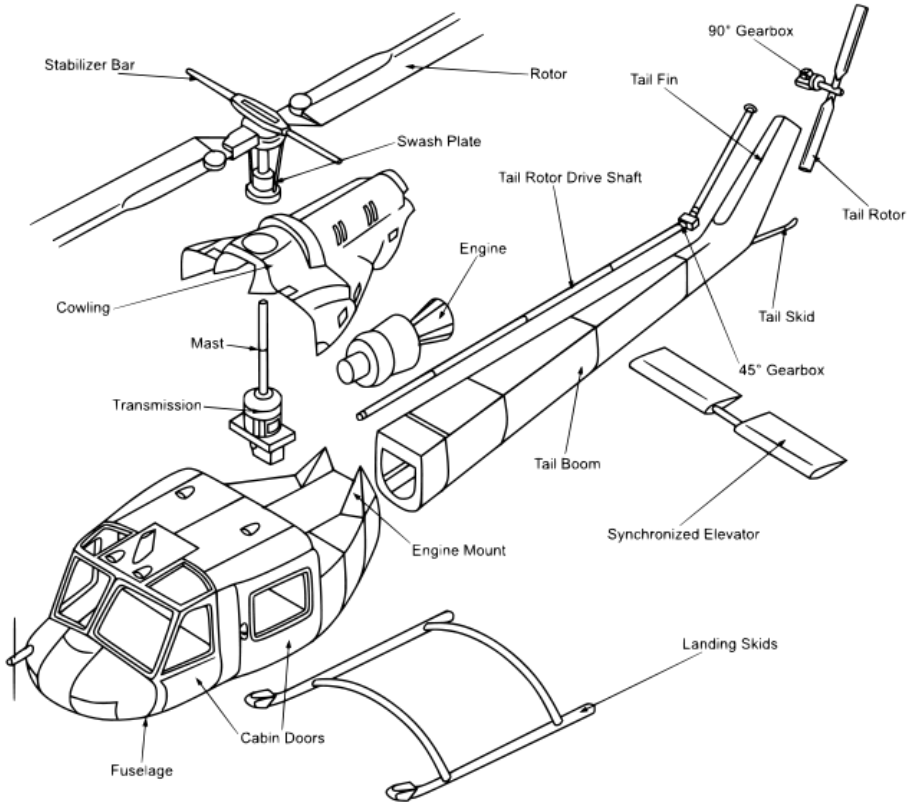
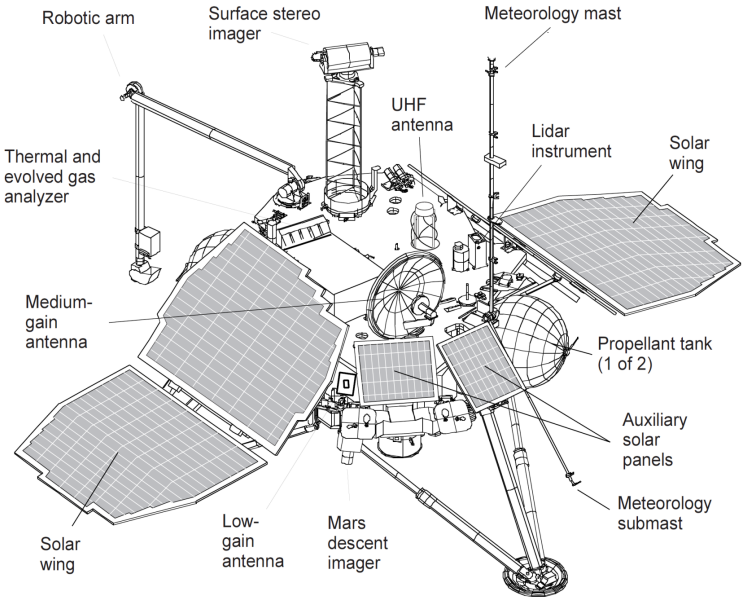
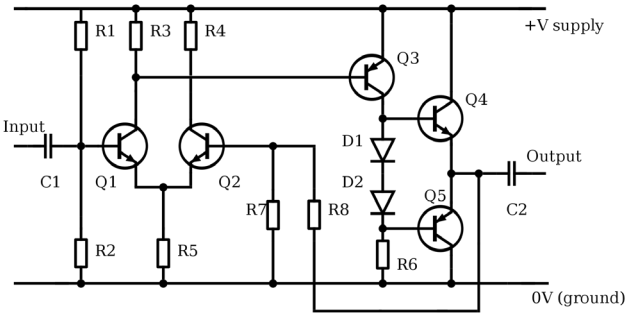




Lecture 6 – Microcontrollers

Andrés Faíña (anfv@itu.dk)



News

You can take an Arduino kit until March 25

- Sign up after the lab
- Practice different circuits at home
- Arduino Kit Guide is a recommended lecture

From March 25, take Arduinos boards, shields, sensors and hardware, but no Arduino kits

Overview

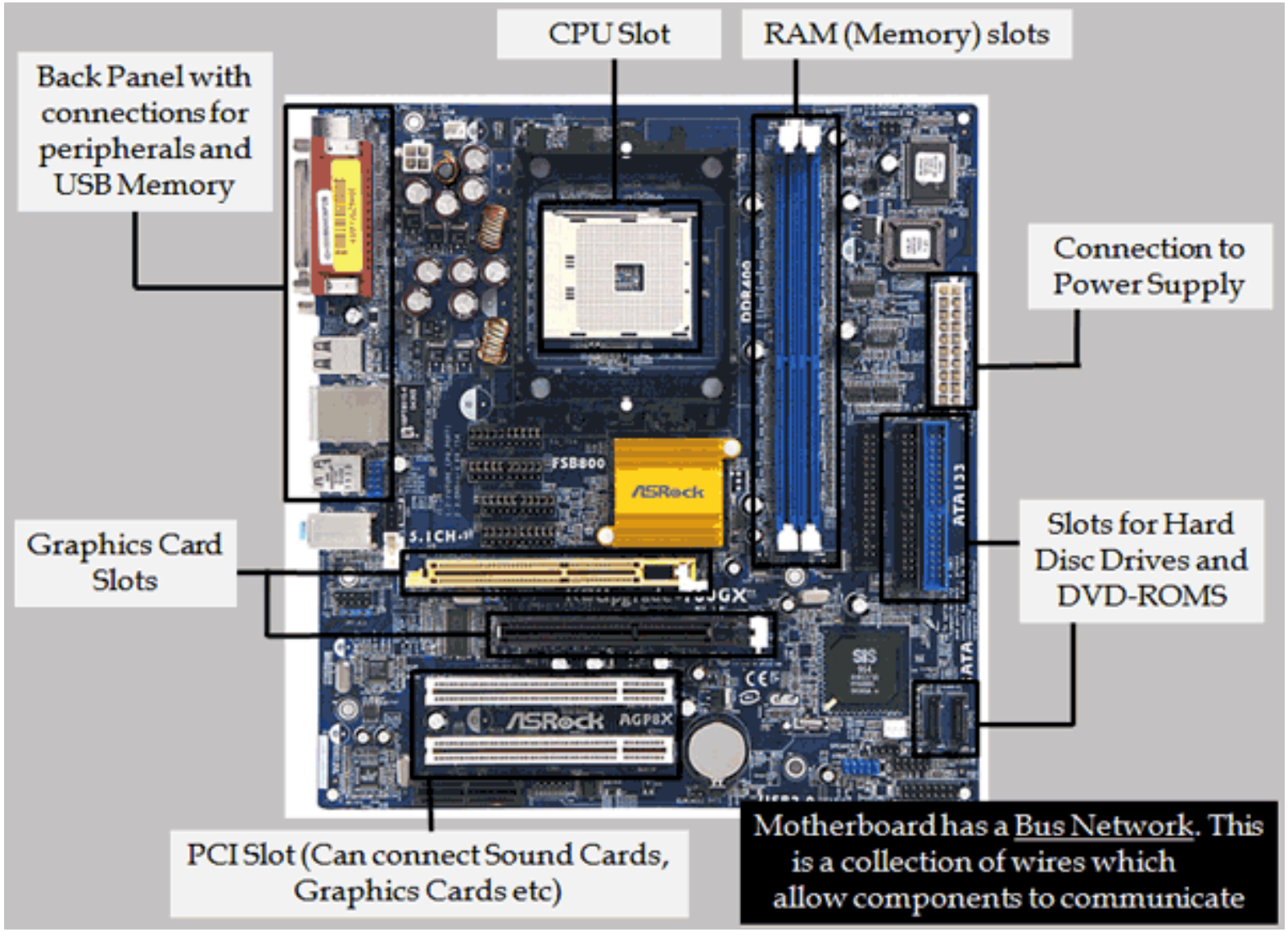
Microcontroller

Microcontrollers simplified: Arduino

Handling basic sensors

Controlling motors

Computers

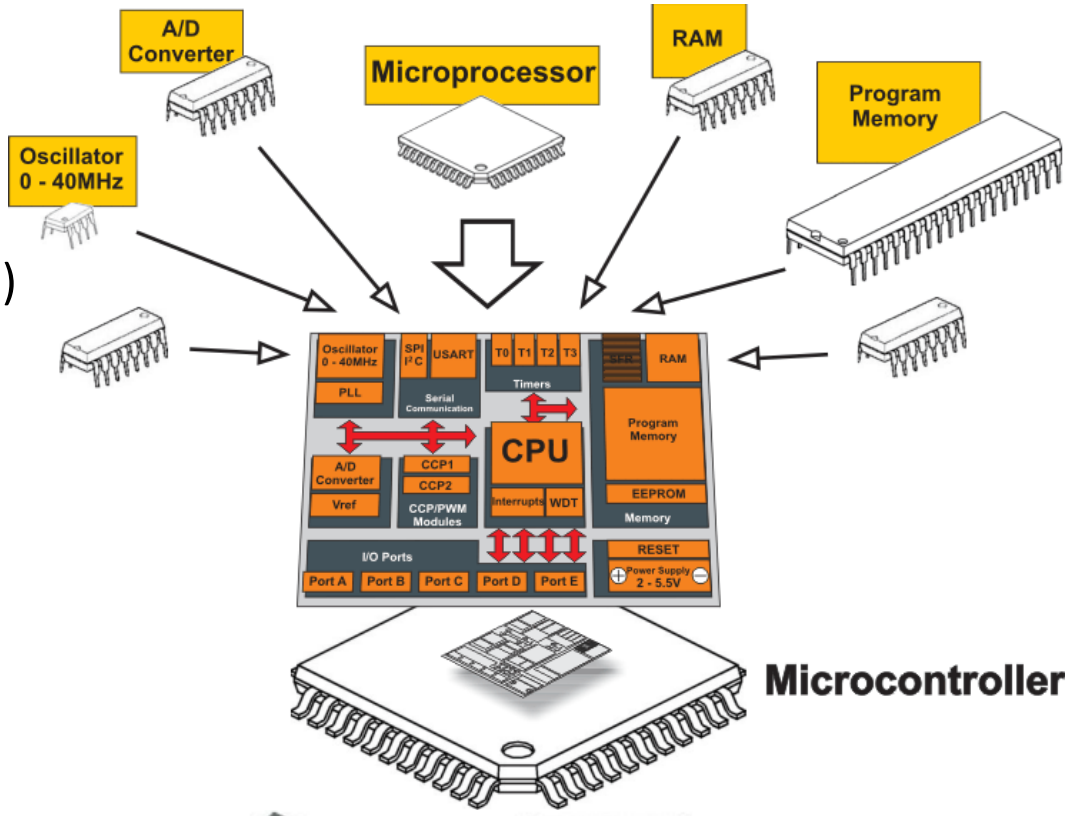

CPU Slot
RAM (Memory) slots
Back Panel with connections for peripherals and USB Memory
Connection to Power Supply
Graphics Card Slots
Slots for Hard Disc Drives and DVD-ROMS
PCI Slot (Can connect Sound Cards, Graphics Cards etc)
Motherboard has a Bus Network. This is a collection of wires which allow components to communicate
FSB800
ASRock
AGP8X
ATA133
SiS

Microcontrollers

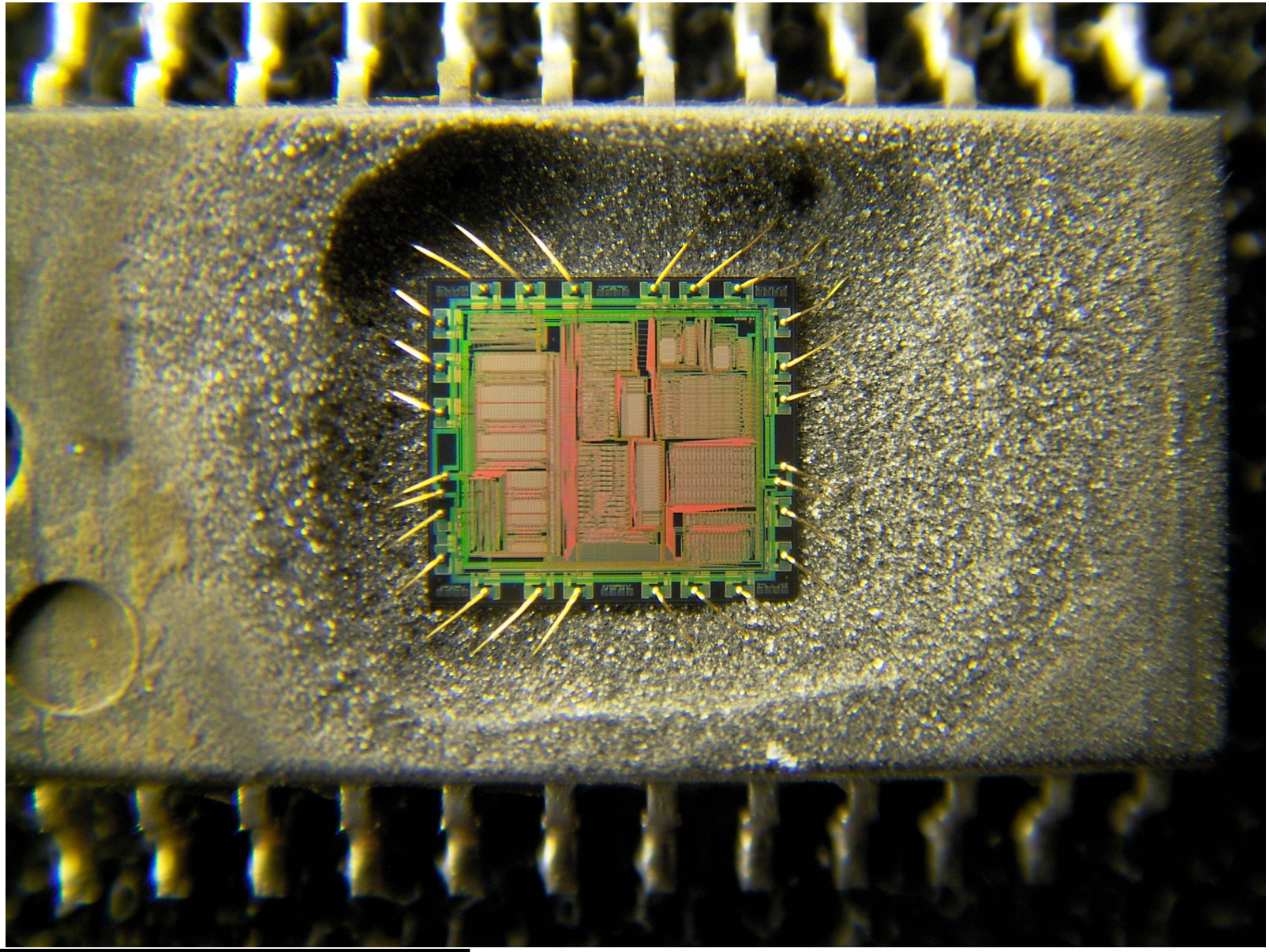
One Integrated Circuit (IC):

Processor
Memory
- ROM:
  - Program (Flash)
  - Permanent data (EEPROM)
- RAM:
  - variables

Input/Output Peripherals
- Digital Input / Output
- Analog Inputs (A/D Converter)
- Timers
- Serial ports
  - UART
  - SPI
  - I2C
  - USB
- Parallel ports



Microcontrollers - ICs



A microcontroller – ATmega328P

High Performance, Low Power Atmel®AVR® 8-Bit Microcontroller Family

- Advanced RISC Architecture
  - 131 Powerful Instructions
  - Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 20 MIPS Throughput at 20MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory Segments
  - 32KBytes of In-System Self-Programmable Flash program Memory
  - 1KBytes EEPROM
  - 2KBytes Internal SRAM
  - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  - Data Retention: 20 years at 85°C/100 years at 25°C(1)
  - Optional Boot Code Section with Independent Lock Bits
    - In-System Programming by On-chip Boot Program
    - True Read-While-Write Operation

I/O and Packages

- 23 Programmable I/O Lines
- 28-pin PDIP, 32-lead TQFP, 28-pad QFN/MLF and 32-pad QFN/MLF

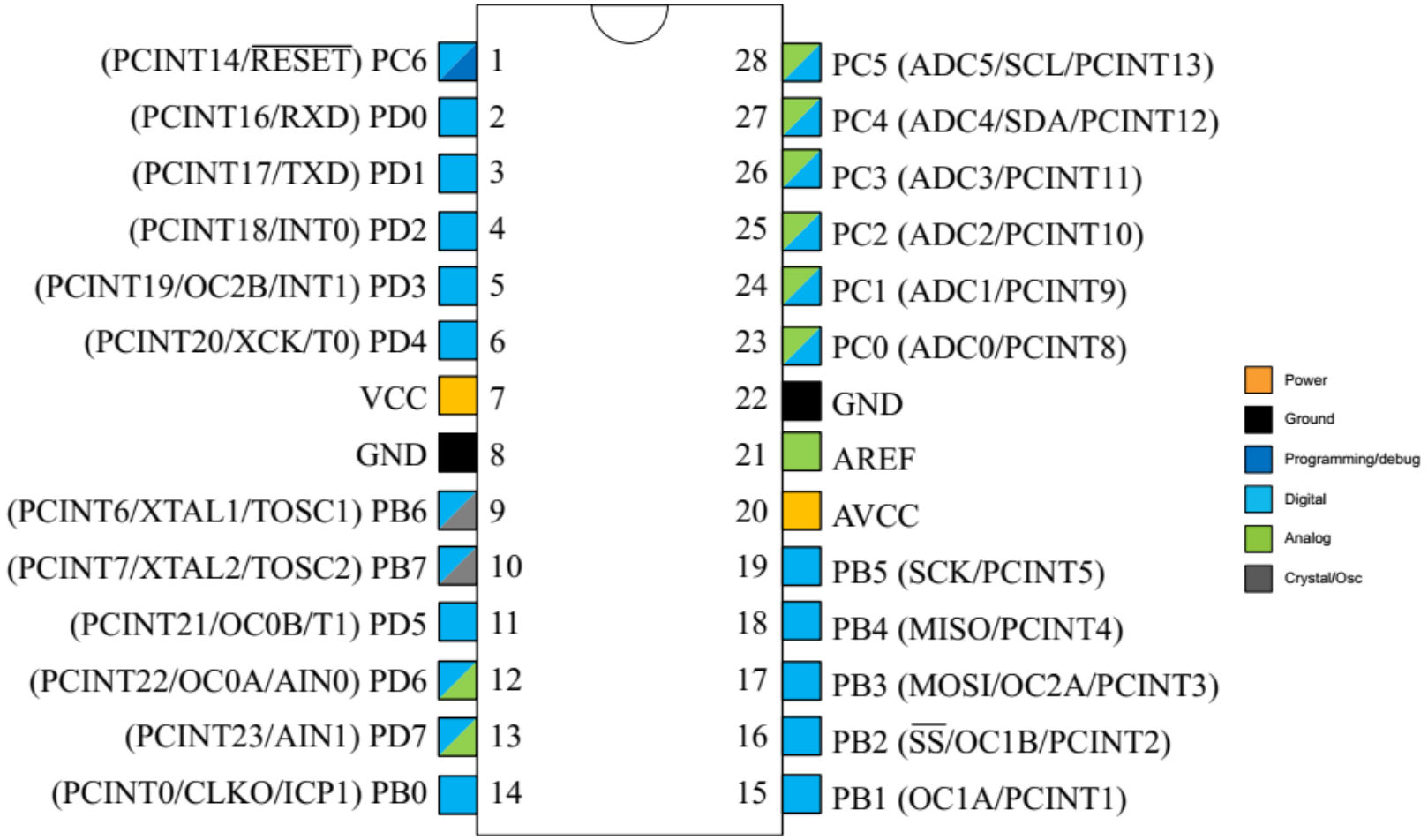
Operating Voltage:

- 1.8 - 5.5V

A microcontroller – ATmega328P

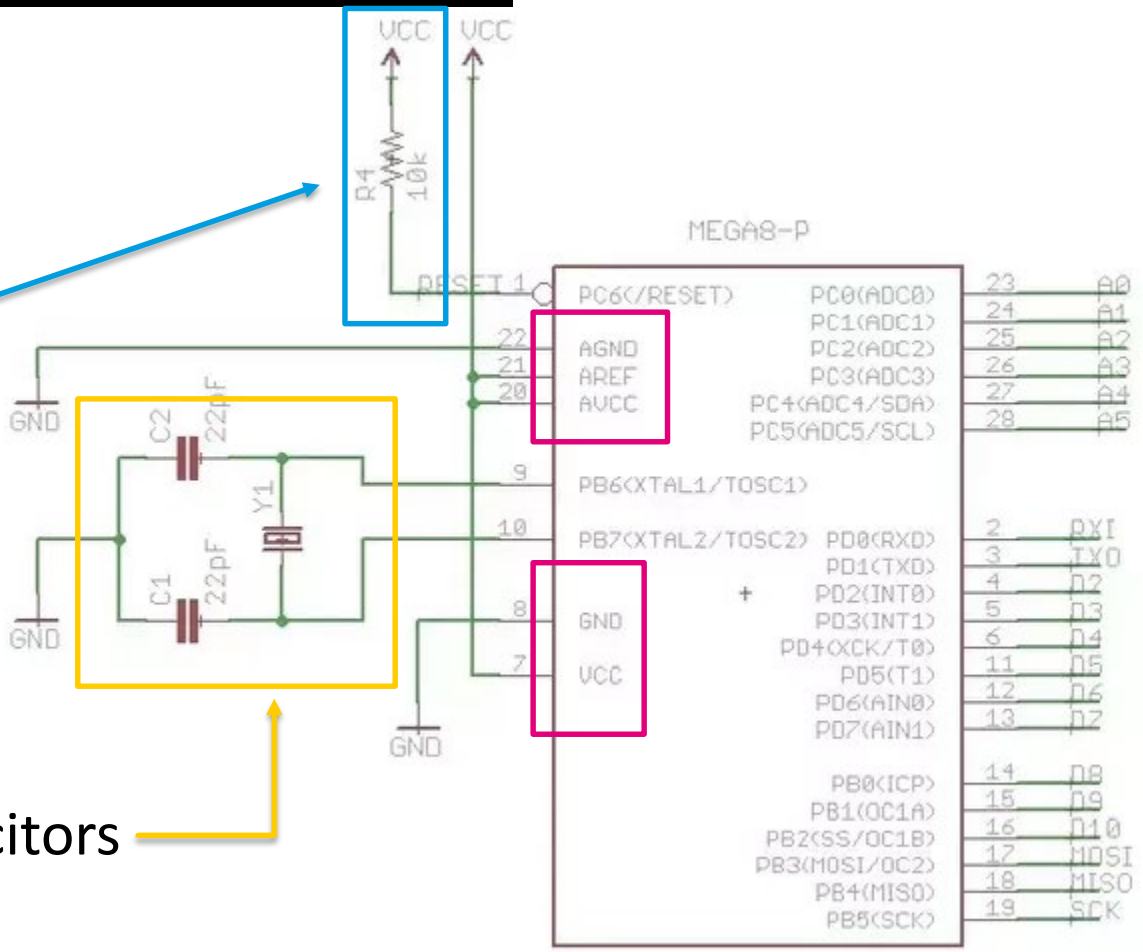
5.1. Pin-out

Figure 5-1. 28-pin PDIP



Microcontrollers – Basic circuit

1. Power
2. Reset circuit
3. Oscillator
    - Internal RC
        - Not precise
        - Cheap
    - Crystal and 2 capacitors
        - Precise
        - More components



Atmega168/328

Exercise #0

Build this circuit using a microcontroller

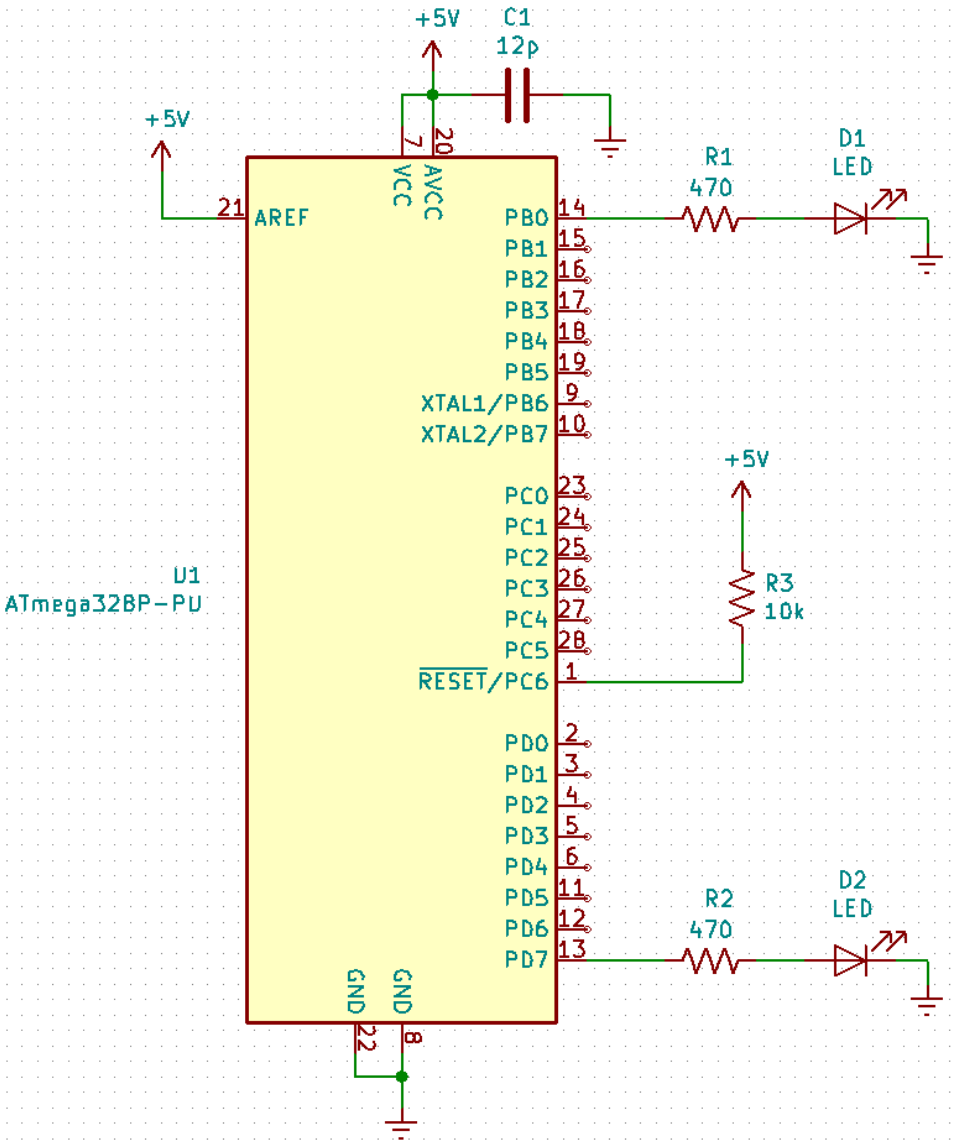
Oscillator:

Internal RC

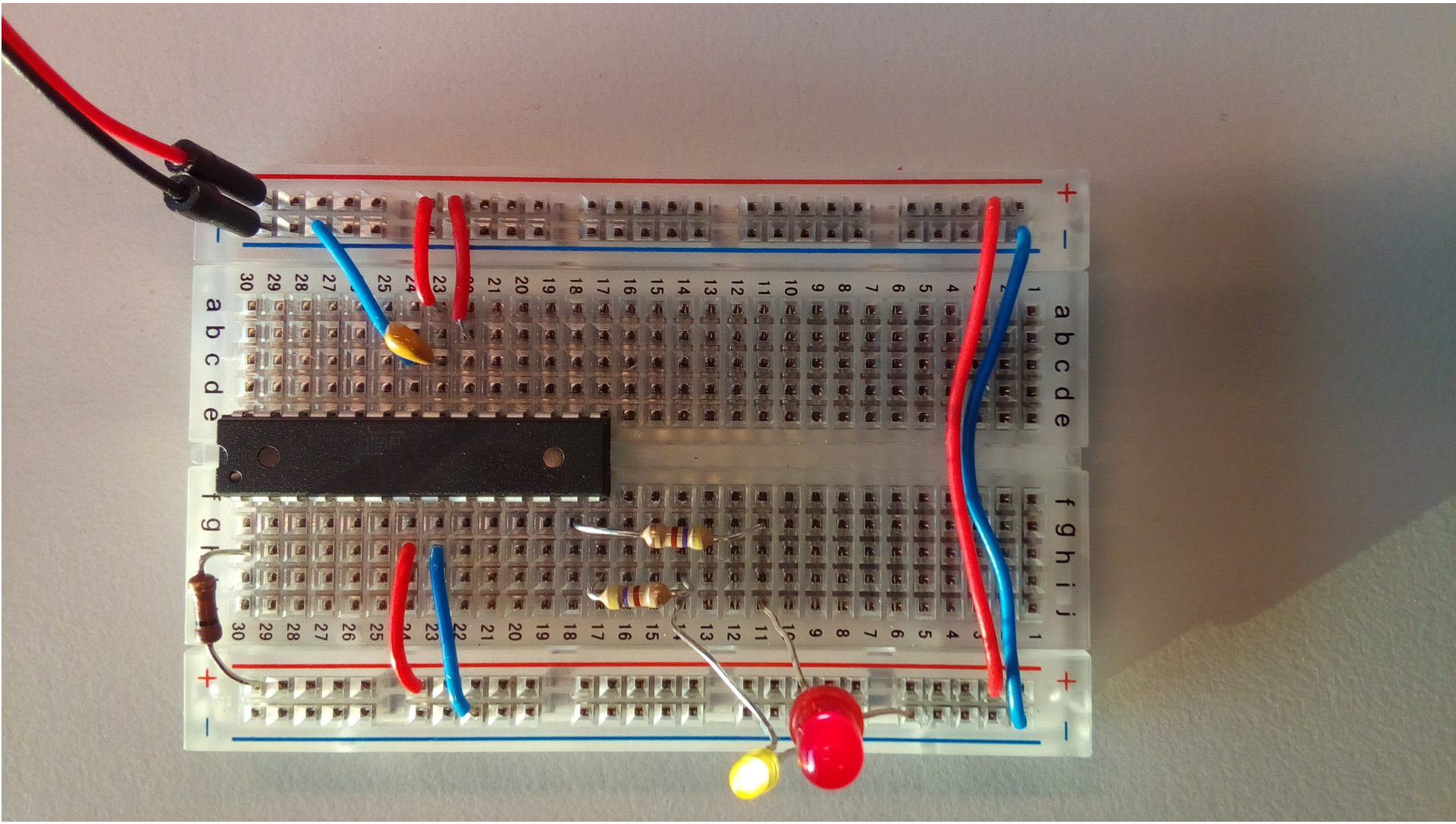
Check the pinout!

The LEDs blink when the circuit is correct

Remember the resistor for the LEDs

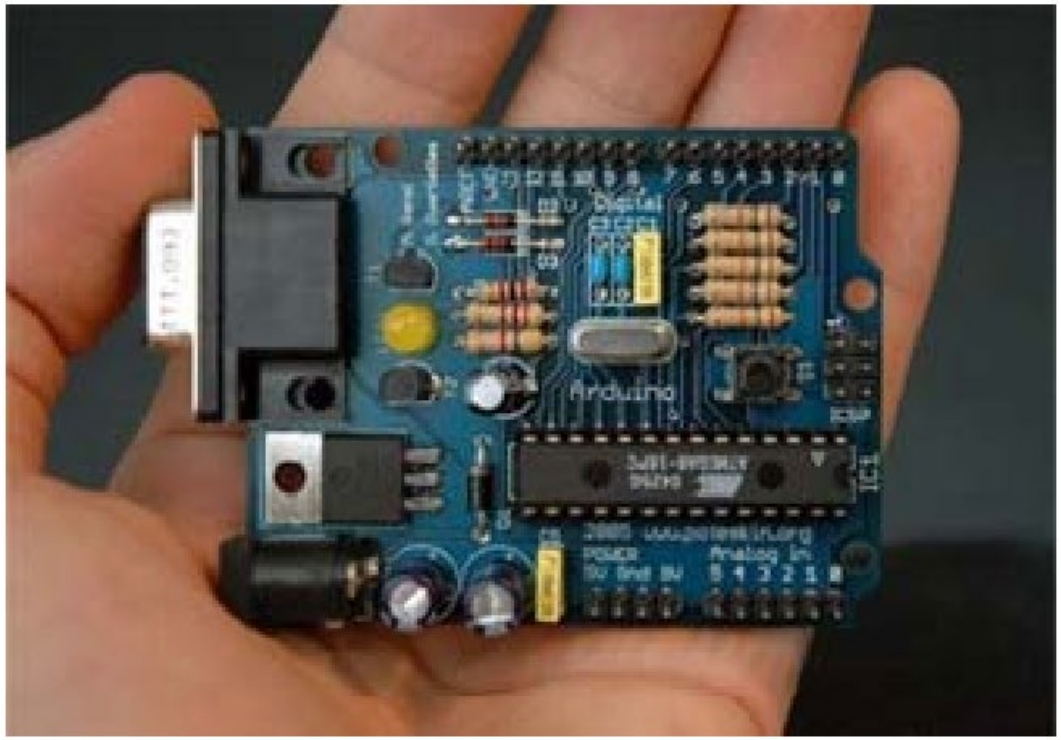


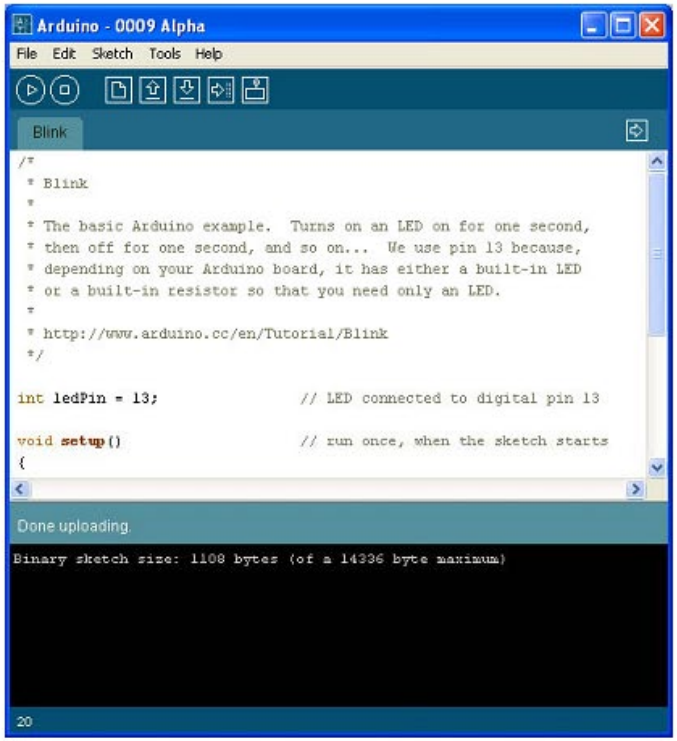
Exercise #0 - Solution



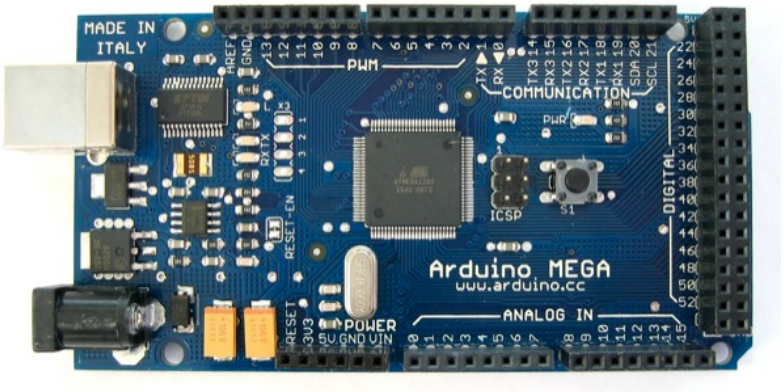
Arduino

Arduino is an open-source electronics platform based on easy-to-use hardware and software. It's intended for anyone making interactive projects.

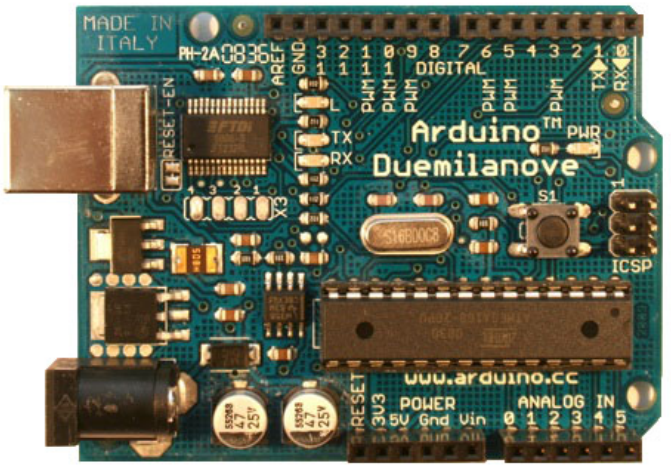




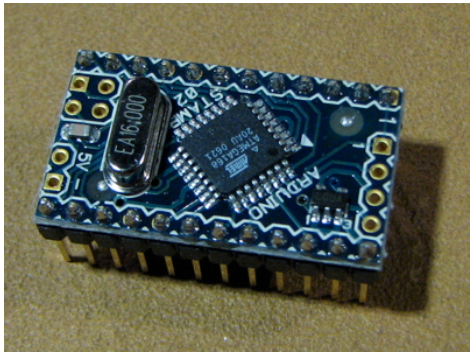
Arduino: Basic Hardware



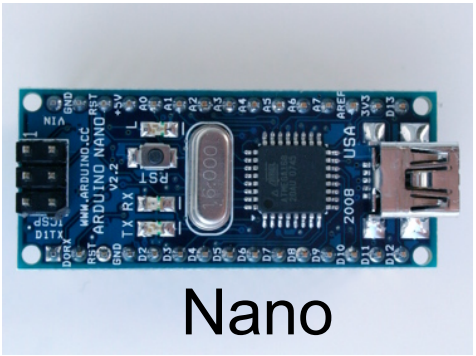
Mega



Duemilanove

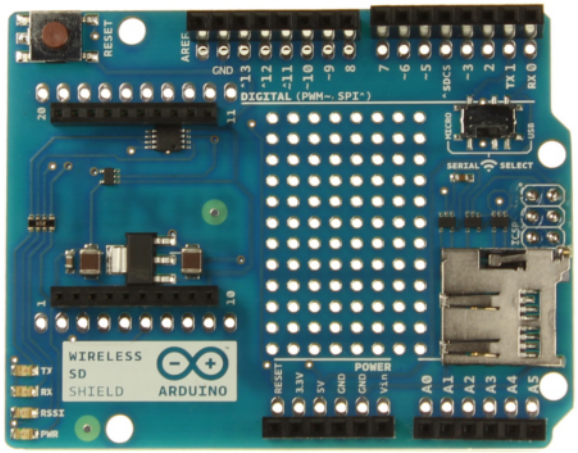


Mini

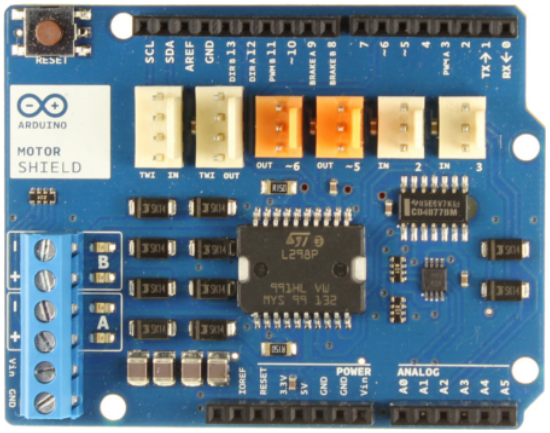


Nano

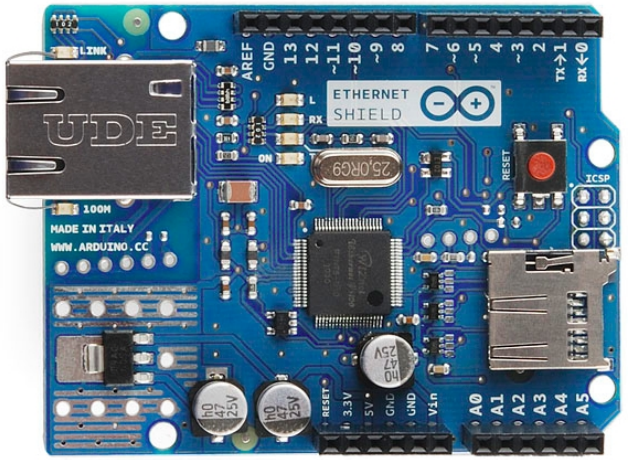
Arduino: Expansion Hardware



Wireless SD Shield



DC Motor Shield

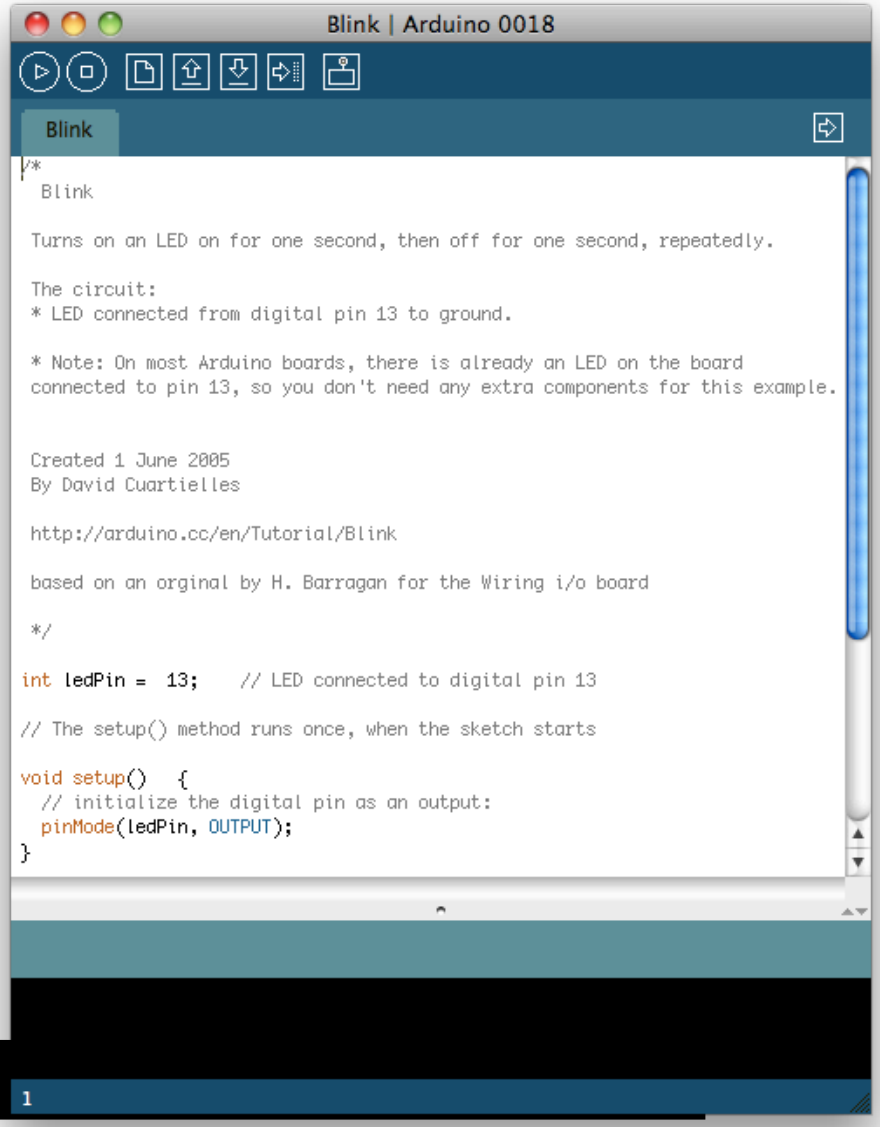


Ethernet Shield

Benefits of Arduino

- Fast development of projects
  - Open source and extensible hardware
  - Open source and extensible software
  - Simple, clear programming environment
- Inexpensive
- Cross-platform
- Projects that need to
  - Handle digital and analog Inputs/Outputs
  - Handle motors
    - Servos
    - DC Motors
    - Stepper motors

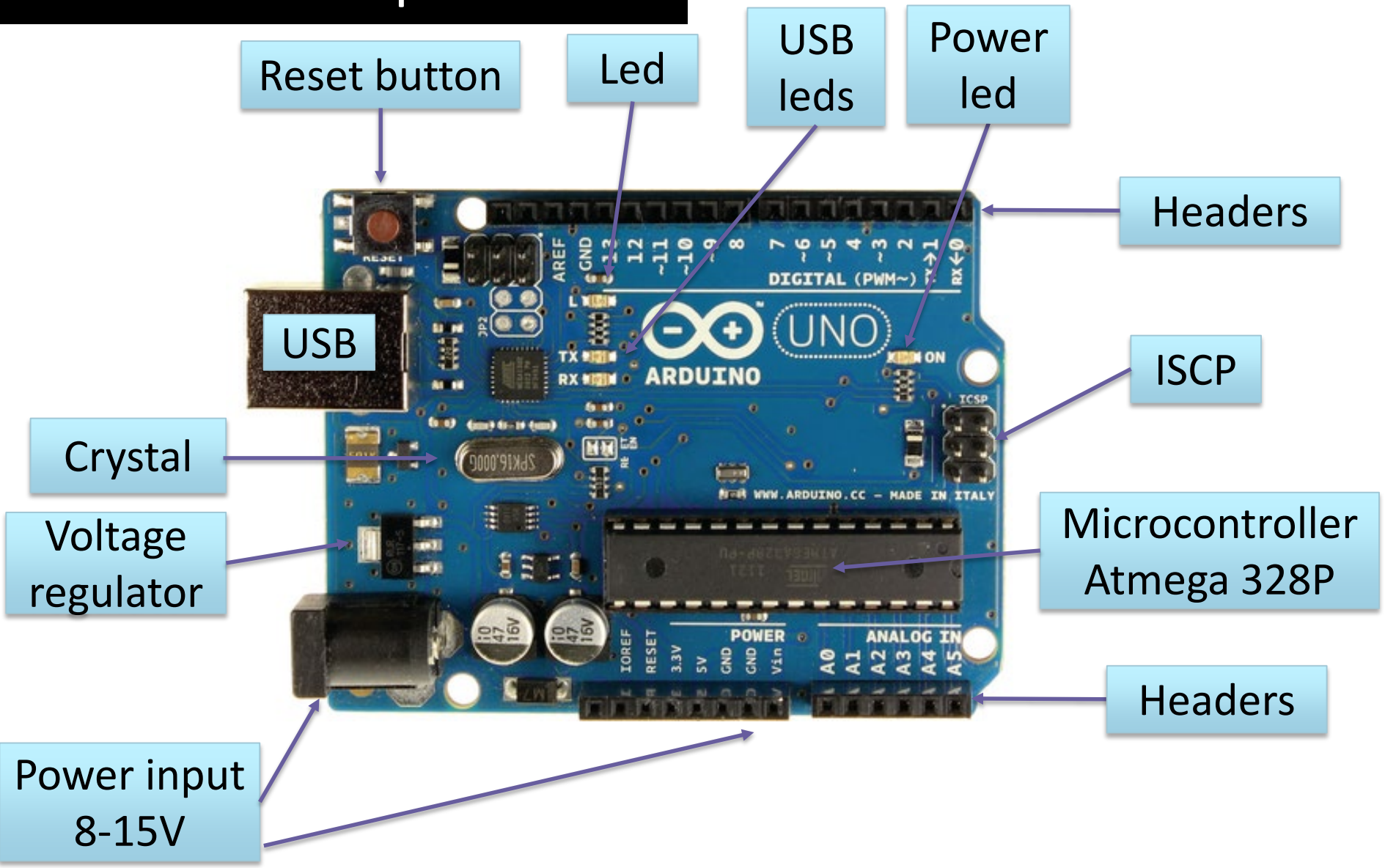
Arduino: Programming



Libraries

EEPROM - reading and writing to "permanent" storage
Ethernet - for connecting to the internet using the Arduino Ethernet Shield
Firmata - for communicating with applications on the computer using a standard serial protocol.
LiquidCrystal - for controlling liquid crystal displays (LCDs)
SD - for reading and writing SD cards
Servo - for controlling servo motors
SPI - for communicating with devices using the Serial Peripheral Interface (SPI) Bus
SoftwareSerial - for serial communication on any digital pins
Wire - Two Wire Interface (TWI/I2C) for sending and receiving data over a net of devices or sensors.

Arduino - Components



Arduino environment

Hardware:

Microcontroller on a PCB
Basic circuits
- Oscillator
- Reset button
- 1 LED
- Voltage regulators

USB communications
Standard Headers (Connectors)

Extensions (shields)

Software:

Libraries
- ucontroller hardware
- Shield boards
- Sensors

Programing interface

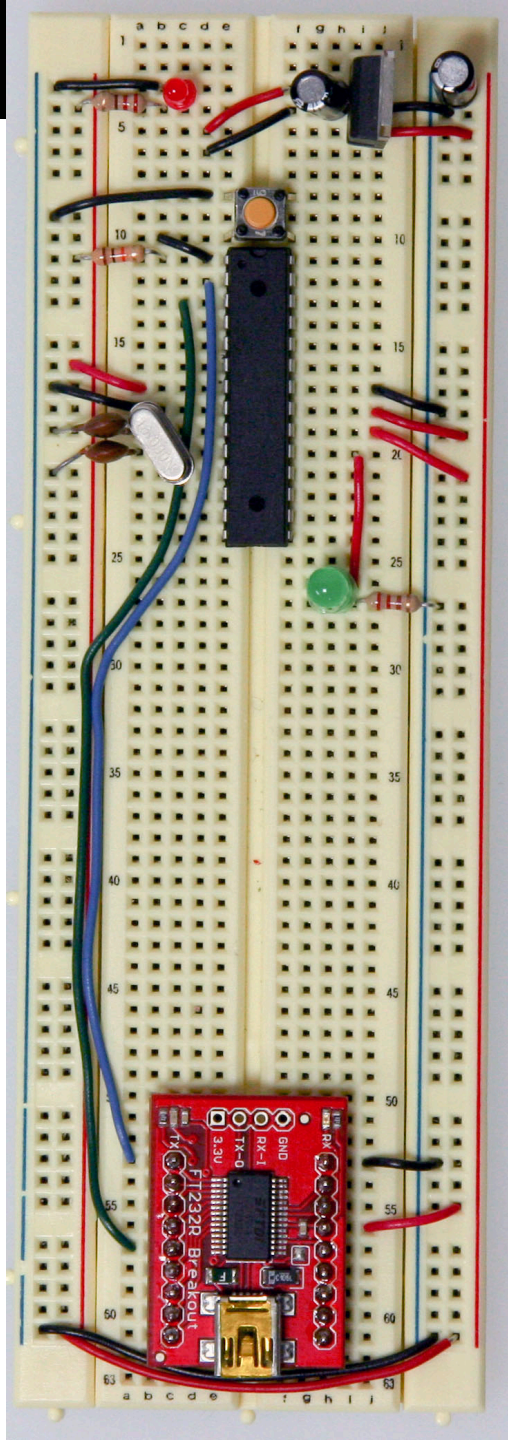
Bootloader

An Arduino is a electronic board with a microcontroller and expansion boards ready to use!

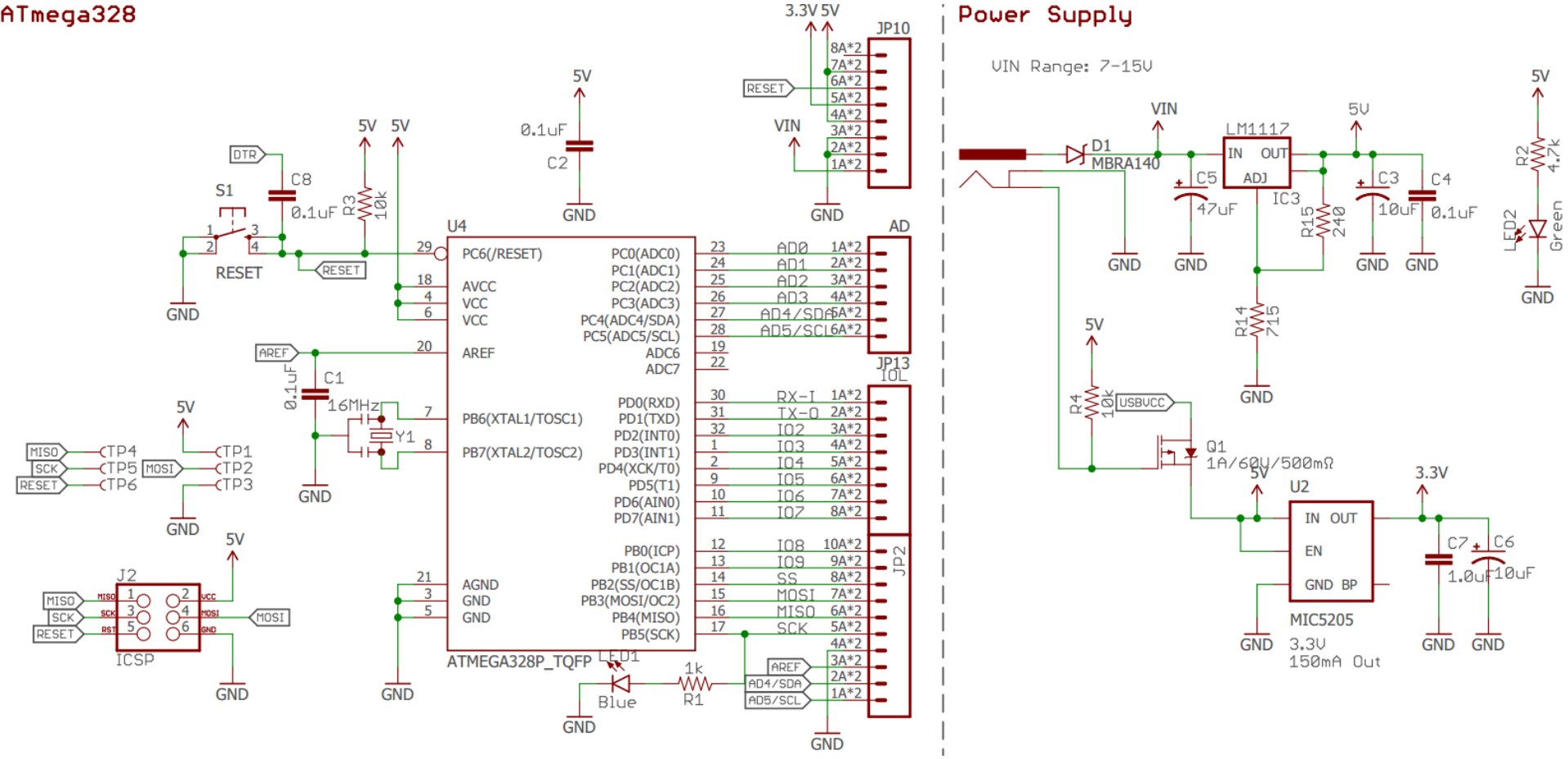
Building an Arduino on a Breadboard

https://www.arduino.cc/en/Main/Standalone
https://www.arduino.cc/en/Tutorial/ArduinoToBreadboard

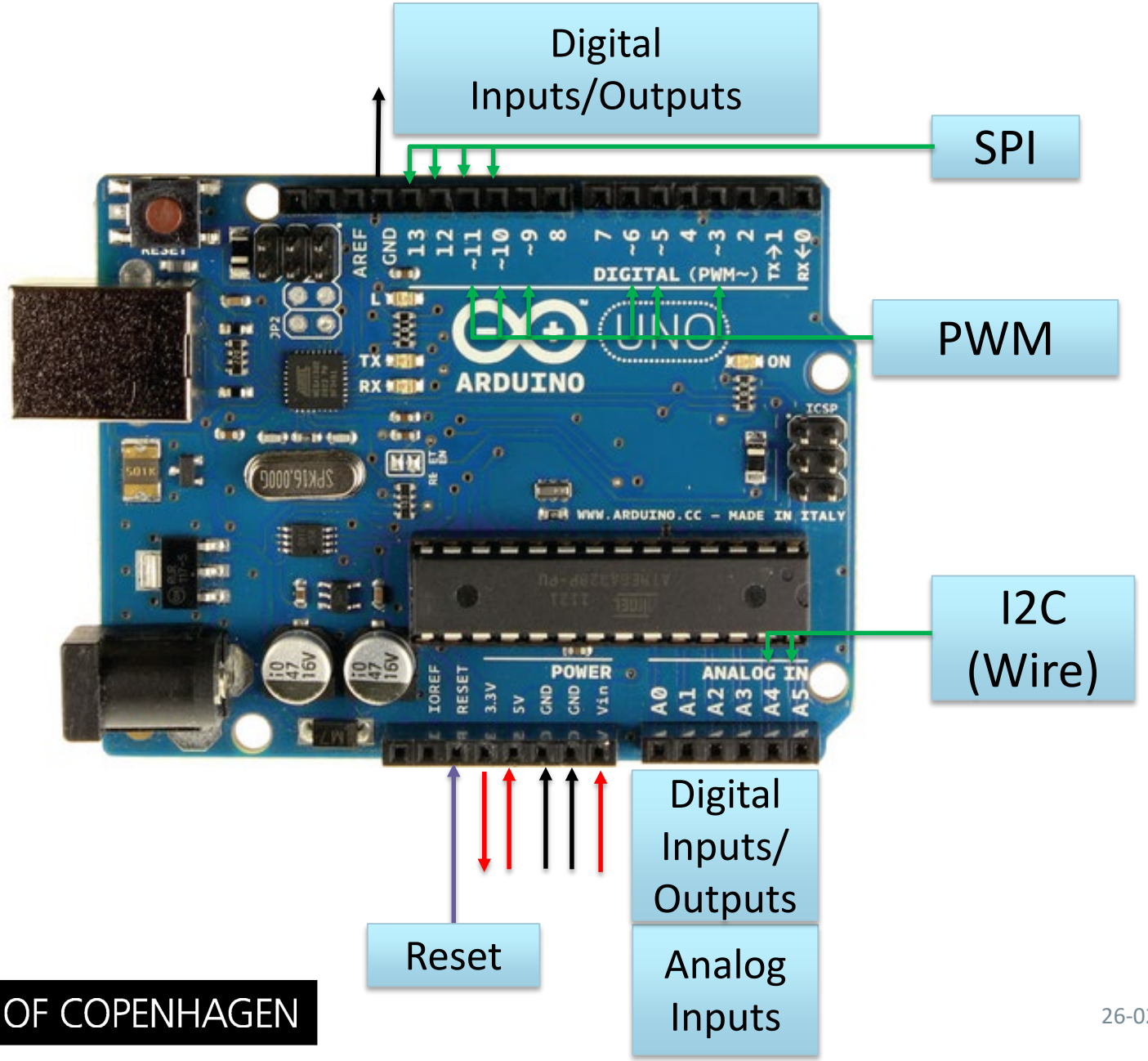
Microcontroller
Crystal + 2 capacitors (oscillator)
Reset button
Power led (red)
Led (green)
Power
USB - Serial communications



A real circuit – Redboard (Arduino Uno)



Arduino - Headers



Getting started – Blinking a led

Go to https://www.arduino.cc/

Download and install Arduino IDE

Read the documentation of your board

https://www.arduino.cc/en/Main/Products

https://www.sparkfun.com/products/13975

Try to blink the led of the board

Open examples/01.Basics/Blink

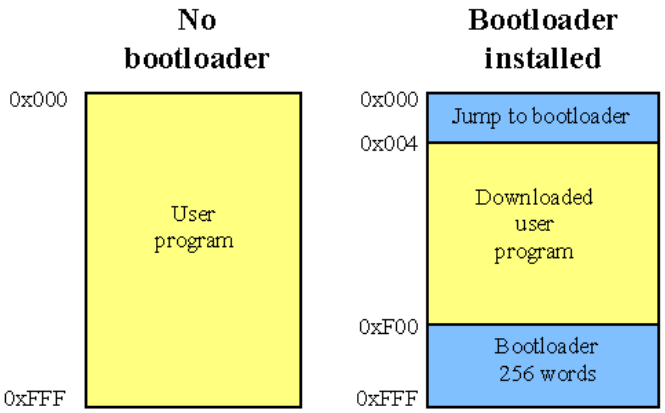
Select board and port

Verify (compile) and upload program

Advanced: can you do it without using delays ?

What have we done?

1. Build an electronic circuit
2. Write a program (C language)
3. Compile program (C -> assembler language)
4. Upload program in the microcontroller
   - Option a: Use a bootloader (friendly)
   - Option b: Use a programmer (expert)
5. The code runs in the microcontroller



https://learn.sparkfun.com/tutorials/installing-an-arduino-bootloader

Code explained

To Arduino…

2 main functions:

- setup – initialize hardware and variables
- loop – infinite loop, handle I/O

https://www.arduino.cc/en/Reference/HomePage

How to learn

- Program, program and program!

Arduino: Blink Without Delay

```
// the number of the LED pin
const int ledPin =  13;

// Variables will change:
int ledState = LOW;
long previousMillis = 0;

// the follow variables is a long because
// the  time, measured in miliseconds,
// will quickly become a bigger number
// than can be stored in an int.
 // interval at which to blink (milliseconds)
long interval = 1000;

void setup() {
  // set the digital pin as output:
  pinMode(ledPin, OUTPUT);
}
```

```
void loop()
{
  // here is where you'd put code that
needs to be running all the time.

  // check to see if it's time to blink the LED
  unsigned long currentMillis = millis();

  if(currentMillis - previousMillis > interval) {
    // save the last time you blinked the LED
    previousMillis = currentMillis;

    // if the LED is off turn it on and vice-versa:
    if (ledState == LOW)
      ledState = HIGH;
    else
      ledState = LOW;

    // set the LED with the ledState of the variable:
    digitalWrite(ledPin, ledState);
  }
}
```

Exercise #1 – Switches and leds

Make a circuit with 2 buttons and one led (example)

- if you press switch 1 -> led blinks
- if you press switch 2 -> led is turned off

Define the circuit

Build the circuit with the Arduino

Write the code

Tips

- Use a breadboard
- Check the guide in the kit
- Ask

Advanced: Can you do it with an empty loop function?

From voltage to registers

Digital

- 0V = 0
- 5V = 1

- And 1V, 2.5V or 4V??????

  Check datasheet

  1V = 0

  2.5V = 0/1

  4V = 1

Analog

- 0V = 0
- 5V = ?

- Check datasheet

  10bits ADC => 5V = 1023

- And 1V, 2.5V or 4V??????

  1V = (1023/5) = 205

  2,5V (1023/2) = 511

  4V = (1023*4/5) = 818

From registers to voltages

Digital

- 0 = 0V?? **(False!)**

To datasheet

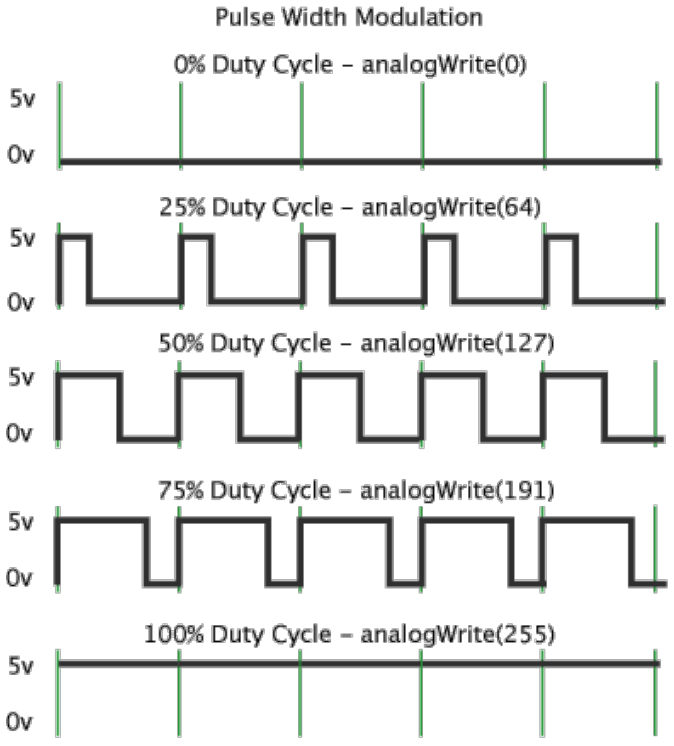
- 0 = <1V
- 1 = >4.1V

Analog

- There is not a DAC in ucontroller
- Solution: Use digital signal

  PWM (analogWrite)

  https://www.arduino.cc/en/Tutorial/PWM



Advanced- Interrupts

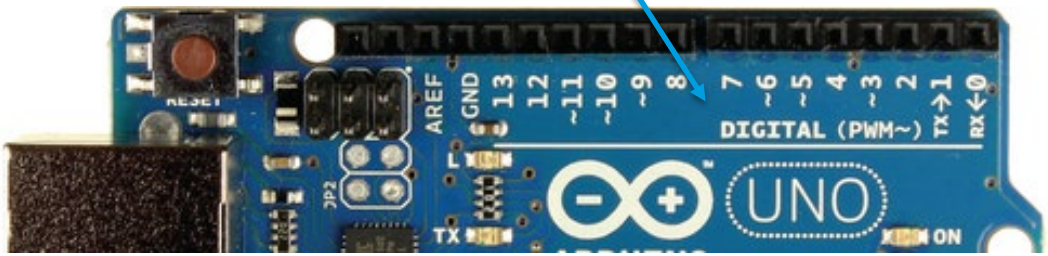
```
const byte ledPin = 13;
const byte interruptPin = 2;
volatile byte state = LOW;

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(interruptPin, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(interruptPin), blink, CHANGE);
}

void loop() {
}

void blink() {
  state = !state;
  digitalWrite(ledPin, state);
}
```

Is called when interrupt is triggered, Halts normal code, **Keep it short!**

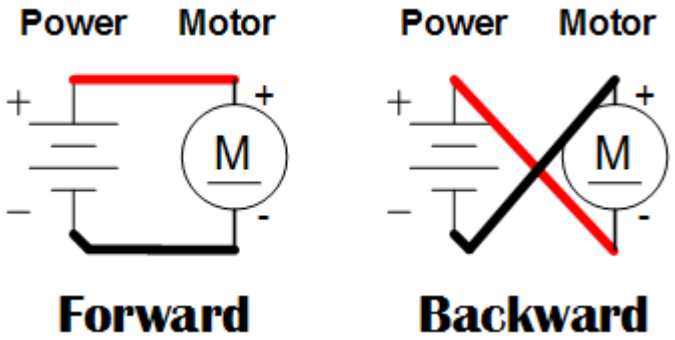
https://www.arduino.cc/en/Reference/AttachInterrupt
https://playground.arduino.cc/Code/Interrupts

Exercise #2 – Analog voltages

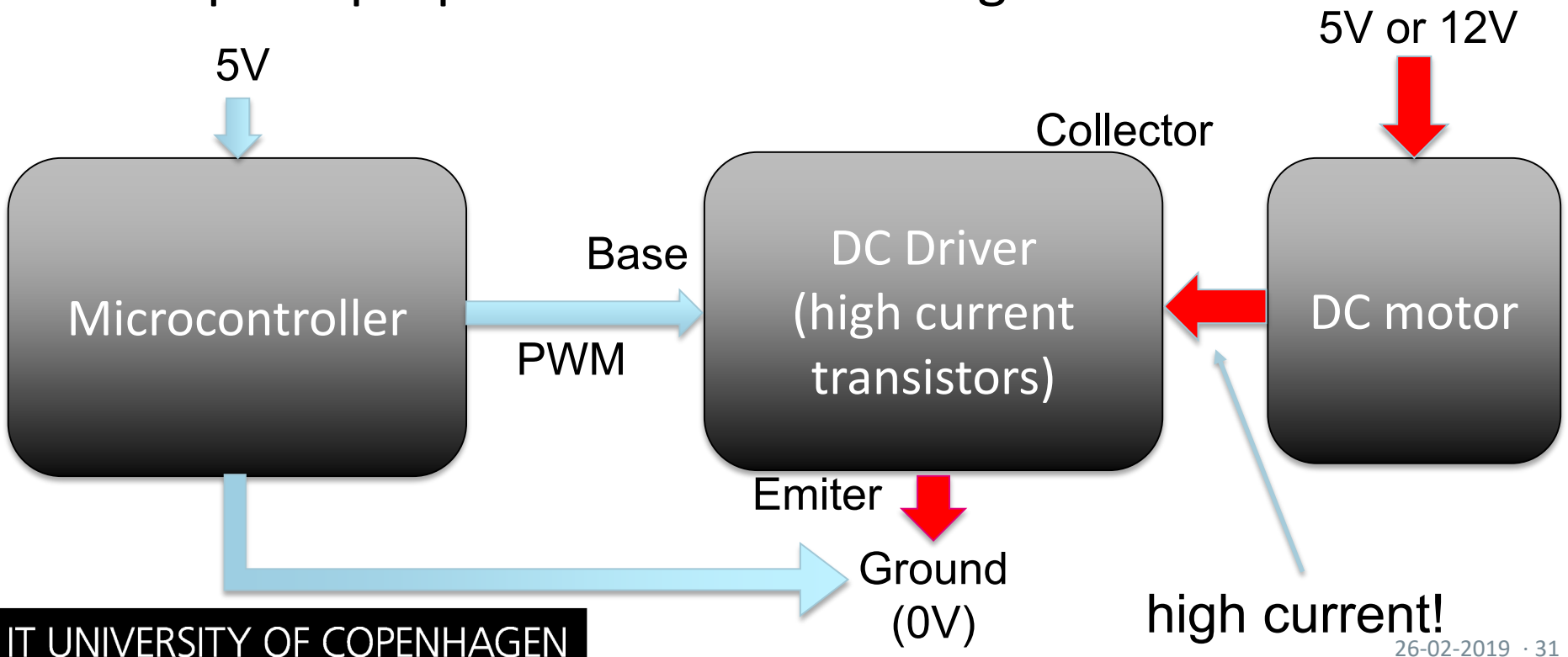
Make a circuit with 1 potentiometer and one led
    the potentiometer controls the brightness of the led
Define the circuit
Build the circuit with the Arduino
Write the code
Tips
    Use a analog pin for input voltage (potentiometer)
    Use a PWM pin to control led and analogWrite
    Check the pins with PWM hardware
    Check the guide in the kit
    Ask



DC motor - Control

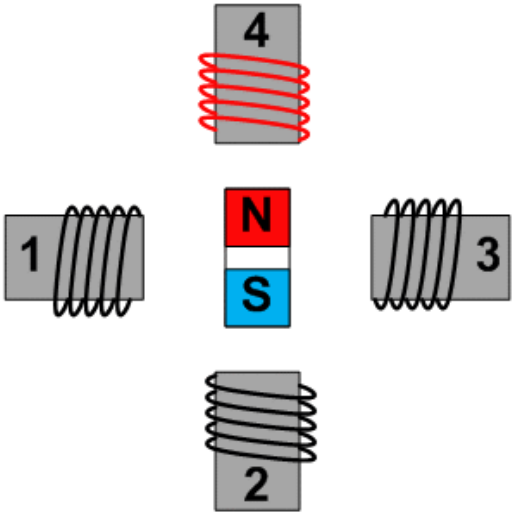


- Change direction
  - Reverse polarity
- Constant voltage
- Speed proportional to the voltage



Stepper motor

Full step control



4 steps/rev

Hybrid

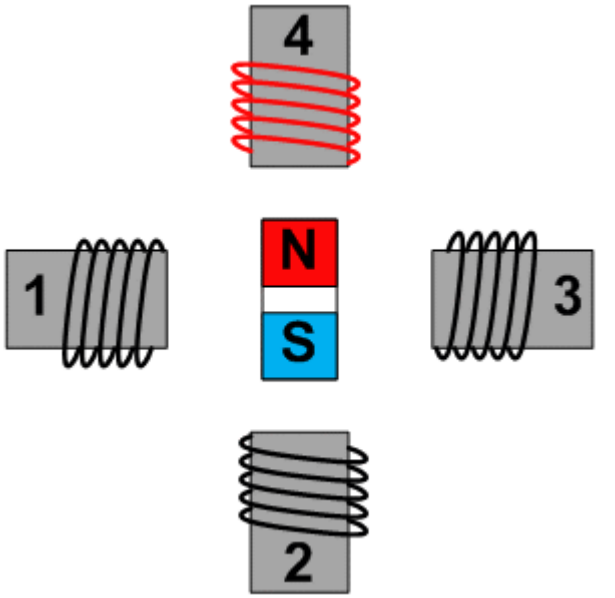
200 steps/rev

Permanent magnet

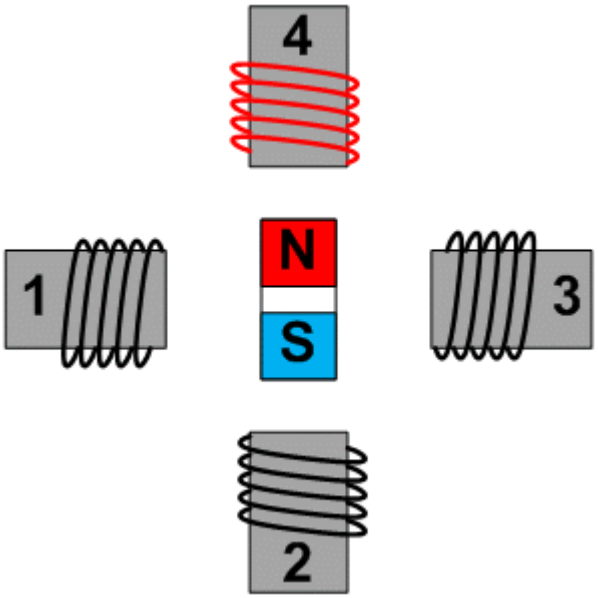
24 steps/rev

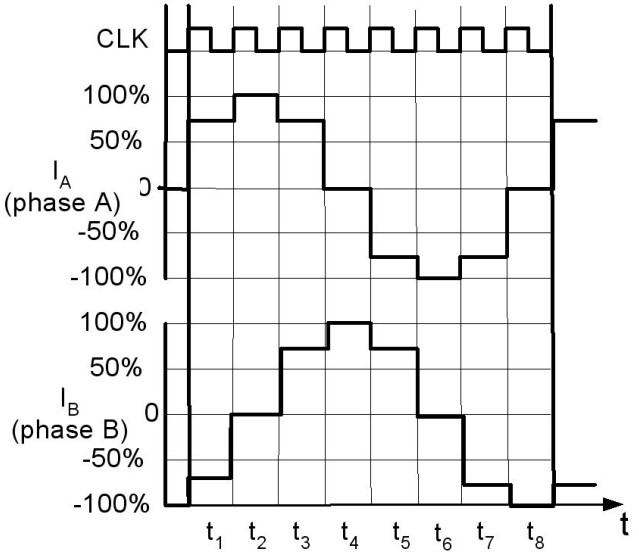
Stepper motor - microstepping

Full step



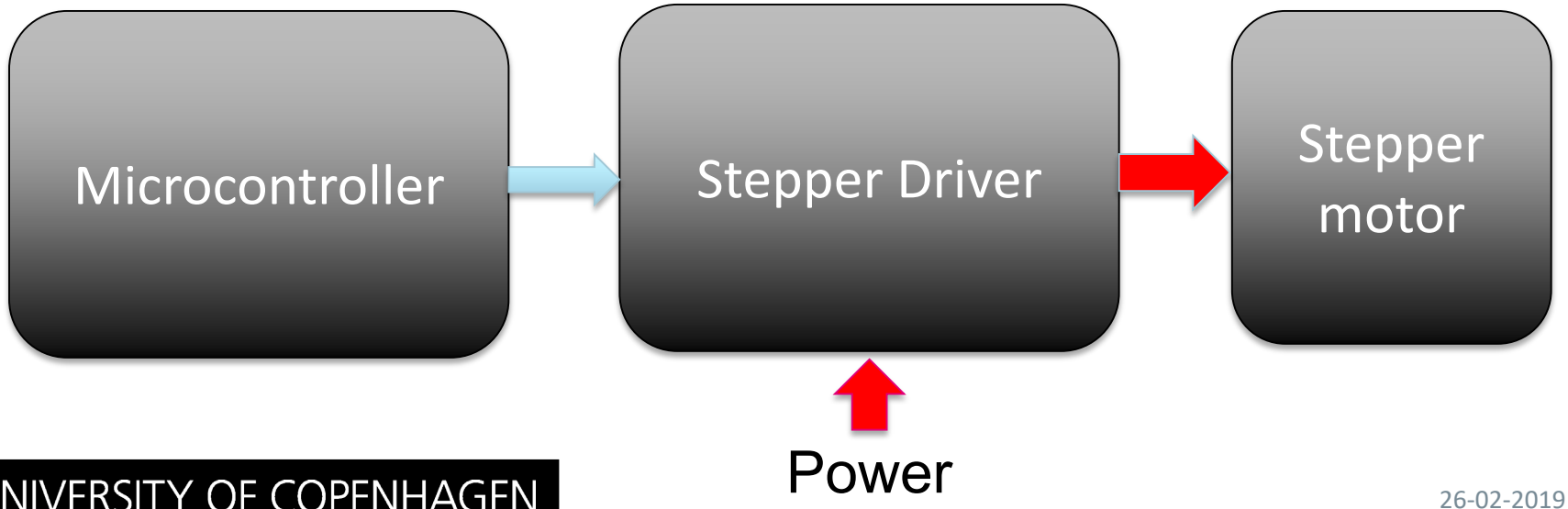
Half step



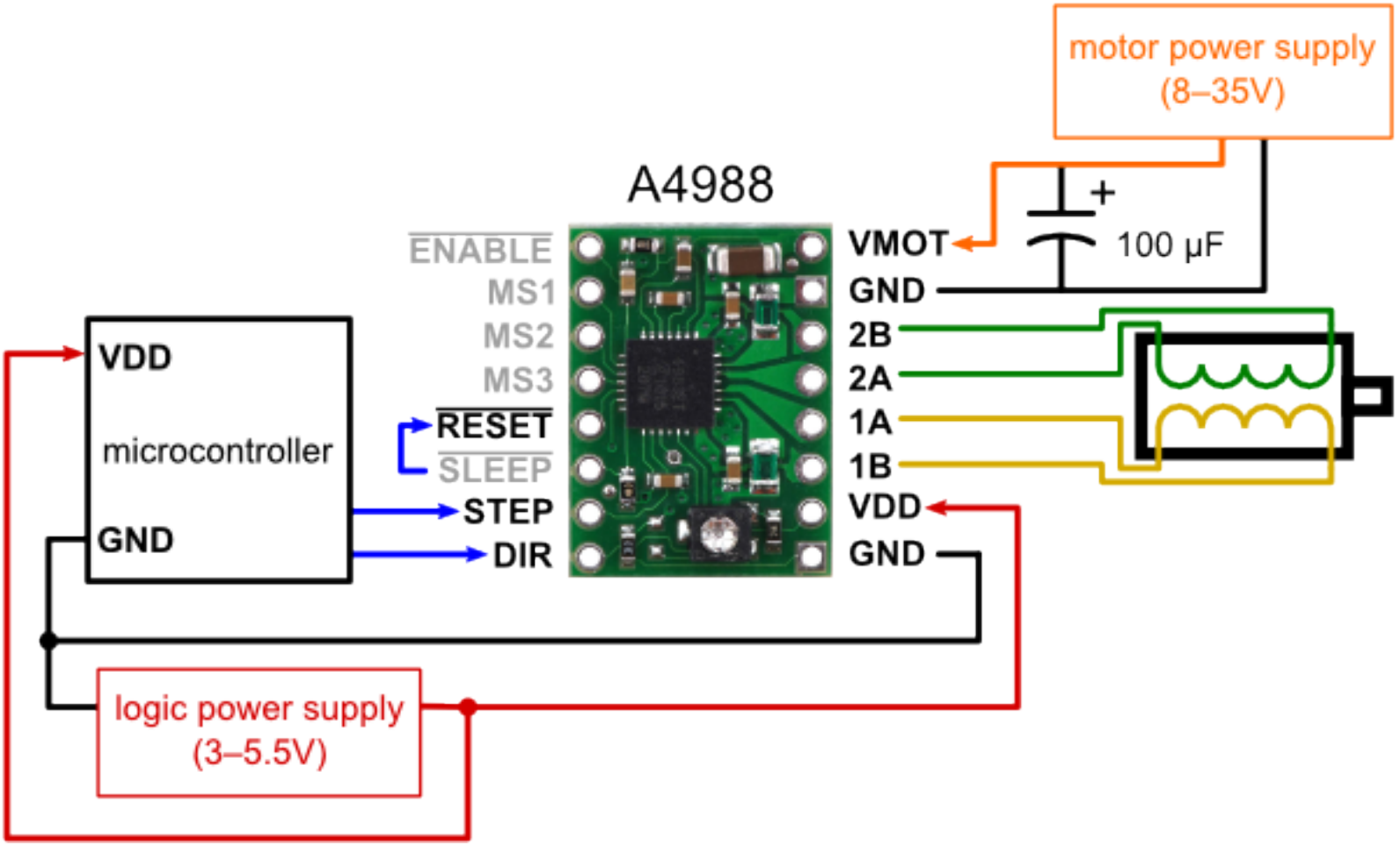


Stepper motor - Control

- Constant current for each step
- A step sequence generates the rotation
- Open loop control (if the force of the motor is enough!!)
- Microcontroller gives direction and step (pulses) signals
- Stepper driver handles the current through the coils



Pololu stepper drivers



https://www.pololu.com/product/1182

Stepper driver – Take care

Put a large (at least 47 µF) electrolytic capacitor across motor power (VMOT) and ground somewhere close to the board.

Electrolytic capacitors can explode if they are connected wrong: longer leg = positive; check "-" marking

Connecting or disconnecting a stepper motor while the driver is powered can destroy the driver.

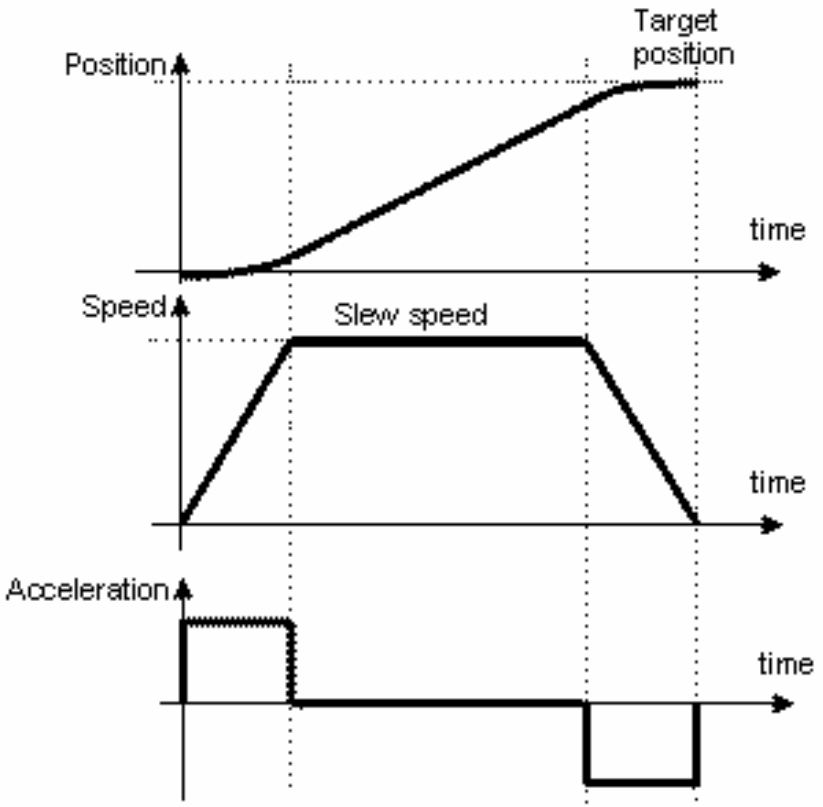
Set the current of the motor – adjust the potentiometer on the Pololu stepper driver

https://www.youtube.com/watch?v=89BHS9hfSUk

Exercise #3 - Moving stepper motors

Try to move the motor using switches or an arduino

- dir
  - 0V -> forward
  - 5V -> backward
- step
  - 1 pulse = 1 step
  - 1 pulse = 0V->5V->0V



Advanced:

Try to use a library

https://www.pjrc.com/teensy/td_libs_AccelStepper.html

https://github.com/laurb9/StepperDriver

Lessons learned

A microcontroller is a IC with CPU + memory (RAM & ROM) and peripherals

Handling sensors

- Digital: ON/OFF – 1/0 – 5V/0V
- Analog, ADC (10bits): 0-1023

Controlling motors

- too much current for ucontrollers
- drivers needed (high current transistors)
  - Types: DC drivers, stepper drivers

In case of doubt,

- check documentation (schematics and datasheets)

Mandatory assignment

Control your robot using an Arduino to follow a line or to avoid obstacles

DC motors:

- Discrete transistors
- Speed control with PWM

(https://www.lynda.com/Arduino-tutorials/Arduino-Pulse-Width-Modulation/372543-2.html?org=itu.dk)

- Power them with batteries

Stepper drivers:

- Use pololu stepper drivers
- Power them with a wall plug power supply

Sensors:

- Switches or/and IR
- Optional: encoder to control the speed

Danger!

Motors, electronic components (and students) can be damaged easily
Double check the wiring
Take care of different voltage levels
Ask before power motors and servos

If you are not sure, ASK!

Ready?

Let´s program!

Andrés Faíña, 4D26
anfv@itu.dk

Where to find electric components?

Components:

- Sensors
- Shields
- Breakout boards

Distributors:

- RS Components (*www.rs-components.com/*)
- Mouser (*www.mouser.dk/*)
- TME (https://www.tme.eu/en/)

DIY shops (with tutorials and libraries):

- Adafruit (*https://www.adafruit.com/*)
- Sparkfun (*https://www.sparkfun.com/*)

Breadboards

How to Use a Breadboard

https://learn.sparkfun.com/tutorials/how-to-use-a-breadboard