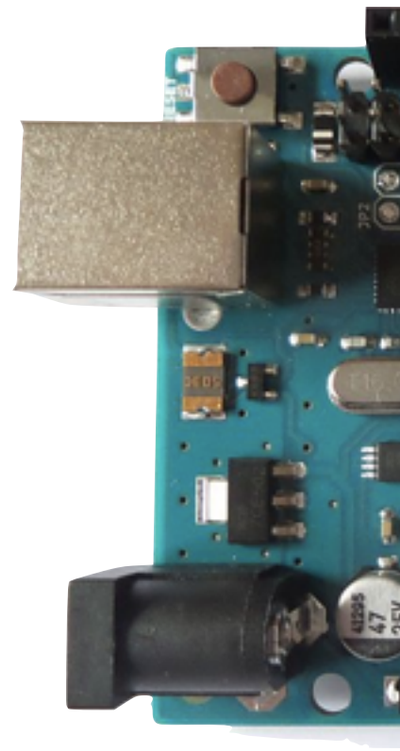
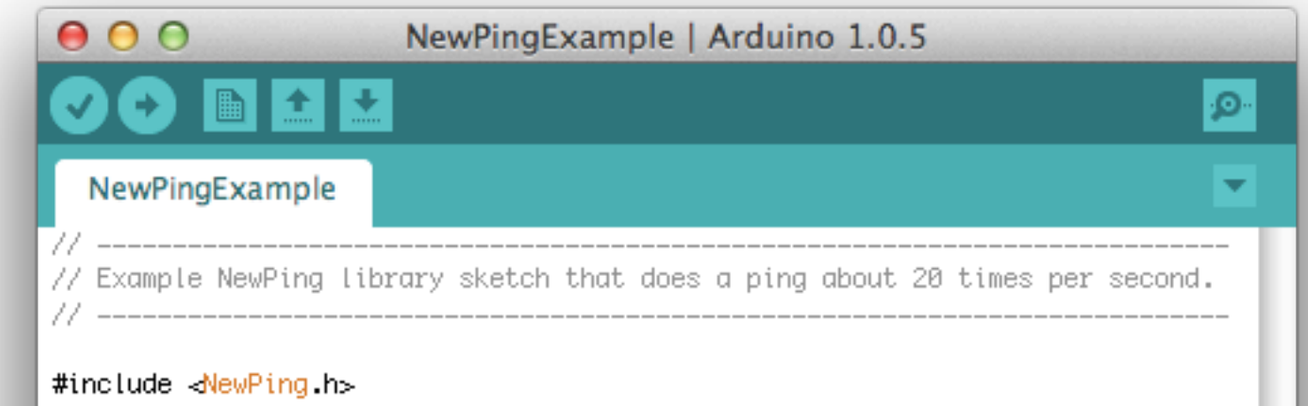


Hacking Audio

... with Arduino, Ruby and an analogue Synth

+ *TI MSP430, Introduction to some (Music-)Studio Technologies*



Recap Arduino Session

- yesterday we built a *Poor Man's Theremin*
- makes a **very cool** beginner's project
- I want to show you **how** we used
  - Arduino New Ping Library
  - MIDI
  - Arduino Midi Library

HC SR04

- sensor uses ultrasonic ~~waves~~ ~~particles~~ signals to determine the distance
- **NO! It doesn't.**
- We *can calculate the distance*, but actually it determines
  - The **time which passes between the emission** of the signal **and the detection back at the sensor again.**

Interfacing the HC SR04

- **Four I/Os:** `Vcc, GND, Trigger, Echo`
- `Vcc` **is positive supply voltage**
- `GND` **is ground**
- It works well with *5V from the Arduino* (**left pin header**, ***don't connect 3.3V to 5V***, unless you want toast. This is how you get toast.)

Interfacing the HC SR04

- When `Trigger` gets a **binary 1** (which happens to be ≥ 3.3V, which is what the GPIOs on the Arduino output) **it will emit 8 ultrasonic pulses**
- When the sensor **detects the signal back**, it sets `Echo` to **1 for a period of time,** which is **proportional** to the time the pulses travelled

Luckily we don’t have to do that ourselves. :-)

Arduino New Ping Library

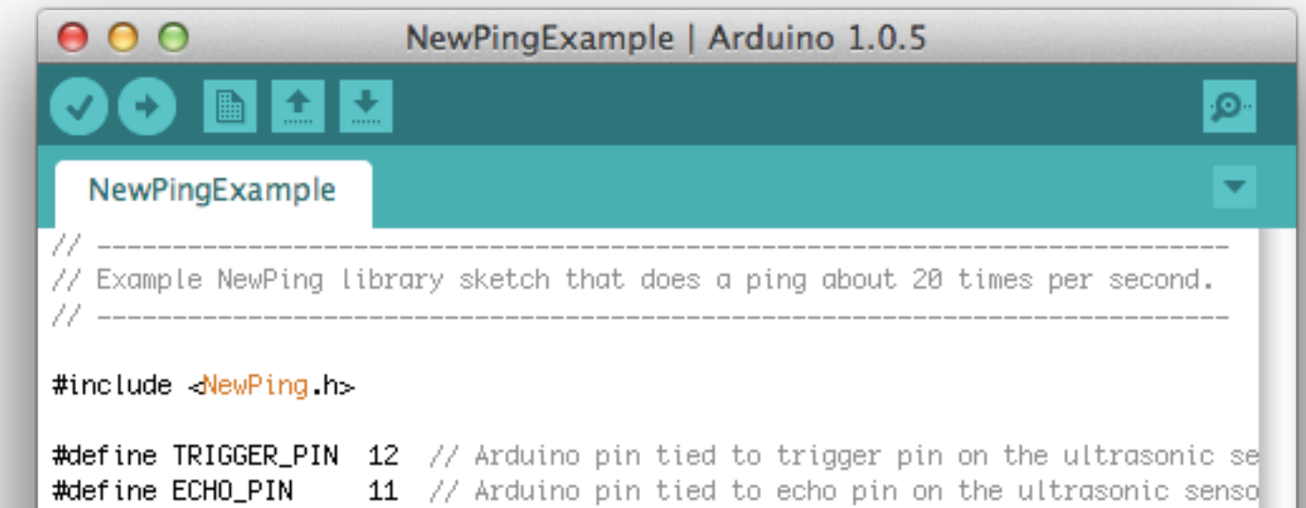
- get it here
  - playground.arduino.cc/Code/NewPing
- Installation (*with the Arduino IDE* installed):
  - **extract** and
  - **put** it to `~/Documents/Arduino/libraries`

Arduino New Ping Library

- practically only one `DEFINE` and one *method* we are interested in:
  - `NewPing#ping`
    - returns the time in μS it took for the signal to travel back
- `US_ROUNDTRIP_CM`
  - **half the time** how *many centimeters* the signal travels per **microsecond**

Arduino New Ping Library

- since libraries often bring examples, have a look in **File > Examples > NewPing**
- **try it out!**



We got the distance from the sensor now, how to make it play sounds?

Using MIDI!

MIDI

- serial protocol
- used in a *variety* of audio equipment
- does **only send information**, no sound!
- e.g. 1001nnnn kkkkkkkk vvvvvvv → Start playing **Note** $(kkkkkkkk)_2$ with a **velocity** of $(vvvvvvv)_2$ on **channel** $(nnnn)_2$

Arduino MIDI Library

- encapsulates **commands and sending** them via an *interface of your choice*
- we used the serial connection over USB, much like `Serial.print(char*)`.
- **e.g.** `MIDI.sendNoteOn(int note, int velocity, int channel)` → **sends** `note` **with** `velocity` **on** `channel`

  (MIDI uses channels to enable multiple devices to use the same connection and **daisy-chain**)

Problems

- We have **no device** which plays our MIDI notes
- The Arduino **does not identify as MIDI device** to the computer, it won't be available to us this way

Solutions

- **Download** *VSTLord* and *TAL BassLine* here:
  - phikes.de/rcg14.zip
- **Open** TAL-BassLine.vst with VSTLord
- **Choose** the IAC-Bus where "No MIDI" is labelled

Solutions

- **Download** *hairless-midi* (its also in the zip) and start it
- For MIDI IN/MIDI OUT **choose** the IAC-Bus
- **Select the serial port** where the arduino is connected

Our Setup

MIDI ⟷ USB

hairless-midi

↕

VSTLord

↕

TAL BassLine

- get the theremin project here:

  `github.com/phikes/theremin`

Ideas

- Use **accelerometers** to determine note, velocity, ...
- Program a **phone app and communicate over bluetooth** with the arduino to **make it output MIDI**
- ...

Stop! Hammer Time!

Try out the project and your ideas!

I will **continue** with the *analogue synth and Sonic Pi* (Ruby sound programming) in 15 mins.

Analogue Synth

- Korg Monotron Duo
- uses *Ribbon* as sound input, **which is fun**, but **doesn't give accurate and reproducible sounds**

Control Voltage

- when there was no MIDI, **Synths were controlled by voltage**
  - e.g. pitch, velocity (*basically the same stuff as MIDI*)
  - looooots of different interpretations (see wikipedia for examples)

Control Voltage on the Korg

- I attached two cables to V_{Rib} and V_{Gate}
- V_{Rib} is used for the Pitch (Note, which is played)
- V_{Gate} is an **enable-Input**, which needs to be set to 1, to hear a note
- (The synth needs two batteries, rated 1.5V each=3V, use the Arduino's 3V output and `GND`)

The Arduino's Analog Output Capabilities

- The Arduino uses **Pulse-Width-Modulation** to emulate an analog output
- You can use it like this `analogWrite(int pin, int value)` where the value is in 0..255

Pulse Width Modulation (PWM)

- basically **toggles the digital pin** in a certain interval on and off
- e.g. *if the pin has 3.3V* when off we can toggle it on and off the same amount of time to get 50% of the voltage (=1.65V)
- or we toggle 75% of the time on and 25% off and get 75%*3.3V.. you get it ;-)
- PWM is kind of hacky, but suffices for control voltage, so we can control the synth! (people also put out audio over pwm, try it, we have a speaker!)

Pins which have PWM enabled, have a ~ in front of their number!

Now you can use the synth to play your sounds from the distance sensor, no more MIDI!

Try out a speaker and program a specific wave (e.g. sine, or square) into the arduino. Make the arduino a synth!

Ask me if you need some help with an idea!

Bonus

- **Sonic Pi** is a musical IDE for mac and the raspberry pi
- It brings a framework which is programed in ruby
- I haven't tried it all too exhaustive, but it looks and sounds very cool!

Bonus

- I brought an **Texas Instruments MSP430** on a *launchpad*, which is also kind of an Arduino, but far cheaper
- It was *initially harder to program*, but today there is a great community
- **Energia** is a port of the Arduino environment, so you can use all the functions you already now

Have Fun! If there are questions, let me know :-).