Potential Tools & Technologies to Implement Rhythmic Dictation
IoT Apps 2021

Jack Power - 20080169

List of Technologies

Sensors

- Accelerometer

  Captures board acceleration for use in detecting beats. 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.

Mapping of Technology to Module Descriptor

The application code will interpret sensor input and communicate the data to a cloud-based service (e.g. AWS).	⟶	Design, develop and implement software applications that interconnect physical devices and sensors, mobile devices and cloud-based services.
Many development environments are offered for micro:bit development, both web-based and local (e.g. MakeCode, Python). Other environments will likely be required for the development of cloud services (e.g. AWS console or scripting), and also for the training of ML models (Jupyter Notebook).	⟶	Evaluate and use a suitable development platform for IoT applications.
For whatever features are ultimately implemented, a software interface will be provided on the device side so that data is neatly encapsulated and easily conceptualised.	⟶	Design, develop and implement software APIs to access physical sensors, actuators and devices.

Data captured by the application will be stored remotely in various forms for the purpose of archiving and presentation to the user.	⟶	Integrate cloud-based data storage solutions with device and sensor data.
Depending on the distribution of computing tasks, the data communicated to the cloud may be in various stages of processing. If it should prove that the micro:bit is capable of completing most of these tasks locally, it will not be necessary to expose real-time sensor data to the cloud.	⟶	Integrate suitable messaging, gateway and middleware solutions to develop applications that provide direct, real-time access to low level devices and sensors.
With a sufficiently sophisticated web-based interface, it will be necessary to convert and present the data in many different forms depending on the user's requirements.	⟶	Develop a knowledge discovery based web application that utilises the storage and computational power of the cloud.