Lab 4: Speaker and Tones

Objectives

1. Understand the concept of tones (notes) and a speaker
2. Practice more complex sketches (switch-case, loop)

Reference:

https://docs.m5stack.com/en/arduino/m5core/speaker

https://www.arduino.cc/reference/en/#structure

Music Notes

Invented by Thomas A. Edison, a speaker is a small device that will convert an electric current into physical sound. By generating a frequency (pulse) to create a magnetic field for vibrating a diaphragm whose resonation generates the corresponding sound.

To make a musical note, the vibration has to match the frequency of the corresponding note. The standard frequency for each note is presented in the table below. The frequencies of higher/lower pitches notes are the double/half of the standard note. (From the table, the frequency of C' is double of that of C.)

Note	Exact Frequency (Hz)	Frequency Used (Hz)
c	130.81	131
C	261.63	262
D	293.67	294
E	329.63	330
F	349.23	349
G	392.00	293
A	440.00	440
B	493.89	494
C'	523.26	524

Note*: Frequency of C’ is double of C and frequence of c is half of C.

NOTE	OCTAVE 0	OCTAVE 1	OCTAVE 2	OCTAVE 3	OCTAVE 4	OCTAVE 5	OCTAVE 6	OCTAVE 7	OCTAVE 8
C	16.35 Hz	32.70 Hz	65.41 Hz	130.81 Hz	A piano middle C 261.63 Hz	523.25 Hz	1046.50 Hz	2093.00 Hz	A piano's highest note 4186.01 Hz

Melody/Rhythm

A melody is a sequence of musical notes. Rhythm is the timing and duration of each note in the melody. To create a melody, we need to create a sequence of notes with the duration between each note.

Arduino Tone functions

Arduino has provided a programmer with functions related to tone.

- `M5.Speaker.tone(frequency)`: Sets the speaker to sound at a frequency.
- `M5.Speaker.beep()`: Sets speaker to beep.
- `M5.Speaker.mute()`: Stops the speaker to make sound.
- `M5.Speaker.setVolume(level)`: Sets the volume to level (0 to 11 loudest).

The example codes show how to make the C note for 1 second. (see highlight)

```
#include <M5Stack.h>

void setup() {
    M5.begin();
    M5.Speaker.tone(262);
    delay(1000);
    M5.Speaker.mute();
}
```

More Arduino Language Reference

'for'

The `'for'` is useful for repeating a block of statements (especially for a number of times). The `'for'` structure has three parts, initialize, test, and update. In this example, the system will repeat the `println()` statement for 100 times (0 to 99).

```
for (int i = 0; i < 100; i++) {
  println(i); // print 0 to 99
}
```

'while'

The `'while'` loop will run indefinitely as long as the test expression is `true`.
Example

```
int a = 0;
while (a < 100) {
    a++;
}
while (true) {
    // this is an infinite loop
}
```

'do ... while'

The `'do .. while'` loop is similar to the while loop. However, the test expression is tested at the end of the loop. This means the **loop will always run at least once**. Example

```
int a = 0;
do {
    a++;
} while (a < 100);
```

Array

An array is a collection of variables that are accessed with an index number. Arrays in the C++ programming language Arduino sketches are written in can be complicated, but using simple arrays is relatively straightforward.

```
// Declare an array without explicitly choosing a size (the compiler
// counts the elements and creates an array of the appropriate size):
int factors[] = {4, 4, 2, 4, 1, 2, 2, 4};

// accessing array using array name and index
int wait = 1000/factors[3];

// Finding the number of elements in an array
int n_factor = sizeof(factors) / sizeof(factors[0])
```

The first element of the array is at index 0, and the last element is at index n-1 when n is the total number of elements in the array. Accessing array out of its boundary can cause unexpected result.

To access each element of the array using **for-loop**, see the following code.

```
for (int i = 0; i < n_factor; ++i) {
    wait = 1000/factors[i];
}
```

Lab Exercises

Task 1: Serial Piano (practice switch-case statement)

Complete the given sketch below to read a key from serial output and play a note. For the keys: 'c', 'd', 'e', 'f', 'g', 'a', 'b', 'C', 'D', 'E', 'F', 'G', 'A', 'B', play the corresponding note C, D, E, F, G, A, B, C', D', E', F', G', A', B' for one second. For any other input, play a lower C (half the frequency of normal C) for one second.

```
#include <M5Stack.h>

#define C 262
#define D 294
#define E 330
#define F 349
#define G 392
#define A 440
#define B 494

void setup() {
  M5.begin();
  M5.Speaker.setVolume(2); // 0-11 (loudest)
}

int note_freq = 0;

void loop() {
  if (Serial.available() > 0) {
    menu();
    int key = Serial.read();
    Serial.write(key);
    switch (key) {
      case 'c' :
        note_freq = C;
        break;
      case 'C' :
        note_freq = 2*C;
        break;
      default:
        break;
    }
    M5.Speaker.tone(note_freq);
    delay(1000);
    M5.Speaker.mute();
    delay(50);
   }
}

void menu() {
   Serial.println("\nTask 1:\nPlease enter note [c..b, C..B]");
   Serial.print("> ");
}
```

Task 2: Short Melody

Try to understand the given pseudo and code that plays a short melody.

```
DEFINE:
  notes, melody,
  factors (used to calculate delay for each note)
  base_delay, total notes in melody

SETUP:
  Set speaker volume

LOOP:
  for each note in melody:
    calculate wait time for delay based on base_delay and factor
    play current note in melody with wait time
  pause for 2 seconds before playing the melody again
```

```
#include <M5Stack.h>

#define C 262
#define D 294
#define E 330
#define F 349
#define G 392
#define A 440
#define B 494

const int melody[] = { G, G, G, D, E, E, D, B, B, A, A, G};
const int factors[] = { 4, 4, 4, 4, 4, 4, 2, 4, 4, 4, 4, 2};
const int base_delay = 1000; // 1 second
int total_notes = sizeof(melody)/sizeof(melody[0]);

void setup() {
  M5.begin();
  M5.Speaker.setVolume(1); // 0-11 (loudest)
}

void loop() {
  for (int i=0; i<total_notes; i++) {
    int wait = 1000/factors[i];
    M5.Speaker.tone(melody[i]);
    delay(wait);
    M5.Speaker.mute();
    delay(50);
  }
  delay(2000); // pause for 2 second
}
```

Task 3: Short Melody with Button

Modify the sketch from task 2 to play subsequent notes of the short melody for the duration **when the button A is pressed and hold**. After releasing the button, the note played should stop, and the song will proceed to the next note when the button A is pressed and hold again. When the last note is played, the next note will be the first note in the melody.

```
DEFINE:
  notes, melody
  initialize i to be the first index in melody

SETUP:
  Set speaker volume

LOOP:
  check for button pressed
    play current note
  check for button released
    stop play sound
    set index to next note in the melody
```

Task 4: Buttons to control speed (0.25x, 0.5x, 1x, 1.5x, 2x)

Use the melody from the previous task and make the playback speed adjustable across 5 levels (**0.25x**, **0.5x**, **1x**, **1.5x**, **2x**). **Button C** increases the speed, **Button A** decreases it, and **Button B** resets it to the default (**1x**). The LCD should show the current speed. Follow the provided pseudocode.

```
DEFINE:
  notes, melody, factors
  speeds = {0.5, 0.75, 1.0, 1.5, 2.0};
  set speed to normal (1.0)

SETUP:
  set speaker volume

LOOP:
  for each note to play
    check button C pressed
      increase speed if not reach fastest
    check button A pressed
      decrease speed if not reach slowest
    check button B pressed
      reset speed to normal
    calculate wait time from base_delay, factor and speed
    show current speed to LCD
    play current note in the melody with wait time
  pause for 2 seconds before playing the melody
```

Group

Lab 4: Speaker and Tone

Section: _____ Date: _____

Members

Name: _____ Student ID: _____

Name: _____ Student ID: _____

Name: _____ Student ID: _____

Task		Graded by
1	Serial piano	
2	Short melody	
3	Short melody with buttons (play note when press and hold a button, no sound when release)	
4	Buttons to control speed (**0.5x, 0.75x, 1.0x, 1.5x, 2.0x**)	

5. Reflection about this lab. See assignment in **MycourseVille**