Microcontrollers

Les 4

Interrupts

Polling:

- continu hardware controleren op veranderingen
- veel tijdsverlies voor CPU
- trage reactie bij veel taken

Interrupts:

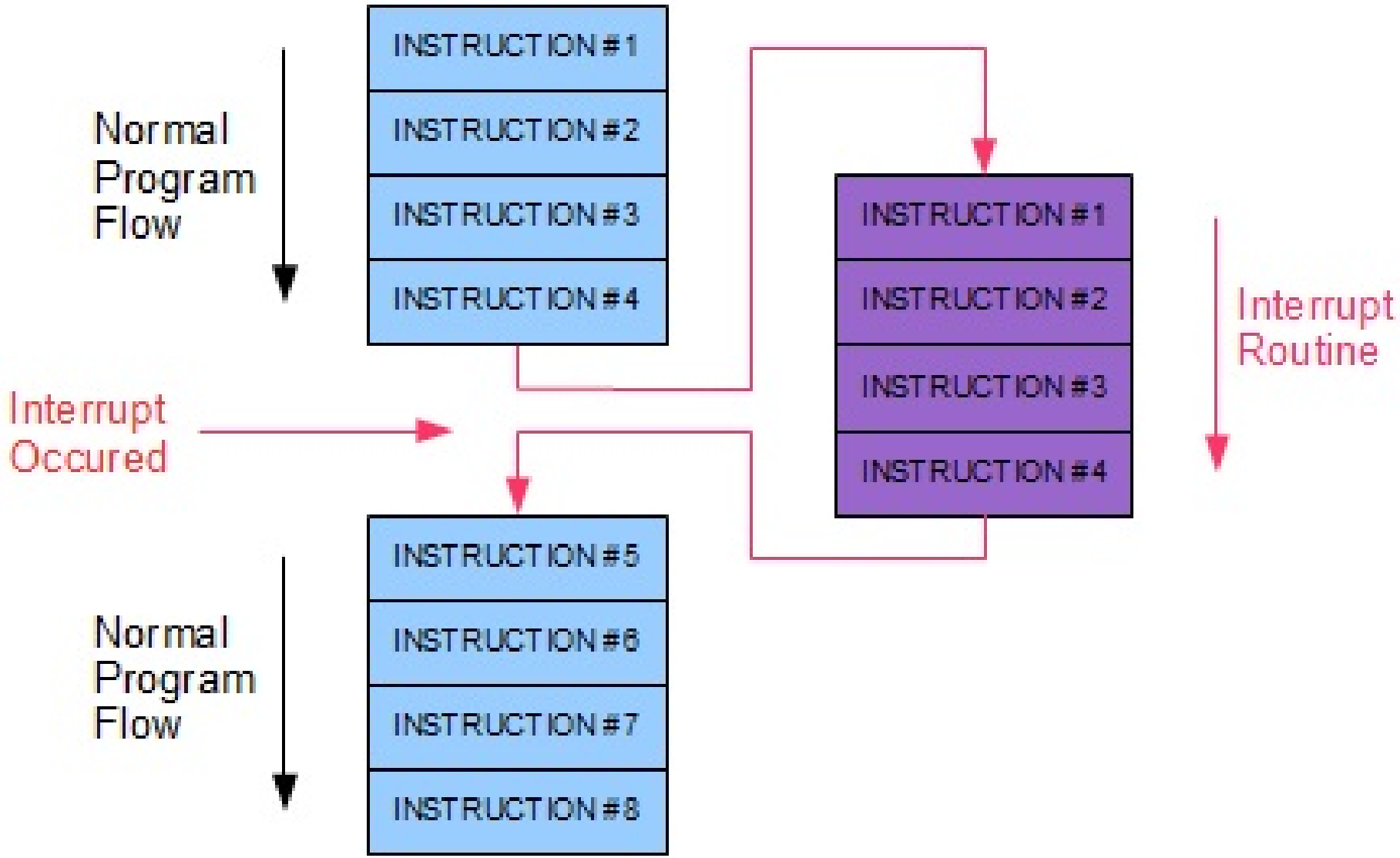
- hardware genereert signaal
- CPU is vrij om andere taken te doen
- hardware complexer

Afhandeling interrupt

CPU onderneemt volgende stappen bij interrupt:

- interrupt-vlag wordt nul gemaakt, om invloed andere interrupts te beperken
- huidige programmapositie wordt tijdelijk bewaard
- programma springt naar code voor interrupt
- op einde van interrupt terug naar gewone code

Afhandeling interrupt



Interrupts

- Arduino ondersteunt ook interrupts
- enkele voorbeelden:
  - externe interrupts
  - pin change interrupts
  - timer interrupts

Externe interrupts

- worden getriggerd door digitale ingang
- hardware monitort continu deze pinnen
- zijn zeer snel
- helaas beperkt tot specifieke pinnen
- op Uno: INT0 (pin 2) en INT1 (pin 3)

Gebruik externe interrupts

- interrupts aanzetten:
  `attachInterrupt(interrupt, isr, mode);`
- `interrupt`: nummer van de interrupt (0 of 1)
- `isr`: interrupt routine die aangeroepen wordt
- `mode`: RISING, FALLING, CHANGE of LOW

Snelheid

- hou de interrupt routine zo kort mogelijk
- tijdens de uitvoer ervan wordt de rest gestaakt
- ook andere interrupts worden genegeerd
- gebruik geen seriële functies in `isr`
- deze maken ook gebruik van interrupts

Interrupt uitschakelen

- tijdens uitvoer code kan interrupt terug uitgeschakeld worden
- bv. om inputs (tijdelijk) te negeren
- uitschakelen met: `detachInterrupt(interrupt)`

Voorbeeld

```
1  #define LED 4   // LED op pin 4
2
3  void knopISR() {
4    digitalWrite(LED,HIGH);   // LED aan
5  }
6
7  void setup() {
8    pinMode(LED, OUTPUT);       // LED uitgang
9    pinMode(2,INPUT_PULLUP);   // pin 2 input
10   attachInterrupt(0, knopISR, FALLING);
11   digitalWrite(LED,LOW);   // LED uit
12 }
13
14 void loop() {}
```

Oefening

- pas de code terug aan zodat de LED kan uitgezet worden
- gebruik hiervoor de interrupt op poort 3 (INT1)

Pin Change interrupts

- laat toe om op elke pin interrupts op te vangen
- is softwaregebaseerd: dus trager
- reageert op $change$: dus zowel $rise$ als $fall$
- meerdere pinnen op één interrupt (per poort):
  - PCINT0: pin D8 tot D13
  - PCINT1: pin A0 tot A5
  - PCINT2: pin D0 tot D7

Gebruik

- gebruik functie `pciSetup()` of bibliotheek
- in `setup()` wordt `pciSetup()` opgeroepen voor gewenste pin
- ISR voorzien per poort
- in ISR verder controleren op pin-niveau

Clock shield

- K1, K2, K3 verbonden met pin 9, 10 en 11
- dit komt overeen met PCINT0
- gebruiken allemaal dezelfde ISR

pciSetup()

```
1 void pciSetup(byte pin)
2 {
3   *digitalPinToPCMSK(pin) |= bit (↘
    →digitalPinToPCMSKbit(pin)); //enable pin
4   PCIFR  |= bit (digitalPinToPCICRbit(pin));↘
    → // clear any outstanding interrupt
5   PCICR  |= bit (digitalPinToPCICRbit(pin));↘
    → // enable interrupt for the group
6 }
```

setup()

```
1  #define K1   9    // K1 op pin 9
2  #define K2   10   // K2 op pin 10
3  #define LED3 4    // LED3 op pin 4
4
5  void setup() {
6    pinMode(LED3, OUTPUT);
7    pinMode(K1, INPUT_PULLUP);
8    pinMode(K2, INPUT_PULLUP);
9    pciSetup(K1);  // interrupt op K1
10   pciSetup(K2);  // interrupt op K2
11 }
12
13 void loop() {}
```

ISR()

```
1 ISR (PCINT0_vect) // afhandeling interrupt ↘
   →pin D8 tot D13
2 {
3   if (digitalRead(K1) == LOW) digitalWrite(↘
     →LED3, HIGH);
4   if (digitalRead(K2) == LOW) digitalWrite(↘
     →LED3, LOW);
5 }
```

Volatile

- let op met variabelen die in een interrupt routine gebruikt worden
- standaard worden variabelen bewaard in een tijdelijk register
- in bepaalde omstandigheden kunnen de variabelen onbedoeld gewijzigd worden
- bij de Arduino UNO kan dit gebeuren bij het aanroepen van een interrupt
- oplossing: declareer de variabele als $volatile$
- op die manier wordt die bewaard in het RAM geheugen en niet in een register

Volatile

```
1  #define LED 4
2  volatile bool toestand = LOW;
3
4  void setup() {
5    pinMode(LED, OUTPUT);
6    attachInterrupt(0, knopISR, FALLING);
7  }
8  void loop() {
9    digitalWrite(LED, toestand);
10 }
11 void knopISR() {
12   toestand = !toestand;
13 }
```

Delay in interrupt

- functie `delay()` vermijden in interrupt
- uitwerking meestal met veel kortere tijd
- eventueel zelf vertraging genereren met nutteloze code:

```
1 void wacht() {
2   // 200ms = +/- 450000
3   for (unsigned long i=0; i<450000; i++) {
4     __asm__("nop\n\t");
5   }
6 }
```

Output signaal versterken

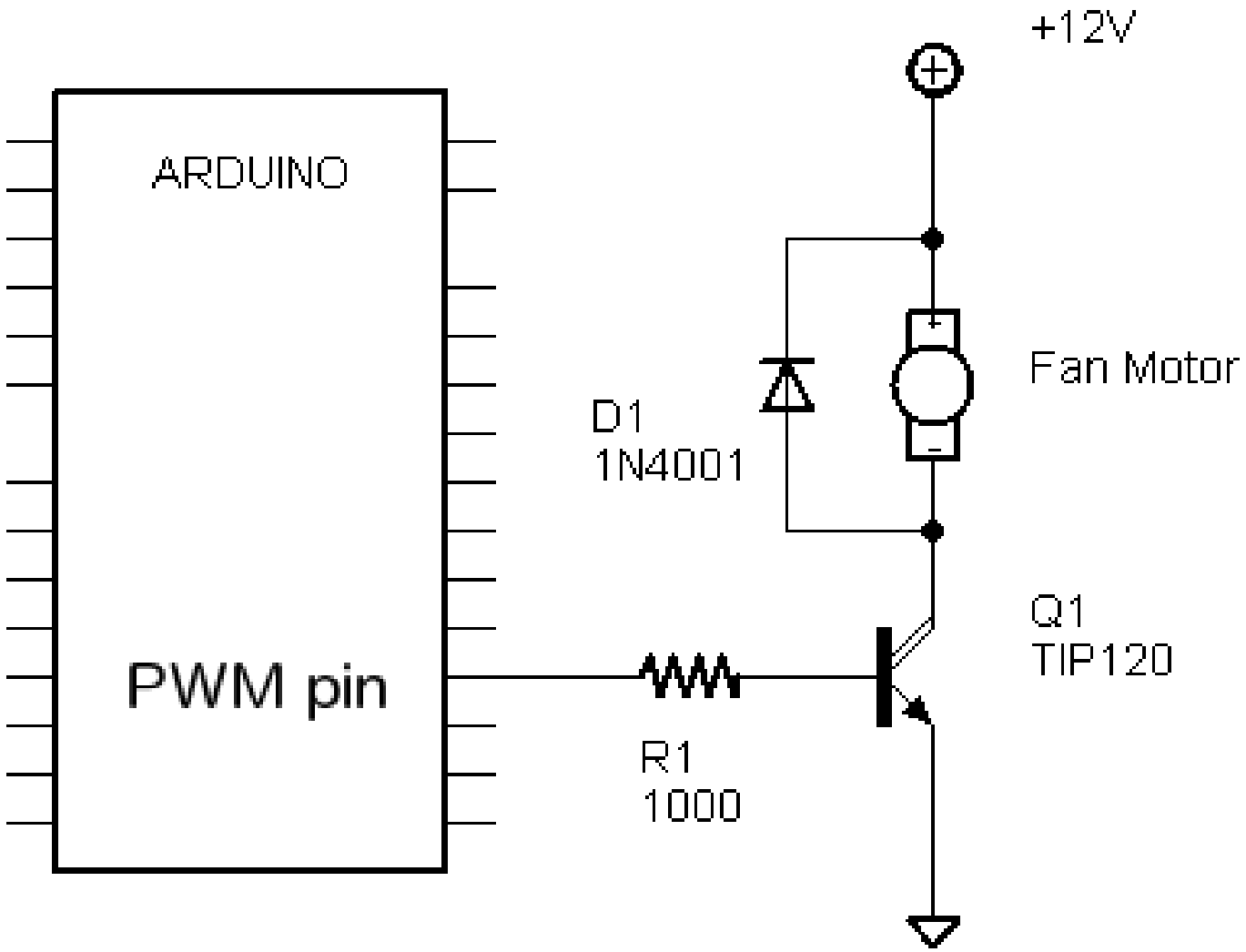
Verschillende mogelijkheden:

- transistor
- relais
- H-brug
- optocoupler

Transistor

- transistor als schakelaar
- kan grote stromen aan (naargelang type)
- snel, laat PWM toe
- laat geen ompoling toe

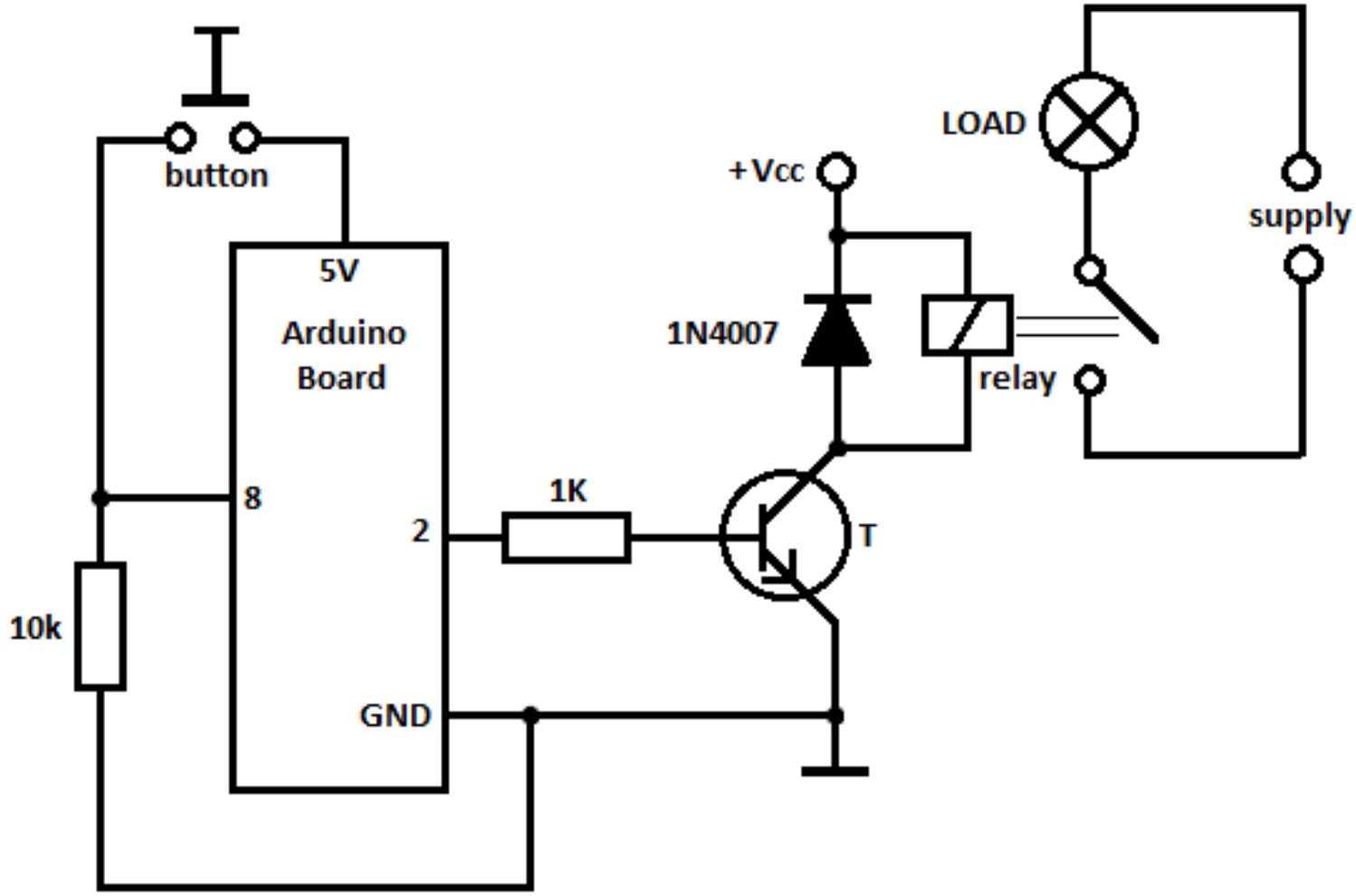
Voorbeeld



Relais

- galvanisch gescheiden
- kan (zeer) grote stromen aan
- traag, geen PWM mogelijk
- laat geen ompoling toe
- meestal transistor nodig om spoel aan te sturen
- wordt meestal gebruikt om netspanning te schakelen

Voorbeeld

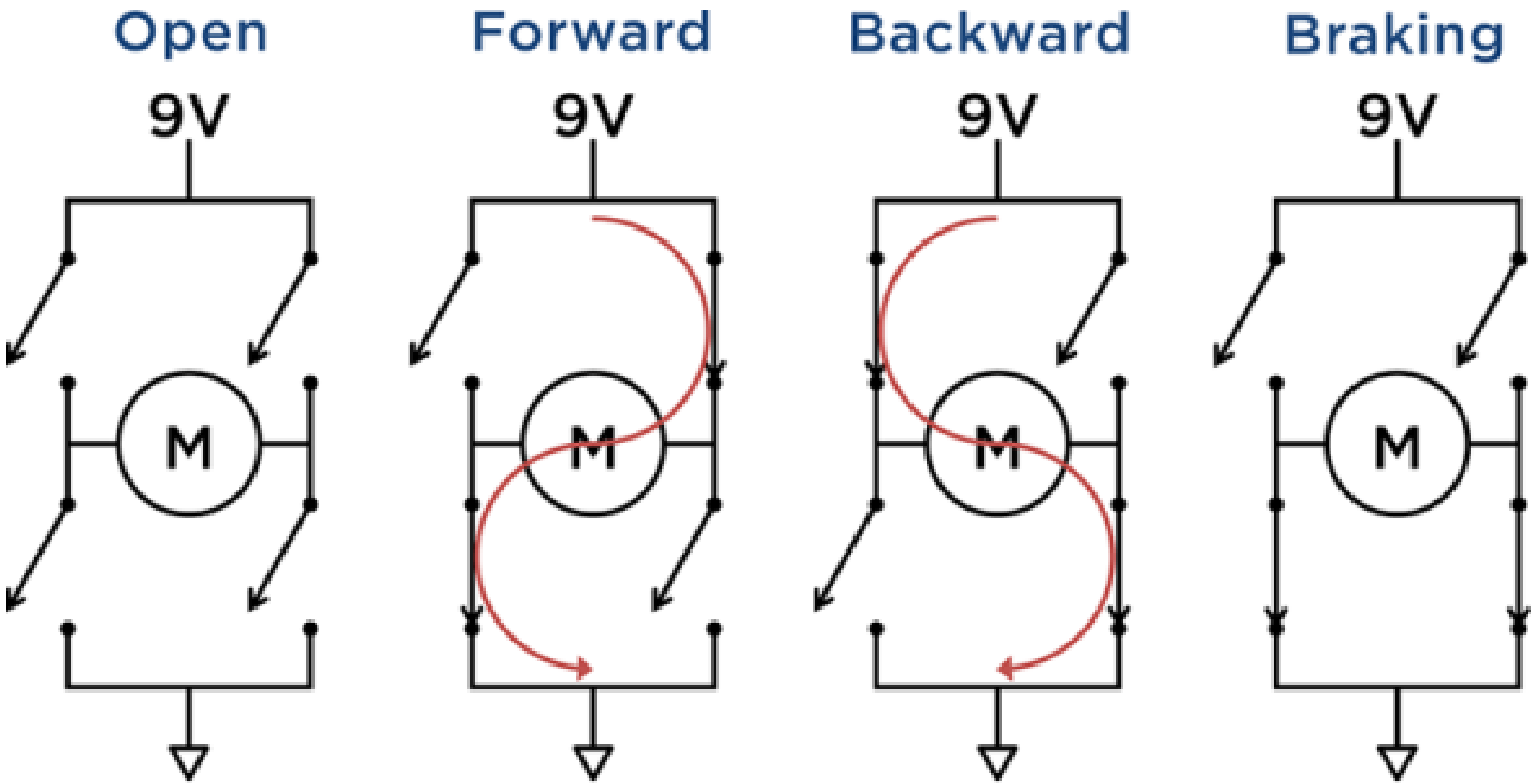

button
5V
Arduino
Board
8
2
GND
10k
1K
T
1N4007
+Vcc
LOAD
supply
relay

Relay shield

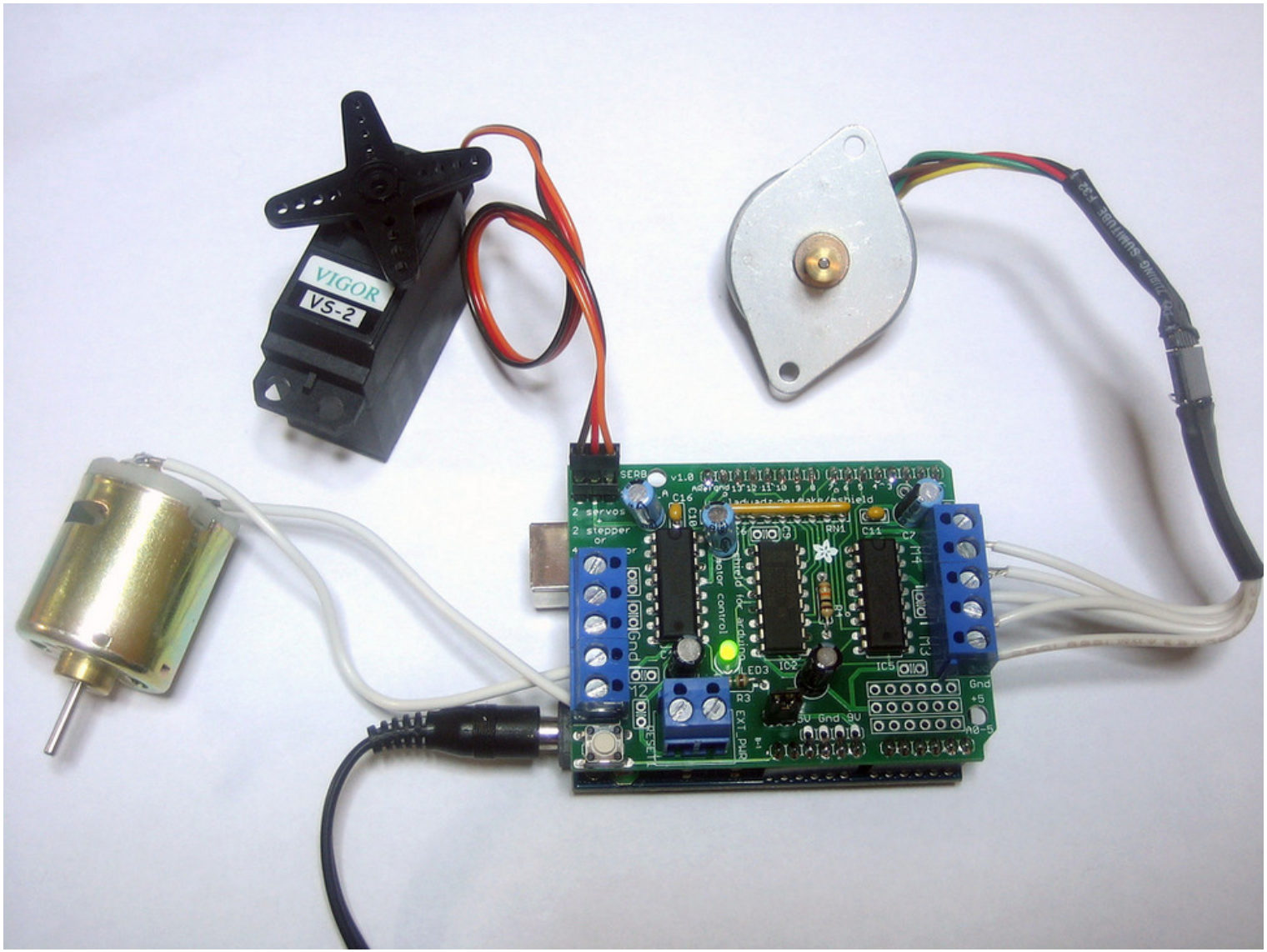
H-brug

- kan grote stromen aan
- meestal uit transistoren of MOSFET's opgebouwd
- snel, laat PWM toe
- laat ompoling toe
- wordt meestal gebruikt voor (DC) motoren
- speciale IC's beschikbaar (bv. L293D)

Voorbeeld



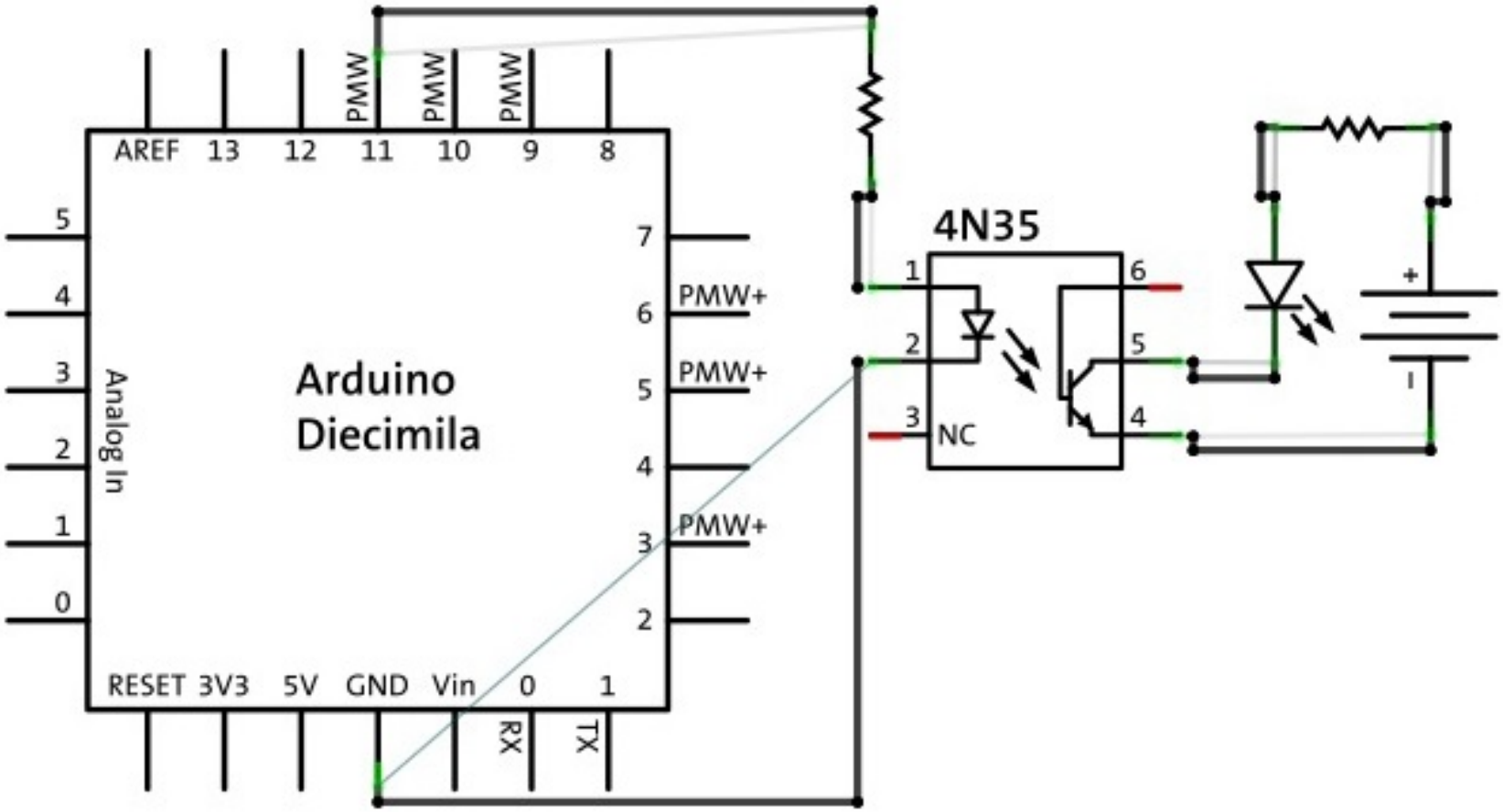
Motor shield



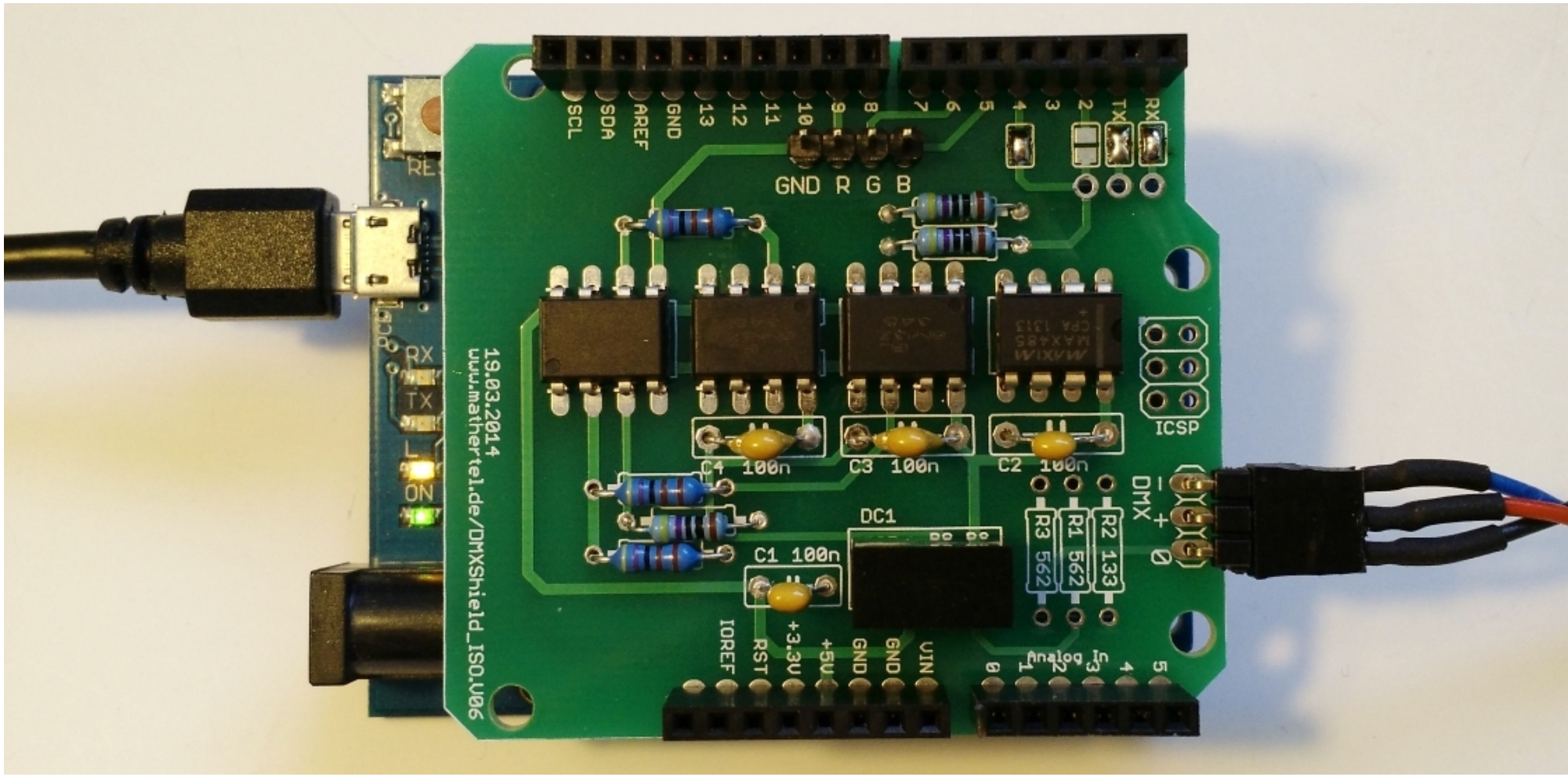
Optocoupler

- voor relatief kleine stromen
- galvanisch gescheiden
- snel, laat PWM toe
- laat geen ompoling toe
- wordt gebruikt om DC-netwerken galvanisch van elkaar te scheiden
- speciale IC's beschikbaar

Voorbeeld


AREF
13
12
PMW
11
PMW
10
PMW
9
8
5
4
3
2
1
0
Analog In
Arduino
Diecimila
7
PMW+
6
PMW+
5
4
PMW+
3
2
RESET
3V3
5V
GND
Vin
0
RX
1
TX
4N35
1
2
3
NC
6
5
4
+
I

DMX-shield


SCL
SDA
AREF
GND
13
12
11
10
9
8
7
6
5
4
3
2
TX
RX
GND R G B
RESET
RX
TX
L
ON
19.03.2014
www.matherlel.de/DMXShield_ISO.V06
C4 100n
C3 100n
C2 100n
ICSP
DC1
C1 100n
R3 562
R1 562
R2 133
DMX
IOREF
RST
+3.3V
+5V
GND
GND
VIN
Analog In
0
1
2
3
4
5

Bibliotheken

- Arduino is heel populair door enorme codeaanbod op internet
- meeste aanbod zit in bibliotheken
- bibliotheek bevat een aantal functies rond een bepaald item
- dit kan gaan over een sensor, timer, wiskunde, ...

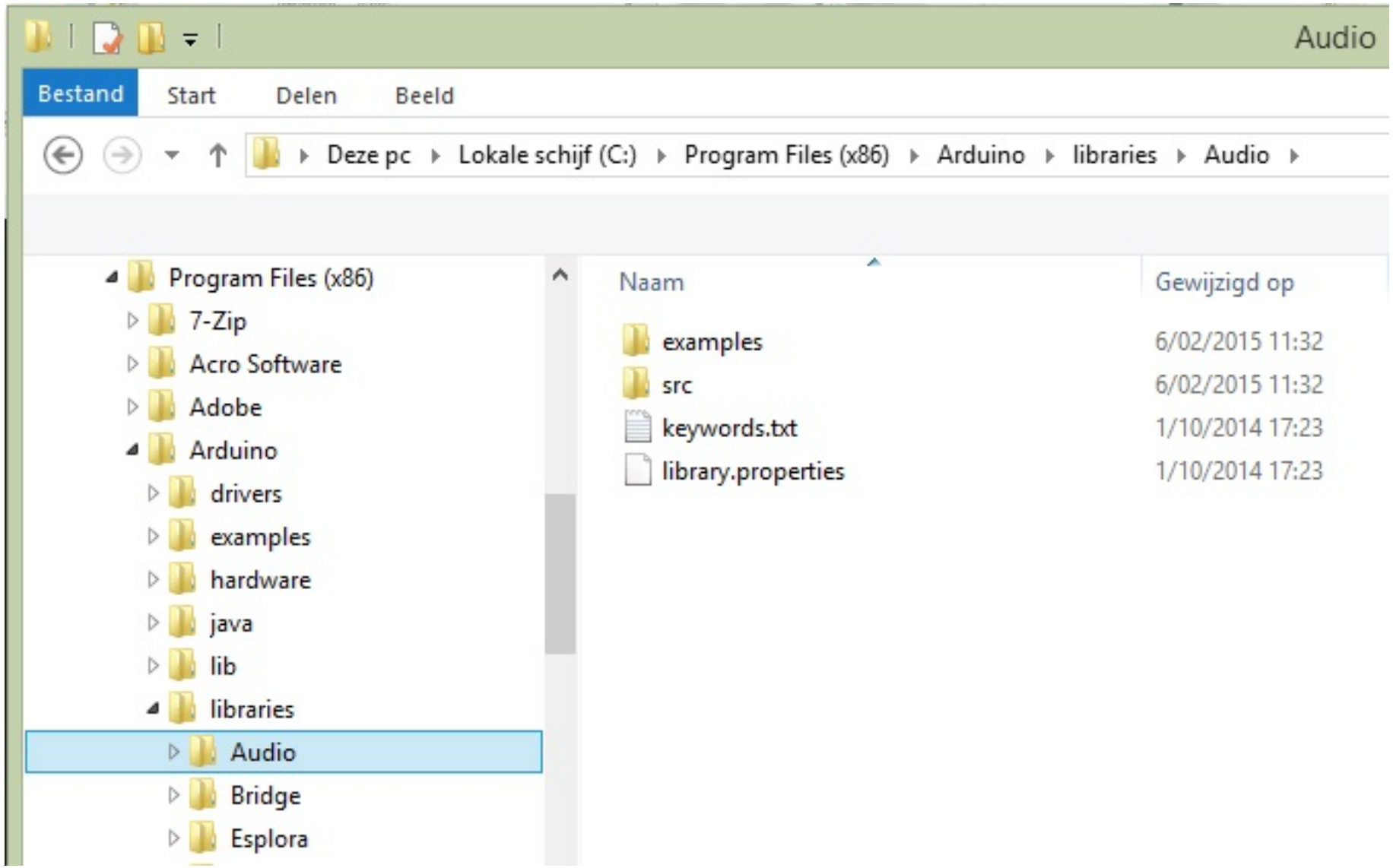
Opbouw

- bibliotheek bevat 2 bestanden: header en codebestand
- header bevat de functie-aanroepen, geen code
- code bevat de eigenlijke code
- de code moet in C++ formaat zijn

Locatie

- de bibliotheken worden bewaard in de `Program Files (x86)` map
- in Arduino map is er een map `libraries` aanwezig
- elke bibliotheek wordt bewaard in een eigen map waarbij de naam overeenkomt met de naam van de bibliotheek
- iedere map bevat ook een map `examples` met voorbeelden die de bibliotheek gebruikt
- welke bibliotheken er ter beschikking staan kan je ook zien in het menu `Schets - Importeer bibliotheek`

Locatie


Audio
Bestand
Start
Delen
Beeld
Deze pc
Lokale schijf (C:)
Program Files (x86)
Arduino
libraries
Audio
Program Files (x86)
7-Zip
Acro Software
Adobe
Arduino
drivers
examples
hardware
java
lib
libraries
Audio
Bridge
Esplora
Naam
Gewijzigd op
examples
6/02/2015 11:32
src
6/02/2015 11:32
keywords.txt
1/10/2014 17:23
library.properties
1/10/2014 17:23

Persoonlijke bibliotheken

- elke gebruiker kan ook persoonlijke bibliotheken installeren
- ideaal voor gebruiker met beperkte rechten
- andere gebruikers kunnen deze niet gebruiken
- deze staan in `Documents\Arduino\libraries`

Standaard bibliotheken

- EEPROM: om intern geheugen te gebruiken (1024 bytes op Uno)
- Ethernet en Wifi: netwerktoegang
- GSM: gebruik van het GSM-netwerk
- LiquidCrystal en TFT: displays
- SD: lezen en schrijven op SD-kaart
- Servo en Stepper: motoren aansturen
- SPI en Wire: SPI en I^2C protocollen gebruiken

Gebruik

- via het menu: `Schets - Importeer bibliotheek`
- of via `#include <libraryheader>`
- soms meerdere nodig
- of door een bestaand voorbeeld te gebruiken
- daarna kunnen alle functies uit de bibliotheek gebruikt worden

Referentie

- bij gebruik functie steeds naar bibliotheek refereren:

```
1 #include <SPI.h>
2 #include <Audio.h>
3
4 void setup()
5 {
6   SPI.setClockDivider(4);
7   Audio.begin(88200, 100);
8 }
```

Extra bibliotheken

- deze zijn ontwikkeld door de Arduino gebuikers
- worden niet standaard meegeleverd
- groot deel te vinden via: http://playground.arduino.cc
- indien keuze gevonden: downloaden en installeren
- kan via: `Schets - Importeer bibliotheek - Bibliotheek toevoegen`
- bibliotheek wordt dan in persoonlijke map geïnstalleerd

Demo IR-ontvanger

- we willen een eenvoudige afstandsbediening kunnen gebruiken
- zoek de hardware samen: IR-ontvanger + ontvangst
- zoek de juiste bibliotheek: bv. IRremote
- in code pinnummers aanpassen

Eigen bibliotheken

- interessant om eigen functies in meerdere projecten te gebruiken
- kan ook gebruikt worden om je functies te delen met de gemeenschap
- code moet in C++ formaat zijn
- C++ is objectgeörienteerd

Bestand MijnBib.cpp

```
1 #include "Arduino.h"
2 #include "MijnBib.h"
3
4 int MijnBibClass::som(int a, int b) {
5   int result = a + b;
6   return result;
7 }
8
9 MijnBibClass MijnBib;  // declaratie ↘
  →instantie klasse
```

Bestand MijnBib.h

```
1  #ifndef MijnBib_h
2  #define MijnBib_h
3  #include "Arduino.h"
4
5  class MijnBibClass
6  {
7    public:
8      int som(int, int);
9  };
10 extern MijnBibClass MijnBib;
11 #endif
```

Zip bestand maken

- bibliotheken worden als zip-bestand geïmporteerd
- plaats de bestanden in een map: `MijnBib`
- comprimeer die map naar: `MijnBib.zip`
- let op de gelijke namen
- daarna kan je eigen bibliotheek toegevoegd worden

Gebruik

```
1 #include <MijnBib.h>
2
3 void setup() {
4   Serial.begin(9600);
5   Serial.print(MijnBib.som(4, 5));
6 }
```

Voorbeelden

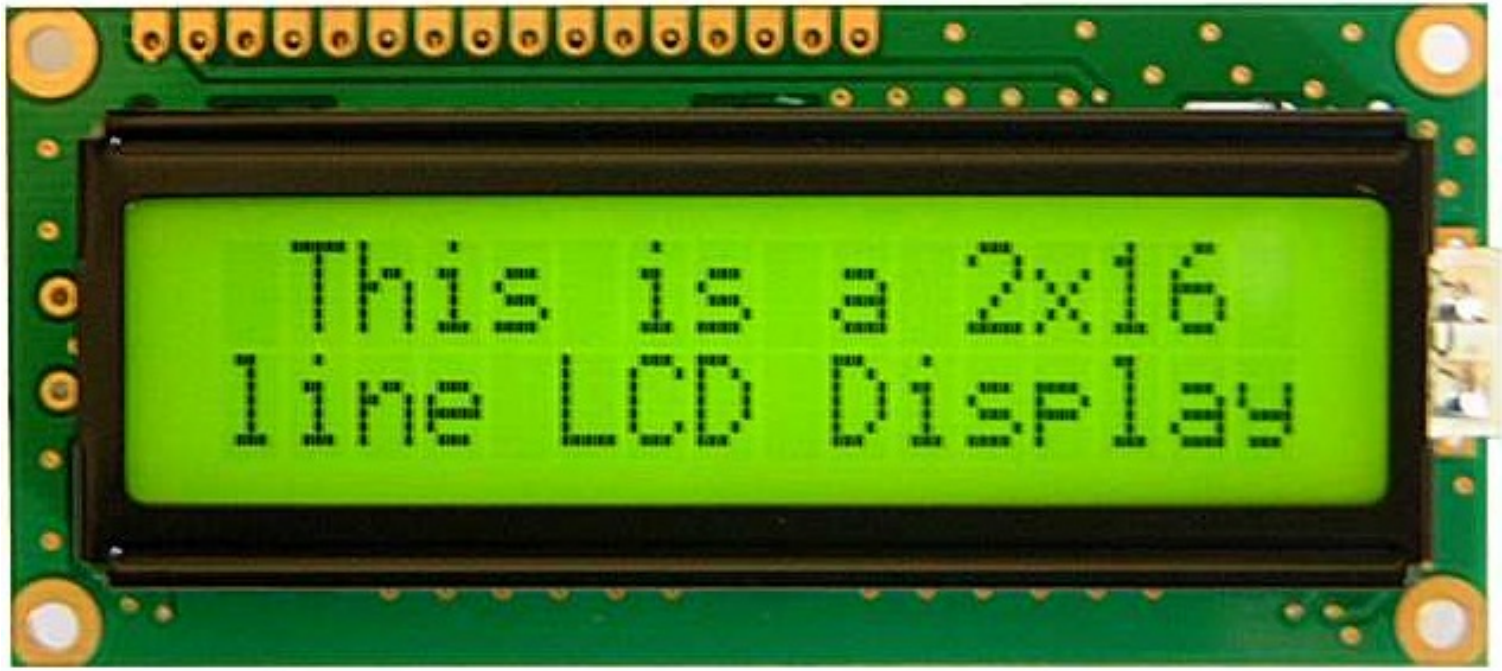
- indien je een bibliotheek publiceert best voorbeelden toevoegen
- maak een aantal voorbeelden in de IDE
- maak een map `examples` aan in de map `MijnBib` onder *Documents*
- bewaar daar je voorbeeld sketches
- na het opnieuw starten van de Arduino IDE zouden de voorbeelden zichtbaar moeten zijn
- maak de zip-opnieuw aan

Displays

- LED: om een status te tonen
- LCD: om meerdere karakters weer te geven
- 7-segment: om een beperkt aantal cijfers te tonen
- TFT: meestal kleur en grafisch

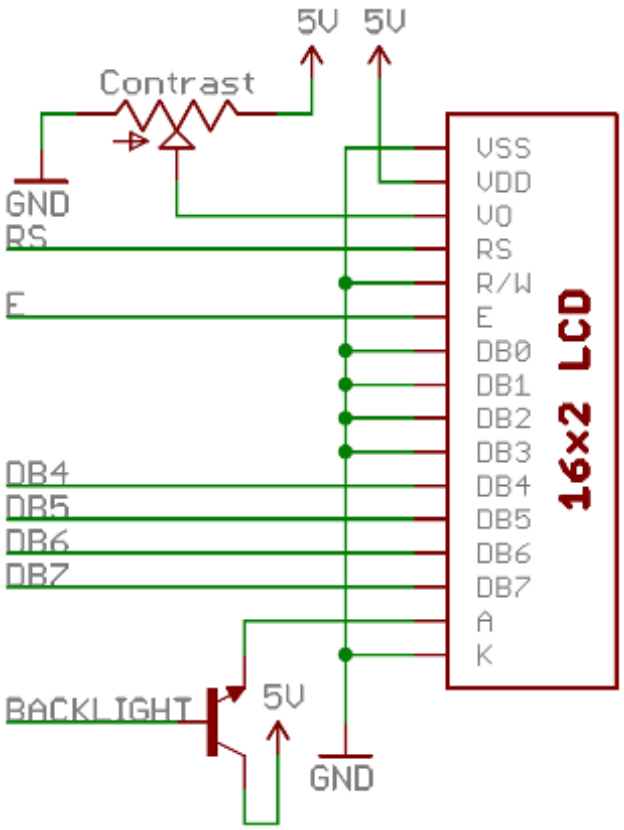
LCD

- Liquid Crystal Display
- standaard 2 lijnen van 16 karakters
- verbruikt zeer weinig stroom (zonder verlichting)



LCD aansluiten

- LCD heeft 14 tot 16 pinnen
- keuze: 4 of 8 datapinnen



Protocol

- LCD dient geïnitialiseerd te worden
- eerst wachten op voedingsspanning (15ms)
- vervolgens initialisatiebytes versturen
- telkens wachten tussen de verschillende stappen
- bij karakters versturen: data + puls op enable pin

Arduino bibliotheek

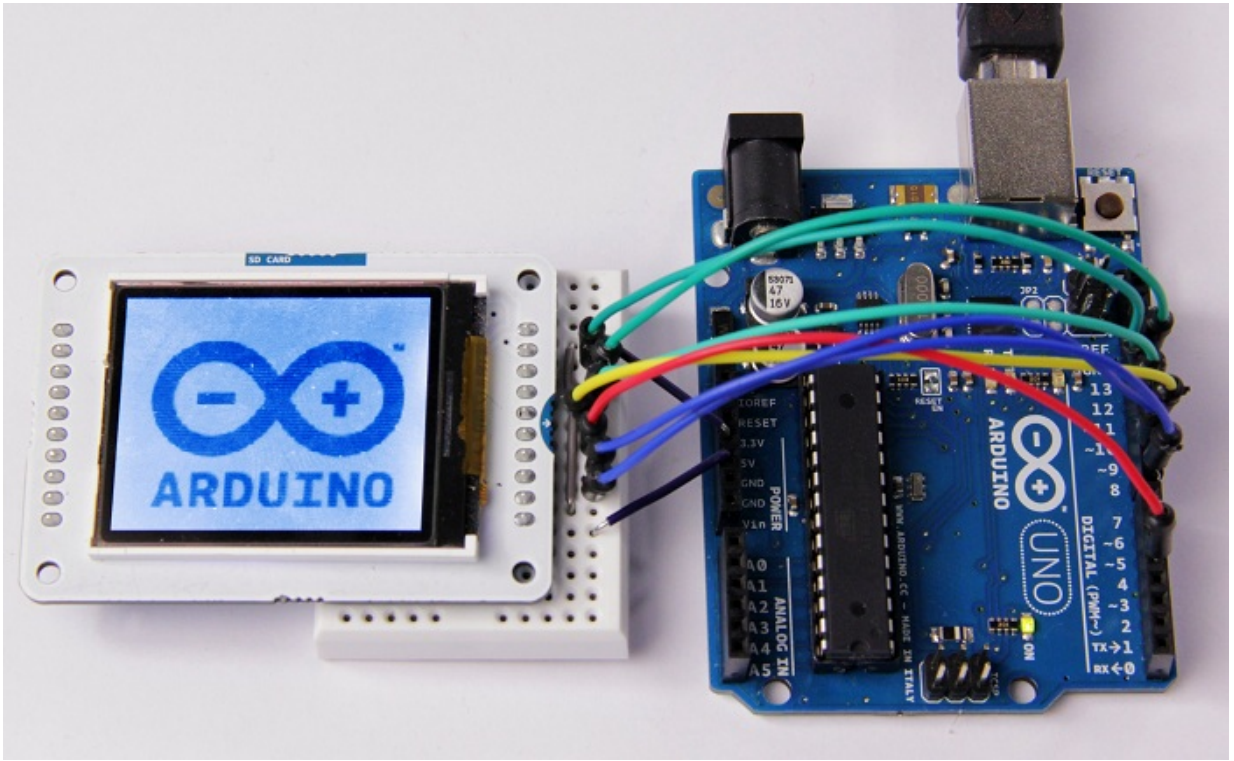
- gemakkelijk gebruik met $LiquidCrystal$ bibliotheek
- functies: initialisatie, scrollen, wissen, schrijven
- cursor: blink, underscore, ...
- eigen karakters aanmaken (beperkt aantal)

Wat komt er op het scherm?

```
1  #include <LiquidCrystal.h>
2
3  LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
4    // RS, enable, D4..D7
5
6  void setup() {
7    lcd.begin(16, 2);
8    lcd.print("hello, world!");
9  }
10
11 void loop() {
12   lcd.setCursor(0, 1);  // 2de lijn
13   lcd.print(millis() / 1000);
14 }
```

TFT

- grafisch display
- meestal kleuren
- wordt aangestuurd via matrix of serieel protocol

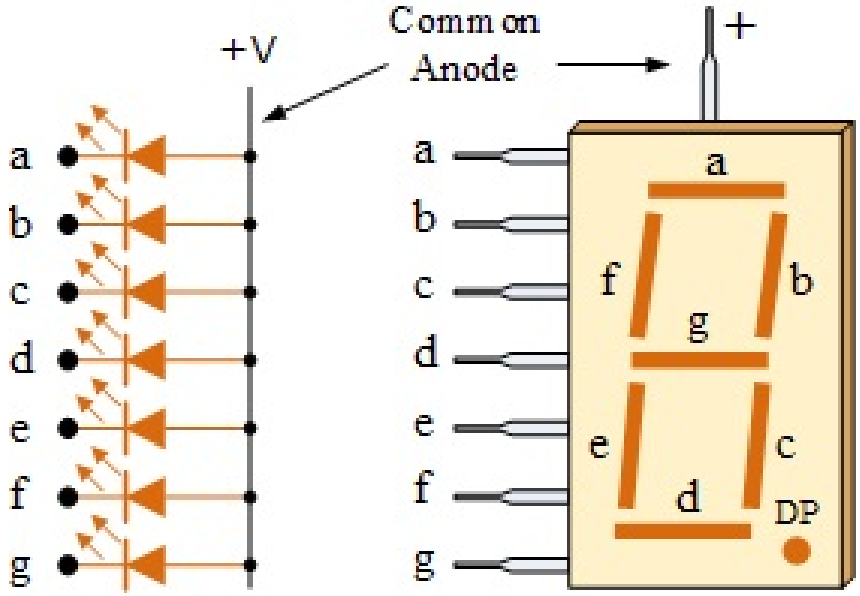


7-segment

- wordt gebruikt om (kleine) getallen weer te geven
- bv. weegschalen, klokradio's, ...
- bestaat uit individuele LED's
- betrouwbaar en zuinig (met verlichting)

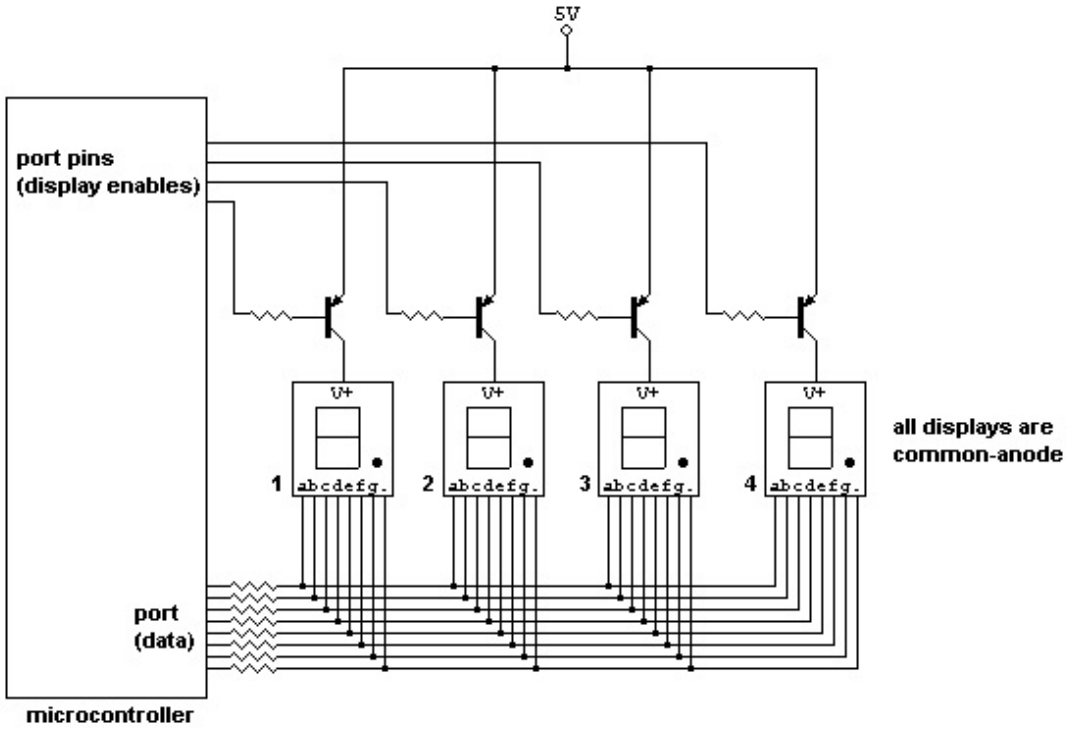
Opbouw

- ieder segment (+punt) heeft aansluiting
- gemeenschappelijke anode of kathode
- stroombegrenzing nodig (weerstand)



Multiplex

- alle gelijke segmenten worden verbonden met elkaar
- snel in de tijd afwisselen met hogere stroom
- rekenintensief voor microcontroller en veel pinnen nodig

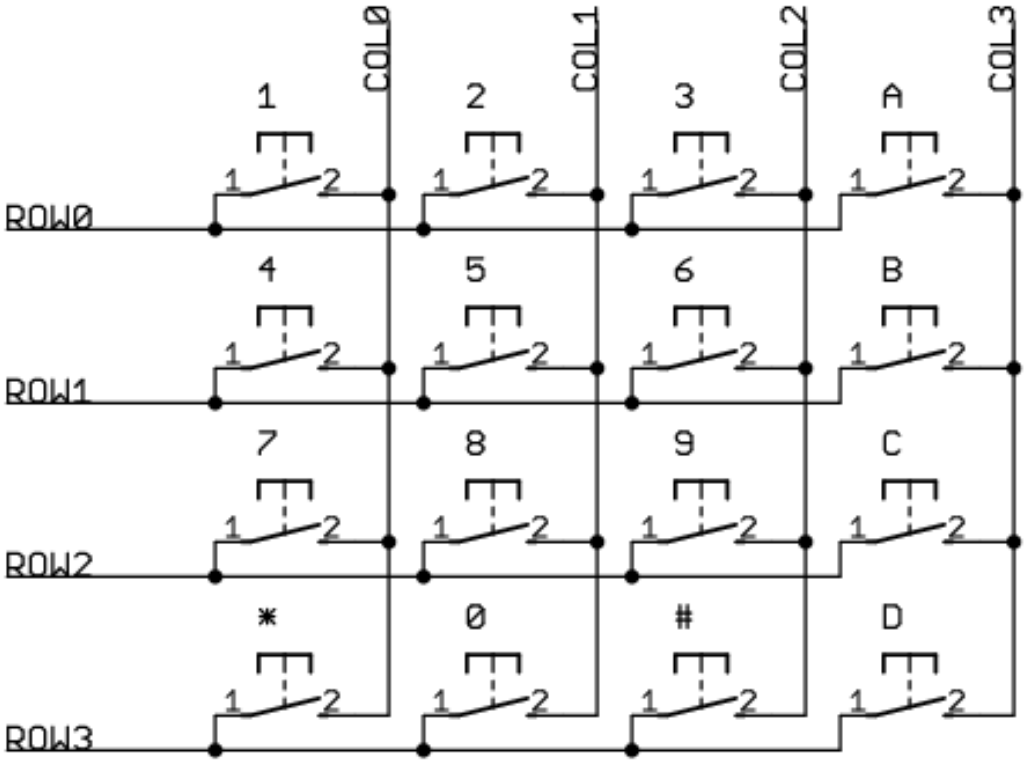


Keypad

- mini toetsenbord voor input van getallen
- bv. in snoepautomaat

Opbouw

- opgebouwd uit schakelaars in rijen en kolommen
- matrixstructuur



Uitlezen

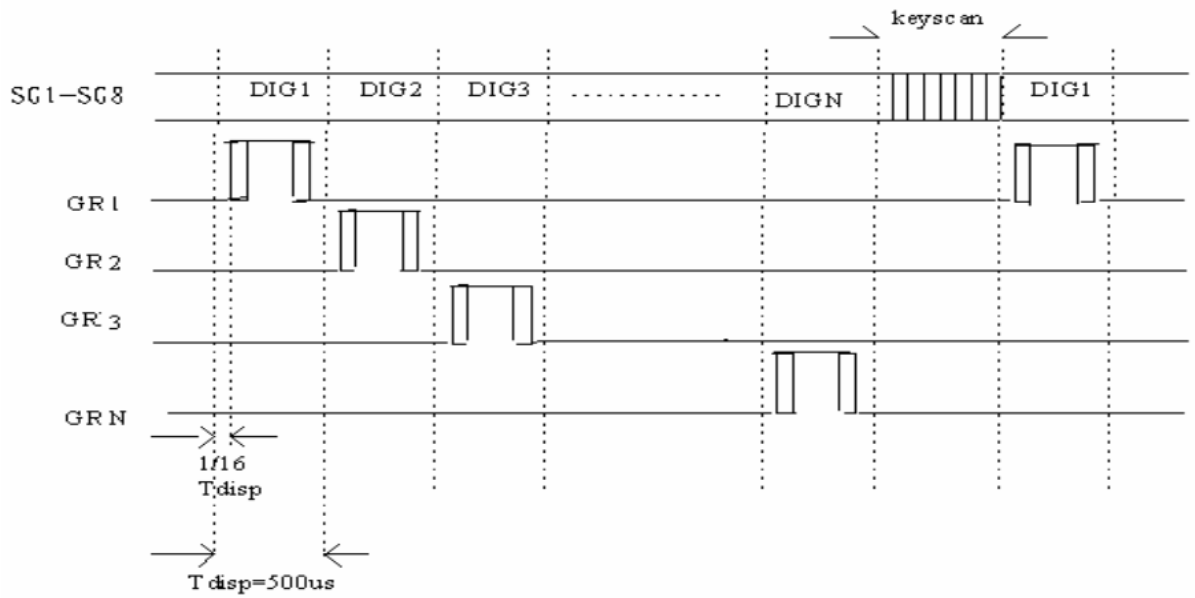
- rijen als input (met pullup weerstanden)
- kolommen als output
- kolommen 1 voor 1 laag maken en telkens elke rij inlezen
- combinatie kolom/rij geeft toetswaarde weer
- rekenintensief (aansturen + uitlezen)
- veel pinnen nodig

TM1636 display driver

- speciaal IC om display en keypad aan te sturen
- vermindert aantal benodigde pinnen tot 2 (data + clk)
- kan LED's rechtstreeks aansturen (zonder weerstand)
- helderheid kan geregeld worden
- matrix uitgang (max. 4 x 7-segment)

Protocol

- wordt aangestuurd via 2-draads protocol
- dialect van I^2C (zie later)
- bij elk kloksignaal kan bit doorgestuurd worden
- 7-segment en keypad kunnen samen gebruikt worden



Bibliotheek TM1636

- functies om 7-segment display aan te sturen
- `init()` : initialiseren (helderheid)
- `display()` : array van 4 bytes tonen op display
- `point()` : vlag dubbele punt instellen
- `clearDisplay()`: display wissen

Voorbeeld

```
1  #include <TM1636.h>
2  TM1636 tm1636(7,8);  // CLK, DATA
3  int8_t disp[4];
4
5  void setup() {
6    tm1636.init();
7  }
8
9  void loop() {
10   disp[0]=1; disp[1]=2;
11   disp[2]=3; disp[3]=4;
12   tm1636.display(disp);
13   delay(500);
14 }
```

Eigen karakters

TM1636.cpp

```
1 static int8_t TubeTab[] =
2    {0x3f,0x06,0x5b,0x4f,
3     0x66,0x6d,0x7d,0x07,
4     0x7f,0x6f,0x77,0x7c,
5     0x39,0x5e,0x79,0x71,
6     0x40,0x00};//0~9,A,b,C,d,E,F,"-"," "
```

- er kunnen eigen karakters achteraan toegevoegd worden
- decoderen welke bits bij welke segmenten horen