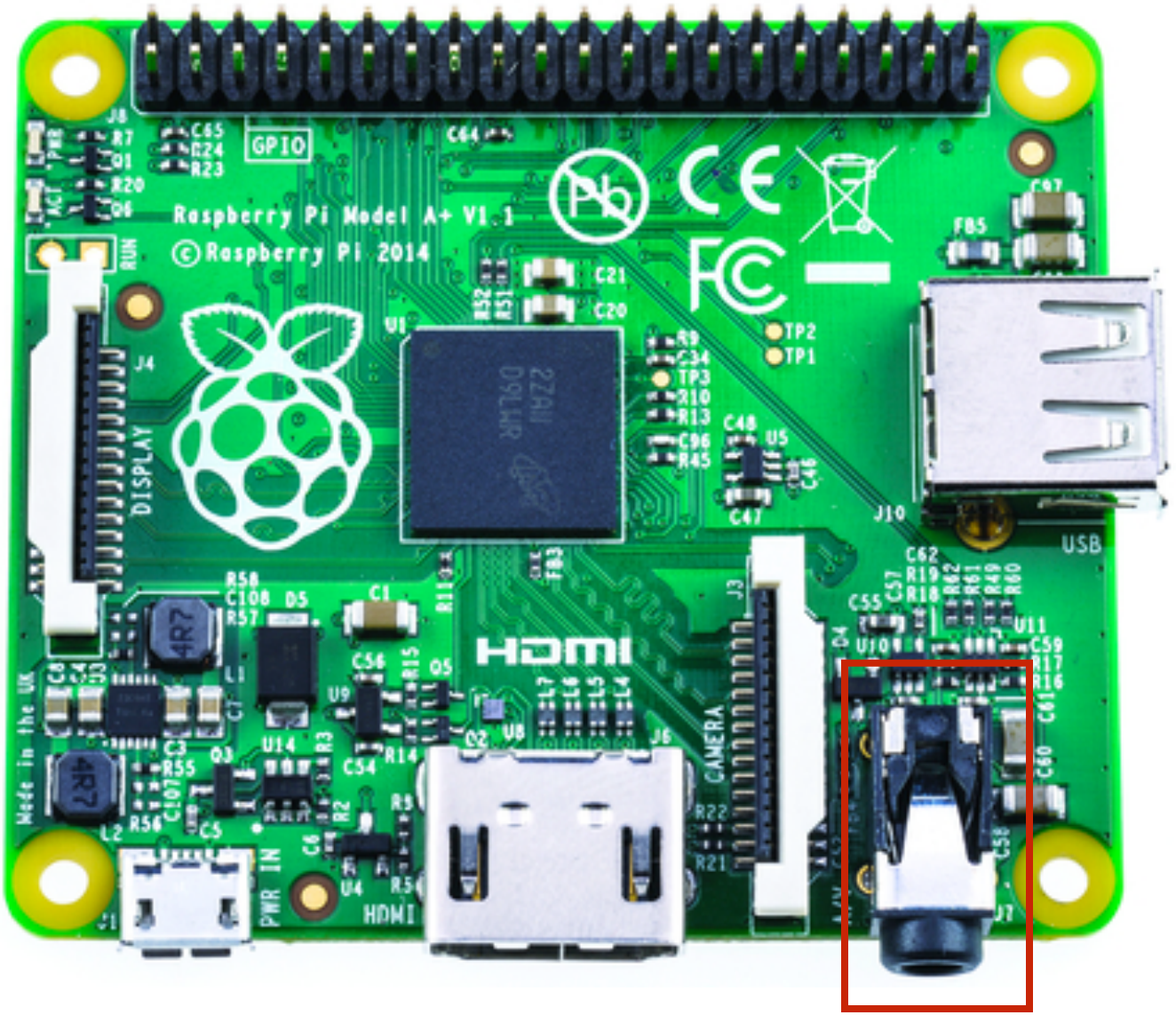
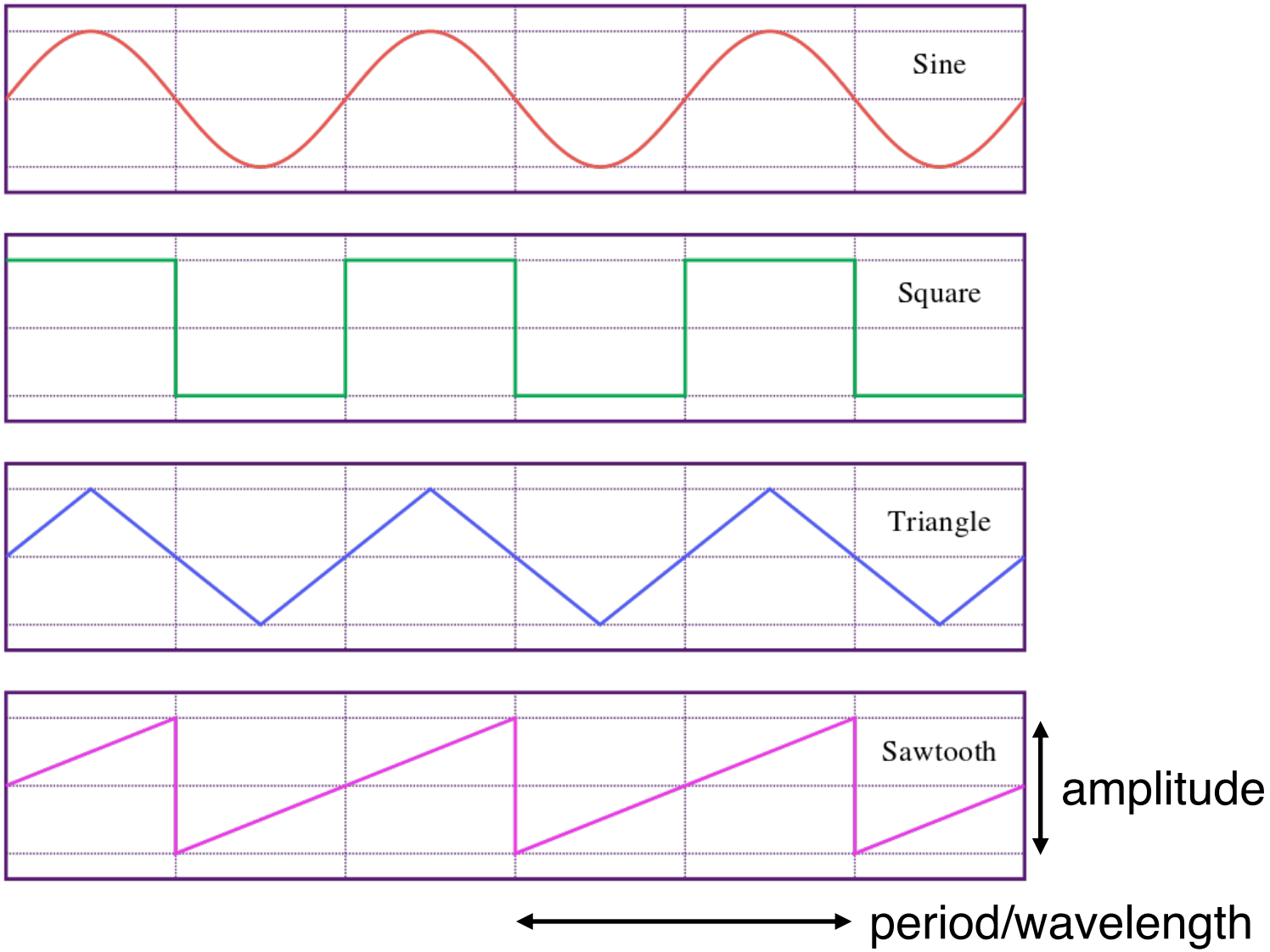
Sound and MIDI

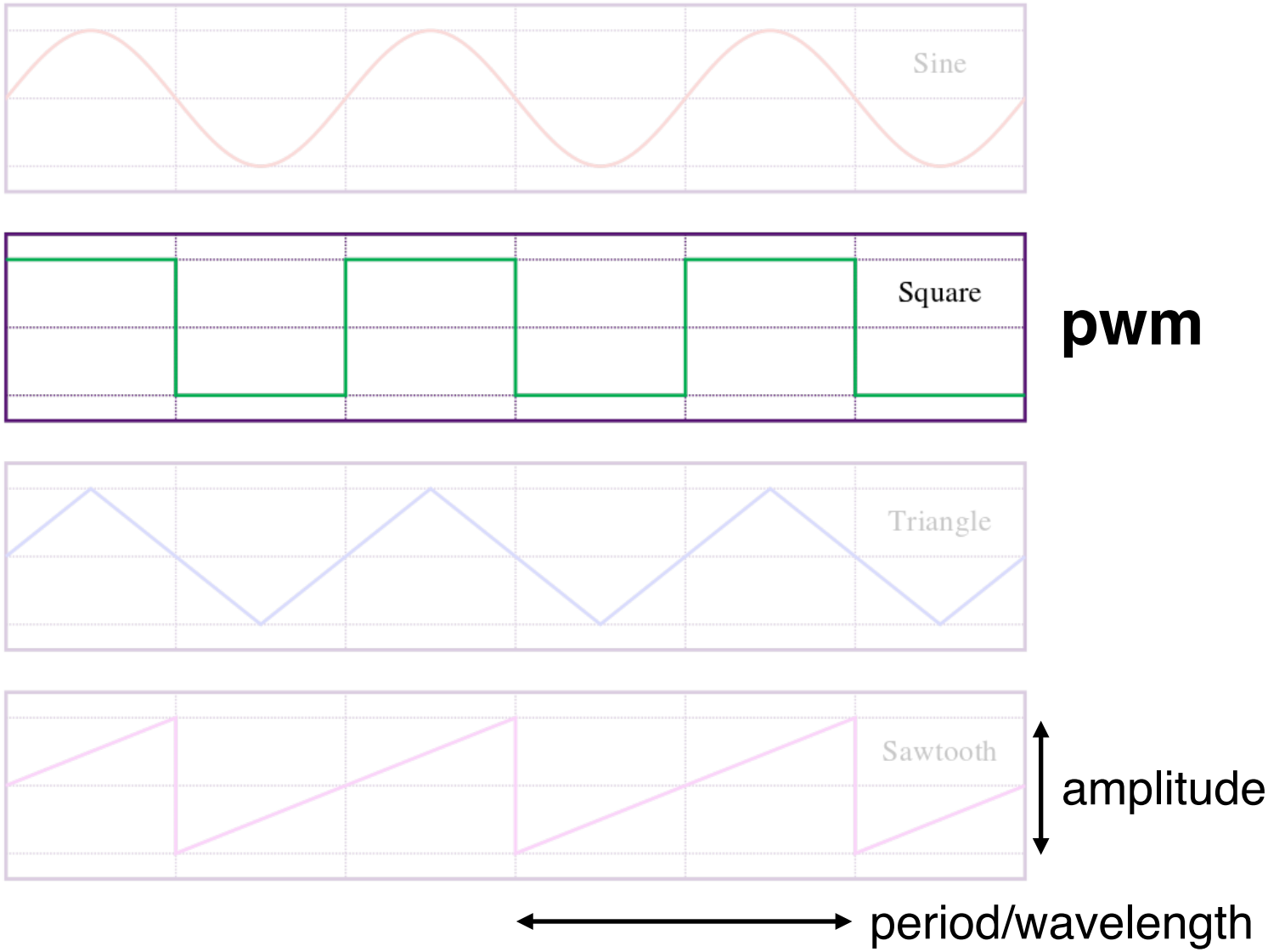
3.5mm Audio Jack



Waveforms


Sine
Square
Triangle
Sawtooth
amplitude
period/wavelength

Waveforms


Sine
Square
pwm
Triangle
Sawtooth
amplitude
period/wavelength

PWM Waveform Demo

square, sine, triangle, and saw

Digital != Analog

Our CPU is generating a square pulse wave

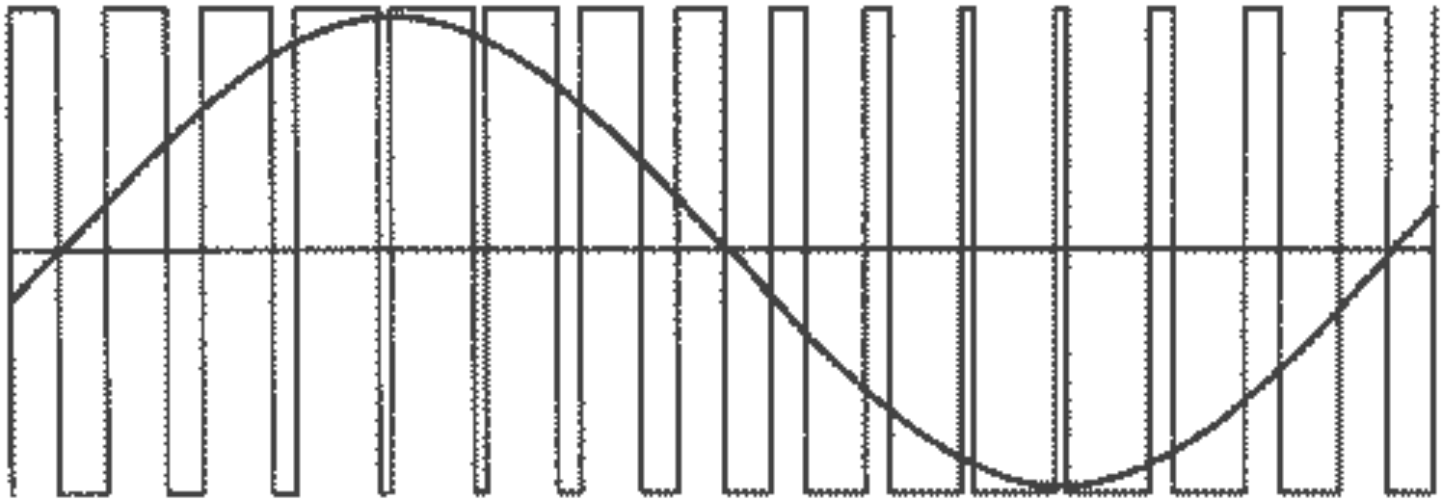
Interacts with electrical components: changing electric field, impedance, capacitance, etc.

- **Note: cannot actually send pulse wave**

These details are why building high-frequency circuits (e.g., radio, HDMI) requires very careful engineering

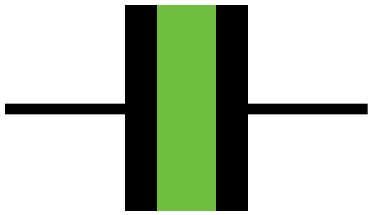
PWM to the Rescue!

Can simulate continuous values with fast enough PWM clocking: need hardware help

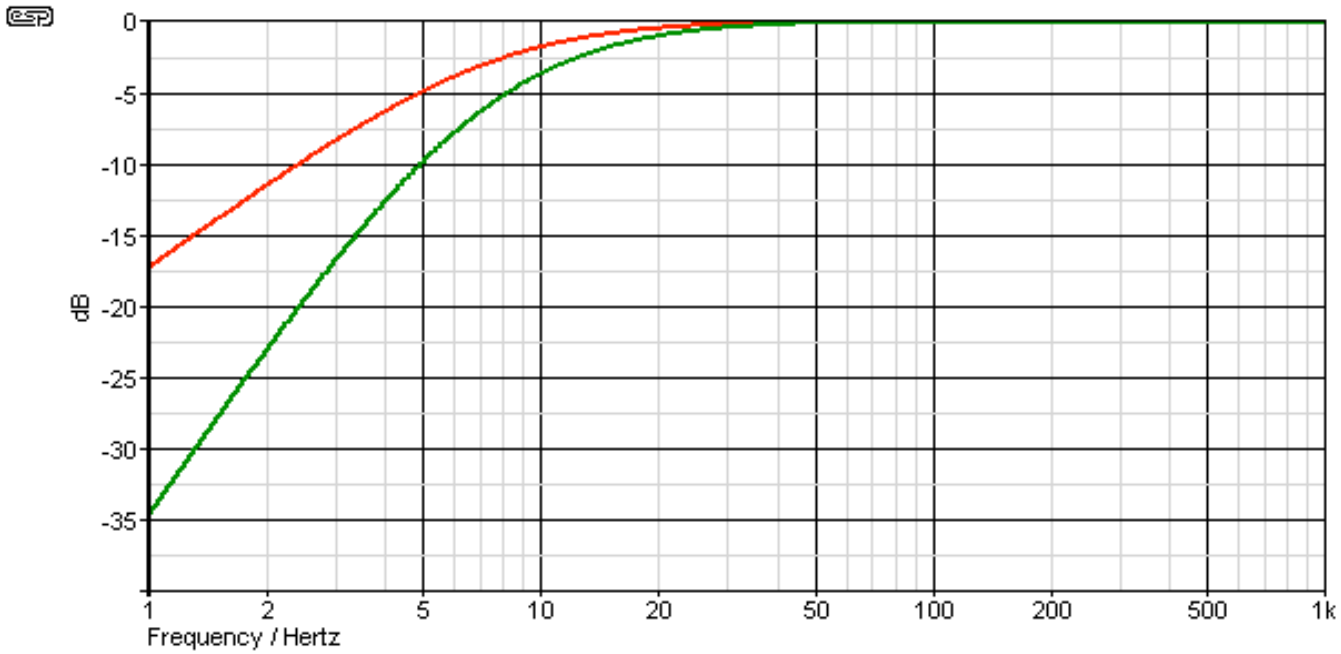


Capacitors

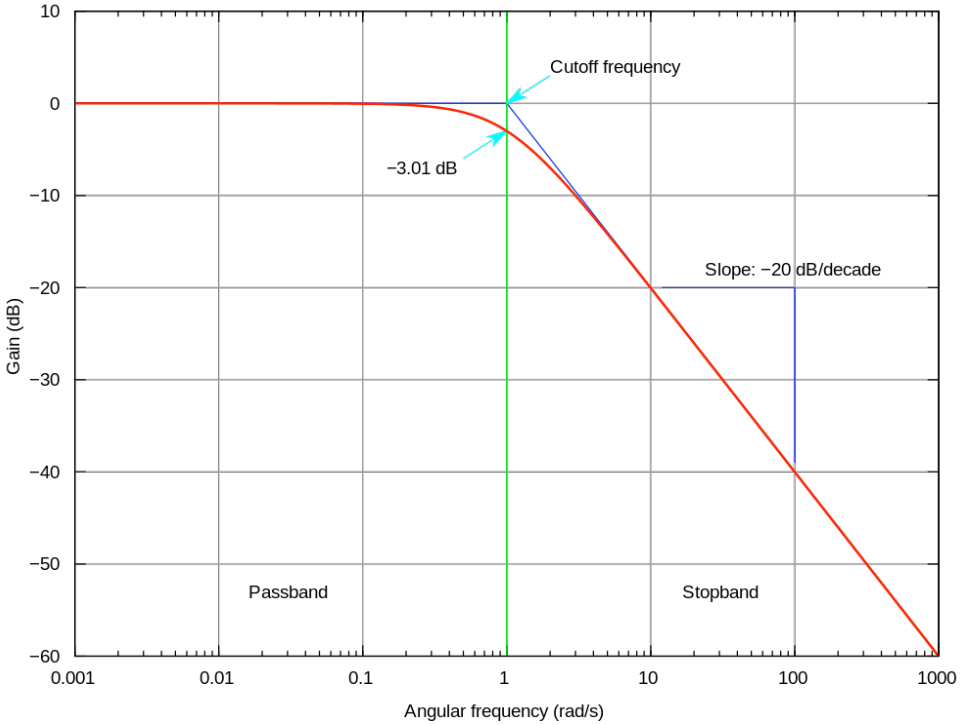
Two (tiny) plates separated by a non-conductive material (dielectric)



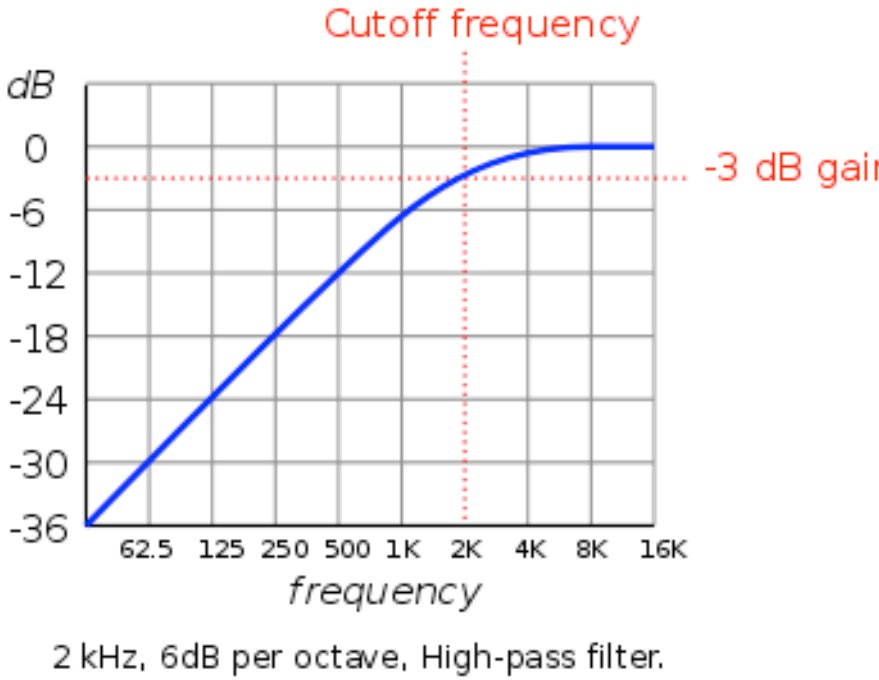
Frequency-selective impedance



Filters

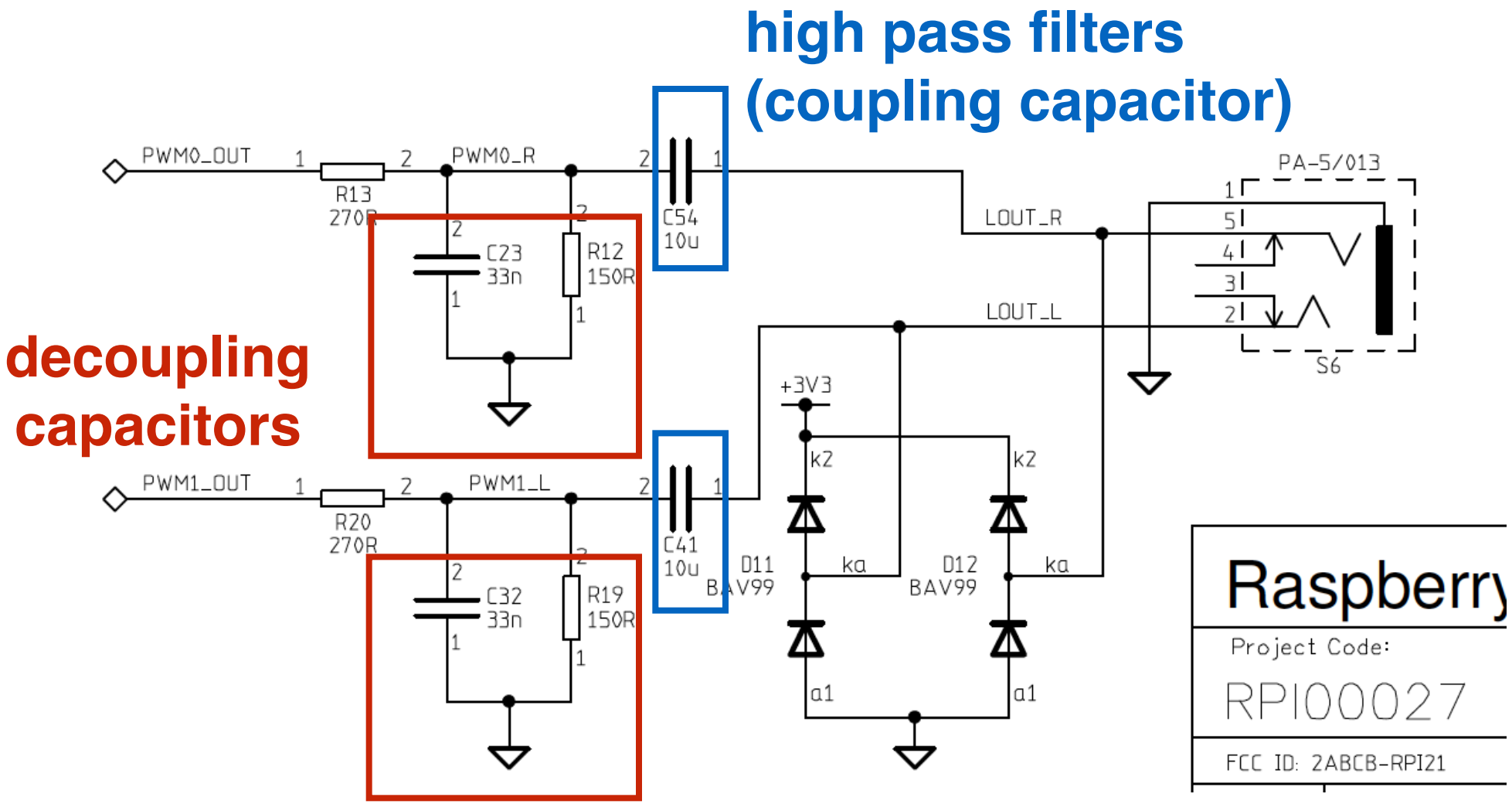


low pass



high pass

Raspberry Pi Audio Circuit



Hardware PWM Support

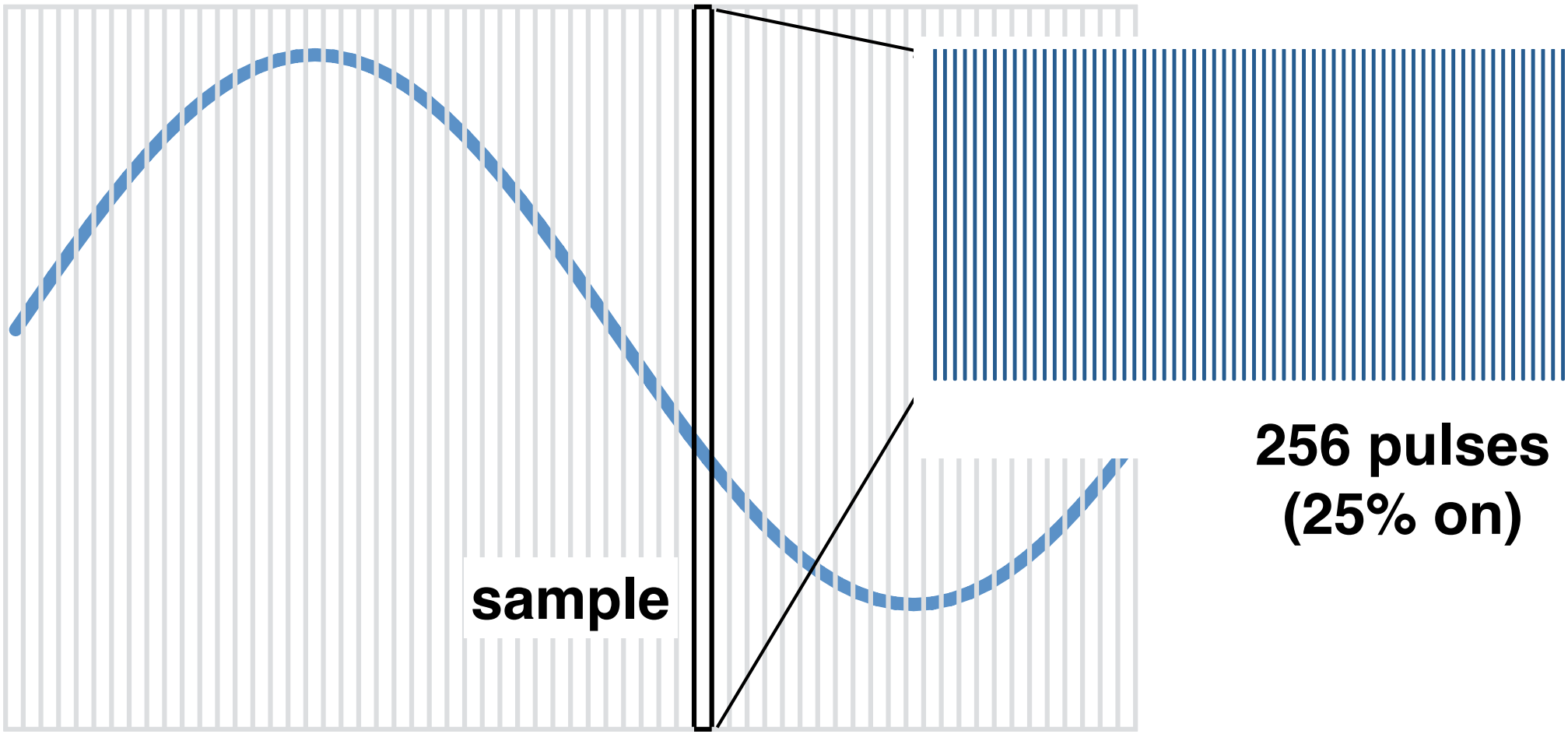
Start with a 19.2MHz clock, divide it to specify the time slots of on/off

E.g., divider of 2.375 = 8,192kHz

Divide wave into steps (e.g., 64)

Divide each step into train of (e.g., 256) pulses: tell hardware how many pulses should be high

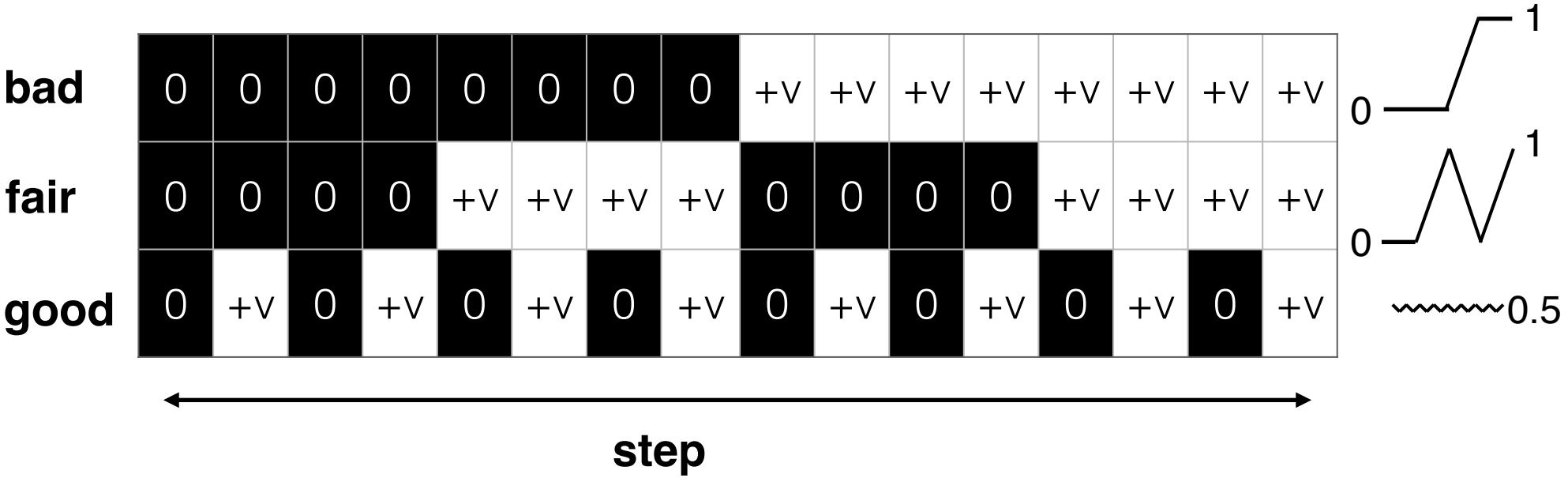
Example: Sine



64 samples

1kHz wave * 64 samples * 256 pulses = 8,192kHz

PWM Clocking of Pulses



Slowing the Waveform

50Hz

What if we want real music?

MIDI

MIDI: Musical Instrument Digital Interface

Simple interface to control musical instruments

Emerged from electronic music and instruments in 1970s

First version described in Keyboard magazine in 1982

A bit of “music”

MIDI

31.25 kbps 8-N-1 serial protocol

Commands are 1 byte, with variable parameters (c=channel, k=key, v=velocity, l=low bits, m=high bits)

Command	Code	Param	Param
Note on	`1001cccc`	`0kkkkkkk`	`0vvvvvvv`
Note off	`1000cccc`	`0kkkkkkk`	`0vvvvvvv`
Pitch bender	`1110cccc`	`0lllllll`	`0mmmmmmm`

UART (2+ pins)

Bidirectional data transfer, no clock line — “asynchronous”.

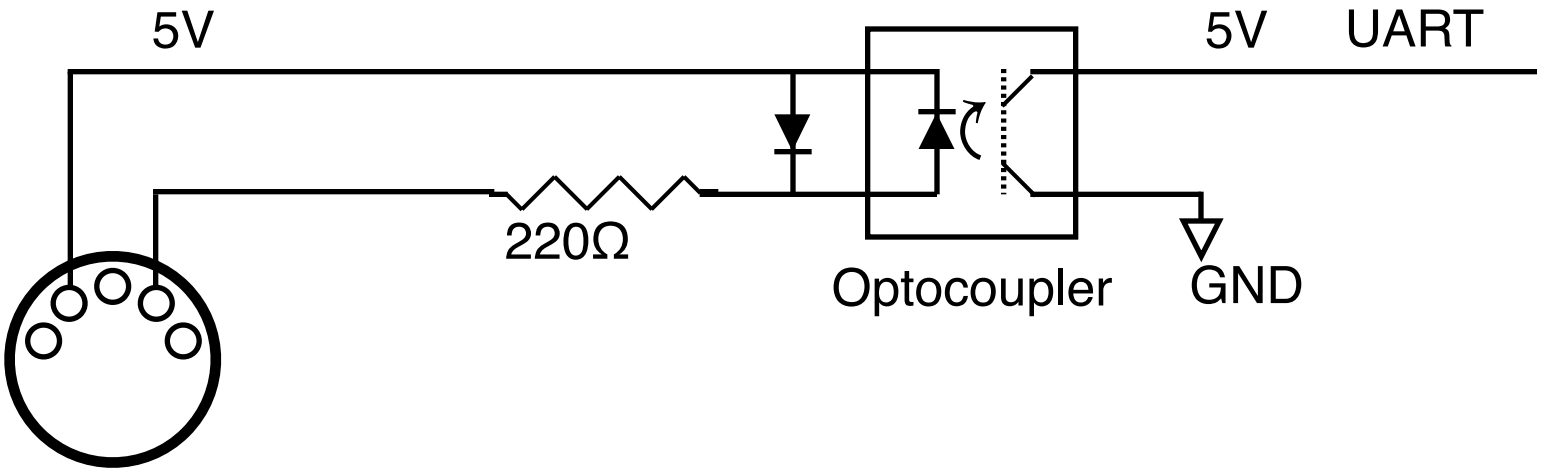
Additional pins for flow control (“I’m ready to send”), old telephony mechanisms.

Start bit, (5 to 9) data bits, (0 or 1) parity bit, (1 or 2) stop bit. 8-N-1:

start	data	data	data	data	data	data	data	data	parity	stop	stop
0	d_1	d_1	d_1	d_1	d_1	d_1	d_1	d_1		1	1

MIDI Circuit

0 is high, 1 is low!



Optocoupler completely isolates circuits electrically: no noise in instrument

MIDI Hack!