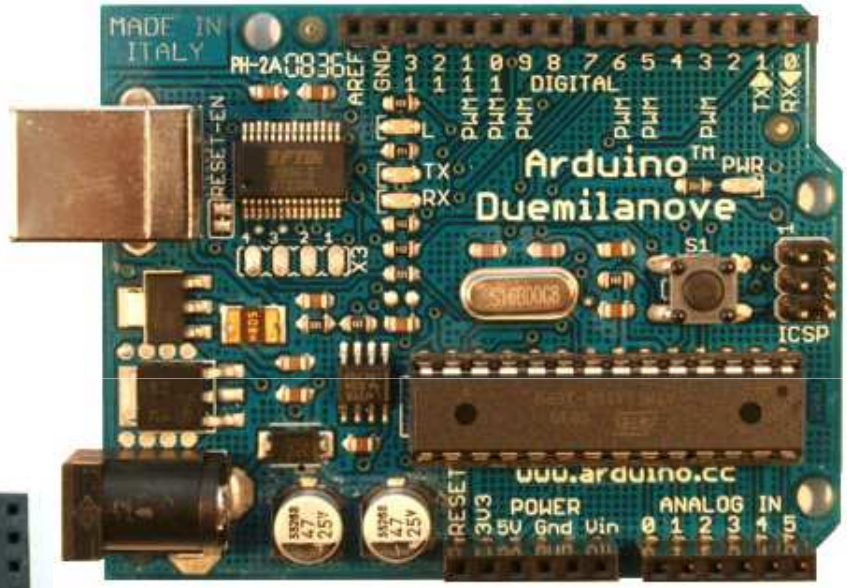
Lecture 8: Programming for the Arduino

- The hardware
- The programming environment
- Binary world, from Assembler to C
- Programming C for the Arduino: Basics
- Programming C for the Arduino: more …
- Programming style

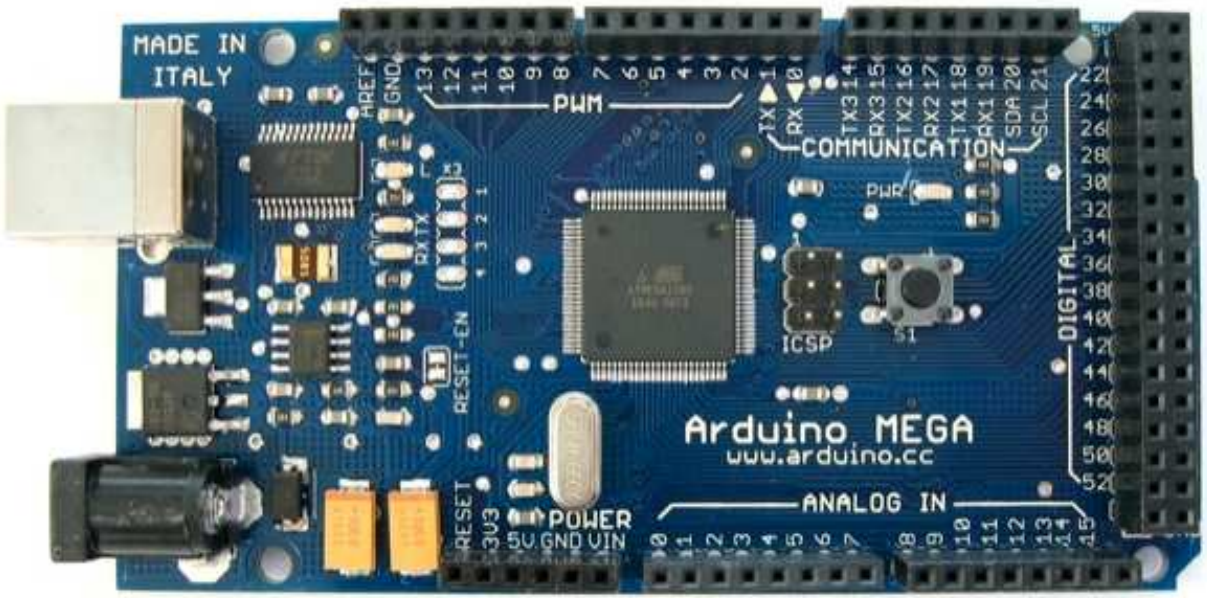
The hardware

Arduino Duemilanove



http://www.arduino.cc/

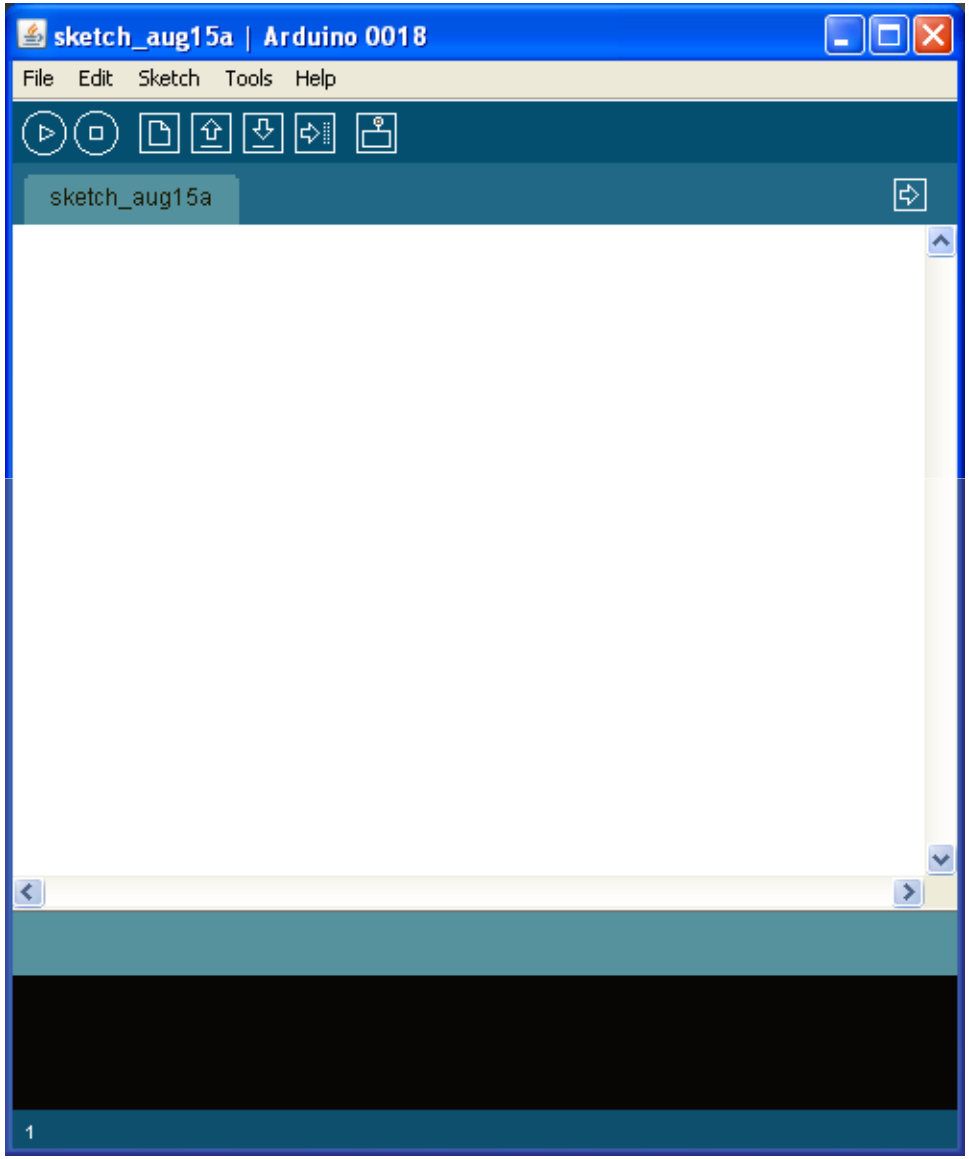
Arduino Mega



Lecture 8: Programming for the Arduino

- The hardware
- The programming environment
- Binary world, from Assembler to C
- Programming C for the Arduino: Basics
- Programming C for the Arduino: more …
- Programming style

Programming environment



Download from:
http://arduino.cc/en/Main/Software

Tools

Auto Format	Strg+T
Archive Sketch	
Fix Encoding & Reload	
Serial Monitor	Strg+Umschalt+M
Board	▸
Serial Port	▸
Burn Bootloader	▸

Sketch

Verify / Compile	Strg+R
Stop	
Show Sketch Folder	Strg+K
Import Library...	▸
Add File...	

Edit

Undo	Strg+Z
Redo	Strg+Y
Cut	Strg+X
Copy	Strg+C
Copy for Forum	Strg+Umschalt+C
Copy as HTML	Strg+Alt+C
Paste	Strg+V
Select All	Strg+A
Comment/Uncomment	Strg+Slash
Increase Indent	Strg+Close Bracket
Decrease Indent	Strg+Open Bracket
Find...	Strg+F
Find Next	Strg+G

File

New	Strg+N
Open...	Strg+O
Sketchbook	▸
Examples	▸
Close	Strg+W
Save	Strg+S
Save As...	Strg+Umschalt+S
Upload to I/O Board	Strg+U
Page Setup	Strg+Umschalt+P
Print	Strg+P
Preferences	Strg+Comma
Quit	Strg+Q

Lecture 8: Programming for the Arduino

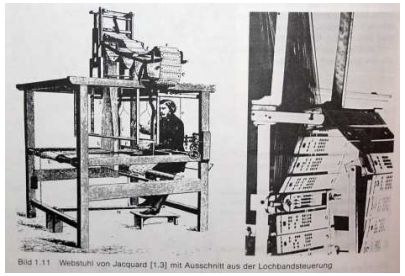
- The hardware
- The programming environment
  - Binary world, from Assembler to C
  - Programming C for the Arduino: Basics
  - Programming C for the Arduino: more …
  - Programming style

Binary world, programming from Assembler to C

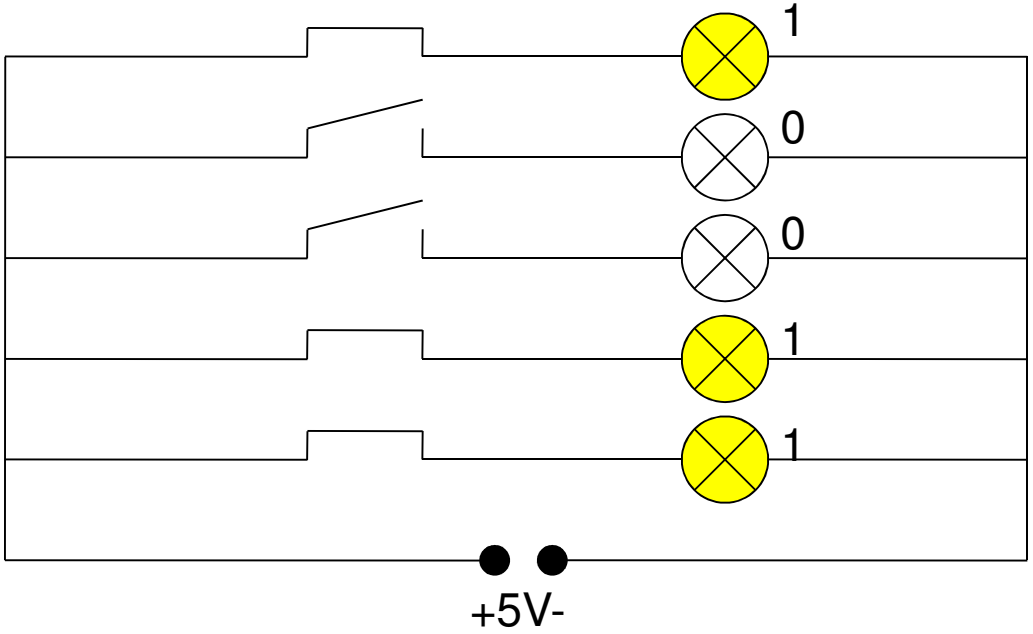
Programmable, mechanical calculation machines (19th/early 20th cent. AD)

Falcon
(1728)

Joseph-Marie Jacquard
(1805)
-> punchcard looms for work steps (program)



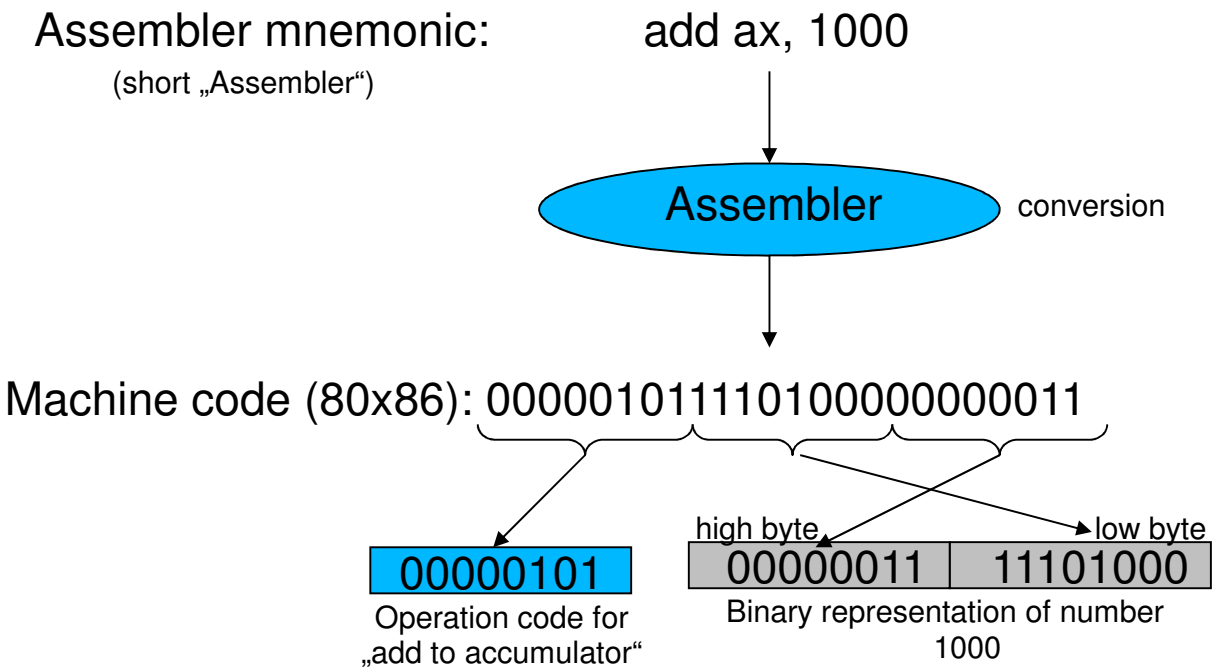
- Origins at China and also developed by the mathematician Leibniz (17th century AD)
- Positional numeral system which represents each number just with 2 symbols, 0 and 1
- These values can be represented by voltage levels in electronic circuits
- For human use very inefficient but with electronic circuits it is possible to create very efficient arithmetic and logic units (ALUs) for the basic operations addition, subtraction, multiplication and division



See: Rembold, Ulrich et. al.: Einführung in die Informatik für Naturwissenschaftler und Ingenieure. Hanser München Wien 1991
See: http://privat.swol.de/SvenBandel/index.html, Download 01.09.2007
See: http://en.wikipedia.org/wiki/Image:Boulier1.JPG, 01.09.2007
See: Rembold, Ulrich et. al.: Einführung in die Informatik für Naturwissenschaftler und Ingenieure. Hanser München Wien 1991

Binary world, programming from Assembler to C

The machine instructions



See: http://ivs.cs.uni-magdeburg.de/bs/lehre/sose99/bs1/seminare/assembler.shtml, Download 30.08.2007

Binary world, programming from Assembler to C

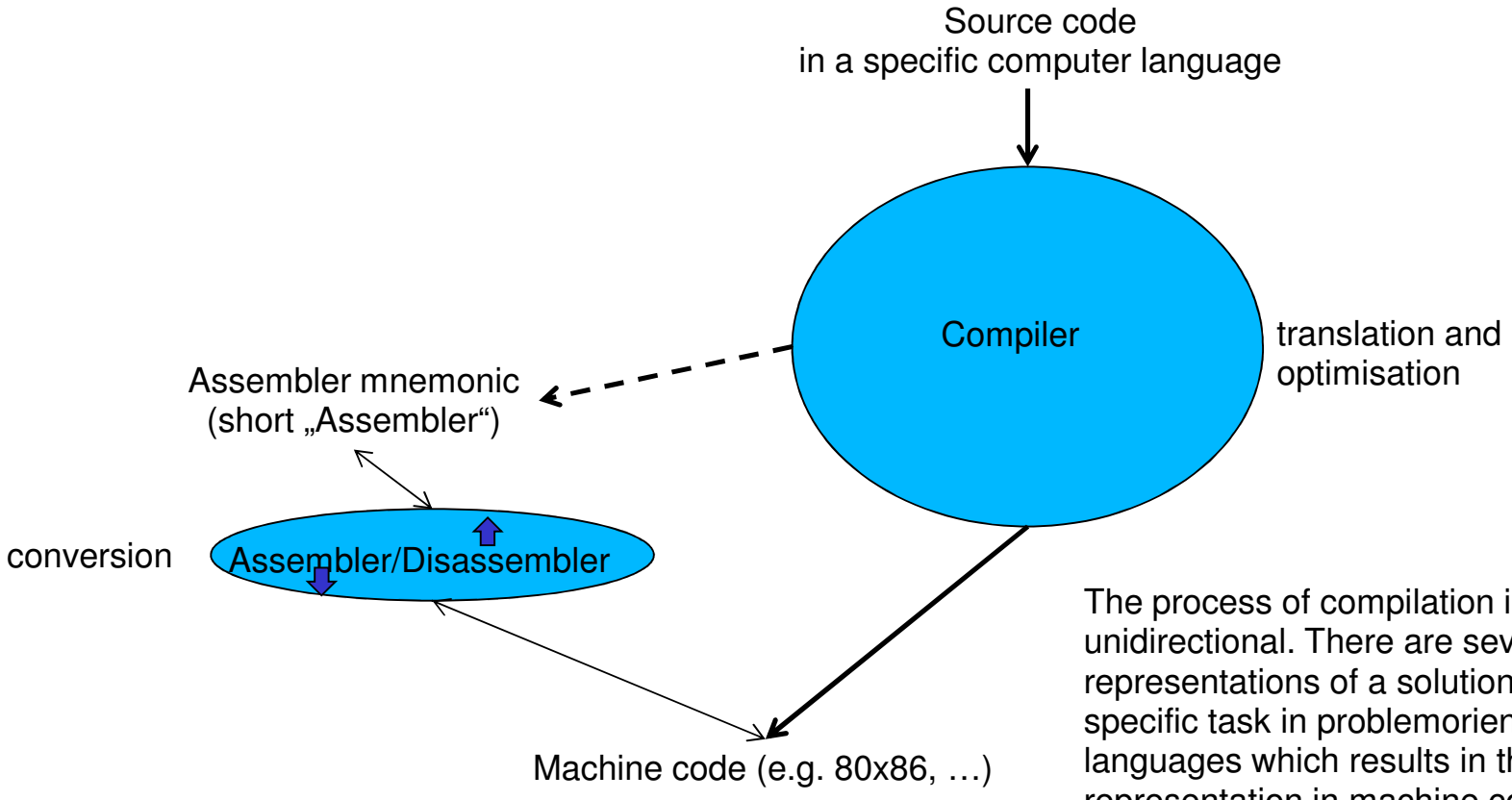
Programming paradigm of the problem oriented computer language C

- Procedural programming with structured programming as subset
  - *Code is splitted into several, reusable sections called procedures or functions with own scopes, which can be called at given code positions*
  - *Logical procedure units are combined to modules*
  - *Jumps (like goto) are not allowed*
  - *E.g. C*
  - *Case sensitive*

See: http://en.wikipedia.org/wiki/Programming_paradigm, Download 08.10.2007

Binary world, programming from Assembler to C

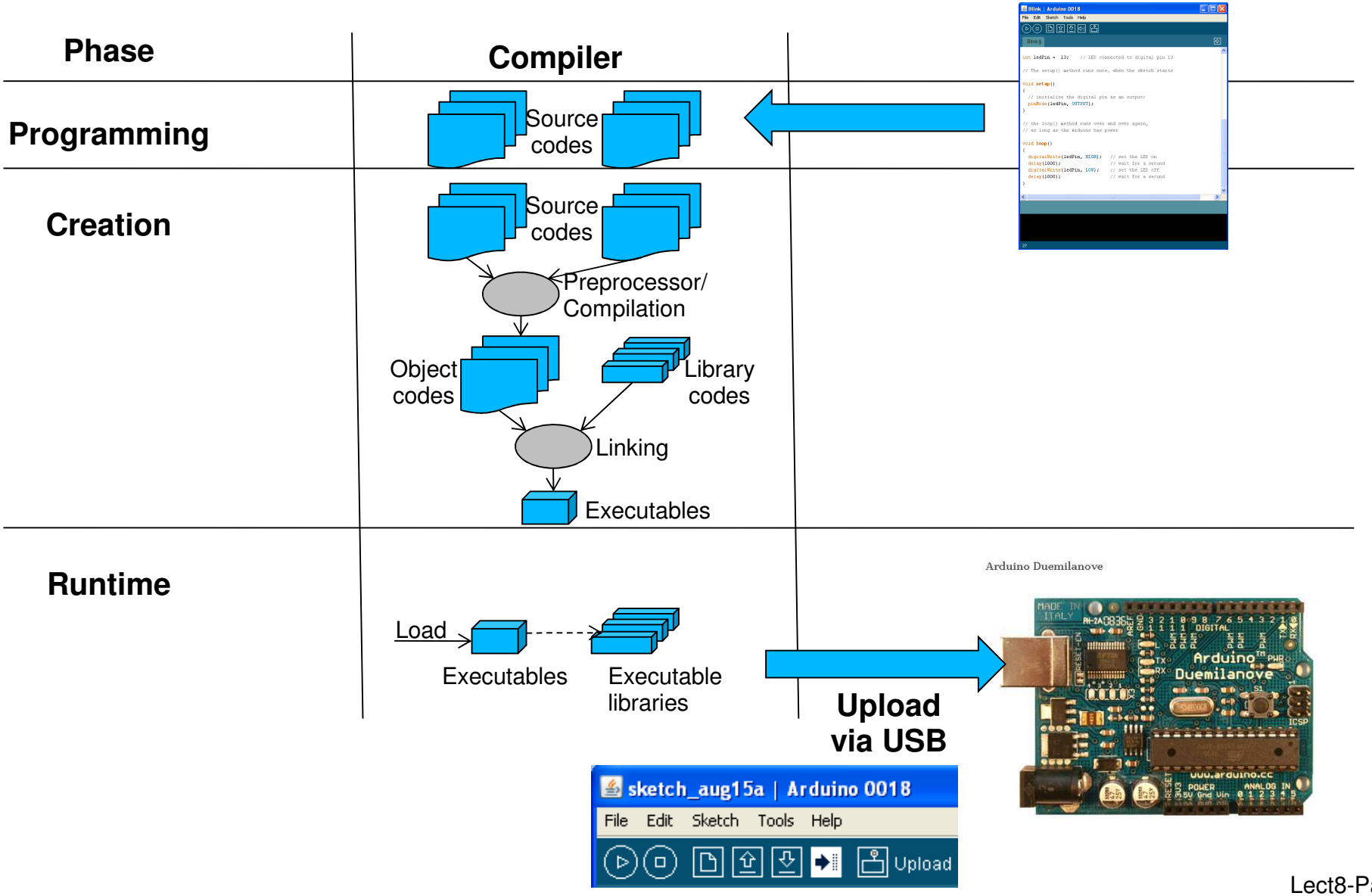
From source code to machine instructions



The process of compilation is unidirectional. There are several representations of a solution for a specific task in problemoriented languages which results in the same representation in machine code. On the other hand the optimisation is not ideal, so that the knowledge about the results after comiplation can improve the runtime of a program.

See: http://ivs.cs.uni-magdeburg.de/bs/lehre/sose99/bs1/seminare/assembler.shtml, Download 30.08.2007

Binary world, programming from Assembler to C



Lecture 8: Programming for the Arduino

- The hardware
- The programming environment
- Binary world, from Assembler to C
  - Programming C for the Arduino: Basics
  - Programming C for the Arduino: more …
  - Programming style

Programming C for the Arduino: Basics

Blink | Arduino 0018

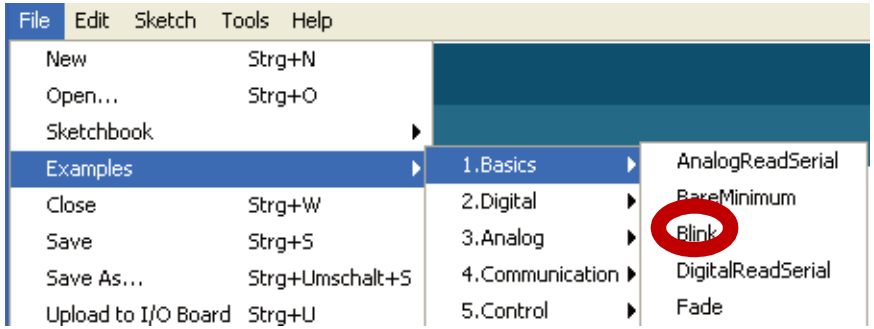
File Edit Sketch Tools Help

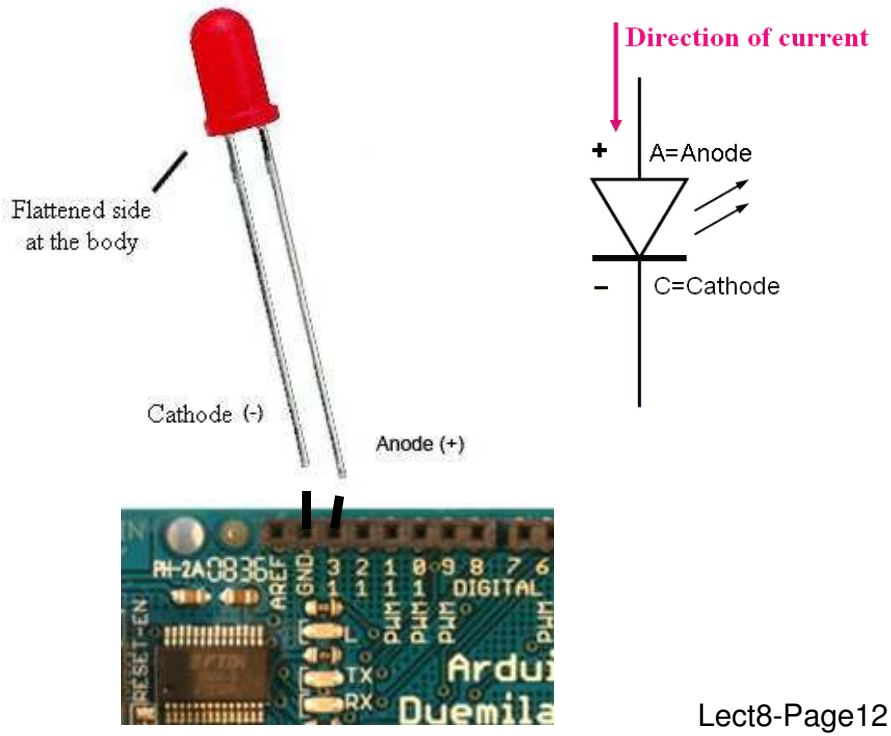
Blink §

```
int ledPin =  13;    // LED connected to digital pin 13

// The setup() method runs once, when the sketch starts

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

// the loop() method runs over and over again,
// as long as the Arduino has power

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

27

Our first program: Blink (LED)





http://www.led-shop.com/images/LED_Anschluss.jpg

Programming C for the Arduino: Basics

Blink | Arduino 0018

File Edit Sketch Tools Help

Blink §

```
int ledPin =  13;    // LED connected to digital pin 13

// The setup() method runs once, when the sketch starts

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

// the loop() method runs over and over again,
// as long as the Arduino has power

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

27

Memory Elements

short, int, long,
float, double,
char,
structures,
…

Variables
Arrays (e.g. int Ar[5];)
Indexes start with 0!!!
Pointers

Programming C for the Arduino: Basics

1 Bit = 1 0 1 1 0 0 1 0 bin = 1 Byte

$0 * 2^0 = 0 * 1 = 0_{dec}$

$1 * 2^1 = 1 * 2 = 2_{dec}$

$0 * 2^2 = 0 * 4 = 0_{dec}$

$0 * 2^3 = 0 * 8 = 0_{dec}$

$1 * 2^4 = 0 * 16 = 16_{dec}$

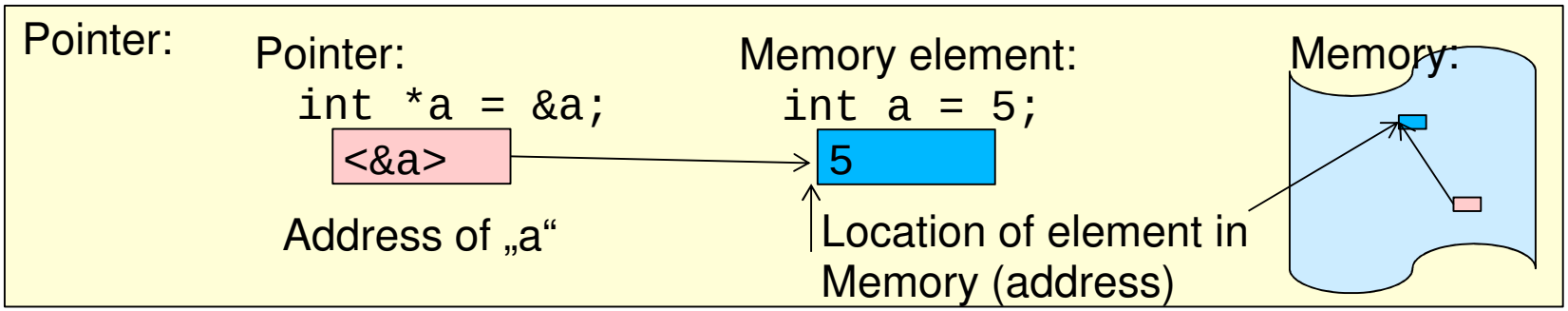
$1 * 2^5 = 0 * 32 = 32_{dec}$

$0 * 2^6 = 0 * 64 = 0_{dec}$

$1 * 2^7 = 0 * 128 = 128_{dec}$

$\sum = 178_{dec}$

Programming C for the Arduino: Basics



Programming C for the Arduino: Basics

Blink | Arduino 0018

File Edit Sketch Tools Help

Blink §

```
int ledPin =  13;    // LED connected to digital pin 13

// The setup() method runs once, when the sketch starts

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

// the loop() method runs over and over again,
// as long as the Arduino has power

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

27

Functions

Functiondefinition

```
int Add (int a, int b)
{
    return a+b;
}
```

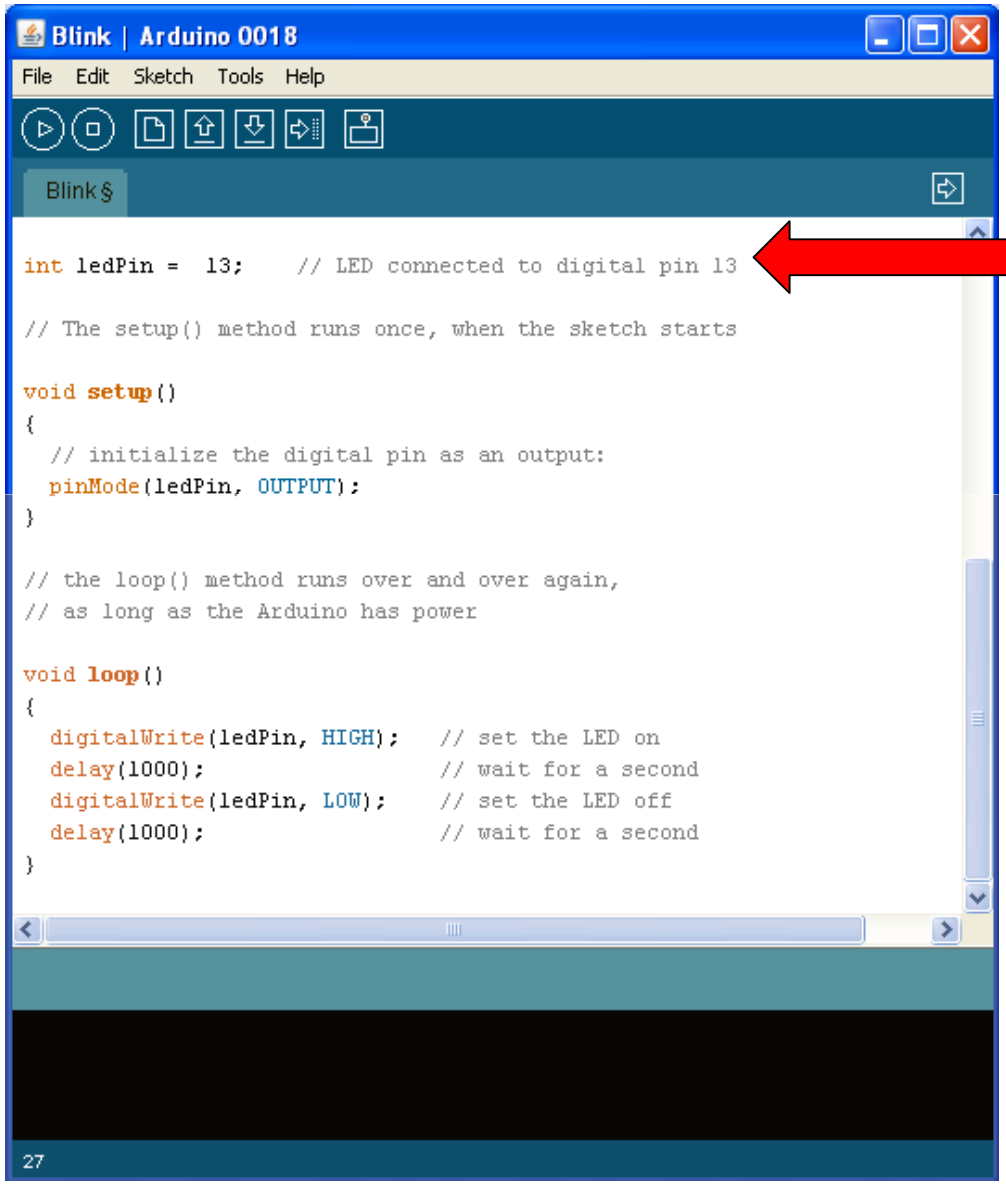
Functioncall

```
X = Add (5, 6);

      => X=11
```

Programming C for the Arduino: Basics

Operators	Standard operator:
Assign	=,
Plus	+,
Minus	-,
Multiplication	*,
Division	/,
Modulo-Div.	%,
AND	&& (Bit-AND &)
OR	|| (Bit-OR |)
NOT	!
	…
	Additional operators:
	Add one, subtract one:
	++,--, (e.g. i++; is i=i+1)
	and a lot of others:
	+=, -=, …, [,], ->, *, …

Programming C for the Arduino: Basics



Comments

// This is a comment

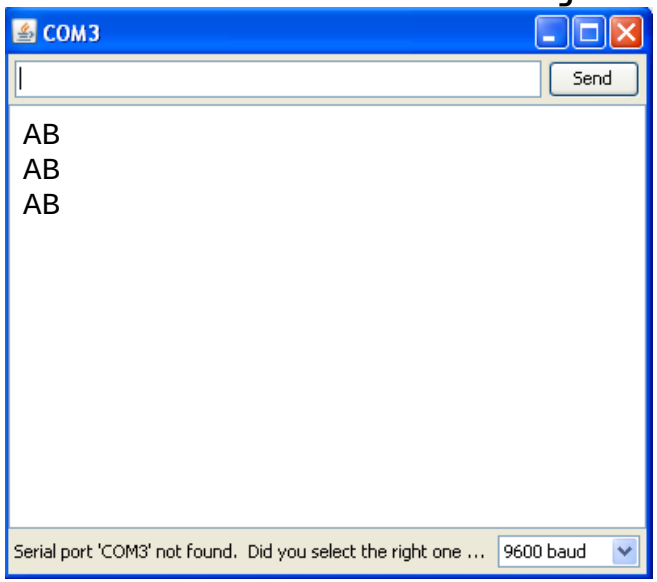
/* This is also a comment */

Programming C for the Arduino: Basics

Interaction with users

Serial write

```
void setup() {
  // initialize serial communication:
  Serial.begin(9600);
}
void loop()
{
  Serial.print("On"); // Write without new line
  Serial.println("Off"); // Wite with new line
}
```



Programming C for the Arduino: Basics

Application workflow

Conditions

```
if (iIndex < 10)
{
    Serial.print("A");
}
else
{
    Serial.print("B");
}
```

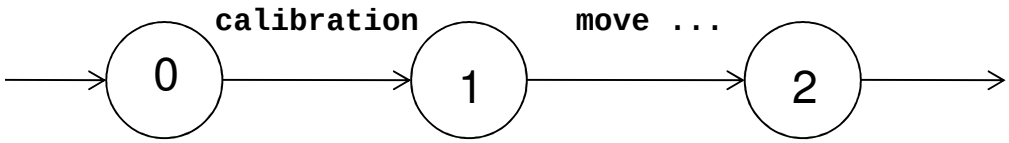
Conditions

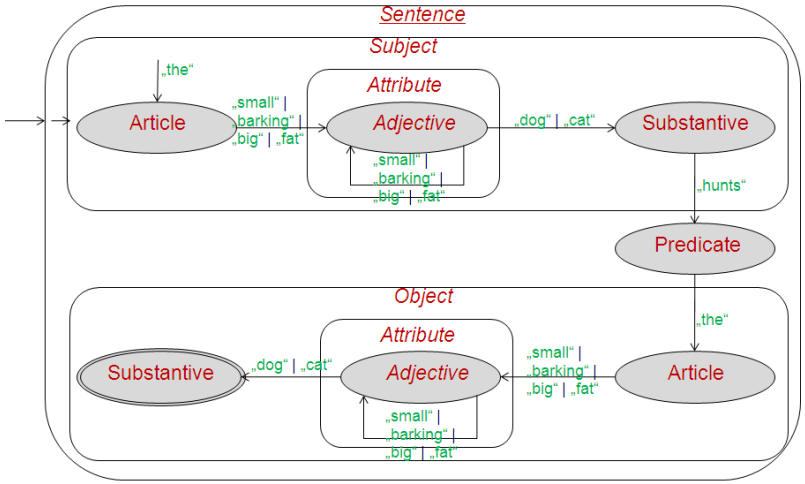
```
switch (iIndex)
{
  case 1:
    Serial.print("A");
    break;
  case 2:
    Serial.print("B");
    break;
  ...
}
```

Programming C for the Arduino: Basics

Realizing a state machine using switch condition

```
int iState;

void setup ()
{
  iState = 0;
}

void loop ()
{
  switch (iState)
  {
    case 0:
       //calibration
       iState = 1;
       break;
    case 1:
       // move just forward as long
       // as wall closer than 10 cm
       iState = 2;
       break;
    case 2:
       ...
  }
}
```





Programming C for the Arduino: Basics

Application workflow

Loops

```
int i = 0;
while (i < 10)
{
    Serial.print(i);
    i = i +1;
}
```

Loops

```
int i;
for (i = 1; i < 10; i++)
{
    Serial.print(i);
}
```

Programming C for the Arduino: Basics

Digital Pins

Output

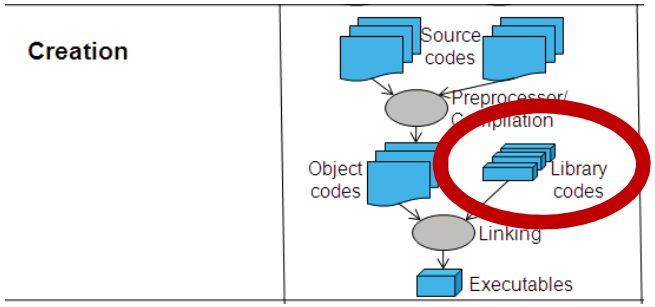
```
pinMode(iPingPin, OUTPUT);
digitalWrite(iPingPin, LOW);
delayMicroseconds(2);
digitalWrite(iPingPin, HIGH);
delayMicroseconds(10);
digitalWrite(iPingPin, LOW);
```

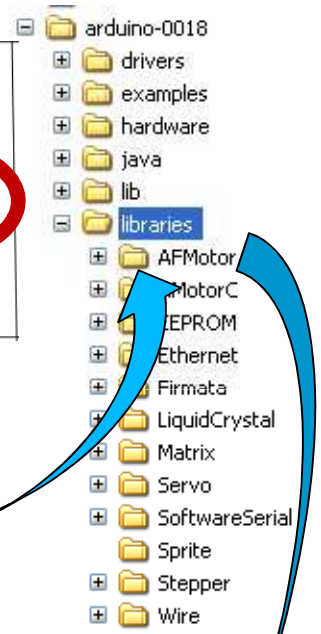
Input

```
pinMode(iPingPin, INPUT);
lDuration = pulseIn(iPingPin, HIGH);
```

Programming C for the Arduino: Basics

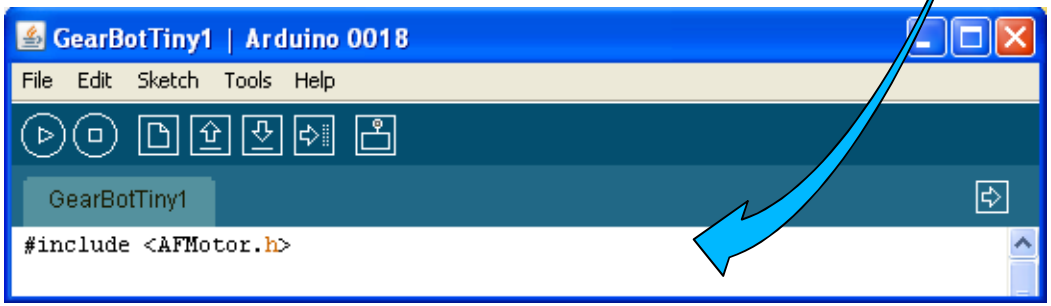
External modules (Libraries)

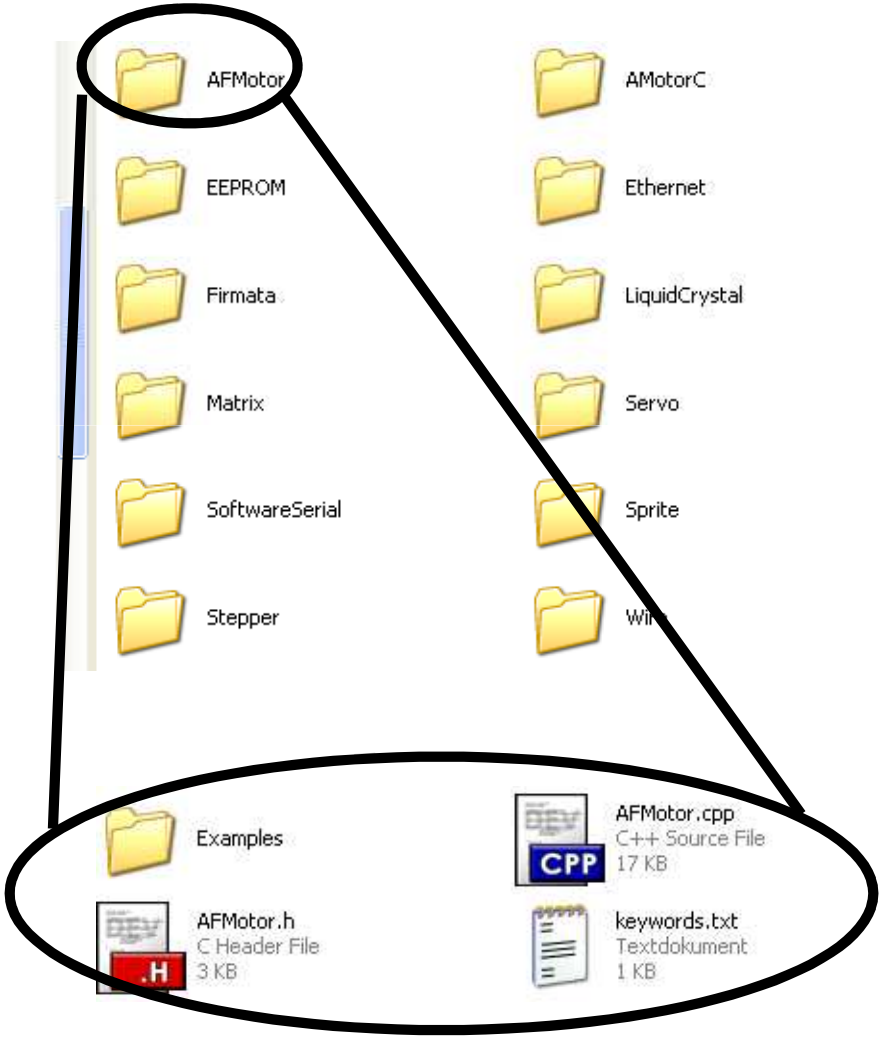




Copy library modules to here

Use module with #include <…>



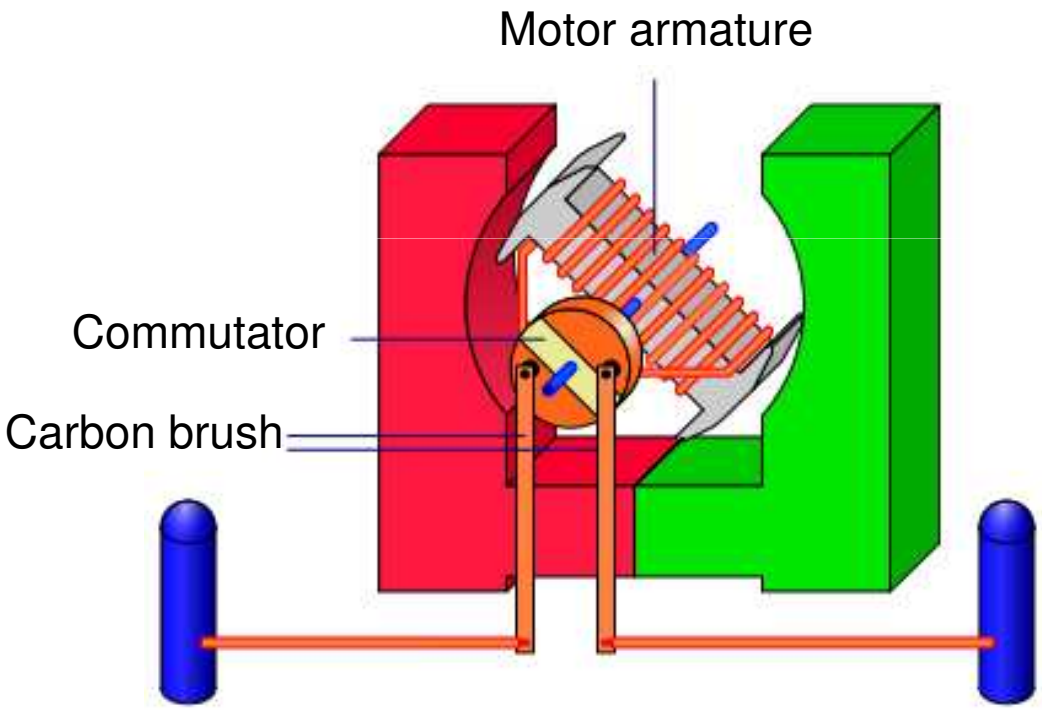


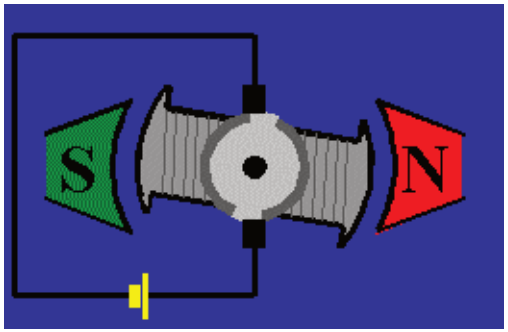
Lecture 8: Programming for the Arduino

- The hardware
- The programming environment
- Binary world, from Assembler to C
- Programming C for the Arduino: Basics
  - Programming C for the Arduino: more …
  - Programming style

Programming C for the Arduino: more ...

Motordriver for DC-motors: basic functionality

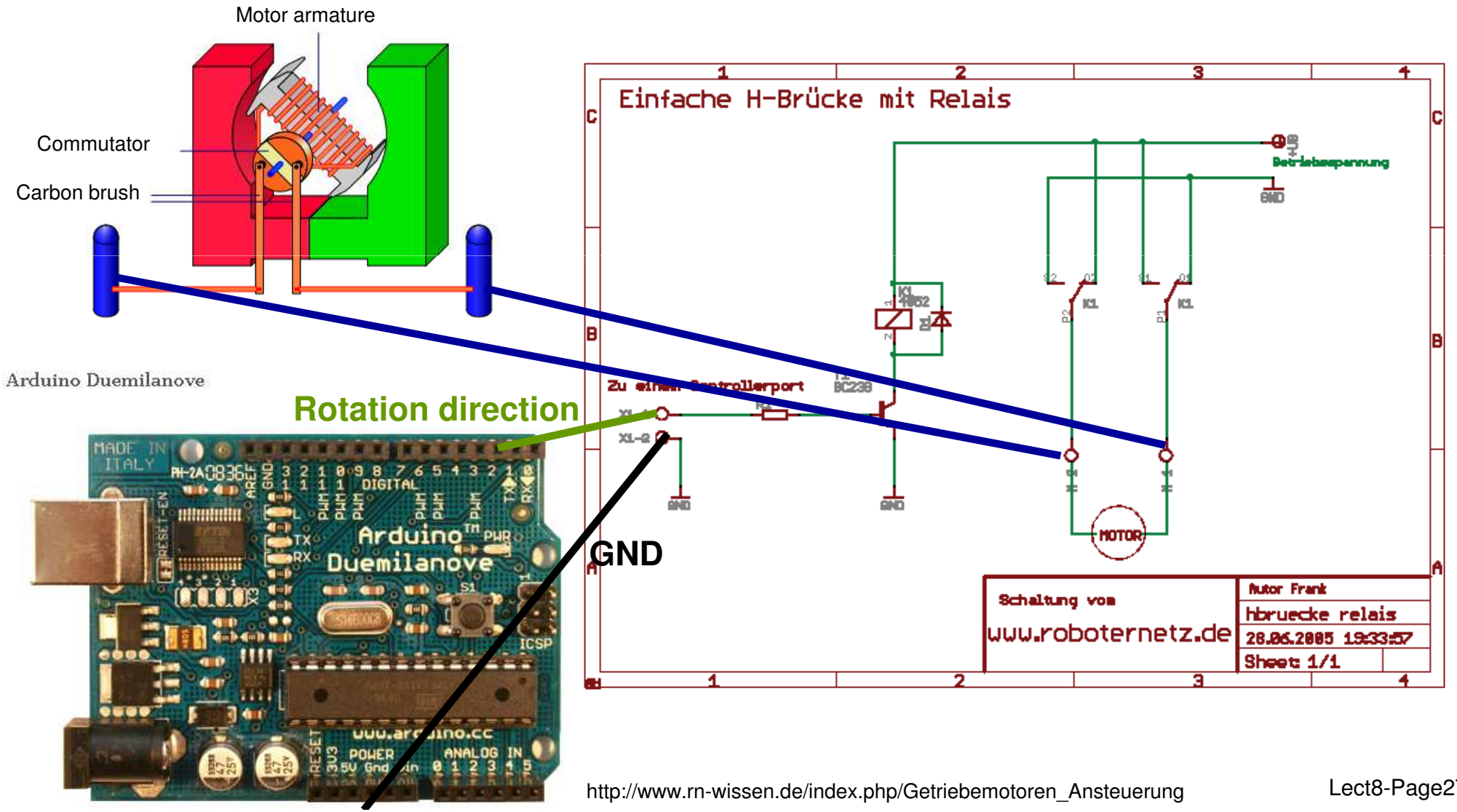




http://de.wikipedia.org/w/index.php?title=Datei:Gleichstrommaschine.svg&filetimestamp=20070820012802
http://www.hpw-modellbahn.de/eisenbahntechnik/motoren.htm

Programming C for the Arduino: more ...

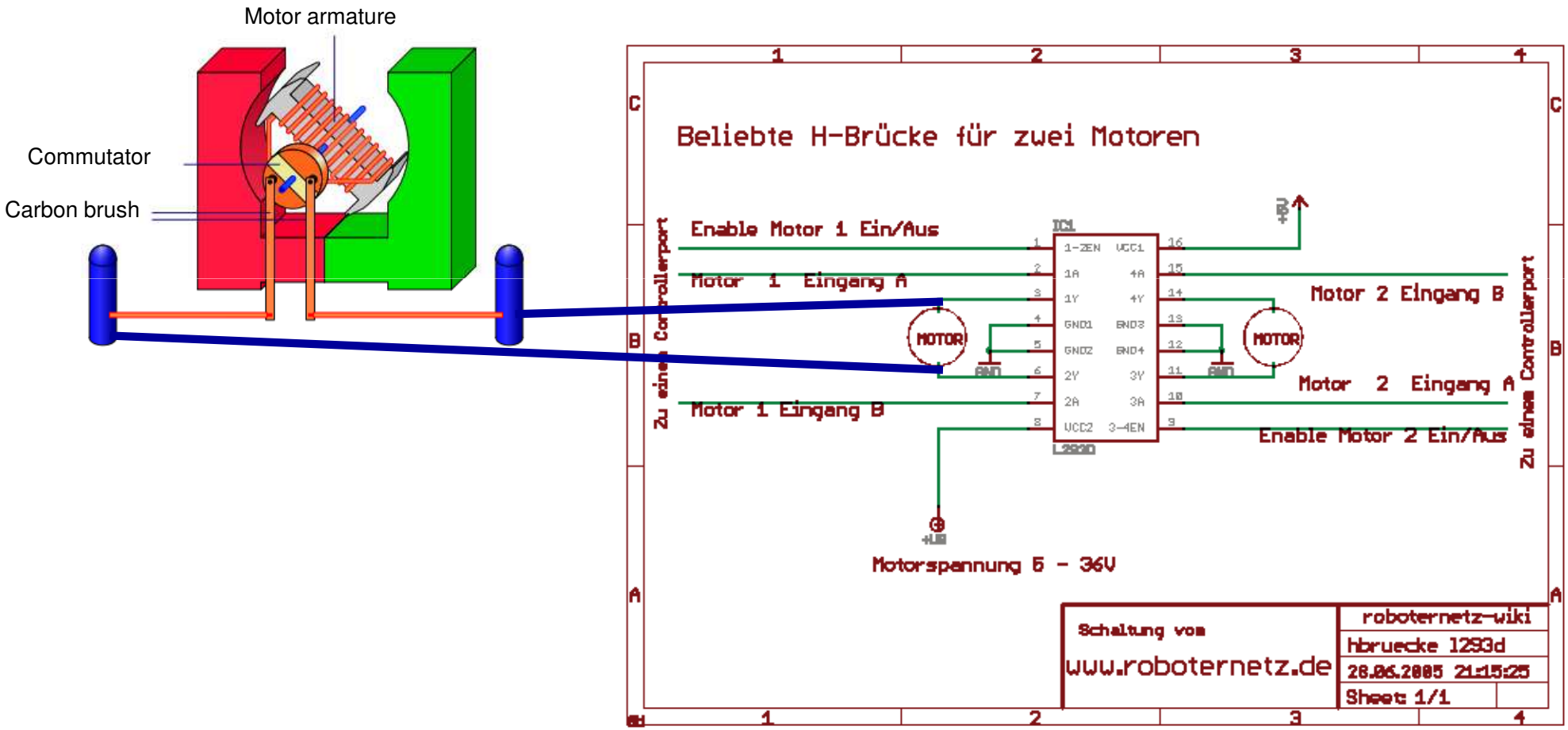
Motordriver for DC-motors: basic functionality



http://www.rn-wissen.de/index.php/Getriebemotoren_Ansteuerung

Programming C for the Arduino: more ...

Motordriver for DC-motors: basic functionality (with driver IC L293 D)



http://www.rn-wissen.de/index.php/Getriebemotoren_Ansteuerung

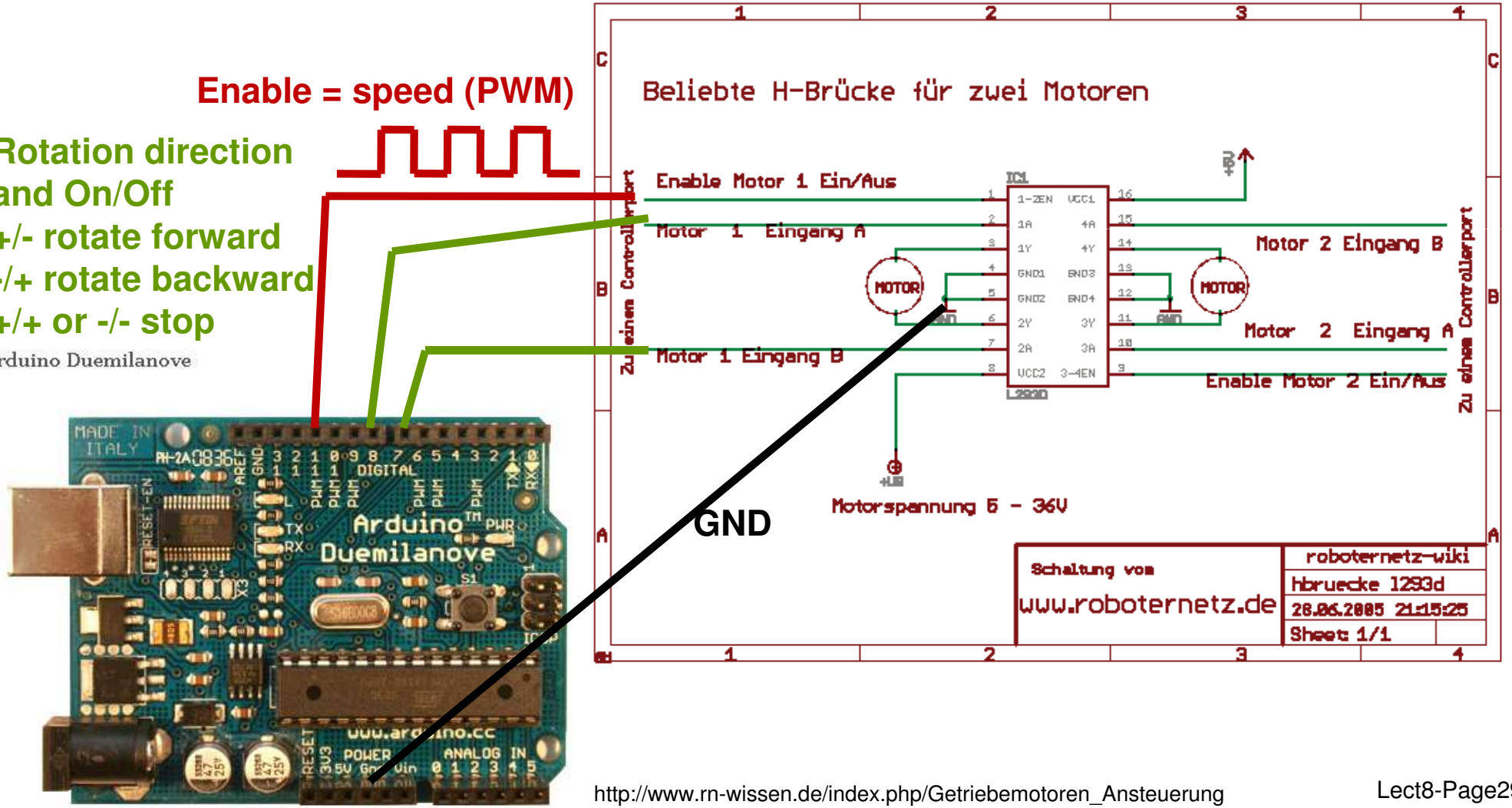
Programming C for the Arduino: more ...

Motordriver for DC-motors: basic functionality (with driver IC L293 D)

Enable = speed (PWM)

Rotation direction and On/Off
+/- rotate forward
-/+ rotate backward
+/+ or -/- stop

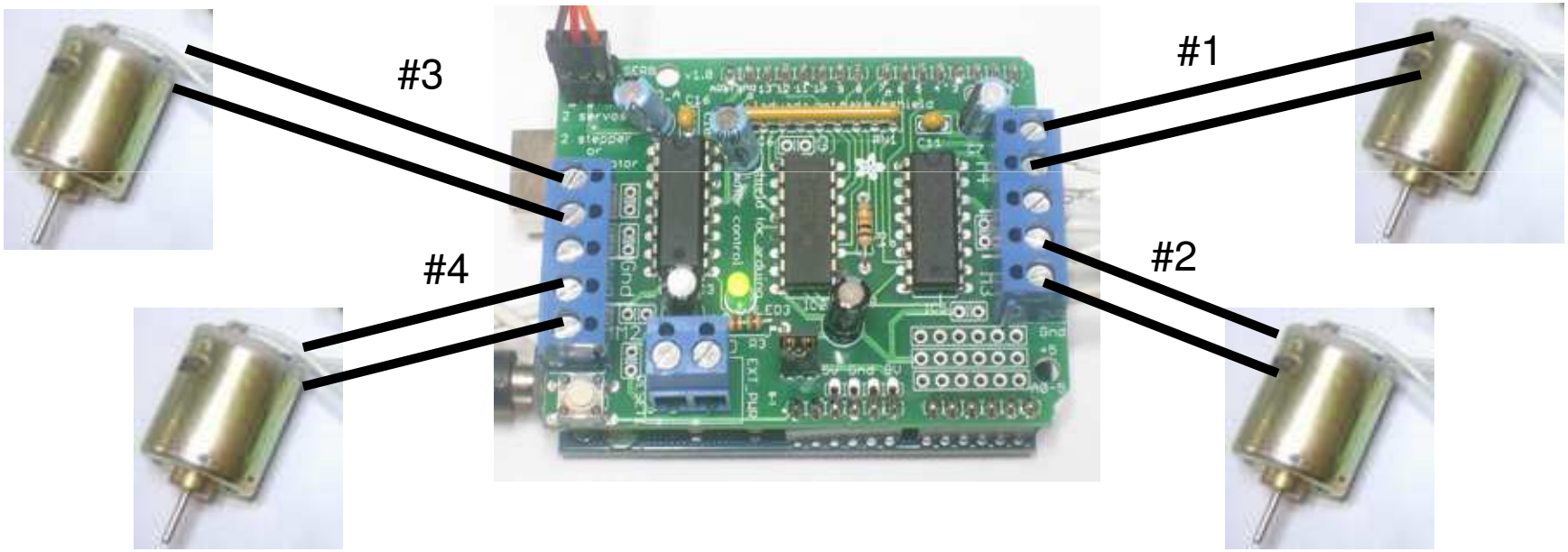
Arduino Duemilanove



http://www.rn-wissen.de/index.php/Getriebemotoren_Ansteuerung

Programming C for the Arduino: more ...

Motordriver for DC-motors:
Adafruit Motor/Stepper/Servo Shield for Arduino kit - v1.0



http://www.adafruit.com/index.php?main_page=product_info&cPath=17_21&products_id=81

Programming C for the Arduino: more ...

Motordriver for DC-motors:
Adafruit Motor/Stepper/Servo Shield for Arduino kit - v1.0

```
// Use Adafruit module
#include <AFMotor.h>

// create motor #2, 64KHz pwm
AF_DCMotor MotorRight(2, MOTOR12_64KHZ);

// set the speed
MotorRight.setSpeed(50);

// Move forward
for (int i=0;i<500;i++)
{
   MotorRight.run(FORWARD);
}
// Move forward
for (int i=0;i<500;i++)
{
   MotorRight.run(BACKWARD);
}
```

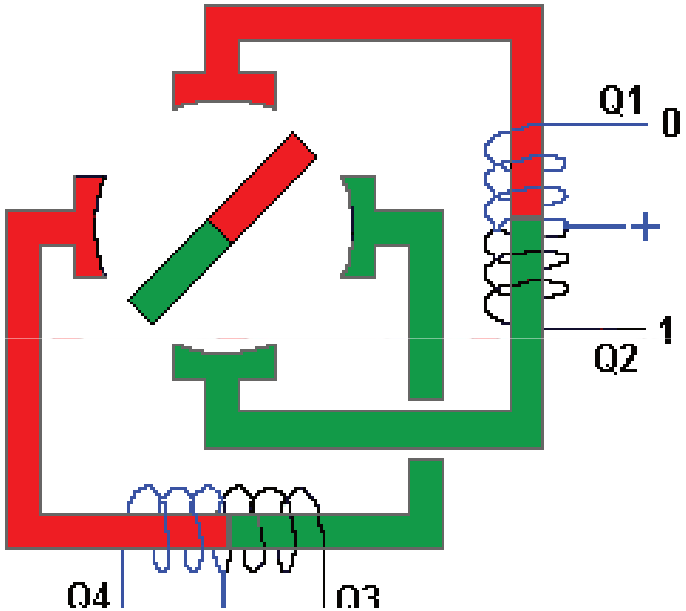
setup()

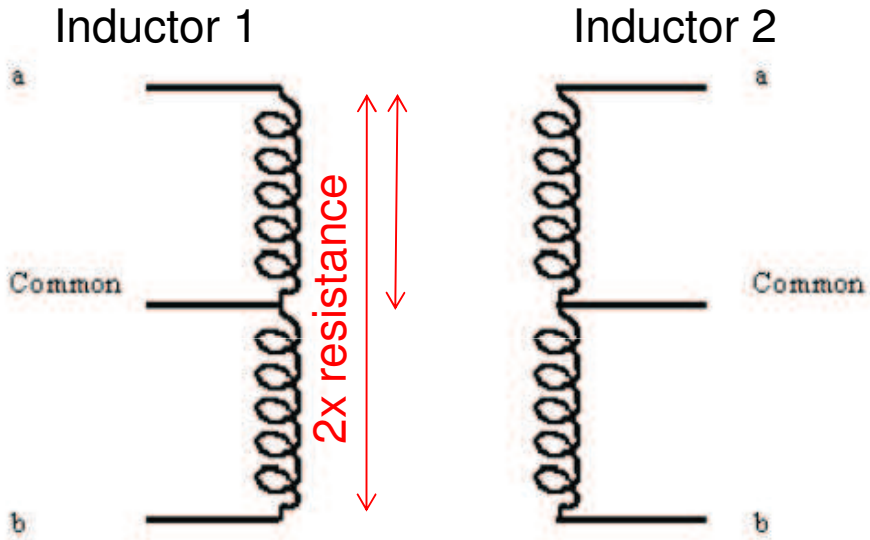
loop()

http://www.ladyada.net/make/mshield/

Programming C for the Arduino: more ...

Motordriver for step-motors (steppers): basic functionality

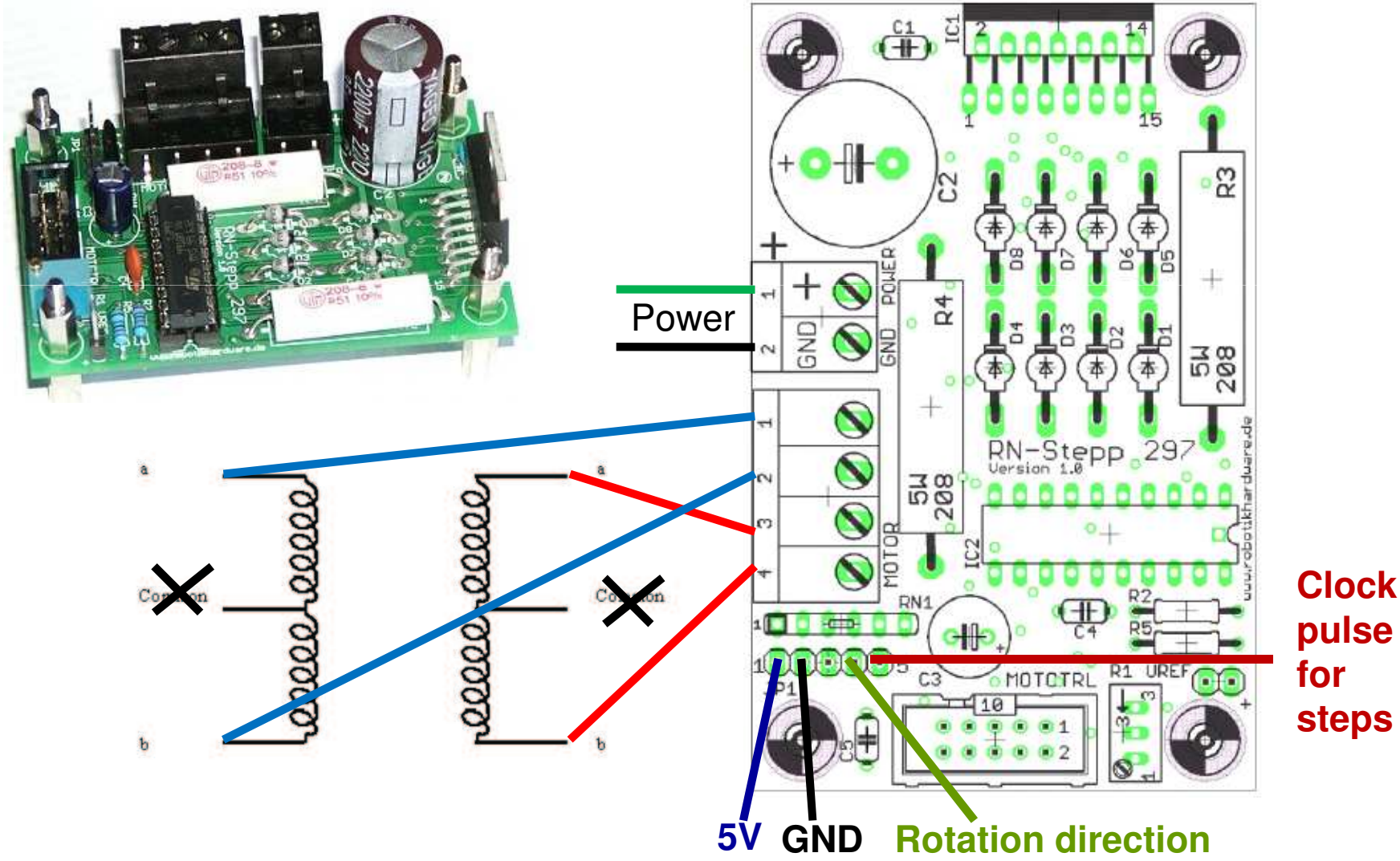




http://www.shop.robotikhardware.de/
http://www.rn-wissen.de/index.php/Schrittmotoren
http://www.schule-bw.de/unterricht/faecher/physik/mess/soundkarte/hardware/interfsound/schrittschlitt.htm

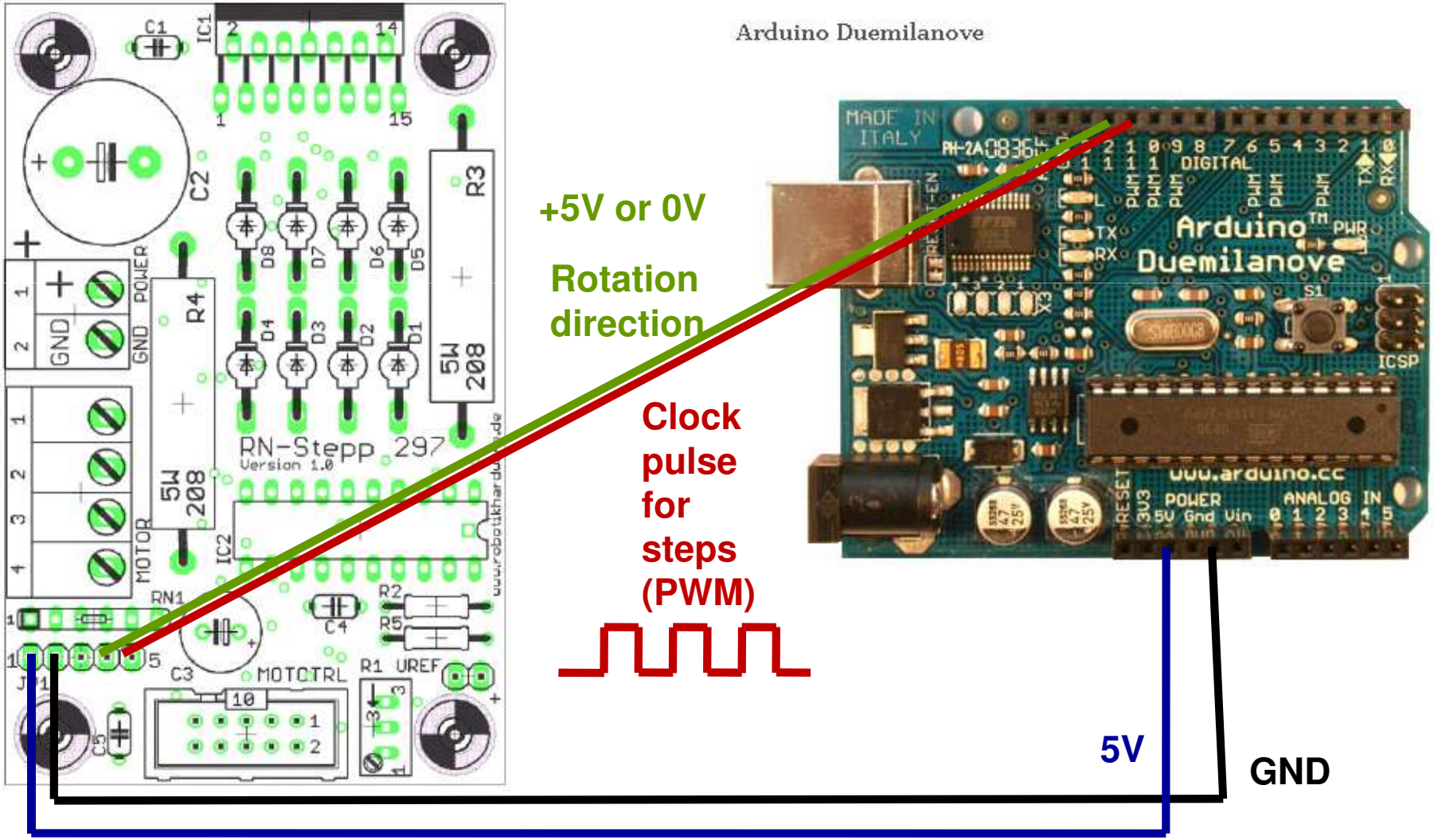
Programming C for the Arduino: more ...

Motordriver for step-motors (steppers): RN-Stepp297



Programming C for the Arduino: more ...

Motordriver for step-motors (steppers): RN-Stepp297



Programming C for the Arduino: more ...

Motordriver for step-motors (steppers): RN-Stepp297 - programming

```
// Use stepper module
#include <Stepper.h>

// Create stepper class variable
Stepper StepperLeft (steps, pin_direction, pin_clockpulse);
 e.g.   StepperLeft  (200, 8, 9)
        StepperRight (200, 7, 10)

// set the speed of the motor to 200 RPMs
StepperLeft.setSpeed(200);
```

setup()

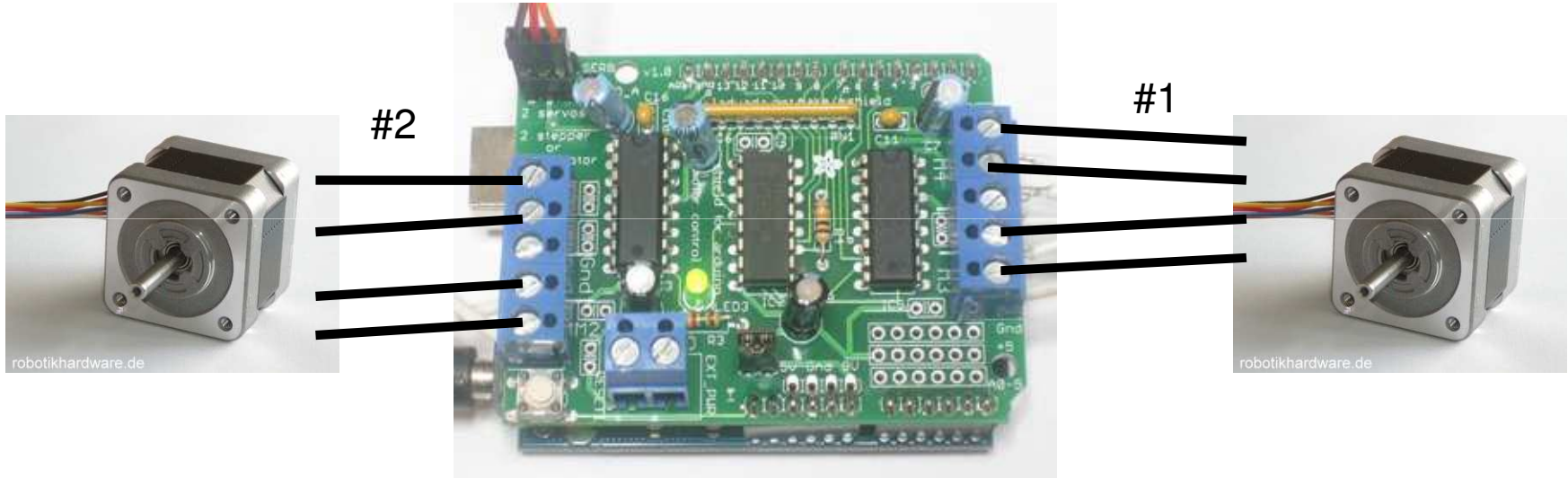
```
// Move one step forward
StepperLeft.step(1);

// Move one step backward
StepperLeft.step(-1);
```

loop()

http://arduino.cc/en/Reference/Stepper?from=Tutorial.Stepper

Programming C for the Arduino: more ...

Motordriver for step-motors (steppers):
Adafruit Motor/Stepper/Servo Shield for Arduino kit - v1.0



Programming C for the Arduino: more ...

Motordriver for step-motors (steppers):
Adafruit Motor/Stepper/Servo Shield for Arduino kit - v1.0

```
// Use Adafruit module
#include <AFMotor.h>

// create motor stepper #2 with 48 steps
AF_Stepper StepperLeft(48, 2);

// set the speed
StepperLeft.setSpeed(60);
StepperLeft.release();

// Move forward
StepperLeft.step(1, FORWARD, SINGLE);

// Move backward
StepperLeft.step(1, BACKWARD, SINGLE);
```

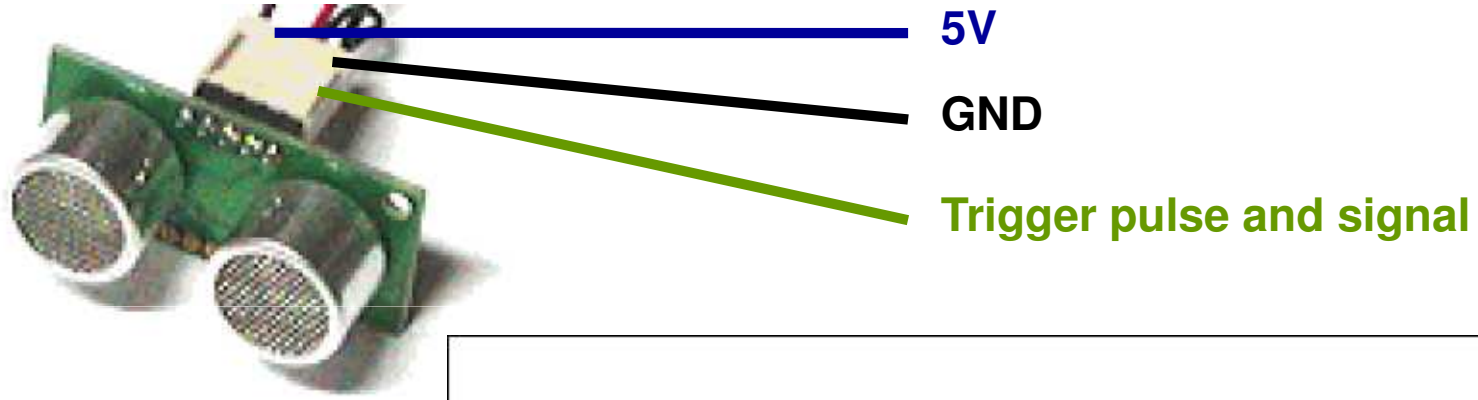
setup()

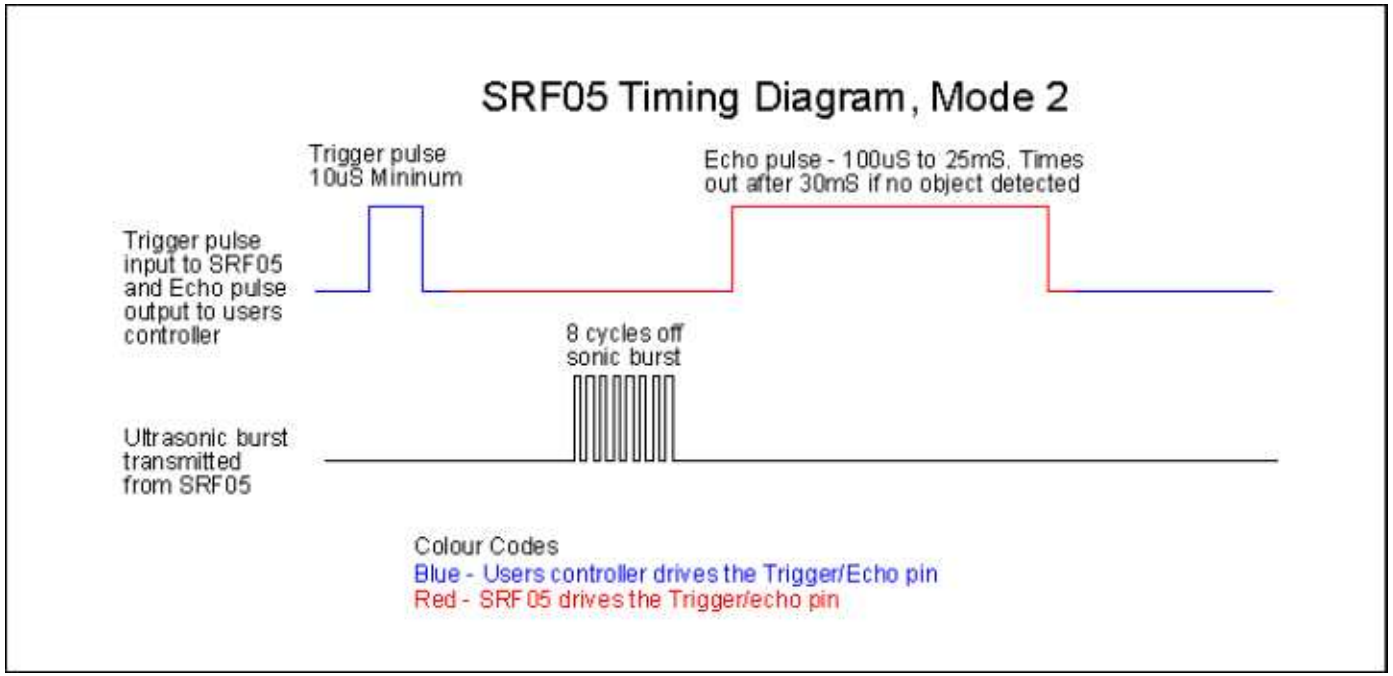
loop()

http://www.ladyada.net/make/mshield/

Programming C for the Arduino: more ...

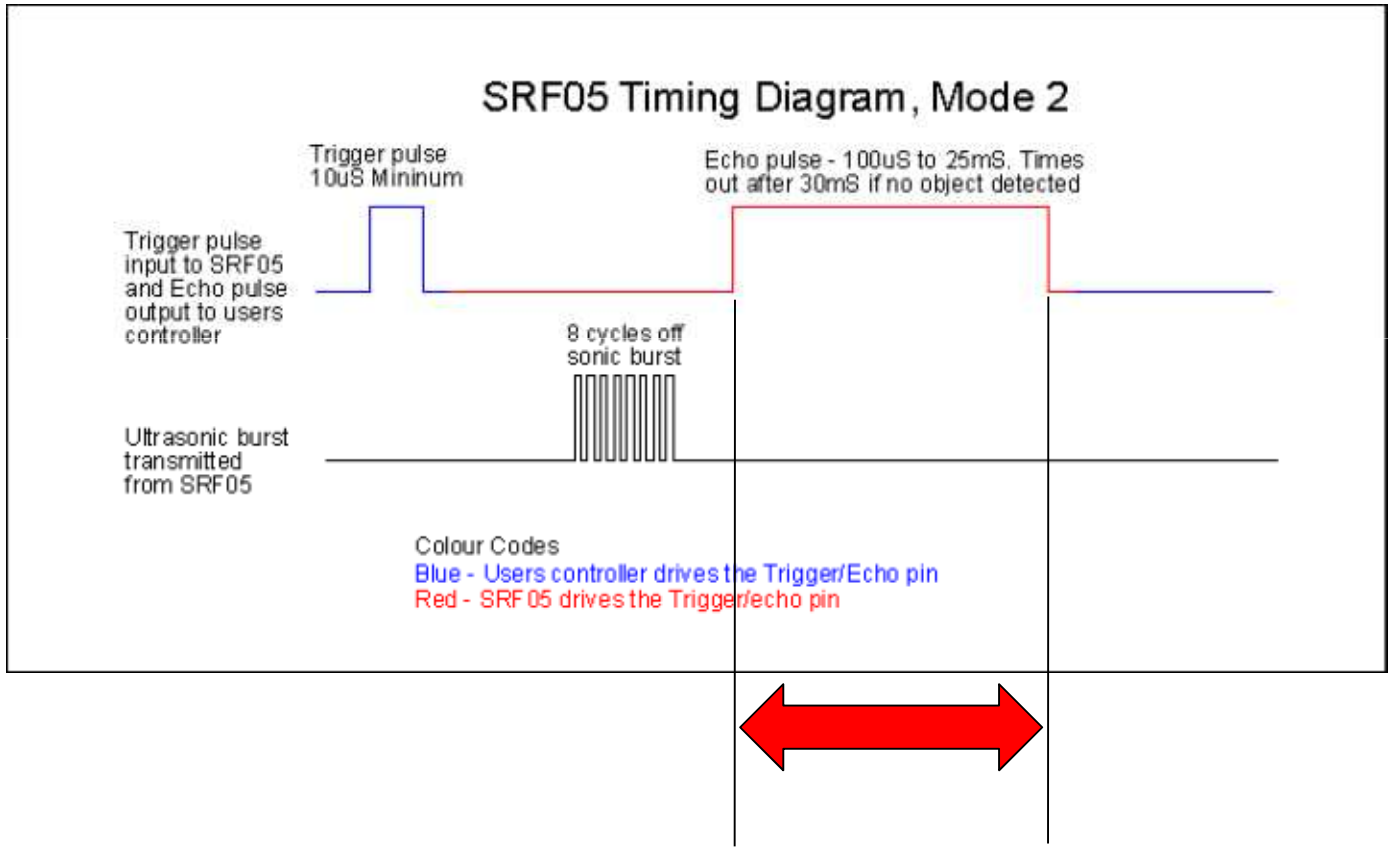
Low Cost Ultra Sonic Range Finder





Programming C for the Arduino: more ...

Low Cost Ultra Sonic Range Finder

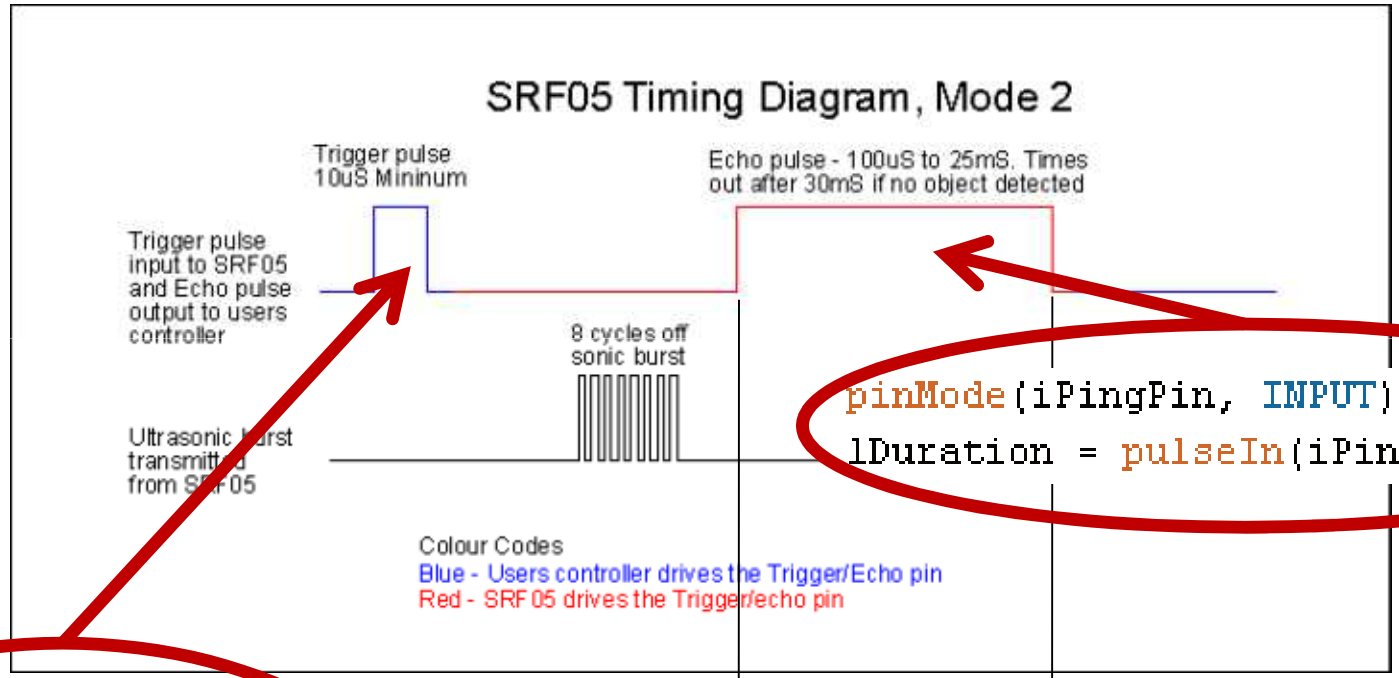


Range = uSec/58=cm or uSec/148=inches.

Programming C for the Arduino: more ...

Low Cost Ultra Sonic Range Finder

```
e.g.   int iPingPin = 13;
```



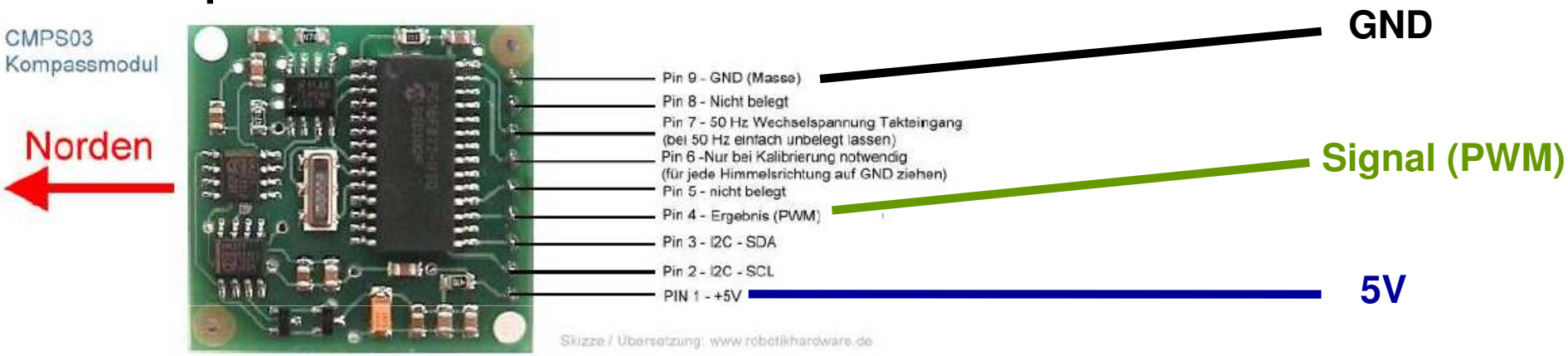

```
pinMode(iPingPin, INPUT);
lDuration = pulseIn(iPingPin, HIGH);
```

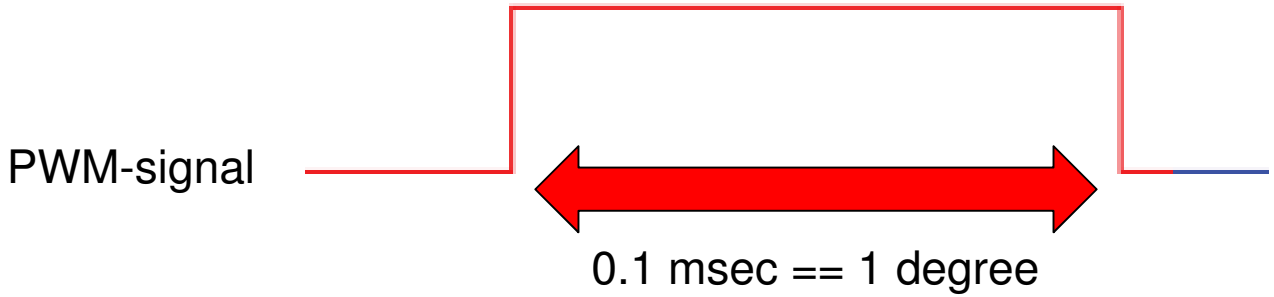
```
pinMode(iPingPin, OUTPUT);
digitalWrite(iPingPin, LOW);
delayMicroseconds(2);
digitalWrite(iPingPin, HIGH);
delayMicroseconds(10);
digitalWrite(iPingPin, LOW);
```

Range = uSec/58=cm or uSec/148=inches.

Programming C for the Arduino: more ...

Robot Compass Modul



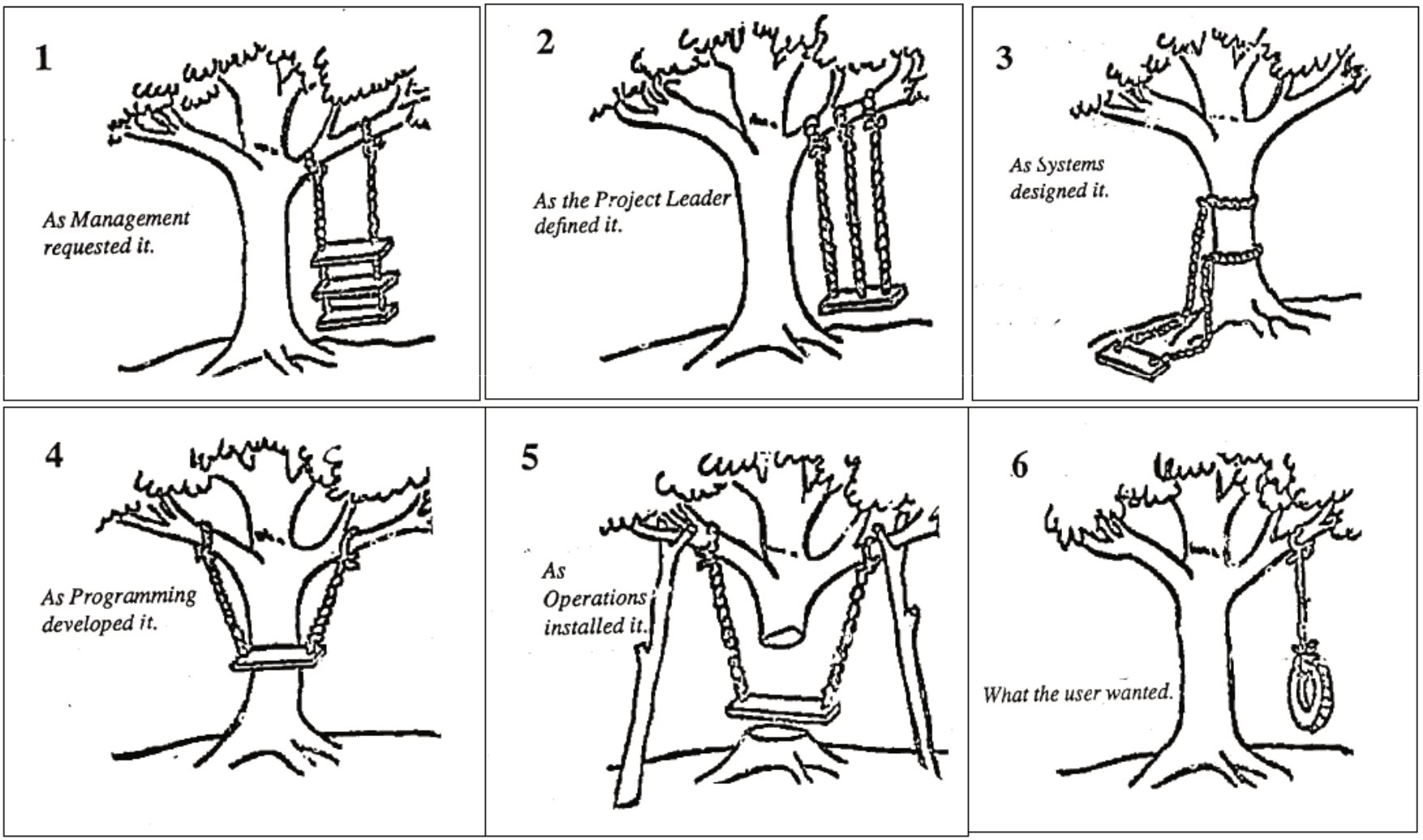



```
e.g.   int iCompassPin = 11;
```

Lecture 8: Programming for the Arduino

- The hardware
- The programming environment
- Binary world, from Assembler to C
- Programming C for the Arduino: Basics
- Programming C for the Arduino: more …
  - Programming style

Programming style



See: *http://casablanca.informatik.hu-berlin.de/database/MDA-UML/VL/V1-MODSOFT-I.1.Ueberblick.pdf*, Download 07.11.2006

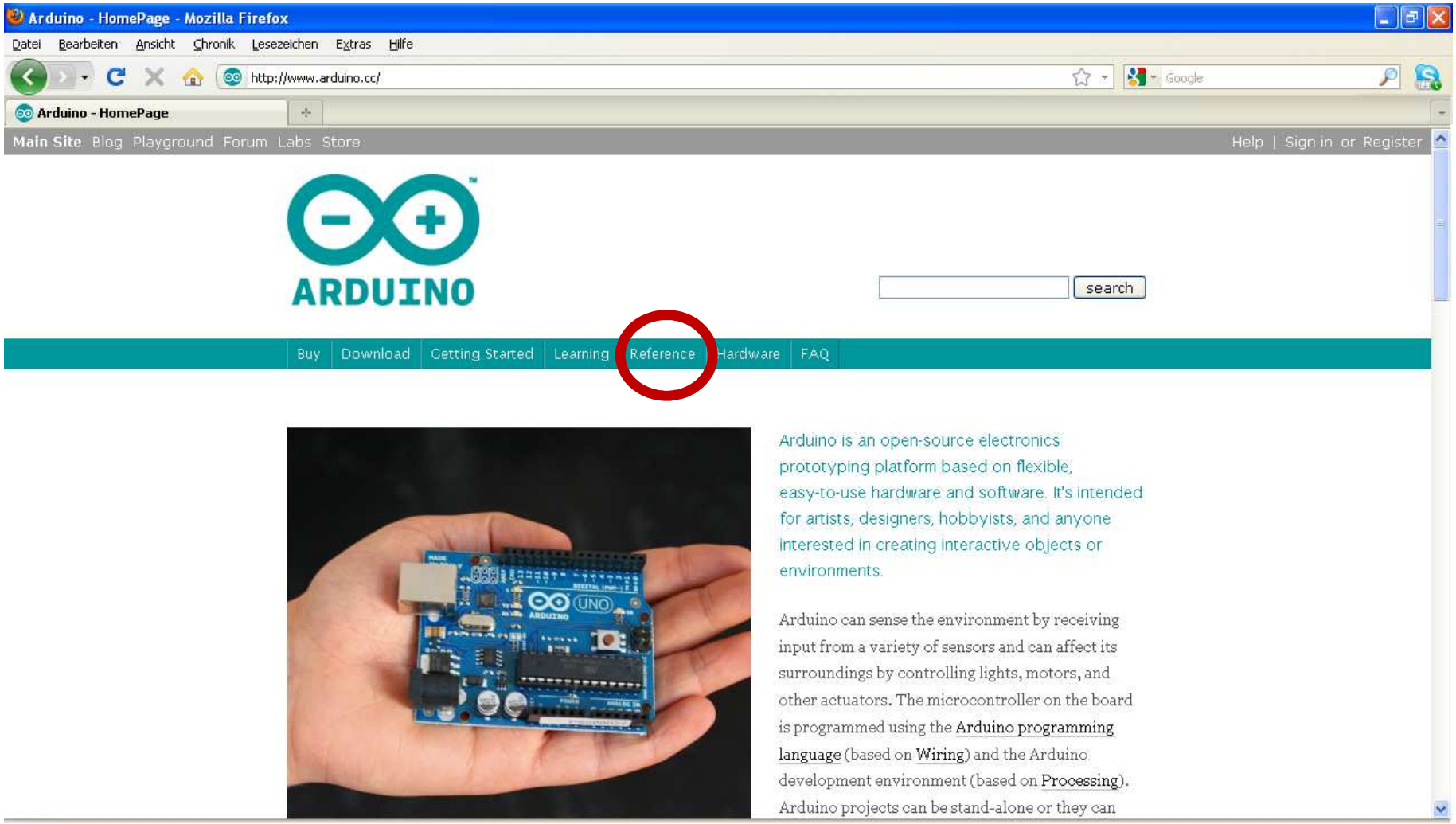
Programming style

Try to write readable and efficient code:

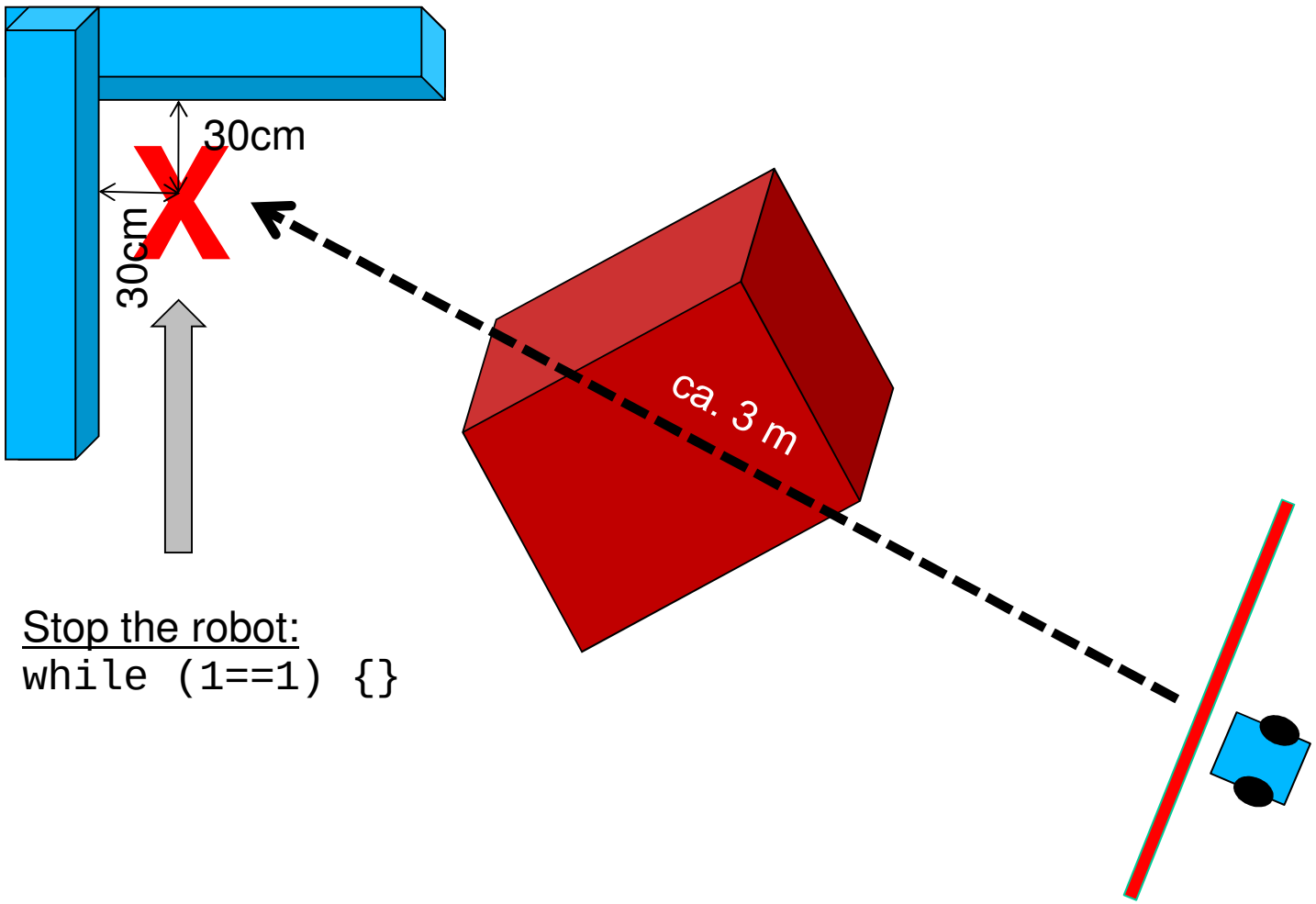
- Add comments to explain the code
- Use understandable (variable, function) names
- Structure the code into logical units
- Initialize variables
- Search for existing code first
- …

Programming style

And for help look at:



The goal!



<u>Stop the robot:</u>

```
while (1==1) {}
```

Thank you