Sonification Project Final Report
IoT Apps 2021

Jack Power - 20080169

Introduction

The following report details the research and iterative development completed over the course of the IoT Applications module. The focus of the development was to design and implement a solution based around the concept of sonification. Sonification can essentially be defined as *'the technique of rendering sound in response to data and interactions'* (Hermann, Hunt and Neuhoff, 2011). Sonification is commonly used to provide auditory feedback to a user, augmenting their experience and creating entirely new channels to communicate information. As the preliminary research showed, the human sense of hearing exceeds the others in many aspects, a fact which has largely been underutilised in traditional human-machine interfaces.

The aim of this project was to select a use case and implement a sonification mapping which would generate meaningful audio feedback based on real-time sensor data. To facilitate the collection of many basic physical properties the BBC micro:bit was used. The micro:bit is equipped with sensors such as an accelerometer, magnetometer, microphone and tactile push buttons to name a few. The board also provides the necessary communications protocols such as serial communication and debugging over USB, a fully qualified Bluetooth low energy stack, as well as I^2C and SPI to interface with any additional sensors.

After researching sonification concepts and some existing applications, each team member proposed potential use cases which would use sonification to achieve their goal. Small test applications were implemented using data from the micro:bit which were sonified first using MIDI and later using the Web Audio API within a web app environment.

The web app environment mentioned culminated in a developmental framework integrating the training of machine learning models for gesture recognition with associated sonification mapping rules. A web app approach was selected for the sake of portability, a key feature of many of the proposed use cases being deployment on a Bluetooth-enabled mobile device.

Research

Sonification

The main area of study with regard to sonification is the field of Auditory Display. Auditory Display aims to develop human-machine interfaces which rely on the human auditory system as the primary interface channel. This is in contrast to many conventional user interfaces, where any auditory feedback is included merely as an afterthought, often simply to ornament visual features with pleasing sounds.

There are some very useful aspects of our sense of hearing which provide significant advantages over that of vision. A classic example is the ease with which our ears can discern the various parts of a large collection of acoustic sources, such as in an orchestra. Bregman (1990) theorised that this was achieved by the processes of *grouping* and *streaming*. In nature, sounds which emanate from the same source are likely to share common characteristics, so-called *grouping cues*. These can include the timing of the sounds, if they are perfectly synchronous they are likely to be coming from the same source. We may also group frequencies based on their harmonic relationship, those that share a common fundamental frequency are likely products of the very same source. By grouping these sounds a single source can be identified, and by tracking any common changes over time we can mentally represent this source as an auditory stream. It is by this mechanism that we may discern the orchestra, a single instrument, or the person talking in the row behind us.

An understanding of the way our brain processes and extracts information from sounds, known as psychoacoustics, is fundamental to the design of auditory display interfaces. The behaviour described by Bregman, which he calls Auditory Scene Analysis, gives an indication of how one could integrate many simultaneous auditory streams to achieve the information bandwidth necessary to compete with well-established visual displays.

Due to the relative infancy of sonified feedback, the underlying principles which dictate how our brain responds to various characteristics of sound are still a matter of ongoing research. Carlile (2011) observes that '*in the complexity of real world listening, many factors will contribute to the perception of individual sound sources.*' He goes on to describe these sound sources as perceptual objects, and highlights the importance of understanding '*how variations in the physical properties of the sound will affect different perceptual objects.*'

In contrast to conventional visual displays, which began with simple plain text before new design principles could influence their pragmatic beginnings, sonification is designed to leverage our natural sensory intuition from the ground up.

Micro:bit

The BBC micro:bit is enabled with many onboard peripherals which provide both sensing and output capabilities. Critical to any IoT application, varied means of communication and data transfer are also available to suit a wide array of possible uses. Hardware information (Micro:bit Educational Foundation, 2021a).

Sensing

Buttons

The most obvious available inputs are the two front buttons, A and B. These are debounced in software, which also provides the ability to distinguish a short press, long press and both buttons pressed at once. The state of the buttons may also be exposed on the GPIO pins.

Capacitive Touch Sensors

The logo on the front of the board acts as a touch sensor. The 3 ring connectors numbered 0, 1 and 2 may also be used as capacitive touch sensors, provided the user is also in contact with the ground pin.

Microphone

A MEMS microphone provides sound input to the micro:bit. An LED indicates when the microphone is powered and listening.

Motion Sensor

A combined accelerometer and magnetometer chip provides 3-axis sensing and measurement of magnetic field strength. Onboard gesture recognition is provided, with free fall detection being implemented in hardware and other gestures being available through software (face up, face down and shake). An algorithm in the standard runtime automatically calibrates the chip for use as a compass, allowing for direction readings independent of board orientation.

Light Sensor

The LED matrix which provides the display for the micro:bit is also capable of estimating ambient light level by measuring voltage decay time within the diodes. The phenomenon measured is roughly proportional to ambient light, which provides ten discrete levels of light intensity detection.

Temperature Sensing

The Nordic nRF52 application processor is capable of measuring its own surface temperature. Uncalibrated, this allows for an approximation of ambient temperature ±5°C. It is easily calibrated in software by simply supplying the actual temperature so that an offset can be generated. The sensor has an operating range of -40 to 105°C.

Output

Display

A 5x5 red LED matrix is set on the front of the device. It is capable of 255 levels of light intensity via software control. The 5 columns of LEDs are also coupled to the GPIO pins, along with the associated light sensing mode.

Speaker

The micro:bit v2 has introduced an onboard speaker. It provides a considerable 80dB sound pressure level from 10cm, running on 5V. The sound output is mirrored on the GPIO pins via PWM.

Communications

Bluetooth & BLE

The Nordic S113 coprocessor supplies a fully qualified Bluetooth low energy stack. It operates at 2.4GHz, with 40 2MHz channels.

Micro:bit Radio

Making further use of the same 2.4GHz transceiver, a simple protocol was developed for use between micro:bits. It provides a small-packet broadcast radio interface, with simple user-managed device addressing achieved through radio group codes in the software runtime. 80 channels are provided, capable of 1Mbps transfer speeds with a default MTU of 32 bytes, configurable up to 1024 bytes. The transmission range is approximately 20m. Due to privacy considerations, micro:bits are not uniquely identifiable unless implemented through application code.

USB Interface Chip & SWD

The interface chip supplies a USB communications stack in firmware. This stack allows for the drag-and-drop flashing of application code, as well as serial communication at a rate of 12Mbps. It also provides the interface to the host debugger over CMSIS-DAP.

GPIO

19 assignable GPIO pins are provided, as well as 3 large IO rings (0, 1, 2) and 2 large power rings (3V, GND). Features include external I²C, SPI and UART, as well as up to 3 simultaneous PWM channels, 6 analog inputs and 3 capacitive touch sensor pads.

Platforms

Micro:bit Runtime

The micro:bit runtime written by Lancaster University provides a convenient abstraction of the necessary device drivers used to control the hardware capabilities of the micro:bit. It also provides useful runtime mechanisms such as lightweight multithreading, a scheduler, a memory allocator and management of the protocol stacks.

This runtime for C/C++ development also serves to support the higher level languages which target the micro:bit, such as Microsoft's Block Editor and Code Kingdom's JavaScript and MicroPython implementations.

Use Case Proposals

The following are the two potential use cases proposed after initial research was completed.

Time Management Application

Living under the seemingly perpetual shadow of the COVID-19 pandemic, remote working has become a significant portion of many of our lives. For me, the most persistent challenge brought on by this new (and yet too familiar) environment is to keep focused on the task at hand, and not to be led astray by the many distractions lurking in most of our homes.

I aim to make use of one of the most pervasive aspects of sonification, an awareness of time and of our passage through it. All too often I find that a ten minute coffee break has degenerated into an hour of aimless wandering, spending my time neither well nor enjoyably.

In order to sonify our perception of time, in particular to enforce the consumption of a short break, I propose to implement a timer which gradually increases in frequency and grows more assertive as the designated time expires.

The goal is to suggest a passive but persistent awareness of time as it passes, with the gradual build up hopefully impressing upon the user the increasing need to return to their task, as opposed to a simple alarm which is silenced and quickly forgotten.

The particular parameters are a subject of further research, they may ultimately be user-defined or there may be optimal values supported by psychoacoustic findings. The simplicity of the application makes hardware requirements of little consideration, the best implementation would likely be for a smartphone or similar device.

Rhythmic Dictation Tool

In transcribing music, perhaps the most difficult part lies in determining how long a note is, and when exactly it occurs. Compounding this problem, if one is transcribing a piece it is very likely implied that they have no reference with which to compare. Technology offers a possible solution with the use of scorewriter programs, a user can input their best guess and have it played back to them to determine if they are right. However, installing an entire music notation suite seems to me an overly complicated way of achieving this.

Using a microcontroller such as the BBC micro:bit with onboard motion sensors, one could directly input their desired rhythm simply by shaking the board. Assuming an accurate implementation, this would not only provide a more natural

interface but an absolutely correct representation of the desired rhythm.

Extending this idea, musical information could be loaded onto the board and played back to the user to provide a reference. The user would have the opportunity to train against this rhythm, attempting to synchronise to the beat in a way that has been proven time and time again in countless sonification projects (e.g. The Soundbike. Maes, Lorenzoni and Six, 2019).

Use Case Refinement & Constraints

The use case which was selected for further development was the rhythm dictation tool. It was felt that this use case provided more scope to address the requirements of an IoT application. The various components required to implement the solution were then related to technologies previously explored.

Sensors

- Accelerometer

  Captures board acceleration for use in detecting beats. The existing gesture recognition in the micro:bit runtime is implemented via simple threshold algorithms using only the accelerometer data.

Platforms

- Micro:bit Runtime

  Provides access to sensor data at multiple levels of abstraction. For the sake of tunability raw data is likely the most desirable for this application. Beat recognition may then be implemented using thresholding or perhaps even some machine learning techniques.

Outputs

- Speaker

  To implement the rhythm training aspect of the application, it will be necessary to provide a reference of the ideal timing to the user. Subject to possible sound quality considerations, the micro:bit's onboard speaker should suffice to communicate simple beats of appropriate duration.

Communications

- Wired/Wireless

  The micro:bit offers both wired and wireless communications. Data transfer speeds for both (of the order of Mbps) should prove sufficient for this application, thus power and form factor become the primary concerns. The specific requirements will become apparent as the application is developed.

Cloud

- Remote data storage/interface

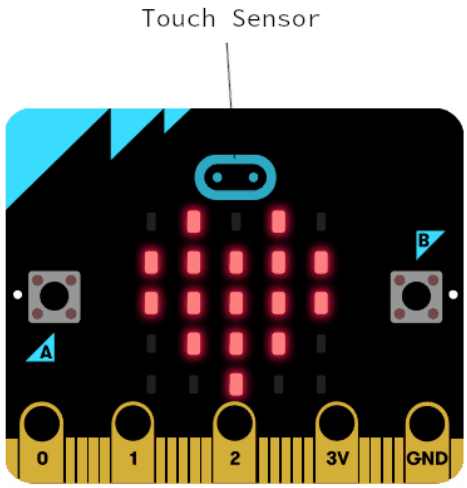
  To make use of available cloud technology, the application could be extended to include a remote archive of recorded rhythms. Depending on the scope of the interface, it would be possible to store and interact with raw accelerometer data, the detected beats and inferred rhythm.

  For example, one could download the generated rhythm in different formats (MIDI, sheet music, etc.). To further develop the training concept, if some accuracy metric could be produced for the collected data against an ideal, a leaderboard could be generated indicating personal best, trends, etc.

Design Adjustments

Sensors

During development of the micro:bit's beat recognition program, the accelerometer data proved too erratic to obtain consistent results. The micro:bit's capacitive touch sensor was therefore substituted which greatly improved the accuracy and reliability of the detection.



(Micro:bit Educational Foundation, 2021b)

Platforms

Initial development of the micro:bit code took place using the MakeCode Python implementation. Due to the modest complexity of the application Python was preferred over the more educational programming alternatives. However, constant automatic rearranging of code and mysterious type errors made development using MakeCode prohibitive. The micro:bit MicroPython implementation was therefore selected instead. This greatly simplified both development and the program architecture, with the library supplying functionality which perfectly suited the application's requirements.

Outputs

Due to time constraints, not all aspects of the application were fully developed. The speaker performs very well in repeating the captured phrase to the user, however this serves more as a confirmation rather than a training reference.

Communications

Serial communication was selected as a straightforward means of transmitting rhythm data. The MicroPython implementation makes this exceedingly trivial, with standard print statements being automatically redirected to the USB serial connection.

Over the course of developing the group web app framework the complexity of Bluetooth communication became apparent. In addition, the behaviour was not consistent among different users making it inherently unreliable and high-maintenance. Nothing was sacrificed by choosing wired communication over wireless, if anything it actually assists in stabilising the micro:bit while entering a rhythm.

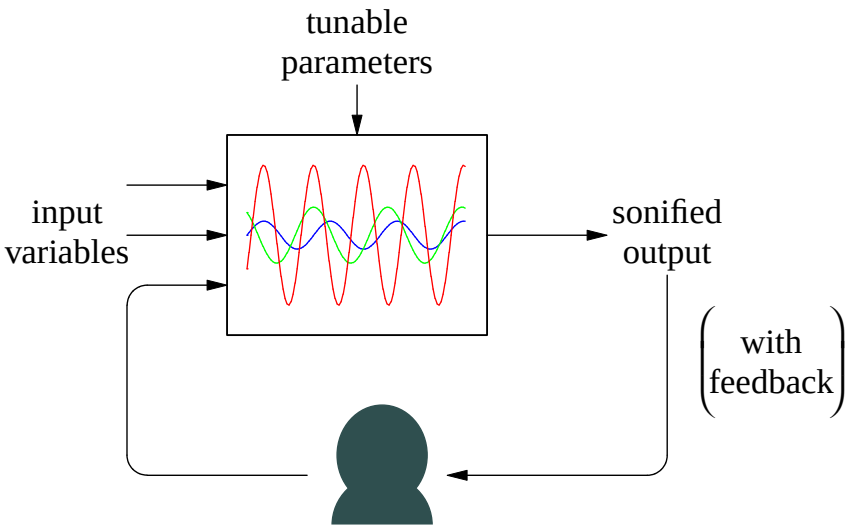
Cloud

Due to last-minute difficulties with the formatting of data for InfluxDB, it was not possible to include the raw sensor data as originally desired. However, the generation and display of music scores was achieved. For the sake of simplicity, the generated score PNGs are served locally using Apache HTTPd and displayed in Grafana via an image URL. Ideally the rhythm data and score generated would be stored in the cloud, using a service such as AWS S3 would be straightforward and would integrate seamlessly into the existing application with the use of the Python Boto3 API.

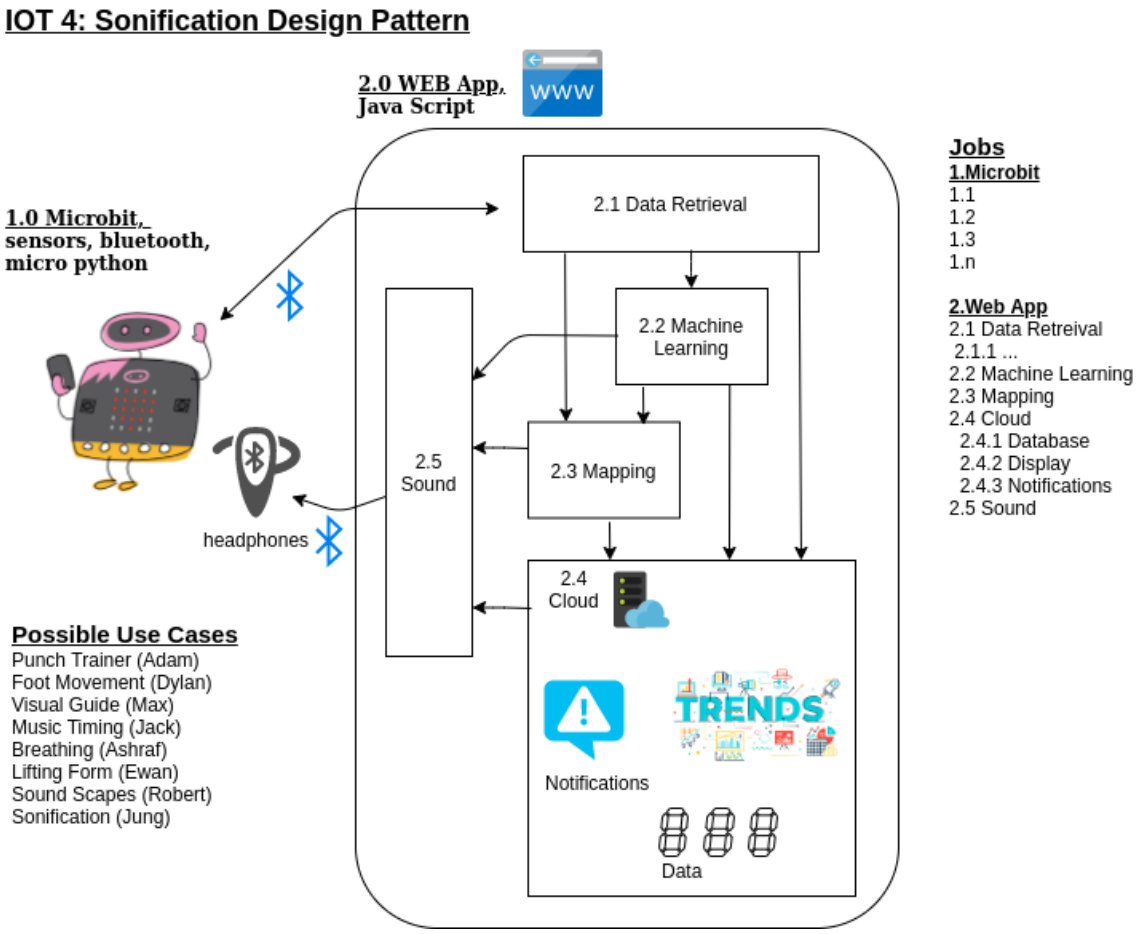
Generic Design Pattern

A generic design pattern was identified which would support many of the suggested use cases. As mentioned previously, it would use a web app based approach to maximise portability. As many of the use cases required some form of gesture recognition as a core component, it was decided to implement a machine learning workflow which would allow each team member, and possibly ultimately any user, to train a model to recognise a specific set of gestures and to allow actions to be performed upon recognition of a gesture. Due to the requirement of applying the design to many different use cases, genericity was kept firmly in mind while developing the framework.

Shown is a high-level conceptual diagram of a generic sonification application.



Below is the realisation of the framework with machine learning, sonification mapping and cloud components. This diagram was created by our lecturer Jason Berry to solidify the discussion and planning that had taken place over previous design meetings.



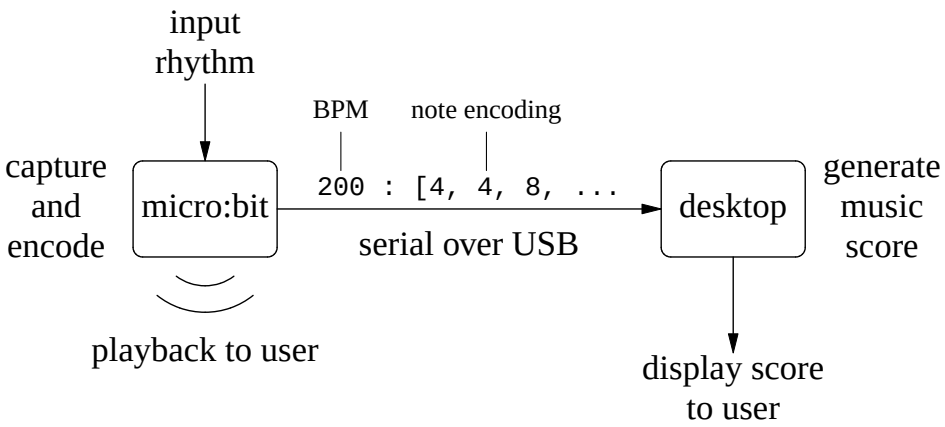
Initial experimentation with implementing sonification using the micro:bit involved sending MIDI signals over a serial connection to a desktop. After deciding on a web-based framework, a toolset was needed which would allow the necessary audio manipulation to be achieved from within the browser. The first iteration of the web app allowed a micro:bit to be connected via Bluetooth and the streamed accelerometer data would affect the playback rate of a HTML5 audio element.

However, in order to satisfy the complex audio manipulation required for a sonification application, a more powerful toolset was needed. The Web Audio API serves this purpose, superseding HTML5 audio and providing far greater control over the sound generated. Research into some existing applications and demonstrations of the API showcased that it was very capable of meeting the needs of practically any sonification application (Bellen, 2016; Racette, 2020).

Further work on the framework involved using TensorFlow to train gesture recognition models and supply mappings to generate sounds. For the purposes of my use case, I felt that machine learning would introduce more complication than was necessary and so I opted to implement my application without using the framework.

Rhythm Dictation Tool Implementation

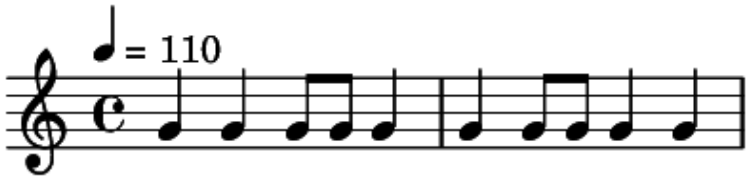
By tapping the micro:bit's touch sensor, a rhythm can be input which is played back via the onboard speaker and communicated to a desktop. A music score is then generated and displayed to the user.



The micro:bit is connected via USB so that the captured rhythm data can be communicated over serial. A Python script polls the serial connection at the desktop end and performs operations on the incoming data. Upon reading a line of rhythm data, a LilyPond source file is generated. LilyPond is a music engraving program which has an input syntax similar to TeX and is capable of producing

output in various formats, here PNG. A bash script calls LilyPond to generate the image file and then uses ImageMagick to crop it for presentation. The file is then moved to a directory where it is served locally by Apache HTTPd. The score can then be viewed by accessing the appropriate link, here achieved through a simple Markdown panel on Grafana. Shown is a sample of the rhythm data communicated over serial and the resulting score generated.

```
110 : [4, 4, 8, 8, 4, 4, 8, 8, 4, 4]
```



The application performs very well in recognising simple rhythms, it is easy to use and the generated score output is exactly as hoped. The main goal of the use case originally described was to allow a user struggling to write out musical rhythm to input the rhythm as they hear it and have the application generate the appropriate notation, so that they may learn through example and train their own ability to transcribe rhythms. Although some additional aspects of the application could not be completed in time, I believe that this original goal has been fulfilled and it would prove to be a useful tool for many developing musicians.

References

Hermann, T., Hunt, A., Neuhoff, J., ed., (2011). *The Sonification Handbook*, Berlin, Germany: Logos Verlag.

Bregman, A. (1990). *Auditory Scene Analysis: The Perceptual Organisation of Sound*. Cambridge: MIT Press.

Carlile, S. (2011). Psychoacoustics. In: T. Hermann, A. Hunt, J. Neuhoff, ed., *The Sonification Handbook*, 1st ed. Berlin, Germany: Logos Verlag. pp. 61-81.

Micro:bit Educational Foundation, (2021a). *Hardware*. [online] Available at: https://tech.microbit.org/hardware/ [Accessed 26 May 2021]

Maes, P., Lorenzoni, V., Six, J. (2019). The Soundbike: Musical Sonification Strategies to Enhance Cyclists' Spontaneous Synchronisation to External Music. *Journal on Multimodal User Interfaces*, 13(3), pp. 155-166.

Micro:bit Educational Foundation, (2021b). *Make it: code it*. [image] Available at: https://www.microbit.org/projects/make-it-code-it/ [Accessed 26 May 2021]

Bellen, S. (2016). *Changing live audio with the web-audio-api - JSConf Budapest 2016*. YouTube. Available at: https://www.youtube.com/watch?v=BEdFcnI-ppk [Accessed 26 May 2021]

Racette, M. (2020). *Soundscape*. [online] Github. Available at: https://github.com/mracette/soundscape [Accessed 26 May 2021]