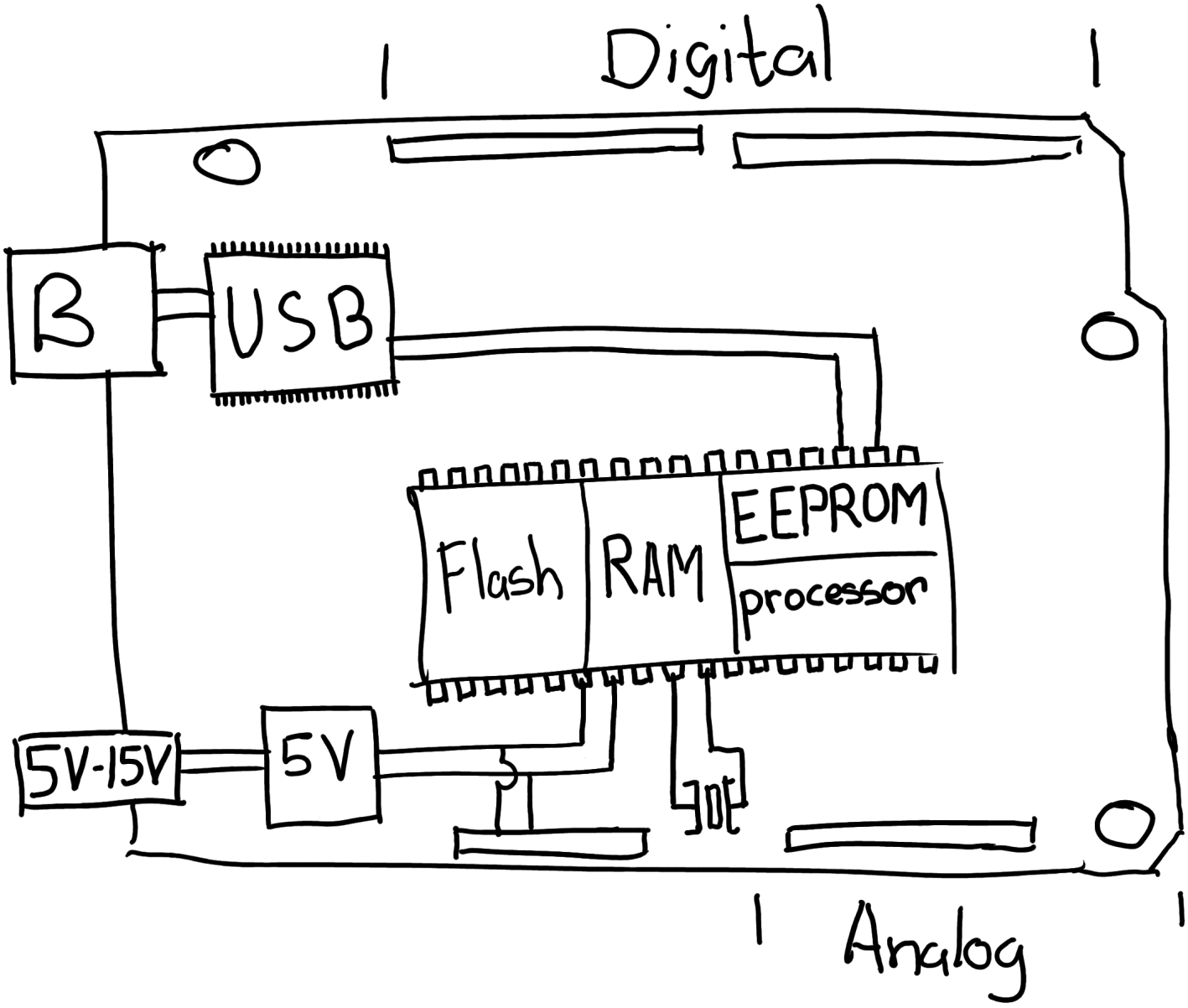
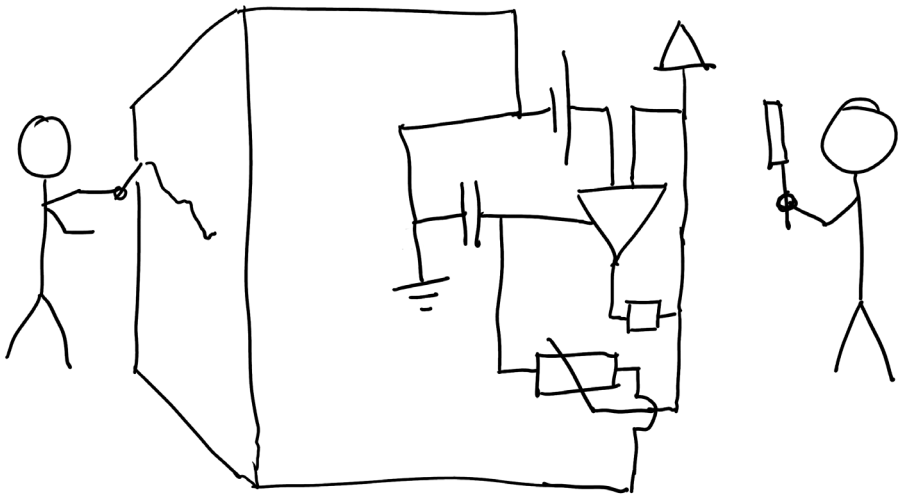
Arduino Workshop

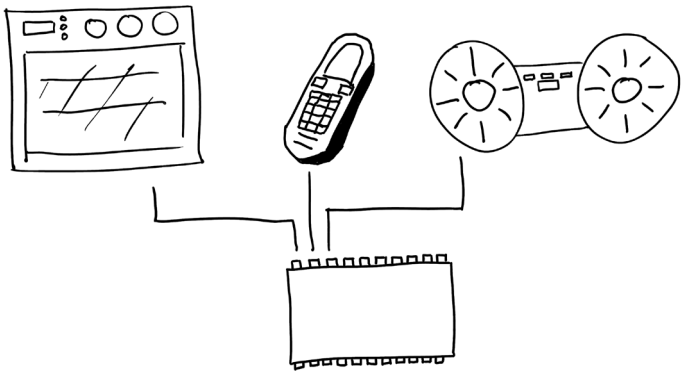
door Geert Roumen
(CC) BY-SA

lemio.nl/arduino/





Arduino?



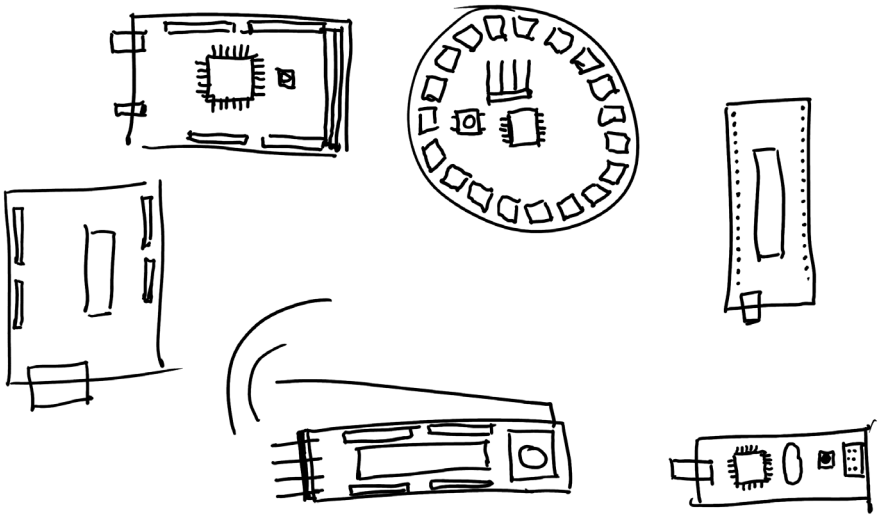
Arduino is een open-source ontwikkelbord, waarmee het mogelijk wordt om informatica te koppelen aan fysieke objecten. Het is dus een apparaat om fysische informatica makkelijk en toegankelijk te maken. De Arduino wordt dan ook door verschillende groepen gebruikt, hij is oorspronkelijk ontworpen voor designers om in hun kunst interactie te kunnen verwerken. Daarna zijn ook andere hobbyisten, en Industrial Designers de Arduino gaan gebruiken. Hij wordt nu zelfs door elektrotechnici gebruik omdat 'hij zo gemakkelijk werkt', en er heel veel uitbreidingen te vinden zijn op het internet.
Het is dus eigenlijk een kleine computer die fysieke componenten kan aansturen. Je kunt er dus hele robots mee bouwen, of ermee het internet op gaan. Eigenlijk is het dus een makkelijk te programmeren stukje hardware die vrijwel alles kan. Van een vliegtuig aansturen tot het brein zijn van een 3D printer.

Waarom hoort Arduino in het onderwijs thuis?

Je kunt je afvragen leuk allemaal, maar waarom zou ik mijn leerlingen dit leren? Het antwoord op die vraag ligt voor de hand, iedereen gebruikt elke dag wel een object waarin een microchip zit. Van een koffiezetapparaat tot een afstandsbediening. Door te leren hoe deze apparaten werken, hoe ze communiceren en hoe ze in elkaar zitten begrijp je veel van de wereld om je heen. Naast dat het gewoon 'vet' is doordat de grens (tot hoever je kunt denken, en kunt uitvoeren) verlegd wordt. Je kunt opeens veel meer wanneer je microcontrollers beheerst.

Microcontrolers bestaan toch al lang, waarom dan een Arduino?

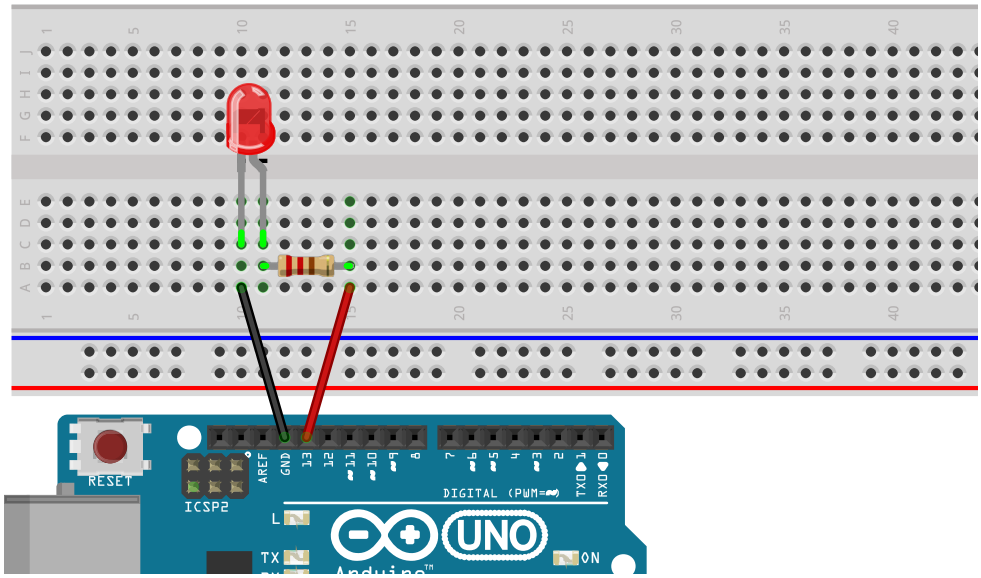
Een Arduino is een stukje hardware dat gemakkelijk te programmeren is, de meeste microcontrolers zitten in de apparaten verwerkt en zijn OTP (One Time Programmable). Nu zijn er een



paar microcontrolers speciaal voor hobbyisten gemaakt, dit zijn onder andere de AVR series (van Atmel), PIC series (van Microchip) en STAMP. Nu is het zo dat je wanneer je een ruw chipje wilt programmeren, je eerst veel verstand moet hebben van externe componenten zoals kristallen etc. en je een duur programmeer apparaat nodig hebt. Ook is het zo dat je losse chipjes veel sneller opblaast, of de interne instellingen (fuses) verandert. Zo heb ik nu ook een kerkhof aan Atmel chipjes liggen waarvan de interne instellingen verkeerd zijn ingesteld. Bij Arduino heb je deze problemen allemaal niet, je koopt gewoon een Arduino. Deze is gebaseerd op de AVR 8-bit serie en heeft een ingebouwde USB poort, een bootloader, 5V voeding en 16Mhz kristal. Je kunt dus in principe zo beginnen, zonder veel verstand van fysische informatica te hebben. Het is zelf zo simpel dat basisschool kinderen ermee aan de slag kunnen (met behulp van ardublocks of modkit)

Waarom is opensource hardware zo geweldig?

Het voordeel van Open Source hardware is dat iedereen zijn eigen idee kan uitvoeren. Heb jij bijvoorbeeld het idee om een Arduino variant te maken die i.p.v. USB geluid gebruikt om geprogrammeerd te worden, perfect: ‘the SoundDuino’ is geboren. En zo zijn alle Arduino’s ontstaan. Je hebt nu bijvoorbeeld de Arduino Nano (voor in breadboards), Arduino Mega (met meer pinnen), Lilypad (voor in kleren), JeeNode (via draadloze signalen programmeerbaar), Sanguino (gebaseerd op een andere chip), NodeMCU (gebaseerd op de ESP8266, ESP32) een zeer goedkope mircocontroller met wifi etc... Verder zijn er ook nog dingen als shields, een geweldige uitvinding. Je plugt simpelweg een shield op je Arduino en deze kunnen dan met elkaar communiceren. Net alsof je een USB apparaat in je PC stopt. Je hebt alleen de goede libraries (drivers) nodig. Voorbeelden van shields zijn het Ethernet shield (de naam zegt het al), Motor shield, prototype shield, USB shield, TellyMate shield, VirtualWire Shield, wave shield (om complexe geluiden of muziek af te spelen), GPS shield, RAMPS shield (voor 3D printers) etc... Je kunt je dus voorstellen dat mede dankzij dat Arduino open-source is, het zo’n succes is. De community groeit alsmaar, dus de projecten worden steeds groter en uitdagender, dus doe mee!



/software

Wanneer Arduino al geïnstalleerd is op je computer hoeft je alleen Arduino IDE (de Arduino software) te starten. De example code te openen (file>examples>blink) en de juiste hardware te selecteren via (tools>board>) en via (tools>port>COM# (Arduino).
Nu is het alleen nog maar een kwestie van uploaden. Dit doe je door op de 'upload' knop te drukken. (Zie afbeelding.)

A B
(A) Compile: test het programma op fouten zonder te uploaden naar de Arduino
(B) Upload: upload/verzend het programma naar de Arduino

/code

```
void setup() {
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);
   //Zet LED aan
  delay(1000);
   //Wacht 1000 miliseconden (1s)
  digitalWrite(13, LOW);
   //Zet LED uit.
  delay(1000);
   //Wacht 1000 miliseconden (1s)
}
```

/hardware

In dit geval zal je niet bijzonder veel hardware nodig hebben aangezien er in de meeste Arduino boards al een LED op pin 13 is aangesloten met een weerstandje. Mocht je meer licht willen dan dat kleine ingebouwde LEDje dan kan je een LED lampje aansluiten. Wanneer je een ledje aansluit op een pin moet je een weerstandje in serie met het ledje zetten. Anders veroorzaakt het ledje overload of kortsluiting.

/breadboard

Het witte ding wat je voor je ziet is een breadboard, een breadboard is een bordje, vaak van wit of geelplastic, dat gebruikt wordt om elektrische schakelingen tijdelijk op te bouwen. Op de afbeelding is te zien hoe de gaten zijn doorverbonden in het breadboard. De horizontale verbindingen worden vaak voor de voeding gebruikt, en de verticale verbindigen voor componenten.

horizontale verbindingen

verticale verbindingen

/LEDjes

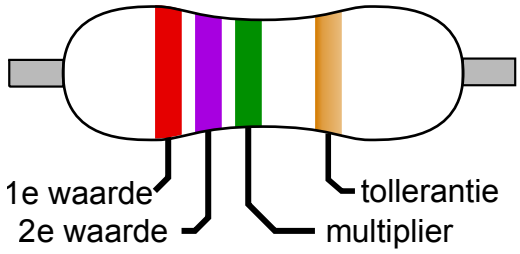
LEDjes (licht-emitterende diodes) zijn kleine lampjes die wanneer er stroom via de doorlaatrichting stroomt licht gaan geven.

Dit is de min-kant van het ledje (cathode), deze verbind je met GND (deze kant is vlak)

Dit is de pluskant van het ledje, (anode) deze verbind je met D13 (deze kant is bol)

/weerstand

Een weerstand is een component om de stroom door een circuit te limiteren. Er wordt een kleurcodering gebruikt om de waarde van de weestand aan te geven.



Color	Significant figures	Multiplier	Tolerance		Temp. Coefficient (ppm/K)	
Black	0	$\times 10^0$	–		250	U
Brown	1	$\times 10^1$	±1%	F	100	S
Red	2	$\times 10^2$	±2%	G	50	R
Orange	3	$\times 10^3$	–		15	P
Yellow	4	$\times 10^4$	(±5%)	–	25	Q
Green	5	$\times 10^5$	±0.5%	D	20	Z
Blue	6	$\times 10^6$	±0.25%	C	10	Z
Violet	7	$\times 10^7$	±0.1%	B	5	M
Gray	8	$\times 10^8$	±0.05% (±10%)	A	1	K
White	9	$\times 10^9$	–		–	
Gold	–	$\times 10^{-1}$	±5%	J	–	
Silver	–	$\times 10^{-2}$	±10%	K	–	
None	–	–	±20%	M	–	

void setup(){...}

De code binnen de {...} wordt alleen aan het begin uitgevoerd. Dit is wanneer de Arduino gereset wordt.

void loop(){...}

De code binnen de {...} wordt als een oneindige loop uitgevoerd.

pinMode(pin, mode)

Met pinMode definieer je een bepaalde pin als in of outgang. Mode kan INPUT of OUTPUT zijn.

digitalWrite(pin, output)

Hiermee bepaal je de waarde van een bepaalde pin. Output kan HIGH of LOW zijn.

delay(time)

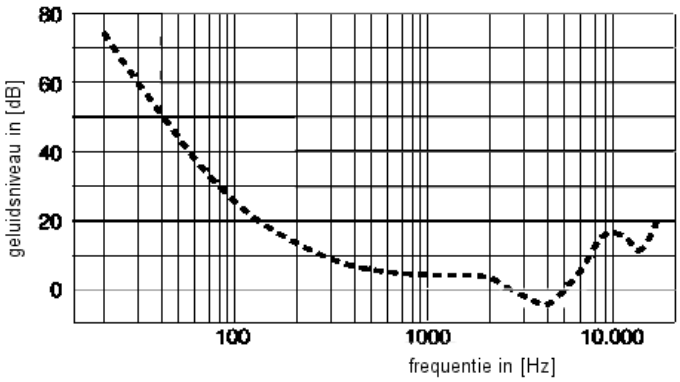
Hiermee laat je het programma pauzeren, de tijd wordt aangegeven in milliseconden.

delayMicroseconds(time)

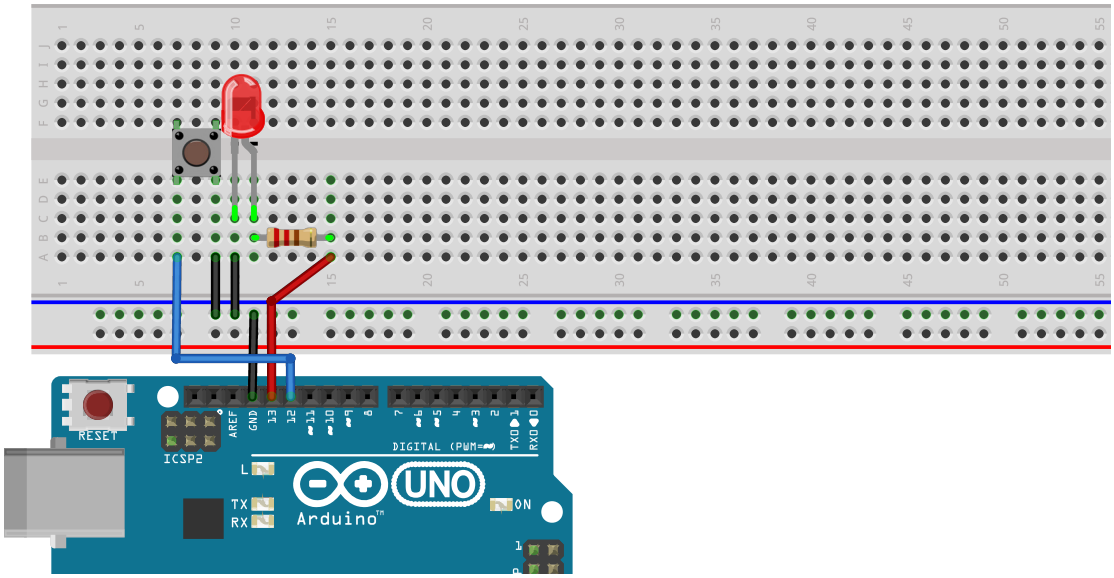
Zelfde als delay alleen dan in mircoseconden.

/geluid

De natuurkundigen onder ons hebben het misschien al bedacht, maar wanneer je een signaal snel aan en uit kan doen, zou je hier in principe een speaker op aan kunnen sluiten. Klinkt dat als een uitdaging voor jou? Probeer dat dan. (denk aan de formule t(in sec)=1/f(in Hz), het best hoorbaar geluid is 200Hz - 6000Hz). Wat ook interessant is om te kijken wanneer mensen knipperend licht als continu licht gaan zien, denk aan het lichtnet.



De gemiddelde gehoordrempel van jonge, gezonde mensen, CC BY-SA 3.0, Ellywa



/sensors

Het is natuurlijk geweldig wanneer we onze projecten ook op de omgeving kunnen later reageren. Dit gebeurt in de elektronica via sensoren. Er zijn tal van sensoren (licht, druk, magnetisme, temperatuur, versnelling, draaiing, geluid...) wij gaan nu een project bouwen dat werkt met een een drukknopje.

/digitaal

Een digitale sensor is simpelweg een sensor die aan of uit is, geen andere waarde is mogelijk. Een voorbeeld hiervan is een drukknopje. Deze maakt contact of niet, maar niets ertussenin.

/variablen

Je kunt bepaalde waarden door de Arduino (tijdelijk) laten opslaan. Deze waarden kun je ook in het programma aanspreken.

/code

```
int buttonPin = 12;
int ledPin = 13;
void setup() {
  pinMode(buttonPin,INPUT);
  digitalWrite(buttonPin,HIGH);
  pinMode(ledPin,OUTPUT);
}

void loop() {
  if (digitalRead(buttonPin)==HIGH){
    digitalWrite(ledPin,HIGH);
    delay(20);
    digitalWrite(ledPin,LOW);
    delay(20);
  }else{
    digitalWrite(ledPin,HIGH);
    delay(1000);
    digitalWrite(ledPin,LOW);
    delay(1000);
  }
}
```

digitalRead(pin)

Leest de digitale waarde van een digitale INPUT pin. Geeft een booloan terug (HIGH,LOW of TRUE, FALSE)

digitalWrite(inputpin)

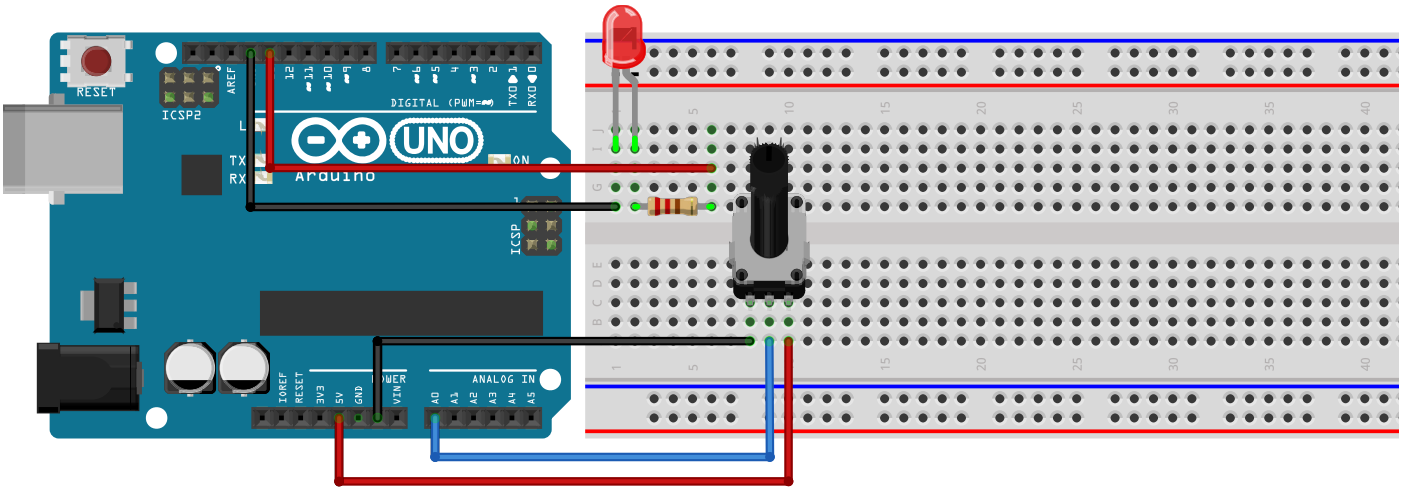
Wanneer je digitalWrite bij een inputpin gebruikt wordt er een pull-up weerstand geactiveerd waardoor deze pin standaard HIGH wordt tenzij er een verbinding met de ground wordt gemaakt.

if(condition){...}else{...}

Voert de code binnen de eerste haakjes uit wanneer wat tussen de ronde haakjes staat waar is, anders voert hij de code tussen na else uit.

==

Betekend simpelweg 'is gelijk aan', niet te verwarren met de enkele '=' want deze kent een variabele een waarde toe.



/analoog

Hoe laat je een digitaal apparaat met een analoge sensor communiceren? Hiervoor heeft men een ADC (analoog digitaal converter) bedacht. Arduino 6 AD converters. Arduino geeft je een gehele waarde terug tussen de 0 en de 1023 (10bit) dit komt lineair overeen met 0V en 5V. Dus 512 is ongeveer 2,5V.

Analoge waarde, aangesloten op een A# pin op de arduino

5,00v	*1023*
2,50v	*512*
0,00v	*0*

Digitale waarde (te gebruiken in de code)

/code

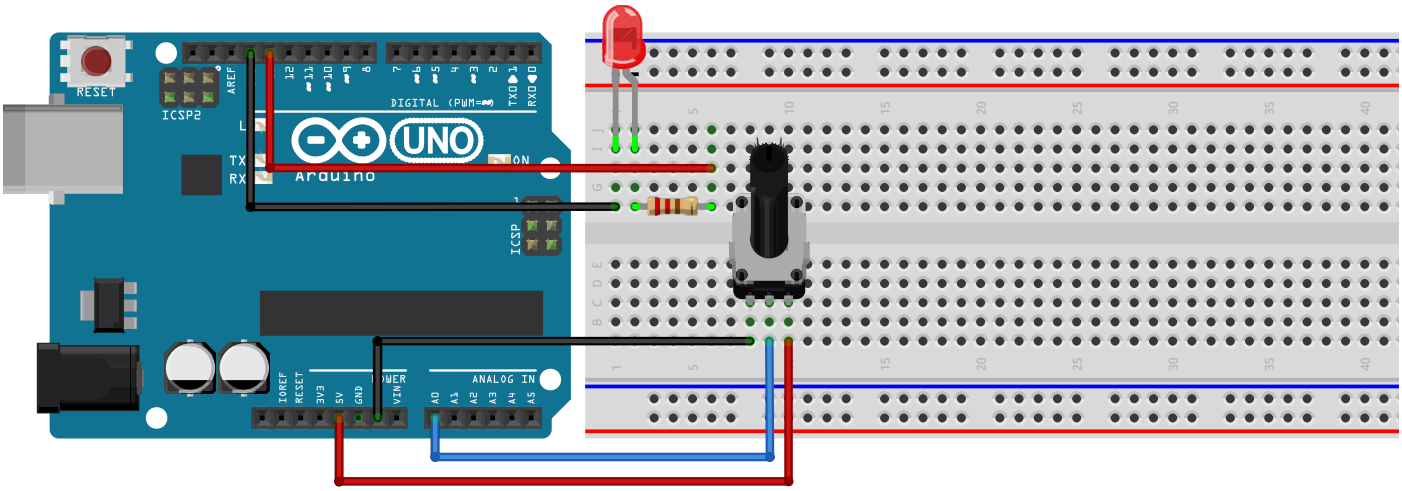
```
int ledPin = 13;
int potMeterPin = 0;
void setup() {
  pinMode(ledPin, OUTPUT);
}

void loop() {
  int a = analogRead(potMeterPin);
  digitalWrite(ledPin, HIGH);
  delay(a);
  digitalWrite(ledPin, LOW);
  delay(a);
}
```

/hardware

Nog steeds zou je het ingebouwde LEDje kunnen gebruiken, ook kun je een extern LEDje aansluiten. Je zult ook een potentiometer moeten aansluiten. Dit is een spanningsdeler met 3 pinnen. De buitenste sluit je aan op de 5V en GND. De middelste is de loper, de spanning tussen de GND en de loper veranderd wanneer je aan de potentiometer draait. Dit is ook de spanning die de Arduino meet wanneer de loper is aangesloten op de analoge pin 0. De Arduino zal dus waarden tussen de 0 en 1023 meten, dit houdt in dat het ledje op zijn langst 1,023 seconden aan of uit is.

/geluid

Ook kun je weer een piezo elementje aansluiten, wanneer je delay verangt door delayMicroseconds heb je je eigen syntesizer gemaakt! Probeer de frequentie van je geluid te veranderen door het draaien van de potentiometer.



/communicatie

Soms is het handig als je weet wat welke waarde een bepaalde variabele op een bepaald moment heeft. Nu kan je zien wat de variabele *a* is doordat het lampje sneller of langzamer gaat knipperen, maar nog handiger zou zijn wanneer je op je computer kunt zien wat de waarde van *a* is.
Daar is iets op bedacht; namelijk seriële communicatie. Je arduino verzend een boodschap en via de Arduino software (en vele andere software) kan je die boodschap zien en eventueel een boodschap terug sturen.

/code

```
int ledPin = 13;
void setup() {
    pinMode(ledPin, OUTPUT);
    Serial.begin(9600);
}

void loop() {
   //sla de analoge waarde van A0 op
  int a = analogRead(0);
  Serial.println(a);
  digitalWrite(ledPin, HIGH);
  delay(a);
  digitalWrite(ledPin, LOW);
  delay(a);
}
```

Serial.begin(baudrate);

Hiermee begin je communicatie met de seriële poort met een bepaalde baudrate (bits per seconde). Meestal is dit 9600bps (staat standaard ingesteld in de Arduino software)

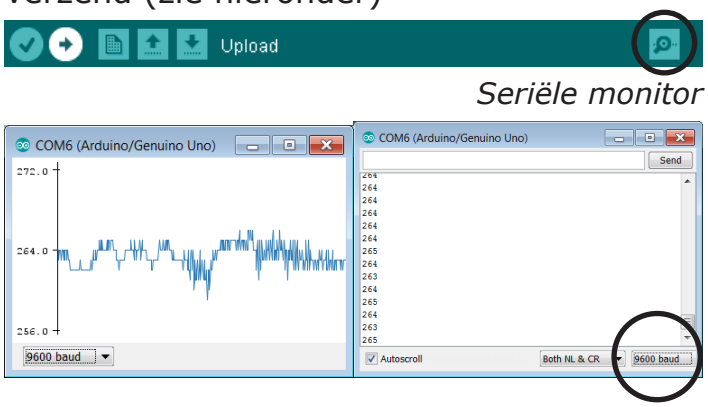
Serial.print(val);

Je verzend een bepaalde waarde via de seriële poort in een voor mensen leesbare taal (decimaal in ASCII).

Serial.println(val);

Je verzend een bepaalde waarde via de seriële poort met een regeleinde erachter (ook bekend als "\n").

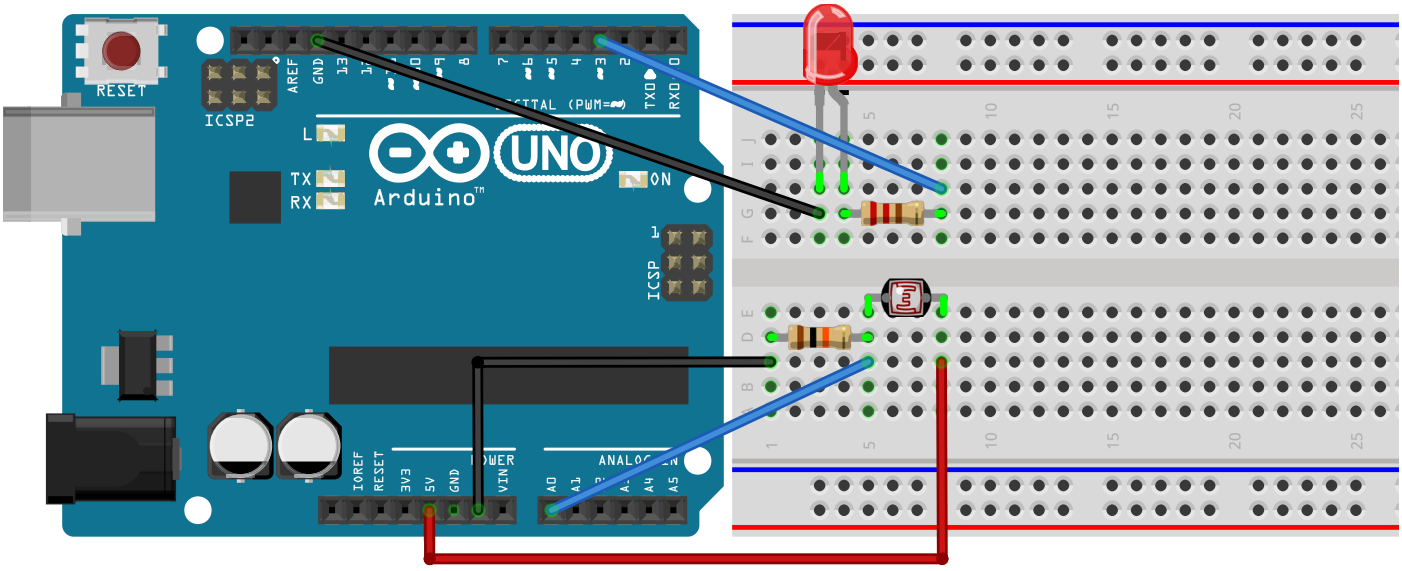
Via de seriële monitor kun je de berichten zien die de arduino naar de computer verzend (zie hieronder)



Seriële monitor

De seriële plotter is ook beschikbaar via 'tools>serial plotter'. Deze kan de waarden weergeven in een grafiek

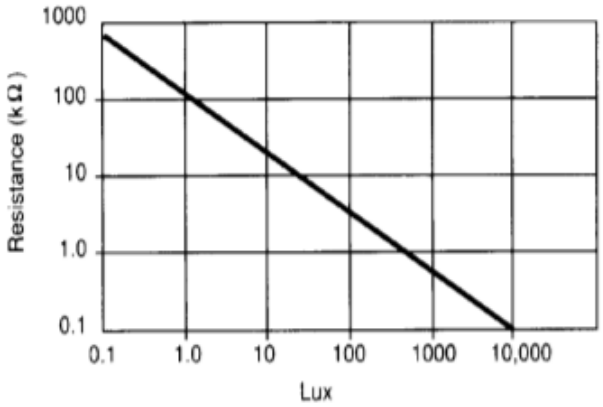
Selecteer de juiste baudrate, deze moet overeenkomen met die in de Serial. begin() functie



/LDR

De 'Light dependent resistor' is een lichtgevoelige weestand, waarbij de weerstand lager wordt naarmate er meer licht op de weestand valt.

FIGURE 2 RESISTANCE AS FUNCTION OF ILLUMINATION



Hier een voorbeeld van een lux, resistance diagram van een LDR (maanlicht: 0.1 lux, fel zonlicht 30.000 lux) - Sunrom technologies

/hardware

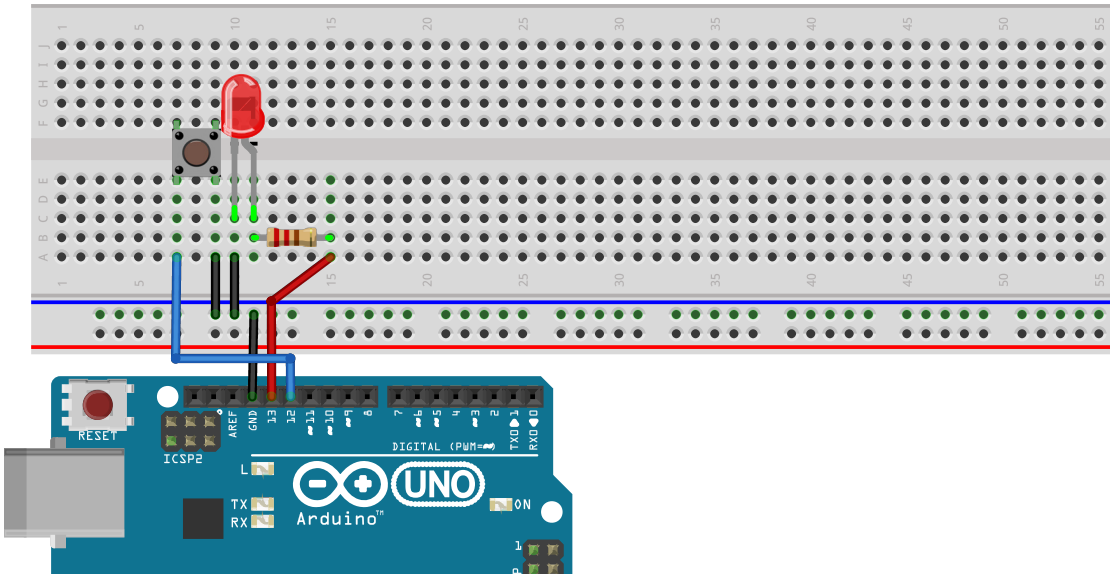
De arduino moet worden aangesloten op de LDR dit doe je door een spanningsdeler te maken, de spanning wordt verdeeld over een vaste weerstand, en een variable weerstand. Hiermee veranderd de verhouding tussen de weestanden, en dus ook de verhouding tussen de spanningen die daar overheen staan. Je kunt dezelfde code als voor het vorige project gebruiken aangezien Arduino nog steeds een spanning aan het meten is.

/andere analoge sensors

Er zijn vele andere sensors beschikbaar, zo is er een NTC voor temperatuur (of de DS18B20). Ook zijn er druksensoren, kleuren sensoren.

/slimme sensors

Naast analoge sensors zijn er ook veel sensors die met Arduino communiceren via een bepaalde protocol, of waarbij de interval belangrijk is. Voorbeelden zijn (ultrasone) afstandssensoren, GPS, acceleratie... de mogelijkheden zijn eindeloos.



/reactie

Nu wordt het tijd voor een wat groter project. We gaan de reactietijd van mensen meten. Dit doen we door een ledje op een willekeurig moment aan te laten gaan, en dan zo snel mogelijk op een drukknopje te drukken. Er zal dan dus de tijd tussen het aangaan van het lampje en het indrukken van de drukknop gemeten worden. Deze tijd wordt dan doorgestuurd naar de computer via een seriële verbinding. Mijn record is 0,189 sec.

/code

```
int ledPin = 13;
int buttonPin = 12;
boolean start = true;
unsigned long starttime = 0;

void setup() {
    pinMode(ledPin, OUTPUT);
    pinMode(buttonPin, INPUT);
    digitalWrite(buttonPin,HIGH);
    Serial.begin(9600);
}

void loop() {
  if (start){
    randomSeed(analogRead(0));
    delay(random(1000, 5000));
    start = false;
    digitalWrite(ledPin,HIGH);
    starttime = micros();
  }else{
    if (digitalRead(buttonPin)==LOW){
      Serial.println(micros()-starttime);
      start = true;
      digitalWrite(ledPin,LOW);
    }
  }
}
```

randomSeed(val);

Willekeur bestaat niet bij computers, daarom moet er iets externs zijn dat de willekeur bepaalt. In de bovenstaande code is dat een analoge onaangesloten pin, die dan ruis weergeeft. Dat is perfect voor willekeur.

random(min, max);

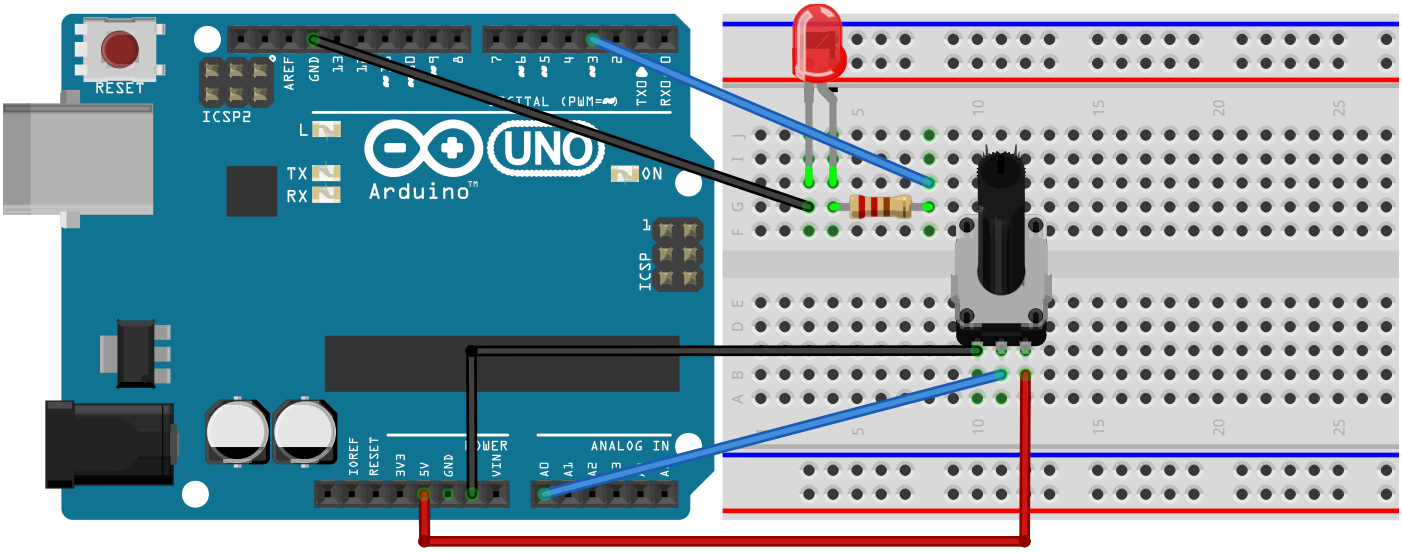
Genereert een willekeurig getal tussen de min en max.

mircos();

Geeft het aantal microseconden sinds de vorige reset weer.

/boolean

Een boolean is een variabele die slechts twee waarden kan aannemen of deze is waar (true) of deze is onwaar (false).



/PWM

PWM ookwel Pulse Width Modulation is een manier om met een digitaal signaal een analoog signaal te imiteren. Het principe is erop gebaseerd dat als je een ledje voor 50% van de tijd aanzet, het ledje dan ook 50% van de intensiteit laat zien. Het knipperen van een ledje kan je voorkomen door deze met meer dan 70Hz te laten knipperen, doordat men dit dan ziet als een continu licht. Ditzelfde principe wordt ook bij elektromotoren gebruikt, om de snelheid te bepalen. Er zijn 6 pinnen met PWM functionaliteit (op een Arduino UNO of soortgelijk board) (pin 3, 5, 6, 9, 10 en 11).

/code

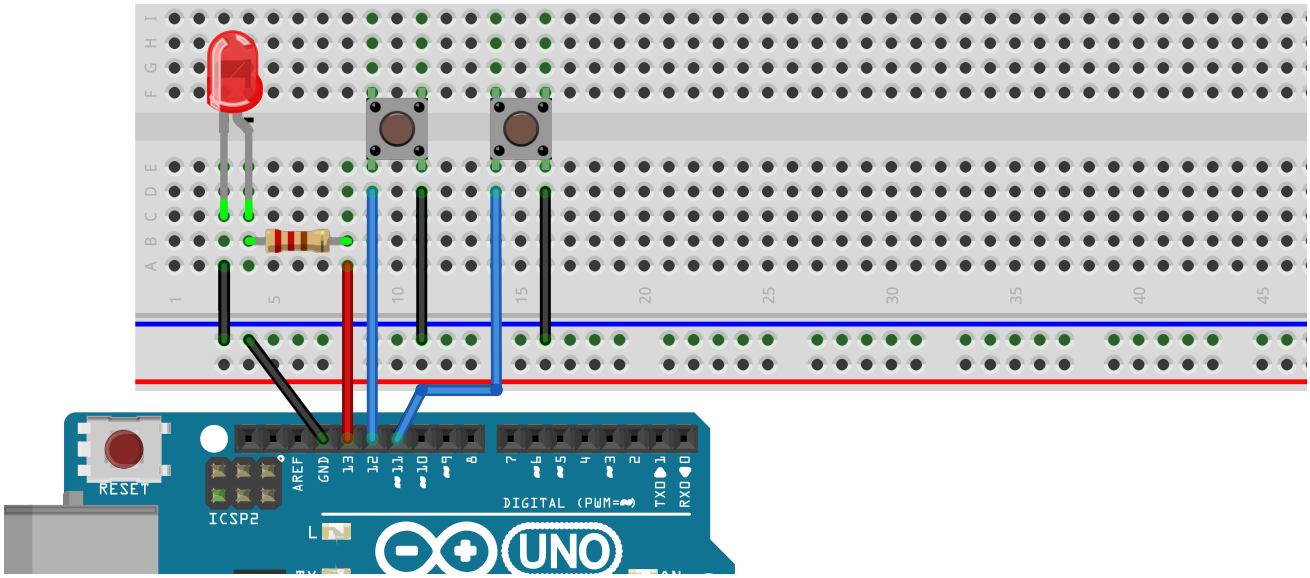
```
int pwmPin = 11;
int analogPin = 0;
void setup() {
}
void loop() {



analogWrite(pwmPin,
map(analogRead(analogPin), 0, 1023,
0, 255));
}
```

analogWrite(pin,value);

Maakt een bepaalde PWM pin knipperend met de verhouding value/255 aan.

map(value, minIn, maxIn, minOut, maxOut);

Maakt verhoudingsgewijs van een waarde van minIn tot maxIn een waarde van minOut tot maxOut.



/poorten

Vroeger bestond de digitale elektronica alleen maar uit poorten (AND, OR, xOR) en de inverse daarvan. Nog steeds in programmeertalen kun je deze functies gebruiken.

/code

```
void setup() {
  pinMode(11,INPUT);
  pinMode(10,INPUT);
  digitalWrite(11,HIGH);
  digitalWrite(10,HIGH);
}
void loop() {
  digitalWrite(13,
digitalRead(11)&&digitalRead(10));
}
```

var1&&var2

Wanneer var1 true is en var2 ook dan geeft deze true terug, anders false.

var1||var2

Wanneer var1 true is of var2 true is dan geeft dit true terug.

/HIGH/LOW

Wat je misschien is opgevallen bij de vorige code is dat HIGH hetzelfde doet als true, en LOW hetzelfde doet als false. Dit komt gewoonweg doordat het hetzelfde betekent binnen Arduino.

/lussen

Naast de basislus (de loop) zijn er nog andere manieren om een handeling vaker te herhalen. Dit is bijvoorbeeld handig als je van alle pinnen van 2 tot 13 een OUTPUT pin wilt maken. En er zijn nog veel meer toepassingen.

/code

```
void setup() {
  for(int i=2;i<=13;i++){
    pinMode(i,OUTPUT);
    digitalWrite(i,HIGH);
  }
}
void loop() {
  for(int i=2;i<=13;i++){
    for(int j=2;j<=13;j++){
      digitalWrite(j,HIGH);
    }
    digitalWrite(i,LOW);
    delay(100);
  }
}
```

for(int i=0;i<10;i++){...}

Deze code wordt gebruikt om dingen te herhalen en meteen ook te tellen hoe vaak iets herhaald wordt. Het eerste stuk wordt aan het begin uitgevoerd (int i = 0). Wanneer het 2e stuk true is voert hij de code tussen de haakjes uit (i<10). Na het uitvoeren van ieder stuk voert hij het laatste stuk uit (i++) ookwel i=i+1.

/Servo
Een servo is een motortje met een regelsysteem zodat de positie nauwkeurig kan worden ingesteld, dit type motor wordt veel gebruikt in hobbyprojecten (om te sturen) maar bijvoorbeeld ook om de ramen van kassen automatisch te bedienen. Een servo wordt aangestuurd door een puls, de lengte van deze puls bepaald de positie van de servo. Hierdoor hoeft de Arduino zich niet continu bezig te houden met het regelen van de motors.

/VirtualWire
VirtualWire is een library waarmee je met zeer goedkope 433Mhz radio zend en ontvangst modules berichten kan verzenden. Een zend en ontvangstmodule kosten slechts 5 euro.

/wifi
Er zijn ook vele mogelijkheden om Arduino te koppelen aan het internet, hiermee wordt het gemakkelijk om met het 'internet of things' concept te spelen. De ESP8266 (nodeMCU) is een krachtige, goedkop chip die met de Arduino omgeving geprogrammeerd kan worden. Ook heeft Arduino zelf een bord uitgebracht wat met het internet kan communiceren; de Arduino Yún. Deze is een stuk prijziger, maar biedt ook meer mogelijkheden.

/IR
Infrarood is een vorm van communicatie die in elk huishouden bijna elke dag voorkomt. Alle televisie - , radio - , beamer afstandsbedieningen werken via infrarood. Arduino kan heel eenvoudig met een IR ontvanger dit signaal opvangen en ook uitzenden met een IR ledje. Wat ook interessant is om te weten is dat IR niet met het oog kan worden gezien, maar met een goedkope camera (bijvoorbeeld die op je mobieltje) kun je IR licht zichtbaar maken.

/CapSense
Met de CapSense library kun je van ieder metaal object een aanraaksensor maken. Zelfs wanneer je tape over het metaal heen plakt werkt dit systeem nog steeds redelijk goed. Het enige dat je aan componenten nodig hebt is een weerstand en een metalen oppervlak. Het systeem stuurt een stroompje uit een output pin en meet hoelang het duurt voordat deze aankomt bij een input pin. Hierdoor kan deze meten hoe groot het oppervlak is dat wordt aangeraakt.

Wanneer je de basis van Arduino onder de knie hebt zal je zien dat je met Arduino gave dingen kunt maken. Doordat er talloze libraries online zijn voor uiteenlopende onderwerpen kan je zeer snel redelijk complexe dingen maken. Zie de onderstaande website (QR code) en Arduino.cc voor meer informatie, voorbeelden, projecten en libraries.

Dit werk is gelicenseerd onder een Creative Commons Naamsvermelding 3.0 Unported licentie. En gepubliceerd op http://lemio.nl/arduino/
Alle technische illustraties zijn met Fritzing gemaakt.

Arduino Programming Cheat Sheet

Primary source: Arduino Language Reference
http://arduino.cc/en/Reference/

Structure & Flow

Basic Program Structure

```
void setup() {
  // Runs once when sketch starts
}
void loop() {
  // Runs repeatedly
}
```

Control Structures

```
if (x < 5) { ... } else { ... }
while (x < 5) { ... }
for (int i = 0; i < 10; i++) { ... }
break;      // Exit a loop immediately
continue;   // Go to next iteration
switch (var) {
  case 1:
    ...
    break;
  case 2:
    ...
    break;
  default:
    ...
}
return x;   // x must match return type
return;     // For void return type
```

Function Definitions

```
<ret. type> <name>(<params>) { ... }
e.g. int double(int x) {return x*2;}
```

Operators

General Operators

=	assignment		
+	add	-	subtract
*	multiply	/	divide
%	modulo		
==	equal to	!=	not equal to
<	less than	>	greater than
<=	less than or equal to		
>=	greater than or equal to		
&&	and	||	or
!	not		

Compound Operators

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

Bitwise Operators

&	bitwise and	|	bitwise or
^	bitwise xor	~	bitwise not
<<	shift left	>>	shift right

Pointer Access

&	reference: get a pointer
*	dereference: follow a pointer

Variables, Arrays, and Data

Data Types

boolean	true | false
char	-128 - 127, 'a' '$' etc.
unsigned char	0 - 255
byte	0 - 255
int	-32768 - 32767
unsigned int	0 - 65535
word	0 - 65535
long	-2147483648 - 2147483647
unsigned long	0 - 4294967295
float	-3.4028e+38 - 3.4028e+38
double	currently same as float
void	i.e., no return value

Strings

```
char str1[8] =
  {'A','r','d','u','i','n','o','\0'};
  // Includes \0 null termination
char str2[8] =
  {'A','r','d','u','i','n','o'};
  // Compiler adds null termination
char str3[] = "Arduino";
char str4[8] = "Arduino";
```

Numeric Constants

123	decimal
0b01111011	binary
0173	octal - base 8
0x7B	hexadecimal - base 16
123U	force unsigned
123L	force long
123UL	force unsigned long
123.0	force floating point
1.23e6	1.23*10^6 = 1230000

Qualifiers

static	persists between calls
volatile	in RAM (nice for ISR)
const	read-only
PROGMEM	in flash

Arrays

```
int myPins[] = {2, 4, 8, 3, 6};
int myInts[6];    // Array of 6 ints
myInts[0] = 42;   // Assigning first
                  // index of myInts
myInts[6] = 12;   // ERROR! Indexes
                  // are 0 though 5
```

Built-in Functions

Pin Input/Output

```
Digital I/O - pins 0-13 A0-A5
  pinMode(pin,
    [INPUT, OUTPUT, INPUT_PULLUP])
  int digitalread(pin)
  digitalWrite(pin, [HIGH, LOW])

Analog In - pins A0-A5
  int analogRead(pin)
  analogReference(
    [DEFAULT, INTERNAL, EXTERNAL])

PWM Out - pins 3 5 6 9 10 11
  analogWrite(pin, value)
```

Advanced I/O

```
tone(pin, freq_Hz)
tone(pin, freq_Hz, duration_ms)
noTone(pin)
shiftOut(dataPin, clockPin,
  [MSBFIRST, LSBFIRST], value)
unsigned long pulseIn(pin,
  [HIGH, LOW])
```

Time

```
unsigned long millis()
  // Overflows at 50 days
unsigned long micros()
  // Overflows at 70 minutes
delay(msec)
delayMicroseconds(usec)
```

Math

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

Random Numbers

```
randomSeed(seed) // long or int
long random(max) // 0 to max-1
long random(min, max)
```

Bits and Bytes

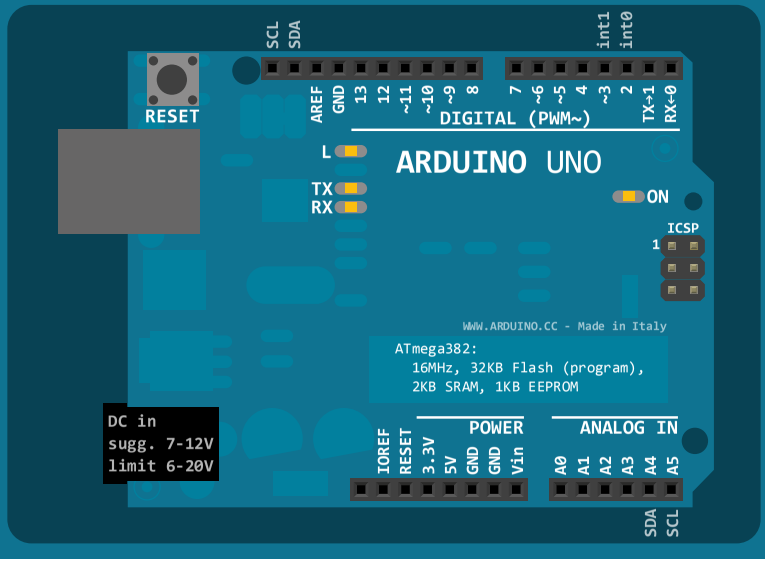
```
lowByte(x)    highByte(x)
bitRead(x, bitn)
bitWrite(x, bitn, bit)
bitSet(x, bitn)
bitClear(x, bitn)
bit(bitn)  // bitn: 0=LSB 7=MSB
```

Type Conversions

```
char(val)     byte(val)
int(val)      word(val)
long(val)     float(val)
```

External Interrupts

```
attachInterrupt(interrupt, func,
 [LOW, CHANGE, RISING, FALLING])
detachInterrupt(interrupt)
interrupts()
noInterrupts()
```



Libraries

Serial - comm. with PC or via RX/TX

```
begin(long speed)  // Up to 115200
end()
int available()  // #bytes available
int read()   // -1 if none available
int peek()   // Read w/o removing
flush()
print(data)    println(data)
write(byte)    write(char * string)
write(byte * data, size)
SerialEvent()  // Called if data rdy
```

SoftwareSerial.h - comm. on any pin

```
SoftwareSerial(rxPin, txPin)
begin(long speed)  // Up to 115200
listen()       // Only 1 can listen
isListening()  // at a time.
read, peek, print, println, write
  // Equivalent to Serial library
```

EEPROM.h - access non-volatile memory

```
byte read(addr)
write(addr, byte)
EEPROM[index]  // Access as array
```

Servo.h - control servo motors

```
attach(pin, [min_uS, max_uS])
write(angle)  // 0 to 180
writeMicroseconds(uS)
   // 1000-2000; 1500 is midpoint
int read()    // 0 to 180
bool attached()
detach()
```

Wire.h - I²C communication

```
begin()     // Join a master
begin(addr) // Join a slave @ addr
requestFrom(address, count)
beginTransmission(addr) // Step 1
send(byte)              // Step 2
send(char * string)
send(byte * data, size)
endTransmission()       // Step 3
int available() // #bytes available
byte receive()  // Get next byte
onReceive(handler)
onRequest(handler)
```


CC BY SA

by Mark Liffiton

Adapted from:
- Original: Gavin Smith
- SVG version: Frederic Dufourg
- Arduino board drawing: Fritzing.org