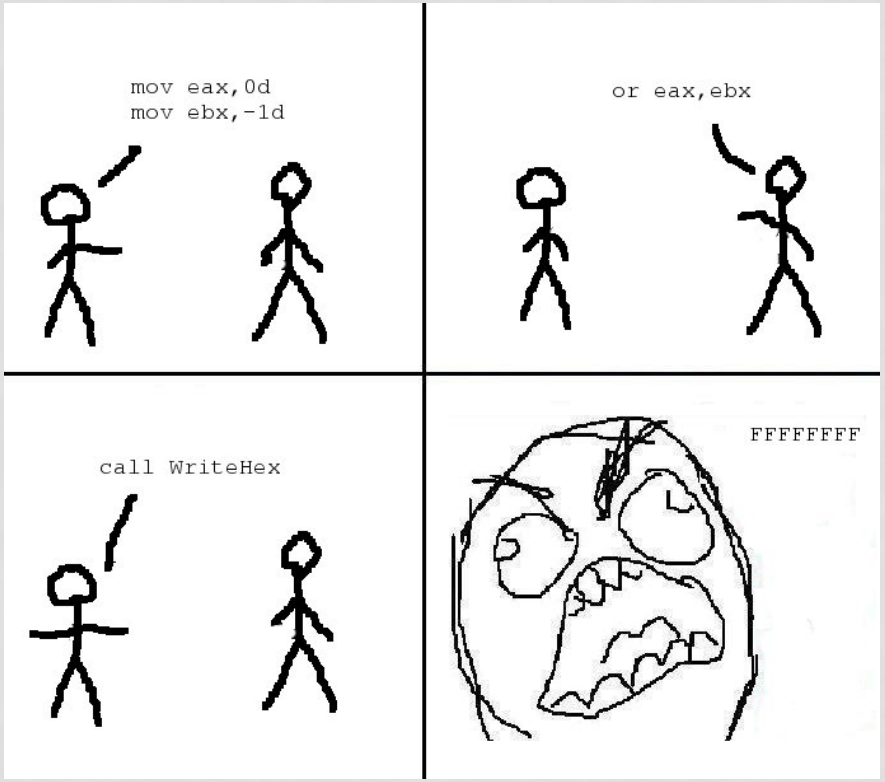


ASSEMBLY & C

WEEK 2-1

AGENDA

week	onderwerp	week	week
1	de structuur van AVR-assembly AVR instructies AVR registers en I/O ATmega memory map Atmel Studio AVR expressies en directives AVR addressing modes	3	de structuur van C-programma's ATMEL studio en AVR libc typen, constanten en operatoren AVR register access in C control statements functies & stackframe visibility scope arrays & strings struct & enum
2	flow of control spring instructies, control structuren Arduino UNO AVR studio stack & subroutines interrupts timer/counters switch bounce	4	interrupts in C TM1638 led&key UART PWM & ADC using a TTC-scheduler state diagram

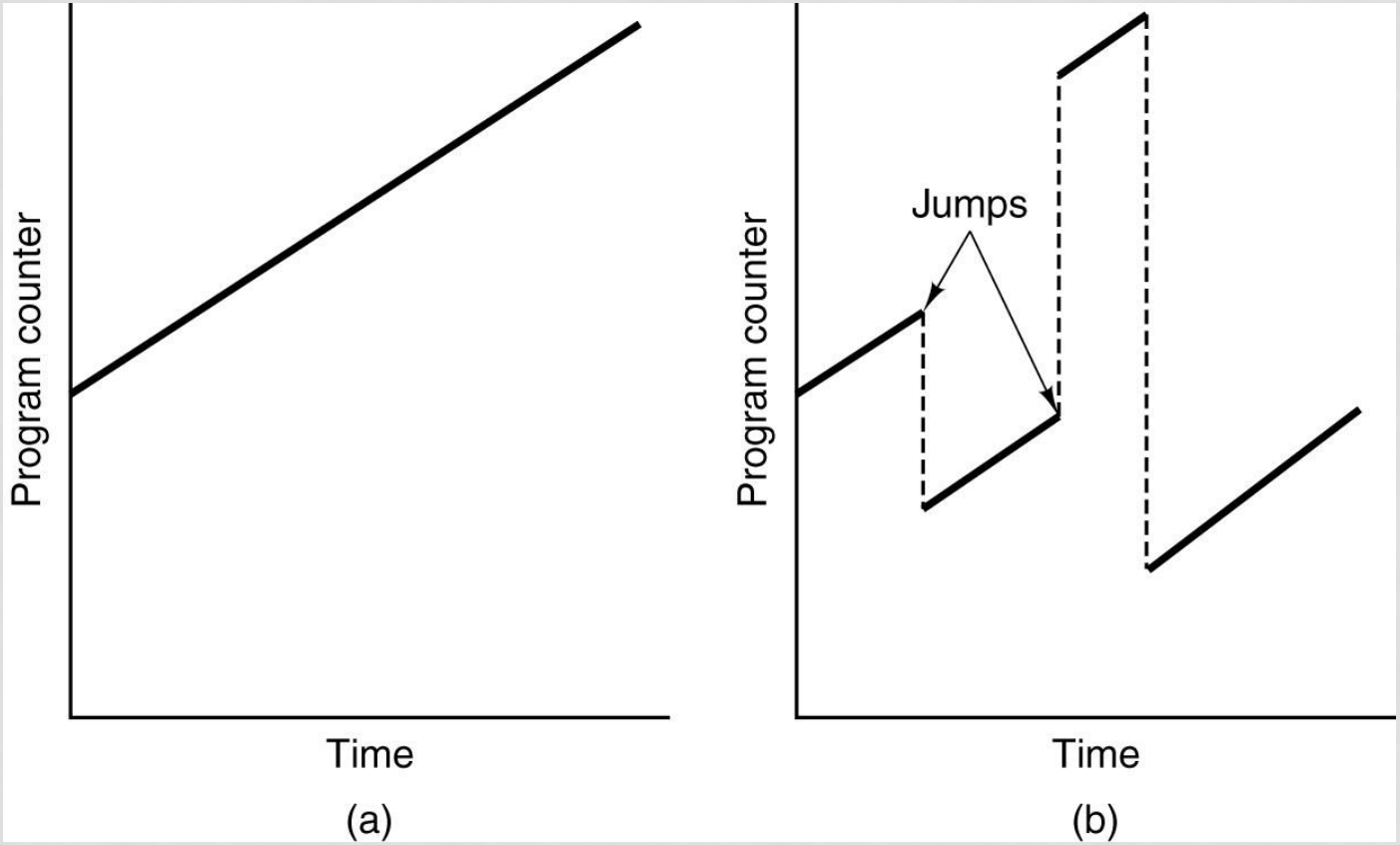
AGENDA

- **spring instructies**
- control structuren
- Arduino UNO

FLOW OF CONTROL

- sequentieel, of
- springen (AVR : jump, branch en skip)
- subroutines (AVR : call + ret)
- interrupts (AVR : jmp + reti)

PROGRAM COUNTER



JUMP EN BRANCH INSTRUCTIES

- jump, branch en skip instructies worden gebruikt voor control structuren
  - if/then en lussen
- 2 typen :
- unconditional = jump-zonder-test
  - jump en call instructies
- conditional = test-en-jump
  - branch en skip instructies

JUMP : ABSOLUUT EN RELATIEF

- jmp k
  - PC ← k met 0 ≤ k < 4M (absoluut)
  - 32-bit instructie, 3 cycles
- rjmp k
  - PC ← PC + k + 1 met -2048 ≤ k <2048 (relatief)
  - 16-bit instructie, 2 cycles
- zelfde bij call en rcall

RJMP

```
  cpi r16, 11    ;compare r16 to 11
  brne error     ;branch if r16 != 11
  rjmp ok        ;PC = PC + 3
error:
  add r16, r17
  inc r16
ok:
  nop
```

BRANCH & SKIP

- alle branch instructies kijken naar een of meerdere bits in het Status Register
  - zie AVR instruction set : conditional branch summary
  - na een compare, een subtract of logische instructie
- skip instructies slaan volgende instructie over als testresultaat waar is

Conditional Branch Summary

Test	Boolean	Mnemonic	Complementary	Boolean	Mnemonic	Comment
Rd > Rr	Z•(N ⊕ V) = 0	BRLT(1)	Rd ≤ Rr	Z+(N ⊕ V) = 1	BRGE*	Signed
Rd ≥ Rr	(N ⊕ V) = 0	BRGE	Rd < Rr	(N ⊕ V) = 1	BRLT	Signed
Rd = Rr	Z = 1	BREQ	Rd ≠ Rr	Z = 0	BRNE	Signed
Rd ≤ Rr	Z+(N ⊕ V) = 1	BRGE(1)	Rd > Rr	Z•(N ⊕ V) = 0	BRLT*	Signed
Rd < Rr	(N ⊕ V) = 1	BRLT	Rd ≥ Rr	(N ⊕ V) = 0	BRGE	Signed
Rd > Rr	C + Z = 0	BRLO(1)	Rd ≤ Rr	C + Z = 1	BRSH*	Unsigned
Rd ≥ Rr	C = 0	BRSH/BRCC	Rd < Rr	C = 1	BRLO/BRCS	Unsigned
Rd = Rr	Z = 1	BREQ	Rd ≠ Rr	Z = 0	BRNE	Unsigned
Rd ≤ Rr	C + Z = 1	BRSH(1)	Rd > Rr	C + Z = 0	BRLO*	Unsigned
Rd < Rr	C = 1	BRLO/BRCS	Rd ≥ Rr	C = 0	BRSH/BRCC	Unsigned
Carry	C = 1	BRCS	No carry	C = 0	BRCC	Simple
Negative	N = 1	BRMI	Positive	N = 0	BRPL	Simple
Overflow	V = 1	BRVS	No overflow	V = 0	BRVC	Simple
Zero	Z = 1	BREQ	Not zero	Z = 0	BRNE	Simple

Note: 1. Interchange Rd and Rr in the operation before the test, i.e., CP Rd,Rr → CP Rr,Rd

VOORBEELD BRANCH

- brne & breq zijn complementair :
  - brne: branch if not equal (Z=0)
  - breq: branch if equal (Z=1)
- let op signed vs. unsigned
  - is `10000000 > 00000000` ?
- voorbeeld : BRLO na CPI
  - BRLO : jump if C = 1
  - CPI Rd, K : C = 1 if K > Rd

Example:

```
        eor   r19,r19     ; Clear r19
loop:   inc   r19         ; Increase r19
        ...
        cpi   r19,$10     ; Compare r19 with $10
        brlo  loop        ; Branch if r19 < $10 (unsigned)
        nop               ; Exit from loop (do nothing)
```

BRANCH

```
cpi r20, 16   ; compare
breq label1   ; branch if r20=16
brlo label2   ; branch if r20<16
```

```
dec r16       ; r16--
brne label1   ; branch if r16!=0
```

SKIP & BRANCH

```
sbrc r1,7   ; skip if bit in register is cleared
            ; skip volgende instructie als bit7=0 in r1
rcall MySub ; aanroep subroutine
```

```
  ldi r16,8
loop:
  inc r17
  dec r16
  brne loop ; while r16 > 0
  out PORTB, r17
```

AGENDA

- spring instructies
- **control structuren**
- Arduino UNO

SELECTION : IF-THEN-ELSE

```
if (a == 3){
  a++
} else {
  a--
}
```

```
   cpi a,3
   brne else
   inc a
   rjmp end_if
else:
   dec a
end_if:
```

SWITCH

```
switch (a){
case 1:
  x++;
  break;
case 2:
  x += y;
 break;
default:
  y = x;
}
```

```
  cpi a,1
  breq case_1
  cpi a,2
  breq case_2
  rjmp case_default
case_1:
  inc x
  rjmp case_end
case_2:
  add x, y
  rjmp case_end
case_default:
  mov y, x
case_end:
```

REPETITION : DO..WHILE

```
do {
   x--;
} while (x!=0)
```

```
loop:
   dec x
   brne loop
;loop exit
```

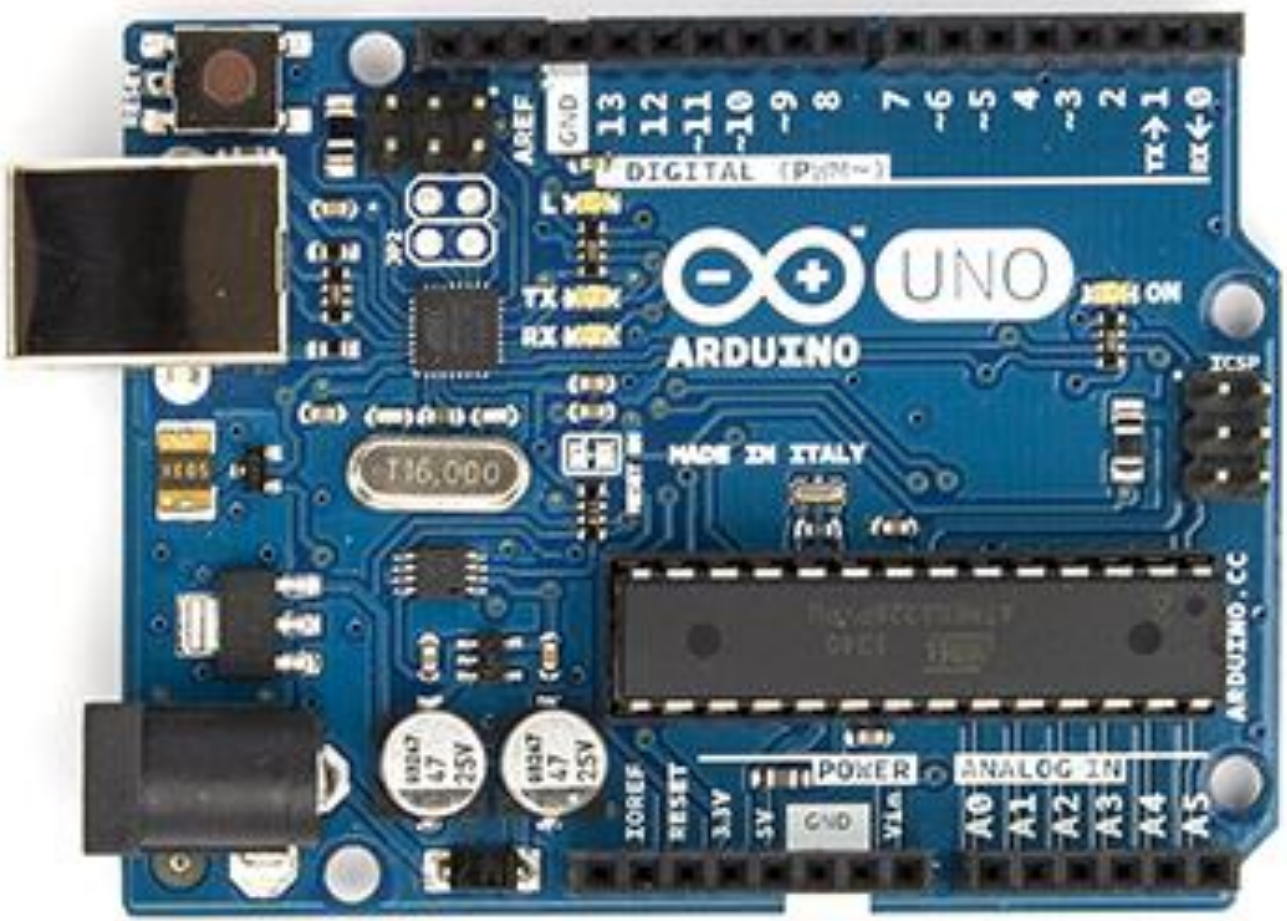
REPETITION : WHILE .. DO

```
while (x!=0) {
   x--;
}
```

```
loop:
   tst x
   breq end_while
   dec x
   rjmp loop
end_while:
;loop exit
```

AGENDA

- spring instructies
- control structuren
- **Arduino UNO**



Arduino UNO Rev. 3

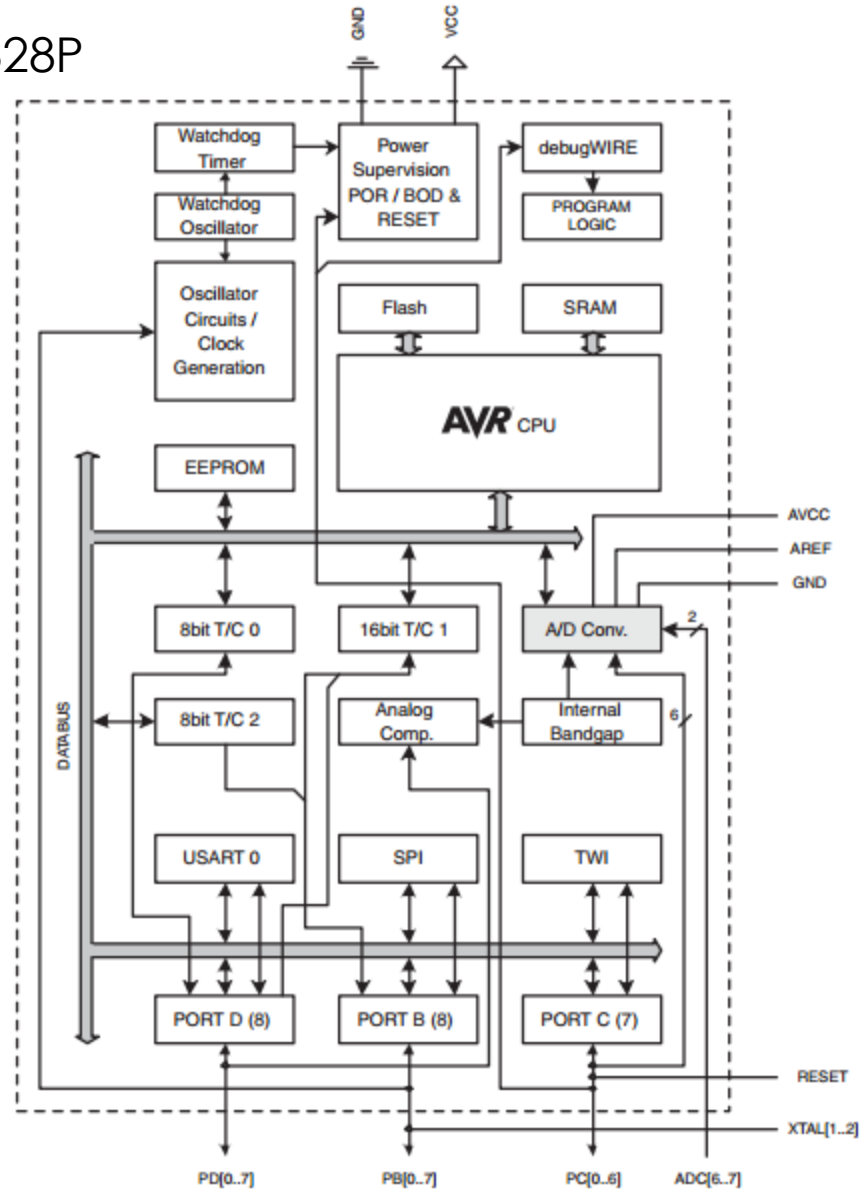
ARDUINO CORE TEAM

Table 1-1. *Atmel Microcontrollers Commonly Used in Arduino Boards*

Microcontroller	Flash memory (bytes)	SRAM (bytes)	EEPROM (bytes)	Clock speed	Digital I/O pins	Analog input pins	Voltage
Arduino Uno	32K	2K	1K	16Mhz	14	6	5V
Arduino Nano	32K	2K	1K	16Mhz	14	8	5V
Digispark Pro	16K	2K	1K	16Mhz	14	10	5V
RoboRED	32K	2K	1K	16Mhz	14	6	5 or 3.3V
ATmega1280	128K	8K	4K	16Mhz	54	16	5V
ATmega2560	256K	8K	4K	16Mhz	54	16	5V
Arduino Leonardo	32K	2.5K	1K	16Mhz	20	12	5V
Arduino Due	512K	96K	-	84Mhz	54	12/2[1]	3.3V
ChipKIT Max32[2]	512K	128K	-	80Mhz	83	16	3.3

ATmega328P

Figure 2-1. Block Diagram



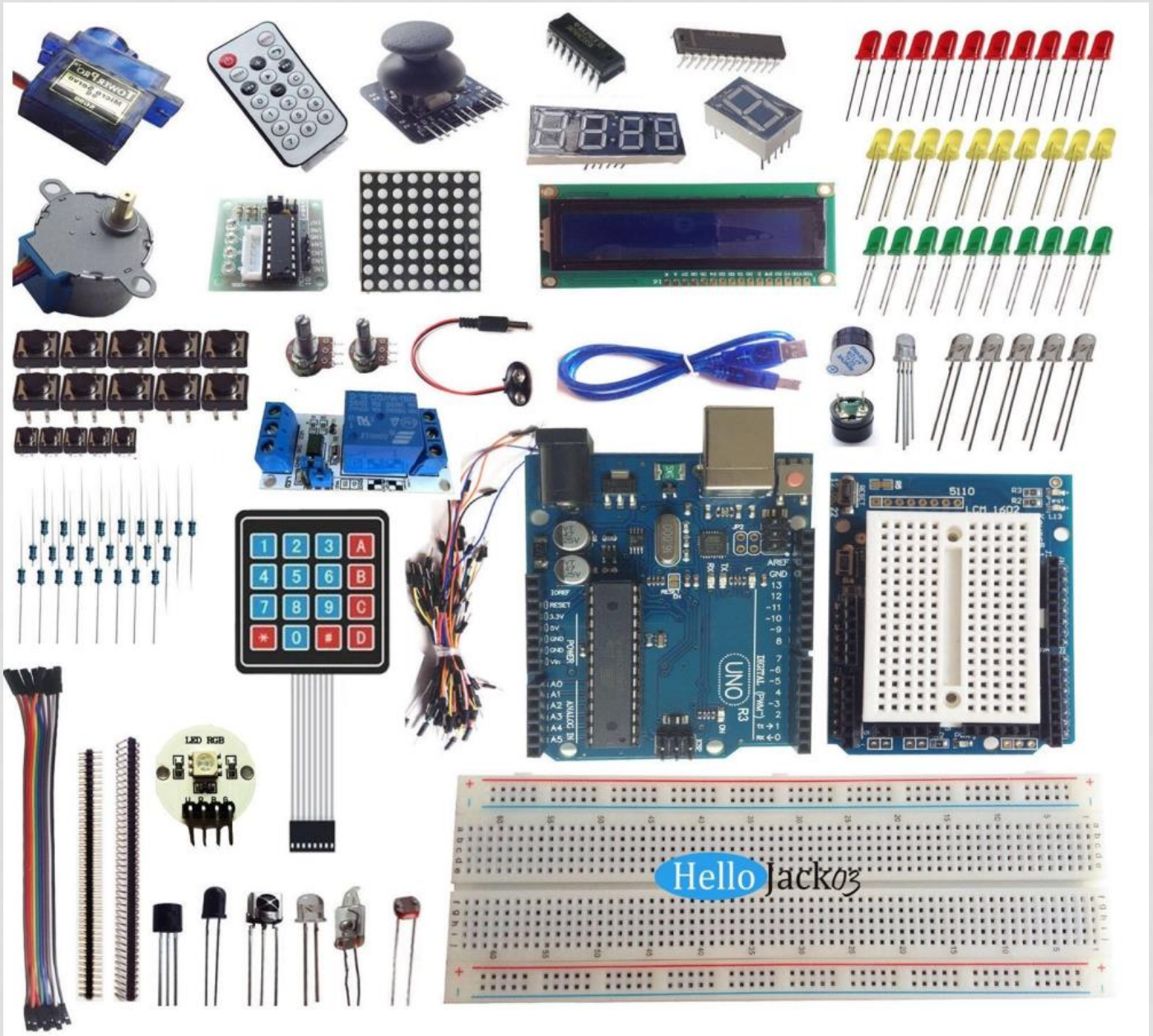
ARDUINO UNO

- Arduino : an open-source computer hardware and software project
- started in Ivrea, Italy
  - allowing Arduino boards to be manufactured by anyone
- inexpensive and simple tools for non-engineers to create digital projects
- based on Atmel AVR MCU's
- Arduino UNO uses ATmega328P on 16MHZ
- standard connectors allow for add-on modules called 'shields'
  - shield are connected directly or via I2C bus


Liechtenstein
Bern
Luzern
Zwitserland
Lausanne
Genève
Mâcon
Annecy
Chamonix
Zermatt
Lyon
Chambéry
Étienne
Grenoble
Valence
Gap
Avignon
Nice
Aix-en-Provence
Parc national de la Vanoise
Parc national du Mercantour
Parco Nazionale dello Stelvio
Bozen
Trento
Varese
Como
Bérgamo
Milaan
Verona
Padua
Pavia
Turijn
Asti
Parma
Modena
Bologna
Ravenna
Genua
Savona
La Spezia
Cesena
Pisa
Florence
Livorno
A31
A36
A6
A39
A40
A41
A43
A48
A49
A7
A51
E25
E43
E62
E35
E45
E70
E612
E717
E76

ARDUINO UNO

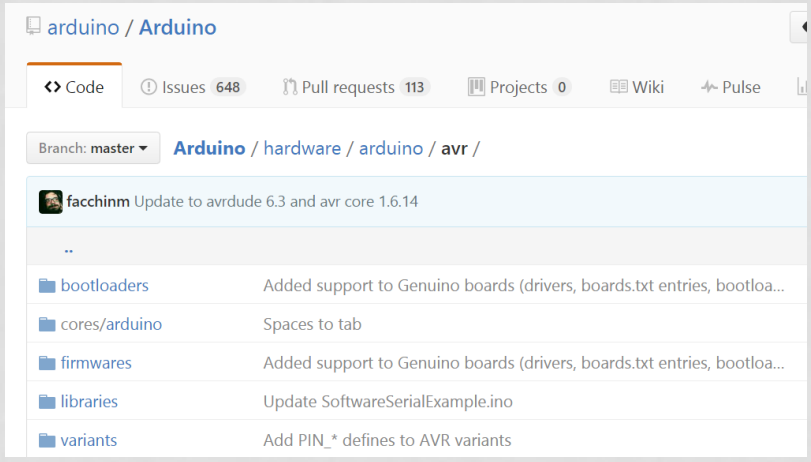
- includes a 5 V regulator and 16 MHz crystal oscillator
- pre-programmed with a boot loader called Optiboot (no external programmer required)
- can be programmed via ATMEL studio or Arduino IDE (both support C/C++)
- programmed via USB/RS232 using USB-to-serial firmware (and ATmega16U2 USB i/f)



http://stores.ebay.com/hellojack03

WIRE LIBRARY

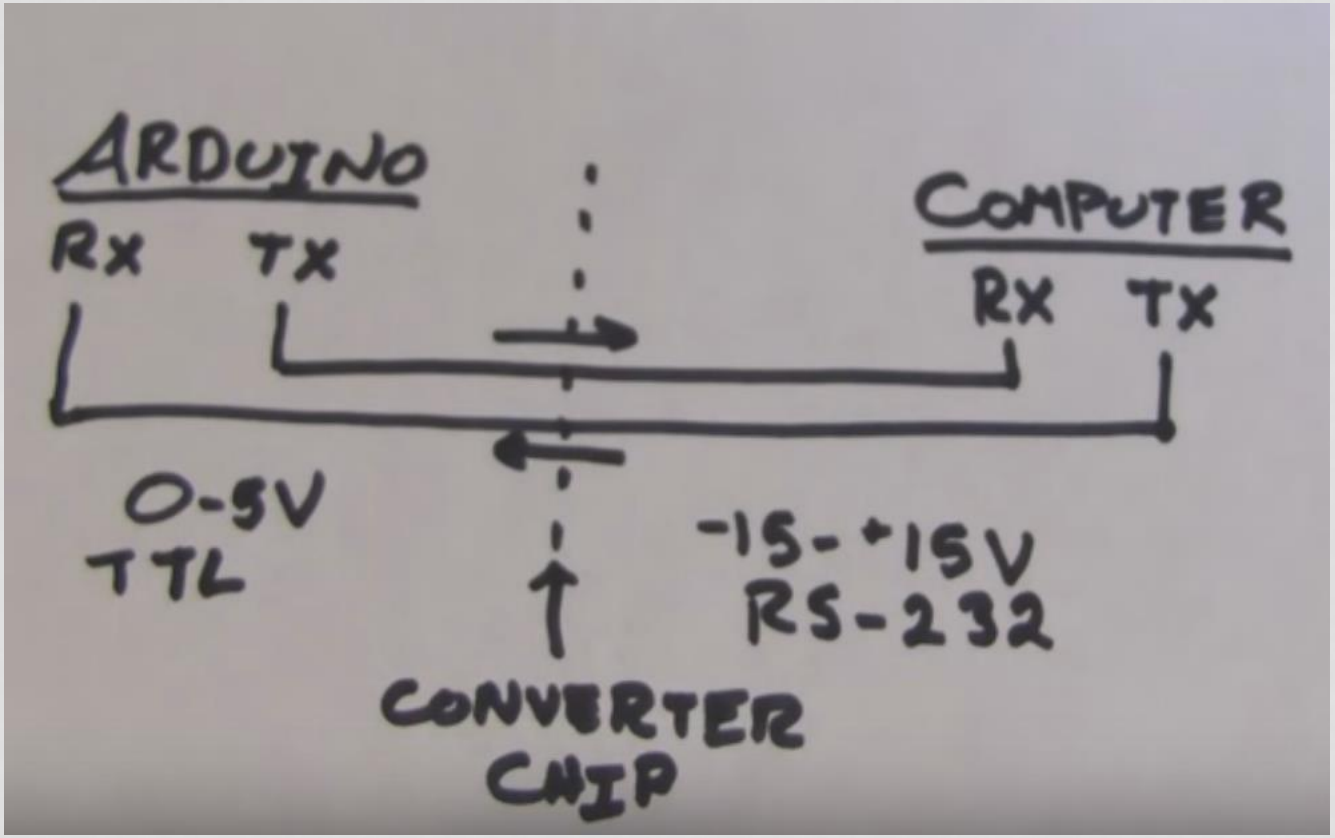
- we will **not** use Wire
- Arduino IDE includes a library called Wire which provides a HAL (Hardware Abstraction Layer)
  - Wiring is an open-source programming framework for microcontrollers
  - Wire library for Arduino is written in C++
- https://github.com/arduino



WIRING : EXAMPLE

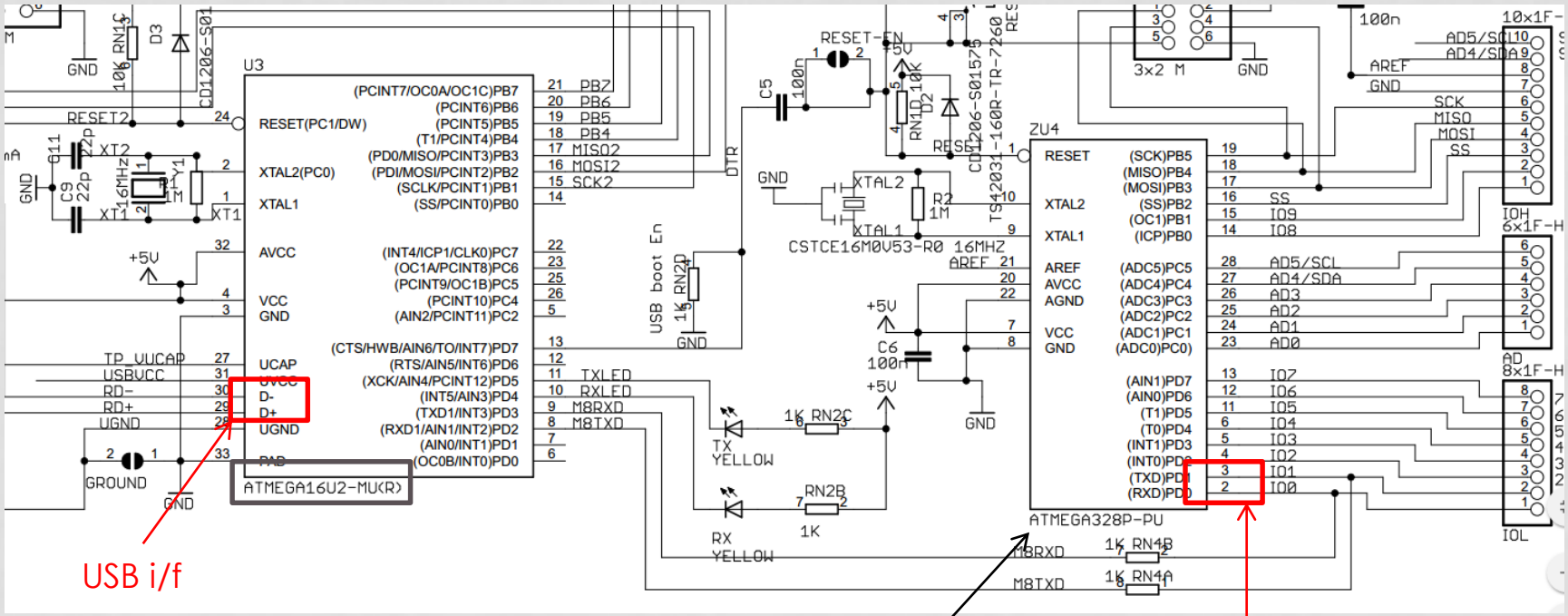
```
1  void digitalWrite(uint8_t pin, uint8_t val)
2  {
3          uint8_t timer = digitalPinToTimer(pin);
4          uint8_t bit = digitalPinToBitMask(pin);
5          uint8_t port = digitalPinToPort(pin);
6          volatile uint8_t *out;
7
8          if (port == NOT_A_PIN) return;
9
10         // If the pin that support PWM output, we need to turn it off
11         // before doing a digital write.
12         if (timer != NOT_ON_TIMER) turnOffPWM(timer);
13
14         out = portOutputRegister(port);
15
16         uint8_t oldSREG = SREG;
17         cli();
18
19         if (val == LOW) {
20                 *out &= ~bit;
21         } else {
22                 *out |= bit;
23         }
24
25         SREG = oldSREG;
26 }
```

USB/RS232

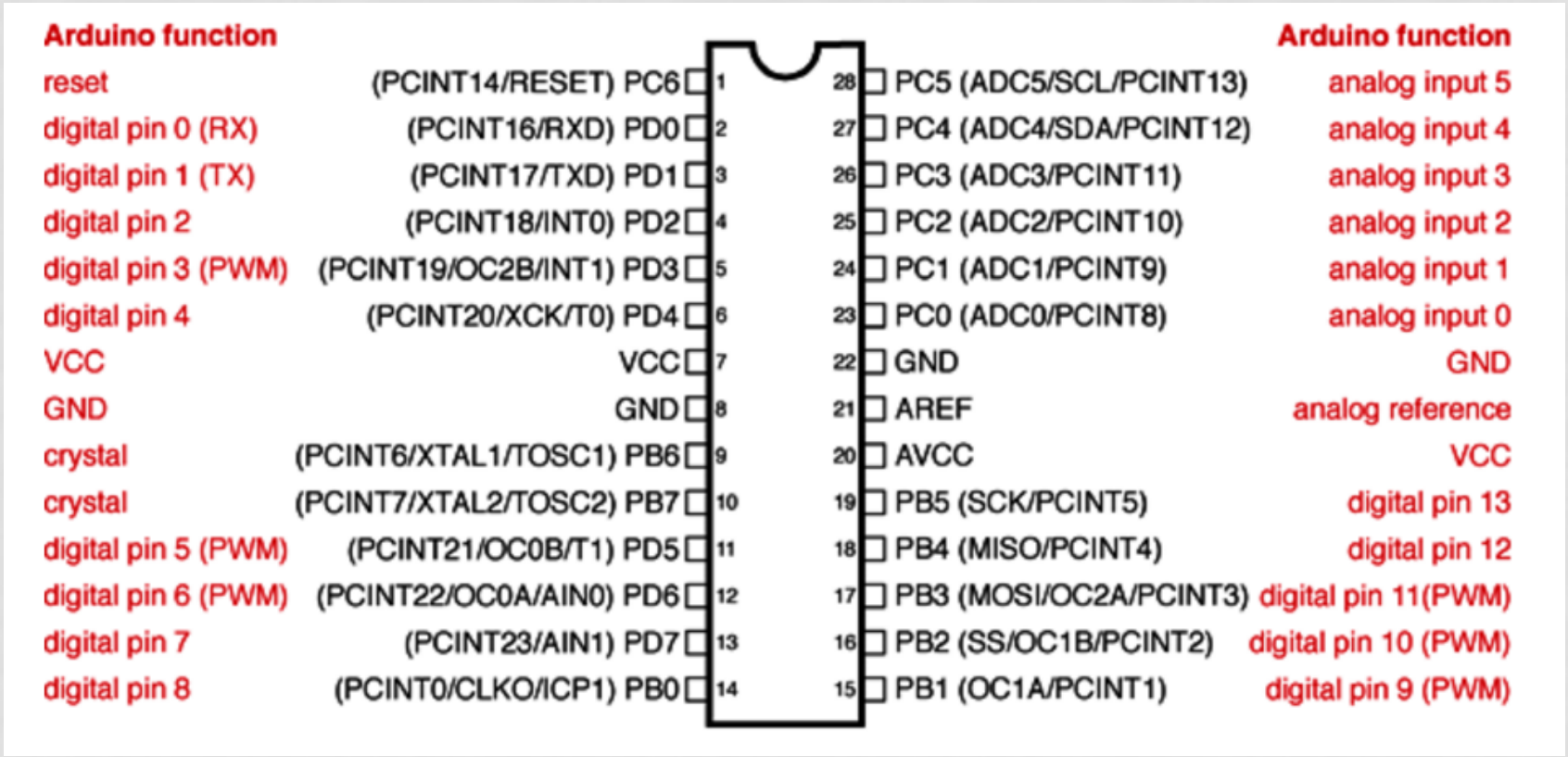

ARDUINO
RX
TX
COMPUTER
RX
TX
0-5V
TTL
-15-+15V
RS-232
CONVERTER
CHIP

USB/RS232

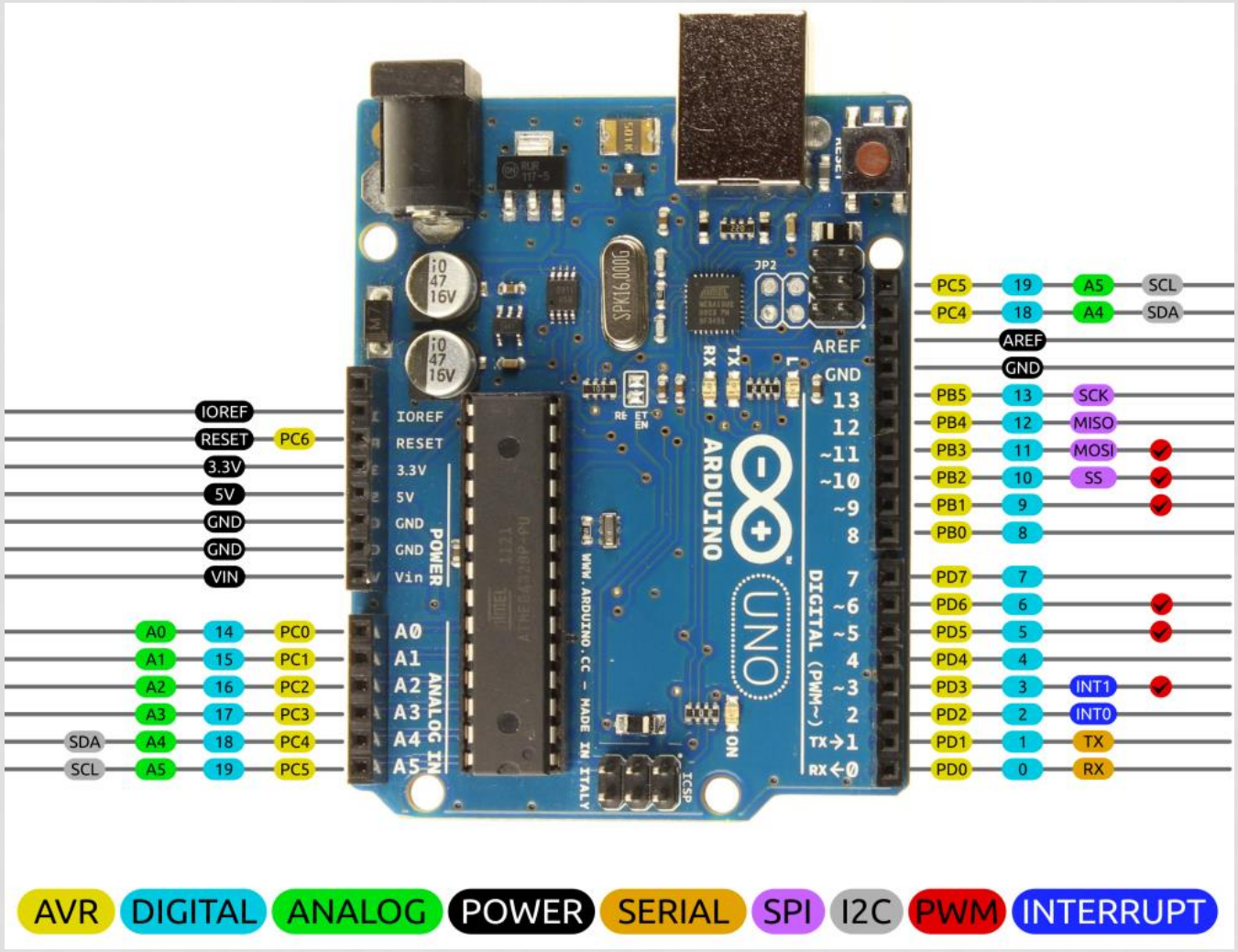
- the ATmega16U2 chip acts as a bridge between the computer's USB port and Atmega serial port



PIN LAYOUT


Arduino function
reset
digital pin 0 (RX)
digital pin 1 (TX)
digital pin 2
digital pin 3 (PWM)
digital pin 4
VCC
GND
crystal
crystal
digital pin 5 (PWM)
digital pin 6 (PWM)
digital pin 7
digital pin 8
(PCINT14/RESET) PC6 1
(PCINT16/RXD) PD0 2
(PCINT17/TXD) PD1 3
(PCINT18/INT0) PD2 4
(PCINT19/OC2B/INT1) PD3 5
(PCINT20/XCK/T0) PD4 6
VCC 7
GND 8
(PCINT6/XTAL1/TOSC1) PB6 9
(PCINT7/XTAL2/TOSC2) PB7 10
(PCINT21/OC0B/T1) PD5 11
(PCINT22/OC0A/AIN0) PD6 12
(PCINT23/AIN1) PD7 13
(PCINT0/CLKO/ICP1) PB0 14
28 PC5 (ADC5/SCL/PCINT13)
27 PC4 (ADC4/SDA/PCINT12)
26 PC3 (ADC3/PCINT11)
25 PC2 (ADC2/PCINT10)
24 PC1 (ADC1/PCINT9)
23 PC0 (ADC0/PCINT8)
22 GND
21 AREF
20 AVCC
19 PB5 (SCK/PCINT5)
18 PB4 (MISO/PCINT4)
17 PB3 (MOSI/OC2A/PCINT3)
16 PB2 (SS/OC1B/PCINT2)
15 PB1 (OC1A/PCINT1)
Arduino function
analog input 5
analog input 4
analog input 3
analog input 2
analog input 1
analog input 0
GND
analog reference
VCC
digital pin 13
digital pin 12
digital pin 11(PWM)
digital pin 10 (PWM)
digital pin 9 (PWM)

PIN LAYOUT


IOREF
RESET
PC6
3.3V
5V
GND
GND
VIN
A0 14 PC0
A1 15 PC1
A2 16 PC2
A3 17 PC3
SDA A4 18 PC4
SCL A5 19 PC5
PC5 19 A5 SCL
PC4 18 A4 SDA
AREF
GND
PB5 13 SCK
PB4 12 MISO
PB3 11 MOSI
PB2 10 SS
PB1 9
PB0 8
PD7 7
PD6 6
PD5 5
PD4 4
PD3 3 INT1
PD2 2 INT0
PD1 1 TX
PD0 0 RX
ARDUINO
UNO
POWER
ANALOG IN
DIGITAL (PWM~)
AVR
DIGITAL
ANALOG
POWER
SERIAL
SPI
I2C
PWM
INTERRUPT

ARDUINO HEADER PINS

- digital 0 - 7 = Port D [0:7]
  - digital pins 0 and 1 are RX and TX for serial communication
- digital 8 - 13 = Port B [0:5]
  - pin 13 is connected to on-board LED
- analog A0 - A5 = Port C [0:5]
  - also connected to ADC-channel : can convert 0 - 5V to a number (10 bit)

- digital 3, 5, 6, 9, 10 and 11 can output PWM signals
- SCL/SDA : 2-wire Serial Bus