

Arduino Synth

Processadores de Efeitos DIY



Digitópia

A Digitópia é uma plataforma de música digital sediada na Casa da Música, no Porto, que incentiva a audição, a performance e a criação musical. Baseando-se em ferramentas digitais, embora não exclusivamente, enfatiza a criação musical colaborativa, o design de software, a educação musical e a inclusão social, promovendo a emergência de comunidades multiculturais de performers, compositores, curiosos e amantes de música. Com uma presença física composta por 12 computadores e diversos controladores musicais, recebe alunos de várias escolas, promove seminários, é visitada por curiosos, turistas, crianças e seniores, proporcionando experiências musicais auditivas e criativas, enriquecedoras para todos aqueles que as vivenciam.

página: digitopia.github.io

página oficial: www.casadamusica.com/digitopia

facebook: www.facebook.com/DigitopiaCasaDaMusica

grupo faceboo: www.facebook.com/groups/digitopiacasadamusica



Digitópia Collective

Singular no panorama nacional, o Digitópia Collective é constituído pelos formadores do Serviço Educativo associados à Digitópia, a plataforma artística da Casa da Música reservada à criação musical em suporte tecnológico.

No seu trabalho, o ensemble aplica processos e modelos tão diversos quanto o design de instrumentos digitais, a concepção de hardware próprio, o circuit-bending, a exploração das relações entre imagem e som, a prática de VJ's e DJ's, a digital media ou os sistemas digitais interactivos. Da confluência de linguagens, trazidas por cada elemento do grupo, surge um repertório de música electrónica e digital com um declarado carácter performativo.



Tiago Ângelo

website: plnh0.github.io
mail: plnh0.c0d1ng@gmail.com

Académico

- Conservatório de Música de Coimbra
- Música Electrónica e Produção Musical - ESART/IPCB
- Mestrado Multimédia, Perfil de Sound Design e Música Interactiva - FEUP/UP

Profissional

- Freelancer / Digitópia - Casa da Música / Sonocospia
- Composição e media interactivos para artes performativas (DEMO, Radar360)

Artístico

- Composição e performance Electrónica
- Instalações interactivas
- Construção de instrumentos musicais e desenvolvimento de software musical



Hello World



Princípios da electricidade (caso necessário):
<https://learn.sparkfun.com/tutorials/voltage-current-resistance-and-ohms-law>

yin-yang : balance

Plano (sábado)

Manhã

- (Breve) Introdução ao Arduino e Áudio Digital
- Gerar um sinal áudio
- Entrada áudio com piezo
- Feedback Delay
- Controlo de parâmetros

Tarde

- Controlo do tamanho de delay + Dicas para melhorar o processamento
- Circuitos 1 - melhorar a qualidade do sinal de saída
- Circuitos 2 - entrada de linha
- Extras: outros exemplos





<http://arduino.cc/>

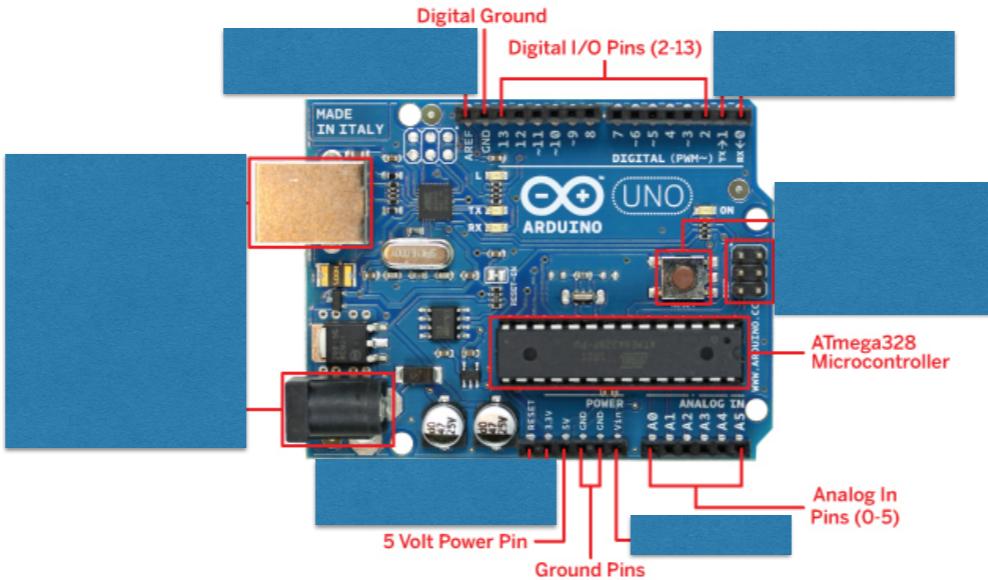


retirado de <http://makezine.com/2015/03/19/massimo-banzi-fighting-for-arduino/>

- Plataforma de prototipagem electrónica *open source*;
- Desenvolvida por Maximo Banzi, Tom Igoe, David Cuartielles, David Mellis e Gianluca Martino que começou por ser desenvolvida em 2005 em Ivrea, Itália;
- Documentário: <https://vimeo.com/18539129>



Hardware



retirado de <https://myp-tech.wikispaces.com/+The+Arduino+Project>



TX significa *transmit*
RX significa *receive*
Para que serve o pin AREF ? (ver link: <http://blog.arduino.cc/2010/12/13/tutorial-arduino-and-the-aref-pin/>)

Software (IDE)



- IDE - *Integrated Development Environment*
- permite editar e carregar um programa para o microcontrolador
- semelhante ao Processing
- baseado em C++

Arduino - 101



- Exemplos Arduino

- Abrir “Blink.ino”

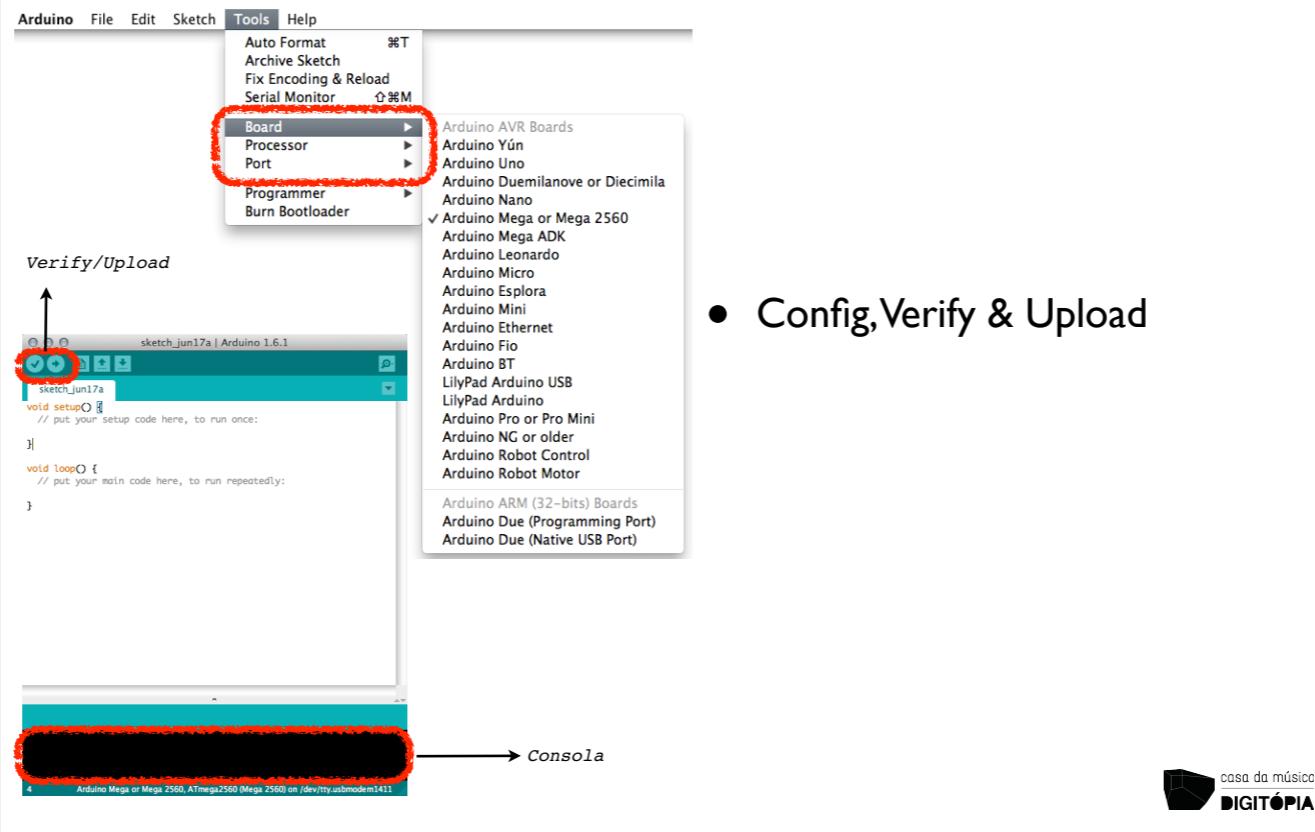


NOTA: para instalar os exemplos fornecidos com este tutorial bastará arrastar a pasta 'ArduinoSynth_examples' (descarregada no github) para ~/Documentos/Arduino/libraries/

Os exemplos referidos em cada página deste documento serão assinalados por baixo do título. Exemplo:

[~/ArduinoSynth_examples/Dial_Efeitos/_01_LED_LigarDesligar.ino](#)

ARDUINO - 101



● Config, Verify & Upload

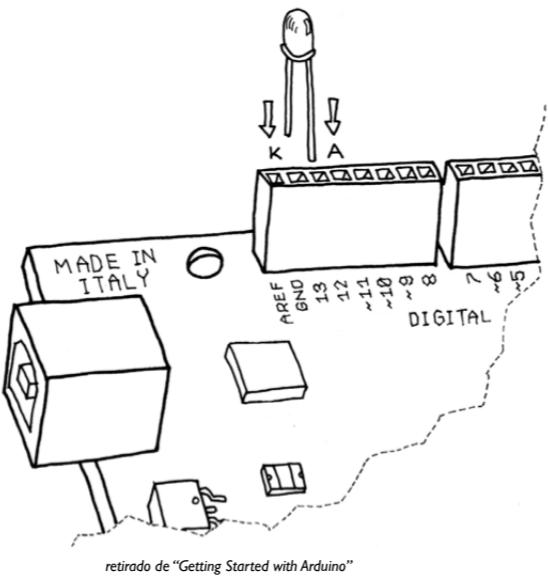
Antes de carregar código para a placa Arduino é necessário configurar o IDE de modo a corresponder à placa utilizada.
Para tal deverá escolher a sua placa, o processador que esta possui e a identificação da porta de comunicação.

Para verificar se o código contém erros deverá carregar no botão 'v' (verify) ou alternativamente navegar através do menu da aplicação em 'Sketch —> Verify/Compile'.
A informação relativa à verificação/compilação será então disposta na consola.

Se o código não tiver erros poderá então carregá-lo através do botão '—>' (upload) ou alternativamente através do menu da aplicação em 'File —> Upload'.
(NOTA: não é necessário verificar o código antes de carregar uma vez que o código é verificado antes, não sendo carregado caso contenha erros)

BLINK LED 1/2

~/ArduinoSynth_examples/Dia1_Efeitos/_01_ArduinoExamples.txt



- Saídas digitais
(digitalWrite)



Para este exemplo iremos usar as saídas digitais do Arduino para controlar um LED. Para tal é necessário ligar o LED conforme indicado na figura, com a perna mais pequena para o *ground (GND)* e a perna maior para o *pin 13*.

A maioria dos Arduino possui um pequeno LED embutido directamente ligado ao pin 13

BLINK LED 2/2

~/ArduinoSynth_examples/Dia1_Efeitos/_01_ArduinoExamples.txt

```
/*
Blink
Turns on an LED on for one second, then off for one second, repeatedly.

This sketch code is in the public domain.
*/
// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
    // initialize the digital pin as an output.
    pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
    digitalWrite(led, HIGH);    // turn the LED on (HIGH is the voltage level)
    delay(1000);               // wait for a second
    digitalWrite(led, LOW);     // turn the LED off by making the voltage LOW
    delay(1000);               // wait for a second
}
```

→ Declarações

→ Função *setup()*

→ Função *loop()*



A estrutura do código Arduino é definido por duas funções base: *setup()* e *loop()*.

A função ***setup()*** corre apenas uma vez quando a placa é ligada, enquanto que a função ***loop()*** é executada repetidamente.

As funções são representadas por dois parêntesis a seguir ao seu nome. Exemplos de outras funções:

pinMode() : define o comportamento, i.e. se actua como entrada (***INPUT***) ou saída (***OUTPUT***), de um determinado pin;

digitalWrite() : envia um determinado valor para um determinado pin;

delay() : interrompe a execução do código durante um determinado período de tempo (em milissegundos).

→ noção de blocos de código {}

→ Pedir aos participantes par ir alterando o código, especialmente com o **argumento** da função *delay()*

???

E se quisermos controlar
a luminosidade do LED ???

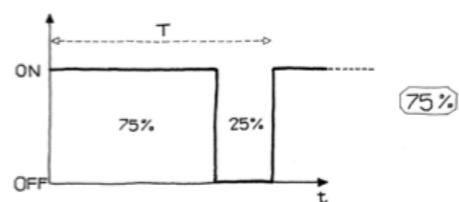
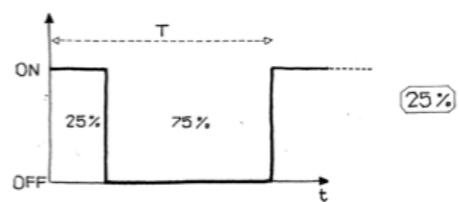
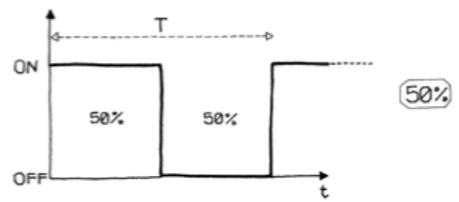


Para controlar a intensidade do LED seria necessário usar uma saída analógica, isto é, uma saída capaz de nos dar valores intermédios entre LIGADO (HIGH) e DESLIGADO (LOW)

MAS NÃO HÁ SAÍDAS ANALÓGICAS NO ARDUINO ??!!??!

Saídas digitais e PWM

~/Arduino/Examples/01.Basics/Fade



retirado de "Getting Started with Arduino"

- Pulse Width Modulation
- `analogWrite()` vs. `digitalWrite()`
- Saídas digitais com emulação analógica (~ ou PWM)

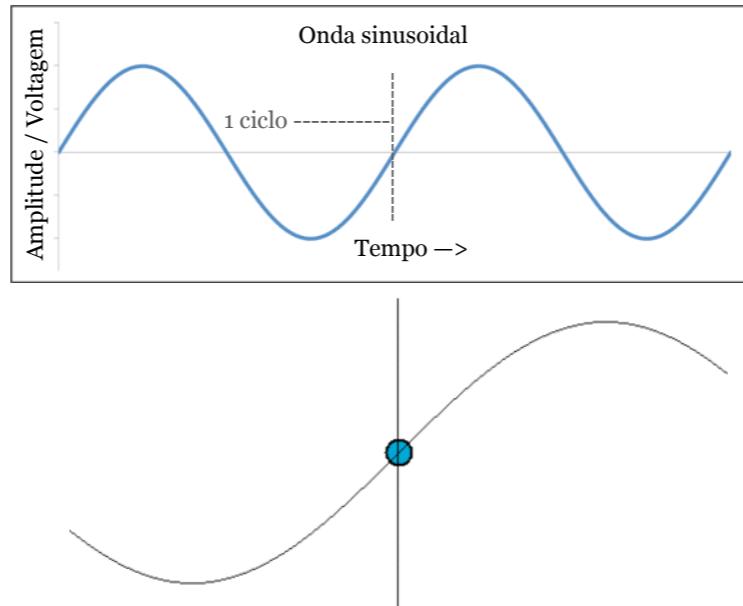


PWM significa *Pulse Width Modulation* (modulação por largura de pulso) e representa uma técnica capaz de obter resultados semelhantes a um sinal analógico através de meios digitais, criando uma onda quadrada - um sinal que alterna entre ligado e desligado (*on/off*). Este padrão *on/off* pode simular voltagens "contínuas" entre 0 Volts e 5 Volts ao alterar a porção de tempo em que o sinal está ligado versus a porção de tempo em que o sinal está desligado (ver imagem). Sendo a duração de tempo em que o sinal está ligado denominado de *pulse width* (largura de pulso). Para obter valores analógicos (entre 0 e 5V) pode-se modular a largura de pulso. Neste exemplo iremos usar esta técnica, implícita na função `analogWrite`, para alterar o brilho de um LED em vez de apenas o manter aceso ou apagado conforme demonstrado nos exemplos anteriores.

Note que nem todos os pins digitais do Arduino permitem utilizar esta técnica, sendo que normalmente têm uma indicação (`pwm` ou `~`) para informar que estes pins suportam PWM.

Propagação Sonora

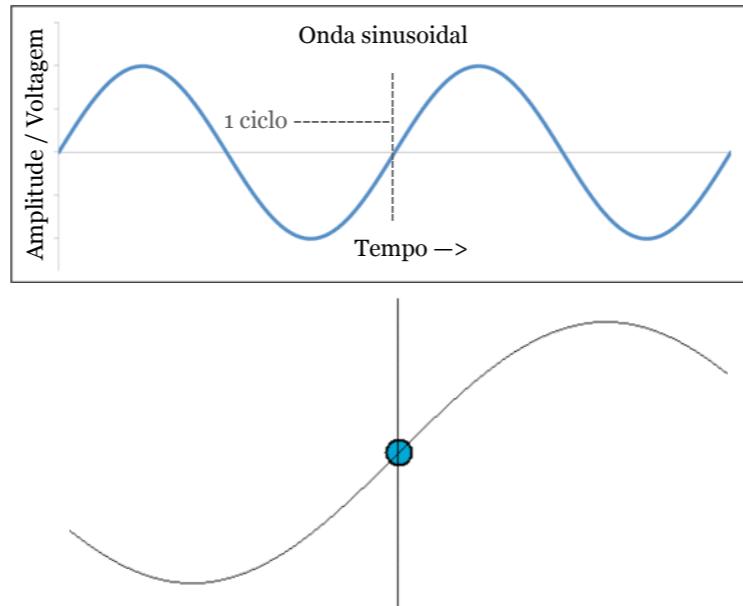
<https://musiclab.chromeexperiments.com/Sound-Waves>



Do mesmo modo que alterámos a intensidade do LED podemos também gerar um sinal sonoro

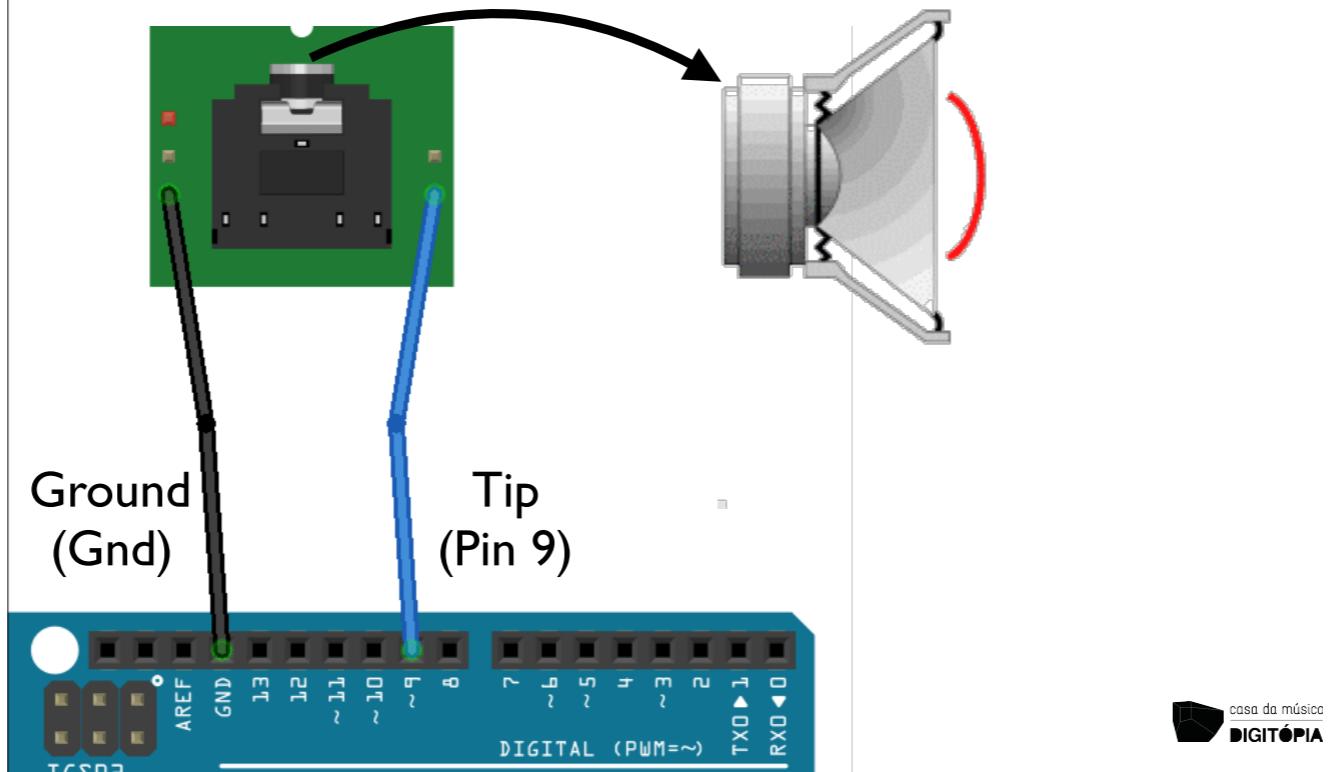
Propagação Sonora

<https://musiclab.chromeexperiments.com/Sound-Waves>



Do mesmo modo que alterámos a intensidade do LED podemos também gerar um sinal sonoro

Saída Áudio (simples)



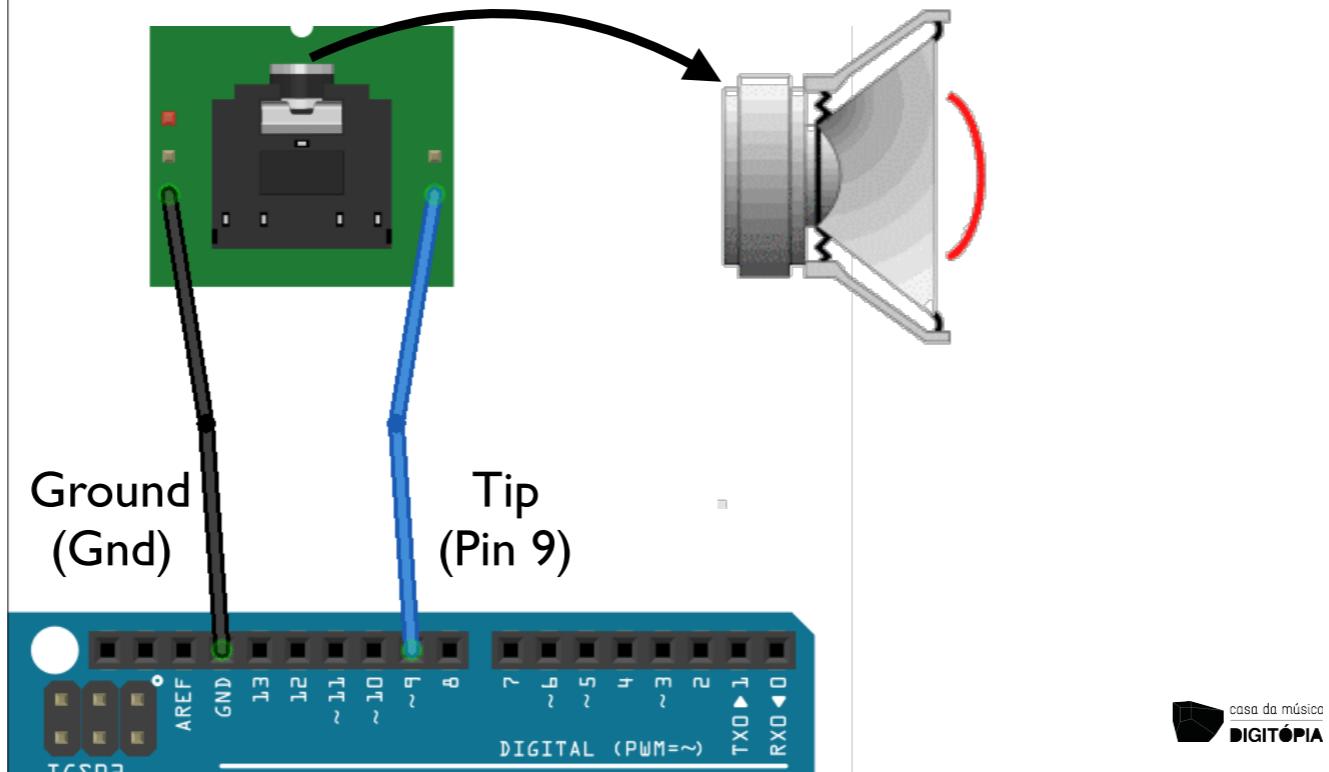
A forma mais simples de criar uma saída áudio no Arduino é a seguinte:

na maioria dos Arduinos usa-se o Pin 9

no Arduino MEGA usa-se o Pin 11

(consultar ~/Arduino/libraries/README.md)

Saída Áudio (simples)



A forma mais simples de criar uma saída áudio no Arduino é a seguinte:

na maioria dos Arduinos usa-se o Pin 9

no Arduino MEGA usa-se o Pin 11

(consultar ~/Arduino/libraries/README.md)



Mozzi

~/ArduinoSynth_examples/Dial_Efeitos/_02_OndaSinusoidal.ino

```
#include <MozziGuts.h>
#include <Oscil.h> // oscillator template
#include <tables/sin2048_int8.h> // sine table for oscillator
Oscil <SIN2048_NUM_CELLS, AUDIO_RATE> aSin(SIN2048_DATA);
#define CONTROL_RATE 64 // powers of 2 please

void setup(){
    startMozzi(CONTROL_RATE); // set a control rate of 64 (powers of 2 please)
    aSin.setFreq(440); // set the frequency
}

void updateControl(){}
    // put changing controls in here
}

int updateAudio(){
    return aSin.next(); // return an int signal centred around 0
}

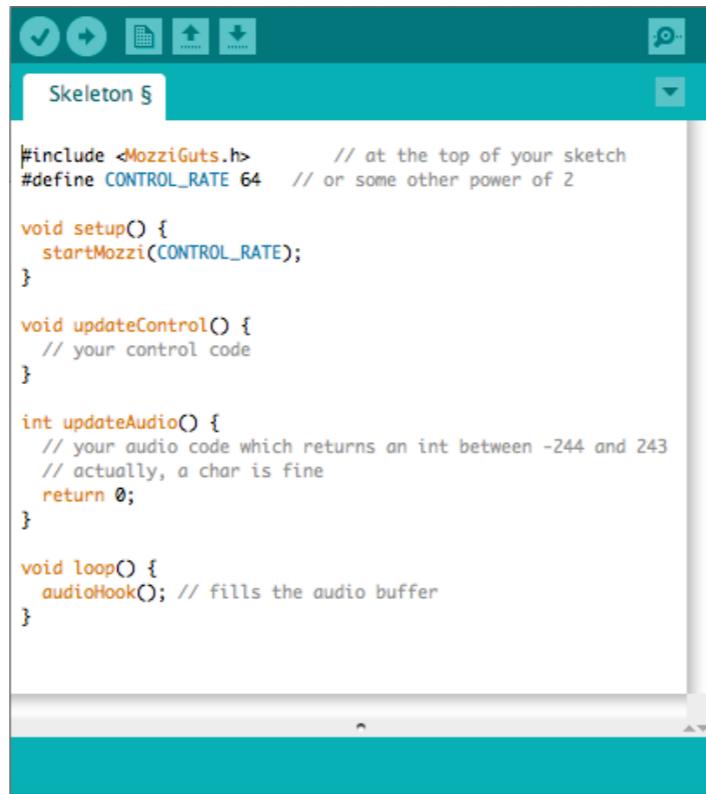
void loop(){
    audioHook(); // required here
}
```

- Instalar a biblioteca Mozzi (desenvolvida por Tim Barrass) em ~/Arduino/libraries/
- Abrir o exemplo do workshop
- Carregar para o Arduino



Arquitectura

~/Arduino/libraries/Mozzi/examples/01.Basics/Skeleton/Skeleton.ino



The screenshot shows the Arduino IDE interface with the title bar "Skeleton §". Below the title bar is a toolbar with icons for file operations like Open, Save, and Print. The main code editor area contains the following C++ code:

```
#include <MozziGuts.h>      // at the top of your sketch
#define CONTROL_RATE 64 // or some other power of 2

void setup() {
    startMozzi(CONTROL_RATE);
}

void updateControl() {
    // your control code
}

int updateAudio() {
    // your audio code which returns an int between -244 and 243
    // actually, a char is fine
    return 0;
}

void loop() {
    audiohook(); // fills the audio buffer
}
```

- include Mozzi
- startMozzi + Control Rate
- updateControl
- updateAudio
- audioHook

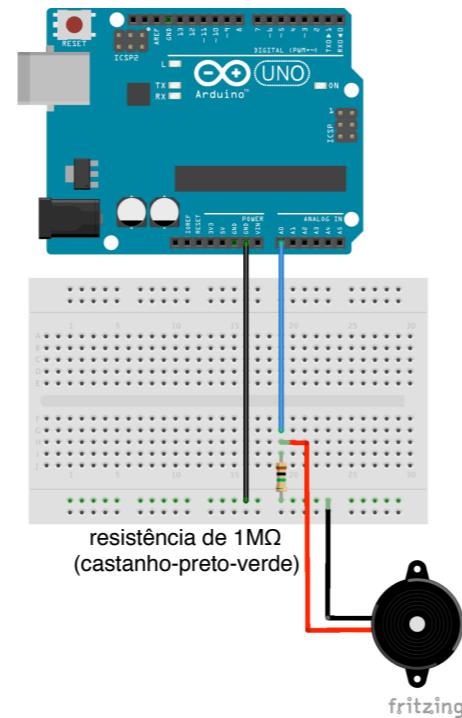


Arquitectura de um sketch Mozzi:

- 1) incluir a biblioteca **#include <MozziGuts.h>**
- 2) definir o **CONTROL_RATE** – quantas vezes por segundo é chamada a função **updateControl()**
- 3) **updateAudio**: onde colocamos o código para gerar áudio
- 4) **audioHook()** – preenche o buffer com o áudio que é enviado para o pin de saída

Entrada áudio (piezo)

~/ArduinoSynth_examples/Dia1_Efeitos/_03_EntradaPiezo/_03_EntradaPiezo.ino



- conexões

- editar *mozzi_config.h*

```
72 #define USE_AUDIO_INPUT true  
73 //#define USE_AUDIO_INPUT false  
83 #define AUDIO_INPUT_PIN 0
```

- função *getAudioInput()*

```
int asig = getAudioInput();
```



- 1) criar conexões (a resistência serve para atenuar a voltagem do piezo (tem que estar abaixo dos 5V))
- 2) editar o ficheiro ~Arduino/libraries/Mozzi/mozzi_config.h
- 3) usa-se a função **getAudioInput()**

-> abrir sketch e explicar conversão de 10-bits (0~1023) para saída STANDARD – 244~243

Feedback Delay

~/ArduinoSynth_examples/Dial_Efeitos/_04_FeedbackDelay/_04_FeedbackDelay.ino

The screenshot shows the Arduino IDE with the file `_04_FeedbackDelay.ino` open. The code is as follows:

```
/*
 * Adicionar feedback com delay aos sons captados pelo piezo
 */

#include <MozziGuts.h>
#include <mozzi_fixmath.h> // needed for other variable types used by AudioDelayFeedback

#include <AudioDelayFeedback.h>
AudioDelayFeedback <128> meuDelay;

void setup(){
    startMozzi();
    meuDelay.setFeedbackLevel(-124); // maximo = -128, minimo = 127
}

void updateControl(){}

int updateAudio(){
    int osig = getAudioInput() / (1024/488) - 244; // convertin from 0~1023 to -244~243
    return meuDelay.next(osig); // mistura o som do piezo com o delay
}

void loop(){
    audioHook();
}
```

Annotations (from left to right):

- `#include <AudioDelayFeedback.h>` → *Incluir Classe*
- `AudioDelayFeedback <128> meuDelay;` → *Declarar Classe*
- `meuDely.setFeedbackLevel(-124);` → *Definir ganho*
- `return meuDelay.next(osig);` → *Ler sinal gravado na memória do delay*

At the top right, there is a block diagram of a feedback delay circuit:

```
graph LR
    Input((Input)) --> Delay[Delay]
    Delay --> Output((Output))
    Feedback((Feedback path)) --> Delay
```

A red circle with the letters "SOS" is placed next to the output node.

At the bottom right, there is a logo for "casa da música DIGITÓPIA".

- 1) na biblioteca Mozzi cada classe está basicamente num ficheiro dedicado de modo a ocupar apenas memória estritamente necessária
- 2) Declara-se o objecto **meuDely** que é do tipo **AudioDelayFeedback** e tem 128 samples ($128/16384 = 7.8$ ms)
- 3) Existem dois parâmetros num Feedback Delay: 1) tamanho do delay e 2) nível de feedback, isto é, a quantidade de sinal que é realimentado
- 4) na função **updateAudio()** usa-se o método **.next()** para ler o próximo bloco de memória do delay

???

E se quisermos controlar
parâmetros do Delay ???

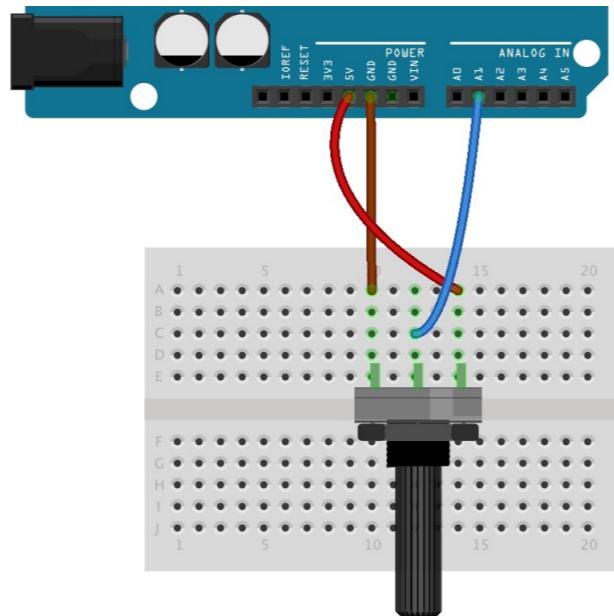


Para controlar a intensidade do LED seria necessário usar uma saída analógica, isto é, uma saída capaz de nos dar valores intermédios entre LIGADO (HIGH) e DESLIGADO (LOW)

MAS NÃO HÁ SAÍDAS ANALÓGICAS NO ARDUINO ?!?!?!

Entradas Analógicas

~/ArduinoSynth_examples/Dial_Efeitos/_05_EntradaPotenciometro/_05_EntradaPotenciometro.ino



- Entradas analógicas (`analogRead`)
- Biblioteca Serial (`Serial.println`)
- E se quisermos inverter os valores ?
- potenciómetros lineares vs. logarítmicos



- 1) as entradas analógicas do arduino são a 10-bits (0~1023) e é possível aceder às voltagens introduzidas através do método `analogRead()`
- 2) para comunicarmos entre o Arduino e o Computador usa-se a biblioteca **Serial**, assim podemos visualizar os dados através do **Serial Monitor** (botão com a loop no canto superior direito)
- 3) para inverter os valores do potenciômetro não é preciso alterar o código, basta trocar o cabo a castanho pelo cabo a vermelho
- 4) também existem potenciômetros logarítmicos que nos dão valores com uma determinada curva ao contrário dos lineares

Controlar o Feedback

~/ArduinoSynth_examples/Dial_Efeitos/_06_FeedbackDelayPot/_06_FeedbackDelayPot.ino

```
void updateControl(){
    feedback = (mozziAnalogRead(1) >> 2) - 128;
    meuDelay.setFeedbackLevel(feedback);
}

int updateAudio(){
    char asig = (getAudioInput() >>2) - 128; // converting from 0~1023 to -244~243
    return meuDelay.next(asig); // mistura o som do piezo com o delay
}
```

- analogRead vs. mozziAnalogRead
- bitshift (>> e <<) para multiplicações e divisões eficientes
- outros tipos de variáveis: char



- 1) em vez de usarmos o método **analogRead** do Arduino usamos o método **mozziAnalogRead** porque é mais eficiente

Controlar o tamanho do delay

~/ArduinoSynth_examples/Dial_Efeitos/_07_FeedbackDelayPotTime/_07_FeedbackDelayPotTime.ino

```
Q16n16 delaySize;
int feedback = -124;

void setup(){
    startMozzi(CONTROL_RATE);
    setupFastAnalogRead(FASTER_ADC); // use faster analogReads
    meuDelay.setFeedbackLevel(feedback);
}

void updateControl(){
    feedback = (mozziAnalogRead(FBKPIN) >> 2) -128;
    meuDelay.setFeedbackLevel(feedback);
    delaySize = Q8n0_to_Q16n16(mozziAnalogRead(DLYPIN)>>2);
}

int updateAudio(){
    char asig = (getAudioInput() >>2) - 128;
    return meuDelay.next(asig, delaySize);
}

void loop(){
    audioHook();
}
```

- delay size
- Q16n16 e outros
- Conversões
(Q8n0_to_Q16n16)
- Questões de eficiência
(dicas)



- 1) ligar um potenciômetro ao pin 2 (análogo)
- 2) e 3) ver a referência do Mozzi (~/Arduino/libraries/Mozzi/extras/doc/index.html) e (<http://wwwcplusplus.com/doc/tutorial/variables/>)
- 4) #define, setupFastAnalogRead,

Output Modes

~/ArduinoSynth_examples/Dial_Efeitos/_07_

```
26 //#define AUDIO_MODE STANDARD  
27 #define AUDIO_MODE STANDARD_PLUS  
28 //#define AUDIO_MODE HIFI
```

```
61 #define AUDIO_RATE 16384  
62 //#define AUDIO_RATE 32768
```

- Modos (STANDARD, STANDAR_PLUS e HIFI)
- AUDIO_RATE
- Bit Depth (resolução de amplitude)



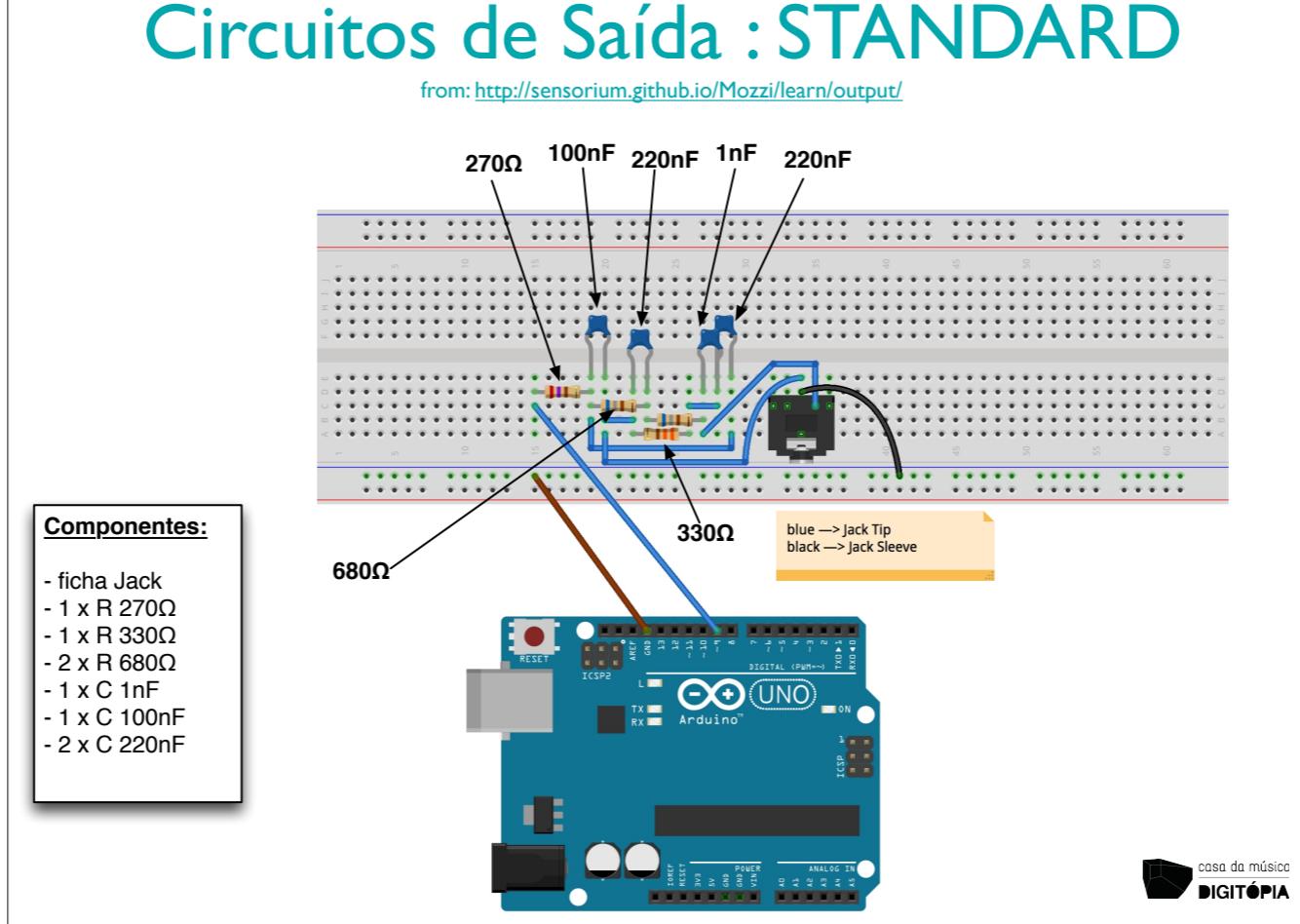
No modo STANDARD a resolução é de 488 (-244 a 243), ou seja quase 9-bits

which provides some headroom above the 8 bit table resolution currently used by the oscillators. You can look at utility/TimerOne library for more info about how interrupt rate and pwm resolution relate.

HIFI audio mode (14-bit) é 16384 (-8192 a 8191)

Circuitos de Saída : STANDARD

from: <http://sensorium.github.io/Mozzi/learn/output/>

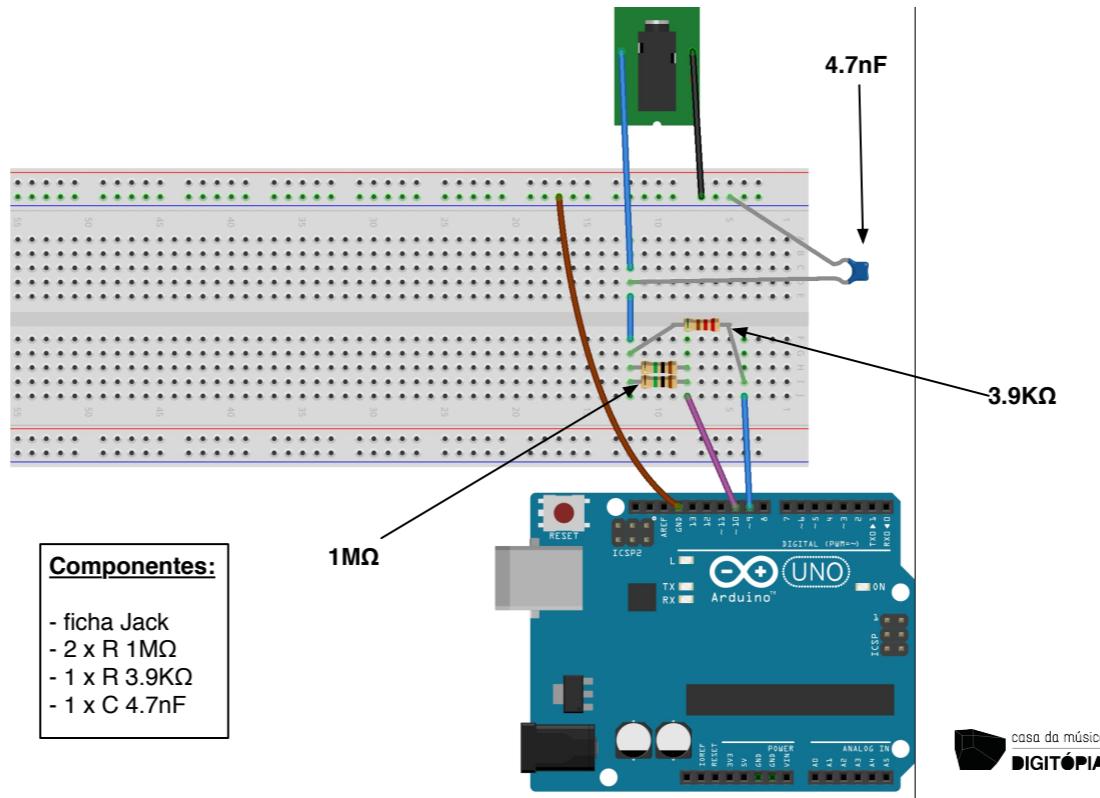


indicado para os processadores ATmega168

from: <http://sensorium.github.io/Mozzi/learn/output/>

Circuitos de Saída : HIFI

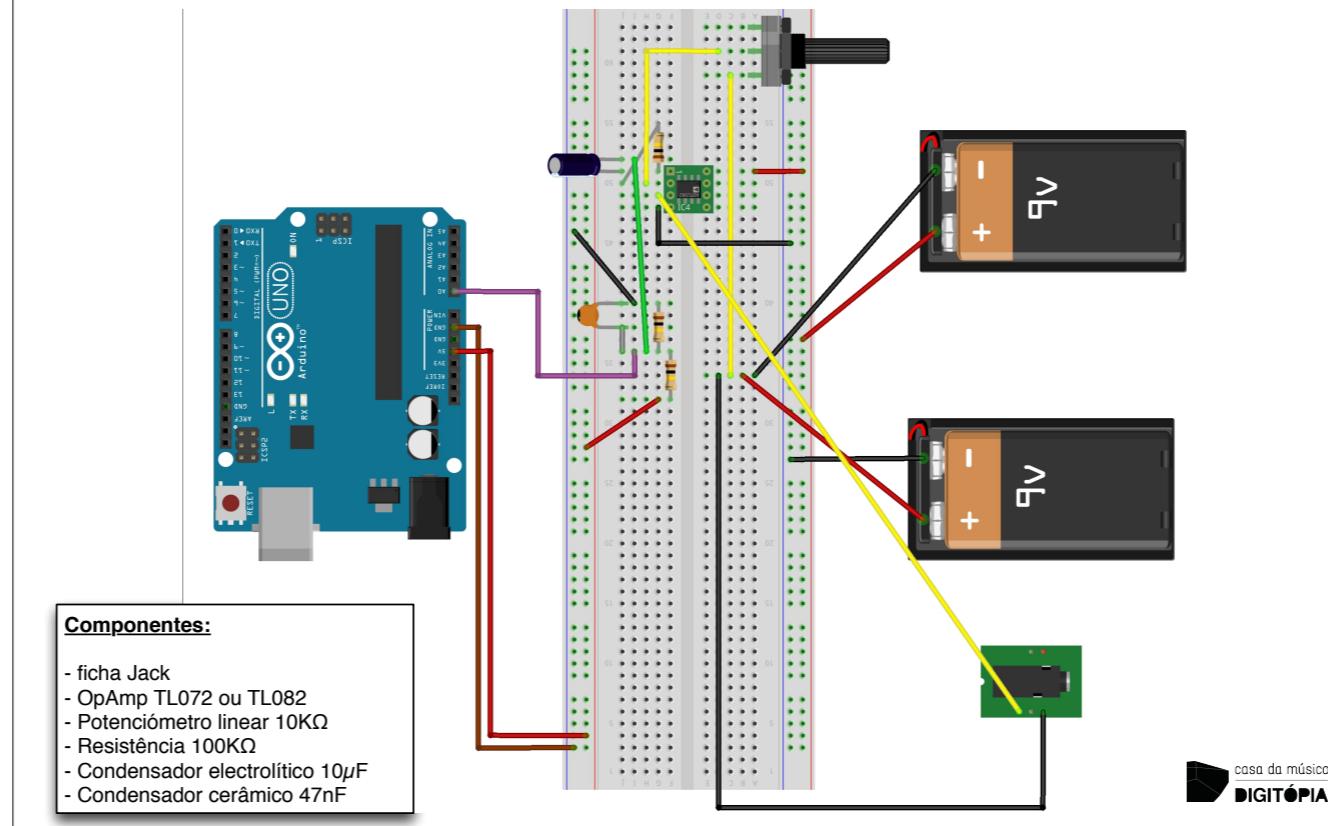
from: <http://sensorium.github.io/Mozzi/learn/output/>



- > escolher as resistências de $1\text{M}\Omega$, de modo a que quando ligadas em paralelo obtenham uma resistência o mais aproximada possível dos $499\text{K}\Omega$
- > ATENÇÃO: o código de cores da resistência de 3.9K está errado!

Circuitos de Entrada : Line In

from: <http://www.instructables.com/id/Arduino-Audio-Input/>



Outros Exemplos

- ~/ArduinoSynth_examples/Dial_Efeitos/_08_FeedbackDelayPotTime_HIFI/

Usar o piezo para controlo de parâmetros:

- ~/Arduino/Mozzi/examples/03.Sensors/Piezo_Frequency
- ~/Arduino/Mozzi/examples/03.Sensors/Piezo_SampleScrubber
- ~/Arduino/Mozzi/examples/03.Sensors/Piezo_SampleTrigger
- ~/Arduino/Mozzi/examples/03.Sensors/Piezo_Switch_Pitch



Leitura recomendada

- Electricity basics: <https://learn.sparkfun.com/tutorials/voltage-current-resistance-and-ohms-law>
- Getting Started in Electronics (Forrest Mims) :
<https://docs.google.com/file/d/0B5jcnBPSPWQyaTU1OW5NbVJQNW8/edit>
- Getting Started with Arduino (Massimo Banzi) :
<http://it-ebooks.info/book/1338/>
- (Fórum Arduino :<http://forum.arduino.cc/>)
- Áudio Digital e MIDI (J.A.Gomes-Digitópia): https://github.com/Digitopia/Workshops-Crashcourses/releases/download/ws_11/AUDIO.DIGITAL.MIDI.pdf.zip



Arduino Synth

Sintetizadores DIY



Plano (Domingo)

Manhã

Play a note (botões, envolventes, mtotf)

MIDI to Arduino (Hairless-MIDI Serial and MIDI lib setup)

Tarde

Synthesis Overview - A matter of taste

Synthesis (1/4) - Abstract Synthesis (AM, RM, FM + outras formas de onda e tabelas)

Synthesis (2/4) - Espectro (síntese subtractiva: filtros)

Synthesis (3/4) - Gravações processadas (Processed Recordings)

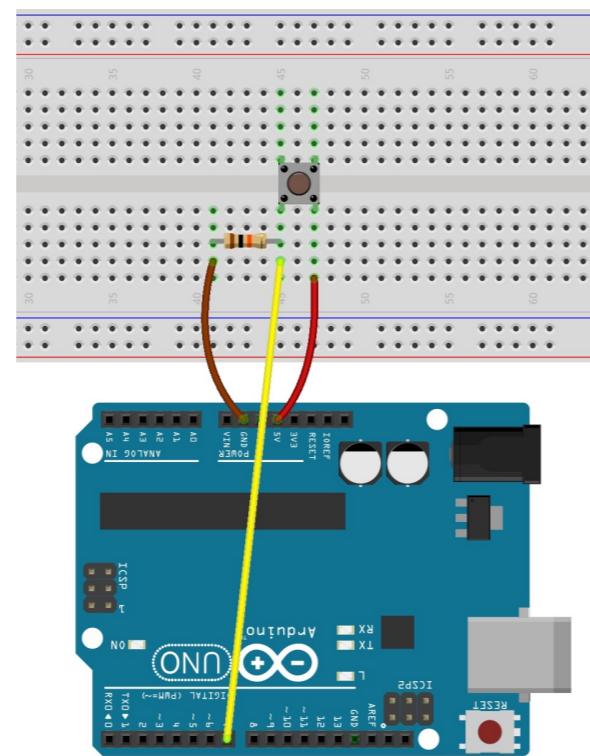
Synthesis (4/4) - Modelação Física (Physical Modeling)

Jam/Questions

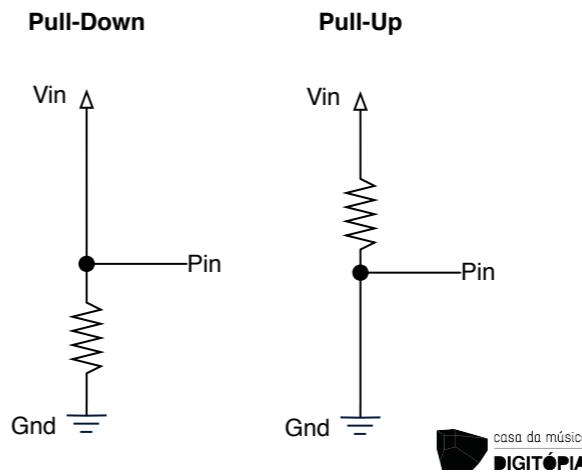


Entradas Digitais

~/ArduinoSynth_examples/Dia2_Synths/_01_Botao



- Entradas digitais
(pinMode, digitalRead)
- (pull-down & pull-up)

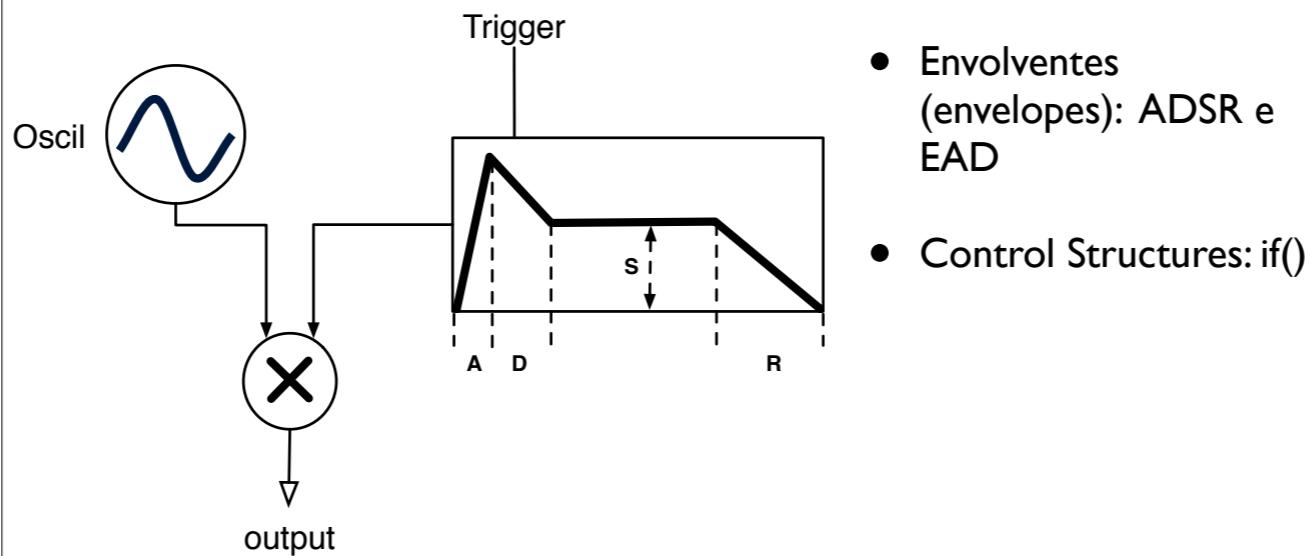


A resistência “obriga” o fluxo de corrente (electrões) a seguir a conexão com menos resistência

Se não tivermos a resistência temos um curto-círcuito!!!

Play a note

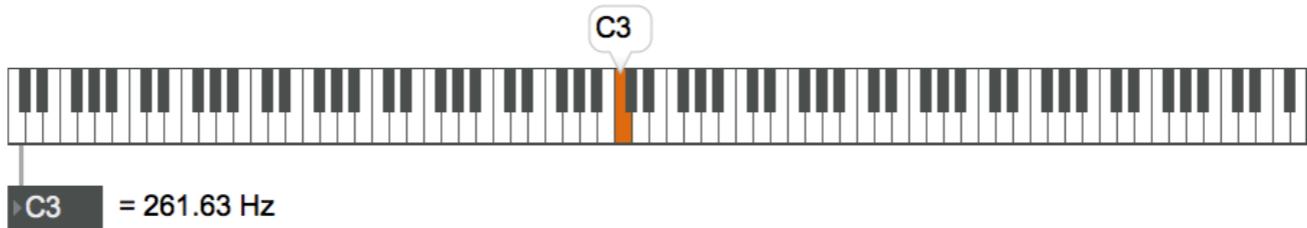
~/ArduinoSynth_examples/Dia2_Synths/_02_PlayNote



- 1) envolventes : ADSR – Attack, Decay, Sustain, Release (A, D e R são temporais, S é ganho)
- 2) ver ref. Arduino em “Control Structures” <http://www.arduino.cc/en/Reference/HomePage>

Pitch (tuned frequencies)

~/ArduinoSynth_examples/Dia2_Synths/_03_PlayPitchedNote



- Função mtos()



mtos significa MIDI to frequency e transpõe uma Nota midi (C3 = 60) para frequência (Hz)

Enviar MIDI p/ Arduino

INSTALAÇÃO

- Instalar hairless-midiserial :

<http://projectgus.github.io/hairless-midiserial/>

- Instalar biblioteca “MIDI”

<http://playgroundarduino.cc/Main/MIDILibrary>

- Usar software capaz de enviar mensagens p/ dispositivos MIDI
- Windows: <http://www.midiox.com/> || Linux: <http://www.varal.org/ttymidi/> ???? (ver website hairless-midiserial)

Recomendações: software com o qual estejam habituados ou Ableton Live (usado no workshop) ou VMPK (<http://vmpk.sourceforge.net>)



VMPK – virtual midi piano keyboard, é bastante básico mas pelo menos dá para testar caso não tenham outro software

MIDI (short intro)

https://github.com/Digitopia/Workshops-Crashcourses/releases/download/ws_11/AUDIO.DIGITAL.MIDI.pdf.zip

- MIDI - Musical Instrument Digital Interface
- Tipos de mensagens: MIDI Channel, Note-On, Note-Off, Velocity, Control Change (cc), etc...
- Velocity: é a intensidade com que se pressiona uma tecla (0~127)
- Note-On: pressionar uma tecla devolve o número relativo à nota* da tecla e o valor de intensidade (Velocity)
- Note-Off: soltar a tecla devolve o número relativo à tecla e o valor zero (Velocity = 0 significa uma mensagem Note-Off)

*(notas entre 0 e 128 (7-bits), ou seja de C-2 a G5 para C3=60)



MIDI é um protocolo de comunicação (está errado quando se fala em “som MIDI...”)

Pontecialidades

- Integração com hardware/software MIDI
- Podem não ser necessário tantos componentes (botões, potenciômetros, etc...) para controlar o nosso sintetizador Arduino sendo que podemos enviar notas MIDI e/ou mensagens Control Change (para controlar parâmetros), ou outras (ModWheel, BendWheel, AfterTouch, etc...)



Limitações da utilização MIDI no Arduino através do hairless-midiserial

Limitações

- Tem que se desactivar a comunicação com o *hairless-midiserial* para conseguir carregar código para o Arduino
- MIDI Standard BaudRate = 38400
- O *hairless-midiserial* não suporta este BaudRate (por norma usa-se 115200), tendo que se alterar o ficheiro midi_Settings.h

```
68     static const long BaudRate = 115200;  
69 //static const long BaudRate = 38400;
```

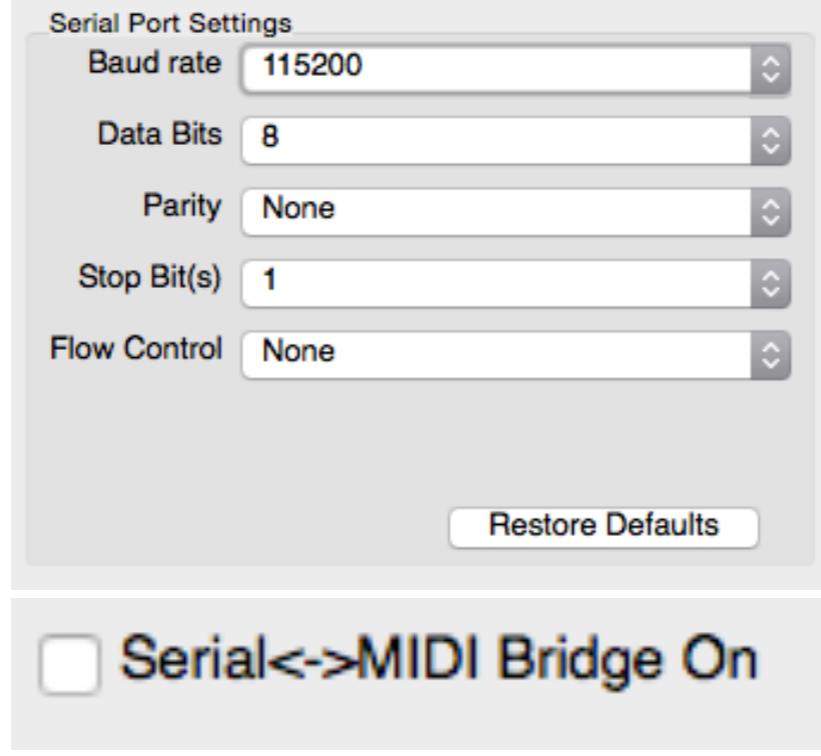
- Para enviar MIDI por USB para o Arduino através (por ex.) de um teclado MIDI tem que se usar o BaudRate de 38400 mas é necessário alimentar o Arduino através da ficha jack DC
- Pode-se criar um circuito para enviar MIDI para o Arduino através de uma ficha DIN (5pinos/180°)



Limitações da utilização MIDI no Arduino através do hairless-midiserial

Setup 1/2

```
68     static const long BaudRate = 115200;  
69 //static const long BaudRate = 38400;
```



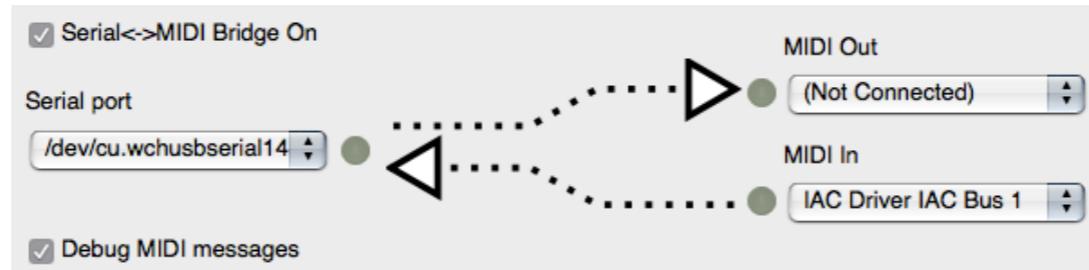
- Alterar ficheiro “midi_Settings.h” da biblioteca MIDI
- Definir preferências no *hairless-midiserial* para ter um BaudRate correspondente
- Desactivar a comunicação no *hairless-midiserial* para poder carregar código p/ o Arduino



Testar envio de MIDI para o Arduino através do middleware hairless-midiserial

Setup 2/2

~/WS_ArduinoSynth_2016/SKETCHES/Dia2_Synths/_04_Mozzi_MIDI_Input



- Carregar o sketch nº 4 (HIFI ou STANDARD)
- Abrir 1º o Hairless e só depois o Live (ou outro software)
- Configurar o software c/ o mesmo output MIDI que o MIDI In no Hairless (neste caso, em OSX, usamos o IAC Driver)
- press play and pray :)



Testar envio de MIDI para o Arduino através do middleware hairless-midiserial

Arduino Synth

note-off : time for a break...



Arduino Synth

Synthesis Overview - a matter of taste



Modelos (taxonomia)		Implementação	Controlo	Semântica (exemplos)	Modelos (exemplos)
<i>Loose Modeling</i>	Abstractos	acessível	dificultado	<i>amplitude, modulation depth, phase, frequency, ...</i>	AM, FM, RM, Waveshaping, Phase Distortion, ...
	Gravações processadas	acessível	dificultado	<i>samples, position, speed rate, ...</i>	Granular, Wavelet Transform, Wavetable, ...
	Espectrais	média	médio	<i>frequency bins, magnitude, formant, ...</i>	Aditiva, Subtrativa, Phase Vocoder, LPC, Inverse FFT, ...
	Físicos	difícil	facilitado	<i>dampen, strike position, rotation, ...</i>	Waveguide, Karplus-Strong extensions, Ruiz Strings, Modal, ...

Taxonomia e comparação de modelos de síntese sonora (Smith 1991) (Tolonen et al. 1998) (Miranda 2000 e 2002)

retirado da tese de mestrado (Tiago Ângelo, 2012):

<https://repositorio-aberto.up.pt/bitstream/10216/65229/2/55974.pdf>

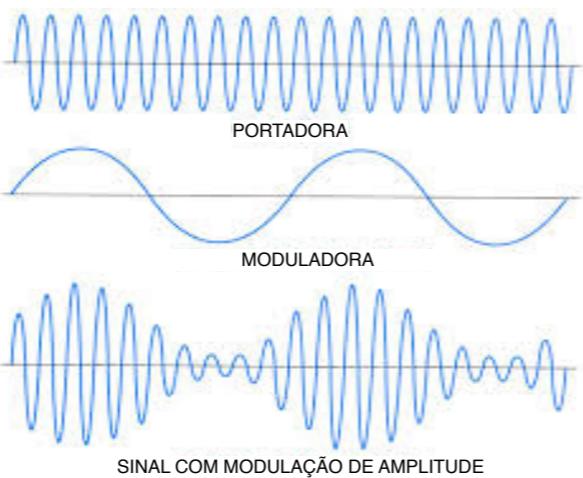
Referências:

Computer Music Tutorial (Curtis Roads)

Theory and Technique of Electronic Music (<http://msp.ucsd.edu/techniques.htm>)

Abstract Models :AM/RM

~/ArduinoSynth_examples/Dia2_Synths/_05_ModulacaoAmplitude



- Usa um sinal modulador para alterar a amplitude de um sinal portador
- Tremolo (freq. moduladora < 16Hz)
- AM - moduladora unipolar
- RM - moduladora bipolar

DICA:

Modulation Depth = amplitude da moduladora



AM – Amplitude Modulation

RM – Ring Modulation

NOTA: estes modelos podem ser usados em conjunto com os exemplos de sábado!

Abstract Models :AM/RM

~/ArduinoSynth_examples/Dia2_Synths/_05_ModulacaoAmplitude

- Usa um sinal modulador para alterar a amplitude de um sinal portador
- Tremolo (freq. moduladora < 16Hz)
- AM - moduladora unipolar
- RM - moduladora bipolar

DICA:

Modulation Depth = amplitude da moduladora



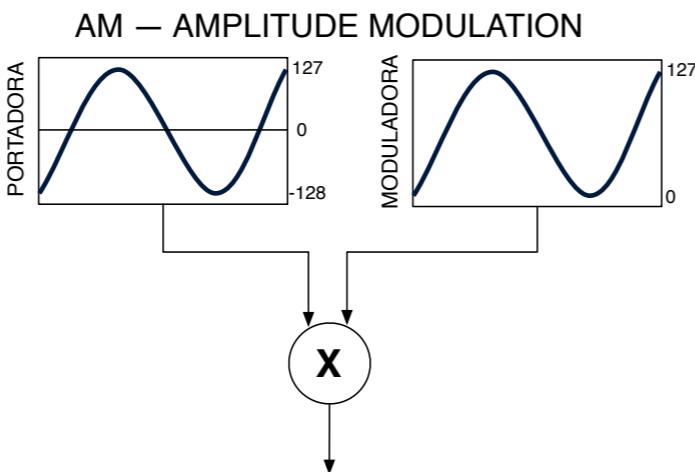
AM – Amplitude Modulation

RM – Ring Modulation

NOTA: estes modelos podem ser usados em conjunto com os exemplos de sábado!

Abstract Models :AM/RM

~/ArduinoSynth_examples/Dia2_Synths/_05_ModulacaoAmplitude



DICA:

Modulation Depth = amplitude da moduladora

- Usa um sinal modulador para alterar a amplitude de um sinal portador
- Tremolo (freq. moduladora < 16Hz)
- AM - moduladora unipolar
- RM - moduladora bipolar



AM – Amplitude Modulation

RM – Ring Modulation

NOTA: estes modelos podem ser usados em conjunto com os exemplos de sábado!

Abstract Models :AM/RM

~/ArduinoSynth_examples/Dia2_Synths/_05_ModulacaoAmplitude

- Usa um sinal modulador para alterar a amplitude de um sinal portador
- Tremolo (freq. moduladora < 16Hz)
- AM - moduladora unipolar
- RM - moduladora bipolar

DICA:

Modulation Depth = amplitude da moduladora



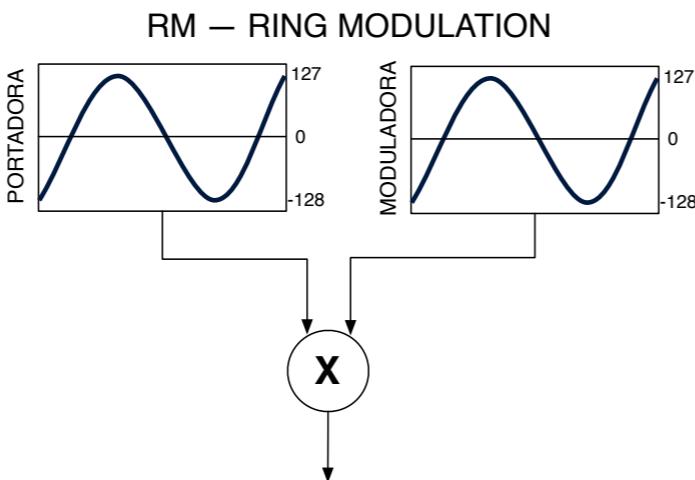
AM – Amplitude Modulation

RM – Ring Modulation

NOTA: estes modelos podem ser usados em conjunto com os exemplos de sábado!

Abstract Models :AM/RM

~/ArduinoSynth_examples/Dia2_Synths/_05_ModulacaoAmplitude



DICA:

Modulation Depth = amplitude da moduladora

- Usa um sinal modulador para alterar a amplitude de um sinal portador
- Tremolo (freq. moduladora < 16Hz)
- AM - moduladora unipolar
- RM - moduladora bipolar



AM – Amplitude Modulation

RM – Ring Modulation

NOTA: estes modelos podem ser usados em conjunto com os exemplos de sábado!

Abstract Models :AM/RM

~/ArduinoSynth_examples/Dia2_Synths/_05_ModulacaoAmplitude

- Usa um sinal modulador para alterar a amplitude de um sinal portador
- Tremolo (freq. moduladora < 16Hz)
- AM - moduladora unipolar
- RM - moduladora bipolar

DICA:

Modulation Depth = amplitude da moduladora



AM – Amplitude Modulation

RM – Ring Modulation

NOTA: estes modelos podem ser usados em conjunto com os exemplos de sábado!

Abstract Models :Additive

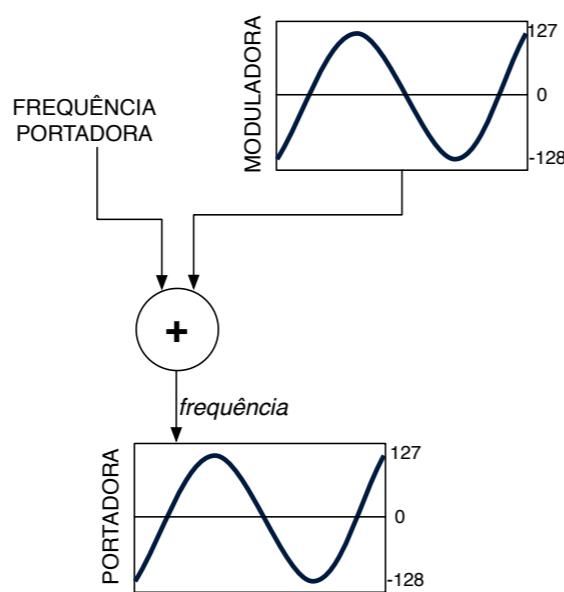
~/ArduinoSynth_examples/Dia2_Synths/_06_SinteseAditiva

- segundo Fourier é possível uma onda complexa pode ser reconstruída através da adição de várias ondas sinusoidas
- este tipo de síntese consiste basicamente na adição de vários osciladores
- o timbre é então determinado pela amplitude e frequência de cada oscilador, sendo que frequências próximas poderão criar batimentos, ou seja, as frequências interagem entre si criando depressões e aumentos de amplitude
- ver também exemplo Mozzi “Detuned_Beats_Wash.ino”



Abstract Models : FM

~/ArduinoSynth_examples/Dia2_Synths/_07_ModulacaoFrequencia



- Descoberta por John Chowning em 1967
- O seu ponto forte era a possibilidade de criar tons complexos (semelhante à síntese aditiva) com apenas dois osciladores



aqui a frequência portadora (carrier frequency) pode ser vista como uma “frequência central”

Tabelas e Osciladores

~/Documents/Arduino/Libraries/Mozzi/tables/

```
/*
  Basicamente um oscilador le uma tabela
  de valores que representam uma forma de onda
*/

// Incluimos a classe 'Oscil' para usar osciladores
#include <Oscil.h>
// Incluimos a tabela que queremos usar
#include <tables/sin2048_int8.h>

//Declaramos o oscilador
Oscil <2048_NUM_CELLS, AUDIO_RATE> oMeuOscilador(SIN2048_DATA);
//< Tamanho da tabela, Quantas x faz update> nomeDoOscilador(Dados para o oscilador)
```

- Formas de onda:
~/Documentos/Arduino/
libraries/Mozzi/tables/
- As tabelas têm uma
resolução de amplitude de
8-bits, variando entre -128
e 127



explicar onde está a referência do Mozzi : ~/Documents/Arduino/libraries/Mozzi/extras/doc/index.html

Subtractive Synthesis

- Also check Mozzi examples “10.Audio_Filters”
- LowPass.h
- StateVariable.h : LOWPASS, BANDPASS, HIGHPASS and NOTCH



E se criarmos vários osciladores com samples de curta duração ? —> Síntese Granular

Sampling

~/ArduinoSynth_examples/Dia2_Synths/_08_SimpleSampler



- Os samples são em tudo semelhantes às tabelas, mas por norma são mais longos (>NUM_CELLS)
- Estão em Documentos/Arduino/libraries/Mozzi/samples/
- Tal como os osciladores também é possível alterar a frequência!



E se criarmos vários osciladores com samples de curta duração ? —> Síntese Granular

Making Samples 4Mozzi

~/ArduinoSynth_examples/Dia2_Synths/_09_SimpleSampler



- Os samples são em tudo semelhantes às tabelas, mas por norma são mais longos (>NUM_CELLS)
- Estão em Documentos/Arduino/libraries/Mozzi/samples/
- Tal como os osciladores também é possível alterar a frequência!



E se criarmos vários osciladores com samples de curta duração ? —> Síntese Granular

Mozzi Caveats

<http://sensorium.github.io/Mozzi/> (Caveats and workarounds)

- ATENÇÃO!!! O Mozzi desactiva as seguintes funções do Arduino:
 - `delay()`, `delayMicroseconds()`, `millis()` e `micros()`
 - em substituição tem: `EventDelay()`, `Metronome()` e

Recording from our Synth:

In Audacity,

- set the Input to Built-in Input, 1 (Mono).
- in the Project Rate (Hz) box, type 16384. This is Mozzi's sample rate and will help show your waveforms clearly, otherwise they'll appear as scrambled, aliased Pulse Width Modulated square waves.



Arduino Synth

thank you
see you around

