The Electronic Metronome

1) Fixing the Perception of Time

According to Dawson J. and Sleek S. in their online article “The Fluidity of Time”, the human judgment of time varies wildly depending on one’s emotional state. Keeping time is vital for all types of professions, but none more-so than for musicians who, not only have to ensure their performances are identical to one another, but also that they are also consistent throughout the score.

This is a common problem you can easily find on any Music Forum, and often the solution is to use a metronome. In fact, one Stack Exchange post from Raskolnikov strongly recommends the use of a metronome “especially when the rhythms are trickier”.

Metronomes themselves have been around and used since 1812, but none of them let the user customise their own tempo, or save the desired tempo to a cloud storage. That is what our team is trying to rectify with the aid of the Arduino Yun.

References

Stack Overflow Forum:
https://music.stackexchange.com/questions/837/how-useful-is-a-metronome-for-playing-the-piano

Fluidity of Time Article:
https://www.psychologicalscience.org/observer/the-fluidity-of-time

2) The Method of the Madness

Our idea was to take a sound input from the user for a set amount of time. Any sounds that were loud enough not to be considered “background noise” were to be added to a counter, which in turn would be divided into a minute in order to figure out how long the interval was between each beat.

This would ensure that the tempo evenly divided into a minute and that it was not constrained to the 40 – 208 BPM of a regular metronome.

The electronic metronome would then save this interval to a Google Sheet, which was set up specifically for this product, over the internet so that it could be called on and reused as desired.

3) Building the Machine

a) Recording and Displaying Tempo

For this section of the problem, we used an analogue input microphone to port A0 to record the sounds from the user, and set a const int THRESHOLD of 150 to use as a minimum value in order to avoid picking up unrelated audio.

To see the fruits of our labour, we added a digital output LED in port 7 to flash twice before the Electronic Metronome was ready to record, and to flash once for 2 milliseconds between each interval.

We then made the following function to calculate the BPM:

```
// CALCULATES HOW MANY BEATS THERE ARE IN A MINUTE BASED ON THE INPUT OF THE MIC
void GetBeats(int sound)
{
  a = millis(); // Getting initial time here so it doesn't update within the loop
  while(b < 6000) // Until 6 seconds have passed
  {
    if(sound > THRESHOLD)
    {
      beat++; // increments the number of beats
    }
    if(digitalRead(BUTTON) == HIGH)
    {
      interval = GetLastBeat();
      break;
    }
    sound = analogRead(MIC);
    b = millis() - a; // this will update the length of time stored in b
  }// end while

  // avoiding 'division by 0' error
  if(beat == 0)
  {
    beat++;
  }

  // ensuring that 'interval' doesn't get re-written if the value was requested from Google Sheets
  if(interval == 0)
  {
  interval = (MINUTE / beat) - 200;
  }
}// end GetBeats()
```

b) Sending and Receiving Data

With the use of PushingBox, GoogleSheets and Google Developer, we were able to add a second digital input switch to let the program know if the user wanted to re-use the last recorded tempo, instead of clapping to create a beat. When pushed, the switch moved the program to a second function called GetLastBeat() as seen here:

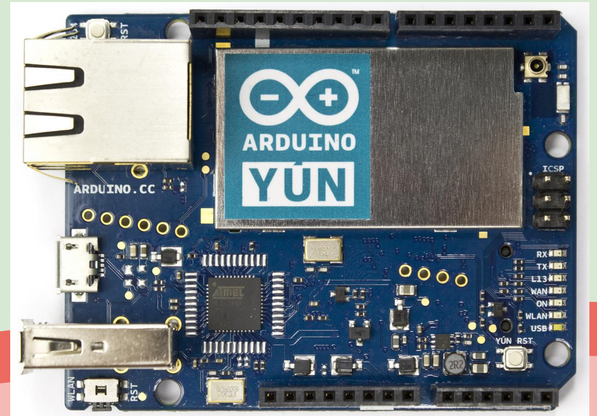
```
// REQUESTS THE LAST SAVED 'interval' VALUE ON GOOGLE SHEETS
unsigned long GetLastBeat()
{
// Initializing the Client Library
    HttpClient client;

  String intervalRequest = "https://sheets.googleapis.com/v4/spreadsheets/858919912724-gcmeice7emiijf9ubpj6fktdcfpk7vvm.apps.googleusercontent.com/values/C2:C2";
  return client.get(intervalRequest);
}// end GetLastBeat
```

4) Locking Down the System

As a small measure of security, Google Developer allowed us to restrict the use of the Google Sheets API to Http requests from our specified Google Sheets page only.

This made sure that no other website, Android, ISO other than what we gave exceptions to could access our data or be accessed by our API.



Arduino Yun Board
Image Taken from:
https://commons.wikimedia.org/wiki/File:ArduinoYunFront_2.jpg

5) Onwards to Greatness

Admittedly, our Arduino Yun based Electronic Metronome is a mess of parts with very little thought put in to the ergonomics of UI. If we were to continue working on this project, we would first implement a console UI to choose from possible options such as:

i) Favourite Tempo
Simply put, this would be an option to scroll through a column of the Google Sheet specified as ‘Favourites’ that would either have a 0 or 1 depending on whether it is actually on the list or not, and then giving the user the option to set one of these as their desired interval.

ii) Time Signature
With the aid of a second digital output LED, we could show the desired time signature by initialising the first beat with both LEDs, and all subsequent beats with a single LED.

iii) Search Scores with Tempo
This would require to give the Electronic Metronome the permission to access a web browser to search for scores of the specified BPM.

Author and Resources

Name: Abigail Herron
Student ID: S00200536
Year: 1 **Group:** D

More Information on Electronic Metronome Project
Trello: https://trello.com/b/inEctDv3/iot-project

GitHub: https://github.com/AbigailHerron/MetronomeProject

Google Sheet:
https://docs.google.com/spreadsheets/d/1HVSxekao9DAXwtTgJMPxqG1Bi8GDmG-GuQAPQ4u0X_E/edit#gid=0