|  |
| --- |
| // \* Star Wars Song Selector |
|  | // \* Original Composition : Star Wars |
|  | // \* Composed by : I do not know |
|  | // \* Coded By : https://code.google.com/p/rbots/source/browse/trunk/StarterKit/Lesson5\_PiezoPlayMelody/Lesson5\_PiezoPlayMelody.pde |
|  | // \* Use BSD Clause 2 License for Distribution |
|  | // \* Collection by GitHub User @abhishekghosh - https://github.com/AbhishekGhosh/Arduino-Buzzer-Tone-Codes |
|  | // \* Program to choose between two melodies by using a potentiometer and a piezo buzzer.  //\* Program borrowed by Kirtan Jha to edit.//\*  //\*Task: This program is using potentiometer to control two melodies. Your task is to remove portions of potentiometer and use two push buttons to control the melodies.//8 |
|  |  |
|  | // TONES // |
|  | // Defining the relationship between note, period & frequency. |
|  |  |
|  | // period is in microsecond so P = 1/f \* (1E6) |
|  |  |
|  | #define c3 7634 |
|  | #define d3 6803 |
|  | #define e3 6061 |
|  | #define f3 5714 |
|  | #define g3 5102 |
|  | #define a3 4545 |
|  | #define b3 4049 |
|  | #define c4 3816 // 261 Hz |
|  | #define d4 3401 // 294 Hz |
|  | #define e4 3030 // 329 Hz |
|  | #define f4 2865 // 349 Hz |
|  | #define g4 2551 // 392 Hz |
|  | #define a4 2272 // 440 Hz |
|  | #define a4s 2146 |
|  | #define b4 2028 // 493 Hz |
|  | #define c5 1912 // 523 Hz |
|  | #define d5 1706 |
|  | #define d5s 1608 |
|  | #define e5 1517 // 659 Hz |
|  | #define f5 1433 // 698 Hz |
|  | #define g5 1276 |
|  | #define a5 1136 |
|  | #define a5s 1073 |
|  | #define b5 1012 |
|  | #define c6 955 |
|  |  |
|  | #define R 0 // Define a special note, 'R', to represent a rest |
|  |  |
|  |  |
|  | // SETUP // |
|  |  |
|  | int speakerOut = 10; // Set up speaker on digital pin 7 |
|  | int potPin = A0; // Set up potentiometer on analogue pin 0. |
|  |  |
|  |  |
|  | void setup() { |
|  | pinMode(speakerOut, OUTPUT); |
|  | Serial.begin(9600); // Set serial out if we want debugging |
|  | } |
|  | //} |
|  |  |
|  | // MELODIES and TIMING // |
|  | // melody[] is an array of notes, accompanied by beats[], |
|  | // which sets each note's relative length (higher #, longer note) |
|  |  |
|  | // Melody 1: Star Wars Imperial March |
|  | int melody1[] = { a4, R, a4, R, a4, R, f4, R, c5, R, a4, R, f4, R, c5, R, a4, R, e5, R, e5, R, e5, R, f5, R, c5, R, g5, R, f5, R, c5, R, a4, R}; |
|  | int beats1[] = { 50, 20, 50, 20, 50, 20, 40, 5, 20, 5, 60, 10, 40, 5, 20, 5, 60, 80, 50, 20, 50, 20, 50, 20, 40, 5, 20, 5, 60, 10, 40, 5, 20, 5, 60, 40}; |
|  |  |
|  | // Melody 2: Star Wars Theme |
|  | int melody2[] = { f4, f4, f4, a4s, f5, d5s, d5, c5, a5s, f5, d5s, d5, c5, a5s, f5, d5s, d5, d5s, c5}; |
|  | int beats2[] = { 21, 21, 21, 128, 128, 21, 21, 21, 128, 64, 21, 21, 21, 128, 64, 21, 21, 21, 128 }; |
|  |  |
|  | int MAX\_COUNT = sizeof(melody1) / 2; // Melody length, for looping. |
|  |  |
|  | long tempo = 10000; // Set overall tempo |
|  |  |
|  | int pause = 1000; // Set length of pause between notes |
|  |  |
|  | int rest\_count = 50; // Loop variable to increase Rest length (BLETCHEROUS HACK; See NOTES) |
|  |  |
|  | // Initialize core variables |
|  | int toneM = 0; |
|  | int beat = 0; |
|  | long duration = 0; |
|  | int potVal = 0; |
|  |  |
|  | // PLAY TONE // |
|  | // Pulse the speaker to play a tone for a particular duration |
|  | void playTone() { |
|  | long elapsed\_time = 0; |
|  | if (toneM > 0) { // if this isn't a Rest beat, while the tone has |
|  | // played less long than 'duration', pulse speaker HIGH and LOW |
|  | while (elapsed\_time < duration) { |
|  |  |
|  | digitalWrite(speakerOut,HIGH); |
|  | delayMicroseconds(toneM / 2); |
|  |  |
|  | // DOWN |
|  | digitalWrite(speakerOut, LOW); |
|  | delayMicroseconds(toneM / 2); |
|  |  |
|  | // Keep track of how long we pulsed |
|  | elapsed\_time += (toneM); |
|  | } |
|  | } |
|  | else { // Rest beat; loop times delay |
|  | for (int j = 0; j < rest\_count; j++) { // See NOTE on rest\_count |
|  | delayMicroseconds(duration); |
|  | } |
|  | } |
|  | } |
|  |  |
|  | // LOOP // |
|  | void loop() { |
|  | potVal = analogRead(potPin); //Read potentiometer value and store in potVal variable |
|  | Serial.println(potVal); // Print potVal in serial monitor |
|  |  |
|  | if (potVal < 511) { // If potVal is less than 511, play Melody1... |
|  |  |
|  | // Set up a counter to pull from melody1[] and beats1[] |
|  | for (int i=0; i<MAX\_COUNT; i++) { |
|  | toneM = melody1[i]; |
|  | beat = beats1[i]; |
|  |  |
|  | duration = beat \* tempo; // Set up timing |
|  |  |
|  | playTone(); // A pause between notes |
|  | delayMicroseconds(pause); |
|  | } |
|  | } |
|  | else // ... else play Melody2 |
|  | for (int i=0; i<MAX\_COUNT; i++) { |
|  | toneM = melody2[i]; |
|  | beat = beats2[i]; |
|  |  |
|  | duration = beat \* tempo; // Set up timing |
|  |  |
|  | playTone(); // A pause between notes |
|  | delayMicroseconds(pause); |
|  | } |
|  | } |