# Arduino ABC

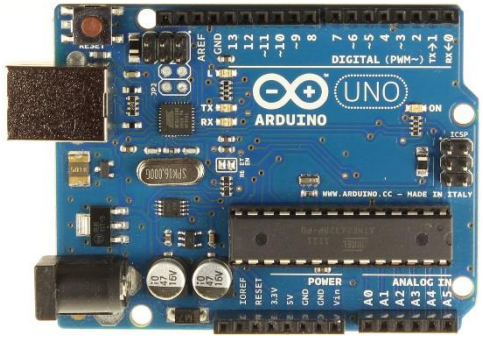
Take a good start with Arduino

{{ softshake }} 2014

http://soft-shake.ch

@GENEVE

# Agenda

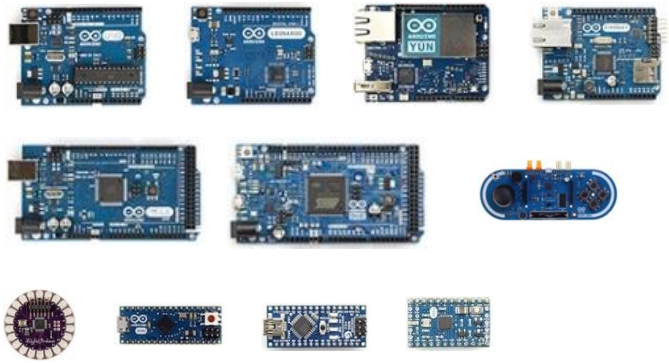



- Arduino, what's that?
- What can you do with Arduino?
- How to start?
- Arduino UNO guts
- Arduino Tools
- Hello, Arduino!
- Electronics Basics: crash course
- Sensors & Actuators
- Practical experiments: digital & analog, inputs & outputs
- Arduino «shields»
- Arduino limits
- Useful links

# Arduino, what's that?

- "Arduino is an open-source electronics platform based on easy-to-use hardware and software. It's intended for anyone making interactive projects."
- Initially a project for IDII Ivrea students (2005)
- Objectives
  - Cheap board to learn electronics
  - Quick to start with
  - Possible to "build your own"

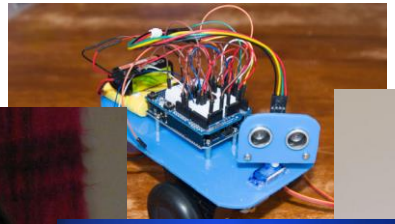
- Open Source Hardware (Arduino boards schematics)
- Open Source Software (Arduino tools and libraries)
- Today, Arduino is an "**ecosystem**"
  - Many different kinds of boards
  - Many libraries for all kinds of devices
  - Community: web site, forums, fairs…

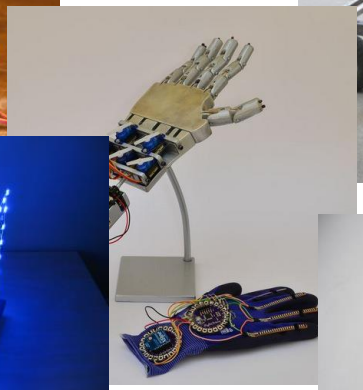


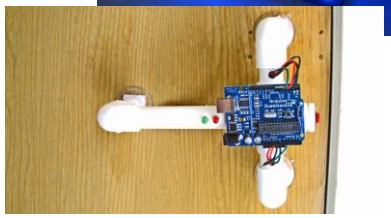
(*) photo Randi Silberman Klett

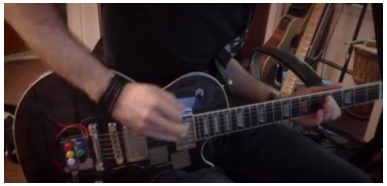
# What can you do with Arduino?

- Robots!
- Wearables
- Light decorations
- All kinds of alarms
- Weather stations
- Music
- Games
- …
- Your imagination is the limit!



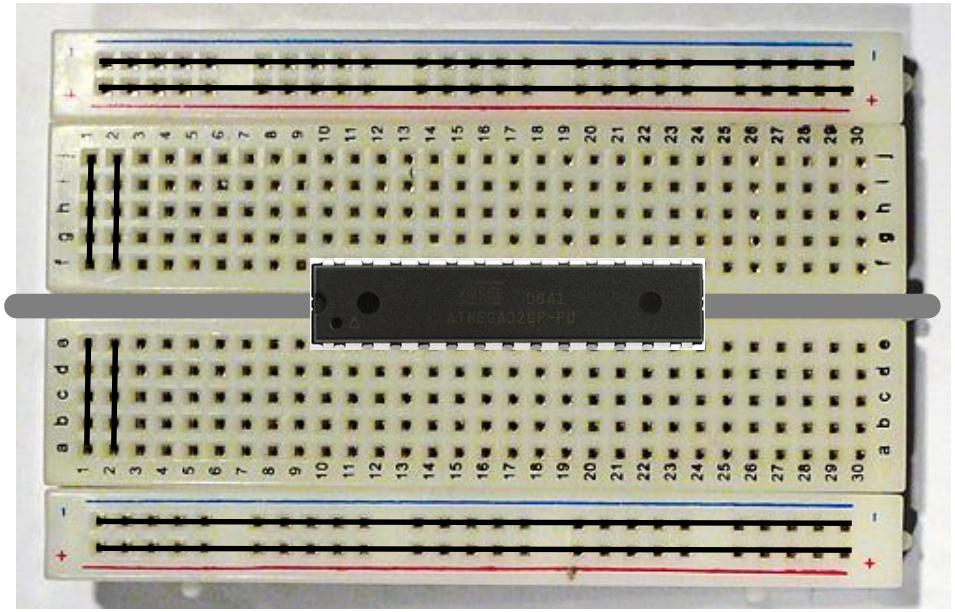



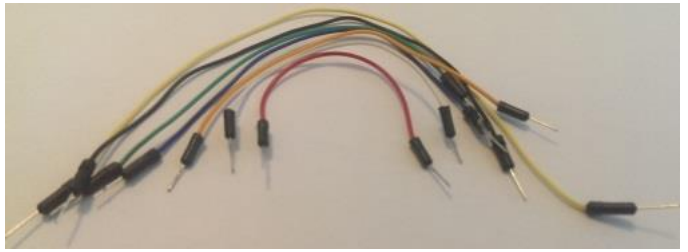



# How to start?

- Afford a starter kit
  - Arduino Starter Kit
  - Sparkfun Inventor Kit
  - Fritzing Creator Kit
  - Adafruit Experimentation Kit

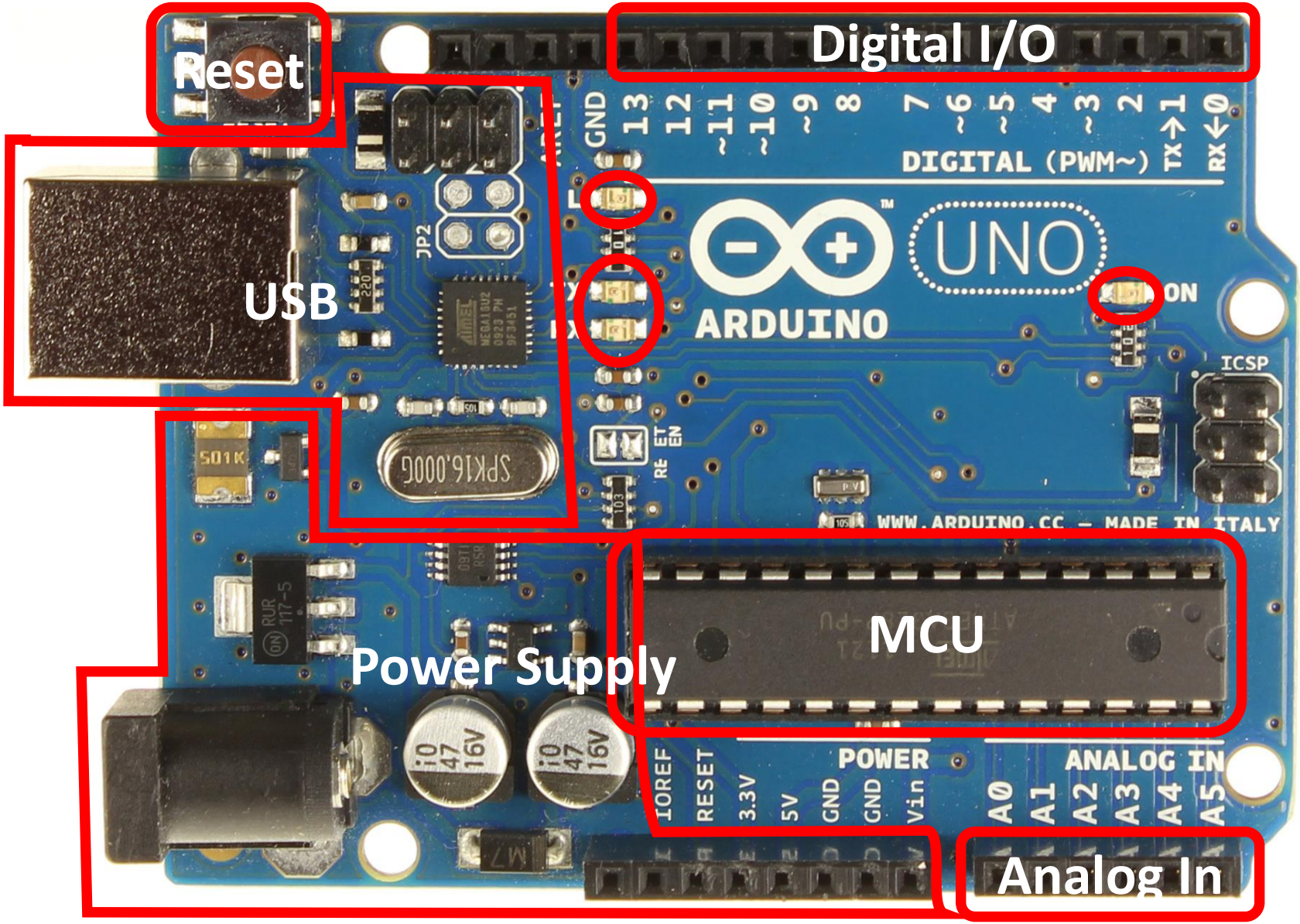

- Get comfortable with a breadboard



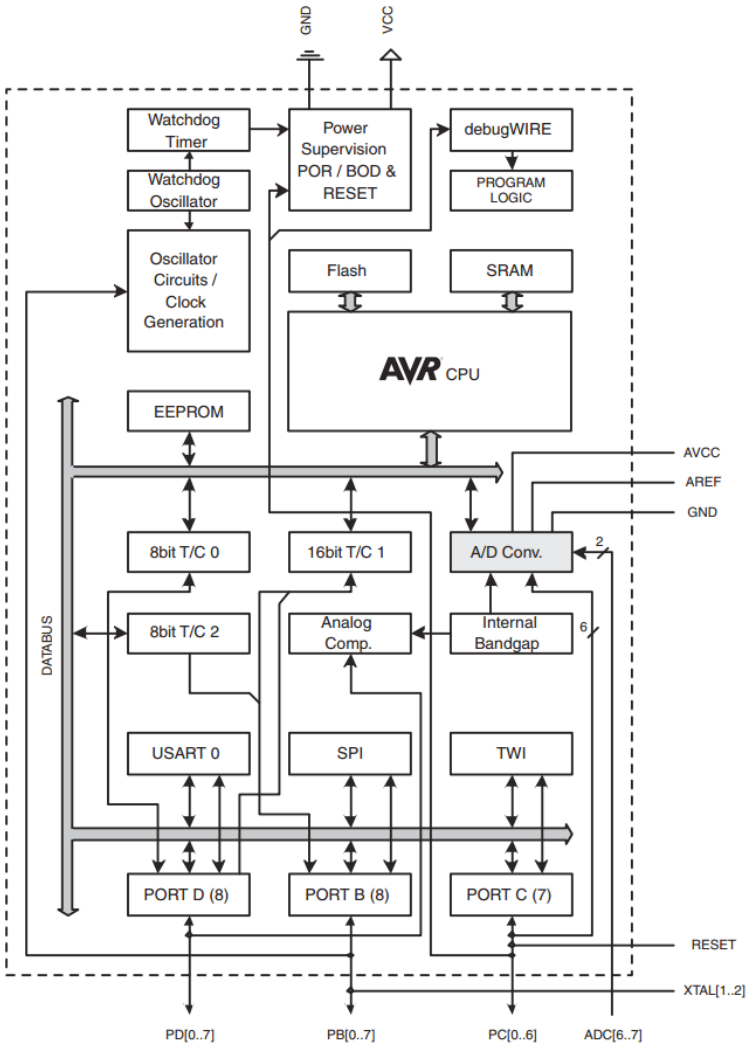



- Afford a few often useful tools

# Arduino UNO guts (board)

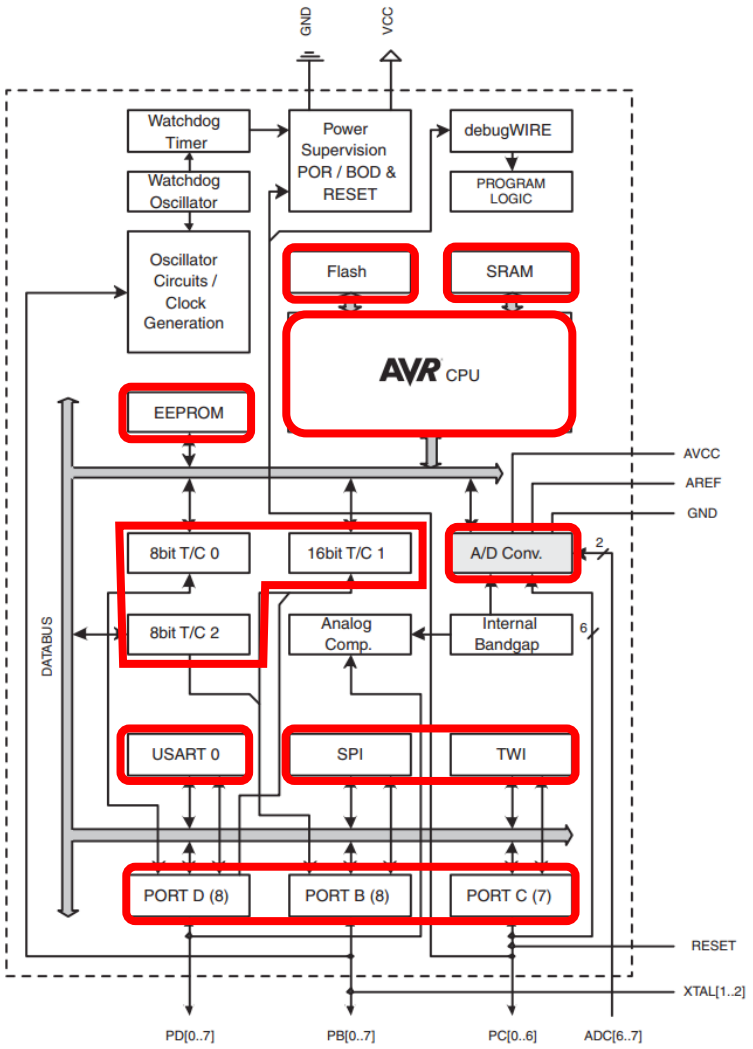


# Arduino UNO guts (MCU)




- Atmel **ATmega 328 P**
  - RISC CPU 8-bits 16 MHz
  - Flash (programs) 32 KB
  - SRAM (live data) 2 KB
  - EEPROM (non volatile data) 1 KB
  - 20 Logical IO
  - 6 Analog Inputs
  - 3 Timers
  - USART: serial communication
  - SPI/TWI: interfaces bus

# Arduino UNO guts (MCU)




- Atmel **ATmega 328 P**
  - RISC CPU 8-bits 16 MHz
  - Flash (programs) 32 KB
  - SRAM (live data) 2 KB
  - EEPROM (non volatile data) 1 KB
  - 20 Logical IO
  - 6 Analog Inputs
  - 3 Timers
  - USART: serial communication
  - SPI/TWI: interfaces bus

# Arduino Tools

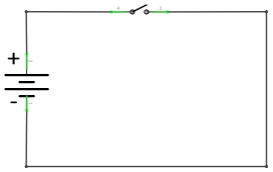
- MCU can be programmed in C, C++ or assembly
  - ATmel build tools (command-line): avr-g++, avr-gcc, avr-as, avr-ld…
- Arduino IDE integrates
  - Code editor (C/C++)
  - Builder
  - Uploader
  - Serial Monitor
- Arduino Libraries
  - Simplify developing programs (aka “sketches”) for Arduino
  - Allow the same source code to target different boards (different hardware)
- Arduino “bootloader” (flashed on Arduino Boards MCU)
  - Allows new programs upload via USB
  - Starts your programs after reset
- Other tools exist (advanced only):
  - Eclipse plugin
  - ATmel Studio (Windows only)

# Hello, Arduino!

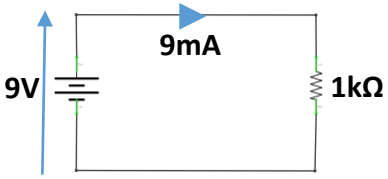
- Simplest Arduino experiment
  - No additional hardware required!
  - Uses “pin 13” on-board LED
- Connect the board through USB
- Open Arduino IDE
  - Select Arduino type and port
  - Type first “sketch” in C language
  - Save it → new folder created with “.ino” source file
  - Build it → check size
  - Upload to board
- Arduino Sketch fundamentals:
  - `void setup()`
  - `void loop()`
  - `pinMode()`
  - `digitalWrite()`
  - `delay()`

# Electronics Basics (1/2)

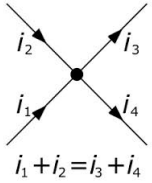
- Measures
  - U, V: voltage in Volt
  - I: current, intensity in Ampère
  - P: power in Watt
  - R: resistance in Ohm Ω
- Electric Circuit
  - A path (or network) through which current (electrons) can flow
  - Circuit must have a Voltage source (e.g. battery)



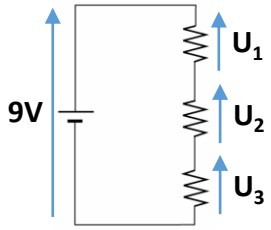

- Fundamental Laws
  - Ohm Law: U = R I




  - Kirchoff's Current Law (nodal rule): $\sum I_k = 0$



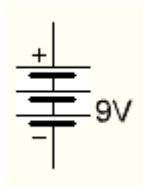

  - Kirchoff's Voltage Law (mess rule): $\sum U_k = 0$

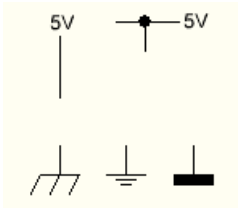



  - All 3 laws are constantly applied together

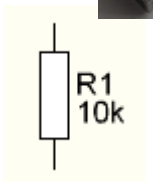
# Electronics Basics (2/2)

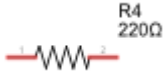
- Components
  - Voltage Generator

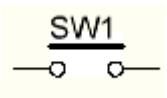



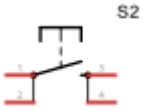

  - Resistor



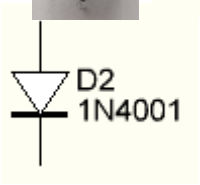



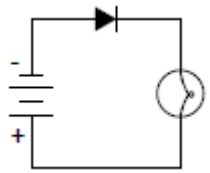

  - Switch / Button

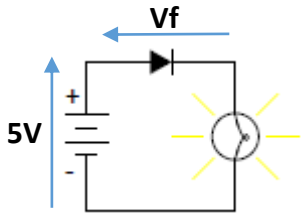


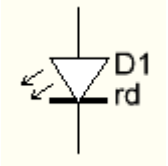

  - Diode



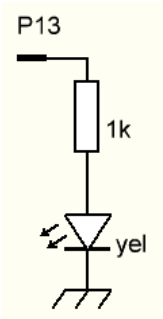






    - Vf (forward voltage) is a constant depending on the diode model
  - LED
    - Vf depends on LED color and model
    - Current must be limited to protect the LED

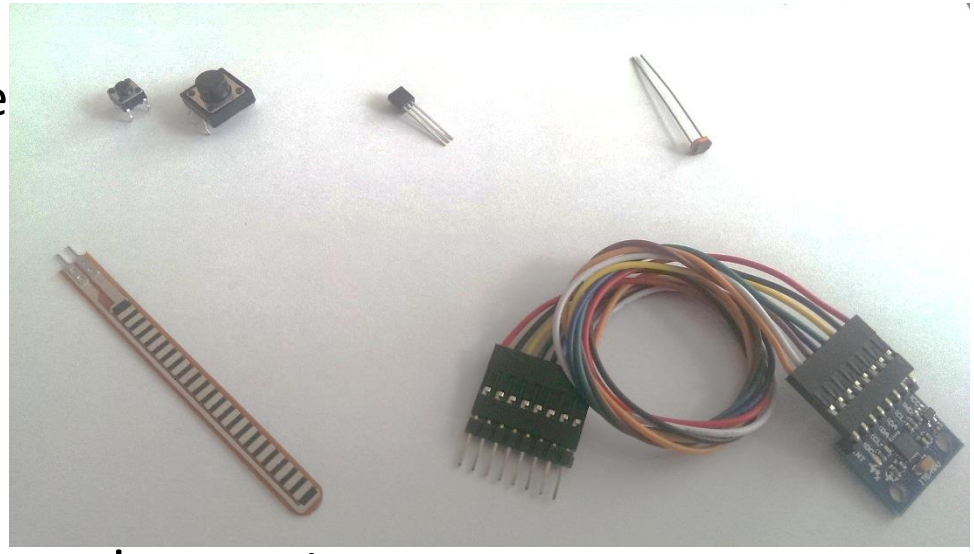



- "Hello Arduino" circuit (on-board, simplified)
  - P13 outputs either 0V or 5V
  - Why 1kΩ resistor?
  - Vf = 2.0V → i = (5 – 2) / 1000 = 3mA

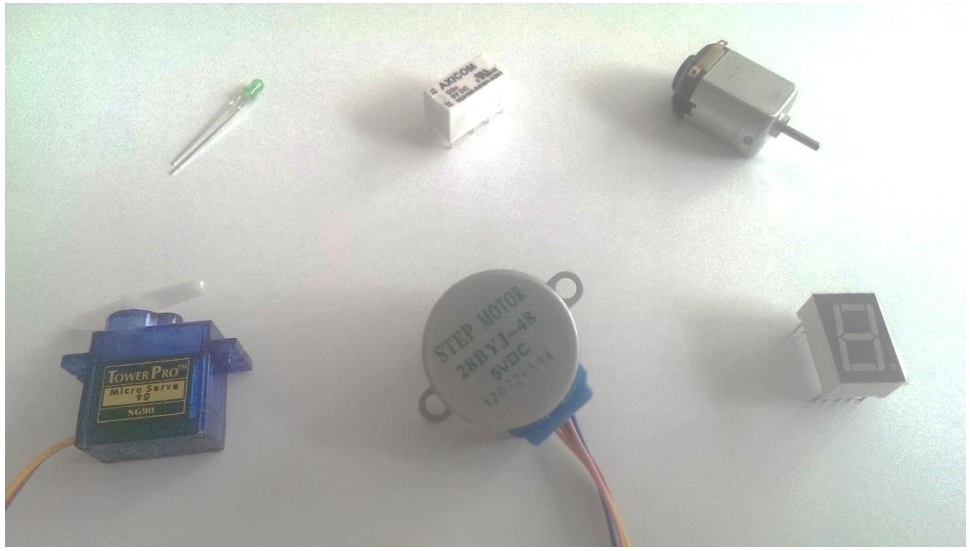


# Sensors & Actuators

- Sensors: devices to capture a physical property
  - Button
  - Temperature
  - Light
  - Flexion
  - Movement…

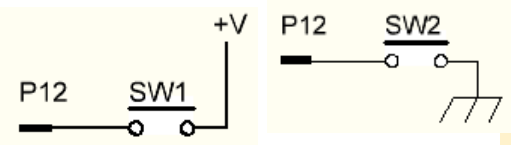


- Actuators: devices to act on environment
  - LED
  - Relay
  - DC motor
  - Servo motor
  - Stepper motor
  - Displays (LED, LCD)…

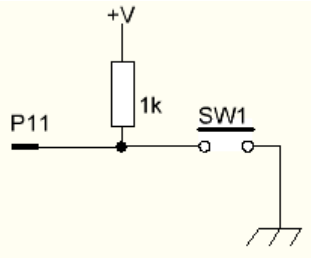


# Experiment #1 – Digital inputs

- Do something if button pushed
  - Read button state (HIGH / LOW)
  - Echo state to serial monitor
- Circuit #1.1




  - Problem? floating input!
- #1.2 with pull-up resistor

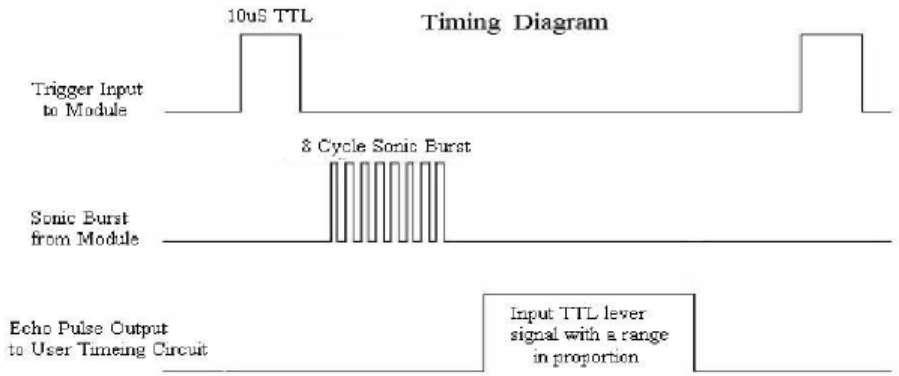



- #1.3 internal pull-up resistor
- #1.4 button bouncing issue
  - Hardware debouncing
  - Software debouncing

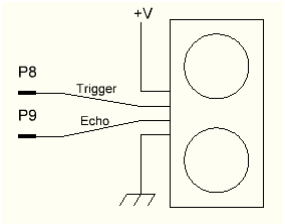
- Digital Levels in Volt:
  - 0 (LOW) → 0 – 1.67V (5V/3)
  - 1 (HIGH) → 3.3 (2x5V/3) – 5V
  - 1.67 – 3.3V → undefined!
- Digital input API
  - `pinMode(pinNum, INPUT)`
  - `digitalRead(pinNum)`
  - `pinMode(pinNum, INPUT_PULLUP)`
- Serial monitor API
  - `Serial.begin(baudRate)`
  - `Serial.println(text)`

# Experiment #2 – Distance Meter

- Use an ultrasonic sensor
  - Datasheet
  - Trigger pin
  - Echo pin
  - Range: 2cm – 4m


- Circuit #2.1

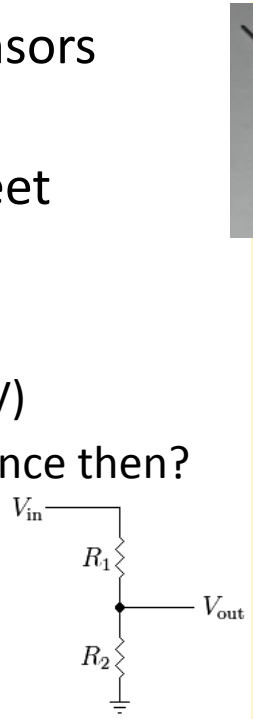



- #2.2 Distance calculation
  - Sound celerity ≈ 340m/s
    dist (mm) = echo(µs) x 340 x 1'000 / 1'000'000 / 2
- #2.3 Timeout (distance > 4m)
  - If distance > 4m
    - `pulseIn` will block **1s**!
    - specify timeout TO ≈ 23'530µs
      TO = 2 x 4 / 340 x 1'000'000

- Digital input API
  - `pinMode(pinNum, OUTPUT)`
  - `pinMode(pinNum, INPUT)`
  - `pulseIn(pinNum, level)`
  - `pulseIn(pinNum, level, to)`
- Timing API
  - `delayMicroseconds(time)`

# Experiment #3 – Analog inputs

- Variable resistance sensors
  - Light, Force, Flexion…
- Flexion sensor datasheet
  - Low accuracy (30%)!
- Arduino analog inputs
  - Measure voltage (0 – 5V)
  - How to measure resistance then?
- Voltage Divider
  - $V_{out} = V_{in} \times R_2 / (R_1+R_2)$
  - $R_1$: flex sensor
  - $R_2$: 47kΩ optim. for largest range
- Circuit #3.1: calibration
  - 0° → 133
  - 90° → 85
- #3.2: convert $V_{out}$ to angle



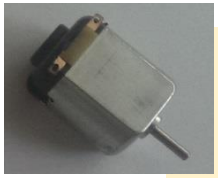

Electrical Specifications

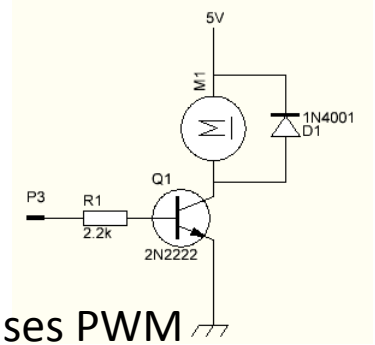
-Flat Resistance: 25K Ohms
-Resistance Tolerance: ±30%
-Bend Resistance Range: 45K to 125K Ohms (depending on bend radius)
-Power Rating : 0.50 Watts continuous. 1 Watt Peak

- Analog input API
  - `analogRead(pinNum)`
    - Returns int [0;1023]
- Utility API
  - `map(value, srcLow, srcHigh, dstLow, dstHigh)`

# Experiment #4 – Analog outputs

- Drive a DC Motor
  - Vary voltage → vary speed
- Caution! High current!
  - Can't drive directly from Arduino!
  - Use a transistor
- Lenz's Law* → Back emf!
  - Fly-back diode



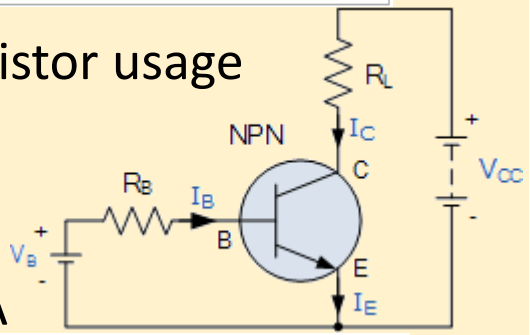



- [Circuit]
  - Pin 3 ~
  - Analog output uses PWM
  - Pulse-Width Modulation

**General specifications**

| | |
|---|---|
| **Free-run speed @ 6V:** | 11500 rpm |
| **Free-run current @ 6V:** | 70 mA |
| **Stall current @ 6V:** | 800 mA |

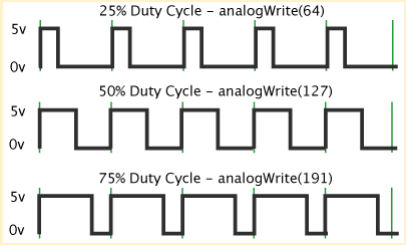
- NPN transistor usage




- P2N2222A

| Characteristic | Symbol | Value | Unit |
|---|---|---|---|
| Collector – Emitter Voltage | $V_{CEO}$ | 40 | Vdc |
| Collector – Base Voltage | $V_{CBO}$ | 75 | Vdc |
| Emitter – Base Voltage | $V_{EBO}$ | 6.0 | Vdc |
| Collector Current – Continuous | $I_C$ | 600 | mAdc |

- Analog output API
  - `analogWrite(pinNum, value)`

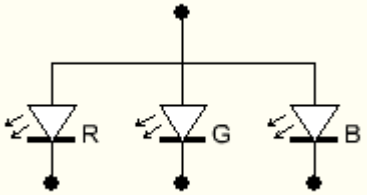



(*) An induced electromotive force (emf) always gives rise to a current whose magnetic field opposes the original change in magnetic flux.

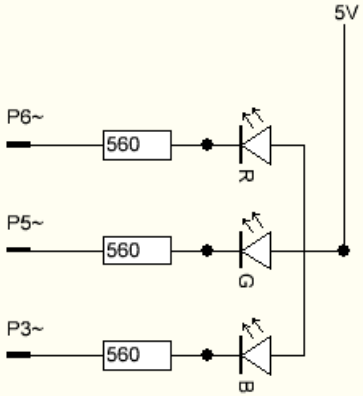
# Experiment #4 – Analog outputs

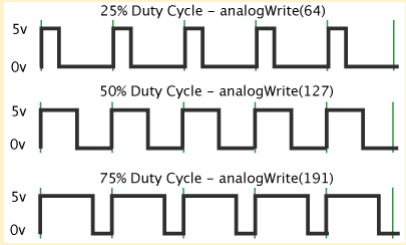
- Display colors on a LED
  - 3 LED (R,G,B) in one
  - Common anode
  - Datasheet

- Forward Voltage (RGB): (1.9, 3.0, 3.0)V
- Max Forward Current (RGB): (20, 20, 20)mA
- Max Luminosity (RGB): (2800, 6500, 1200)mcd




- Circuit
  - Only Analog Output (~) pins
  - Resistors to limit current
  - Ohm law:
    - Red: I = 3.1 / 560 = 5.5mA
    - Green, Blue: 2 / 560 = 3.6mA
  - Analog output uses PWM
  - Pulse-Width Modulation



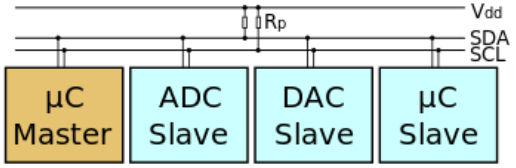




- Analog output API
  - `analogWrite(pinNum, value)`

# Buses – I²C & SPI

- Devices can perform complex functions & need complex communication with an MCU:
  - 3 axes accelerometer, IO multiplexers, LCD displays, Flash IC, SD Card, RTC…
- I²C & SPI are "standard" buses for electronic devices to interact

- **I²C / TWI**



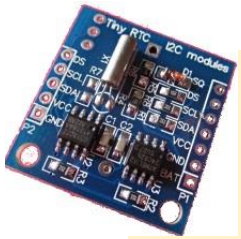

- By Philips, standardized
- N Masters, P Slaves (address)
- 2 wires
- Connection wires need pull-up
- Half-duplex communication
- Speed: 100/400kHz (Arduino)
- Rather simple API
  - `#include <Wire.h>`

- **SPI**


- By Motorola, de facto standard
- 1 Master, N Slaves
- (3 + N) wires
- Direct connection wiring
- Full-duplex communication
- Speed: up to 4MHz (Arduino)
- Rather simple API
  - `#include <SPI.h>`

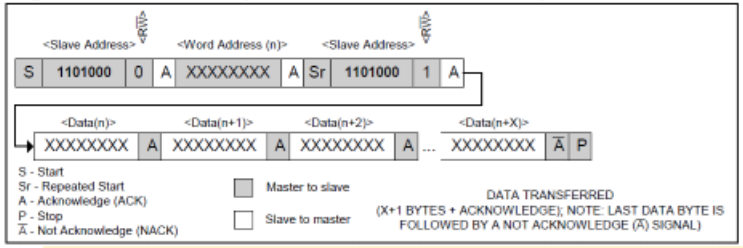
# Experiment #5 – I²C Real Time Clock

- Keep accurate date/time
  - even when switched off
- IC DS1307 + breakout board
  - I²C based, 100kHz, slave only
  - Imposed address: 0x68
  - Supply voltage: 5V
  - Time stored in “registers”
  - Reading Time =
    - Write address 1st register to read
    - Read all registers
- Circuit
  - I²C uses A4 (SDA) & A5 (SCL)
  - Pullup resistors (4.7kΩ each)



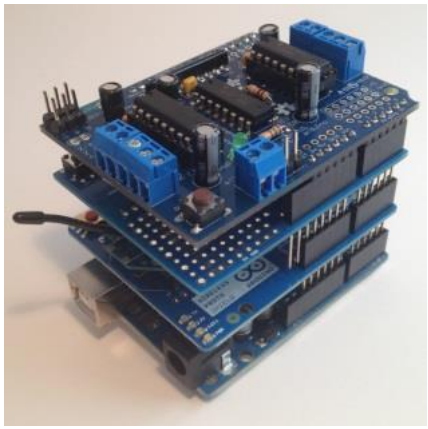

| ADDRESS | BIT 7 | BIT 6 | BIT 5 | BIT 4 | BIT 3 | BIT 2 | BIT 1 | BIT 0 | FUNCTION | RANGE |
|---|---|---|---|---|---|---|---|---|---|---|
| 00h | CH | 10 Seconds | | | Seconds | | | | Seconds | 00–59 |
| 01h | 0 | 10 Minutes | | | Minutes | | | | Minutes | 00–59 |
| 02h | 0 | 12 | 10 Hour | 10 Hour | Hours | | | | Hours | 1–12 +AM/PM |
| | | 24 | PM/ AM | | | | | | | 00–23 |
| 03h | 0 | 0 | 0 | 0 | 0 | DAY | | | Day | 01–07 |
| 04h | 0 | 0 | 10 Date | | Date | | | | Date | 01–31 |
| 05h | 0 | 0 | 0 | 10 Month | Month | | | | Month | 01–12 |
| 06h | 10 Year | | | | Year | | | | Year | 00–99 |
| 07h | OUT | 0 | 0 | SQWE | 0 | 0 | RS1 | RS0 | Control | — |

**Figure 6. Data Read (Write Pointer, Then Read)—Slave Receive and Transmit**




- Wire API
  - `Wire.begin()`
  - `Wire.beginTransmission(addr)`
  - `Wire.write(byte)`
  - `Wire.endTransmission()`
  - `Wire.requestFrom(addr, size)`
  - `Wire.read()`

# Arduino «shields»



- Shields are boards that “plug” on an Arduino board
- You can piggy-pack several shields on the same Arduino board
- Official Arduino shields
- Third-party shields
- Various kinds
  - Ethernet
  - WIFI
  - Motors
  - GSM
  - Prototype…

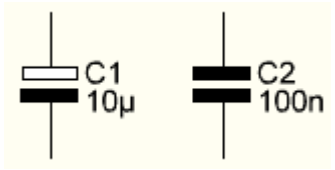
- Easy to use but rather expensive
- Conflicts (pins sharing) between several boards

# Arduino limits (& workarounds)

- Clock speed: 16MHz
  - Generally not a problem unless you need image processing…
  - If needed, use 2nd processor (e.g. Raspberry Pi) for heavy processing
- Flash (program) size: 32KB
  - Linker is smart enough to remove unused functions from final binary
  - Rule of thumb: remove all libs you don't use
- SRAM (data) size: 2KB
  - Don't use dynamic memory allocation
  - Remove useless global variables; prefer local variables
  - Put literal strings to Flash
- Power consumption:
  - Advanced: use ATmel MCU sleep modes
- IO pins number
  - Use "shift register" IC (present in most starter kits)
  - Use IO multiplexer IC (uses $I^2C$ bus)

# Electronics Pitfalls

- Voltage drops due to high current consumption
  - Experiment #2.4: distance detector + LED display
  - Strange values can appear due to voltage drop!
  - → Separate power sources (keep common ground)
  - → Use decoupling capacitor
  - A capacitor is like an accumulator that loads & unloads charges very quickly




- Voltage levels 3.3V Vs. 5V
  - Mixing voltages may fry your Arduino or other circuits
  - → Voltage adapters: various kinds with different advantages
- Switching high-currents
  - Directly plugging high-current devices to Arduino will fry it!
  - → Transistors, Darlington (cascaded transistors), “Drivers”
  - → MOSFET
- Self inductance back emf when motors switched off
  - Can burn your complete circuit (very high voltage in short time)
  - → Fly back diode

# What I wish I had time to talk about…

- Transistors to drive higher current devices
- MOSFET (to do the same)
- Capacitors (always useful in electronics)
- Experiment driving a servo motor
- Experiment driving a stepper motor
- $I^2C$ and SPI buses and devices examples
- Interrupts on pin level changes
- More details about Arduino memory organization, tips and tricks
- …

- All that might be in a future Arduino Advanced Talk…

# Useful links

- Where to buy Arduino in Switzerland
  - http://www.play-zone.ch/
  - http://boxtec.ch/
  - http://www.conrad.ch/
- Arduino reference, Q&A
  - http://arduino.cc/
  - http://arduino.stackexchange.com/
- Major contributors
  - http://www.adafruit.com/
  - https://www.sparkfun.com/
- Interesting sites
  - Debouncing: http://www.ikalogic.com/de-bouncing-circuits/
  - Memory: https://learn.adafruit.com/memories-of-an-arduino/you-know-you-have-a-memory-problem-when-dot-dot-dot
  - Several advanced topics: http://www.gammon.com.au/forum/bbshowpost.php?bbtopic_id=123

# Thank you!

**Contact:**

Jean-François Poilprêt


@jfpoilpret


https://github.com/jfpoilpret


https://www.linkedin.com/pub/jean-francois-poilpret