included the `Software Serial` Library, built to replicate the hardware serial capabilities of pin 0 and pin 1, on any digital pins (Chen, 2019). To change the baud rate from 9600 to 115200, the command `AT+BAUD8` was required (Rahman, 2018). The command was applied and confirmed as successful after the serial monitor baud rate was set to 115200 and the "OK" reply was received following the "AT" test command.

The receiver of the HC-06 operates at 3.3V, and as the Arduino outputs 5V, a voltage divider circuit was implemented (Chen, 2019). The following equation demonstrates the appropriateness of selecting a $1k\Omega$ and $2k\Omega$ resistor:

$$V_{out} = \frac{V_1 R_2}{(R_1 + R_2)} = \frac{5V \times 2k\Omega}{1\text{k}\Omega + 2k\Omega} = \frac{5V \times 2k\Omega}{(1+2)k\Omega}$$

$$V_{out} = \frac{10V}{3} \approx 3.33V \approx 3.30V$$

With the new baud rate and pinout configuration, the results shown in **Figure 43** were replicated and the HC-06 was therefore confirmed to successfully transmit information from the Arduino at the required baud rate of 115200. The information could then be received and displayed on a mobile device through an MIT App Inventor program.

Through MIT App Inventor, the received data can be displayed and manipulated as either text-based or numerical-based information.

For the Arduino program to receive information from the app, a string was created based on the summation of each individual character, from the `Serial.read()` function, which reads the first byte of incoming serial data (Arduino, 2019d).

The first integration into the application was to display the name of the note, which was stored in an array as a string. The desired variable was sent, and the label assigned accordingly, as observed in the tutorial.

To establish a connection between the HC-06 and the Teensy 3.6, Pin 33 and 34 were defined as `TX5` and `RX5`, which offered serial communication through the declaration of `Serial5`, while `Serial` was maintained to refer to the Arduino Serial Monitor. For communication to occur, the receiver and transmitter of the Teensy were connected to the transmitter and receiver of the HC-06, respectively. Unlike the Arduino, the Teensy 3.6 operated at 3.3V, not 5V, therefore removing the requirement to assemble a voltage divider circuit for the receiver of the HC-06 (PJRC, 2019c).

All connections made on the breadboard from the HC-06 and the three pushbuttons to the Teensy 3.6 are shown in **Figure 44**. As the Audio Adaptor Board was mounted onto the Teensy 3.6, the board was made partially transparent, to ensure all connections remain visible. The physical prototype is shown in **Figure 45**, as the differential pressure sensor could not be inserted directly onto a breadboard, the required pins were soldered to an external board and attached.

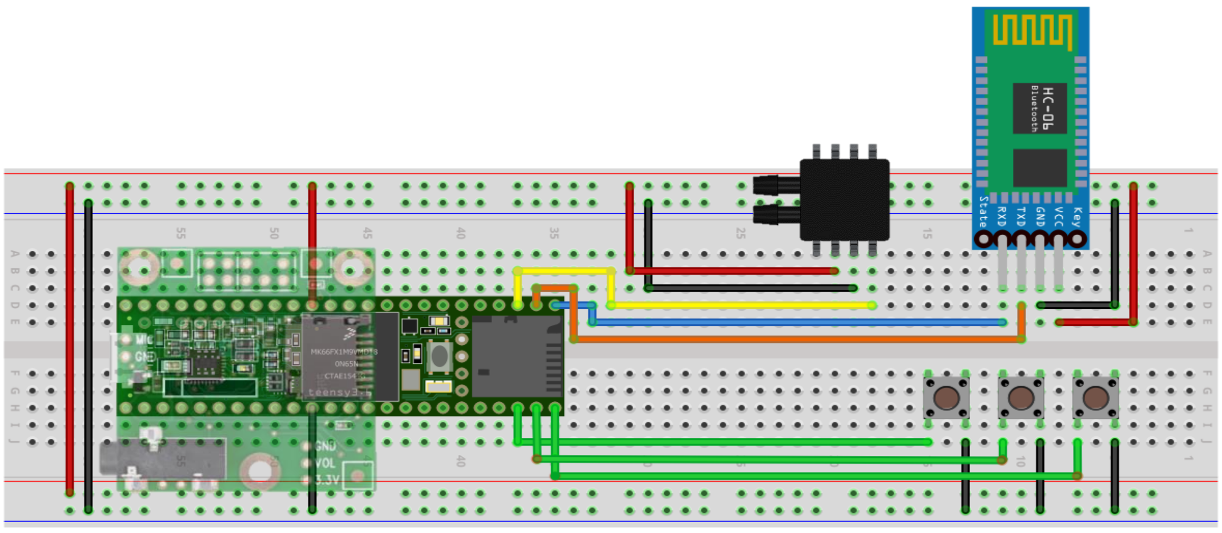


***Figure 44**: HC-06 to Teensy Connections*

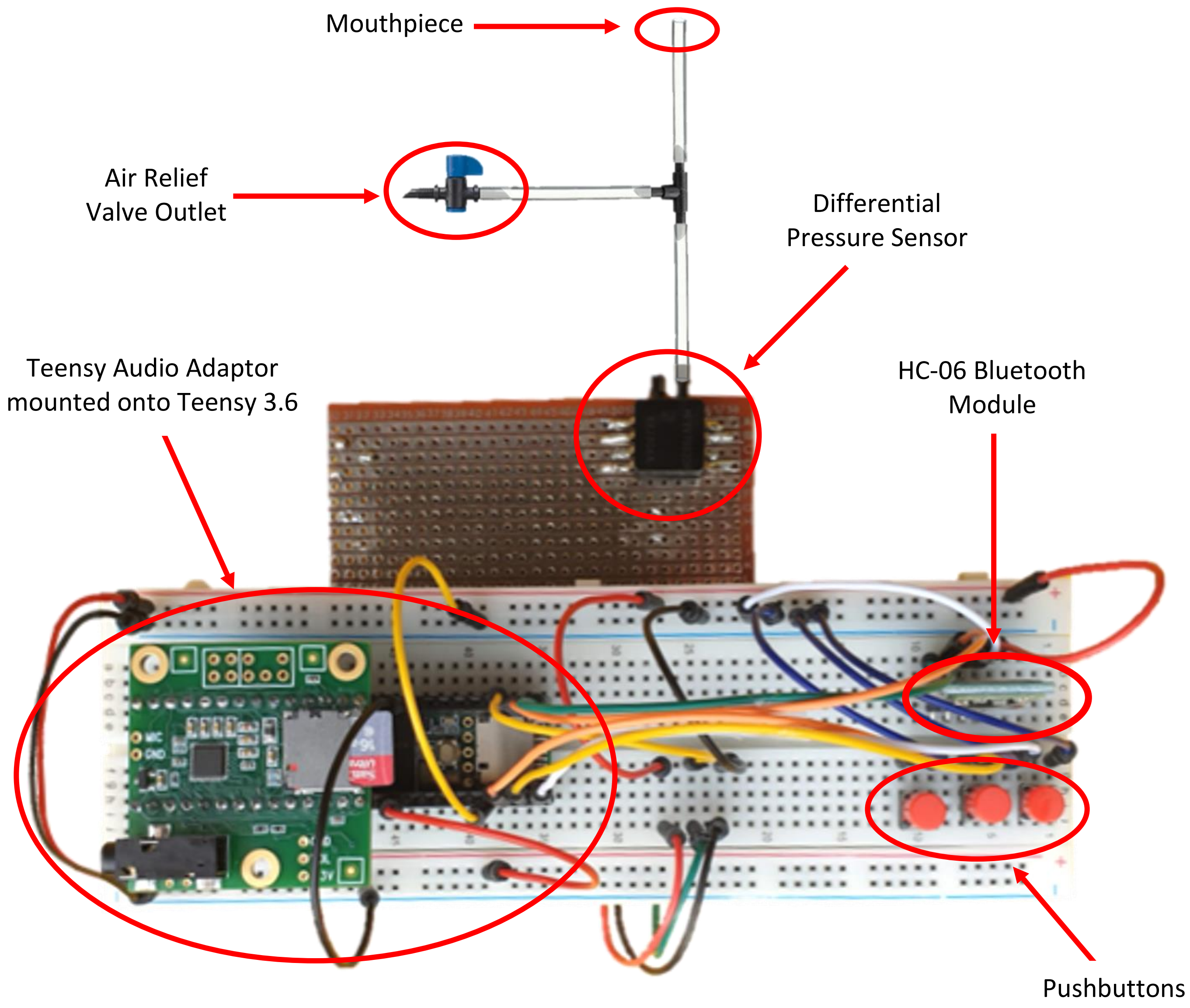

Mouthpiece
Air Relief Valve Outlet
Differential Pressure Sensor
Teensy Audio Adaptor mounted onto Teensy 3.6
HC-06 Bluetooth Module
Pushbuttons

***Figure 45**: HC-06 to Teensy Connections – Physical Prototype*

4.2.2.1 Flow-Orientated Incentive Spirometry Mode

The first interface was based on FIS; music is created based on the flow, or strength, of the breath. The user selects the desired note to be played, which determines the position of a solid red line on the main graphic. To play a note, the user either inhales or expires into the mouthpiece, depending on the selection, the relative position of the note will then be displayed against the desired note in scatter plot form with lines and markers. An increase in breath strength will raise the pitch of the output note; the objective is to learn breath control such that the output will overlay the desired note.

If haptic feedback is enabled, the mobile phone will vibrate upon reaching the desired note or successfully matching the output note to the desired note.

As can be observed in the top-left corner of **Figure 46**, the Bluetooth module icon facilitates the Bluetooth connection between the app and the HC-06. By pressing the Bluetooth icon, the screen displayed in **Figure 42** is generated, enabling the selection of the HC-06 module. The two black buttons in the top-right corner of **Figure 46** enable the app to transfer to either the Music Creation Mode or Volume-Orientated Mode. It is noted that the placement of the Bluetooth icon, and the two toggle buttons which link to the two additional screens, remained consistent between all three screens.

From the bottom of **Figure 46**, it can be observed that the direction of the breath, either inhalation or exhalation, can be selected; the current mode is displayed in green. The vibrate toggle-button is located in the bottom-right corner, and in **Figure 46** is in the `off` position, as the circular component of the button is in the leftmost position, and the button background is grey. In the `on` position, the button background would be yellow, and the circular component would be in the rightmost position.

To select the desired note, the horizontal slider could be moved to one of the 31 positions, or one of the two arrows can be pressed. By pressing either the left or right arrow, the note was decremented or incremented by one note, respectively. In **Figure 46**, the `Desired Note` display is fixed at 31 and is not linked to the output of the slider or arrows. Both the `Current Note` display and the graph are dependent on the value obtained from the pressure sensor.

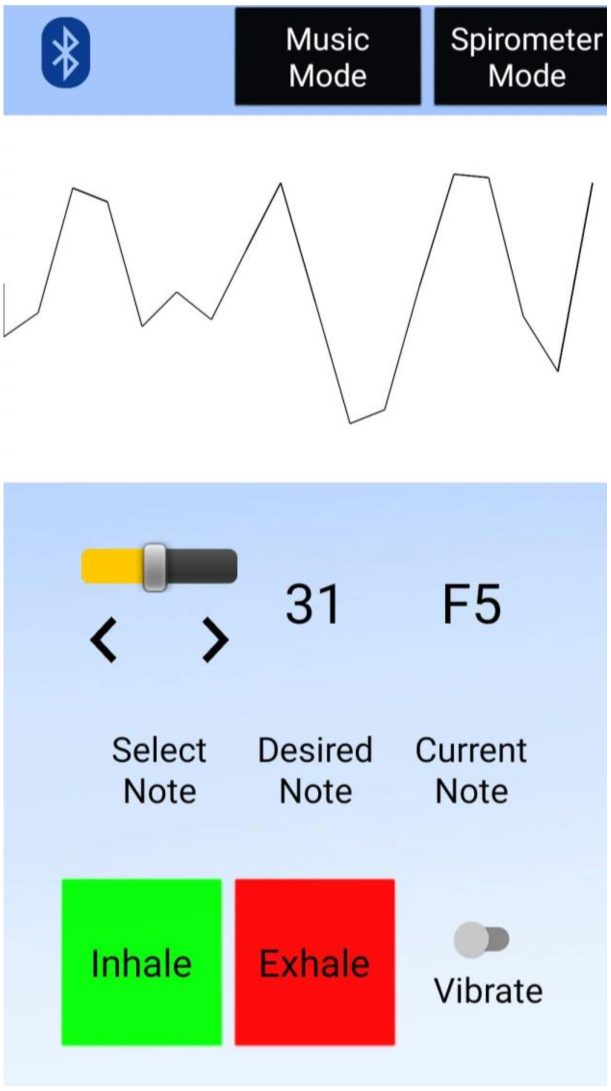


***Figure 46**: Flow-Orientated Initial Interface*

To construct the graph shown in **Figure 46**, a point was assigned to one of 31 positions, dependent on the current note, and drawn on a canvas titled `Graph_Canvas`. Following each clock cycle, a horizontal position increment was applied, and the previous and current points were connected with a line until the width of the canvas was reached. The canvas was then cleared with the `Graph_Canvas.Clear` command and the horizontal position was reset.

The minimum pixel position in both the x-direction and y-direction were found to be 45 and 198, respectively. Conversely, the maximum pixel position in the x-direction was found to 335, and 37 in the y-direction. The coordinates of the canvas have been indicated in red in **Figure 47**, depicting the total usable area.

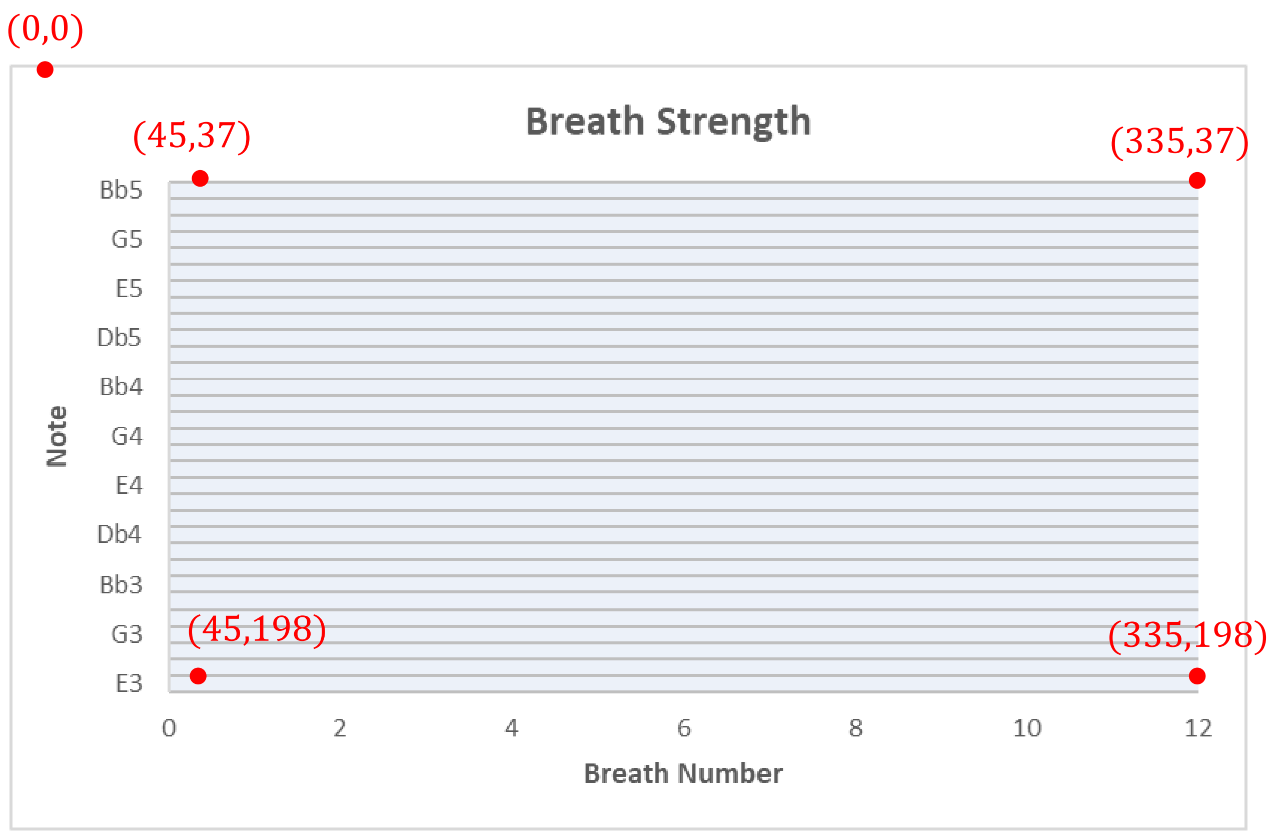


***Figure 47**: Flow-Oriented Graph Background*

The next development stage included a solid line to represent the desired note, to enable comparison between the user's objective and actual performance. For a clear comparison, the desired and observed lines were coloured red and blue, respectively, with the `set Graph_Canvas.PaintColor` command. The width of both lines was also increased from a value of `1` to `3` with the `set Graph_Canvas.LineWidth` command.

The plot was confirmed to operate as expected if the `Desired Note` remained constant, as the canvas was only cleared after 12 breath values were plotted. During testing, it was discovered that if the `Desired Note` was modified before the canvas was cleared, multiple instances of the red indicator would remain, as depicted in **Figure 48**.

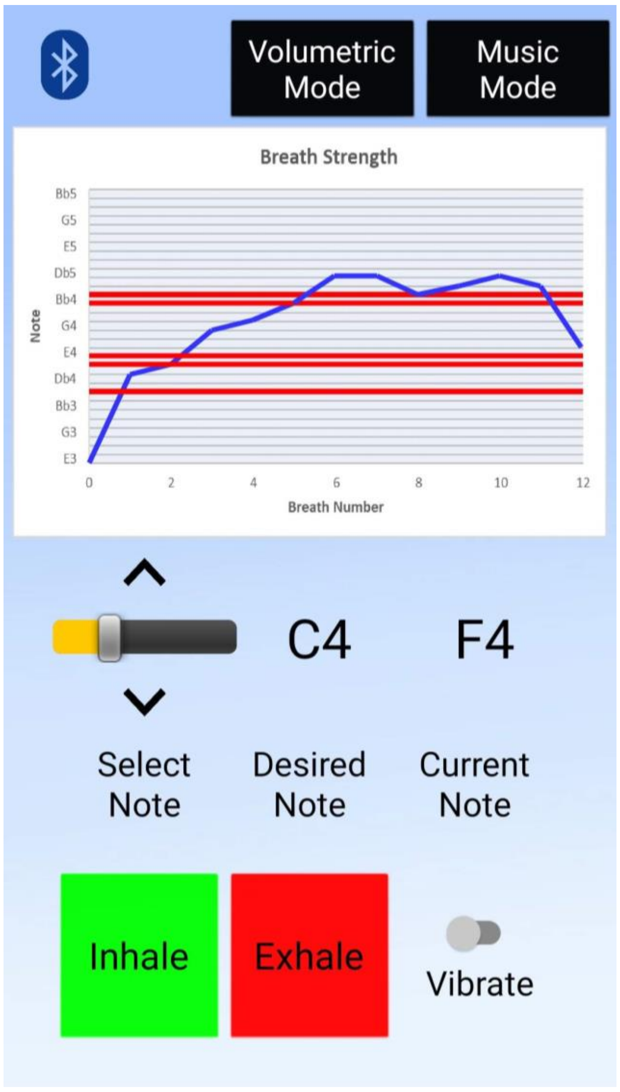


Figure 48: *Flow-Orientated Line Duplication*

For the `Desired Note` line to represent only the most current note, it was required to erase the position of the previous `Desired Note` status. Firstly, a method of detecting a change in the `Desired Note` was required, achieved through a secondary variable that recorded the previous `Desired Note` value and compared it with the current `Desired Note` value. As the graph contained information regarding the recorded breath strength, the canvas could not be cleared upon the detection of a new desired note position.

To simulate the effect of clearing the canvas while maintaining critical breath information, another line matched to the background was drawn over the previous `Desired Note` value.

The RGB index of the graph was found to be (236, 240, 249), and through the `make colour` command, a line of the background colour was redrawn on top of the previous line position. Two unintended effects resulted, the dark gridline of the background graph was removed, leaving a footprint of the previous `Desired Note` position (**Figure 49**). Secondly, the previously plotted breath information was partially erasing leaving horizontal rows in the diagonal lines.

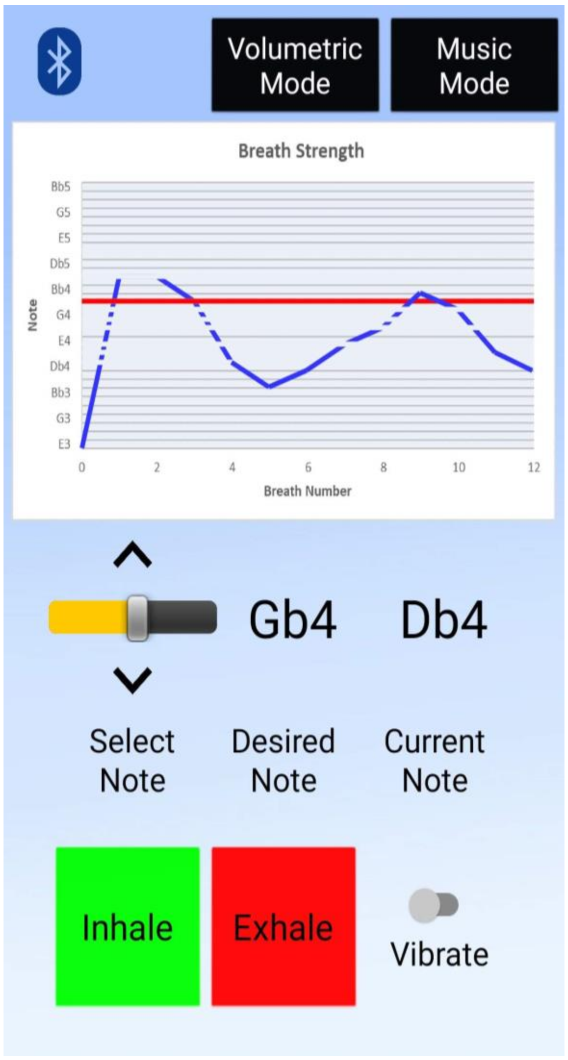
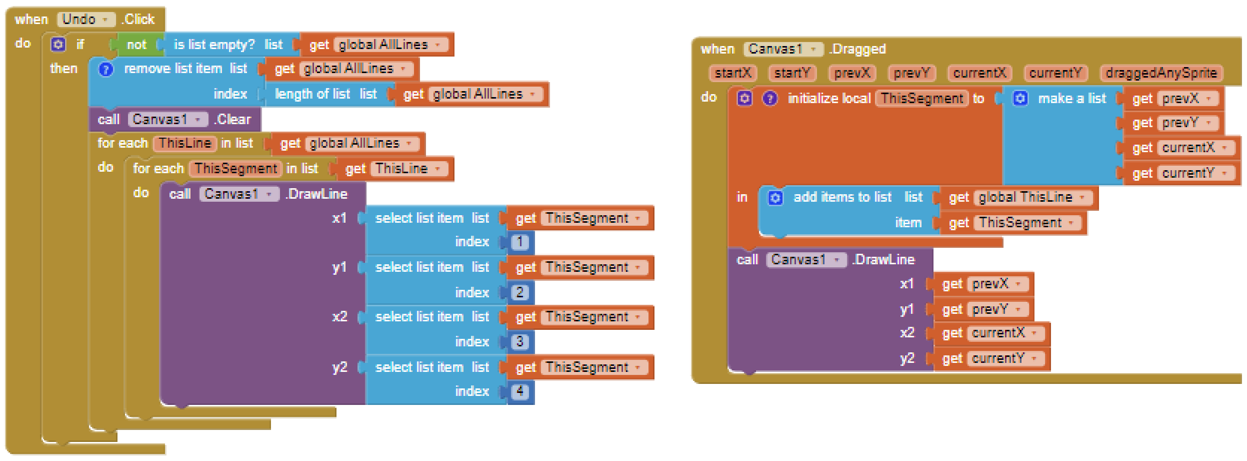


***Figure 49**: Flow-Orientated Line Incorrectly Removed*

It was therefore understood that for the `Desired Note` line to be modified without negatively impacting the appearance of the graph, the breath information could not be modified.

Through online investigation, an app designed for drawing sketches was posted, capable of an `undo` feature. To effectively remove the latest input to the sketch, a `list` was created and updated following each addition to the design (Ferguson, 2015).

The `list` stored the x1, y1, x2 and y2 coordinates of all lines drawn in four list indexes. Upon selection of the `undo` tool, the most recent line information was removed from the list, the canvas was cleared and then redrawn (Ferguson, 2015). **Figure 50** displays the major components of the `Undo` function, through which the most recent line created from a sketch based on a single colour could be removed (Ferguson, 2015).

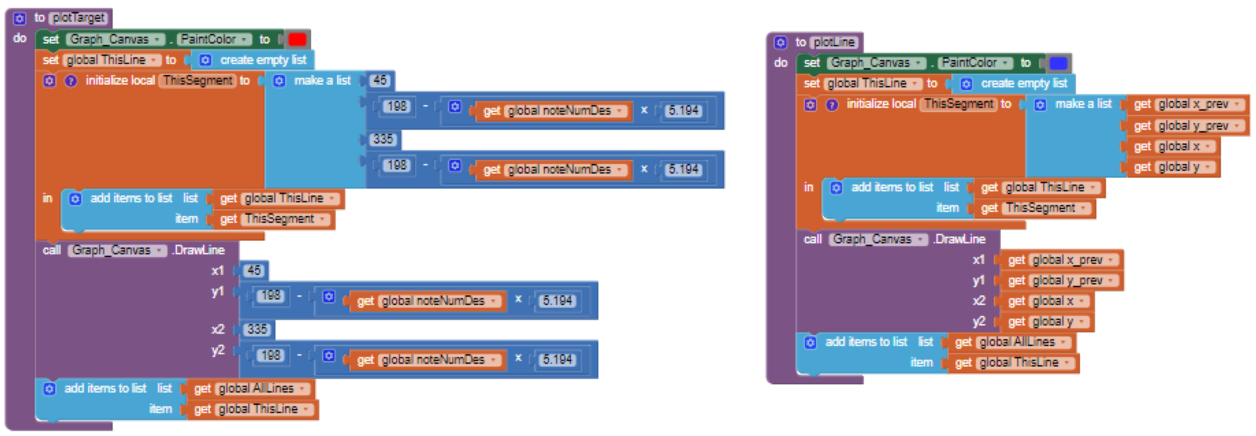


```
when Undo .Click
do  if  not  is list empty? list  get global AllLines
    then  remove list item  list   get global AllLines
                            index  length of list  list  get global AllLines
    call Canvas1 .Clear
    for each ThisLine in list  get global AllLines
    do  for each ThisSegment in list  get ThisLine
        do  call Canvas1 .DrawLine
                x1  select list item  list   get ThisSegment
                                      index  1
                y1  select list item  list   get ThisSegment
                                      index  2
                x2  select list item  list   get ThisSegment
                                      index  3
                y2  select list item  list   get ThisSegment
                                      index  4
```

```
when Canvas1 .Dragged
 startX  startY  prevX  prevY  currentX  currentY  draggedAnySprite
do  initialize local ThisSegment to  make a list  get prevX
                                                  get prevY
                                                  get currentX
                                                  get currentY
    in  add items to list  list  get global ThisLine
                           item  get ThisSegment
    call Canvas1 .DrawLine
                      x1  get prevX
                      y1  get prevY
                      x2  get currentX
                      y2  get currentY
```

__Figure 50__: Undo Capability (Ferguson, 2015)

The applicable components of the `undo` file were incorporated into Flow-Orientated mode; however, the desired result was not obtained. Two instances were added to the list (**Figure 51**), `plotTarget` represented the `Desired  Note` and `plotline` represented the breath information. It therefore, ensured that the most current information was plotted in the desired colour.



```
to plotTarget
do  set Graph_Canvas . PaintColor to 🟥
    set global ThisLine to  create empty list
    initialize local ThisSegment to  make a list  45
                                                  198 - ( get global noteNumDes × 5.194 )
                                                  335
                                                  198 - ( get global noteNumDes × 5.194 )
    in  add items to list  list  get global ThisLine
                           item  get ThisSegment
    call Graph_Canvas .DrawLine
                   x1  45
                   y1  198 - ( get global noteNumDes × 5.194 )
                   x2  335
                   y2  198 - ( get global noteNumDes × 5.194 )
    add items to list  list  get global AllLines
                       item  get global ThisLine
```

```
to plotLine
do  set Graph_Canvas . PaintColor to 🟦
    set global ThisLine to  create empty list
    initialize local ThisSegment to  make a list  get global x_prev
                                                  get global y_prev
                                                  get global x
                                                  get global y
    in  add items to list  list  get global ThisLine
                           item  get ThisSegment
    call Graph_Canvas .DrawLine
                   x1  get global x_prev
                   y1  get global y_prev
                   x2  get global x
                   y2  get global y
    add items to list  list  get global AllLines
                       item  get global ThisLine
```

__Figure 51__: Individual List Generation

It is observed from **Figure 52** that although the most current `Desired Note` was output correctly in red, all previous instances were output in blue. It was realised that as `plotTarget` and `plotline` contributed to the same list, upon redrawing the list after clearing the canvas, the two variables could not be differentiated. Additionally, the previous `Desired Note` information was not appropriately cleared due to the list amalgamation.

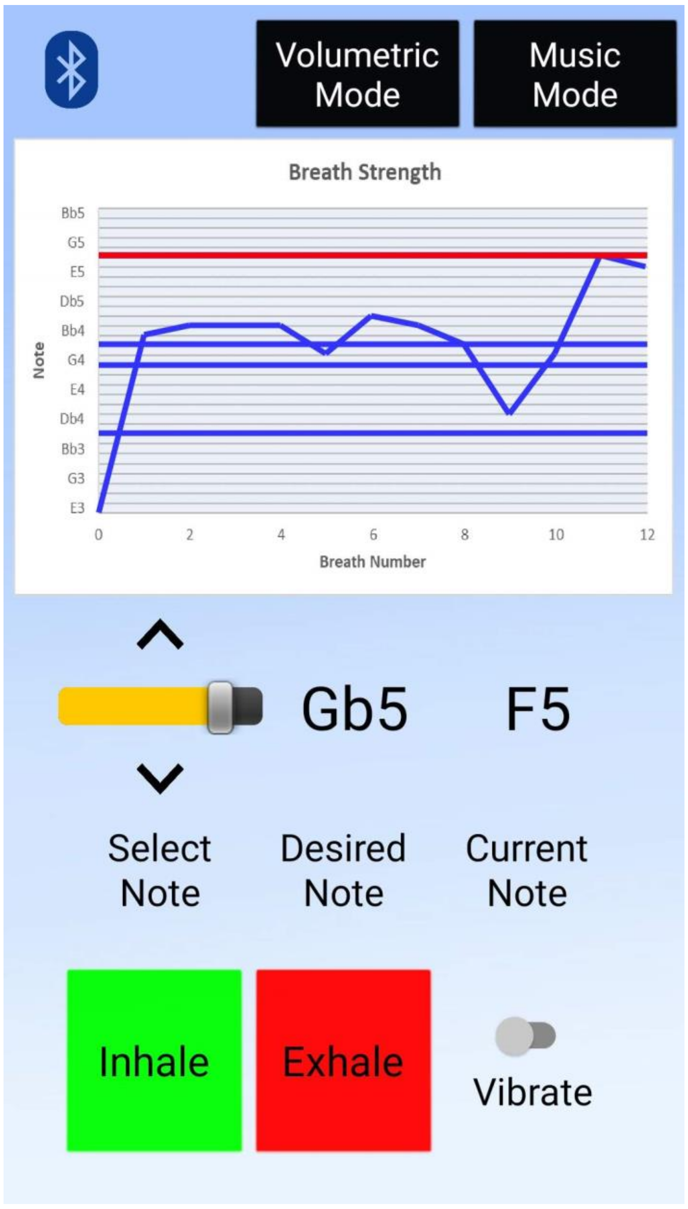
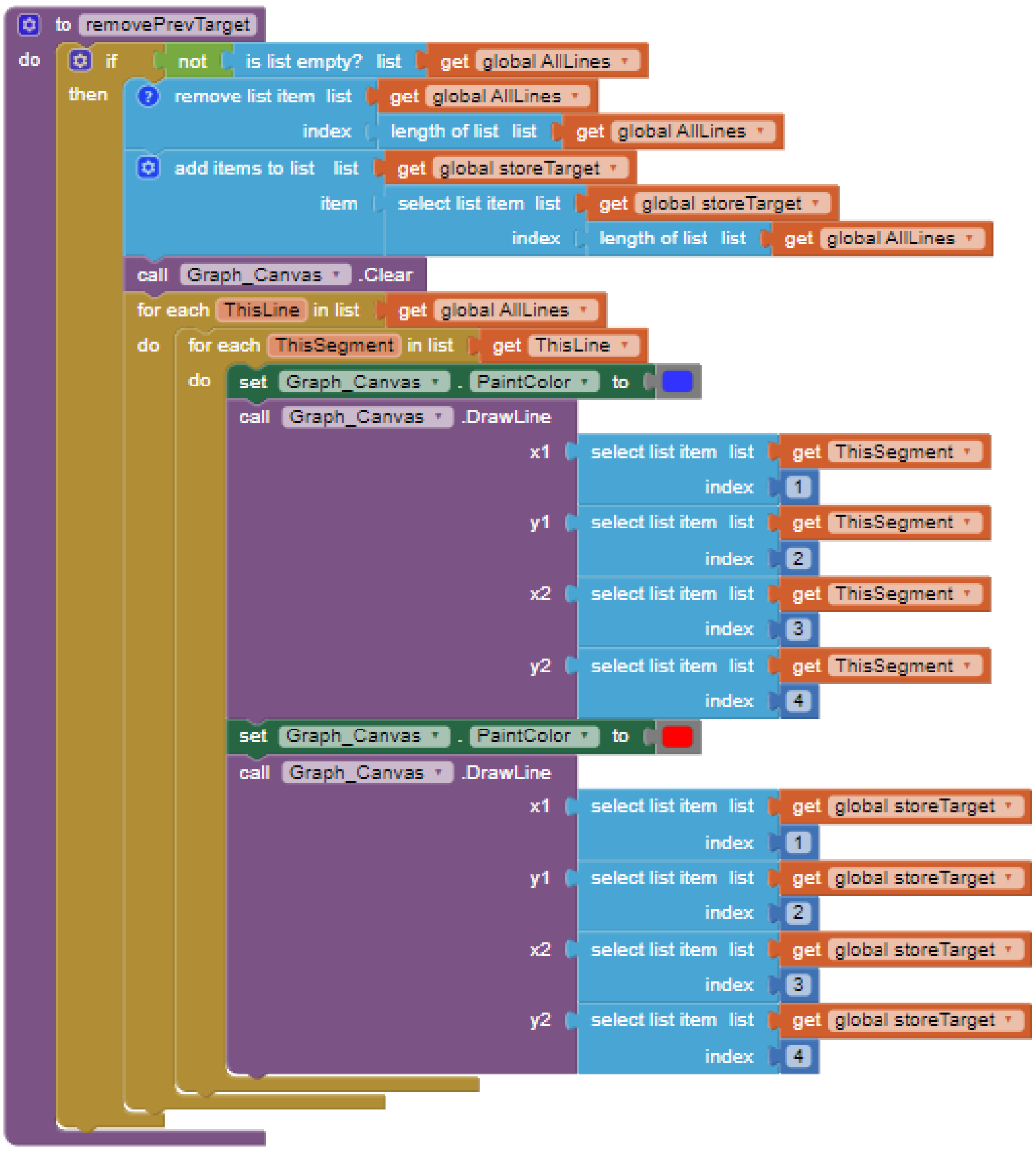


***Figure 52**: Flow-Orientated Undo Error*

As shown in the `removePrevTarget` function (**Figure 53**), representing the removal of the previous `Desired Note`, an additional list titled `storeTarget` was created. The list enabled specific recognition and differentiation between the breath information and `Desired Note` information. Two instances of `Graph_Canvas.DrawLine` were instigated, following the initialisation of either red or blue, dependant on the information being graphed (**Figure 53**).



```
to removePrevTarget
do  if   not  is list empty?  list  get global AllLines
    then  remove list item  list  get global AllLines
                            index  length of list  list  get global AllLines
          add items to list  list  get global storeTarget
                             item  select list item  list  get global storeTarget
                                                     index  length of list  list  get global AllLines
    call Graph_Canvas .Clear
    for each ThisLine in list  get global AllLines
    do  for each ThisSegment in list  get ThisLine
        do  set Graph_Canvas . PaintColor to 🟦
            call Graph_Canvas .DrawLine
                 x1  select list item  list  get ThisSegment
                                       index  1
                 y1  select list item  list  get ThisSegment
                                       index  2
                 x2  select list item  list  get ThisSegment
                                       index  3
                 y2  select list item  list  get ThisSegment
                                       index  4
            set Graph_Canvas . PaintColor to 🟥
            call Graph_Canvas .DrawLine
                 x1  select list item  list  get global storeTarget
                                       index  1
                 y1  select list item  list  get global storeTarget
                                       index  2
                 x2  select list item  list  get global storeTarget
                                       index  3
                 y2  select list item  list  get global storeTarget
                                       index  4
```

***Figure 53**: Remove Previous Desired Note Function*

The red `Desired Note` indicator appears on top of the blue breath information (**Figure 54**). The reference to two separate lists ensured only the current `Desired Note` position is observed.

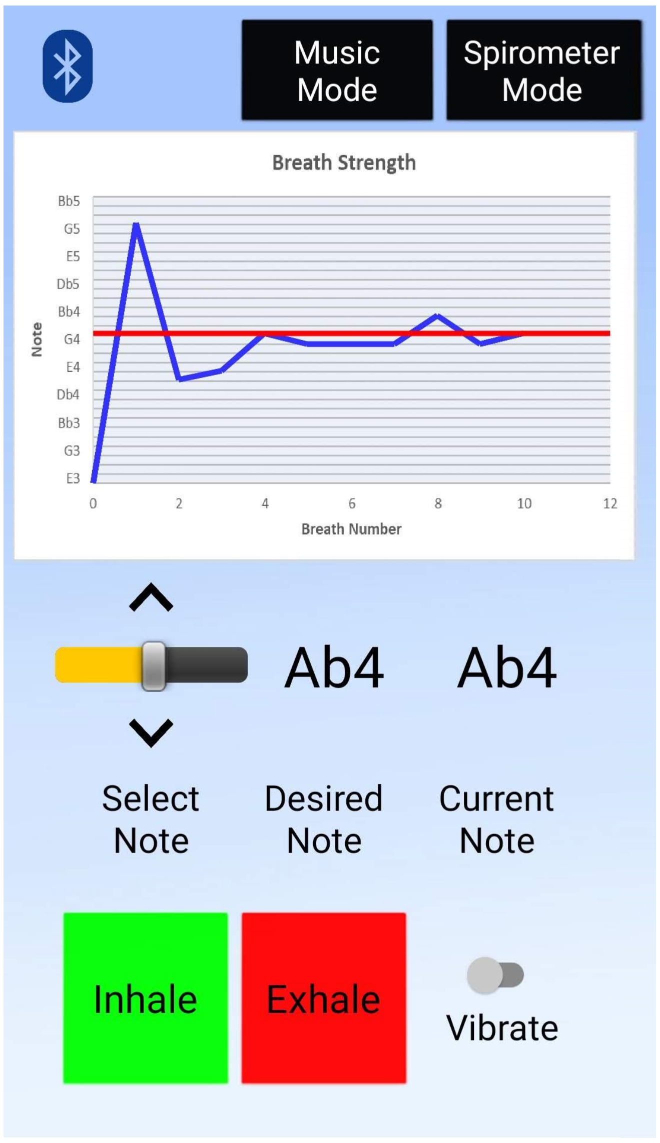
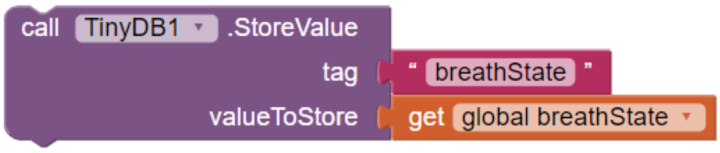


***Figure 54**: Flow-Orientated Final Interface*

Compared to the initial development stage shown in **Figure 46**, the `Select Note` arrows were rotated 90 degrees, from pointing left and right to pointing up and down. To maximise simplicity and intuitive use, the direction of the arrow therefore directly correlated to the movement of the red `Desired Note` line. To further increase usability, it was attempted to also rotate the slider. Upon investigation, it was discovered that an `ImageSprite` is the only the component which can be rotated. As the `Slider` object is comprised of numerous complex elements and could not be simply recreated vertically, the horizontal direction was maintained.

As the Flow-Orientated Mode was designated as the default interface, the `breathState` and `Vibrate` parameters defined were communicated between the two remaining screens. To enable data sharing across multiple screens, the non-visible `TinyDB` component was included on each screen.

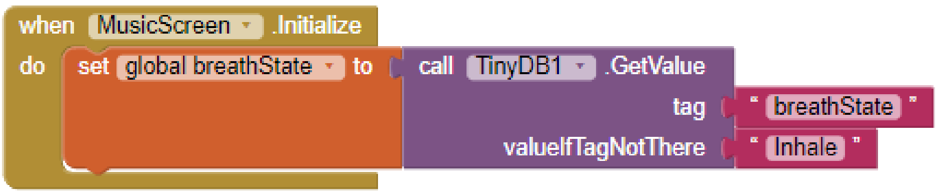
A `TinyDB.StoreValue` block was therefore placed on the Flow-Orientated Mode screen. The `tag` enabled a mechanism to reference the desired value, as multiple values can be stored in a single `TinyDB` component. In **Figure 55**, the state of the breath to be detected is recorded in the form of a string, as either `Inhale` or `Exhale` and assigned to `valueToStore`.



```
call TinyDB1 .StoreValue
                    tag  " breathState "
           valueToStore  get global breathState
```

***Figure 55**: Storing a Value*

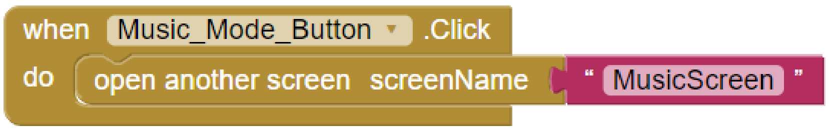
To retrieve the value stored in `breathState`, the `TinyDB.GetValue` block was called following the initialisation of a different screen. The stored value is set to the variable `breathState`; if no value is located, the default state is set to Inhale, defined by the `valueIfTagNotThere` command (**Figure 56**).



```
when MusicScreen .Initialize
do   set global breathState to  call TinyDB1 .GetValue
                                                   tag  " breathState "
                                     valueIfTagNotThere " Inhale "
```

***Figure 56**: Retrieving a Stored Value*

To open one of the two alternative screens, two buttons were created, `Music_Mode_Button` and `VolumetricButton`, located in the top-right position of the app interface. **Figure 57** shows how the `open another screen` block opens the specified screen, actioned following the click of `Music_Mode_Button`.



```
when Music_Mode_Button .Click
do   open another screen  screenName  " MusicScreen "
```

***Figure 57**: Open Alternative Screen*

4.2.2.2 Music Creation Mode

The second interface is designed specifically for music creation and enables chord selection based on the root note played by the user. The root note is played based on FIS techniques; a competent user is expected to arise only after significant exposure to the flow-oriented interface. The major, minor, seventh, minor seventh, major seventh and suspended second and fourth were the seven available chords. An intentional design decision was to ensure all buttons on the musical creation interface were large, such that users lacking fine motor control would not be disadvantaged.

In addition to replication of the trumpet through physical pushbuttons, the three valves were also simulated on the app, in the event that interfacing with an app is more convenient or appropriate for the user. The three trumpet valves were simulated by the three buttons located in a vertical arrangement on the left of **Figure 58**, labelled `PB1`, `PB2` and `PB3`.

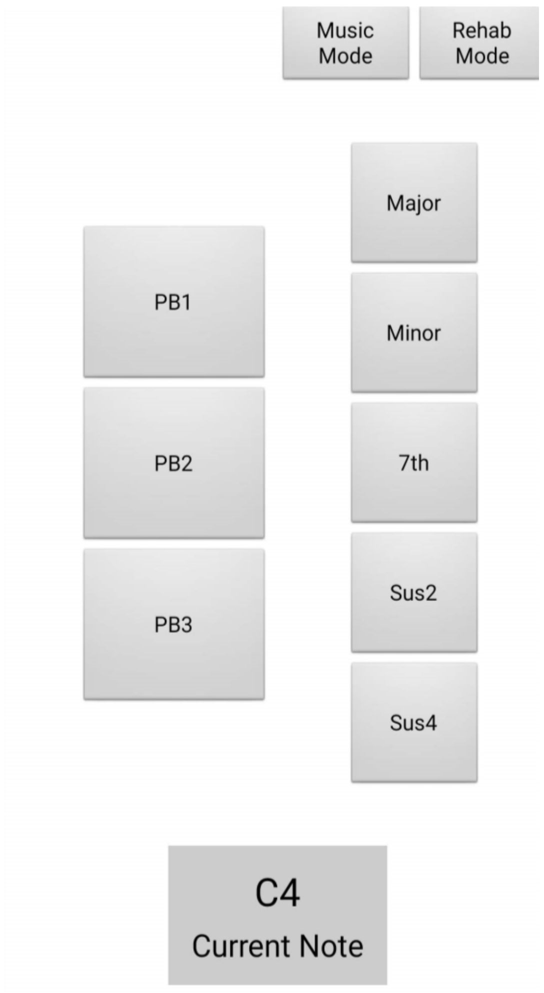
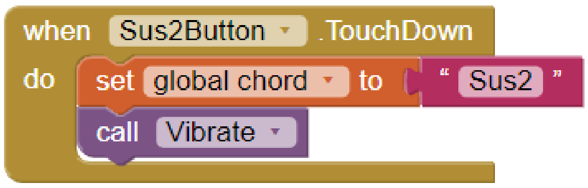


***Figure 58**: Music-Orientated Initial Interface*

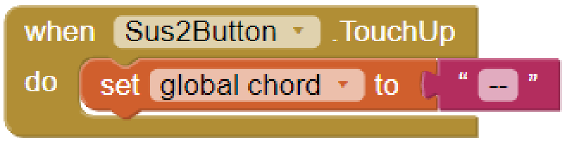
Compared to the Flow-Orientated Mode, the display of the Music Creation Mode was much simpler. The two buttons in the top-right corner maintain the control to the two alternative screens, however, the `Current Note` was the only dynamic information displayed. The current note was received as string information from the Teensy through Bluetooth communication.

Unlike the state of breath from the Flow-Orientated mode, the chord selected is output for the duration the button is pressed. Therefore, it is required to detect when the button is pressed, and when the button is released, for the variable to be set accordingly. The action of pressing the any of the chord buttons instigates the setting of the `chord` variable, as shown by the assignment of the suspended second chord in **Figure 59**, followed by a call to the vibrate function. The variable stored in `chord` remains unchanged, until another button representing a different chord is pressed, or the button is released. To detect the release of a button, the `TouchUp` block was utilised, which set the `chord` variable to a null state. The assignment of `--` (**Figure 60**), to `chord` does not correlate to a chord defined in the Teensy, therefore no chord will be played back.



```
when Sus2Button .TouchDown
do  set global chord to " Sus2 "
    call Vibrate
```

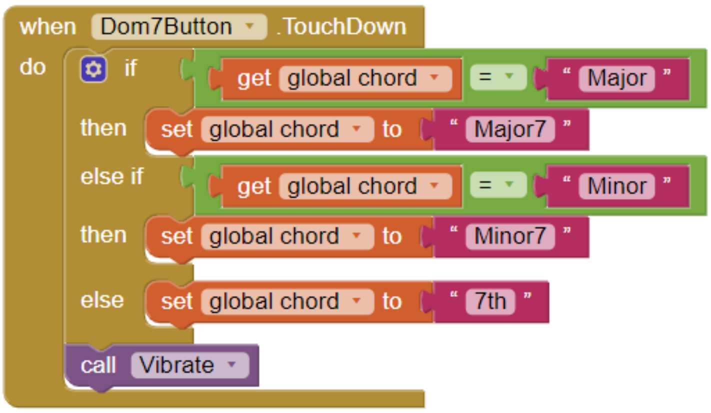
***Figure 59**: Action Following Chord Press*



```
when Sus2Button .TouchUp
do  set global chord to " -- "
```

***Figure 60**: Action Following Chord Release*

As the size of the buttons was maximised, the number of buttons was therefore limited. To enable the output of seven available chords, the five buttons visible on the right of **Figure 58** were pressed in specific combinations. The simultaneous combination of pressing either `Major` and `7th`, or `Minor` and `7th`, would result in a major seventh, or minor seventh chord, respectively. The programming for the `Major`, `Minor` and `7th` buttons, therefore, included an additional statement, which evaluated the current status of all other buttons. As shown in **Figure 61**, if the `7th` button is pressed, titled `Dom7Button`, the string assigned to `chord` is dependent on whether `Major` or `Minor` was also pressed simultaneously.



```
when Dom7Button .TouchDown
do  if    get global chord = "Major"
    then  set global chord to "Major7"
    else if  get global chord = "Minor"
    then  set global chord to "Minor7"
    else  set global chord to "7th"
    call Vibrate
```

Figure 61: Assignment of 7th Chord

As can be observed in **Figure 62**, all buttons have been colourised, such that the valves and chords are distinctly separated into different groups. Additionally, a sheet music background has been applied, as the Music Creation mode focuses on musical expression, rather than therapy.

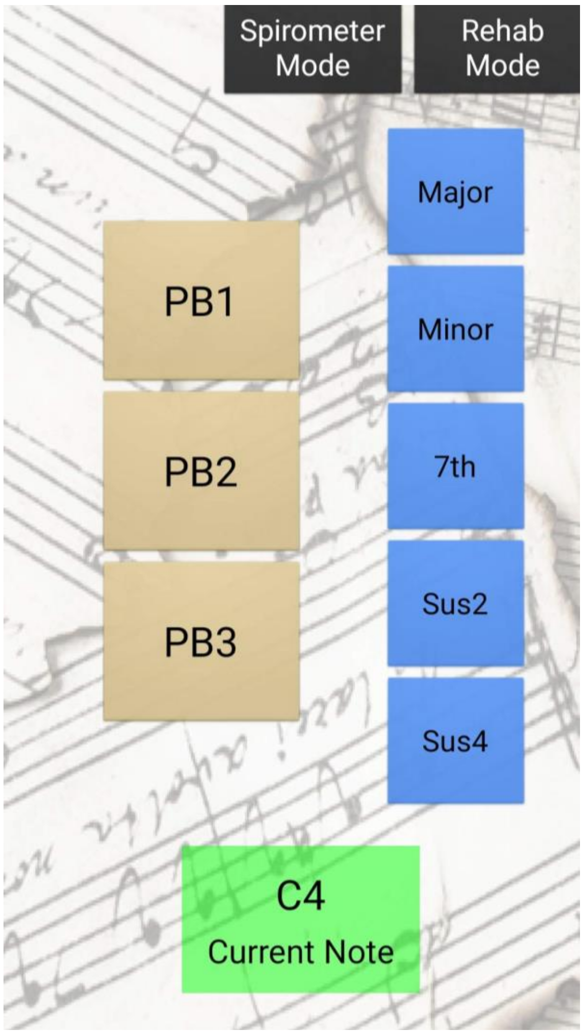
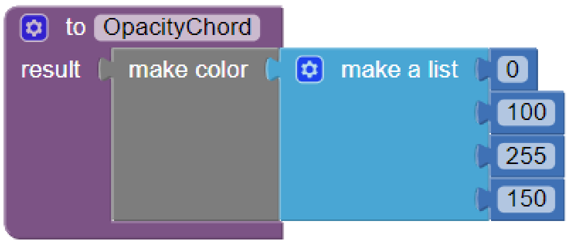


Figure 62: Music-Orientated Final Interface

The final modification implemented transparency and an opacity change following a button press. The `make color` block enabled the definition of a colour in the form of a list. In **Figure 63**, the first three items specify the Red, Green, and Blue, values of and RGB complex, respectively, the fourth item specifies the desired opacity, all values range from 0 – 255.



```
to OpacityChord
result  make color  make a list  0
                                 100
                                 255
                                 150
```

Figure 63: *Design of Blue Chord Buttons*

As can be observed in **Figure 64**, the three buttons have been renamed from `PB1`, `PB2` and `PB3`, to `Valve 1`, `Valve 2`, and `Valve 3`. Additionally, the change in opacity of the `Valve 2` and `Sus2` button can be observed following a press, in comparison to the other buttons.

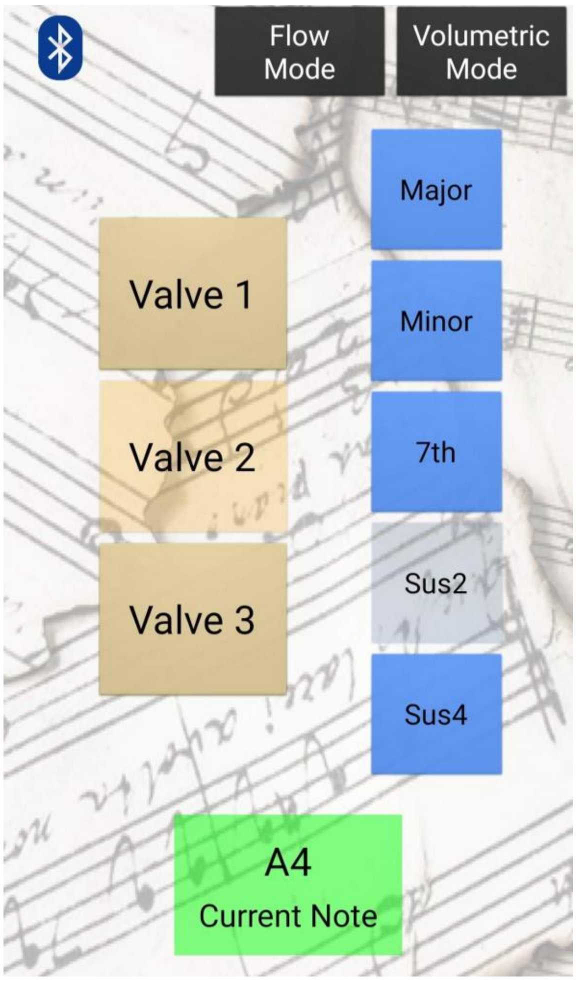


Figure 64: *Music Creation Mode - Button Press*

4.2.2.3 Volume-Orientated Incentive Spirometry Mode

The final interface provides a digital representation of the visual information obtained through VIS. The prescribed volume is entered by the user, through which the app calculates how many seconds of inhalation is required. Following detection of inhalation, the bar graph will rise in increments of 31, during which a scale of the trumpet will be cycled through; both the music and bar graph will cease if the lack of inhalation is detected. To ensure the user inhales at a rate consistent with an IS, a colour-based indicator will be represented against the bar graph, visually confirming if the breath is too fast. Additionally, the volume of the scale will be loudest when correct breath velocity is detected, and proportionally reduce as the velocity is raised. Haptic feedback is also provided through a single vibration once the designated volume has been reached. The varying auditory output combined with the vibration enables outcomes of the IS to be achieved for the vision impaired, currently unachievable with commercial options.

In 2010, 32.4 million people were found to be blind globally, while 191 million people were found to have either moderate or severe vision impairment, leading to a combined 223.4 million people with significant or complete vision loss (Stevens et al., 2013). The global population in 2010 was approximately 7 billion people (Worldometers, 2019), therefore approximately 3% of the population is assumed to be unable to independently engage with ISy. 41.23 million surgeries in 2015 were recorded to be of respiratory, cardiovascular and digestive nature (Rose et al., 2015), each requiring postoperative ISy to minimise the development of PPCs. Therefore, it can be assumed that approximately 1.2 million people will have significant or complete vision loss and will also undergo respiratory, cardiovascular or digestive surgery, who will benefit from this device.

Development was commenced through the design of a bar graph, with a height which could be dynamically set. The bar graph was created with the button `TimeBarGraph`, and the visible height was modified by the `set TimeBarGraph.HeightPercent` block (Hochgraf, 2016), For the bar graph to rise to the desired height, an increment of `1` pixel was initialised to increase every 30ms, until the threshold of 40% was reached. In the initial development stage, the desired input volume did not determine the threshold.

A `Start Exercise` and `Continue Exercise` button were designed (**Figure 65**). A single `Start Exercise` button press initialised the first clock cycle and generated a single increment, while a sustained `Continue Exercise` press maintained the graph increment. For the bar graph height to return to the initial position upon reaching the threshold, an `if` statement was implemented to resent the height of the bar graph, which was actioned the following clock cycle.

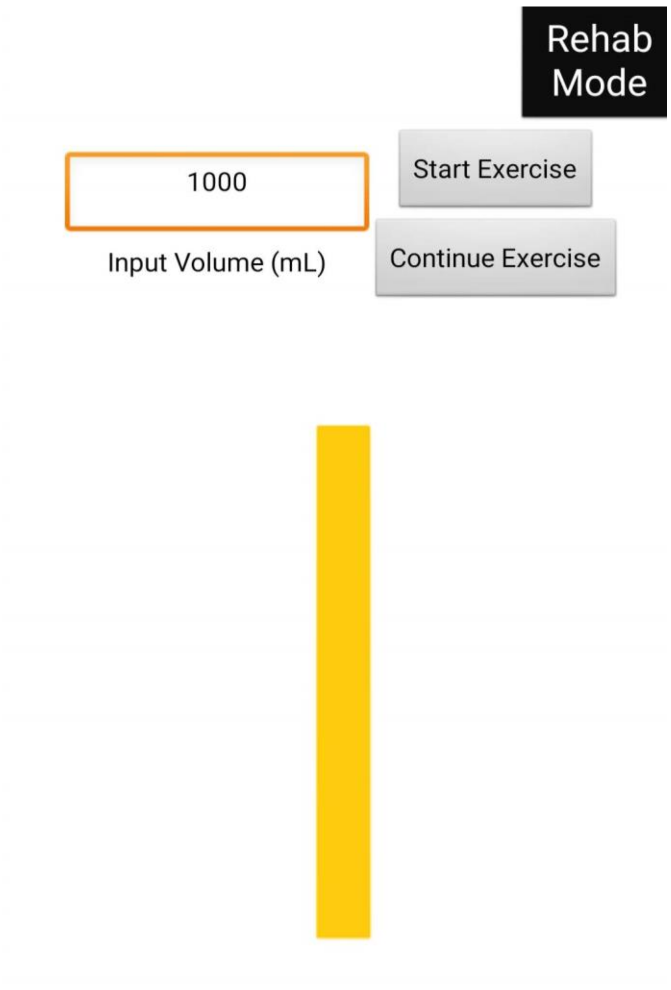


***Figure 65**: Volume-Orientated Initial Interface*

For the bar graph to reflect the desired volume, a reference scale was implemented in the form of a simulated IS background image. The input volume of 3000mL is reflected by the indicator (**Figure 66**), which was programmed to automatically adjust its position based on the specified input volume.

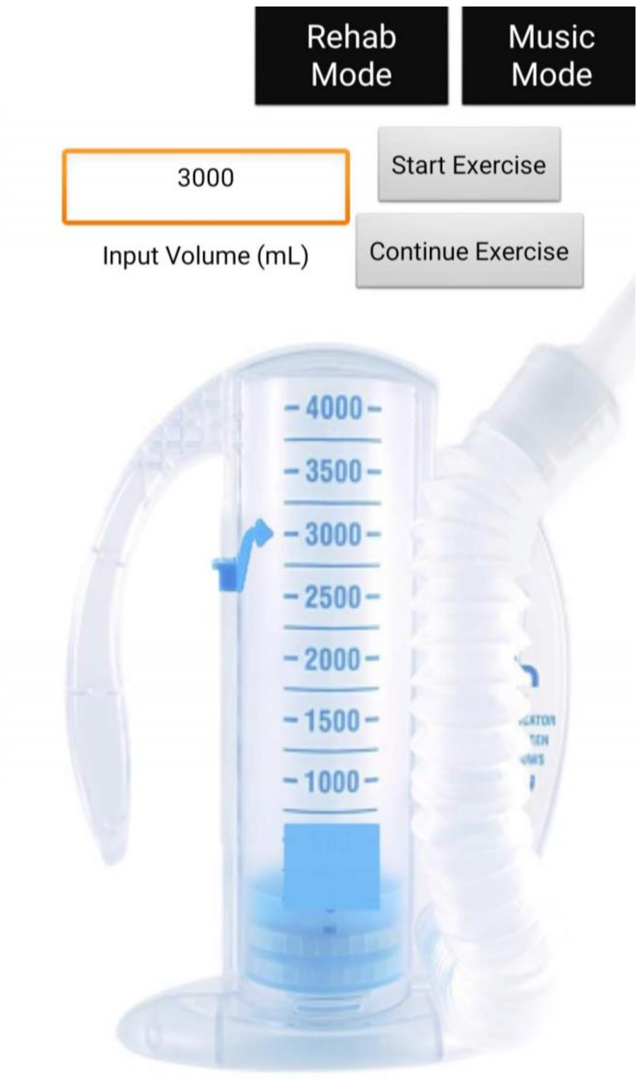
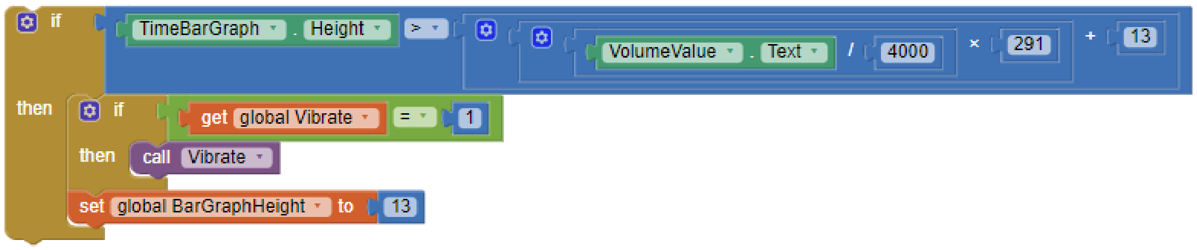


***Figure 66**: Volume-Orientated with Background*

It was identified that the maximum desired height of the IS was represented by 291 pixels, and the minimum was offset to 13 pixels. The desired height of the bar graph could, therefore, be mathematically derived based on the `set TimeBarGraph.Height` block, dependent on a position rather than a percentage. The threshold was calculated by dividing the desired volume by 4000, representing the maximum value of 4000mL, multiplied by the scaling factor of 291 and the result was added by 13. If the height of the bar graph was greater than the threshold, the function to vibrate the phone was called if vibration as enabled, and the bar graph height was reset to 13 (**Figure 67**).



```
if   TimeBarGraph . Height  >  VolumeValue . Text / 4000 × 291 + 13
then  if   get global Vibrate = 1
      then call Vibrate
      set global BarGraphHeight to 13
```

Figure 67: Volume-Orientated Threshold Calculation

The indicator was represented by an `ImageSprite`, an element which can be programmed to change positions if placed on a `canvas`. To control the indicator of the desired volume, a similar calculation was performed, however as the original position of the canvas is located in the top-left corner, the scaled value was subtracted from 291.

Finally, the melodic minor, natural minor, harmonic minor, and major scale blue buttons to select the desired scale from the four options were added (**Figure 68**).

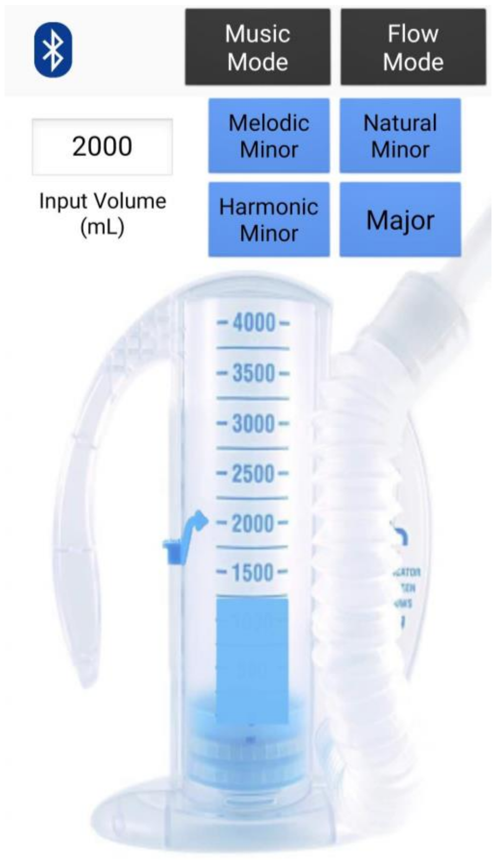


Figure 68: Volumetric-Orientated Final Interface

The full code for the FIS mode, music creation mode and VIS mode is located in Appendix E.

5 Results

The functionality of FIS was simulated through the ability to correlate a breath flow to a note. A graphical user interface with the ability to set the desired note, select the preferred breath mode and toggle the vibrational feedback was produced (**Figure 69**).

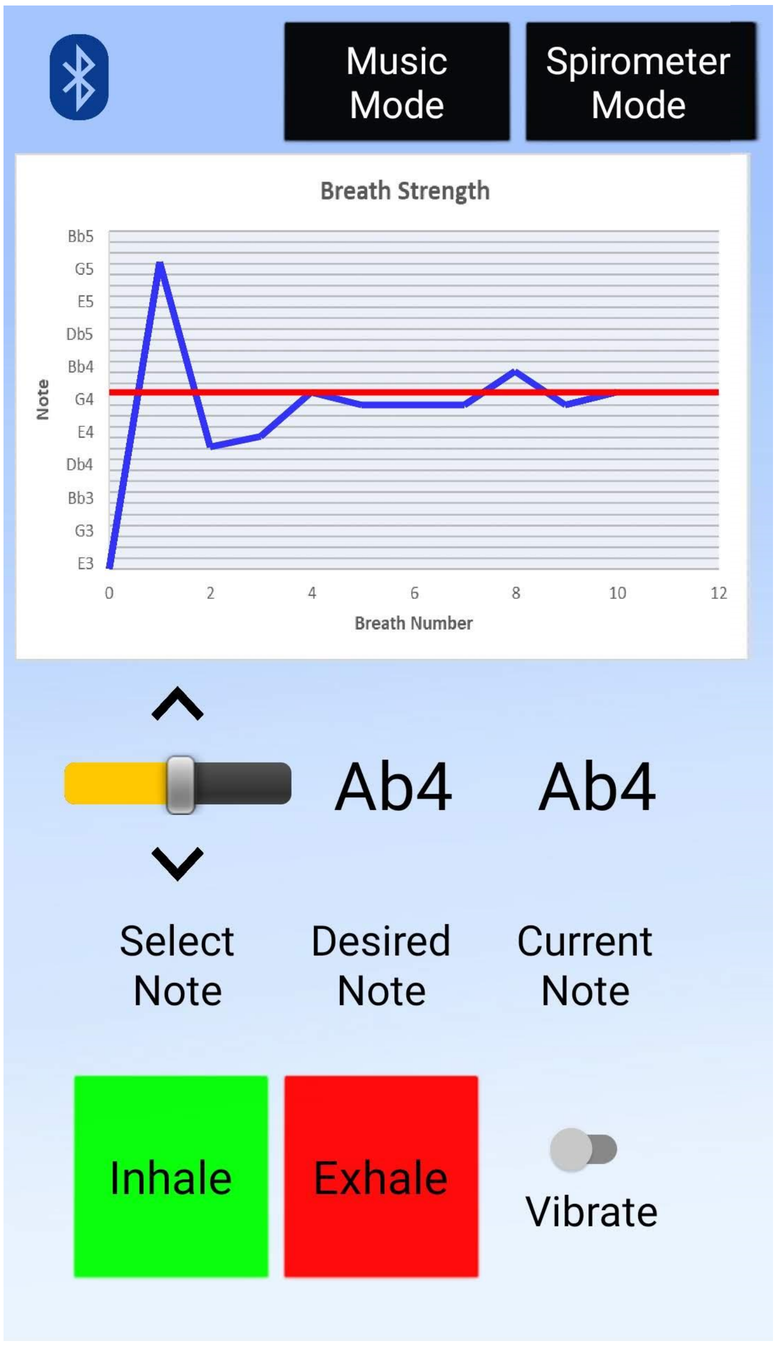


***Figure 69**: Flow-Orientated Final Interface*

An opportunity for the user to exhibit musical expression was enabled through the ability to play the melody with the breath and valves, and harmony with the introduction of chords (**Figure 70**). Acoustic trumpets are monophonic instruments, therefore only one note can be played at a time (Nypaver, 2018). The ability to play chords through the electronic trumpet is a unique feature to further encourage and facility musical creativity.

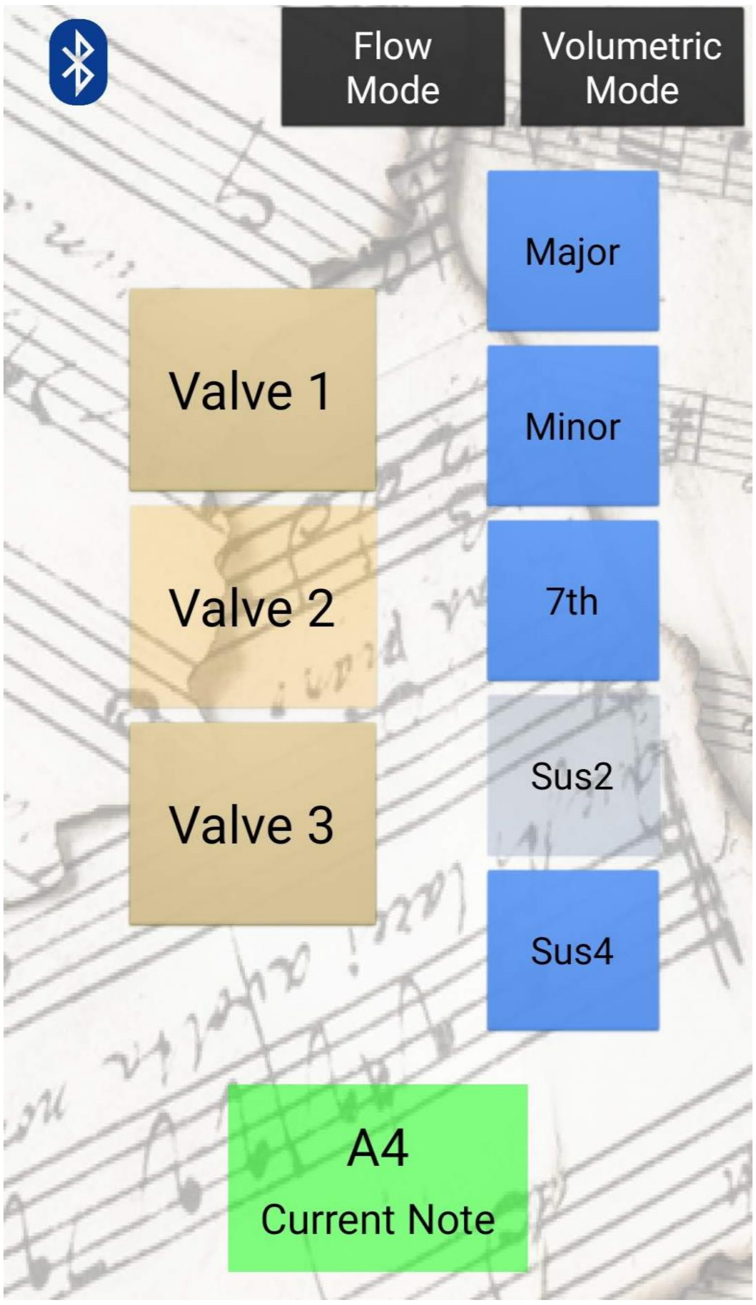


***Figure 70**: Music Creation Mode - Button Press*

The functionality of VIS was simulated through the ability to sustain a breath alongside a timed series of ascending and descending notes in the form a musical scale. VIS usually provides a visual indicator to assist in ensuring minimal breath force is expelled during use, the indicator is usually separated into three categories, good, better and best, as shown in the left column of **Figure 6**. In addition to the proportional reduction in auditory volume of the scale, the breath force information was also provided by the colour of the bar graph. From **Figure 71**, the green, yellow and red graphs represent a breath force which is ideal, intermediate and too high, respectively.

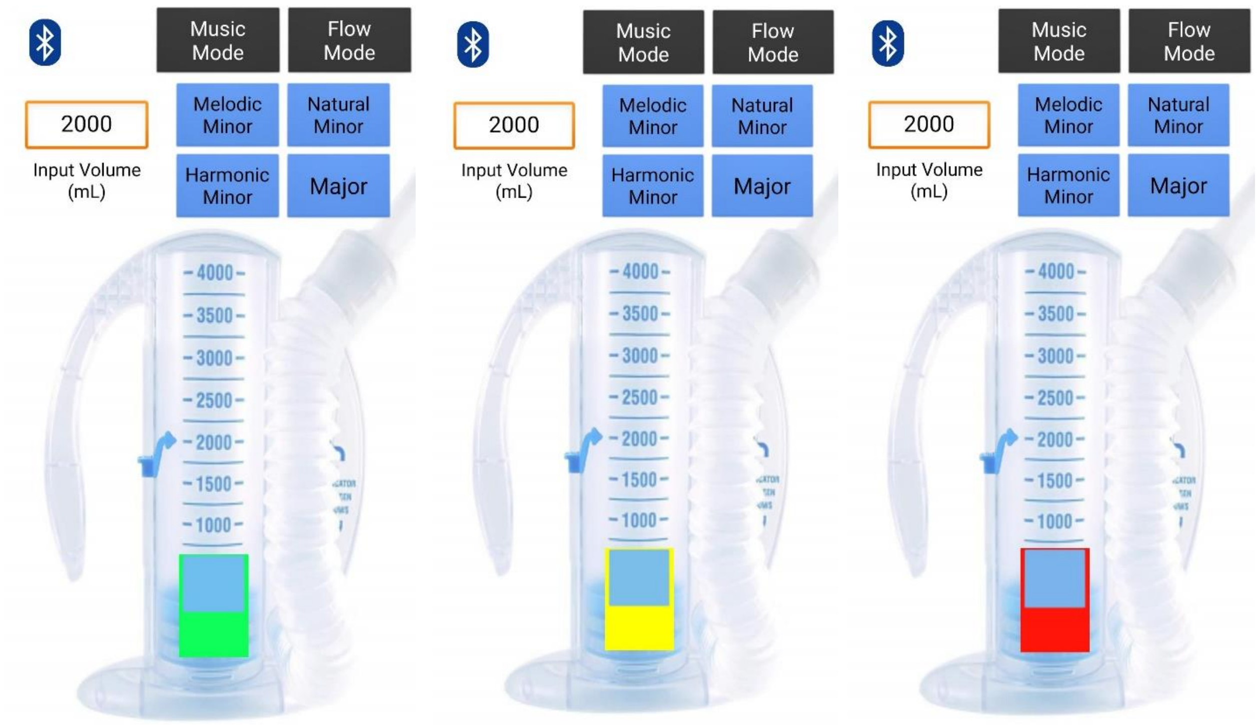


***Figure 71**: Volumetric-Orientated Interface: Left - Ideal Range, Middle - Intermediate Range, Right - Poor Range*

The device is yet to be tested by patients or target users; future focus groups will include trials conducted by those with a PPC, recovering from a surgery, and with physical limitations. Feedback from all demographics will be taken into consideration to improve both the hardware and software component of the device.

6 Discussion

The device was successful in its ability to convert analog breath-based information into a digital integer, that was then mapped to a MIDI note and lastly, output as a trumpet sound, providing auditory feedback.

Designed to operate in two separate modes, the device could successfully be used as a musical instrument or as a therapy tool. Functionality and customisation were maximised through a tailored app environment, that provided a digital interface and visual feedback; haptic feedback could also be provided within the app if desired by the user. Three apps were designed, which focussed on FIS, VIS and music creation. Each app was designed to benefit a different user group most significantly, however, all apps were designed to be useable by those with a physical impairment can be enjoyed by all users.

The cost of the proposed device in electronic components is AUD$122 (**Table 7**), compared to AUD$8 - AUD$40 for a currently-prescribed analog incentive spirometer (**Table 2**). Although a factor of 3 – 15 times greater in cost than the current IS, the proposed device incorporates an electric and musical element and is intended to be utilised beyond therapy, justifying the increase in cost. Compared to current breath-based digital instruments, costing between AUD$1000 (**Figure 24**) - AUD$3000 (**Table 2**), and digital music-based respiratory rehabilitation devices costing AUD$1900 and the requiring an iPad (**Table 2**), the proposed device is a factor of 8 – 24 times more cost-effective than commercially available alternatives.

6.1 Limitations

A major limitation was the lack of worldwide clinical evidence recording the patient compliance of ISy use, but more importantly, the lack of understanding reasons for a potential reduction in prescribed use. It was assumed that the repetitiveness of the exercise and lack of multimodal feedback resulted in the loss of interest for a significant number of patients. It is known that inspiration causes discomfort following abdominal or thoracic surgery (Bartlett et al., 1973), however, the effect of discomfort or pain on IS use were not analysed prior to commencement of the project. Additionally, information of the demographic who were either more likely, or less likely, to complete the prescribed respiratory exercises, such as age, gender, and location, was not available. If the patient was old enough to fully understand the risks of discontinuing therapy, or if the patient was from a location with a high prevalence of respiratory diseases, the likelihood of continuing with therapy may have been higher. Following approval from the Southern Adelaide Clinical Health Research Ethics Committee (SAC HERC), a survey should have been provided to postoperative thoracic and abdominal patients of varying demographics. The survey would have been critical in the understanding of the primary limiting factors to full IS engagement, for a variety of demographics. Ideally, survey responses would be obtained from multiple states within Australia, both rurally and in metro areas.

The device has only been tested by the developer, therefore, has not had an opportunity to receive user feedback from patients or target users, and be modified accordingly. Iterative development based on focus group testing will be critical in creating a convenient and desirable device.

The most significant limitation was the inability to quantitatively correlate the flow rate of an IS with the flow rate of the electronic trumpet. The diameter and length of the tube, the size of the IS body and the arrangement of the air-release holes all contribute to the flow required to cause the float, or ball(s), to rise. The outlet valve of the electronic trumpet was configured to be qualitatively equivalent to an IS, however numerical confirmation could not be obtained.

6.2 Future Directions

The outcomes of this thesis presented a functional prototype which successfully translated both inhalation and exhalation into a numerical value, and finally into a music note.

For the project to become incorporated into ISy therapy, a mouthpiece equivalent to commercial incentive spirometers must be integrated with a gooseneck neck tube, in place of the single tube. The current tube diameter of 4mm requires fine muscular control of the face to breathe exclusively through the tube, unachievable for many physical impairments such as cerebral palsy (Walters Wright, 2019). The current design does not allow sufficient release of air, which causes a higher resistance compared to either VIS or FIS.

The primary objective of the device was to increase user compliance and appropriate engage with respiratory therapy. Following approval from the SAC HERC, clinical trials should be commenced to compare user engagement against current incentive spirometers. A survey should be provided alongside the trial to understand the user experience following the use of the device, including but not limited to:

- Willingness to perform required therapy
- Engagement with the app, and which app mode was preferred
- Simplicity to use without the app
- Simplicity to establish a Bluetooth connection and navigate functionality within the app
- Level of comfort compared to traditional IS

Prior to the trial, it is critical that the user demographic is recorded, and analysed appropriately, specifically, age and previous musical engagement. Compared to play therapy, music therapy has been shown to increase happiness for hospitalised children (Hendon and Bohon, 2008). It is therefore expected that children will participate strongly following the musical additional to ISy. Secondly, engagement from a current professional musician, amateur or past musician, or a non-musician, is expected to vary. Additionally, a variance is expected within the musical class, depending on the musician’s instrument preference. The highest participation is expected from brass players, followed by woodwind players, due to the similarity in the process required to generate a sound.

Following the clinical trial assessing user engagement, the therapeutic outcomes of the device must be assessed post-trial and compared against the current therapeutic outcomes. Quantitative studies should be undertaken to confirm than an equivalent level of effort results in an equivalent benefit.

In relation to the app interface, MIT App Inventor was utilised due to fast turnaround time from conceptualisation to a functional prototype, however, limitations and restrictions were observed. A debugger mode could not be accessed, preventing the ability to analyse each line of the program. Additionally, the block-based layout was simple to interface, however, did not offer a mechanism to understand the order in which commands were being executed. Regarding Bluetooth interfacing, a connection could not be maintained following the selection of a different mode, or screen, resulting in the undesirable requirement to regularly reconnect to the HC-06. Finally, multiple objects could not be overlaid, and exact object positions could not be specified, resulting in the inclusion of empty horizontal and vertical arrangements, intended only to act as a placeholder for object positioning. It is therefore recommended that text-best programming environment, dedicated to mobile app development, is employed in future iterations, such as Android Studio.

An electronic reminder generated in the form of a beep resulted in a twofold increase in patient compliance, compared to those without the reminder (Eltorai et al., 2019). Similarly, the time between uses could be detected by the app interface to instigate a reminder to the user. If the user is found to repeatedly ignore reminders, an alarm my sound on the user’s mobile device which can only be turned off after appropriate inhalation is detected. The physician will initially define the required frequency of use in the hospital, and if internet connectivity is configured, may have access to the profile of the patient and can adjust parameters or communicate accordingly. Reducing health insurance premiums through demonstrations of healthy lifestyles have become increasingly prevalent following the rise in wearable devices (Laycock, 2019). Potential partnering with health insurance companies may provide a monetary incentive to perform sufficient postoperative rehabilitation.

It is acknowledged that following the purchase of the Teensy 3.6, and Audio Adapter Board (Rev C), the Teensy 4.0 has been released. The Teensy 4.0 offers six times the processing capabilities, and a reduction in size following a reduction from 62 to 34 I/O pins, compared to the Teensy 3.6 (PJRC, 2019d). The pin layout of the Teensy 4.0 differs from the Teensy 3.X modules and was not directly compatible with and Audio Adapter Board (Rev C); therefore the Audio Adapter Board (Rev D) has been released (PJRC, 2019a). If released earlier, the Teensy 4.0 and Audio Adapter Board (Rev D) would have been purchased, the superior processor reduces limitations with computationally complex audio analysis and playback functionality.

Incorporating the full capabilities of the Audio System Design Tool into the app interface was beyond the scope of the project. However, the Teensy Audio Library is designed to enable users without extensive programming capabilities to design audio systems. Future iterations may include the ability to select an instrument other than the trumpet and apply different MIDI effects. For example, the output instrument could be converted to a guitar, and typical guitar effects such as distortion, delay and reverb may be introducing, while maintaining control of the notes through the breath.

The final recommendation is the design of a custom printed circuit board (PCB). PCBs offer superior current carrying capacity, the ability to reduce the footprint of the design and permanence of the soldered components (Candor Industries, 2018). As the Teensy can be externally powered, 3.3V can be supplied to the `Vin` pin of the Teensy, and the HC-06. The pushbuttons and airflow sensors do not require a specific voltage, however, the 3.3V supplied by the Teensy has facilitated functional operation and is assumed to remain appropriate following PCB incorporation. Once the size of the prototype has been reduced, the trumpet form factor is expected to be developed. To provide versatility and customisation, critical for an assistive musical device (Frid, 2019), the form factor will be 3D modelled and printed; the position, arrangement and size of the buttons can, therefore, be designed specifically to the requirements and preferences of the user.

7 Conclusion

Initially, the capabilities of the Arduino Uno were explored; the challenges associated with producing a MIDI output without a MIDI-compliant device were identified. The complexities of integrating two additional software packages with the Arduino IDE, combined with the loss of the ability to debug and access the Serial Monitor deemed the Arduino Uno an unsuitable development board for the aims of the project.

The Teensy 3.6, a MIDI class compliant device alleviated all difficulties associated with the Arduino Uno and facilitated simple MIDI communication. A primary objective of the project was to enable convenience in use, as to minimise impediments to participate in respiratory therapy. Although the Teensy 3.6 was capable of transmitting MIDI messages to Ableton Live, the generation of the musical instrument was dependant on the music production software, and thus, connection to a computer. The Teensy Audio Adaptor Board interfaced with the battery-powered Teensy 3.6 and provided onboard audio processing functionalities, through which the trumpet notes were output without computer dependency. The Teensy 3.6 accepted information from the MPXV7002 differential pressure sensor, capable of detecting positive and negative pressures, hence extended potential therapy applications to inhalation and exhalation. Information detected from the differential pressure sensor correlated to the strength of breath, and hence determined the output trumpet note. The primary project objective of developing a cost-effective and engaging respiratory therapy device, to provide health benefits equivalent to the IS is therefore met.

The device was designed to operate in one of two modes, as a musical instrument or as a therapy tool. To maximise functionality and provide customisation, a visual interface was provided through a tailored app environment. A Bluetooth module was incorporated into the electronic trumpet, which communicated the differential pressure sensor and push-button information to the app located on a mobile device. Three apps were designed, which focussed on FIS, VIS and music creation. The FIS and VIS applications were designed to require similar breathing actions and result in analogous therapeutic outcomes as current incentive spirometers. The secondary objective of developing a breath-based assistive musical instrument for people with physical disabilities is also met. The device is capable of being played as an expressive musical instrument, or as a therapeutic tool with a musical output, therefore meeting the secondary project objective.

Conclusion

This thesis reports on an innovative alternative to standard incentive spirometers. Once the system has been fully developed and prototyped, it will be trialled with end-users to determine acceptance and utility, as well as assessing whether multimodal interactive feedback through music from the device increases patient compliance, enjoyment, and a willingness to perform respiratory therapy.

8 References

Ahonen-Eerikainen, H., Lamont, A. and Knox, R. (2008), “Rehabilitation for children with Cerebral Palsy: seeing through the looking glass - enhancing participation and restoring self-image through the virtual music instrument”, *International Journal of Psychoso-Cial Rehabilitation*, Vol. 12 No. 2, pp. 41–66.

Alaparthi, G.K., Augustine, A.J., Anand, R. and Mahale, A. (2016), “Comparison of Diaphragmatic Breathing Exercise, Volume and Flow Incentive Spirometry, on Diaphragm Excursion and Pulmonary Function in Patients Undergoing Laparoscopic Surgery: A Randomized Controlled Trial”, *Minimally Invasive Surgery*, Hindawi Publishing Corporation, Vol. 2016, available at:https://doi.org/10.1155/2016/1967532.

Ali Khan, N., Khan, I., Samo, K.A., Siraj Memon, A. and Karachi, H. (2015), *Role of Incentive Spirometry in Trauma Patients Managed with Tube Thoracostomy.*

All About Microcontrollers. (2017), “Difference between 4 bit and 8 bit lcd interfacing - Microcontroller Projects”, available at: https://www.microcontroller-project.com/lcd-in-4-bit-mode-and-8-bit-mode.html (accessed 7 October 2019).

do Amaral, M.A.S., Neto, M.G., de Queiroz, J.G., Martins-Filho, P.R.S., Saquetto, M.B. and Carvalho, V.O. (2016), “Effect of music therapy on blood pressure of individuals with hypertension: A systematic review and Meta-analysis”, *International Journal of Cardiology*, Vol. 214, pp. 461–464.

Amazon. (2017), “Customer reviews: Yamaha EZ-TP MIDI Trumpet”, available at: https://www.amazon.com/Yamaha-EZ-TP-MIDI-Trumpet/product-reviews/B000E5YP0M (accessed 29 July 2019).

Anghel, A.G. (2017), “Bluetooth Arduino RECEIVE data + Chart”, *YouTube*, available at: https://www.youtube.com/watch?v=JQ3tDhpmSFE.

Arduino. (2017), “Arduino - Getting Started”, available at: https://www.arduino.cc/en/Guide/HomePage (accessed 17 October 2019).

Arduino. (2019a), “tone()”, available at: https://www.arduino.cc/reference/en/language/functions/advanced-io/tone/ (accessed 6 August 2019).

Arduino. (2019b), “analogRead()”, available at: https://www.arduino.cc/reference/en/language/functions/analog-io/analogread/ (accessed 2 September 2019).

Arduino. (2019c), “Serial”, available at: https://www.arduino.cc/en/pmwiki.php?n=Reference/Serial (accessed 31 August 2019).

Arduino. (2019d), “Serial.read()”, available at: https://www.arduino.cc/reference/en/language/functions/communication/serial/read/ (accessed 31 August 2019).

Arozullah, A.M., Khuri, S.F., Henderson, W.G. and Daley, J. (2001), “Development and Validation of a Multifactorial Risk Index for Predicting Postoperative Pneumonia after Major Noncardiac Surgery”, *Annals of Internal Medicine*, American College of Physicians, Vol. 135 No. 10, p. 847.

AudioRhoon. (2018), “The Quintet - playing music with switches”, available at: http://www.audiorhoon.nl/english/quintet (accessed 29 July 2019).

Australian Music Examinations Board (AMEB). (2018), “Theory of Music”.

Australian Music Examinations Board (AMEB). (2019), “About AMEB”, available at: https://www.ameb.edu.au/about/about.html (accessed 9 September 2019).

de Bakker, B. (2019), “16x2 Character LCD Arduino Tutorial”, available at: https://www.makerguides.com/character-lcd-arduino-tutorial/ (accessed 9 October 2019).

Balbag, M.A., Pedersen, N.L. and Gatz, M. (2014), “Playing a Musical Instrument as a Protective Factor against Dementia and Cognitive Impairment: A Population-Based Twin Study”, *International Journal of Alzheimer’s Disease*, Vol. 2014, pp. 1–6.

Bartlett, R.H., Gazzaniga, A.B. and Geraghty, T.R. (1973), “Respiratory Maneuvers to Prevent Postoperative Pulmonary Complications”, *JAMA*, American Medical Association, Vol. 224 No. 7, p. 1017.

Bartlett, R.H., Krop, P., Hanson, E.L. and Moore, F.D. (1970), “Physiology of yawning and its application to postoperative care.”, *Surgical Forum*, Vol. 21, pp. 222–4.

Basoglu, O.K., Atasever, A. and Bacakoglu, F. (2005), “The efficacy of incentive spirometry in patients with COPD”, *Respirology*, Vol. 10 No. 3, pp. 349–353.

Bausewein, C., Booth, S., Gysels, M. and Higginson, I.J. (2008), “Non-pharmacological interventions for breathlessness in advanced stages of malignant and non-malignant diseases”, in Bausewein, C. (Ed.), *Cochrane Database of Systematic Reviews*, John Wiley & Sons, Ltd, Chichester, UK, available at:https://doi.org/10.1002/14651858.CD005623.pub2.

Bekiroğlu, T., Ovayolu, N., Ergün, Y. and Ekerbiçer, H.Ç. (2013), “Effect of Turkish classical music on blood pressure: A randomized controlled trial in hypertensive elderly patients”, *Complementary Therapies in Medicine*, Vol. 21 No. 3, pp. 147–154.

Bendixen, H., Bullwinkel, B., Hedley-Whyte, J. and Layer, M. (1964), “Atelectasis and Shunting During Spontaneous Ventilation in Anesthetized Patients”, *Anesthesiology.*, [American Society of Anesthesiologists, etc.], Vol. 25 No. 3, pp. 297–301.

Black, J. (2013), “The Origin of Music | Ancient Origins”, available at: https://www.ancient-origins.net/ancient-places-europe/origin-music-00972 (accessed 15 October 2019).

Bluth, T., Serpa Neto, A., Schultz, M.J., Pelosi, P. and Gama de Abreu, M. (2019), “Effect of Intraoperative High Positive End-Expiratory Pressure (PEEP) With Recruitment Maneuvers vs Low PEEP on Postoperative Pulmonary Complications in Obese Patients”, *JAMA*, American Medical Association, Vol. 321 No. 23, p. 2292.

Bryant, T.K. (2019), “Incentive spirometry devices by the employment of verbal simulated humanlike voices and using a tilt sensing component for ensuring patient actual use of the improved incentive spirometry devices”.

Burgoyne, J.A. and Wild, J. (2011), *An Expert Ground Truth Set for Audio Chord Recognition and Music Analysis*, available at: http://nielsen.com/us/en/industries/ (accessed 8 September 2019).

Candor Industries. (2018), “Breadboard vs PCB”, available at: https://www.candorind.com/breadboard-vs-pcb/ (accessed 17 October 2019).

Canga, B., Azoulay, R., Raskin, J. and Loewy, J. (2015), “AIR: Advances in Respiration – Music therapy in the treatment of chronic pulmonary disease”, *Respiratory Medicine*, Vol. 109 No. 12, pp. 1532–1539.

Carroll, R.G. (2007), “Pulmonary System”, *Elsevier’s Integrated Physiology*, Elsevier, pp. 99–115.

Chang, A.T., Palmer, K.R., McNaught, J. and Thomas, P.J. (2010), “Inspiratory flow rate, not type of incentive spirometry device, influences chest wall motion in healthy individuals”, *Physiotherapy Theory and Practice*, Vol. 26 No. 6, pp. 385–392.

Chen. (2019), “How to use HC-06 Bluetooth module to enable communication between Arduino and Android. HC-06 ZS 040 AT Commands, texting and LED examples”, available at: http://chenthedesignmaker.com/how-to-use-hc-06-bluetooth-module-to-enable-communication-between-arduino-and-android-hc-06-zs-040-at-commands-texting-and-led-examples/ (accessed 31 August 2019).

Cherkasova, M. V., Clark, L., Barton, J.J.S., Schulzer, M., Shafiee, M., Kingstone, A., Stoessl, A.J., et al. (2018), “Win-concurrent sensory cues can promote riskier choice”, *Journal of Neuroscience*, Society for Neuroscience, Vol. 38 No. 48, pp. 10362–10370.

Chuang, C.-Y., Han, W.-R., Li, P.-C. and Young, S.-T. (2010), “Effects of music therapy on subjective sensations and heart rate variability in treated cancer survivors: A pilot study”, *Complementary Therapies in Medicine*, Vol. 18 No. 5, pp. 224–226.

Core Electronics. (2019a), “Teensy Audio Adaptor Board”, available at: https://core-electronics.com.au/teensy-audio-adaptor-board.html (accessed 2 October 2019).

Core Electronics. (2019b), “Bluetooth Module (HC-06)”, available at: https://core-electronics.com.au/bluetooth-module-hc-06.html (accessed 14 September 2019).

Craven, J.L., Evans, G.A., Davenport, P.J. and Williams, R.H.P. (1974), “The evaluation of the incentive spirometer in the management of postoperative pulmonary complications”, *British Journal of Surgery*, John Wiley & Sons, Ltd, Vol. 61 No. 10, pp. 793–797.

David Jones Pharmacy. (2019), “Able TriFlo Inspiratory Exerciser 600 – 1200cc”, available at: http://www.davidjonespharmacy.com.au/able-triflo-inspiratory-exerciser-600-1200cc?gclid=Cj0KCQjwi7DtBRCLARIsAGCJWBoRedP-fXllEzTZy9KbZDFcN6pKOaC6lRZiUHgq-NNT9Gal2hOj6PgaAiGkEALw_wcB (accessed 21 October 2019).

Davies, O.J., Husain, T. and Stephens, R.C. (2017), “Postoperative pulmonary complications following non-cardiothoracic surgery”, *BJA Education*, Narnia, Vol. 17 No. 9, pp. 295–300.

Department for Education. (2019), “Music in schools”, available at: https://www.education.sa.gov.au/teaching/projects-and-programs/music/school-music-programs/music-schools (accessed 15 July 2019).

DigiKey Electronics. (2019a), “MPXV7002DP”, available at: https://www.digikey.com.au/product-detail/en/nxp-usa-inc/MPXV7002DP/MPXV7002DP-ND/1168436 (accessed 7 April 2019).

DigiKey Electronics. (2019b), “D6R90 F2 LFS”, available at: https://www.digikey.com.au/product-detail/en/c-k/D6R90-F2-LFS/401-1995-ND/1466352 (accessed 2 October 2019).

Dolan, R., Huh, J., Tiwari, N., Sproat, T. and Camilleri-Brennan, J. (2016), “A prospective analysis of sleep deprivation and disturbance in surgical patients”, *Annals of Medicine and Surgery*, Elsevier Ltd, Vol. 6, pp. 1–5.

Element14. (2019), “MCBBJ65 - MULTICOMP - Jumper Wire Assortment, 65 pcs of 22 AWG, Multi-Colours”, available at: https://au.element14.com/multicomp/mcbbj65/wire-gauge-22awg/dp/2396146 (accessed 2 October 2019).

Elliott, D., Carr, S. and Orme, D. (2005), “The effect of motivational music on sub-maximal exercise”, *European Journal of Sport Science*, Vol. 5 No. 2, pp. 97–106.

Eltorai, A.E.M., Baird, G.L., Eltorai, A.S., Healey, T.T., Agarwal, S., Ventetuolo, C.E., Martin, T.J., et al. (2019), “Effect of an Incentive Spirometer Patient Reminder after Coronary Artery Bypass Grafting: A Randomized Clinical Trial”, *JAMA Surgery*, American Medical Association, available at:https://doi.org/10.1001/jamasurg.2019.0520.

Eltorai, A.E.M., Baird, G.L., Pangborn, J., Eltorai, A.S., Antoci, V., Paquette, K., Connors, K., et al. (2018), “Financial impact of incentive spirometry”, *Inquiry (United States)*, SAGE Publications Inc., Vol. 55, available at:https://doi.org/10.1177/0046958018794993.

Eltorai, A.E.M., Szabo, A.L., Antoci, V., Ventetuolo, C.E., Elias, J.A., Daniels, A.H. and Hess, D.R. (2018), “Clinical effectiveness of incentive spirometry for the prevention of postoperative pulmonary complications”, *Respiratory Care*, American Association for Respiratory Care, Vol. 63 No. 3, pp. 347–352.

Epperson, G. (2019), “Music”, available at: https://www.britannica.com/art/music (accessed 15 July 2019).

Evans, J.A. and Whitelaw, W.A. (2009), “The Assessment of Maximal Respiratory Mouth Pressures In Adults”, *Respiratory Care*, Vol. 54 No. 10, pp. 1348–1359.

Ferguson, S. (2015), “Doodle undoing the last change or erasing it.”, available at: https://groups.google.com/forum/#!topic/mitappinventortest/v6TfdKgOoRE (accessed 28 September 2019).

Fernandez-Bustamante, A., Frendl, G., Sprung, J., Kor, D.J., Subramaniam, B., Ruiz, R.M., Lee, J.W., et al. (2017), “Postoperative pulmonary complications, early mortality, and hospital stay following noncardiothoracic surgery: A multicenter study by the perioperative research network investigators”, *JAMA Surgery*, American Medical Association, Vol. 152 No. 2, pp. 157–166.

Fernandez-Bustamante, A., Schoen, J. and Vidal Melo, M.F. (2017), “Incentive Spirometry After Bariatric Surgery”, *JAMA Surgery*, American Medical Association, Vol. 152 No. 10, p. 984.

Ferrer, A.J. (2007), *The Effect of Live Music on Decreasing Anxiety in Patients Undergoing*

Chemotherapy Treatment, *Journal of Music Therapy; Fall*, Vol. 44, available at: https://search.proquest.com/docview/223557371/fulltextPDF/65AF4519911460DPQ/1?accountid=10910 (accessed 17 July 2019).

Flaticon. (2014), “Quavers Pair Icon”, available at: https://www.freepik.com/free-icon/quavers-pair_740298.htm (accessed 9 October 2019).

Forrest, P. (2016), “Postoperative Pulmonary Complications”, *Perioperative Medicine – Current Controversies*, Springer International Publishing, Cham, pp. 211–224.

Frid, E. (2019), “Accessible Digital Musical Instruments—A Review of Musical Interfaces in Inclusive Music Practice”, *Multimodal Technologies and Interaction*, Vol. 3 No. 3, p. 57.

Gehlhaar, R., Rodrigues, P.M., Girão, L.M. and Penha, R. (2014), “Instruments for Everyone: Designing New Means of Musical Expression for Disabled Creators”, Springer, Berlin, Heidelberg, pp. 167–196.

Gratton, A. (2011), “The Hairless MIDI to Serial Bridge”, available at: http://projectgus.github.io/hairless-midiserial/ (accessed 2 September 2019).

Gritti, C. (2016), “Arduino & Ableton Live - How to connect an Arduino MIDI Controller to Ableton Live 9 on Windows”, available at: https://www.youtube.com/watch?v=BzLKtor-6Vg (accessed 2 September 2019).

Gritti, C. (2017), “Arduino Midi Simple Example”, available at: https://github.com/nerdityourself/ArduinoMidiSimpleExample (accessed 2 September 2019).

GroovTube. (2018), “Apps”, available at: https://www.groovtube.nl/en/groovtube-apps/ (accessed 24 July 2019).

GroovTube. (2019), “GroovTube starter pack”, available at: https://www.groovtube.nl/en/product/groovtube-starter-pack/ (accessed 24 July 2019).

Haas, A. (2011), *The Art of Playing Trumpet in the Upper Register*, University of Miami, available at: https://scholarlyrepository.miami.edu/oa_dissertations/554 (accessed 7 April 2019).

Hall, J.C., Tarala, R.A., Tapper, J. and Hall, J.L. (1996), “Prevention of respiratory complications after abdominal surgery: a randomised clinical trial”, *BMJ*, Vol. 312 No. 7024, pp. 148–152.

Harkden, D.E. (1946), “A review of the activities of the thoracic center for the III and IV hospital groups, 160th General Hospital European Theater of Operations, June 10, 1944 to Jan. 1, 1945.”, *The Journal of Thoracic Surgery*, Vol. 15, pp. 31–43.

Healthy Kin. (2019), “Voldyne Incentive Spirometer”, available at: https://www.healthykin.com/p-5364-voldyne-incentive-spirometer.aspx (accessed 6 April 2019).

Hendon, C. and Bohon, L.M. (2008), “Hospitalized children’s mood differences during play and music therapy”, *Child: Care, Health and Development*, Vol. 34 No. 2, pp. 141–144.

Hobbs, D.A. and Worthington-Eyre, B. (2008), “The efficacy of combining augmented reality and music therapy with traditional teaching-Preliminary results”, in Thajchayapong, P., Koh, Z., Phantachat, W. and Ang, W.T. (Eds.), *ICREATe '08 Proceedings of the 2nd International Convention on Rehabilitation Engineering & Assistive Technology*, Singapore Therapeutic, Assistive & Rehabilitative Technologies (START) Centre Kaki Bukit TechPark II, Bangkok, Thailand, pp. 241–244.

Hochgraf, C. (2016), "An easy way to add a bar graph to your app in App Inventor", available at: https://appinventor.mit.edu/explore/blogs/karen/2016/10/easy.html (accessed 27 September 2019).

Hollis, B. (2017), "Scales and Key Signatures", available at: https://musictheoryonline.com/natural-harmonic-and-melodic-minor/ (accessed 9 September 2019).

Hough, A. (1996), *Physiotherapy in Respiratory Care*, Springer US, Boston, MA, available at:https://doi.org/10.1007/978-1-4899-3049-1.

Human Instruments. (2019), "Human Instruments - About", available at: https://www.humaninstruments.co.uk/about (accessed 12 August 2019).

Inspired Acoustics. (2019), "MIDI Note Numbers and Center Frequencies", available at: https://www.inspiredacoustics.com/en/MIDI_note_numbers_and_center_frequencies (accessed 15 September 2019).

Japan Trend Shop. (2019), "EZ TP Electronic Trumpet by Yamaha", available at: https://www.japantrendshop.com/ez-tp-electronic-trumpet-by-yamaha-p-181.html (accessed 29 July 2019).

Johnson, R.G., Arozullah, A.M., Neumayer, L., Henderson, W.G., Hosokawa, P. and Khuri, S.F. (2007), "Multivariable Predictors of Postoperative Respiratory Failure after General and Vascular Surgery: Results from the Patient Safety in Surgery Study", *Journal of the American College of Surgeons*, Vol. 204 No. 6, pp. 1188–1198.

Kelstrom, J.M. (1998), *The Untapped Power of Music: Its Role in the Curriculum and Its Effect on Academic Achievement The Decision to Support Music Cannot Be Made with-out Knowing Music's Effect on Academic Achievement and Its Contribution to a Student's Education*, Anderson, available at: https://journals.sagepub.com/doi/pdf/10.1177/019263659808259707 (accessed 15 July 2019).

Khuri, S.F., Henderson, W.G., DePalma, R.G., Mosca, C., Healey, N.A., Kumbhani, D.J. and Participants in the VA National Surgical Quality Improvement Program. (2005), "Determinants of Long-Term Survival After Major Surgery and the Adverse Effect of Postoperative Complications", *Transactions of the ... Meeting of the American Surgical Association*, Vol. 123 No. NA;, pp. 32–48.

Kigin, C.M. (1981), "Chest Physical Therapy for the Postoperative or Traumatic Injury Patient", *Physical Therapy*, Vol. 61 No. 12, pp. 1724–1736.

Kojouri, K. (2013), "Monitoring Incentive Spirometry", United States.

Laycock, R. (2019), "How wearables can save you money on insurance premiums", available at: https://www.finder.com.au/save-on-life-insurance (accessed 17 October 2019).

Lederer, D.H., Van De Water, J.M. and Indech, R.B. (1980), “Which deep breathing device should the postoperative patient use?”, *Chest*, Vol. 77 No. 5, pp. 610–613.

Little Bird Electronics. (2019), “Teensy 3.6”, available at: https://www.littlebird.com.au/products/teensy-3-6-ac9708b6-3c40-4dc7-81c5-d0fda168a606 (accessed 2 October 2019).

Lunardi, A.C., Porras, D.C., Barbosa, R.C.C., Paisani, D.M., da Silva, C.C.B.M., Tanaka, C. and Carvalho, C.R.F. (2014), “Effect of volume-oriented versus flow-oriented incentive spirometry on chest wall volumes, inspiratory muscle activity, and thoracoabdominal synchrony in the elderly”, *Respiratory Care*, Daedalus Enterprises Inc., Vol. 59 No. 3, pp. 420–426.

MacFarlane, P. (2016), “Chord Construction”, available at: https://www.guitarlessonworld.com/lessons/chord-construction/ (accessed 20 October 2019).

van Manen, J.G. (2002), “Risk of depression in patients with chronic obstructive pulmonary disease and its determinants”, *Thorax*, Vol. 57 No. 5, pp. 412–416.

MarketsAndMarkets Research Private. (2018), “Spirometer Market”, available at: https://www.marketsandmarkets.com/Market-Reports/spirometer-market-18015659.html?gclid=Cj0KCQjw19DlBRCSARIsAOnfRejcMUFcXWoKWJ3Lv-cdauugsJgxCHbIOf6wSSPNwTDuhzWcokk8dg8aAolgEALw_wcB (accessed 15 April 2019).

Matossian, V. and Gehlhaar, R. (2015), “Human Instruments: Accessible Musical Instruments for People with Varied Physical Ability.”, *Studies in Health Technology and Informatics*, Vol. 219, pp. 202–7.

McConnell, D.H., Maloney, J. V and Buckberg, G.D. (1974), “Postoperative intermittent positive-pressure breathing treatments. Physiological considerations.”, *The Journal of Thoracic and Cardiovascular Surgery*, Vol. 68 No. 6, pp. 944–52.

McCord, K.A. (1990), *Children With Special Needs Compose Using Music Technology*, *Journal of Technology in Music Izaming*, Vol. I, Scripp & Meyaard, available at: http://www.atmimusic.com/wp-content/uploads/2013/05/JTML.1.2b_McCord_Children-with-special-needs-compose-using-music-technology.pdf (accessed 29 July 2019).

Mead, J. and Collier, C. (1959), “Relation of volume history of lungs to res-piratory mechanics in anesthetized dogs’”, *Journal of Applied Physiology*, Vol. 14 No. 5, available at: www.physiology.org/journal/jappl (accessed 28 July 2019).

Mellmer, R. (2017), “Teensy Audio Wavetable Tutorial”, available at: https://www.youtube.com/watch?v=5laaNHLhS98 (accessed 15 September 2019).

Méndez-Téllez, P.A., Rodriguez-Paz, J.M. and Dorman, T. (2002), “The use of Intermittent Positive Pressure Breathing in Adults: A Review of the Evidence”, *Anesthesiology*, Vol. 97, p. B29.

Michigan Technological University. (1998), “Frequencies of Musical Notes”, available at: http://pages.mtu.edu/~suits/notefreqs.html (accessed 2 September 2019).

Microsoft. (2018), “Header files (C++)”, available at: https://docs.microsoft.com/en-us/cpp/cpp/header-files-cpp?view=vs-2019 (accessed 21 October 2019).

MIDI Association. (2019), “Summary of MIDI Messages”, available at: https://www.midi.org/specifications-old/item/table-1-summary-of-midi-message (accessed 2 September 2019).

Misic, P., Arandjelovic, D., Stanojkovic, S., Vladejic, S. and Mladenovic, J. (2010), “Music therapy”, *European Psychiatry*, Elsevier Masson, Vol. 25, p. 839.

Miskovic, A. and Lumb, A.B. (2017), “Postoperative pulmonary complications”, *British Journal of Anaesthesia*, Vol. 118 No. 3, pp. 317–334.

MIT App Inventor. (2019), “Explore MIT App Inventor”, available at: http://appinventor.mit.edu/about-us (accessed 21 October 2019).

Morrison, J. and Marshall, S. (2005), “Morrison Digital Trumpet”, available at: http://www.digitaltrumpet.com.au/.

Moses, R., Thomas, F., Mummery, V., Da Costa, J., O’Farrell, C., Friend, S. and Coughlan, L. (2016), “The use of intermittent positive pressure breathing to prevent secondary pulmonary complications in patients with blunt chest wall trauma”, *Physiotherapy*, Elsevier, Vol. 102, pp. e206–e207.

Mouser Electronics. (2019), “MPXV7002DP”, available at: https://au.mouser.com/ProductDetail/NXP-Freescale/MPXV7002DP?qs=N2XN0KY4UWXOkdT07YYQaw%3D%3D&gclid=Cj0KCQjw8svsBRDqARIsAHKVyqHOmN19jPAZH-YlxcwO0wjqfHXMvZ6amJRwuMpzB8oQEFRvR7_OH8YaAhTvEALw_wcB (accessed 2 October 2019).

My Breath My Music. (2007), “The Magic Flute – The Head Sets the Tone”, available at: http://mybreathmymusic.com/en/magic-flute (accessed 19 July 2019).

Narayanan, A.L.T., Ayob, M.A., Nordin, N., Harris, A.R.A. and Supriyanto, E. (2016), “Development of a Novel Device for Monitoring Incentive Spirometry Performance”, *Sains Malaysiana*, Vol. 45 No. 7, pp. 1121–1129.

Narayanan, A.L.T., Hamid, S.R.G.S. and Supriyanto, E. (2016), “Evidence regarding patient compliance with incentive spirometry interventions after cardiac, thoracic and abdominal surgeries: A systematic literature review.”, *Canadian Journal of Respiratory Therapy : CJRT = Revue Canadienne de La Therapie Respiratoire : RCTR*, Vol. 52 No. 1, pp. 17–26.

do Nascimento Junior, P., Módolo, N.S., Andrade, S., Guimarães, M.M., Braz, L.G. and El Dib, R. (2014), “Incentive spirometry for prevention of postoperative pulmonary complications in upper abdominal surgery”, *Cochrane Database of Systematic Reviews*, available at:https://doi.org/10.1002/14651858.CD006058.pub3.

Nedelkovski, D. (2015), “Arduino LCD Tutorial”, available at: https://howtomechatronics.com/tutorials/arduino/lcd-tutorial/ (accessed 7 October 2019).

Nemesh, B. (2016), *Family-Based Music Therapy: Family Therapists’ Perspectives*, available at: https://search-proquest-com.ezproxy.flinders.edu.au/docview/1762587166/fulltextPDF/FE866750EA2F4938PQ/1?accountid=10910 (accessed 18 July 2019).

Nypaver, A. (2018), “Monophonic in Music: Definition & Examples Video with Lesson Transcript | Study.com”, available at: https://study.com/academy/lesson/monophonic-in-music-definition-examples.html (accessed 21 October 2019).

Opentip.com. (2019), “Voldyne Volumetric-Incentive Spirometer”, available at: https://www.opentip.com/product.php?products_id=1473511 (accessed 1 September 2019).

Overend, T.J., Anderson, C.M., Lucy, S.D., Bhatia, C., Jonsson, B.I. and Timmermans, C. (2001), “The Effect of Incentive Spirometry on Postoperative Pulmonary Complications: A Systematic Review”, *Chest*, Vol. 120 No. 3, pp. 971–978.

Paisani, D. de M., Lunardi, A.C., da Silva, C.C.B.M., Cano Porras, D., Tanaka, C. and Fernandes Carvalho, C.R. (2013), “Volume rather than flow incentive spirometry is effective in improving chest wall expansion and abdominal displacement using optoelectronic plethysmography”, *Respiratory Care*, Vol. 58 No. 8, pp. 1360–1366.

Pantel, H., Hwang, J., Brams, D., Schnelldorfer, T. and Nepomnayshy, D. (2017), “Effect of incentive spirometry on postoperative hypoxemia and pulmonary complications after bariatric surgery a randomized clinical trial”, *JAMA Surgery*, American Medical Association, Vol. 152 No. 5, pp. 422–428.

Pasquina, P., Tramér, M.R., Granier, J.-M. and Walder, B. (2006), “Respiratory Physiotherapy To Prevent Pulmonary Complications After Abdominal Surgery”, *Chest*, Vol. 130 No. 6, pp. 1887–1899.

Paterson, J. (2019), “Composer Timelines for different Classical Music Periods”, available at: https://www.mfiles.co.uk/composer-timelines-classical-periods.htm (accessed 15 July 2019).

Pazzianotto-Forti, E.M., Moraes da Costa, C., Merino, D., Rocha, M. and Rasera-Júnior, I. (2015), “Breathing exercises by flow-oriented incentive spirometry in postoperative bariatric surgery”, European Respiratory Society (ERS), p. PA3542.

Pereira, M.C., Porras, D.C., Lunardi, A.C., Da Silva, C.C.B.M., Barbosa, R.C.C., Cardenas, L.Z., Pletsch, R., et al. (2017), “Thoracoabdominal asynchrony: Two methods in healthy, COPD, and interstitial lung disease patients”, *PLoS ONE*, Public Library of Science, Vol. 12 No. 8, available at:https://doi.org/10.1371/journal.pone.0182417.

Pfenninger, J. and Roth, F. (1977), *Intermittent Positive Pressure Breathing (IPPB) Versus Incentive Spirometer (IS) Therapy in the Postoperative Period*, *Intens. Care Med*, Vol. 3, available at: https://link-springer-com.ezproxy.flinders.edu.au/content/pdf/10.1007%2FBF01641120.pdf (accessed 25 July 2019).

PJRC. (2017), “Using USB MIDI”, available at: https://www.pjrc.com/teensy/td_midi.html (accessed 8 September 2019).

PJRC. (2018), “Teensy Pinouts Assignments”, available at: https://www.pjrc.com/teensy/pinout.html (accessed 8 September 2019).

PJRC. (2019a), “Audio Adaptor Board for Teensy 3.0 - 3.6”, available at:

https://www.pjrc.com/store/teensy3_audio.html (accessed 15 September 2019).

PJRC. (2019b), “Audio Tutorial Kit”, available at: https://www.pjrc.com/store/audio_tutorial_kit.html (accessed 18 October 2019).

PJRC. (2019c), “Teensy USB Development Board”, available at: https://www.pjrc.com/store/teensy36.html (accessed 4 August 2019).

PJRC. (2019d), “Teensy 4.0”, available at: https://www.pjrc.com/store/teensy40.html (accessed 10 October 2019).

Rahman, S.S. (2018), “AT Command Mode of HC-05 and HC-06 Bluetooth Module: 5 Steps”, *Instructable Circuits*, available at: https://www.instructables.com/id/AT-command-mode-of-HC-05-Bluetooth-module/ (accessed 31 August 2019).

Renault, J.A., Costa-Val, R., Rosseti, M.B. and Houri Neto, M. (2009), “Comparison between deep breathing exercises and incentive spirometry in the postoperative period of myocardial revascularization surgery”, *Brazilian Journal of Cardiovascular Surgery VO - 24*, Sociedade Brasileira de Cirurgia Cardiovascular, No. 2, p. 165.

Restrepo, R.D., Farrc, R., Wettstein, R., Rrt, M., Wittnebel, L., Rrt, M., Tracy, M., et al. (2011), “AARC Clinical Practice Guideline Incentive Spirometry: 2011”, Vol. 56 No. 10, available at:https://doi.org/10.4187/respcare.01471.

Rose, D., Jones Bartoli, A. and Heaton, P. (2018), “Learning a musical instrument can benefit a child with special educational needs.”, *Psychomusicology: Music, Mind, and Brain*, Vol. 28 No. 2, pp. 71–81.

Rose, J., Weiser, T.G., Hider, P., Wilson, L., Gruen, R.L. and Bickler, S.W. (2015), “Estimated need for surgery worldwide based on prevalence of diseases: a modelling strategy for the WHO Global Health Estimate.”, *The Lancet. Global Health*, NIH Public Access, Vol. 3 Suppl 2 No. Suppl 2, pp. S13-20.

Ruscic, K.J., Grabitz, S.D., Rudolph, M.I. and Eikermann, M. (2017), “Prevention of respiratory complications of the surgical patient: Actionable plan for continued process improvement”, *Current Opinion in Anaesthesiology*, Lippincott Williams and Wilkins, 1 June.

Ruud van der Wel. (2018), “Respiratory Muscle Training”, available at: https://www.groovtube.nl/en/respiratory-muscle-training-groovtube/ (accessed 24 July 2019).

Sabaté, S., Mazo, V. and Canet, J. (2014), “Predicting postoperative pulmonary complications: Implications for outcomes and costs”, *Current Opinion in Anaesthesiology*, Lippincott Williams and Wilkins.

Sengupta, S. (2015), “Post-operative pulmonary complications after thoracotomy”, *Indian Journal of Anaesthesia*, Vol. 59 No. 9, p. 618.

Shander, A., Fleisher, L.A., Barie, P.S., Bigatello, L.M., Sladen, R.N. and Watson, C.B. (2011), “Clinical and economic burden of postoperative pulmonary complications: Patient safety summit on definition, risk-reducing interventions, and preventive strategies”, *Critical Care Medicine*, Lippincott Williams and Wilkins.

Simplifying Theory. (2014), “Music Scales”, available at: http://www.simplifyingtheory.com/music-scales/ (accessed 8 September 2019).

Smith, J., Kreisman, H., Colacone, A., Fox, J. and Wolkove, N. (1989), *Sensation of Inspired Volumes and Pressures in Professional Wind Instrument Players*, available at: www.physiology.org/journal/jappl (accessed 17 July 2019).

Special Needs Computer Solutions Inc. (2019), “Sip/Puff Switch”, available at: https://www.specialneedscomputers.ca/index.php?l=product_detail&p=233 (accessed 3 October 2019).

Stark Medical Pty Ltd. (2019), “RespiVol 5000mL Incentive Spirometer”, available at: https://www.starkmed.com.au/products/respivol-5000ml-incentive-spirometer?gclid=Cj0KCQjww47nBRDlARIsAEJ34bmQPGq8GZB6-opWjNRzLmv8ebMDQJVY6fDZQybC4lJ-ct7BfNKEYCsaAiD7EALw_wcB (accessed 1 September 2019).

Stevens, G.A., White, R.A., Flaxman, S.R., Price, H., Jonas, J.B., Keeffe, J., Leasher, J., et al. (2013), “Global prevalence of vision impairment and blindness: Magnitude and temporal trends, 1990-2010”, *Ophthalmology*, Vol. 120 No. 12, pp. 2377–2384.

Strix SoundFont Team. (2018), “Synth Brass 2”, available at: https://musical-artifacts.com/artifacts?formats=sf2&tags=brass (accessed 15 September 2019).

Talamini, F., Altoè, G., Carretti, B. and Grassi, M. (2017), “Musicians have better memory than nonmusicians: A meta-analysis”, available at:https://doi.org/10.1371/journal.pone.0186773.

The Australian Curriculum. (2019), “Music”, available at: https://www.australiancurriculum.edu.au/f-10-curriculum/the-arts/music/?year=12746&year=12747&year=12748&year=12749&year=12750&capability=ignore&capability=Literacy&capability=Numeracy&capability=Information+and+Communication+Technology+%28ICT%29+Capabili (accessed 15 July 2019).

The MIDI Association. (2016), “MIDI Tutorials”, available at: https://www.midi.org/articles-old/tutorials (accessed 10 October 2019).

Thomas, J.A., McIntosh, J.M. and Dean, E. (1994), “Are incentive spirometry, intermittent positive pressure breathing, and deep breathing exercises effective in the prevention of postoperative pulmonary complications after upper abdominal surgery? A systematic overview and meta-analysis”, *Physical Therapy*, Oxford University Press, Vol. 74 No. 1, pp. 3–17.

Thoren, L. (1954), “Post-operative pulmonary complications: observations on their prevention by means of physiotherapy.”, *Acta Chirurgica Scandinavica*, Vol. 107 No. 2–3, pp. 193–205.

Tomich, G.M., França, D.C., Diório, A.C.M., Britto, R.R., Sampaio, R.F. and Parreira, V.F. (2007), “Breathing pattern, thoracoabdominal motion and muscular activity during three breathing exercises”, *Brazilian Journal of Medical and Biological Research*, Associacao Brasileira de Divulgacao Cientifica, Vol. 40 No. 10, pp. 1409–1417.

Touch the Future. (2019), “Magic Flute”, available at: https://touchthefuture.us/product/magic-flute/ (accessed 24 July 2019).

Tufts University School of Medicine. (2018), “Clinical Affiliates”, available at: https://medicine.tufts.edu/education/doctor-medicine/clinical-affiliates (accessed 4 October 2019).

Twyford, K. (2012), “Getting to know you: Peer and staff perceptions of involvement in inclusive music therapy groups with students with special educational needs in mainstream school settings”, *The New Zealand Journal of Music Therapy*, Vol. 10, pp. 39–73.

U.S. National Library of Medicine. (2017), “Using an incentive spirometer”.

U.S. National Library of Medicine. (2019), “The Digital Incentive Spirometer (DIS): Improving Adherence to Incentive Spirometry (DIS)”, available at: https://clinicaltrials.gov/ct2/show/study/NCT03686631.

Uggla, L., Bonde, L., Svahn, B., Remberger, M., Wrangsjö, B. and Gustafsson, B. (2016), “Music therapy can lower the heart rates of severely sick children”, *Acta Paediatrica*, John Wiley & Sons, Ltd (10.1111), Vol. 105 No. 10, pp. 1225–1230.

Vaughn, K. (2000), *Music and Mathematics: Modest Support for the Oft-Claimed Relationship*, *Journal of Aesthetic Education; Fall*, Vol. 34, available at: https://search-proquest-com.ezproxy.flinders.edu.au/docview/220653269/fulltextPDF/92B1509C47EE41BBPQ/1?accountid=10910 (accessed 16 July 2019).

VexFlow. (2014), “trumpet-mp3”, available at: http://static.vexflow.com/soundfont/7/trumpet-mp3/ (accessed 15 September 2019).

Walters Wright, L. (2019), “Musical Instruments and the Motor Skills They Require”, available at: https://www.understood.org/en/learning-attention-issues/child-learning-disabilities/movement-coordination-issues/musical-instruments-and-the-motor-skills-they-require (accessed 29 July 2019).

Ward, R., Danziger, F., Bonica, J., Allen, G. and Bowes, J. (1967), “An Evaluation of Postoperative Respiratory Maneuvers”, *Survey of Anesthesiology*, Williams & Wilkins, Vol. 11 No. 2, p. 160.

Weiser, T.G., Regenbogen, S.E., Thompson, K.D., Haynes, A.B., Lipsitz, S.R., Berry, W.R. and Gawande, A.A. (2008), “An estimation of the global volume of surgery: a modelling strategy based on available data”, *Www.Thelancet.Com*, Vol. 372, p. 139.

Westerdahl, E., Lindmark, B., Bryngelsson, I. and Tenling, A. (2003), “Pulmonary function 4 months after coronary artery bypass graft surgery”, *Respiratory Medicine*, W.B. Saunders Ltd, Vol. 97 No. 4, pp. 317–322.

Westwood, K., Griffin, M., Roberts, K., Williams, M., Yoong, K. and Digger, T. (2007), “Incentive spirometry decreases respiratory complications following major abdominal surgery”, *The Surgeon*, Elsevier, Vol. 5 No. 6, pp. 339–342.

Worldometers. (2019), “Current World Population”, available at: https://www.worldometers.info/world-population/ (accessed 27 September 2019).

Yamaha. (2017), “WX5”, available at: https://usa.yamaha.com/products/music_production/midi_controllers/wx5/index.html (accessed 29 July 2019).

Yamaha Corporation. (2017), “Trumpet - Musical Instrument Guide”, available at: https://www.yamaha.com/en/musical_instrument_guide/trumpet/ (accessed 29 July 2019).

Zanini, C.R. de O., Jardim, P.C.B.V., Salgado, C.M., Nunes, M.C., Urzêda, F.L. de, Carvalho, M.V.C., Pereira, D.A., et al. (2009), “Music therapy effects on the quality of life and the blood pressure of hypertensive patients.”, *Arquivos Brasileiros de Cardiologia*, Vol. 93 No. 5, pp. 534–40.

9 Appendices

9.1 Appendix A - MPXV7002 Data Sheet

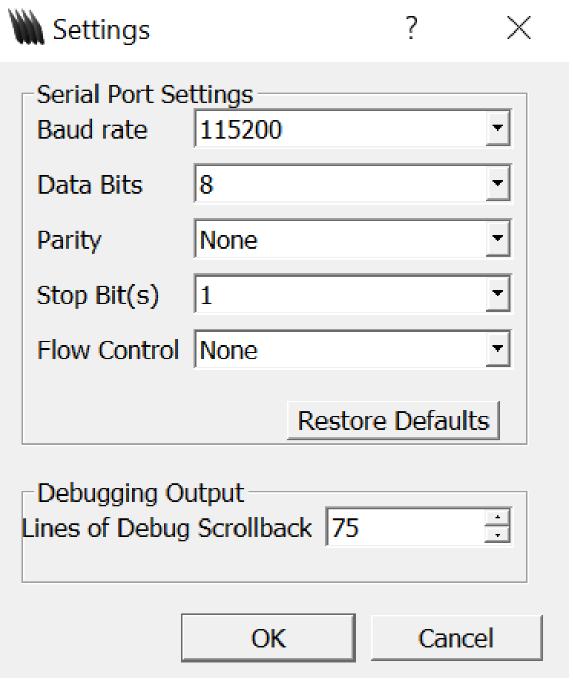
Appendix A has been removed due to copyright restrictions

9.2 Appendix B – MIDI

MIDI files in isolation do not contain audio, the files instead include a list of commands such pitch and tempo that can then be referenced by a sample or synthesised sound to produce audio (The MIDI Association, 2016). The standard MIDI file is a format type that enables music sequences files to be transported, exported and saved. The header of MIDI files detail which file format the music sequence is saved, usually type 0 for a single track to represent the entire performance or type 1 for individually layered tracks arranged synchronously (The MIDI Association, 2016).

The ATmega16U2 microcontroller (MCU) onboard the Arduino Uno does not contain a USB module. The Arduino was therefore not recognised as a MIDI instrument and could not directly interface with music production software (Gratton, 2011). To convert the Arduino Uno into a MIDI class compliant device, it was required to modify the firmware of the MCU. To prevent the possibility of unintentionally causing permanent damage to the Arduino or undesirably modifying its function, the MCU firmware was not modified.

The Hairless MIDI preferences were configured in accordance with the recommendations of Gritti (**Figure 72**). The baud rate of 115200 is optimal for communication between the Arduino and Ableton Live (Gritti, 2017), the selected digital musical production software package.


Settings

Serial Port Settings

Baud rate	115200
Data Bits	8
Parity	None
Stop Bit(s)	1
Flow Control	None

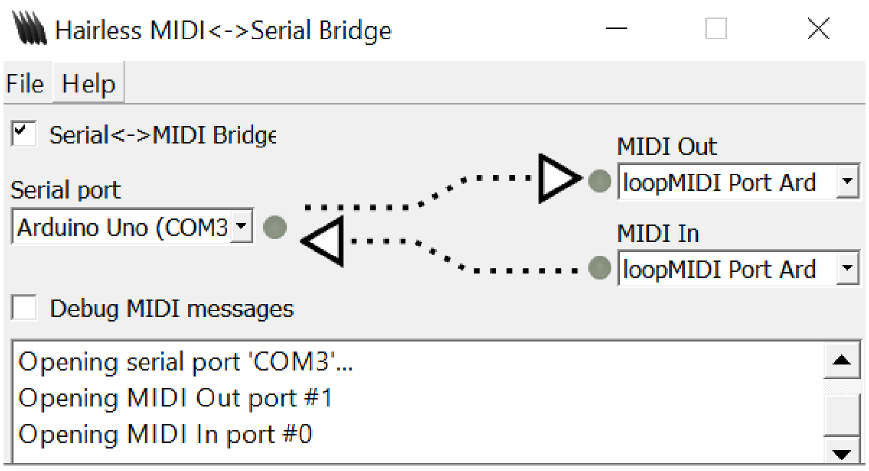
Debugging Output

Lines of Debug Scrollback 75

Figure 72*: Hairless MIDI Preferences*

The Arduino Uno was connected to COM3 port and as shown in **Figure 73**, was selected as the serial port by which the information would be transferred across. The MIDI output and MIDI input were assigned to loopMIDI, a secondary software application capable of translating Windows MIDI data between two software applications.

LoopMIDI is a Virtual MIDI pass-through driver and was configured by creating a MIDI port named `loopMIDI Port Ard` (**Figure 73**).



[✓] Serial<->MIDI Bridge

Serial port: Arduino Uno (COM3

MIDI Out: loopMIDI Port Ard

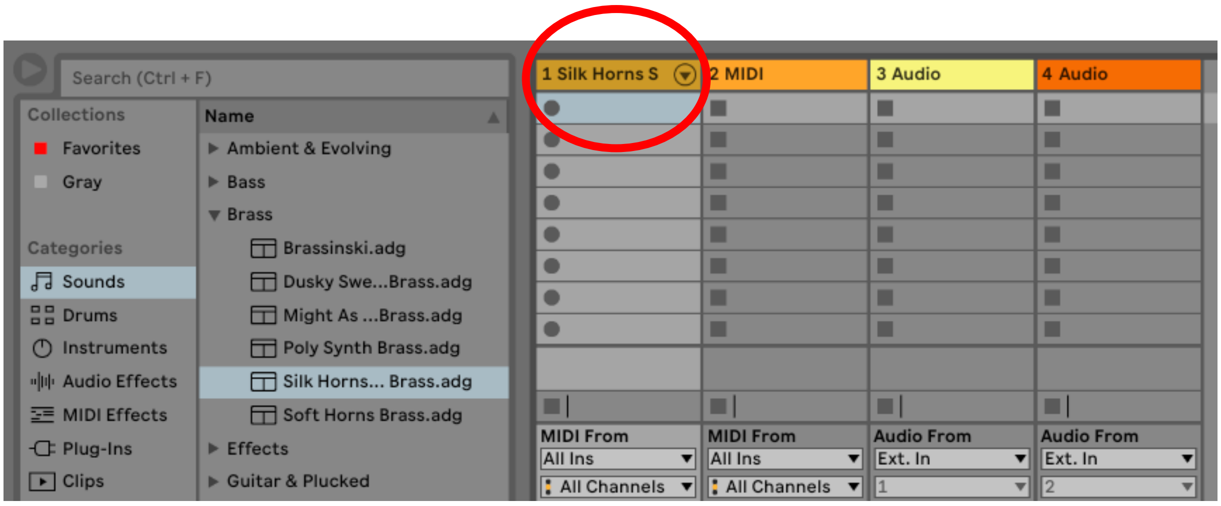
MIDI In: loopMIDI Port Ard

[ ] Debug MIDI messages

```
Opening serial port 'COM3'...
Opening MIDI Out port #1
Opening MIDI In port #0
```

***Figure 73**: Hairless MIDI Interface*

To configure Ableton Live 10 Lite to accept MIDI data from the Arduino, the preferences were modified such that the `track` and `remote` inputs were set to `on`. Hence, Ableton Live was instructed to accept non-standard MIDI controllers, which did not map notes in a format such as an electronic keyboard or other standard MIDI controller (Gritti, 2016). To ensure the desired channel appropriately processed the MIDI command, the track was first defined as a MIDI Track, not an Audio Track. The track was then set to accept MIDI data from all inputs and all channels and output the audio to the Master to enable playback. Ableton Live does not possess an inbuilt trumpet instrument, however, a `Silk Horn Synth Brass` was identified as sounding most similar to a trumpet and was deemed acceptable for testing purposes. As highlighted by the red ellipse in **Figure 74**, the `Silk Horn Synth Brass` instrument was selected within the Ableton Live interface.



MIDI From: All Ins, All Channels
MIDI From: All Ins, All Channels
Audio From: Ext. In, 1
Audio From: Ext. In, 2

***Figure 74**: Ableton Live Musical Instrument Selection*

The Arduino Uno could now be configured to communicate with a MIDI software package, however, as the firmware was not modified an alternative to the MIDIUSB library was required to send MIDI commands. An example program of how to send MIDI messages with an Arduino was provided on GitHub and comprised of the function `MIDI_TX()` (Gritti, 2017). Three parameters were passed through `MIDI_TX()`, the `MESSAGE`, `PITCH` and `VELOCITY`, which correlated to the MIDI status, the note and the speed at which the note was pressed, respectively.

The status byte was primarily utilised to send a Note On or Note Off event. The status byte was separated into two 4-bit nibbles; the most significant nibble was set to either 1001 or 1000 for a Note On or Note Off command, respectively, while the least significant nibble ranged from 0 – 15 and represented MIDI channels 1 – 16 (MIDI Association, 2019). A Note On command to channel 1 was therefore 0b10010000. The Pitch byte ranges from 0b00000000 to 0b01111111, where 0b00111100, or 0d60, correlates to C_4, or Middle C. The Velocity byte ranges from 0b00000000 to 0b01111111, where a higher value correlates to a faster note press and will, therefore, result in a louder sound (MIDI Association, 2019); a default velocity value of 0b01111111 was assigned.

The `MIDI_TX()` function was confirmed to send a MIDI command and the pushbutton environment from the tutorial was replicated. By modifying the code obtained from GitHub, the pushbutton input was converted to a potentiometer which was scaled between E_3 - Bb_5 and confirmed to function as expected. The original code contained a `for` loop which referenced either the lower or higher of the two predefined notes, depending on the state of the loop variable, at the time of the pushbutton activation. The Note On command will only be sent if one of the two pushbuttons were pressed. The value of `pin` will be utilised in the call, `PadNote[pin]`, which sends a pitch number of either 36 or 38, correlating to the note C_2 or D_2.

In the original program, two buttons controlled the only two notes accessible to be output, however, the code was modified such that a button press instigated the analog value to be read and subsequently assigned to one of the 31 notes stored in `PadNote`. The button connected to digital Pin 7 was maintained as a control method, to regulate the timing of the note length and prevent the MIDI commands from continuously being sent. The potentiometer was read from analog Pin A0, and an additional pushbutton was connected to digital Pin 7 to complete the circuit (**Figure 75**).

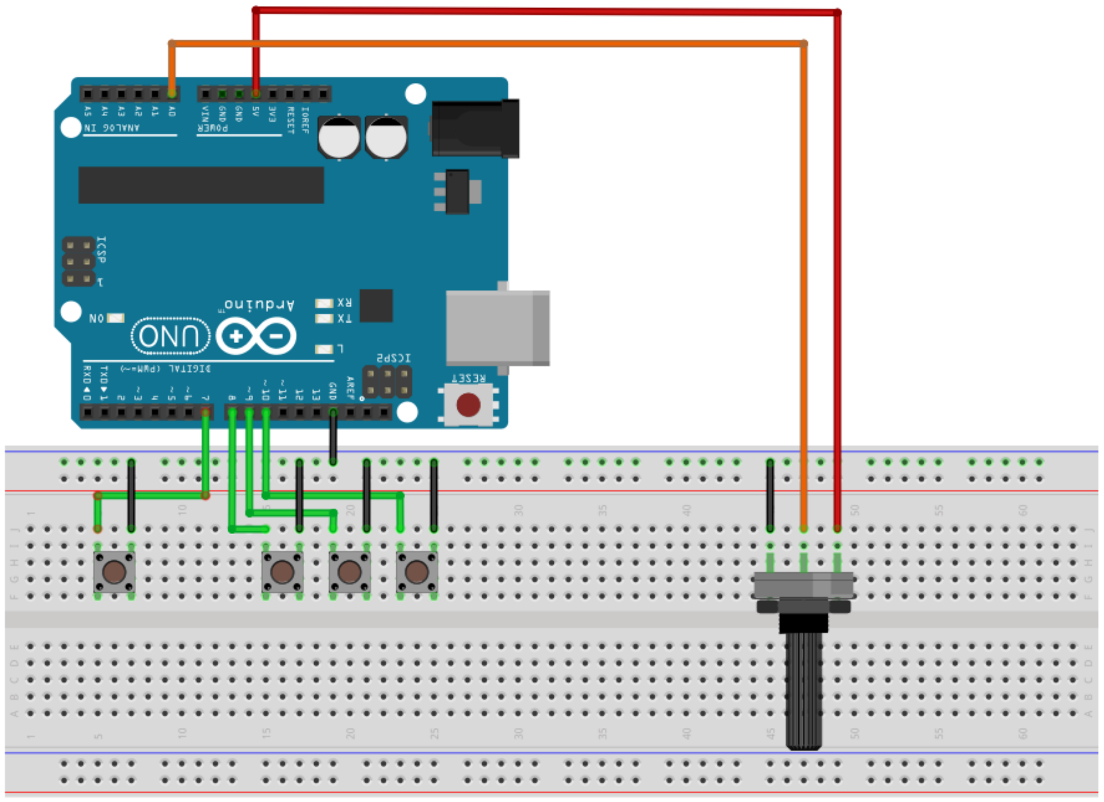


Figure 75*: Preparation for Ableton Live Connections*

The sound of a single note played by a horn was audibly confirmed for the length of the pushbutton press. Although the note could be dynamically set based upon the position of the potentiometer, the note sent to Ableton Live was read at the time of the pushbutton press. To change the note, the pushbutton was required to be released, and pressed again, if the potentiometer was moved to a new position, a different note would be played.

The pushbutton activation was removed to enable smooth transitions between different notes. However, the high number of MIDI commands caused Hairless MIDI to crash, even by adding a one-second delay, therefore terminating the connection between the Arduino and Ableton Live.

The leftmost track of **Figure 76**, representing the `Silk Horn Synth Brass` instrument, was configured to receive a MIDI signal from Channel 1; the orange box to the left of `Ch. 1` indicates that MIDI data is being received. The green bar in the leftmost track represents the volume of the audio received, which was sent to the rightmost `Master` track.

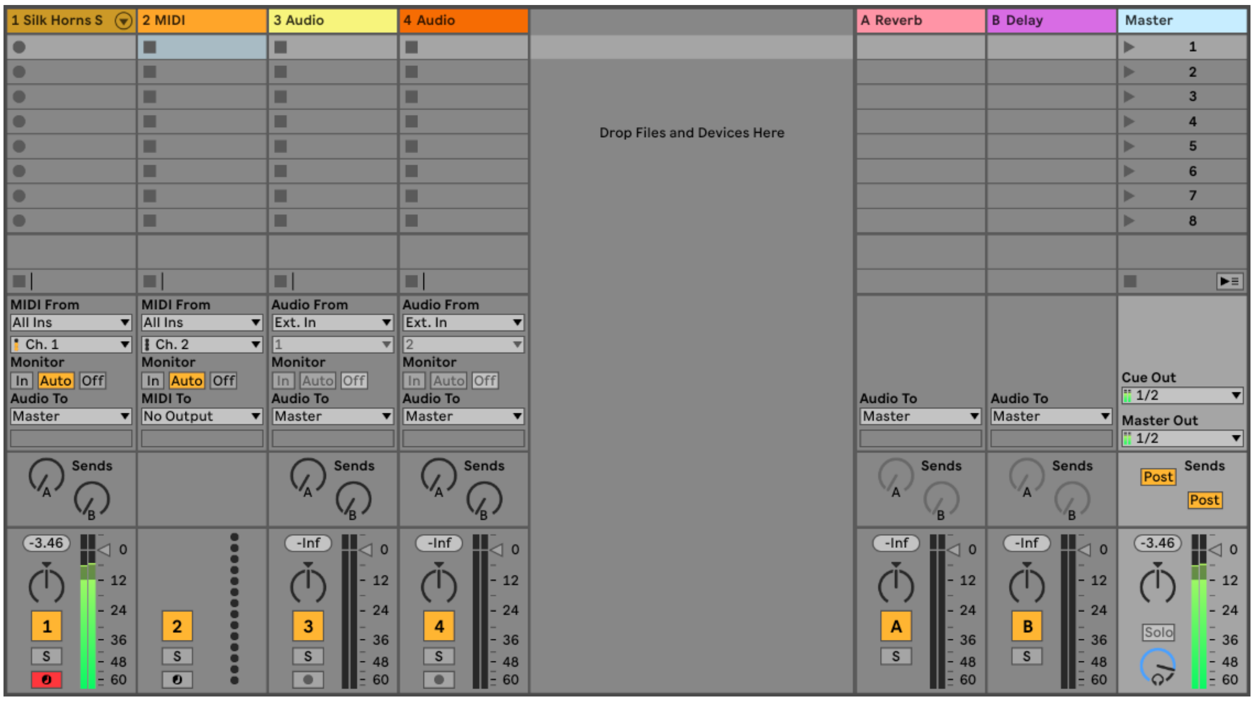


Figure 76: Ableton Live Receiving MIDI Data

The primary disadvantage of utilising the Hairless MIDI application was that any other form of communication across the serial port was prevented. While transmitting MIDI commands to Ableton Live, the Arduino could not be reprogrammed, and information could not be sent to the serial monitor. Hence, Hairless MIDI could not be connected during the debugging of the program if access to values displayed on the serial monitor was required.

9.3 Appendix C – LCD Display

The 1602A is a 16 pin, 16 x 2 character display LCD module, with backlight capabilities. The number of pins connected depended on the mode of operation; 8-bit mode required all 16 pins while 4-bit mode required only 12 (All About Microcontrollers, 2017). Characters displayed on the LCD module are comprised of 8-bit data; in 8-bit mode, one byte is sent in a single pulse, while in 4-bit mode, each byte is sent in two pulses, introducing latency (All About Microcontrollers, 2017). As a high refresh rate and high-speed data transmission were not required, the simplification in wiring associated with the omission of four pins was desirable and therefore implemented.

The pinout of the 1602A is described in **Table 10**, where pins 7 - 10 (DB0 - DB3) were omitted due to the 4-bit mode configuration.

***Table 10**: LCD Module Pinout (SHENZHEN EONE ELECTRONICS CO. LTD, 2010)*

Pin No	Function	Symbol
1	Ground LCD Module (0V)	V_{ss}
2	Supply voltage to LCD Module 5V (4.7V – 5.3V)	V_{dd}
3	Contrast adjustment; through a variable resistor	V_0
4	Selects command register when low, and data register when high	RS
5	Low to write to the register; High to read from the register	R/W
6	Sends data to data pins when a high to low pulse is given	E
7	Data Pin 1	DB0
8	Data Pin 2	DB1
9	Data Pin 3	DB2
10	Data Pin 4	DB3
11	Data Pin 5	DB4
12	Data Pin 6	DB5
13	Data Pin 7	DB6
14	Data Pin 8	DB7
15	Anode - Supply voltage to Backlight (5V)	BLA+
16	Cathode – Backlight Ground (0V)	BLK-

The status of the `Register Select` pin determines which type of information was sent; when high, commands to the LCD module could be sent such as screen-clearing and cursor position, when low, characters for display could be sent (Nedelkovski, 2015).

The `Read/Write` pin selects the LCD mode, `read` or `write` are specified by a status of either 1 or 0. For the purpose of displaying information, only the `write` mode is required and is therefore permanently set low through its connection to ground (Nedelkovski, 2015).

The `Enable` pin is timing-dependent and coordinates the writing to each of the data pins following the detection of a falling edge (Nedelkovski, 2015).

Neither the `Anode` nor `Cathode` are required for the LCD Module to function, however, the two pins do control the backlight (Nedelkovski, 2015).

The LCD module was connected to the Arduino in accordance to a successfully completed project (Nedelkovski, 2015). The description was from that Arduino Mega to the Arduino Uno, and the connections are depicted in **Figure 77**.

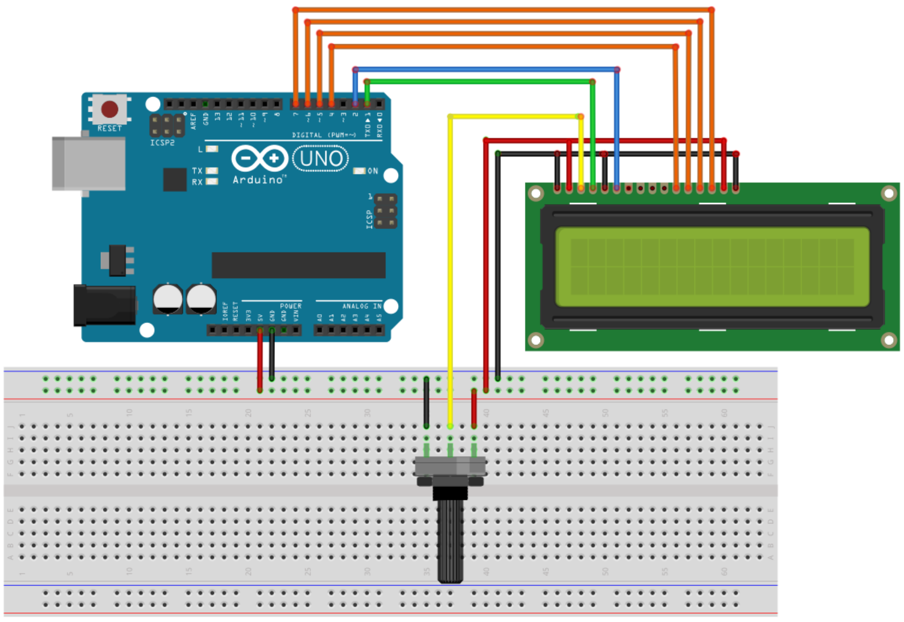


***Figure 77**: LCD to Arduino Connections*

The connections from the LCD module to the Arduino are summarised in **Table 11**. All digital pins were directly connected to the Arduino while the power and ground pins were connected through the power rails of the breadboard.

***Table 11**: LCD to Arduino Connections*

Pin No	Name	Arduino Connection
1	Ground	GND
2	Supply voltage	5V
3	Display Contrast Adjustment	POT
4	Register Select	Digital Pin 1
5	Read/Write	GND
6	Enable	Digital Pin 2
7	Data Pin 1	-
8	Data Pin 2	-
9	Data Pin 3	-
10	Data Pin 4	-
11	Data Pin 5	Digital Pin 4
12	Data Pin 6	Digital Pin 5
13	Data Pin 7	Digital Pin 6
14	Data Pin 8	Digital Pin 7
15	Backlight V_{CC} (5V)	5V
16	Backlight Ground (0V)	GND

To display the desired information, the Liquid Crystal Library was first initialised within the Arduino sketch, with the command `#include <LiquidCrystal.h>`.

A Liquid Crystal object in 4-pin mode was created by specifying the LCD pins connected to the Arduino in the following format: `LiquidCrystal (RS, Enable, D4, D5, D6, D7)`. With respect to each connection specified in **Table 11**, the Liquid Crystal object was created with the following command: `LiquidCrystal lcd(1,2,4,5,6,7)`.

The commands described in **Table 12** were implemented into the sketch; `lcd.begin(16,2)` was called in the `setup()`, while all other functions listed were accessed in the main `loop()`.

Upon initial testing, the LCD module was confirmed to display the desired text and variables, following the `lcd.print(x)` command. The breath mode, index number and output note were displayed, with the position specified by the `lcd.setCursor(w,h)` command.

***Table 12**: LCD Arduino Commands*

Command	Description
`lcd.begin(w,h)`	Initialises the LCD screen interface for dimensions width, `w`, and height, `h`
`lcd.print(x)`	Prints the variable `x`, to the LCD
`lcd.print("x")`	Prints the string `x`, to the LCD
`lcd.setCursor(w,h)`	Defines the location of the next character at position width, `w`, and height, `h`
`lcd.clear()`	Clears the LCD screen
`lcd.blink()`	Displays the position of the cursor by blinking
`lcd.noBlink()`	Removes the blinking cursor
`lcd.cursor()`	Displays the position of the cursor with an underscore
`lcd.noCursor()`	Removes the cursor from view

The LCD output shown in **Figure 78** was generated in a separate sketch to the main project; prior to integration, it was desired to confirm the LCD module was functional.

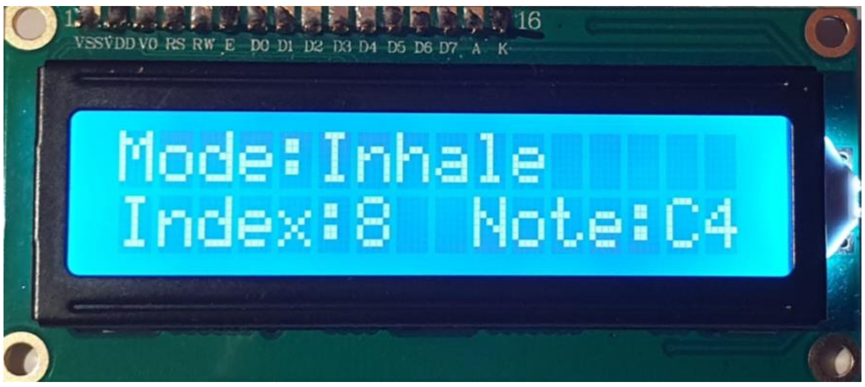


***Figure 78**: LCD Output*

With the exception of the punctuation colon, all characters displayed in **Figure 78** comprised exclusively of numbers and letters. To incorporate a musical association with the name of the note, musical symbols such as a clef or musical note, methods to display non-standard characters were investigated.

Typically, only standard ASCII characters can be directly displayed on an LCD, however, custom characters can be created with the `createChar()` function (de Bakker, 2019).

Each element on the LCD screen contains a 5 x 8 dot display, therefore the equivalent of a 40-pixel symbol or character can be constructed (de Bakker, 2019). Acknowledging the small number of pixels available, a simplistic symbol was selected, a pair of quavers, representative of the duration of one-eighth of a whole note. For reference, a traditional symbol for a pair of quavers is shown in **Figure 79**, the 40-pixel depiction of a pair of quavers is shown in **Figure 80**.

The custom character was converted into an array of eight bytes, each byte representing a dot display row. The five least significant bits of each byte directly correlate to the binary state of the custom character; as can be observed from the first row of **Figure 80**, the first four dots are white, while the final dot is black, hence the byte is set to B00000001, or B00001 (de Bakker, 2019).

All custom characters must be globally declared in the Arduino sketch, above the `setup()`; the pair of quavers is defined as `quaverPair` (**Figure 81**).

Figure 79 has been removed due to copyright restrictions

***Figure 79**: Pair of Quavers (Flaticon, 2014)*

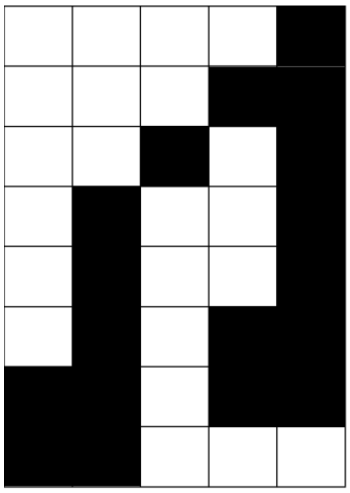


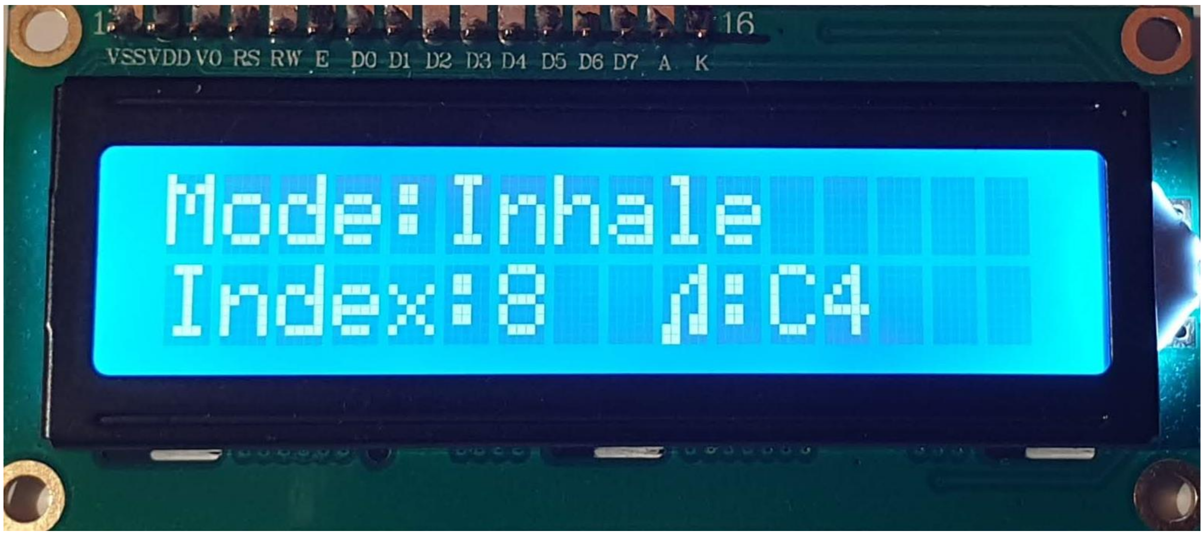
***Figure 80**: Quaver Pair Custom Character*

```
byte quaverPair[] = {
  B00001,
  B00011,
  B00101,
  B01001,
  B01001,
  B01011,
  B11011,
  B11000
};
```

***Figure 81**: Quaver Pair Character Definition*

Within the `setup()`, the Arduino sketch, the custom character is initialised as follows: `lcd.createChar(1, quaverPair)`. Multiple custom characters can be defined in a single sketch; hence the character is assigned an index number; 1 was assigned above, and hence was referenced in the `lcd.write(byte(1))` command.

The custom character was confirmed to output as expected, (**Figure 82**), programmed to replace the string `Note` shown in **Figure 78**.

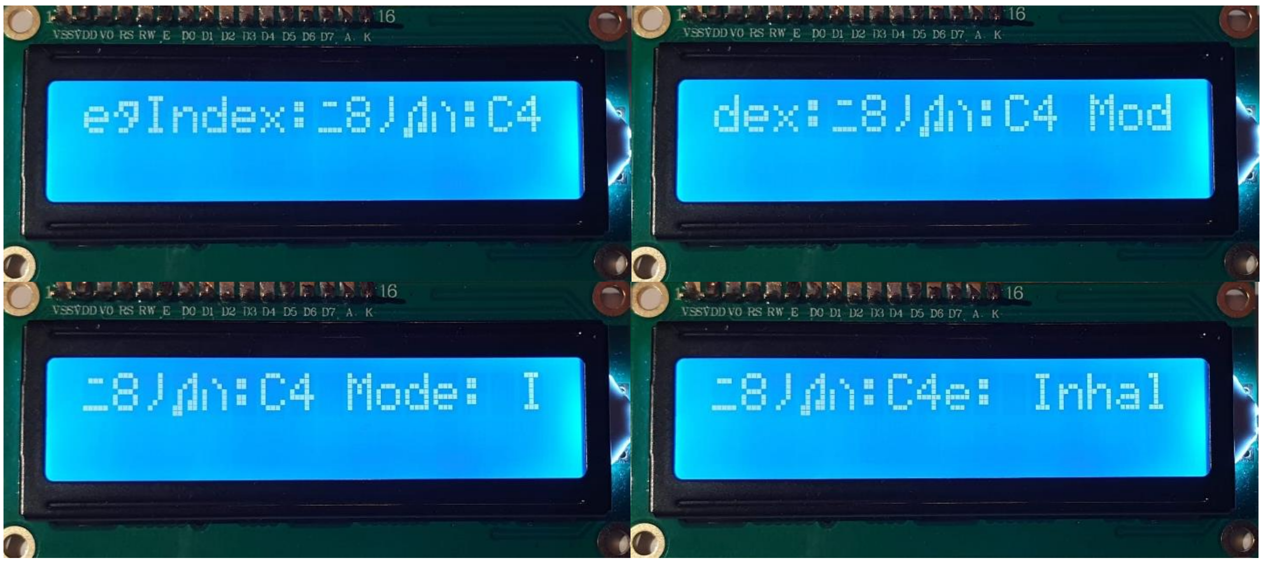


```
Mode:Inhale
Index:8   ♫:C4
```

Figure 82: LCD Output - Customised Quaver Pair Character

Following the success of the LCD configuration and display, the code was transferred into the main project. **Figure 83** displays four instances of the non-static output displayed from the code utilised to produce **Figure 82**.

From **Figure 83**, it was identified that both lines were incorrectly displayed on the first line, hence the cursor position could no longer be controlled, additionally, non-specified characters were also observed.



```
e Index: 8 ♫ :C4
dex: 8 ♫ :C4 Mod
 8 ♫ :C4 Mode: I
 8 ♫ :C4e: Inhal
```

Figure 83: Incorrect LCD Output

It was discovered that by removing the `Serial.begin()` command, the LCD output returned to the previously observed and expected display.

Through an exploration of the serial specifications within the Arduino Uno module, it was learned that during serial communication, Pin 0 and Pin 1 of the Arduino are used as the receiver and transmitter, respectively (Arduino, 2019c). As the RS pin of the LCD was connected to Pin 1 of the Arduino, the unexpected output was attributed to the duplication use of Pin 1.

To maintain intended functionality across the Arduino and LCD module, the RS pin and E pin were shifted from Pin 1 and Pin 2 to Pin 2 and Pin 3, respectively. The connection changes are reflected in **Figure 84**, while the Liquid Crystal object declaration was therefore amended as follows: `LiquidCrystal lcd(2,3,4,5,6,7).`

Following the connection and declaration modifications, the LCD display and the serial monitor were confirmed to operate as expected.

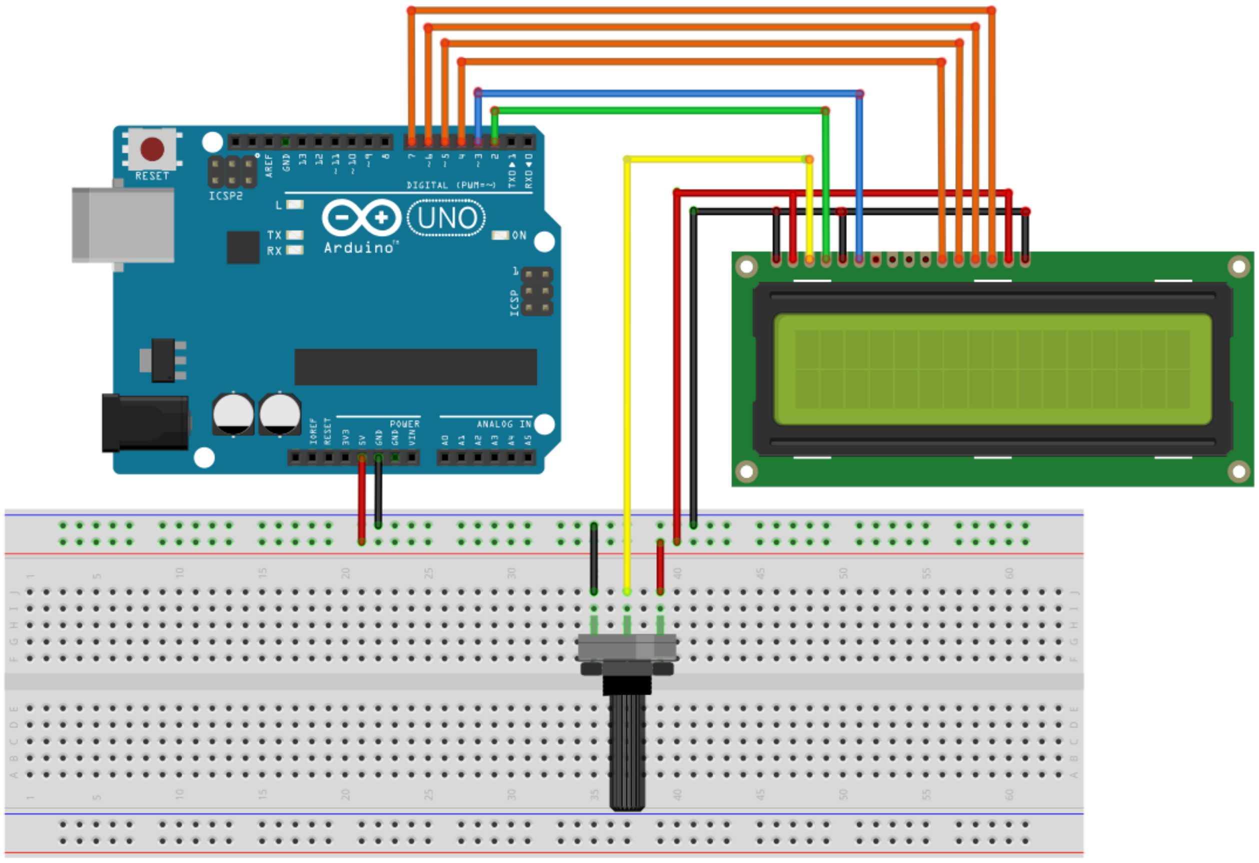


***Figure 84**: Correct LCD to Arduino Connections*

9.4 Appendix D – Teensy

9.4.1 Teensy 3.6

Figure 85 has been removed due to copyright restrictions

***Figure 85**: Teensy 3.6 Pinout (PJRC, 2018)*

Figure 86 has been removed due to copyright restrictions

***Figure 86**: Teensy 3.6 Schematic (PJRC, 2019d)*

9.4.2 Teensy Audio Adapter Board

Figure 87 has been removed due to copyright restrictions

***Figure 87**: Teensy Audio Adapter Board Schematic (PJRC, 2019a)*

Figure 88 has been removed due to copyright restrictions

***Figure 88**: Audio Adapter Board Pin Locations (PJRC, 2019a)*

***Table 13**: Audio Adapter Board Pin Assignments*

Signal	Rev D Teensy 4.0	Rev C Teensy 3.x	Function
MCLK	23	11	Audio Master Clock, 11.29 MHz
BCLK	21	9	Audio Bit Clock, 1.41 or 2.82 MHz
LRCLK	20	23	Audio Left/Right Clock, 44.1 kHz
DIN	7	22	Audio Data from Teensy to Audio Shield
DOUT	8	13	Audio Data from Audio Shield to Teensy
SCL	19	19	Control Clock (I2C)
SDA	18	18	Control Data (I2C)
SCK	13	14	Data Storage (SPI) Clock
MISO	12	12	Data Storage (SPI) from SD/MEM to Teensy
MOSI	11	7	Data Storage (SPI) from Teensy to SD/MEM
SDCS	10	10	Chip Select (SPI) for SD Card
MEMCS	6	6	Chip Select (SPI) for Memory Chip
Vol	15 / A1	15 / A1	Volume Thumbwheel (analog signal)

9.4.3 Module Specification Comparison

Table 14: *Teensy Module Specification Comparison (Core Electronics, 2018)*

Specification	Teensy LC	Teensy 3.2	Teensy 3.5	Teensy 3.6
Clock Speed	48MHz	96MHz	120MHz	180MHz
RAM	8K	64K	192K	256K
Flash	62K	256K	512K	1M
I/O Pins	27 (all on headers)	34 (24 on headers)	62 (42 on headers)	62 (42 on headers)
Pin Voltage	3.3V	3.3V (5V tolerant)	3.3V (5V tolerant)	3.3V
EEPROM	256 Bytes	2K	4K	4K
Connectivity	USB, Serial, SPI, I2C, I2S	USB, Serial, SPI, I2C, CAN, I2S	USB, CAN, I2C, SPI, Ethernet, SD, I2S	USB (High Speed), CAN, I2C, SPI, Ethernet, SD, I2S
Timers	7	12	14	14
PWM Outputs	10	12	20	20
Analog Inputs	13	21	25	25
Price (USD)	$11.65	$19.80	$24.25	$29.25

9.5 Appendix E – Software

9.5.1 Teensy Code

```
#define HWSERIAL Serial5

#include <Bounce.h>
#include <Audio.h>
#include <Wire.h>
#include <SPI.h>
#include <SD.h>
#include <SerialFlash.h>
#include "Trumpet_samples.h"

// GUItool: begin automatically generated code
AudioSynthWavetable      wavetable1;
AudioSynthWavetable      wavetable3;
AudioSynthWavetable      wavetable2;
AudioSynthWavetable      wavetable4;
AudioMixer4              mixer1;
AudioOutputI2S           i2s1;
AudioConnection          patchCord1(wavetable4, 0, mixer1, 3);
AudioConnection          patchCord2(wavetable1, 0, mixer1, 0);
AudioConnection          patchCord3(wavetable3, 0, mixer1, 2);
AudioConnection          patchCord4(wavetable2, 0, mixer1, 1);
AudioConnection          patchCord5(mixer1, 0, i2s1, 0);
AudioConnection          patchCord6(mixer1, 0, i2s1, 1);
AudioControlSGTL5000     sgtl5000_1;
// GUItool: end automatically generated code

// Bounce objects to read pushbuttons
Bounce button0 = Bounce(30, 15);  // 15 ms debounce time
Bounce button1 = Bounce(31, 15);  // 15 ms debounce time
Bounce button2 = Bounce(32, 15);  // 15 ms debounce time

// MIDI notes corresponding from E3 - Bb5
int noteMIDI[31] = {
  52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70,
71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82
};

// Notes names from from E3 - Bb5
String noteName[31] = {
  "E3", "F3", "Gb3", "G3", "Ab3", "A3", "Bb3", "B3", "C4", "Db4", "D4", "Eb4",
"E4", "F4", "Gb4",
  "G4", "Ab4", "A4", "Bb4", "B4", "C5", "Db5", "D5", "Eb5", "E5", "F5", "Gb5",
"G5", "Ab5", "A5", "Bb5"
};

int breathDetect = 0;
int duration = 2000;
int ICLevel;
int ICNote;
int ICVolume;
int maxEx;
int maxIn;
int missedCountBT = 0;
int note;
int noteCharLength = 2;
int noteDes;
int noteIndex = 0;
int notePrev;
int PB1Status = 0;
int PB2Status = 0;
int PB3Status = 0;
int scaleRunning = 0;

String breathStatus = "Inhale";
String chordType;
String ICVolumeS;
String PB1StatusS;
String PB2StatusS;
String PB3StatusS;
String noteDesS;
String readdata;
String rx_byte;
String rx_note;
String scaleType;
String screen = "Solo";
String screenPrev = "Solo";
String vibrate = "Y";
String vibrateNum;

void setup() {
  // Initialise Hardware Serial ports with baud rate 115200
  Serial.begin(115200);
  HWSERIAL.begin(115200);
  // Set active low pins as input with pullup resistor
  pinMode(29, INPUT_PULLUP);
  pinMode(30, INPUT_PULLUP);
  pinMode(31, INPUT_PULLUP);
  pinMode(32, INPUT_PULLUP);
  // Configure Audio elements
  AudioMemory(20);
  sgtl5000_1.enable();
  sgtl5000_1.volume(0.5);
  mixer1.gain(0, 0.9);
  // Set instrument to a trumpet - based on Trumpet_samples files
  wavetable1.setInstrument(trumpet);
  wavetable2.setInstrument(trumpet);
  wavetable3.setInstrument(trumpet);
  wavetable4.setInstrument(trumpet);
}


void loop() {
  // Read input from Serial Monitor
  if (Serial.available() > 0) {
    rx_byte = Serial.readStringUntil('\n');
  }

  // Enter if note is not within defined range
  if (screen == "Invalid Input") {
    Serial.println(screen);
  }
  else {
    // Obtain length of note name
    noteCharLength = noteName[noteIndex].length();
    // Enter if Bluetooth is not connected
```

```
if (screen == "Solo") {
  if (!digitalRead(29)) breathStatus =  "Inhale";
  else breathStatus =  "Exhale";
  // Display variable labels
  Serial.print("Screen");
  Serial.print(" | ");
  Serial.print("Note ");
  Serial.print("| ");
  Serial.print("Index");
  Serial.print("  | ");
  Serial.print("Breath");
  Serial.print(" | ");
  Serial.print("Vibrate");
  Serial.print(" | ");
  Serial.print("PB1");
  Serial.print(" | ");
  Serial.print("PB2");
  Serial.print(" | ");
  Serial.println("PB3");

  // Display variables of interest
  Serial.print(" ");
  Serial.print(screen);
  Serial.print("  |   ");
  Serial.print(noteName[noteIndex]);
  // Vary spacing depending on number of characters in note name
  if (noteCharLength == 2) Serial.print("  |    ");
  else Serial.print(" |    ");
  Serial.print(note);
  Serial.print("   | ");
  Serial.print(breathStatus);
  Serial.print(" |     ");
  Serial.print(vibrate);
  Serial.print("     |  ");
  Serial.print(!digitalRead(30));
  Serial.print("  |   ");
  Serial.print(!digitalRead(31));
  Serial.print("  |   ");
  Serial.println(!digitalRead(32));

  Serial.println(" ");
  delay(500);
}

// Enter if Bluetooth communication is established
if (HWSERIAL.available() > 0) {
  // Read screen type with delimiter '|'
  screen = HWSERIAL.readStringUntil('|');

  // Enter if volume-based incentive spirometry screen is open
  if (screen == "Volume") {

    scaleType = HWSERIAL.readStringUntil('|');
    ICVolumeS = HWSERIAL.readStringUntil('|');
    ICVolume = ICVolumeS.toInt();
    vibrateNum = HWSERIAL.readStringUntil('|');

    HWSERIAL.print(breathDetect);
    HWSERIAL.print("|");
    HWSERIAL.print(noteIndex);
    Serial.print("Screen");
    Serial.print(" | ");
    Serial.print("Note ");
    Serial.print("| ");
    Serial.print("Index");
    Serial.print("  | ");
    Serial.print("Breath");
    Serial.print(" | ");
    Serial.print("Vibrate");
    Serial.print(" | ");
    Serial.print("Scale");
    Serial.print("  | ");
    Serial.println("Volume");

    Serial.print(screen);
    Serial.print(" |  ");
    Serial.print(noteName[noteIndex]);
    if (noteCharLength == 2) Serial.print("  |   ");
    else Serial.print(" |   ");
    Serial.print(note);
    Serial.print("   | ");
    Serial.print(breathStatus);
    Serial.print(" |    ");
    if (vibrateNum == "1") vibrate = "Y";
    else vibrate = "N";
    Serial.print(vibrate);
    Serial.print("    | ");
    Serial.print(scaleType);
    if (scaleType == "Major") Serial.print("  |  ");
    else Serial.print(" |  ");
    Serial.println(ICVolumeS);
    Serial.println("");

    if (ICVolume < 2000) {
      duration = (ICVolume * 3) / 8;
      ICLevel = 1;
    }
    else {
      duration = (ICVolume * 3) / 15;
      ICLevel = 2;
    }

    ICNote = noteMIDI[(ICVolume / 4000) * 30];

    //if (!scaleRunning) scale(ICNote, scaleType, duration, ICLevel);

  }

  // Enter if music creation screen is open
  if (screen == "Music") {
    breathStatus = HWSERIAL.readStringUntil('|');
    chordType = HWSERIAL.readStringUntil('|');
    // Read in string containing numberical data
    PB1StatusS = HWSERIAL.readStringUntil('|');
    // Convert string to integer for use in calculations
    PB1Status = PB1StatusS.toInt();
    PB2StatusS = HWSERIAL.readStringUntil('|');
    PB2Status = PB2StatusS.toInt();
    PB3StatusS = HWSERIAL.readStringUntil('|');
    PB3Status = PB3StatusS.toInt();
    vibrateNum = HWSERIAL.readStringUntil('|');

    HWSERIAL.print(noteName[noteIndex]);
    HWSERIAL.print("|");
    HWSERIAL.print(note);

    Serial.print("Screen");
    Serial.print(" | ");
    Serial.print("Note ");
    Serial.print("| ");
    Serial.print("Index");
    Serial.print("  | ");
    Serial.print("Breath");
    Serial.print(" | ");
    Serial.print("Vibrate");
    Serial.print(" | ");
    Serial.print("PB1");
    Serial.print(" | ");
    Serial.print("PB2");
    Serial.print(" | ");
    Serial.print("PB3");
    Serial.print(" | ");
    Serial.println("Chord");

    Serial.print(" ");
    Serial.print(screen);
    Serial.print(" |  ");
    Serial.print(noteName[noteIndex]);
    if (noteCharLength == 2) Serial.print("  |   ");
    else Serial.print(" |   ");
    Serial.print(note);
    Serial.print("   | ");
    Serial.print(breathStatus);
    Serial.print(" |    ");
    if (vibrateNum == "1") vibrate = "Y";
    else vibrate = "N";
    Serial.print(vibrate);
    Serial.print("    |  ");
    Serial.print(PB1Status);
    Serial.print("  |  ");
    Serial.print(PB2Status);
    Serial.print("  |  ");
    Serial.print(PB3Status);
    Serial.print("  |  ");
    Serial.println(chordType);

    Serial.println(" ");
  }


  // Enter if flow-based incentive spirometry screen is open
  if (screen == "Flow") {
    breathStatus = HWSERIAL.readStringUntil('|');
    noteDesS = HWSERIAL.readStringUntil('|');
    noteDes = noteDesS.toInt();
    vibrateNum = HWSERIAL.readStringUntil('|');
    HWSERIAL.print(noteName[noteIndex]);
```

```
        HWSERIAL.print("|");
        HWSERIAL.print(noteIndex);
        HWSERIAL.print("|");
        HWSERIAL.print(noteName[noteDes]);

        Serial.print("Screen");
        Serial.print(" | ");
        Serial.print("Note ");
        Serial.print("| ");
        Serial.print("Index");
        Serial.print("  | ");
        Serial.print("Breath");
        Serial.print(" | ");
        Serial.print("Vibrate");
        Serial.print(" | ");
        Serial.println ("Desired Note");

        Serial.print(" ");
        Serial.print(screen);
        Serial.print("  |  ");
        Serial.print(noteName[noteIndex]);
        if (noteCharLength == 2) Serial.print("  |   ");
        else Serial.print(" |   ");
        Serial.print(note);
        Serial.print("   | ");
        Serial.print(breathStatus);
        Serial.print(" |    ");
        if (vibrateNum == "1") vibrate = "Y";
        else vibrate = "N";
        Serial.print(vibrate);
        Serial.print("    |     ");
        Serial.println(noteName[noteDes]);

        Serial.println("");

      }


      // Clear Bluetooth after successful connection
      missedCountBT = 0;
    }

    // Refresh rate of code is higher than Bluetooth communication
    else {
      // Increment each time Bluetooth connection is unsuccessful
      missedCountBT++;
      // Device is no longer connected to Bluetooth module
      if (missedCountBT >= 15) {
        screen = "Solo";
        missedCountBT = 0;
      }
    }
  }

  // Update all the button objects
  button0.update();
  button1.update();
  button2.update();

  // Set previous note to current note
```

```
notePrev = note;
// Obtain new note value
note = noteNum(breathStatus);
// Modify note based on pushbutton status
note += valve(digitalRead(30), digitalRead(31), digitalRead(32));
//Serial.println(note);
// Store index of note
//noteIndex = note;
// Enter if note is not within acceptable range
if (note < 0 || note > 30) {
  screen = "Invalid Input";
  wavetable1.stop();
}
// Assign note to MIDI equivalent
else {
  if (screen != "Invalid Input") screenPrev = screen;
  screen = screenPrev;
  noteIndex = note;
  note = noteMIDI[note];
}

// Send command to play a new note note if differnt to the previous note

if (notePrev != note && screen != "Invalid Input") {
  if (screen != "Volume") {
    wavetable1.playNote(note);
  }
  else wavetable1.stop();
  notePrev = note;
}





// Assign chord and scale types based on Serial Monitor input
if (rx_byte == 'M') {
  chordType = "Major";
}
if (rx_byte == 'm') {
  chordType = "Minor";
}
if (rx_byte == '7') {
  chordType = "7th";
}
if (rx_byte == '2') {
  chordType = "Sus2";
}
if (rx_byte == '4') {
  chordType = "Sus4";
}
if (rx_byte == "m7") {
  chordType = "Minor7";
}
if (rx_byte == "M7") {
  chordType = "Major7";
}
if (rx_byte == "SM") {
  scaleType = "Major";
}
```

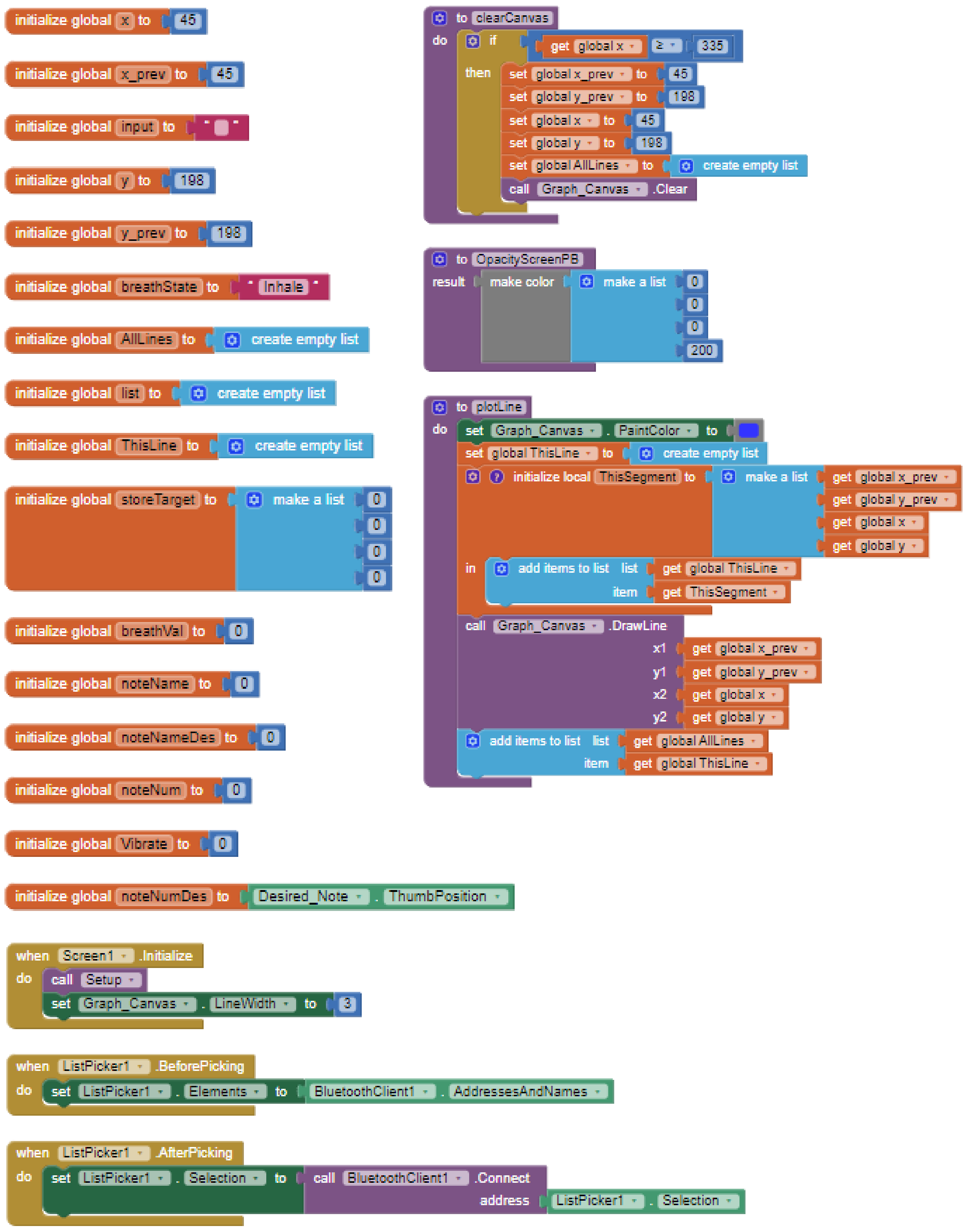
```
  if (rx_byte == "Sm") {
    scaleType = "NMinor";
  }
  if (rx_byte == "Shm") {
    scaleType = "HMinor";
  }
  if (rx_byte == "Smm") {
    scaleType = "MMinor";
  }

  // Change note based on Serial Monitor Input
  if (rx_byte == "note") {
    while (Serial.available() == 0) {}
    rx_note = Serial.readStringUntil('\n');
    note = rx_note.toInt();
  }

  // Stop playback based on Serial Monitor Input
  if (rx_byte == "stop") {
    chordType = "";
    scaleType = "";
    wavetable1.stop();
  }

  if (screen != "Invalid Input") {
    // Call function to play chord
    // chord (note, chordType, duration);
    // Call function to play scale

  }

  delay(100);
}

// Convert analog pressure sensor input into note value between 0 - 30
int noteNum(String breathStatus) {

  int scaling_factor;
  int sensorVal;
  int note;

  // Modify calculation based on breath type expected
  if (breathStatus == "Exhale") {
    maxEx = 1024;
    scaling_factor = (maxEx - 625) / 30;
    sensorVal = analogRead(A16);

    //Serial.println(sensorVal);

    note = (sensorVal - 615) / scaling_factor;
  }
  if (breathStatus == "Inhale") {
    maxIn = 450;
    scaling_factor = (maxIn) / 30;
    sensorVal = analogRead(A16);
    //Serial.println(sensorVal);
    note = (maxIn - sensorVal) / scaling_factor;
  }
  return note;
}
```

```
// Replicate function of valves with physical or virtual push buttons
int valve(int Pb1, int Pb2, int Pb3) {
  int value = 0;
  // First push button represents first valve - lowers pitch by two semitones
  if (!Pb1 || PB1Status) {
    value -= 2;
  }
  // Second push button represents second valve - lowers pitch by one semitone
  if (!Pb2 || PB2Status) {
    value -= 1;
  }
  // Third push button represents third valve - lowers pitch by three
semitones
  if (!Pb3 || PB3Status) {
    value -= 3;
  }
  return value;
}

// Calculate notes to play in scale
void scale(int rootNote, String scaleType, int duration, int ICLevel) {
  wavetable1.stop();
  // Calculate notes to play in Major scale
  if (scaleType == "Major") {
    int note[8];
    note[0] = rootNote;
    note[1] = rootNote + 2;
    note[2] = rootNote + 4;
    note[3] = rootNote + 5;
    note[4] = rootNote + 7;
    note[5] = rootNote + 9;
    note[6] = rootNote + 11;
    note[7] = rootNote + 12;
    if (!scaleRunning) playScale(note[0], note[1], note[2], note[3], note[4],
note[5], note[6], note[7], duration, ICLevel);
  }

  // Calculate notes to play in Natural Minor scale
  if (scaleType == "NMinor") {
    int note[8];
    note[0] = rootNote;
    note[1] = rootNote + 2;
    note[2] = rootNote + 3;
    note[3] = rootNote + 5;
    note[4] = rootNote + 7;
    note[5] = rootNote + 8;
    note[6] = rootNote + 10;
    note[7] = rootNote + 12;
    playScale(note[0], note[1], note[2], note[3], note[4], note[5], note[6],
note[7], duration, ICLevel);
  }

  // Calculate notes to play in Harmonic Minor scale
  if (scaleType == "HMinor") {
    int note[8];
    note[0] = rootNote;
    note[1] = rootNote + 2;
    note[2] = rootNote + 3;
    note[3] = rootNote + 5;
    note[4] = rootNote + 7;
    note[5] = rootNote + 8;
    note[6] = rootNote + 11;
    note[7] = rootNote + 12;
    playScale(note[0], note[1], note[2], note[3], note[4], note[5], note[6],
note[7], duration, ICLevel);
  }

  // Calculate notes to play in Melodic Minor scale
  if (scaleType == "MMinor") {
    int note[8];
    note[0] = rootNote;
    note[1] = rootNote + 2;
    note[2] = rootNote + 3;
    note[3] = rootNote + 5;
    note[4] = rootNote + 7;
    note[5] = rootNote + 9;
    note[6] = rootNote + 11;
    note[7] = rootNote + 12;
    playScale(note[0], note[1], note[2], note[3], note[4], note[5], note[6],
note[7], duration, ICLevel);
  }
}

// Calculate notes to play in chord
void chord (int rootNote, String chordType, int duration) {
  if (chordType == "Major") {
    int note[3];
    note[0] = rootNote;
    note[1] = rootNote + 4;
    note[2] = rootNote + 7;
    playChord3(note[0], note[1], note[2], duration);
  }
  if (chordType == "Minor") {
    int note[3];
    note[0] = rootNote;
    note[1] = rootNote + 3;
    note[2] = rootNote + 7;
    playChord3(note[0], note[1], note[2], duration);
  }
  if (chordType == "7th") {
    int note[4];
    note[0] = rootNote;
    note[1] = rootNote + 4;
    note[2] = rootNote + 7;
    note[3] = rootNote + 10;
    playChord4(note[0], note[1], note[2], note[3], duration);
  }
  if (chordType == "Major7") {
    int note[4];
    note[0] = rootNote;
    note[1] = rootNote + 4;
    note[2] = rootNote + 7;
    note[3] = rootNote + 11;
    playChord4(note[0], note[1], note[2], note[3], duration);
  }
  if (chordType == "Minor7") {
    int note[4];
    note[0] = rootNote;
    note[1] = rootNote + 3;
```

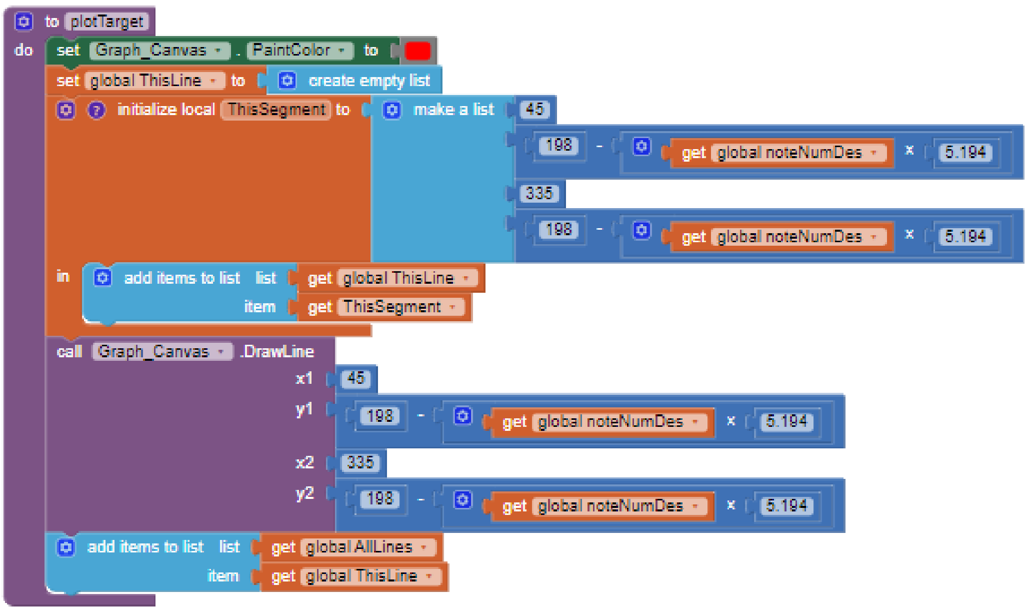
```
      note[2] = rootNote + 7;
      note[3] = rootNote + 10;
      playChord4(note[0], note[1], note[2], note[3], duration);
    }
    if (chordType == "Sus2") {
      int note[3];
      note[0] = rootNote;
      note[1] = rootNote + 2;
      note[2] = rootNote + 7;
      playChord3(note[0], note[1], note[2], duration);
    }
    if (chordType == "Sus4") {
      int note[3];
      note[0] = rootNote;
      note[1] = rootNote + 5;
      note[2] = rootNote + 7;
      playChord3(note[0], note[1], note[2], duration);
    }
  }

  // Play chord containing 3 notes
  void playChord3(int note1, int note2, int note3, int duration) {
    wavetable1.playNote(note1);
    wavetable2.playNote(note2);
    wavetable3.playNote(note3);
    delay(duration);
    wavetable1.stop();
    wavetable2.stop();
    wavetable3.stop();
  }

  // Play chord containing 4 notes
  void playChord4(int note1, int note2, int note3, int note4, int duration) {
    wavetable1.playNote(note1);
    wavetable2.playNote(note2);
    wavetable3.playNote(note3);
    wavetable4.playNote(note3);
    delay(duration);
    wavetable1.stop();
    wavetable2.stop();
    wavetable3.stop();
    wavetable4.stop();
  }

  // Play chord containing 4  notes
  void playScale(int note1, int note2, int note3, int note4, int note5, int
  note6, int note7, int note8, int duration, int ICLevel) {
    scaleRunning = 1;
    wavetable2.playNote(note1);
    delay(duration);
    wavetable2.stop();
    wavetable3.playNote(note2);

    delay(duration);
    wavetable3.stop();
    wavetable4.playNote(note3);

    delay(duration);
    wavetable4.stop();
    wavetable2.playNote(note4);
    delay(duration);
    wavetable2.stop();
    wavetable3.playNote(note5);

    delay(duration);
    wavetable3.stop();
    wavetable4.playNote(note6);

    delay(duration);
    wavetable4.stop();
    wavetable2.playNote(note7);
    delay(duration);
    wavetable2.stop();
    wavetable3.playNote(note8);

    delay(duration);
    wavetable3.stop();


    if (ICLevel == 1) {
      delay(duration);
      scaleRunning = 0;

      return;
    }
    else playScale2(note7, note6, note5, note4, note3, note2, note1, duration);
  }

  void playScale2(int note1, int note2, int note3, int note4, int note5, int
  note6, int note7, int duration) {

    wavetable2.playNote(note1);
    delay(duration);
    wavetable2.stop();
    wavetable3.playNote(note2);

    delay(duration);
    wavetable3.stop();
    wavetable4.playNote(note3);

    delay(duration);
    wavetable4.stop();
    wavetable2.playNote(note4);
    delay(duration);
    wavetable2.stop();
    wavetable3.playNote(note5);

    delay(duration);
    wavetable3.stop();
    wavetable4.playNote(note6);

    delay(duration);
    wavetable4.stop();
    wavetable2.playNote(note7);
    delay(duration);
    wavetable2.stop();

    delay(duration);
    scaleRunning = 0;
  }
```

9.5.2 MIT App Inventor Code Blocks

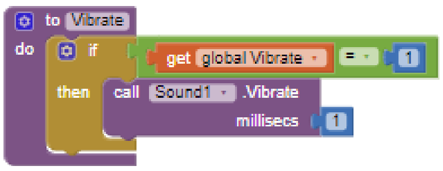
9.5.2.1 Flow-Orientated Mode



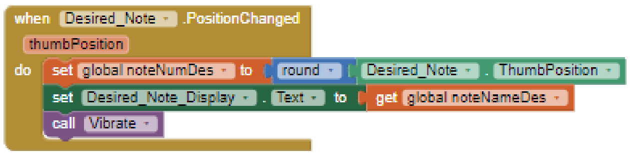
```
initialize global x to 45
initialize global x_prev to 45
initialize global input to " "
initialize global y to 198
initialize global y_prev to 198
initialize global breathState to " Inhale "
initialize global AllLines to create empty list
initialize global list to create empty list
initialize global ThisLine to create empty list
initialize global storeTarget to make a list 0
                                             0
                                             0
                                             0
initialize global breathVal to 0
initialize global noteName to 0
initialize global noteNameDes to 0
initialize global noteNum to 0
initialize global Vibrate to 0
initialize global noteNumDes to Desired_Note . ThumbPosition

when Screen1 .Initialize
do  call Setup
    set Graph_Canvas . LineWidth to 3

when ListPicker1 .BeforePicking
do  set ListPicker1 . Elements to BluetoothClient1 . AddressesAndNames

when ListPicker1 .AfterPicking
do  set ListPicker1 . Selection to call BluetoothClient1 .Connect
                                                 address ListPicker1 . Selection

to clearCanvas
do  if get global x ≥ 335
    then set global x_prev to 45
         set global y_prev to 198
         set global x to 45
         set global y to 198
         set global AllLines to create empty list
         call Graph_Canvas .Clear

to OpacityScreenPB
result make color make a list 0
                              0
                              0
                              200

to plotLine
do  set Graph_Canvas . PaintColor to 🟦
    set global ThisLine to create empty list
    initialize local ThisSegment to make a list get global x_prev
                                                get global y_prev
                                                get global x
                                                get global y
    in  add items to list list get global ThisLine
                          item get ThisSegment
    call Graph_Canvas .DrawLine
                             x1 get global x_prev
                             y1 get global y_prev
                             x2 get global x
                             y2 get global y
    add items to list list get global AllLines
                      item get global ThisLine
```



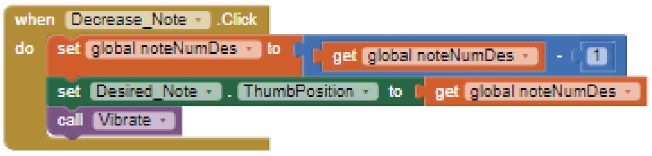
```
to plotTarget
do  set Graph_Canvas . PaintColor to 
    set global ThisLine to  create empty list
    initialize local ThisSegment to  make a list  45
                                                  198 - ( get global noteNumDes × 5.194 )
                                                  335
                                                  198 - ( get global noteNumDes × 5.194 )
    in  add items to list  list  get global ThisLine
                           item  get ThisSegment
    call Graph_Canvas .DrawLine
                         x1  45
                         y1  198 - ( get global noteNumDes × 5.194 )
                         x2  335
                         y2  198 - ( get global noteNumDes × 5.194 )
    add items to list  list  get global AllLines
                       item  get global ThisLine
```



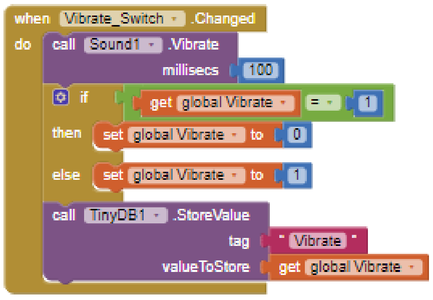
```
to Vibrate
do  if  get global Vibrate = 1
    then  call Sound1 .Vibrate
                      millisecs  1
```



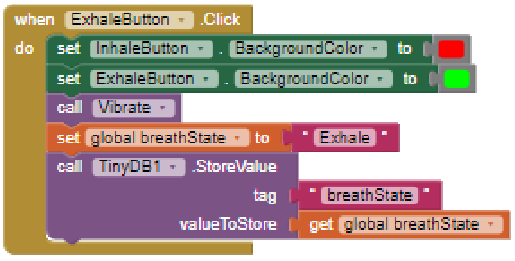
```
when Desired_Note .PositionChanged
thumbPosition
do  set global noteNumDes to  round  Desired_Note . ThumbPosition
    set Desired_Note_Display . Text to  get global noteNameDes
    call Vibrate
```



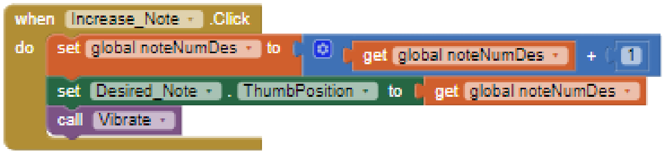
```
when Decrease_Note .Click
do  set global noteNumDes to  get global noteNumDes - 1
    set Desired_Note . ThumbPosition to  get global noteNumDes
    call Vibrate
```



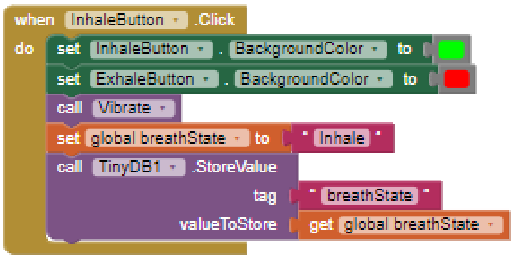
```
when Vibrate_Switch .Changed
do  call Sound1 .Vibrate
                millisecs  100
    if  get global Vibrate = 1
    then  set global Vibrate to  0
    else  set global Vibrate to  1
    call TinyDB1 .StoreValue
                         tag  " Vibrate "
                  valueToStore  get global Vibrate
```



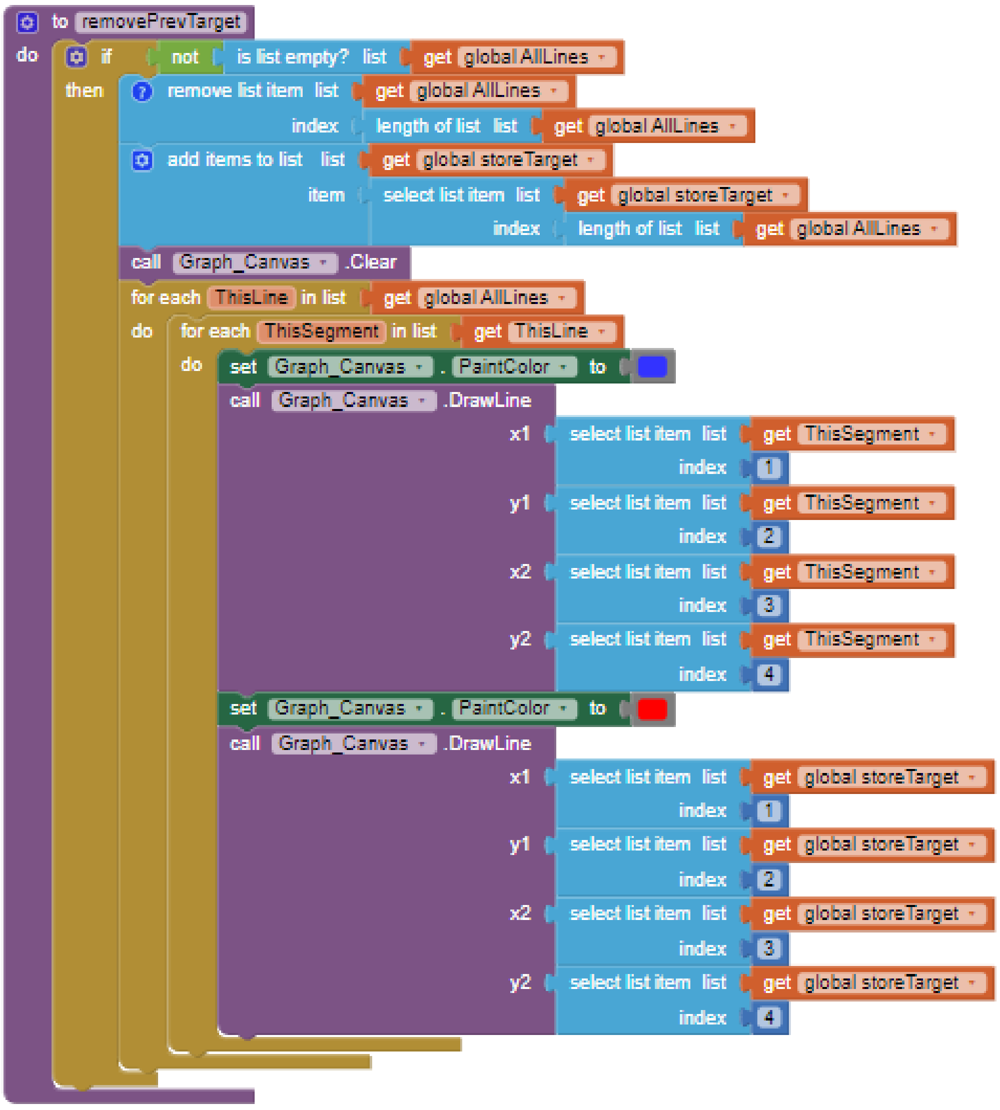
```
when ExhaleButton .Click
do  set InhaleButton . BackgroundColor to 
    set ExhaleButton . BackgroundColor to 
    call Vibrate
    set global breathState to  " Exhale "
    call TinyDB1 .StoreValue
                         tag  " breathState "
                  valueToStore  get global breathState
```



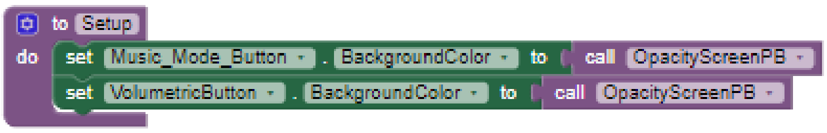
```
when Increase_Note .Click
do  set global noteNumDes to  get global noteNumDes + 1
    set Desired_Note . ThumbPosition to  get global noteNumDes
    call Vibrate
```



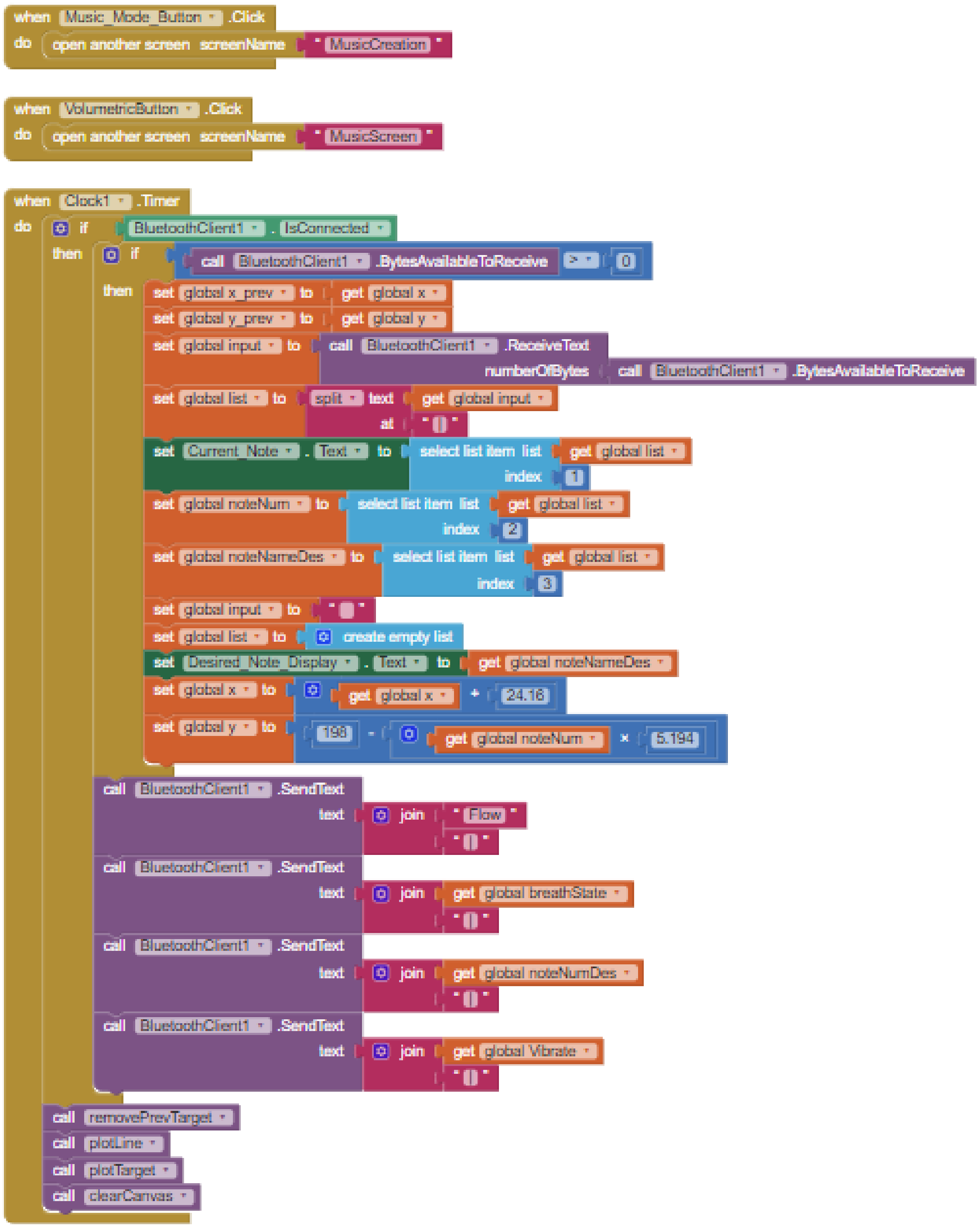
```
when InhaleButton .Click
do  set InhaleButton . BackgroundColor to 
    set ExhaleButton . BackgroundColor to 
    call Vibrate
    set global breathState to  " Inhale "
    call TinyDB1 .StoreValue
                         tag  " breathState "
                  valueToStore  get global breathState
```



```
to removePrevTarget
do  if  not  is list empty?  list  get global AllLines
    then  remove list item  list  get global AllLines
                            index  length of list  list  get global AllLines
          add items to list  list  get global storeTarget
                             item  select list item  list  get global storeTarget
                                                     index  length of list  list  get global AllLines
    call Graph_Canvas .Clear
    for each ThisLine in list  get global AllLines
    do  for each ThisSegment in list  get ThisLine
        do  set Graph_Canvas . PaintColor to
            call Graph_Canvas .DrawLine
                 x1  select list item  list  get ThisSegment
                                       index  1
                 y1  select list item  list  get ThisSegment
                                       index  2
                 x2  select list item  list  get ThisSegment
                                       index  3
                 y2  select list item  list  get ThisSegment
                                       index  4
            set Graph_Canvas . PaintColor to
            call Graph_Canvas .DrawLine
                 x1  select list item  list  get global storeTarget
                                       index  1
                 y1  select list item  list  get global storeTarget
                                       index  2
                 x2  select list item  list  get global storeTarget
                                       index  3
                 y2  select list item  list  get global storeTarget
                                       index  4
```

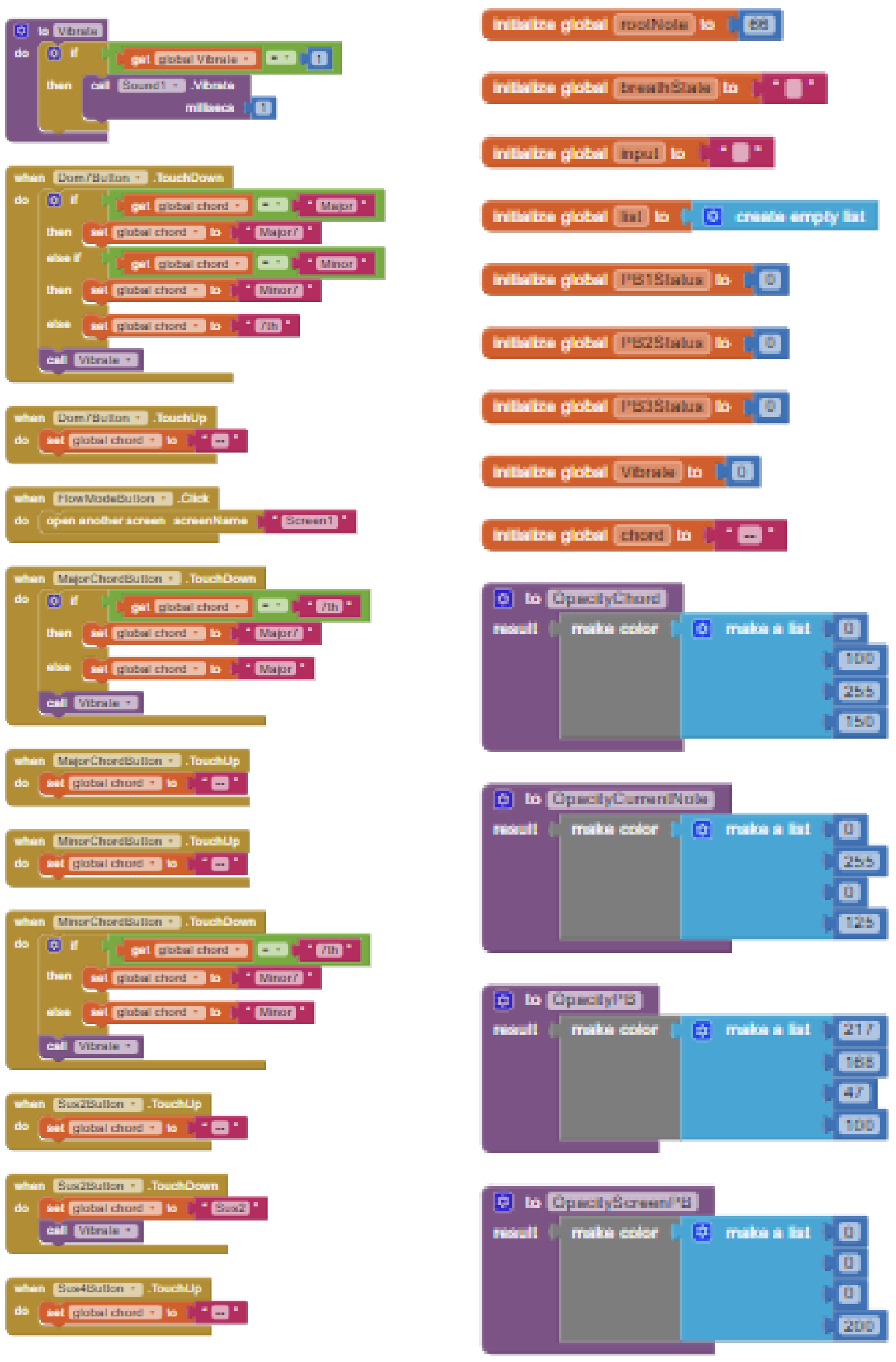


```
to Setup
do  set Music_Mode_Button . BackgroundColor to  call OpacityScreenPB
    set VolumetricButton . BackgroundColor to  call OpacityScreenPB
```

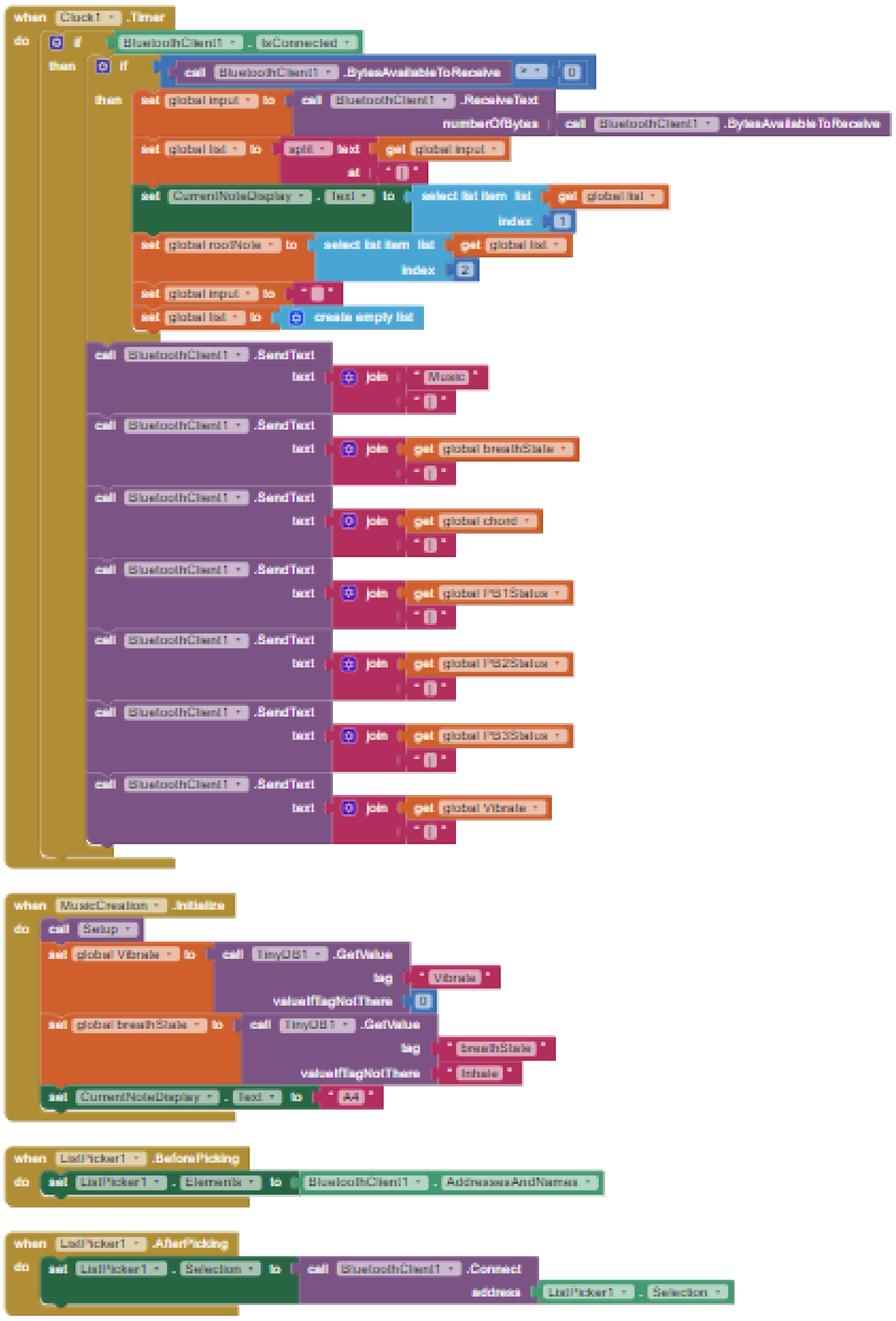


```
when Music_Mode_Button .Click
do   open another screen  screenName  " MusicCreation "

when VolumetricButton .Click
do   open another screen  screenName  " MusicScreen "

when Clock1 .Timer
do   if   BluetoothClient1 . IsConnected
     then if   call BluetoothClient1 .BytesAvailableToReceive  >  0
          then set global x_prev to  get global x
               set global y_prev to  get global y
               set global input to  call BluetoothClient1 .ReceiveText
                                         numberOfBytes  call BluetoothClient1 .BytesAvailableToReceive
               set global list to  split  text  get global input
                                          at  " | "
               set Current_Note . Text to  select list item  list  get global list
                                                             index  1
               set global noteNum to  select list item  list  get global list
                                                        index  2
               set global noteNameDes to  select list item  list  get global list
                                                            index  3
               set global input to  " | "
               set global list to  create empty list
               set Desired_Note_Display . Text to  get global noteNameDes
               set global x to  get global x  +  24.16
               set global y to  198  -  get global noteNum  ×  5.194
          call BluetoothClient1 .SendText
                                text  join  " Flow "
                                            " | "
          call BluetoothClient1 .SendText
                                text  join  get global breathState
                                            " | "
          call BluetoothClient1 .SendText
                                text  join  get global noteNumDes
                                            " | "
          call BluetoothClient1 .SendText
                                text  join  get global Vibrate
                                            " | "
     call removePrevTarget
     call plotLine
     call plotTarget
     call clearCanvas
```

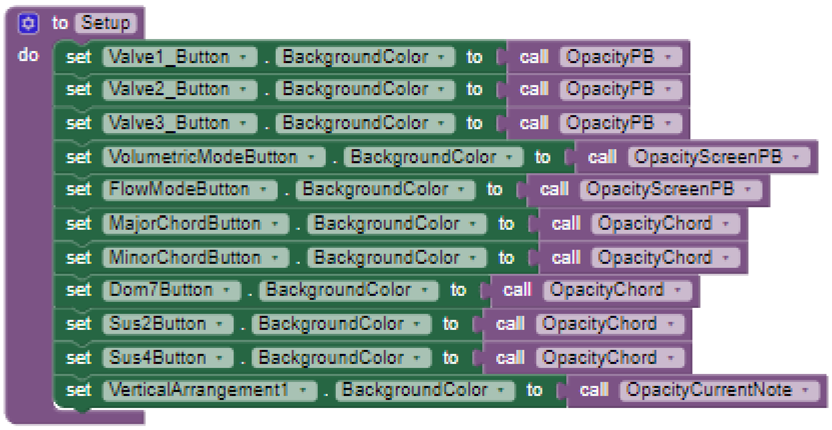
9.5.2.2 Music Creation Mode



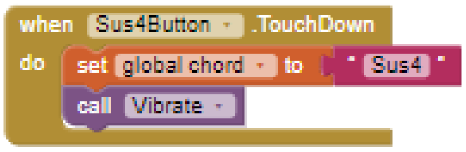
```
to Vibrate
do  if  get global Vibrate = 1
    then call Sound1.Vibrate
              millisecs 1

when Dom7Button.TouchDown
do  if  get global chord = "Major"
    then set global chord to "Major7"
    else if  get global chord = "Minor"
    then set global chord to "Minor7"
    else set global chord to "7th"
    call Vibrate

when Dom7Button.TouchUp
do  set global chord to "--"

when FlowModeButton.Click
do  open another screen  screenName "Screen1"

when MajorChordButton.TouchDown
do  if  get global chord = "7th"
    then set global chord to "Major7"
    else set global chord to "Major"
    call Vibrate

when MajorChordButton.TouchUp
do  set global chord to "--"

when MinorChordButton.TouchUp
do  set global chord to "--"

when MinorChordButton.TouchDown
do  if  get global chord = "7th"
    then set global chord to "Minor7"
    else set global chord to "Minor"
    call Vibrate

when Sus2Button.TouchUp
do  set global chord to "--"

when Sus2Button.TouchDown
do  set global chord to "Sus2"
    call Vibrate

when Sus4Button.TouchUp
do  set global chord to "--"

initialize global rootNote to 68
initialize global breathState to " "
initialize global input to " "
initialize global list to create empty list
initialize global PS1Status to 0
initialize global PS2Status to 0
initialize global PS3Status to 0
initialize global Vibrate to 0
initialize global chord to "--"

to OpacityChord
result  make color  make a list  0
                                 100
                                 255
                                 150

to OpacityCurrentNote
result  make color  make a list  0
                                 255
                                 0
                                 125

to OpacityPS
result  make color  make a list  217
                                 168
                                 47
                                 100

to OpacityScreenPS
result  make color  make a list  0
                                 0
                                 0
                                 200
```



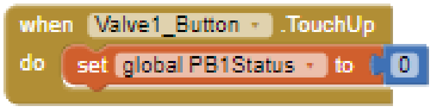
```
when Clock1 .Timer
do  if BluetoothClient1 . IsConnected
    then  if call BluetoothClient1 .BytesAvailableToReceive > 0
          then  set global input to call BluetoothClient1 .ReceiveText
                                         numberOfBytes  call BluetoothClient1 .BytesAvailableToReceive
                set global list to split text get global input
                                         at " "
                set CurrentNoteDisplay . Text to select list item list get global list
                                                                  index 1
                set global rootNote to select list item list get global list
                                                        index 2
                set global input to " "
                set global list to create empty list
          call BluetoothClient1 .SendText
                                text  join " Music "
                                           " "
          call BluetoothClient1 .SendText
                                text  join get global breathState
                                           " "
          call BluetoothClient1 .SendText
                                text  join get global chord
                                           " "
          call BluetoothClient1 .SendText
                                text  join get global PB1Status
                                           " "
          call BluetoothClient1 .SendText
                                text  join get global PB2Status
                                           " "
          call BluetoothClient1 .SendText
                                text  join get global PB3Status
                                           " "
          call BluetoothClient1 .SendText
                                text  join get global Vibrate
                                           " "

when MusicCreation .Initialize
do  call Setup
    set global Vibrate to call TinyDB1 .GetValue
                                  tag " Vibrate "
                                  valueIfTagNotThere 0
    set global breathState to call TinyDB1 .GetValue
                                      tag " breathState "
                                      valueIfTagNotThere " Inhale "
    set CurrentNoteDisplay . Text to " A4 "

when ListPicker1 .BeforePicking
do  set ListPicker1 . Elements to BluetoothClient1 . AddressesAndNames

when ListPicker1 .AfterPicking
do  set ListPicker1 . Selection to call BluetoothClient1 .Connect
                                                        address ListPicker1 . Selection
```



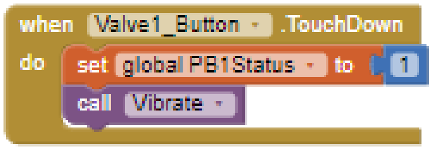
```
to Setup
do  set Valve1_Button . BackgroundColor to call OpacityPB
    set Valve2_Button . BackgroundColor to call OpacityPB
    set Valve3_Button . BackgroundColor to call OpacityPB
    set VolumetricModeButton . BackgroundColor to call OpacityScreenPB
    set FlowModeButton . BackgroundColor to call OpacityScreenPB
    set MajorChordButton . BackgroundColor to call OpacityChord
    set MinorChordButton . BackgroundColor to call OpacityChord
    set Dom7Button . BackgroundColor to call OpacityChord
    set Sus2Button . BackgroundColor to call OpacityChord
    set Sus4Button . BackgroundColor to call OpacityChord
    set VerticalArrangement1 . BackgroundColor to call OpacityCurrentNote
```



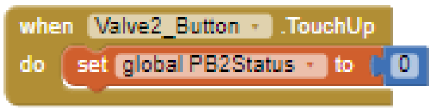
```
when Sus4Button .TouchDown
do  set global chord to " Sus4 "
    call Vibrate
```



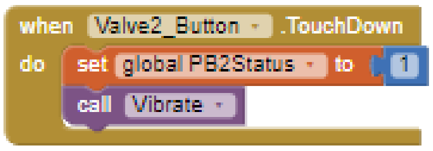
```
when Valve1_Button .TouchUp
do  set global PB1Status to 0
```



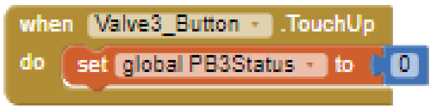
```
when Valve1_Button .TouchDown
do  set global PB1Status to 1
    call Vibrate
```



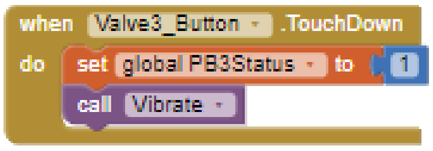
```
when Valve2_Button .TouchUp
do  set global PB2Status to 0
```



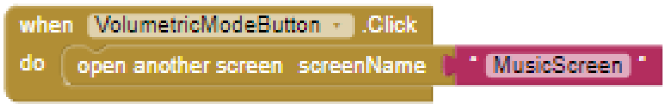
```
when Valve2_Button .TouchDown
do  set global PB2Status to 1
    call Vibrate
```



```
when Valve3_Button .TouchUp
do  set global PB3Status to 0
```

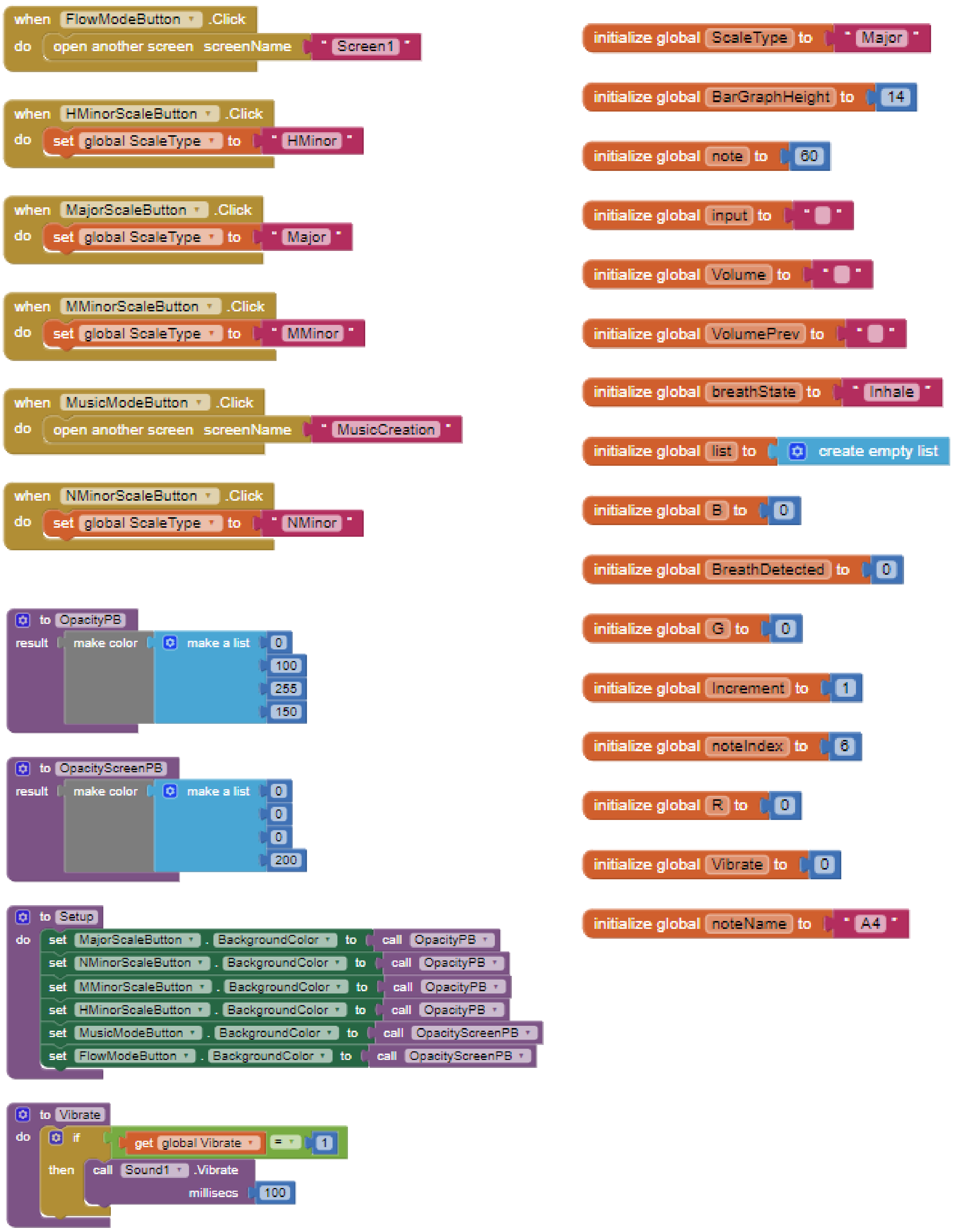


```
when Valve3_Button .TouchDown
do  set global PB3Status to 1
    call Vibrate
```

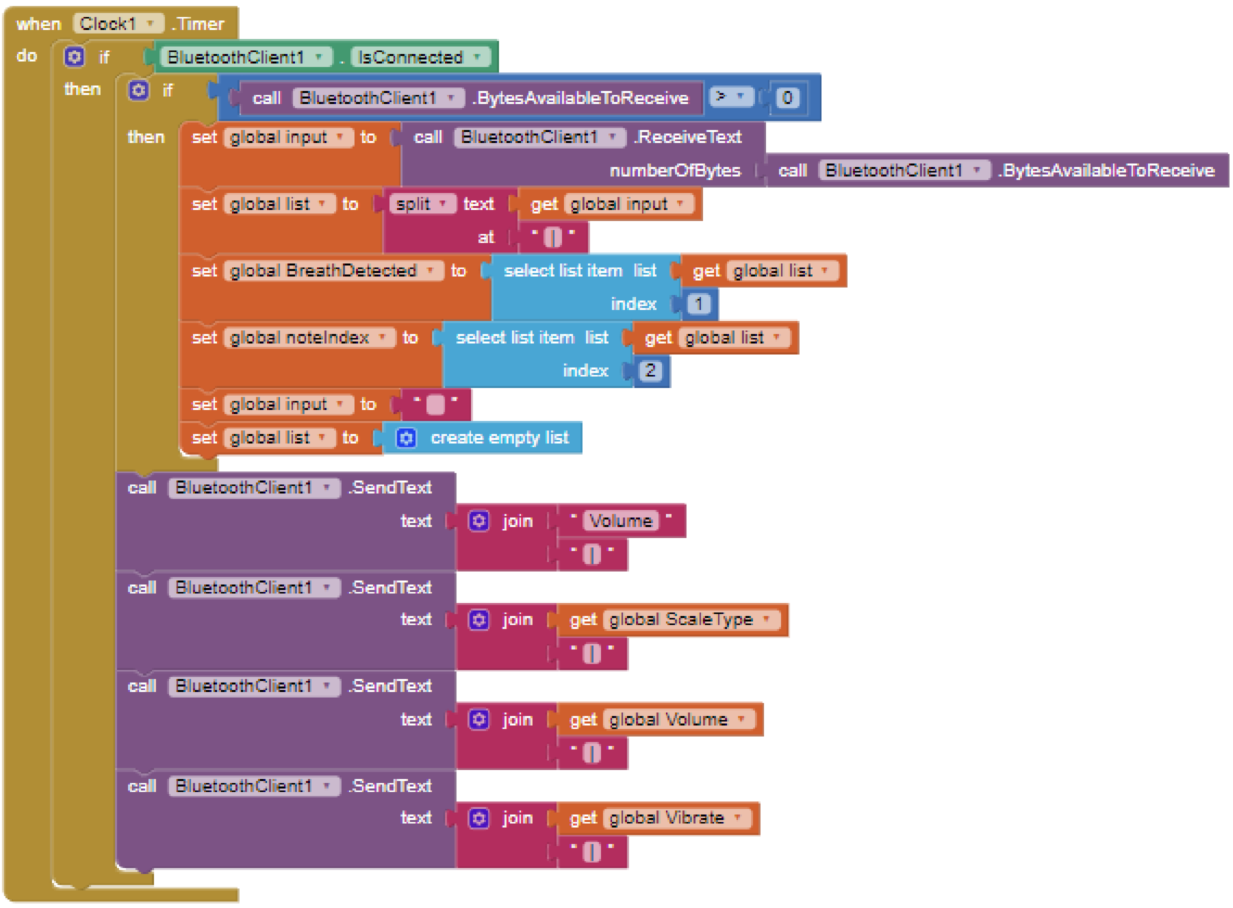


```
when VolumetricModeButton .Click
do  open another screen screenName " MusicScreen "
```

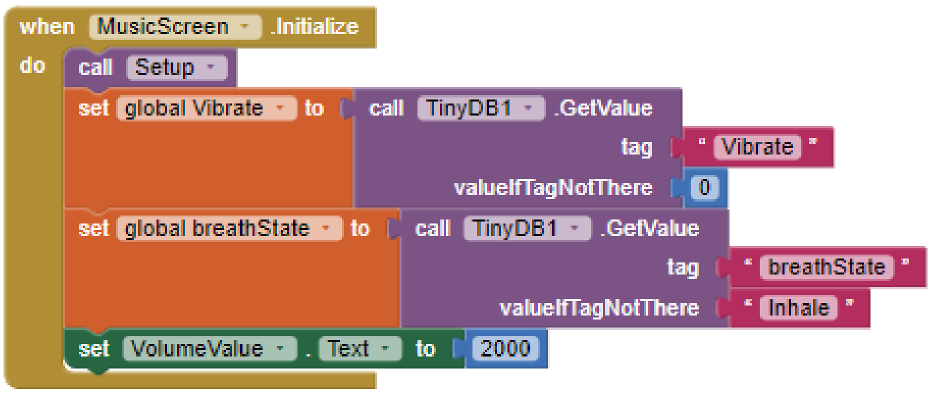
9.5.2.3 Volumetric-Orientated Mode



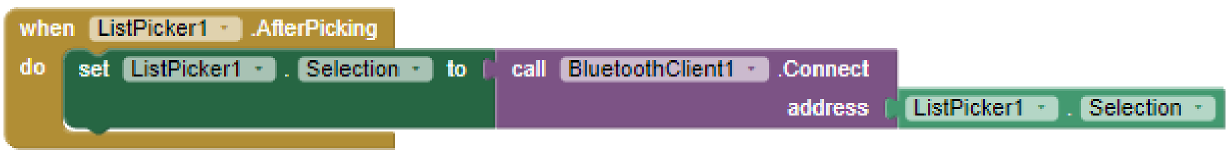
```
when FlowModeButton .Click
do  open another screen  screenName  " Screen1 "

when HMinorScaleButton .Click
do  set global ScaleType to  " HMinor "

when MajorScaleButton .Click
do  set global ScaleType to  " Major "

when MMinorScaleButton .Click
do  set global ScaleType to  " MMinor "

when MusicModeButton .Click
do  open another screen  screenName  " MusicCreation "

when NMinorScaleButton .Click
do  set global ScaleType to  " NMinor "

to OpacityPB
result  make color  make a list  0
                                 100
                                 255
                                 150

to OpacityScreenPB
result  make color  make a list  0
                                 0
                                 0
                                 200

to Setup
do  set MajorScaleButton . BackgroundColor to  call OpacityPB
    set NMinorScaleButton . BackgroundColor to  call OpacityPB
    set MMinorScaleButton . BackgroundColor to  call OpacityPB
    set HMinorScaleButton . BackgroundColor to  call OpacityPB
    set MusicModeButton . BackgroundColor to  call OpacityScreenPB
    set FlowModeButton . BackgroundColor to  call OpacityScreenPB

to Vibrate
do  if    get global Vibrate  =  1
    then  call Sound1 .Vibrate
                    millisecs  100

initialize global ScaleType to  " Major "
initialize global BarGraphHeight to  14
initialize global note to  60
initialize global input to  " "
initialize global Volume to  " "
initialize global VolumePrev to  " "
initialize global breathState to  " Inhale "
initialize global list to  create empty list
initialize global B to  0
initialize global BreathDetected to  0
initialize global G to  0
initialize global Increment to  1
initialize global noteIndex to  8
initialize global R to  0
initialize global Vibrate to  0
initialize global noteName to  " A4 "
```



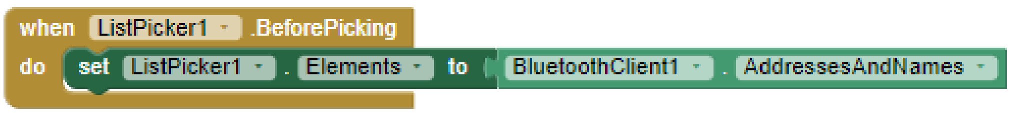
```
when Clock1 .Timer
do  if BluetoothClient1 . IsConnected
    then  if call BluetoothClient1 .BytesAvailableToReceive > 0
          then set global input to call BluetoothClient1 .ReceiveText
                                        numberOfBytes call BluetoothClient1 .BytesAvailableToReceive
               set global list to split text get global input
                                        at " | "
               set global BreathDetected to select list item list get global list
                                                             index 1
               set global noteIndex to select list item list get global list
                                                         index 2
               set global input to " "
               set global list to create empty list
          call BluetoothClient1 .SendText
                              text join " Volume "
                                        " | "
          call BluetoothClient1 .SendText
                              text join get global ScaleType
                                        " | "
          call BluetoothClient1 .SendText
                              text join get global Volume
                                        " | "
          call BluetoothClient1 .SendText
                              text join get global Vibrate
                                        " | "
```



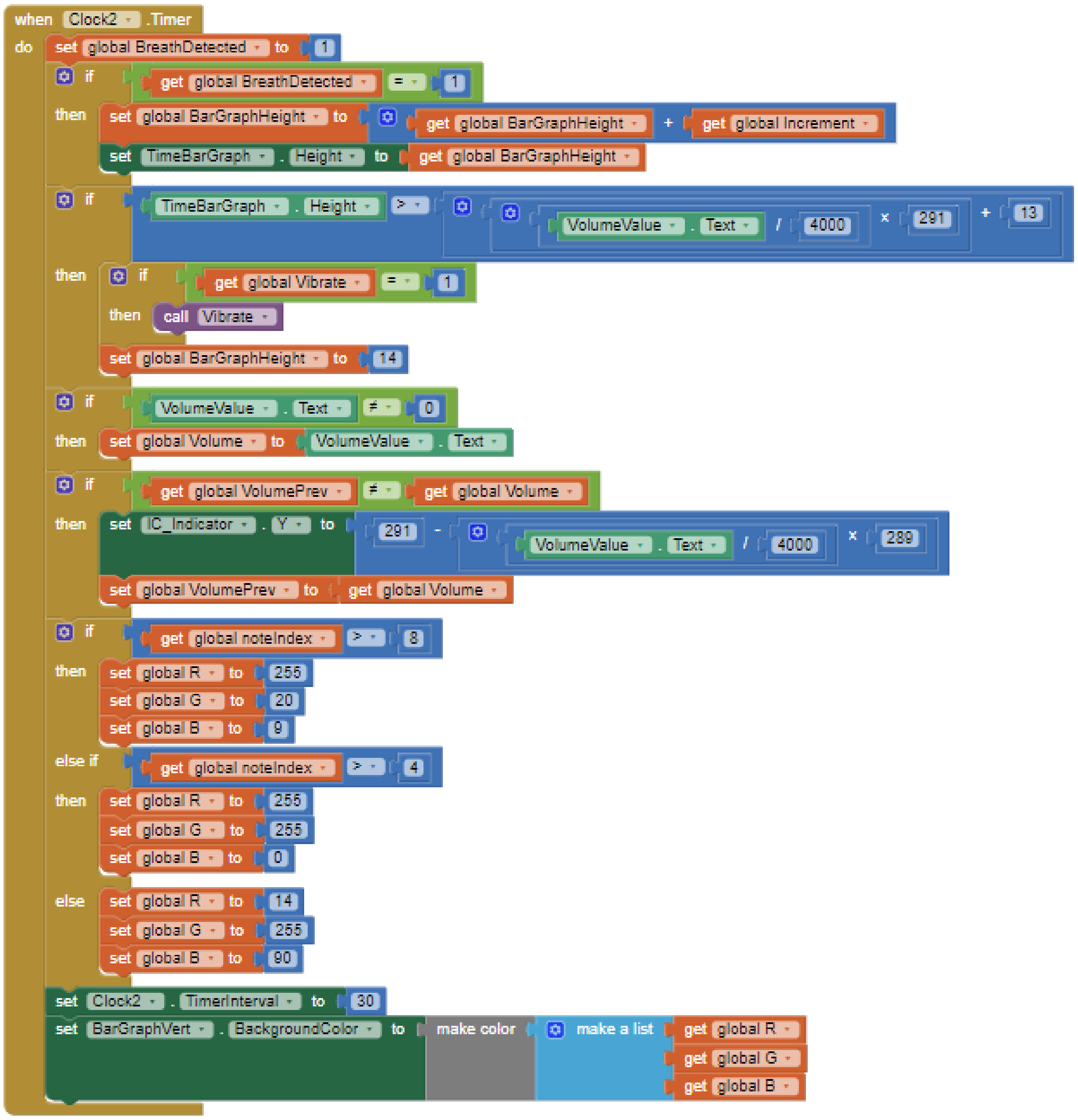
```
when MusicScreen .Initialize
do  call Setup
    set global Vibrate to call TinyDB1 .GetValue
                                     tag " Vibrate "
                       valueIfTagNotThere 0
    set global breathState to call TinyDB1 .GetValue
                                         tag " breathState "
                           valueIfTagNotThere " Inhale "
    set VolumeValue . Text to 2000
```



```
when ListPicker1 .AfterPicking
do  set ListPicker1 . Selection to call BluetoothClient1 .Connect
                                                      address ListPicker1 . Selection
```



```
when ListPicker1 .BeforePicking
do  set ListPicker1 . Elements to BluetoothClient1 . AddressesAndNames
```



```
when Clock2 .Timer
do  set global BreathDetected to 1
    if    get global BreathDetected = 1
    then  set global BarGraphHeight to  get global BarGraphHeight + get global Increment
          set TimeBarGraph . Height to get global BarGraphHeight
    if    TimeBarGraph . Height > VolumeValue . Text / 4000 × 291 + 13
    then  if    get global Vibrate = 1
          then  call Vibrate
          set global BarGraphHeight to 14
    if    VolumeValue . Text ≠ 0
    then  set global Volume to VolumeValue . Text
    if    get global VolumePrev ≠ get global Volume
    then  set IC_Indicator . Y to 291 - VolumeValue . Text / 4000 × 289
          set global VolumePrev to get global Volume
    if    get global noteIndex > 8
    then  set global R to 255
          set global G to 20
          set global B to 9
    else if get global noteIndex > 4
    then  set global R to 255
          set global G to 255
          set global B to 0
    else  set global R to 14
          set global G to 255
          set global B to 80
    set Clock2 . TimerInterval to 30
    set BarGraphVert . BackgroundColor to make color  make a list  get global R
                                                                   get global G
                                                                   get global B
```