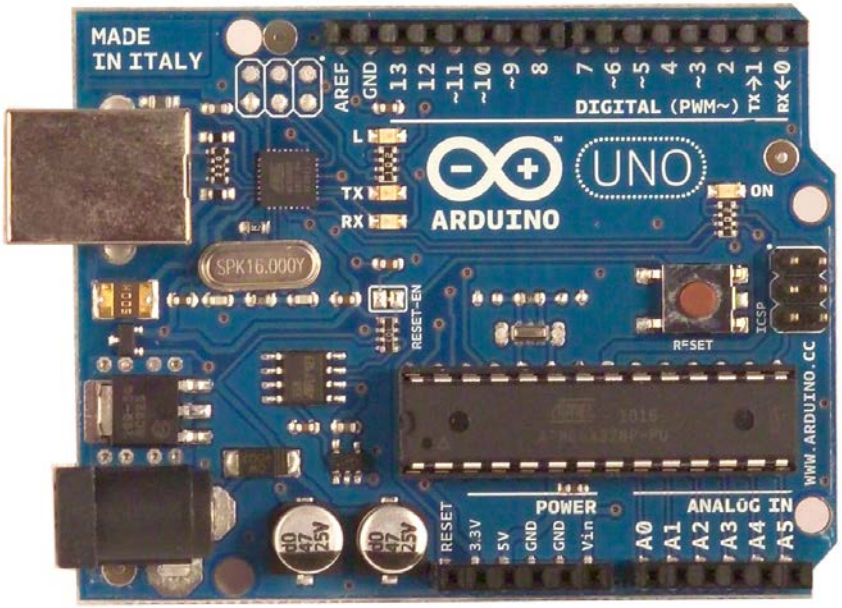
Workshop Arduino

Arduino Workshop

- Background on microcontrollers
- Writing a program
- Connecting hardware



Actions:

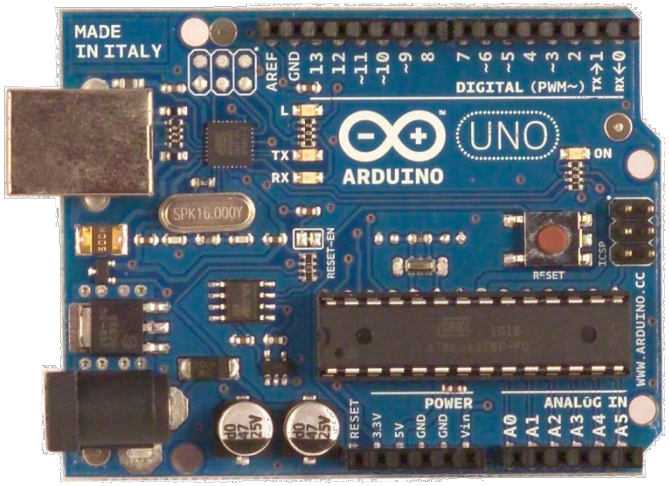
- Making a LED blink
- Read a button
- Read a sensor
- Send text to computer

“Microcontrollers are not scary”

Products with microcontrollers

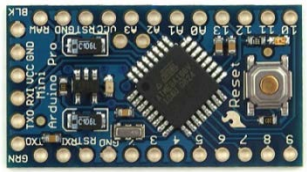
Why microcontrollers for ID?

- They enable you to turn your concepts into **working interactive prototypes**
- Can be used for many purposes in your career at low cost
- Arduino is supported by a huge community; you can build upon existing code and hardware designs (your own or by others)

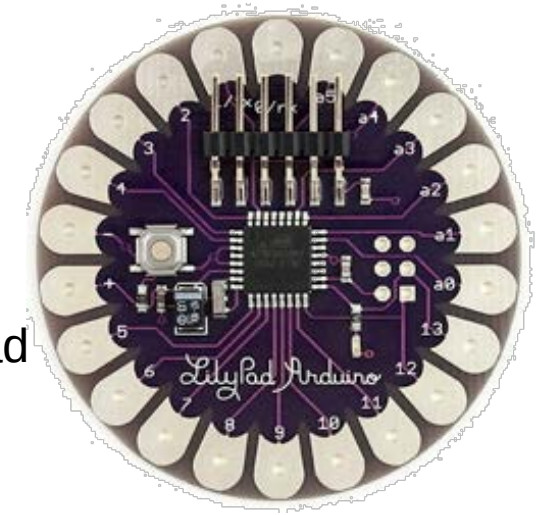


Arduino Uno

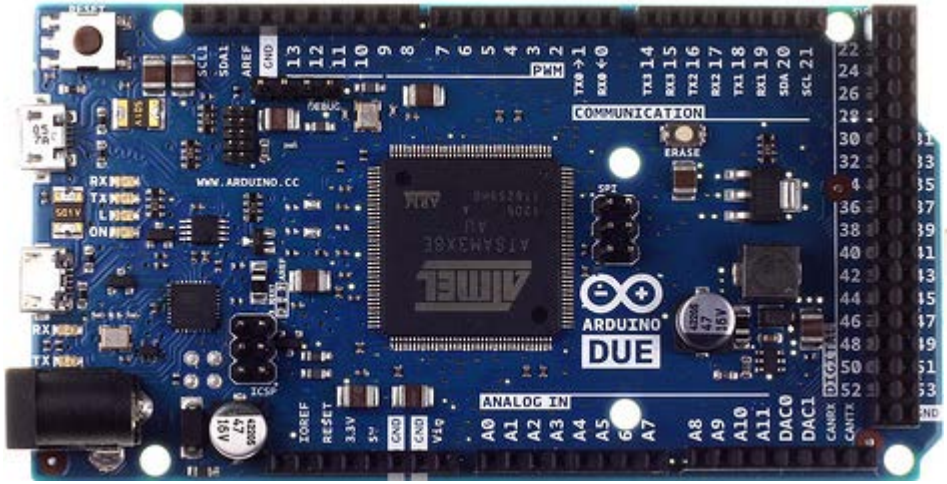
Arduino Nano



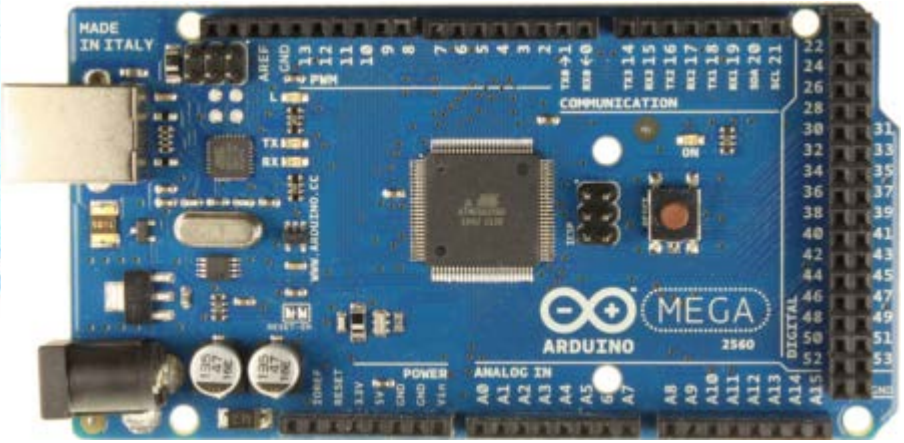
Arduino Pro Mini



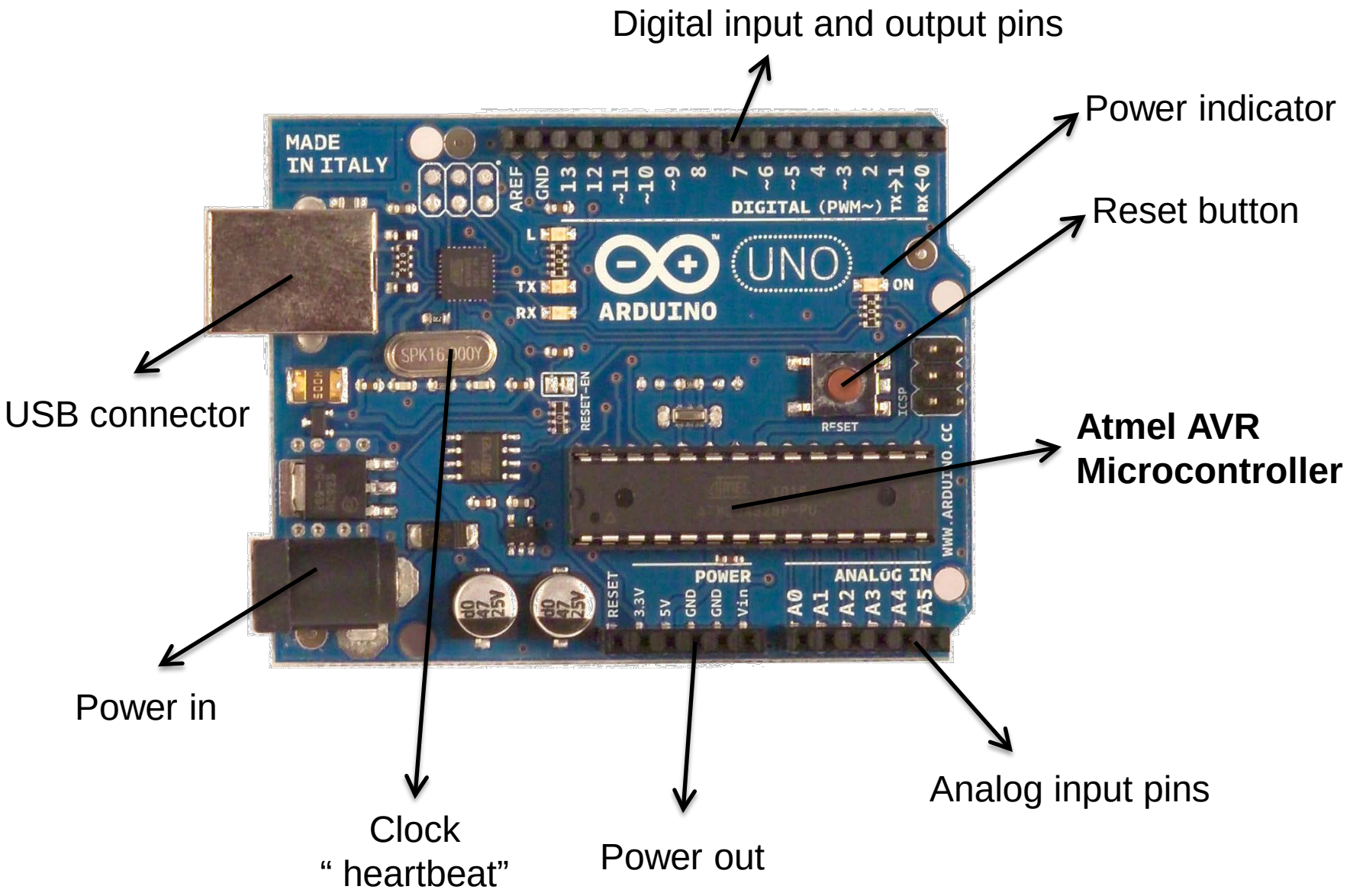
Arduino Lilypad



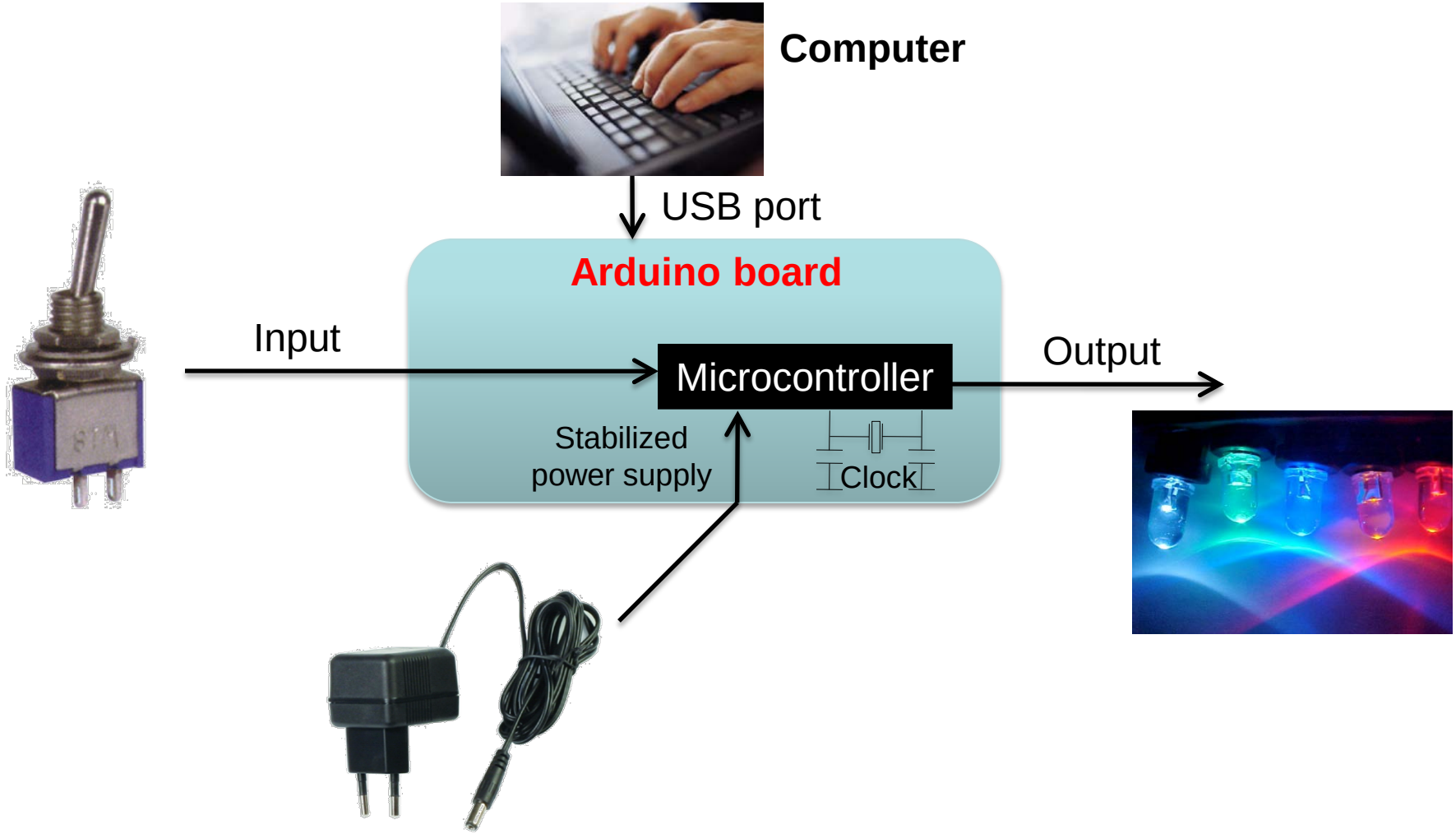
Arduino DUE (ARM based)



Arduino MEGA


Digital input and output pins
Power indicator
Reset button
USB connector
Atmel AVR Microcontroller
Power in
Clock
“ heartbeat”
Power out
Analog input pins
MADE IN ITALY
DIGITAL (PWM~)
UNO
ARDUINO
ON
RESET
POWER
ANALOG IN
WWW.ARDUINO.CC

The development system



A microcontroller . .

Is a processor with

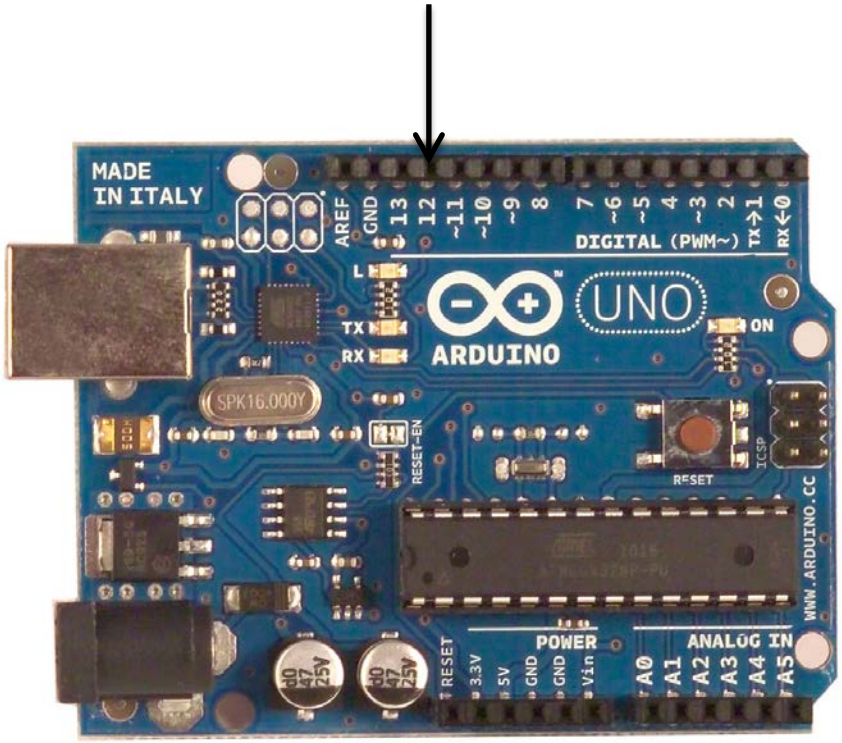
- Low power
- Low cost
- Dedicated for a single task
- On-board program memory
- On-board data memory
- I/O pins (digital and/or analog)
- (Timer/counter circuits)
- (Bus protocols: RS232, USB, I^2C, SPI, …))

Electronics: Digital Inputs

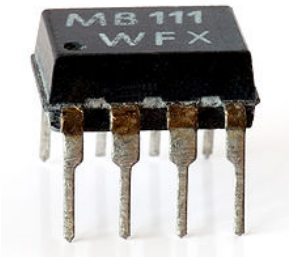
Digital input:

- 0 Volt: Low
- 5 Volt: High
- 2.5 Volt or floating?:

  Undefined !!



Electronics : Digital sensors

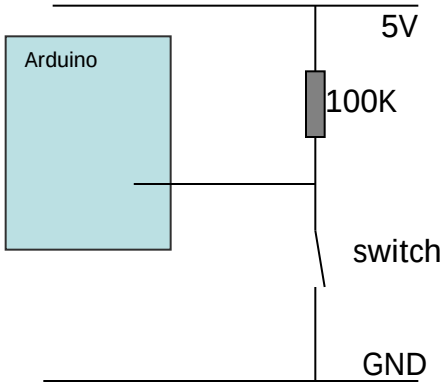


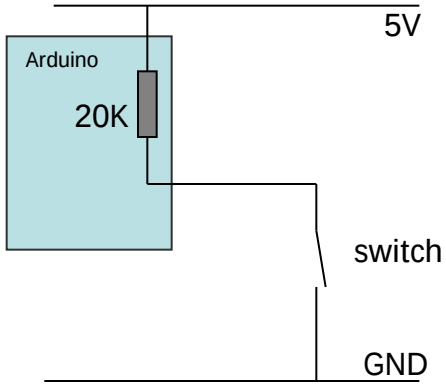
Electronics: Digital Inputs

External pull up
```
pinMode(12, INPUT);
digitalWrite(12, LOW);
int a = digitalRead(12);
```

Internal pull up
```
pinMode(12, INPUT);
digitalWrite(12, HIGH);
int a = digitalRead(12);
```

Internal pull up (new)
```
pinMode(12,INPUT_PULLUP);
int a = digitalRead(12);
```





Electronics: Analog Inputs

Analog input:

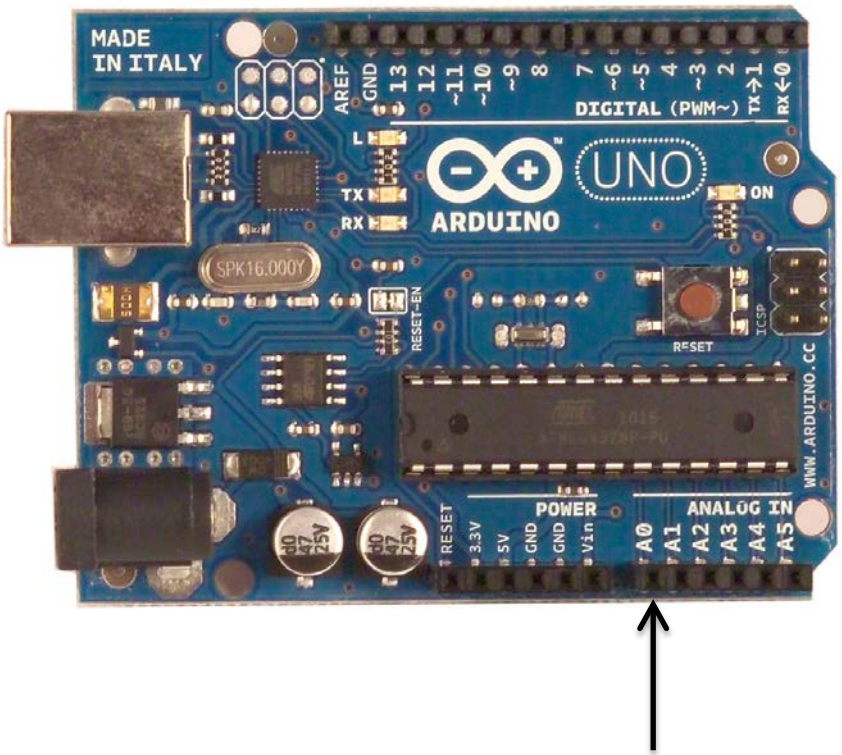
- Default:

  0..5 Volt to 0..1023

  0 => 0

  2,5 => 512

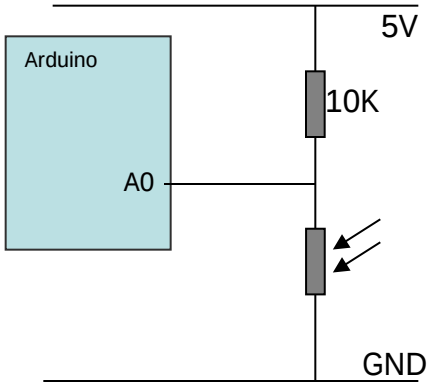
  5 => 1023



Electronics : Analog sensors

Electronics: Analog Inputs

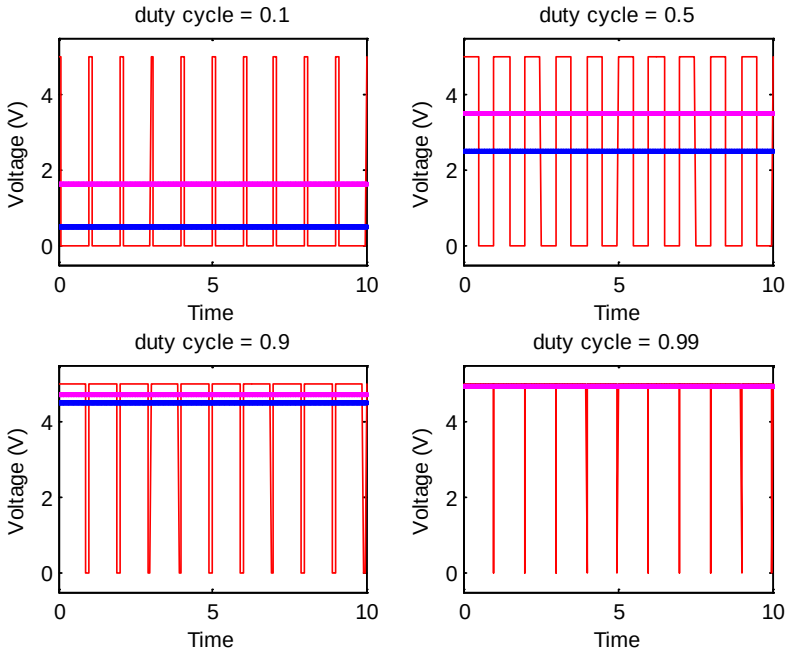
```
pinMode(A0, INPUT);
Int a = analogRead(A0);
```



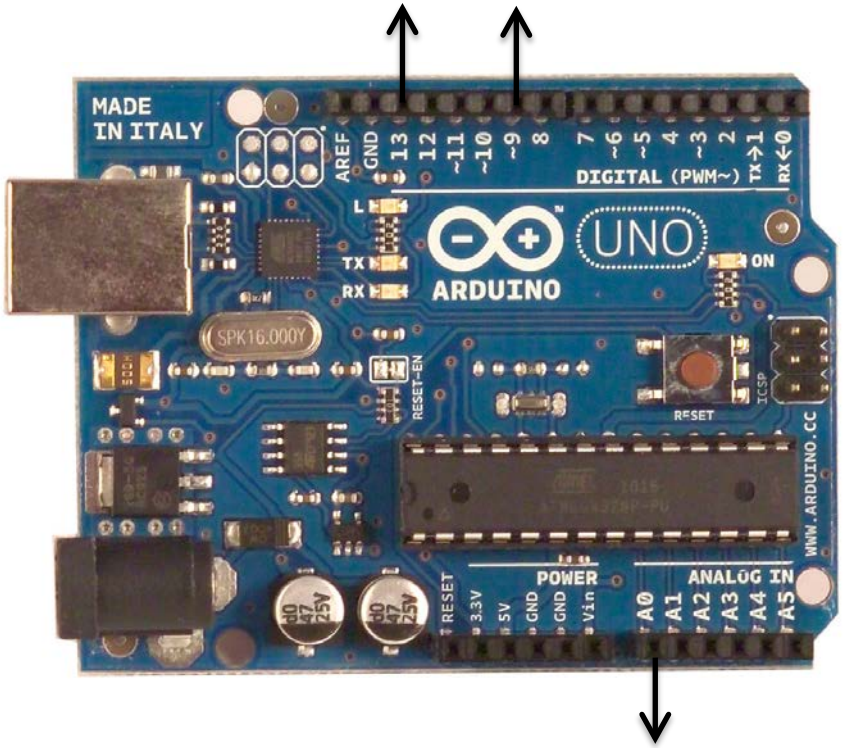
Electronics: Digital Outputs

Digital output:

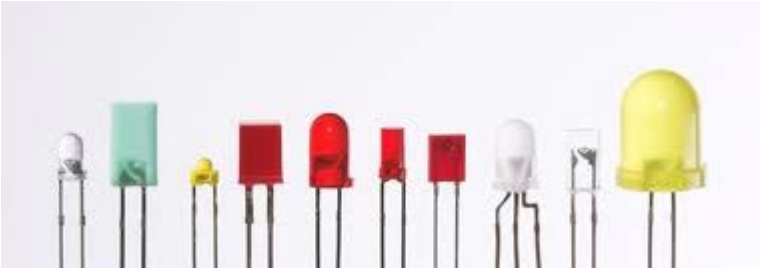
- 0 Volt: Low
- 5 Volt: High
- 2.5 Volt? **NO** (“PWM” : ~ pins)



PWM and average voltage for a number of duty cycles



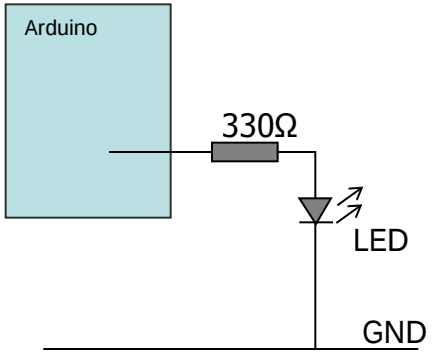
Electronics : Actuators



Electronics: Digital Outputs

```
pinMode(13, OUTPUT);
digitalWrite(13, LOW);
digitalWrite(13, HIGH);
```

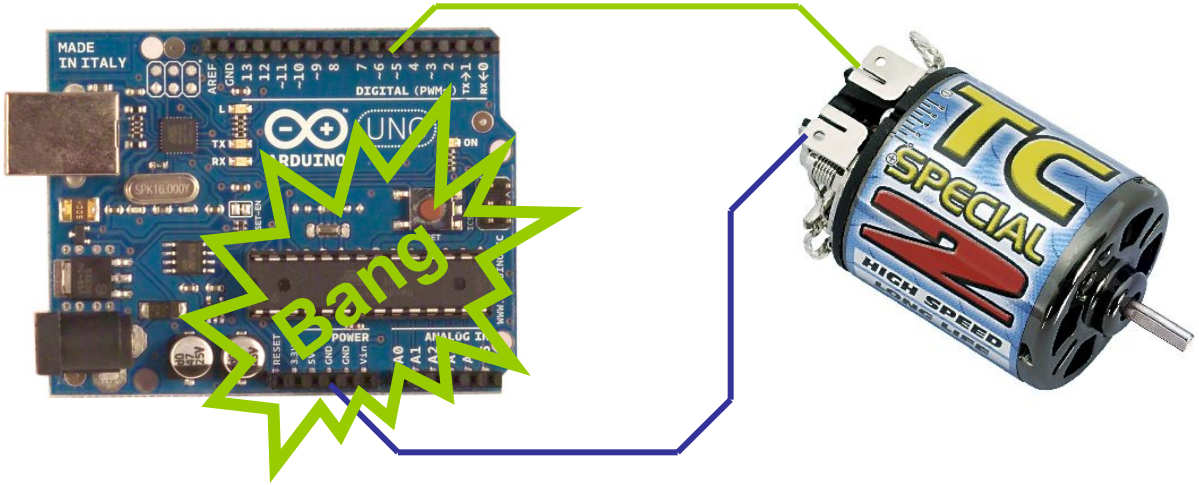
```
pinMode(11, OUTPUT);
analogWrite(11, 186);
```



Electronics: Digital Outputs

Digital output:

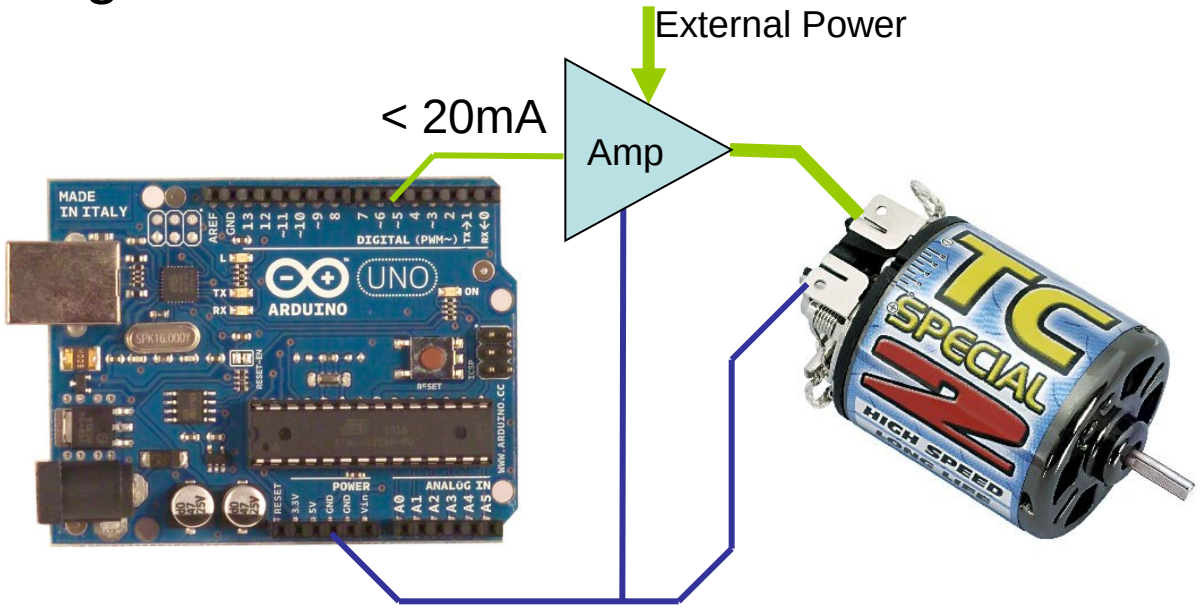
- 0 Volt: Low
- 5 Volt: High



Electronics: Digital Outputs

Digital output:

- 0 Volt: Low
- 5 Volt: High



Some Hardware Limitations (Arduino Uno)

- The Uno can be powered via **USB** or **external power supply**; power source is selected automatically. Usually a PC **USB port is power limited to 100 mA max**.
- External (non-USB) power can come from an **AC-to-DC adapter** or **battery**. The adapter can be connected into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.
- Operating Voltage 5 V
- Input Voltage **(recommended) 7-12 V**, (limits) 6-20 V. If the Uno is supplied with *less than 7 V*, the 5 V pin may supply less than five volts and the board may be *unstable*. If using more than 12 V, the voltage regulator may overheat and damage the board
- DC Current per **I/O pin max. 40 mA**
- DC Current for **3.3 V pin max. 50 mA**
- Depending on supply voltage and current usage (including the current for the processor) **the voltage regulator may overheat** and stop working until it has cooled down. Generally driving one servo or a few normal LEDs is the maximum.

The IDE program

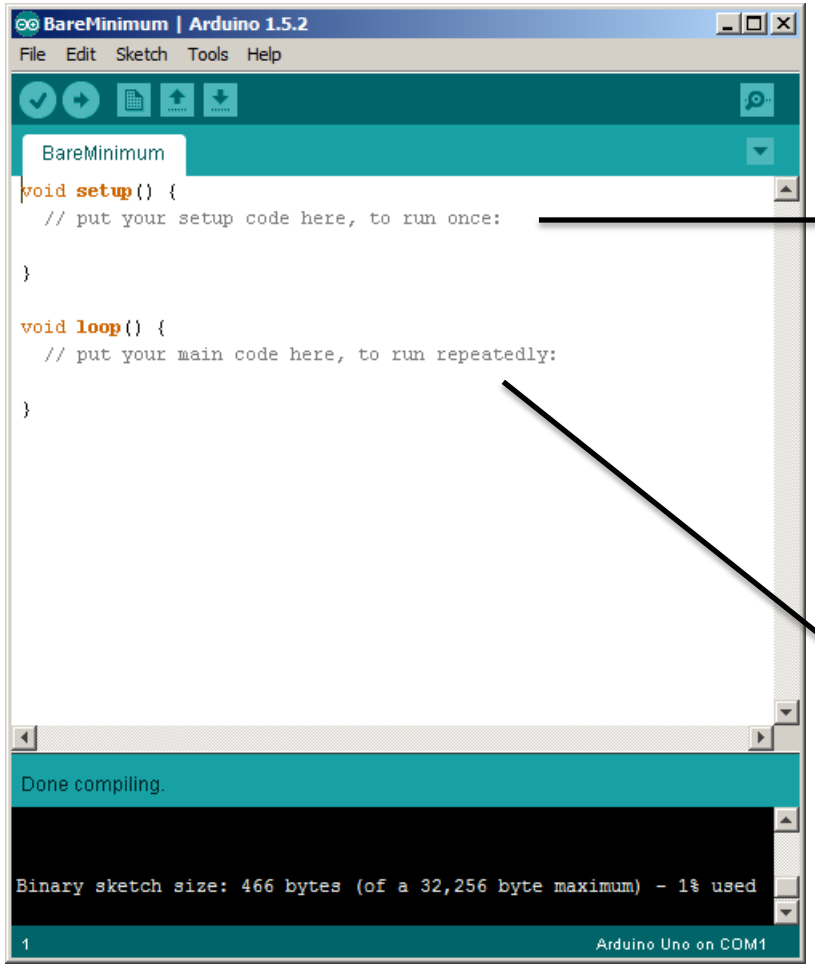
USB port

Acts as serial

Arduino board

A compiled “hex”-file is sent to the program memory of the Arduino board

The program



First, tell Arduino what it is:
- Declare pins as input or output
- Set global variables

This is done only once immediately after reset

Then tell it what to do:
- Read inputs
- Do calculations
- Set outputs

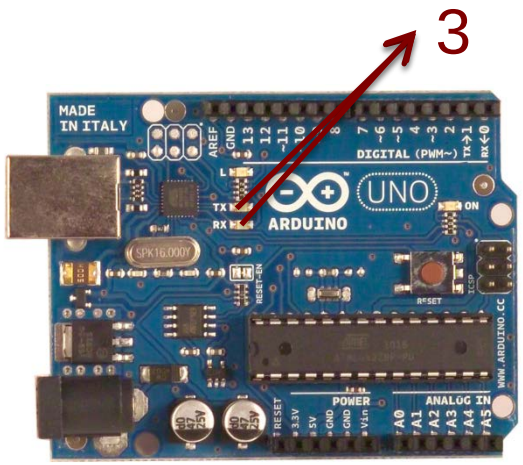
This is done continuously, in a loop after setup is finished

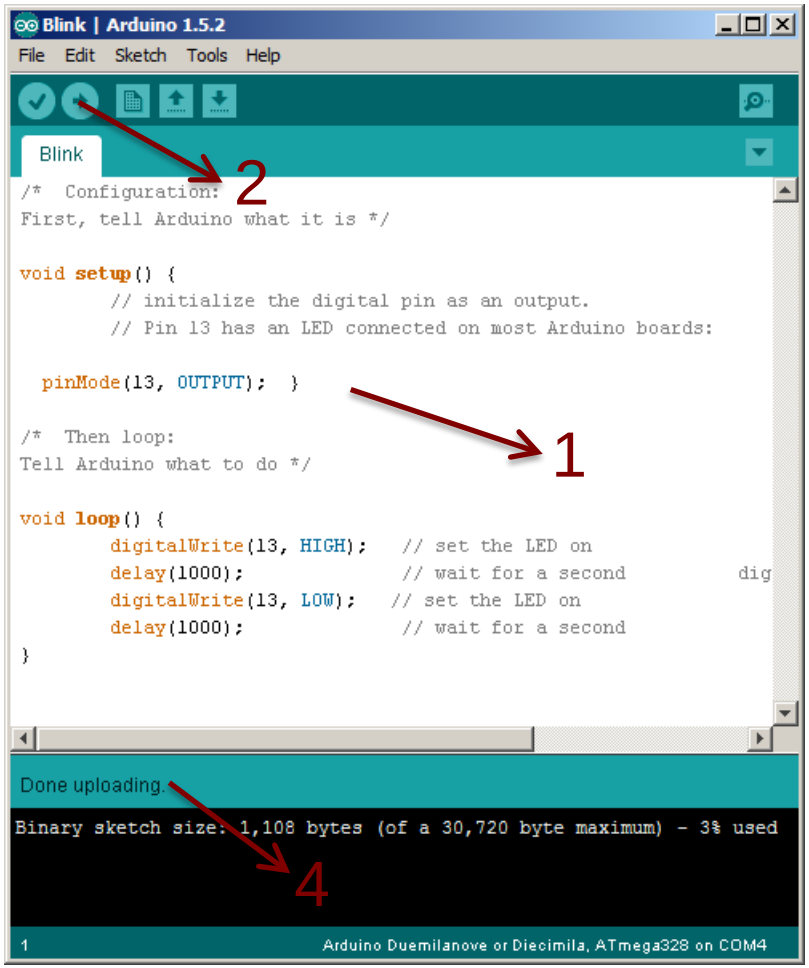
The program

```
/*  Configuration:
First, tell Arduino what it is */

void setup() {
        // initialize the digital pin as an output.
        // Pin 13 has a LED connected on most Arduino boards:
        pinMode(13, OUTPUT);
}

/*  Then loop:
Tell Arduino what to do */

void loop() {
        digitalWrite(13, HIGH);     // switch LED on
        delay(1000);                // wait for a second
        digitalWrite(13, LOW);      // switch LED off
        delay(1000);                // wait for a second
}
```

Compiling and uploading code

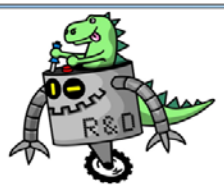
1. Type code in text window
2. Push ‘upload’ button
3. Check if TX and RX Leds are blinking rapidly
4. If the ‘Done uploading’ message displays, the Arduino is ready.





Arduino Cheat Sheet V.02c

Mostly taken from the extended reference:
http://arduino.cc/en/Reference/Extended
Gavin Smith – Robots and Dinosaurs, The Sydney Hackspace



	ATMega168	ATMega328	ATmega1280
Flash (2k for boobtloader)	16kB	32kB	128kB
SRAM	1kB	2kB	8kB
EEPROM	512B	1kB	4kB

	Duemilanove/ Nano/ Pro/ ProMini	Mega
# of IO	14 + 6 analog (Nano has 14+8)	54 + 16 analog
Serial Pins	0 - RX 1 - TX	0 - RX1 1 - TX1 19 - RX2 18 - TX2 17 - RX3 16 - TX3 15 - RX4 14 - TX4
Ext Interrupts	2 - (Int 0) 3 - (Int 1)	2,3,21,20,19,18 (IRQ0- IRQ5)
PWM pins	5,6 - Timer 0 9,10 - Timer 1 3,11 - Timer 2	0-13
SPI	10 - SS 11 - MOSI 12 - MISO 13 - SCK	53 - SS 51 - MOSI 50 - MISO 52 - SCK
I2C	Analog4 - SDA Analog5 - SCK	20 - SDA 21 - SCL

Structure
void **setup**() void **loop**()

Control Structures
if (x<5){ } **else** { }
switch (myvar) {
  case 1:
    break;
  case 2:
    break;
  default:
}
for (int i=0; i <= 255; i++){ }
while (x<5){ }
do { } **while** (x<5);
continue; //Go to next in do/for/while loop
return x; // Or 'return;' for voids.
goto // considered harmful :-)

Further Syntax
// (single line comment)
/* (multi-line comment) */
#**define** DOZEN 12 //Not baker's!
#**include** <avr/pgmspace.h>

General Operators
= (assignment operator)
+ (addition) - (subtraction)
* (multiplication) / (division)
% (modulo)
== (equal to) != (not equal to)
< (less than) > (greater than)
<= (less than or equal to)
>= (greater than or equal to)
&& (and) || (or) ! (not)

Pointer Access
& reference operator
* dereference operator

Bitwise Operators
& (bitwise and) | (bitwise or)
^ (bitwise xor) ~ (bitwise not)
<< (bitshift left) >> (bitshift right)

Compound Operators
++ (increment) -- (decrement)
+= (compound addition)
-= (compound subtraction)
*= (compound multiplication)
/= (compound division)
&= (compound bitwise and)
|= (compound bitwise or)

Constants
HIGH | LOW
INPUT | OUTPUT
true | false
143 // **Decimal** number
0173 // **Octal** number
0b11011111 //**Binary**
0x7B // **Hex** number
7**U** // Force unsigned
10**L** // Force long
15**UL** // Force long unsigned
10.0 // Forces floating point
2.4e5 // 240000

Data Types
void
boolean (0, 1, false, true)
char (e.g. 'a' -128 to 127)
unsigned char (0 to 255)
byte (0 to 255)
int (-32,768 to 32,767)
unsigned int (0 to 65535)
word (0 to 65535)
long (-2,147,483,648 to 2,147,483,647)
unsigned long (0 to 4,294,967,295)
float (-3.4028235E+38 to 3.4028235E+38)
double (currently same as float)
sizeof(myint) // returns 2 bytes

Strings
char S1[15];
char S2[8]={'a','r','d','u','i','n','o'};
char S3[8]={'a','r','d','u','i','n','o','\0'};
//Included \0 null termination
char S4[ ] = "arduino";
char S5[8] = "arduino";
char S6[15] = "arduino";

Arrays
int myInts[6];
int myPins[] = {2, 4, 8, 3, 6};
int mySensVals[6] = {2, 4, -8, 3, 2};

Conversion
char() **byte**()
int() **word**()
long() **float**()

Qualifiers
static // persists between calls
volatile // use RAM (nice for ISR)
const // make read-only
PROGMEM // use flash

Digital I/O
pinMode(pin, [INPUT,OUTPUT])
digitalWrite(pin, value)
int **digitalRead**(pin)
//Write High to inputs to use pull-up res

Analog I/O
analogReference([DEFAULT,INTERNAL,EXTERNAL])
int **analogRead**(pin) //Call twice if switching pins from high Z source.
analogWrite(pin, value) // PWM

Advanced I/O
tone(pin, freqhz)
tone(pin, freqhz ,duration_ms)
noTone(pin)
shiftOut(dataPin, clockPin, [MSBFIRST,LSBFIRST], value)
unsigned long **pulseIn**(pin, [HIGH,LOW])

Time
unsigned long **millis**() // 50 days overflow.
unsigned long **micros**() // 70 min overflow
delay(ms)
delayMicroseconds(us)

Math
min(x, y) **max**(x, y) **abs**(x)
constrain(x, minval, maxval)
map(val, fromL, fromH, toL, toH)
pow(base, exponent) **sqrt**(x)
sin(rad) **cos**(rad) **tan**(rad)

Random Numbers
randomSeed(seed) // Long or int
long **random**(max)
long **random**(min, max)

Bits and Bytes
lowByte() **highByte**()
bitRead(x,bitn) **bitWrite**(x,bitn,bit)
bitSet(x,bitn) **bitClear**(x,bitn)
bit(bitn) //bitn: 0-LSB 7-MSB

External Interrupts
attachInterrupt(interrupt, function, [LOW,CHANGE,RISING,FALLING])
detachInterrupt(interrupt)
interrupts()
noInterrupts()

Libraries:

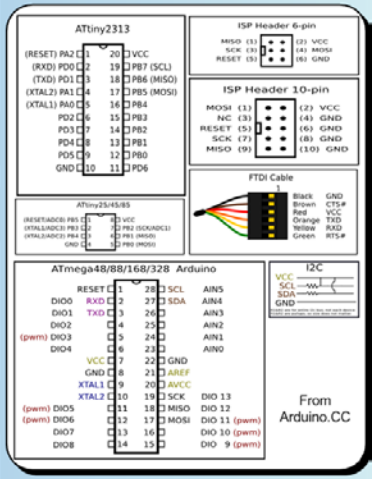
Serial.
begin([300, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, 115200])
end()
int **available**()
int **read**()
flush()
print()
println()
write()

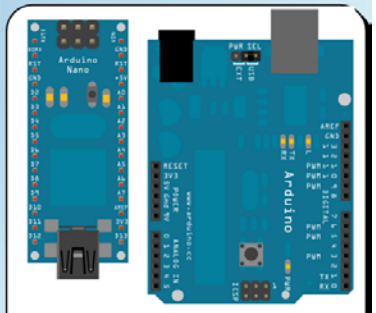
EEPROM (#include <EEPROM.h>)
byte **read**(intAddr)
write(intAddr,myByte)

Servo (#include <Servo.h>)
attach(pin , [min_uS, max_uS])
write(angle) // 0-180
writeMicroseconds(uS) //1000-2000, 1500 is midpoint
read() // 0-180
attached() //Returns boolean
detach()

SoftwareSerial(RxPin,TxPin)
// #include<**SoftwareSerial**.h>
begin(longSpeed) // up to 9600
char **read**() // blocks till data
print(myData) or **println**(myData)

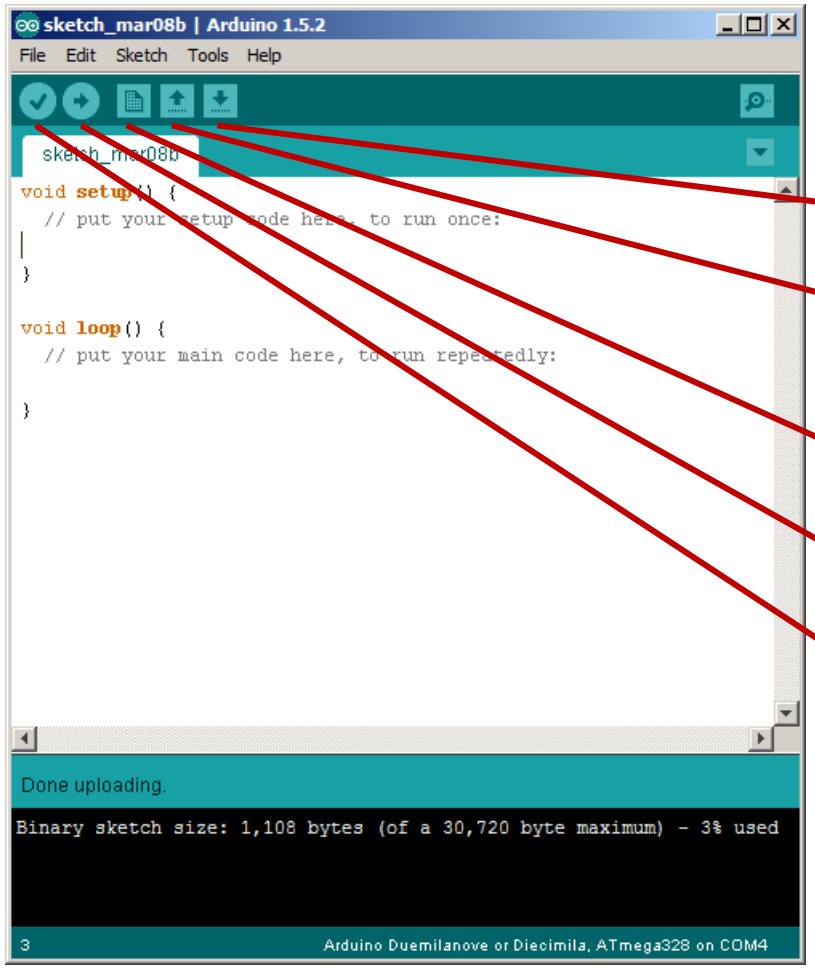
Wire (#include <Wire.h>) // For **I2C**
begin() // Join as master
begin(addr) // Join as slave @ addr
requestFrom(address, count)
beginTransmission(addr) // Step 1
send(mybyte) // Step 2
send(char * mystring)
send(byte * data, size)
endTransmission() // Step 3
byte **available**() // Num of bytes
byte **receive**() //Return next byte
onReceive(handler)
onRequest(handler)





Pics from Fritzing.Org under C.C. license

The Arduino Programming Environment



Save Sketch

Open Sketch
(old: *.pde, new: *.ino)

New Sketch

Compile and Upload

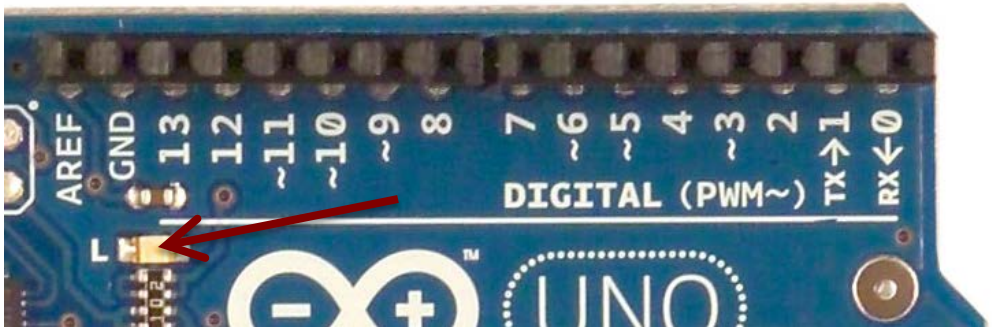
Verify (Compile)

Settings

- Let the Arduino programming environment know which board you have
  - “Tools” menu ➔ “Board”
- Let the Arduino programming environment know to which port the board is connected
  - “Tools” menu ➔ “Serial port”

Blink

```
/*  comment */

void setup() {
        // initialize the digital pin as an output.
        // Pin 13 has a LED connected on most Arduino boards:
        pinMode(13, OUTPUT);
}

void loop() {
        digitalWrite(13, HIGH);     // switch LED on
        delay(1000);                // wait 1000 ms
        digitalWrite(13, LOW);      // switch LED off
        delay(1000);                // wait 1000 ms
}
```



Blink with button

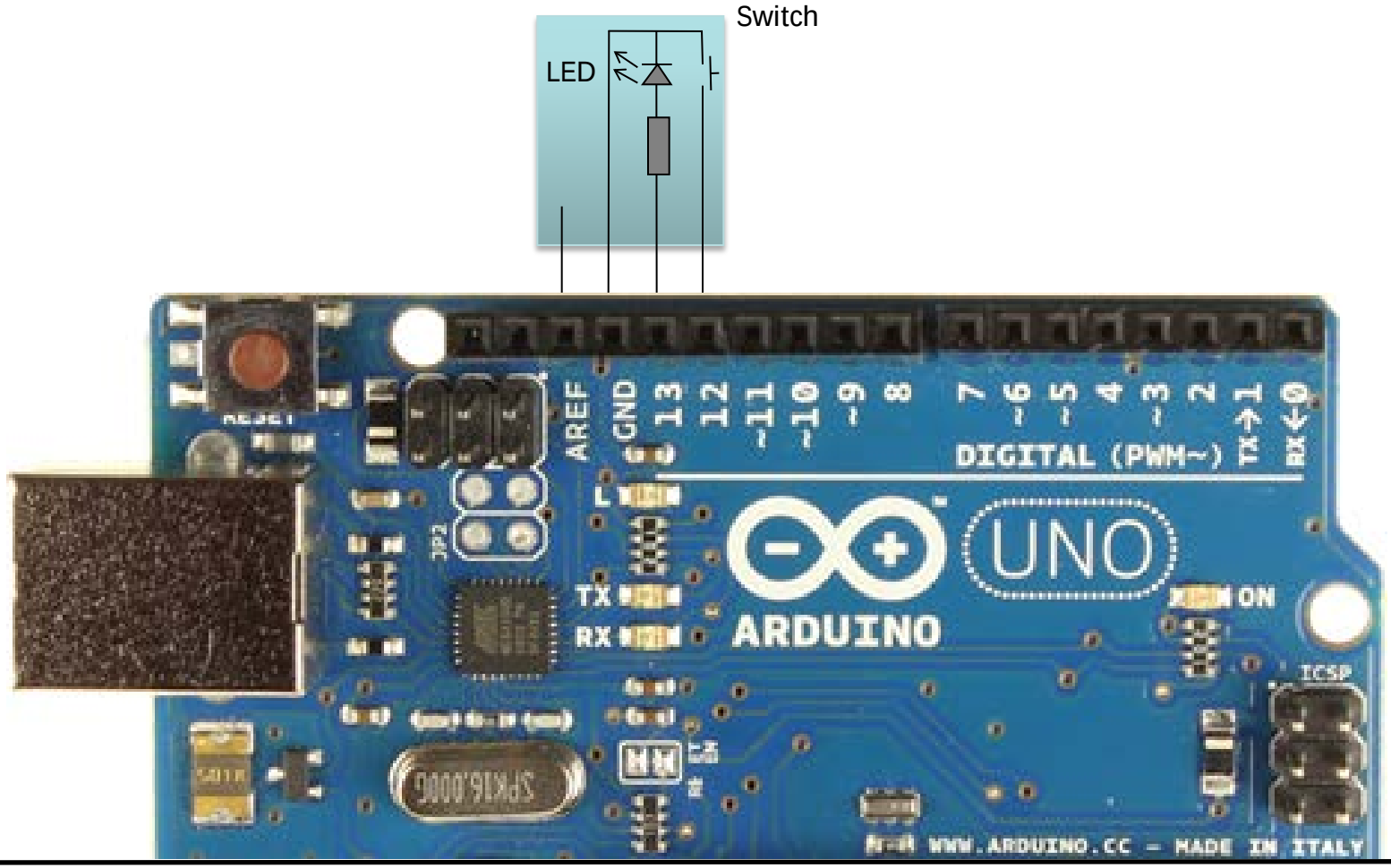
- The period of “Blink” is 2 x 1000 ms = 2 s
- Now we want: if we push a button, the period should become 0.5 s

- To do:
  - Connect a button
  - Read a button
  - Change the “1000” in “delay” depending on the button state

Blink and read button

```
/*  comment */

void setup() {
        // initialize digital pin 13 as an output.
        // Pin 13 has a LED connected on most Arduino boards:
        pinMode(13, OUTPUT);
        pinMode(12, INPUT);          // initialize digital pin 12 as input.
        digitalWrite(12, HIGH);      // activate pull-up resistor
}

void loop() {
        digitalWrite(13, HIGH);      // switch LED on
        delay(1000);                 // wait one second
        digitalWrite(13, LOW);       // switch LED off
        delay(1000);                 // wait one second
}
```

Possible hardware for Blink with button



```
void setup() {
        // initialize the digital pin 13 as an output.
        // Pin 13 has a LED connected on most Arduino boards:
        pinMode(13, OUTPUT);
        pinMode(12, INPUT);          // initialize digital pin 12 as input.
        digitalWrite(12, HIGH);      // activate pull-up resistor
}

void loop() {
        if (digitalRead(12) == HIGH) {
                // Do something
        }
        else {
                // Do something else
        }
        digitalWrite(13, HIGH);      // switch LED on
        delay(1000);                 // wait one second
        digitalWrite(13, LOW);       // switch LED off
        delay(1000);                 // wait one second
}
```

```
int wait = 1000;

void setup() {
        ...
}

void loop() {
        if (digitalRead(12) == HIGH) {
                // Do something
        }
        else {
                // Do something else
        }
        digitalWrite(13, HIGH);     // switch LED on
        delay(wait);                // wait for wait msec
        digitalWrite(13, LOW);      // switch LED off
        delay(wait);                // wait for wait msec
}
```

```
int wait = 1000;

void setup() {
        ...
}

void loop() {
        if (digitalRead(12) == HIGH) {
                wait = 1000; // Wait 1 sec
        }
        else {
                wait = 250;  // Wait 0.25 sec
        }
        digitalWrite(13, HIGH);     // switch LED on
        delay(wait);                // wait for wait msec
        digitalWrite(13, LOW);      // switch LED off
        delay(wait);                // wait for wait msec
}
```

Create some interaction: The “Physical Pixel” example

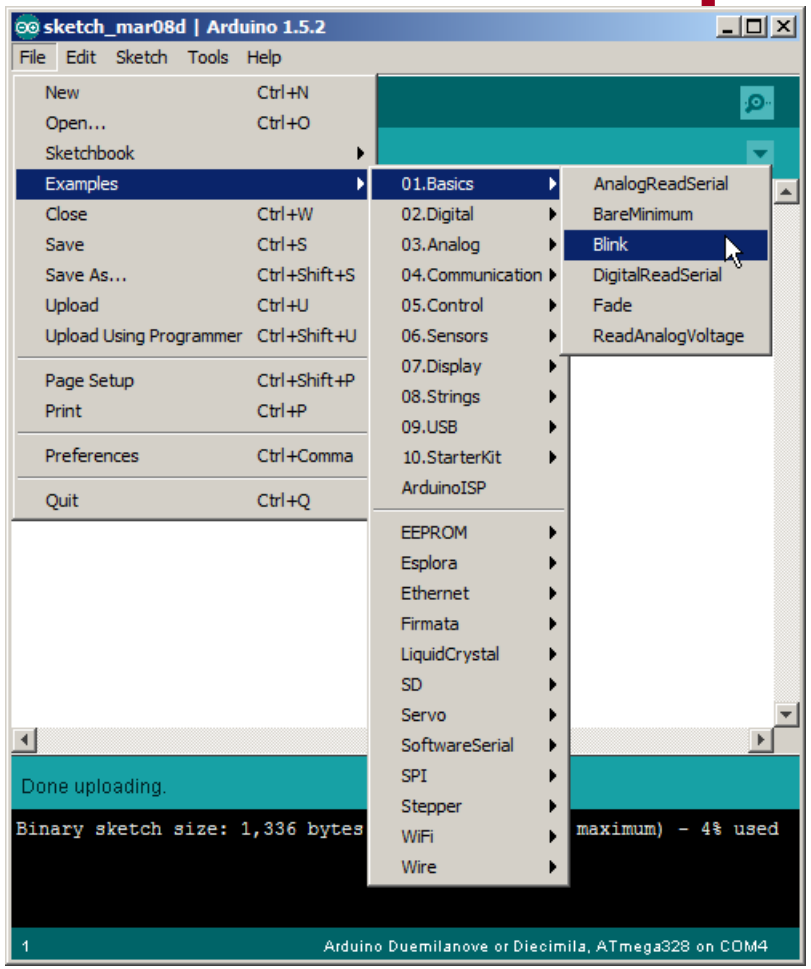
```
const int ledPin = 13;  // the pin that the LED is attached to
int incomingByte;       // a variable to read incoming serial data into

void setup() {
  Serial.begin(9600);                  // initialize serial communication
  Serial.println(”Good morning”);      // Send text to computer
  pinMode(ledPin, OUTPUT);             // initialize LED pin as output
}

void loop() {
    if (Serial.available() > 0) {
       incomingByte = Serial.read();
       if (incomingByte == 'H') digitalWrite(ledPin, HIGH);
       if (incomingByte == 'L') digitalWrite(ledPin, LOW);
    }
}
```

How to proceed?

- Find new functions in the reference (http://arduino.cc/en/Reference/HomePage)
- Arduino playground Manuals and Curriculum (also downloads) (http://playground.arduino.cc/Main/ManualsAndCurriculum)

- Don't be scared by "Making things talk": the book is about connecting to the internet and less to interfacing with electronics

Available examples



For descriptions see http://arduino.cc/en/Tutorial/HomePage

The mini project

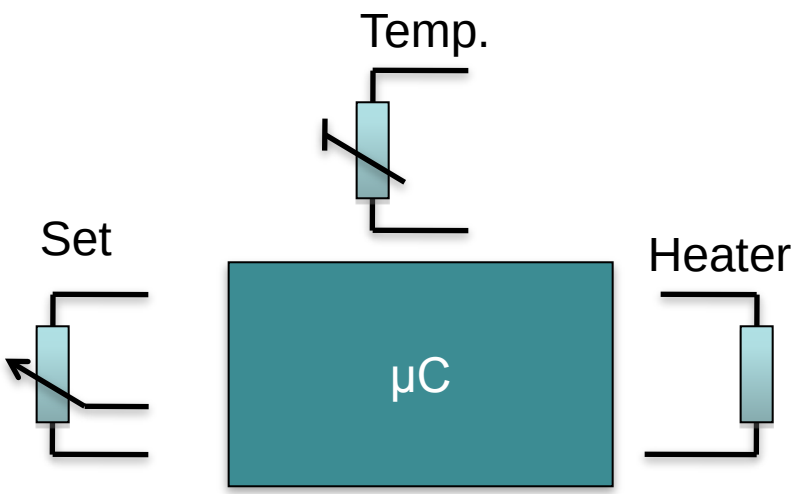
- Session Dec 8 team presents mini project proposal (HG 4.24 Group1 / HG 4.57 Group2)
- Session Dec 11 and Dec 15 no lecture, but support in and around e-Atelier/ID-Café for mini project development
- Session Dec 18 (HG 4.57) team presents mini project result (demo + mini poster) to assignor

The mini project

- You are free to choose your project, but:
  - Use Arduino if possible
  - Use at least one sensor and one actuator
  - Report on design choices (transistor, configuration, etc.)
  - Draw schematics
  - Present it as a prototype (user experience, packaging) !

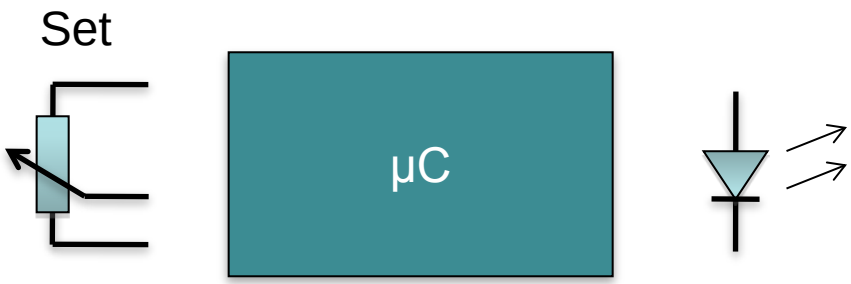
Suggestions

- Make the central heating system with Arduino



Suggestions

- Dim a LED
- Make application / context



Suggestions

- Egg timer

Spaghetti in code, schematics and circuits

Code:
- Use comments /* like this */ or // this
- Use functional variable names
- Use functions

Schematics:
- Draw the complete circuit (correctly!!) (component names, values)

Circuit (soldered or breadboard)
- Should be structured
- Use color coding in wires
- Think about component placement

Otherwise:
- Debugging becomes difficult
- You lack evidence in your report

Code

DG233_DecentCode | Arduino 0022

DG233_DecentCode

```
const int Baudrate = 19200;
const int Analog_Input_Pin_A = 0; // Sensor A is connected to this pin
const int Analog_Input_Pin_B = 1; // Sensor A is connected to this pin

void setup()
{
  Serial.begin(Baudrate);          // For sending data to the computer over USB
  WriteStartCommand();
  InitialiseSensors(Analog_Input_Pin_A, Analog_Input_Pin_B);
  InitialiseTimers();
  }

void loop() {
  Read_Sensors_and_Write_Data();
  if (SerialCommand()) {
    Do_Something_With_Command();
  }
}

/* Implementation of subroutines:        */
void   WriteStartCommand(void) {
```

Done Saving.

```
Binary sketch size: 1372 bytes (of a 30720 byte maximum)
```

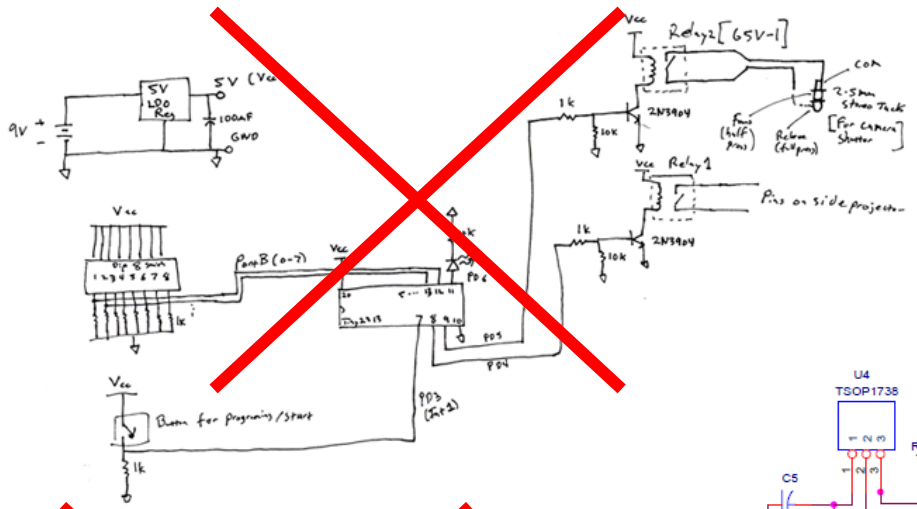
20

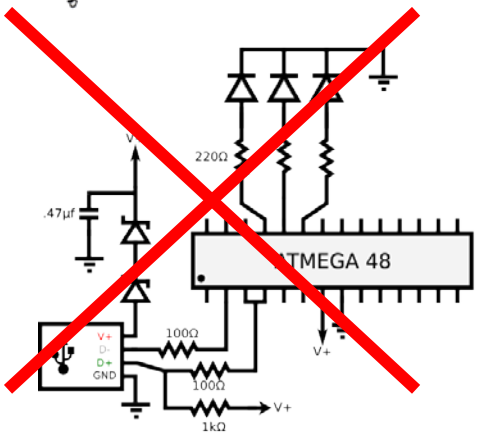
Code:

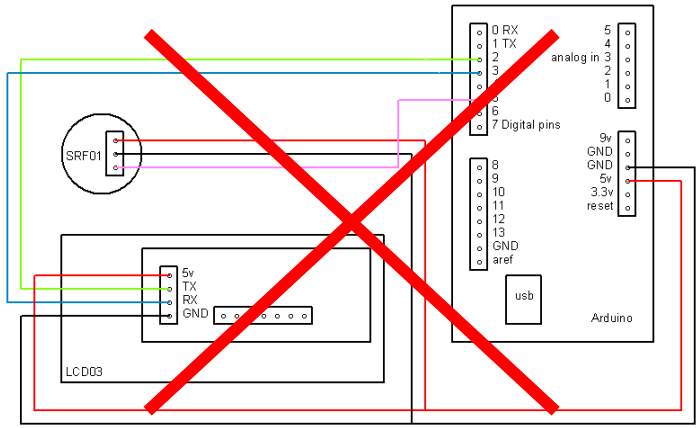
- Clear variable names
- Proper use of local and global variables
- Functions `setup()` and `loop()` are self-explaining
- Proper function names
- Comments (Header)

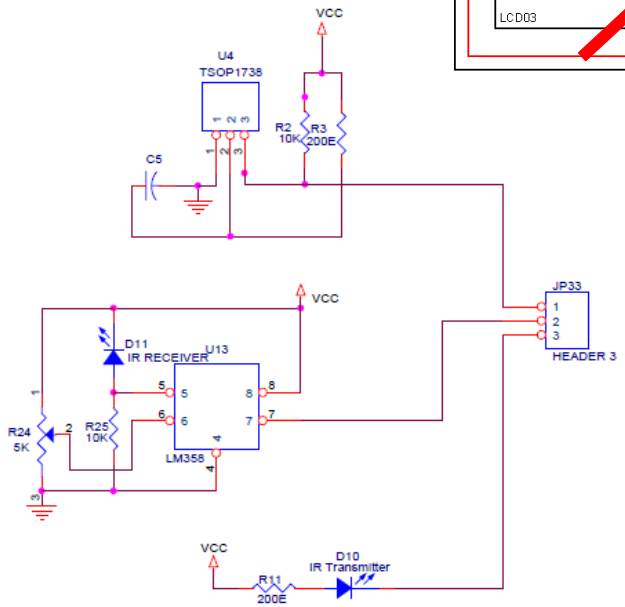
- Examples
  - Bad example
  - Good example

Schematics

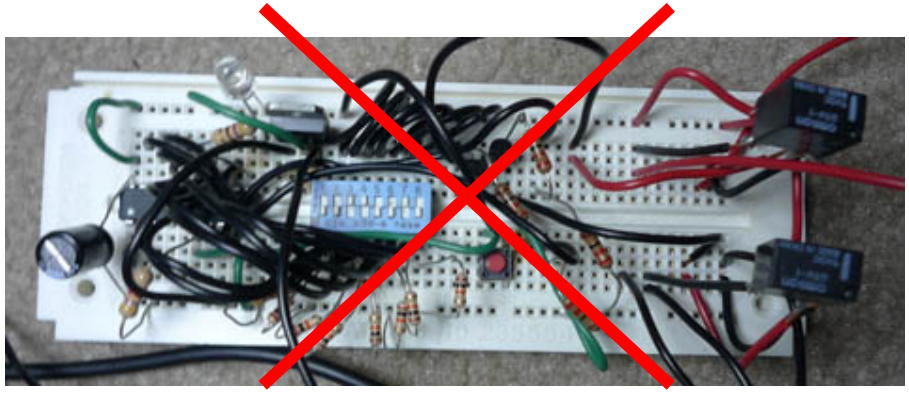

9V
5V
100uF
GND
Vcc
Relay2 [G5V-1]
Relay1
2N3904
1k
10k
Pins on side projector
Button for programming/start

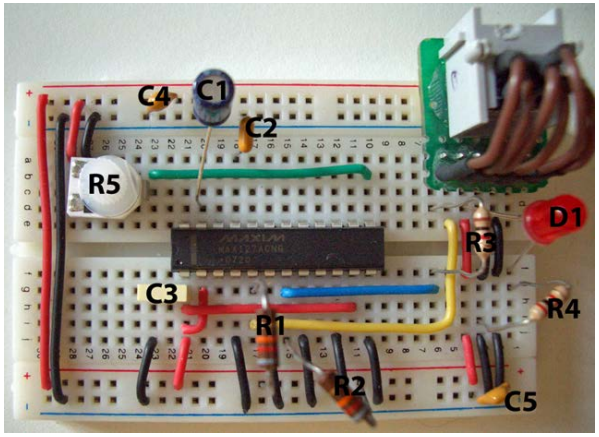

220Ω
.47µf
ATMEGA 48
100Ω
1kΩ
V+
GND


0 RX
1 TX
2
3
6
7 Digital pins
analog in 3
5
4
2
1
0
SRF01
9v
GND
GND
5v
3.3v
reset
8
9
10
11
12
13
GND
aref
usb
Arduino
5v
TX
RX
GND
LCD03


VCC
U4
TSOP1738
R2
10K
R3
200E
C5
D11
IR RECEIVER
U13
LM358
R24
5K
R25
10K
JP33
HEADER 3
D10
IR Transmitter
R11
200E

Circuits





Poster

- Explanation
- Schematic
- Code
- Pictures

DG291 Creative Electronics

Technische Universiteit Eindhoven University of Technology

PrankTron: your nightmare

Geert Langereis (B24.1)

Introduction

PrankTron is a small electronic module that selects randomly one out of four sounds and bothers you with that at irregular times between 10 minutes and 1.5 hours. It is small enough to fit in any obscure location, for example in the board room of a study association. PrankTron is inspired by the commercially available Annoy-a-Trons[1].

Specifications

- Small size (smaller than Arduino)
- Low power (should stay active for 10 days)
- Several sounds

Electronics

The only thing needed for a PrankTron is a buzzer. A 5V buzzer can be connected to an Arduino board directly. However, to make a low-power version, the Atmel microcontroller was taken off the Arduino board, and placed on a piece of experimentation board. Now I could use a 3V battery and a 10MHz clock (instead of 16MHz), which gives a huge reduction of power[2]:

So, what is achieved is:

- ATMEGA current: 3mA
- Battery CR2032H: 240mAh
- Lifetime PrankTron: 80 hours (> 3 days)

Code

The code has two tricky parts. First of all, a function for long delays had to be made, and next, a method to deal with the 10MHz clock. Debugging was done on a normal 16MHz Arduino board with `Serial.println();` commands.

Results

Tests were performed by placing a 4x4 cm² PrankTron on a piece of experimentation board under the sales counter in the Lucid boardroom in November 2011. The result was hilarious, and PrankTron was only discovered after a whole day of total confusion.

References

[1]The ThinkGeek Annoy-a-tron
http://www.thinkgeek.com/product/8c52/

[2]Atmel ATMEGA328P datasheet
http://www.atmel.com/dyn/resources/prod_documents/doc8271.pdf

/ Industrial Design

Resources

- Look at wiki page:
  - http://wiki.id.tue.nl/ce/AssignmentDescription
    (and checkout the “Resources” section)

- Installation
  - http://arduino.cc/en/Guide/Windows
  - http://arduino.cc/en/Guide/MacOSX