

CAR – Dag 8

C, Arduino & Robots

C, Arduino en Robots

- ▶ Programmeertalen C en C++
- ▶ Robotics
 - State machine
 - PID, $X/Y/\varphi$
 - Subsumption
 - Sensoren
- ▶ Embedded programmeren
 - Serial port
 - Peripherals
 - Compiler + linken
 - Interrupts

Interrupts

- ▶ Onderbreek het programma om tussendoor (even) wat anders te doen.
- ▶ Om snel te kunnen reageren.
- ▶ Bronnen
 - Timer (vaste interval)
 - Interrupt pin
 - Serieële poort
 - Pin change

Interrupts Oefening

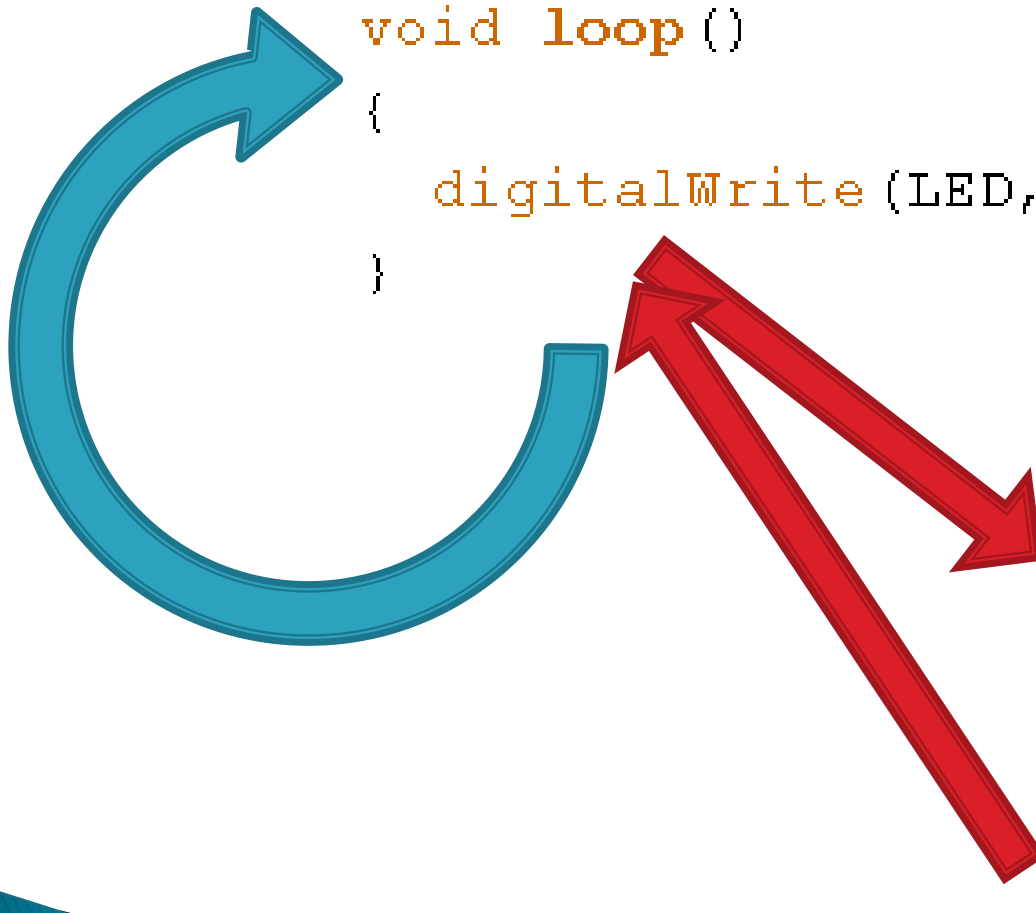
- ▶ P10
- ▶ Knop op pin 2
- ▶ Led op pin 13 (standaard arduino LED).
- ▶ Hoe werkt dit?

Interrupts

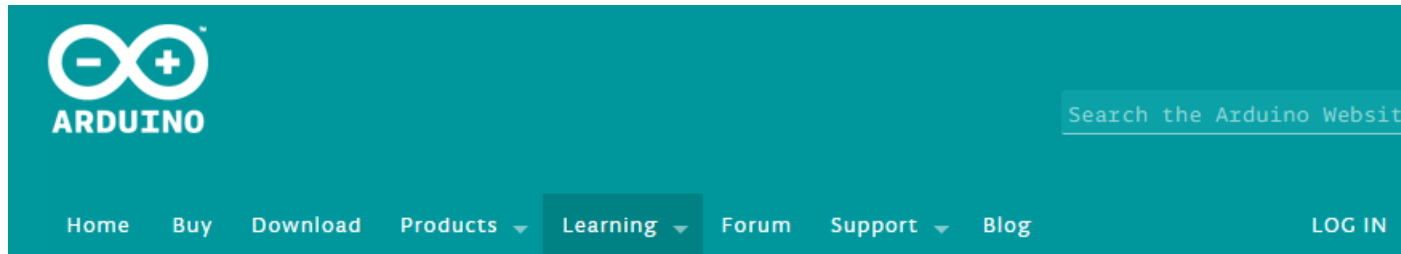
```
attachInterrupt(0, Toggle, FALLING);
```

```
void loop()  
{  
    digitalWrite(LED, state);  
}
```

```
void Toggle()  
{  
    state = !state;  
    Serial.print('*');  
}
```



Interrupts – Documentatie



[Reference](#) [Language](#) | [Libraries](#) | [Comparison](#) | [Changes](#)

attachInterrupt()

Description

Specifies a named Interrupt Service Routine (ISR) to call when an interrupt occurs. Replaces any previous function that was attached to the interrupt. Most Arduino boards have two external interrupts: numbers 0 (on digital pin 2) and 1 (on digital pin 3). The table below shows the available interrupt pins on various boards.

Board	int.0	int.1	int.2	int.3	int.4	int.5
Uno, Ethernet	2	3				
Mega2560	2	3	21	20	19	18
Leonardo	3	2	0	1	7	
Due			(see below)			

Interrupts – aandachtspunten

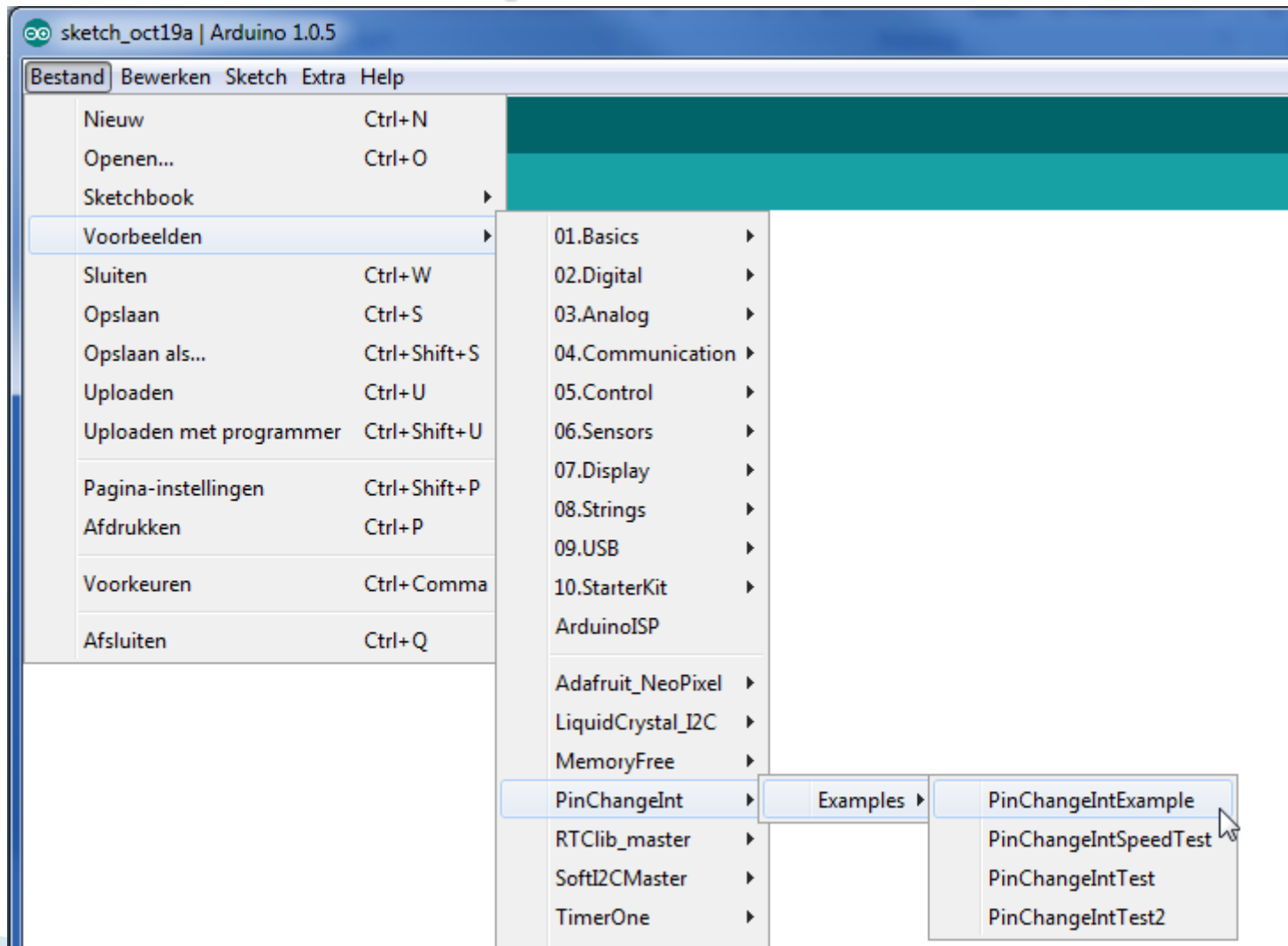
- ▶ volatile
- ▶ hou het kort;
 - niet (veel) printen.
 - geen delay
- ▶ Geen complexe functies (als printf) aanroepen (re-entrancy)
- ▶ delen van variabelen
 - volatile (vertel het de compiler)
 - atomic (niet storen als > 1 byte)
 - goed nadenken over welke functie de variabele verandert en welke functies deze gebruiken.

Arduino library

- ▶ Uitbreiding van de Arduino omgeving.
- ▶ Extra functies
- ▶ Arduino library bevat ook voorbeelden.
- ▶ Let op: niet alle libraries werken even goed samen en op alle boards!

- ▶ Installatie:
 - Downloaden
 - Uitpakken in sketchbook\libraries
 - herstart IDE

Arduino library



Encoders – interrupt

```
void IsrEncoderR()
{
    EncoderRTeller++;
}
```

- ▶ Basisversie telt pulsen
- ▶ Maakt gebruik van PinChangeInt
<https://code.google.com/p/arduino-pinchangeint/>
- ▶ Nooit meer pulsen missen

```
void IsrEncoderL()
{
    EncoderLTeller++;
}
```

```
void EncoderSetup()
{
    pinMode(ENCODER_L_PIN, INPUT_PULLUP);
    pinMode(ENCODER_R_PIN, INPUT_PULLUP);

    PCintPort::attachInterrupt(ENCODER_L_PIN, &IsrEncoderL, CHANGE);
    PCintPort::attachInterrupt(ENCODER_R_PIN, &IsrEncoderR, CHANGE);
}
```

Encoders – interrupt

les_8_p20_encoders_1

EncoderInterrupt_basic

```
volatile int EncoderLTeller, EncoderRTeller;

//-----
// EncoderRead - Lees de encoders
//-----
// Let op: parameters zijn referenties en kunnen dus gebruikt
// worden om waarden terug te geven.
//-----
void EncoderRead (int &Left, int &Right)
{
    // maak copy zonder dat er interrupts tussendoor komen.
    noInterrupts();
    // critical, time-sensitive code here
    Left = EncoderLTeller;
    Right = EncoderRTeller;
    interrupts();
}
```

Snelheidsmeting methode 1

- ▶ Pulsen tellen
- ▶ Stel:
 - Snelheid van 20 cm/s
 - Takt interval 100 ms
 - Encoder resolutie 5 mm/puls

Wat is de uitkomst van de meting?

Snelheidsmeting methode 1

- ▶ 4 pulsen \rightarrow 20 cm/s
 - ▶ 3 pulsen \rightarrow 15 cm/s
 - ▶ 5 pulsen \rightarrow 25 cm/s
-
- \Rightarrow Resolutie van de meting is 5 cm/s.
 - \Rightarrow 25% van de meetwaarde @ 20 cm/s

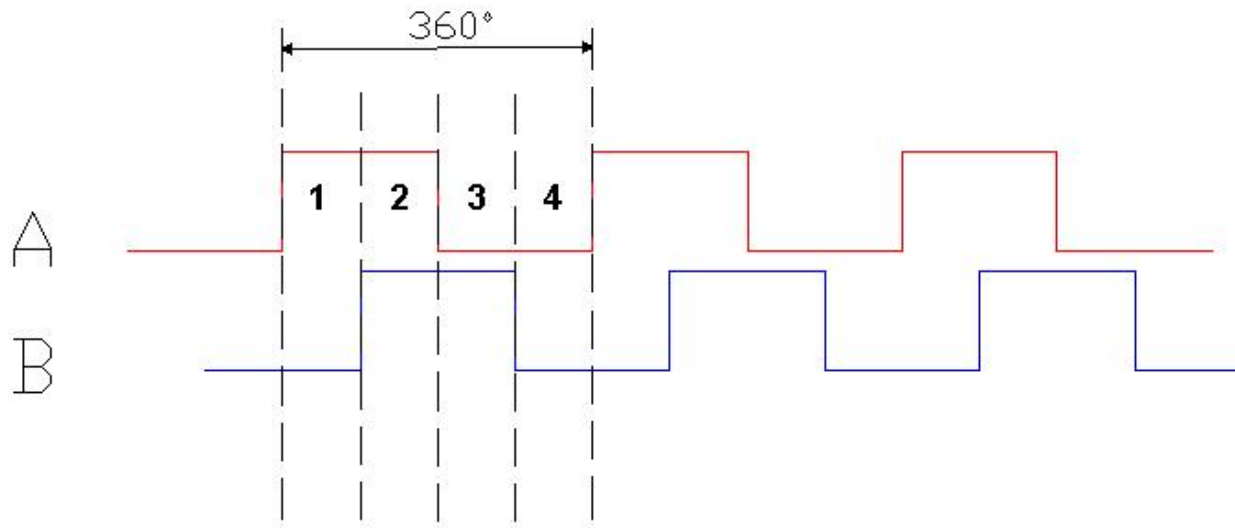
Snelheidsmeting methode 2

- ▶ Periodetijd meten
 - $18 \text{ cm/s} \Rightarrow 1000 * 0.5 / 18 = 28 \text{ ms periode}$
 - $20 \text{ cm/s} \Rightarrow 1000 * 0.5 / 20 = 25 \text{ ms periode}$
 - $22 \text{ cm/s} \Rightarrow 1000 * 0.5 / 22 = 23 \text{ ms periode}$
- ▶ Interrupt routine legt exacte tijd van flank vast.
- ▶ Variabele meettijd, gemiddeld 0.025 seconde

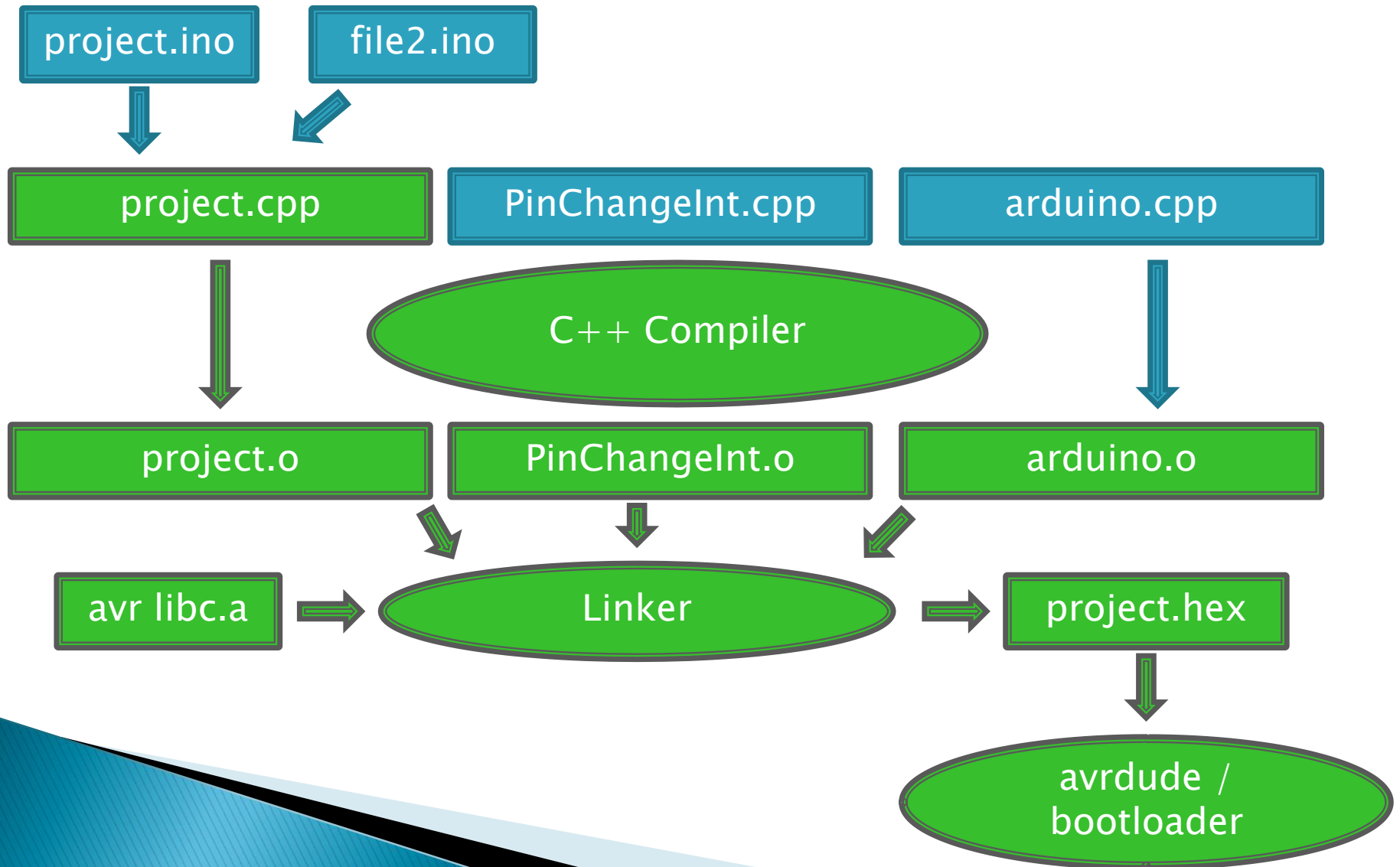
```
void EncoderPrint()  
{  
    printf("EncoderInterrupt Speed: %d / %d, Count: %d / %d\n",  
        LeftSpeed(), RightSpeed(), EncoderLTeller, EncoderRTeller);  
}
```

Encoders en draairichting

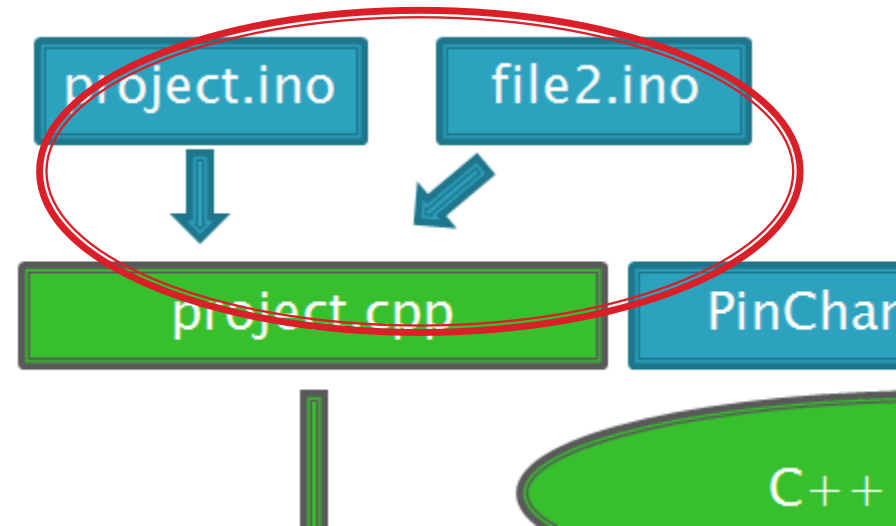
- ▶ Quadratuur encoders: 2 uitgangen



Arduino bouwproces

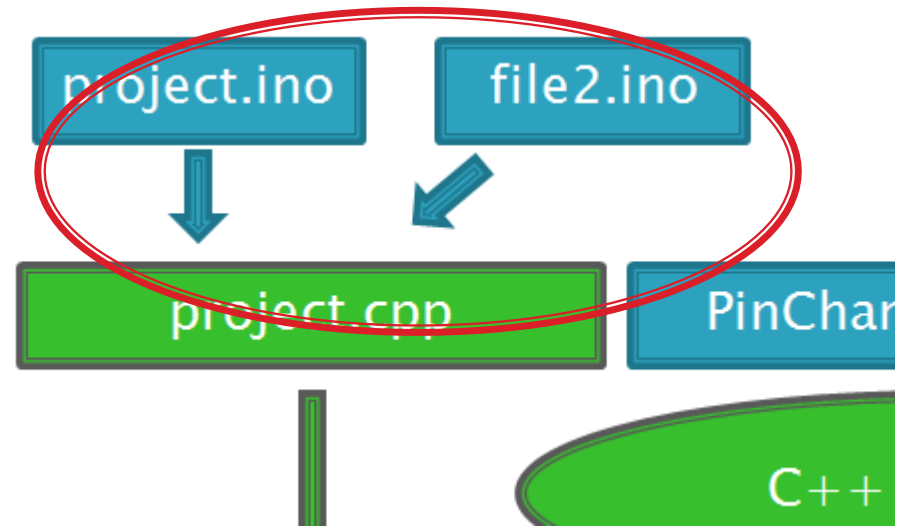


Kracht & zwakte



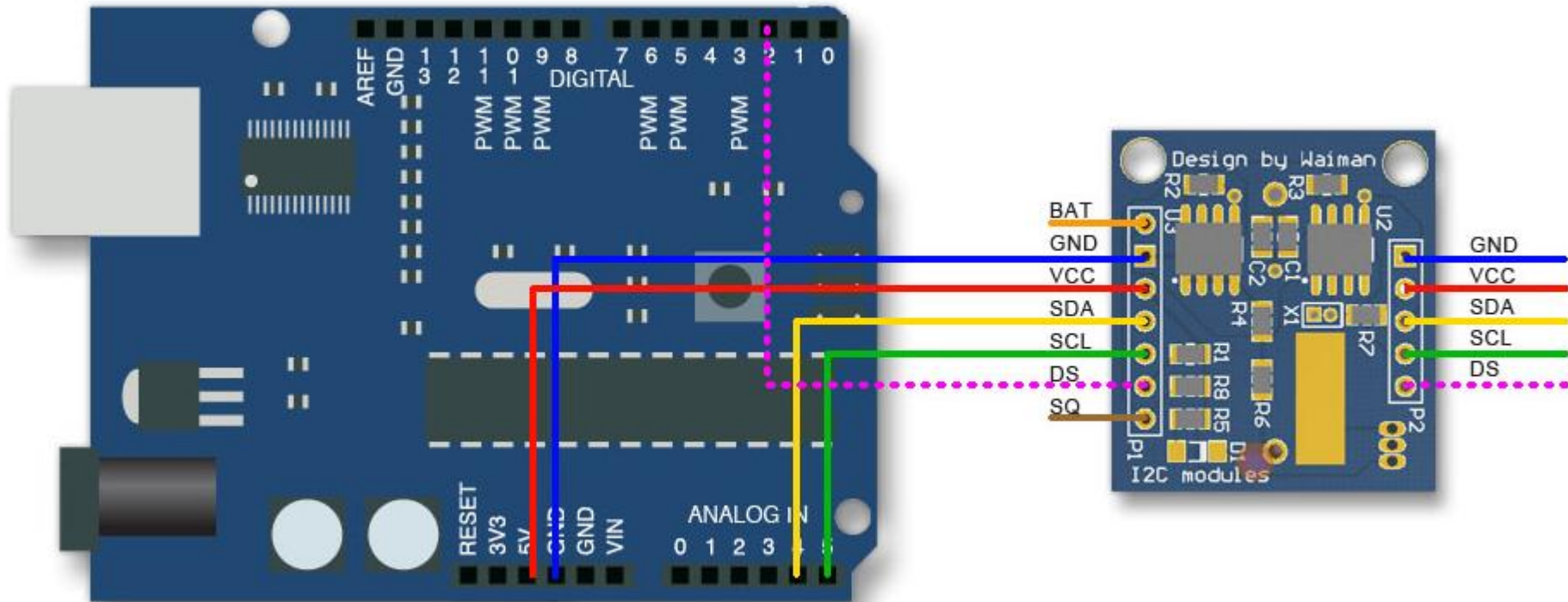
- ▶ Snelle start
- ▶ Prototypes worden automatisch gemaakt
- ▶ cross platform (meerdere bordjes)
- ▶ veel standaard functies onderdeel van Arduino
- ▶ Via 'libraries' kunnen eenvoudig nieuwe functies met voorbeelden worden toegevoegd.

Kracht & zwakte

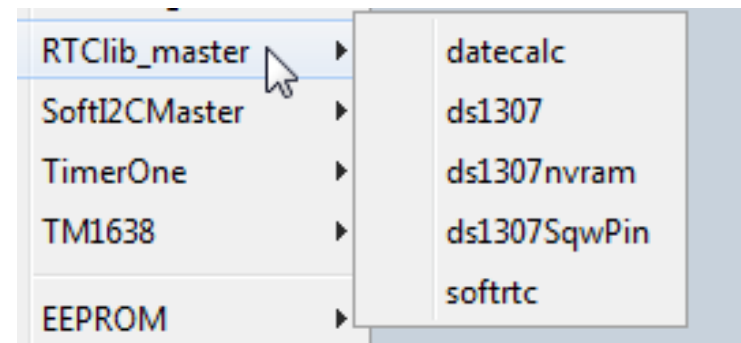
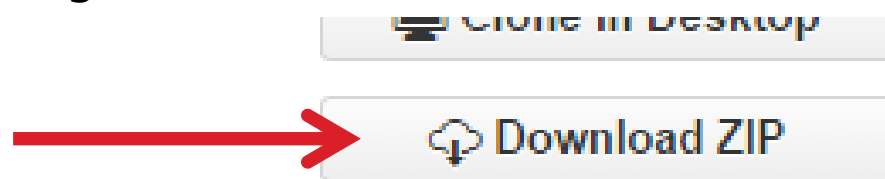


- ▶ ‘Libraries’ worden vaak gecompileerd (kost tijd).
 - ▶ ‘Libraries’ werken niet altijd samen.
 - ▶ Onduidelijk welke resources gebruikt worden.
 - ▶ Geen controle over volgorde .ino bestanden.
 - ▶ Slechte meldingen als er iets mis gaat buiten de .ino bestanden (bijvoorbeeld in eigen ‘library’).
 - ▶ IDE heeft beperkte functionaliteit
- => Minder geschikt voor grotere projecten.

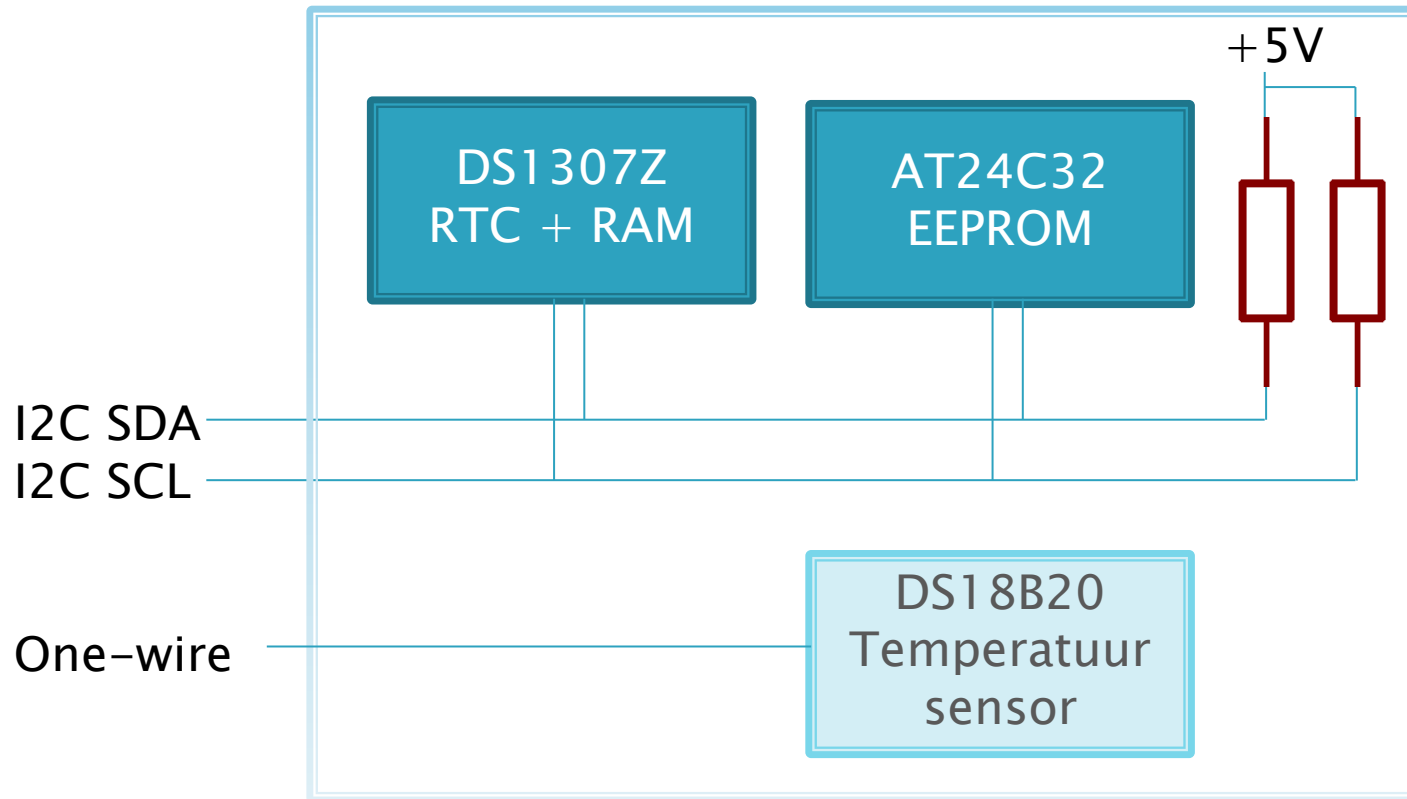
RTC



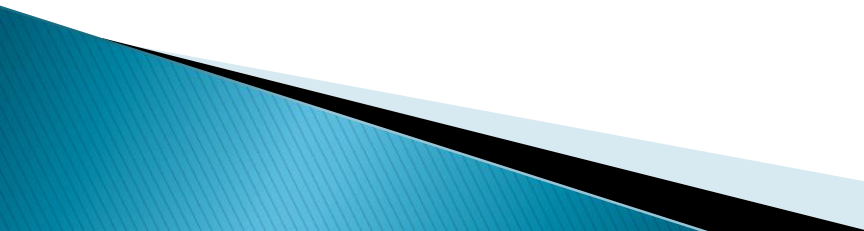
<https://github.com/adafruit/RTClib>



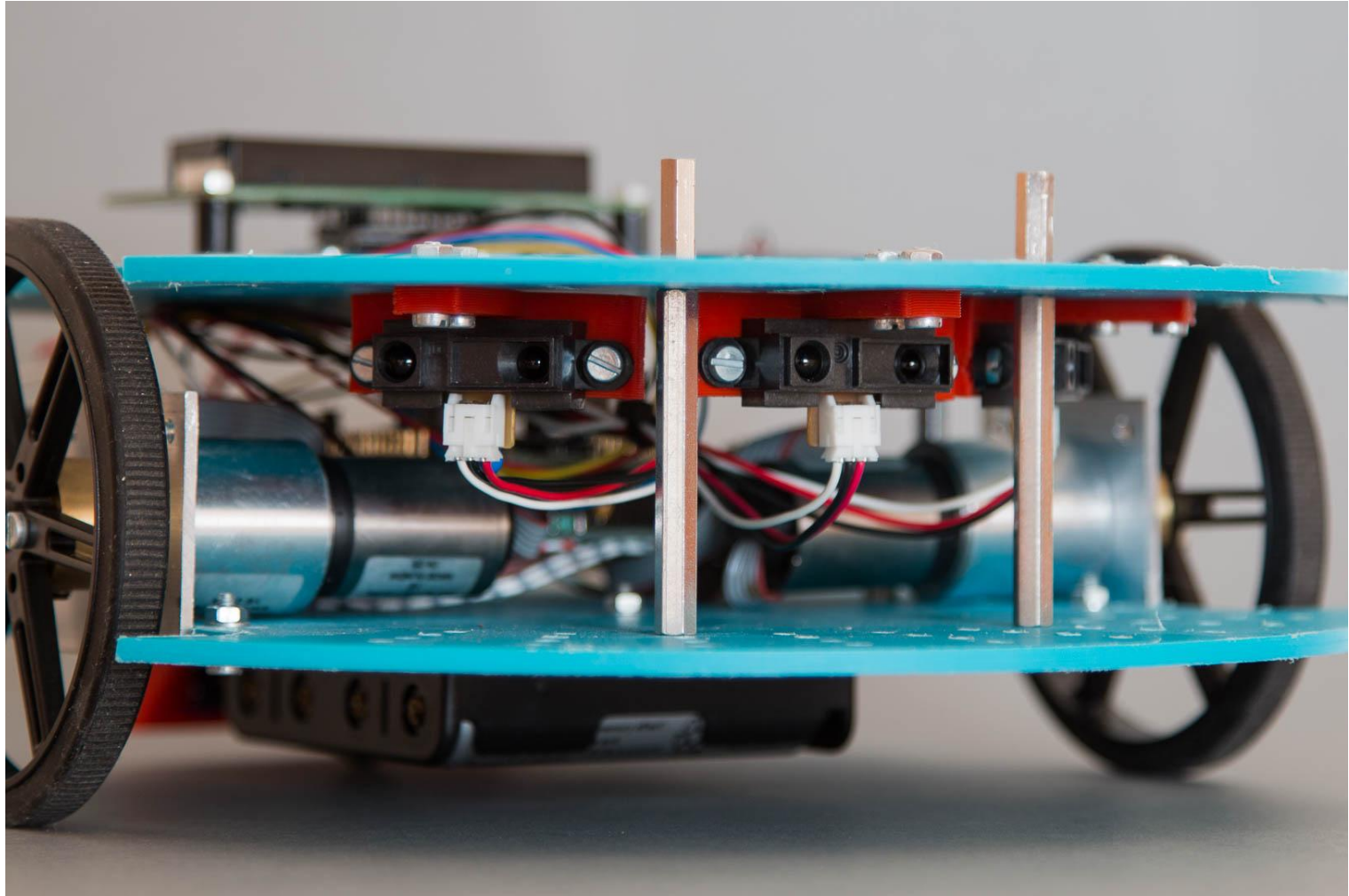
RTC



I2C Interface

- ▶ 2-draads master/slave bus, tot 127 slaves
 - ▶ vanaf 0, 100 kbps, 400 kbps, 1 Mbps, 3.4 Mbps
 - ▶ Master bepaalt timing
 - ▶ 6 bytes lezen (+ 4 bytes overhead) in minder dan 1 ms @ 100kpbs.
 - ▶ Ideale interface voor uitbreiding van je robot of arduino project.
 - ▶ Arduino noemt dit 'wire' interface.
 - ▶ Atmel noemt dit TWI (Two Wire Interface)
- 

2 motoren en 3 Sharp sensoren



Acht lijn sensoren



- ▶ Sensor 0 tot en met 7

Array

```
int Waarden[12]; // array met positie 0 t/m 11
```

0	1	2	3	4	5	6	7	8	9	10	11
---	---	---	---	---	---	---	---	---	---	----	----



Array voorbeeld

les_8_p50_array

```
int Waarden[12]; // array met positie 0 t/m 11
```

```
for (int i=0; i<12; i++) {  
    Waarden[i] = 0;  
}
```

```
PrintWaarden();
```

```
Waarden[3] = 7;
```

```
Waarden[4] = Waarden [3] + 2;
```

```
PrintWaarden();
```

```
for (int i=0; i<12; i++) {  
    Waarden[i]++;  
}
```

```
PrintWaarden();
```

```
void PrintWaarden()  
{
```

```
    // uitgeschreven
```

```
    printf("Array waarden: %d %d %d %d %d %d %d %d %d %d %d %d  
        Waarden[0], Waarden[1], Waarden[2], Waarden[3],  
        Waarden[4], Waarden[5], Waarden[6], Waarden[7],  
        Waarden[8], Waarden[9], Waarden[10], Waarden[11]);
```

```
    // alternatief: in lus
```

```
    for (int i=0; i<12; i++) {  
        printf("Waarde[%d] = %d\n", i, Waarden[i]);  
    }  
}
```

Structures

les_8_p60_struct

my.h

```
struct Sharp {  
    int Pin;  
    int Raw;  
    long Middelen;  
  
    int Afstand;  
};
```

les_8_p60_struct

my.h

```
#include "my.h" // struct definitities  
                //includen *voor* alle andere code  
  
Sharp SharpL, SharpM, SharpR;  
  
void setup()  
{  
    SharpL.Pin = A0;  
    SharpM.Pin = A1;  
    SharpR.Pin = A3;  
}
```

Struct

```
void loop()
{
    SharpAfstand(SharpL);
    SharpAfstand(SharpM);
    SharpAfstand(SharpR);
}

int SharpAfstand(Sharp &Sh)
{
    Sh.Raw = analogRead(Sh.Pin);

    Sh.Middelen = Sh.Middelen - Sh.Middelen / 8 + Sh.Raw;

    int Gemiddelde = Sh.Middelen / 8;

    Sh.Afstand = (40*148) / Gemiddelde;
    return Sh.Afstand;
}
```

les_8_p60_struct

my.h

```
struct Sharp {
    int Pin;
    int Raw;
    long Middelen;

    int Afstand;
};
```

Classes

les_8_p70_class

my.h

```
#include "my.h" // class definitions includen
                // *voor* alle andere code

Sharp SharpL, SharpM, SharpR;

void setup()
{
    SharpL.Pin = A0;
    SharpM.Pin = A1;
    SharpR.Pin = A3;
}
```

les_8_p70_class

my.h

```
class Sharp {

public:

    int Pin;
    int Raw;

    int Afstand;

    void Sample();

private:
    long Middelen;

};
```

Classes

```
void loop()
{
    SharpL.Sample();
    SharpM.Sample();
    SharpR.Sample();
}

void Sharp::Sample()
{
    Raw = analogRead(Pin);

    Middelen = Middelen - Middelen / 8 + Raw;

    int Gemiddelde = Middelen / 8;

    Afstand = (40*148) / Gemiddelde;
}
```

```
les_8_p70_class  my.h

class Sharp {

public:

    int Pin;
    int Raw;

    int Afstand;


    void Sample();

private:
    long Middelen;

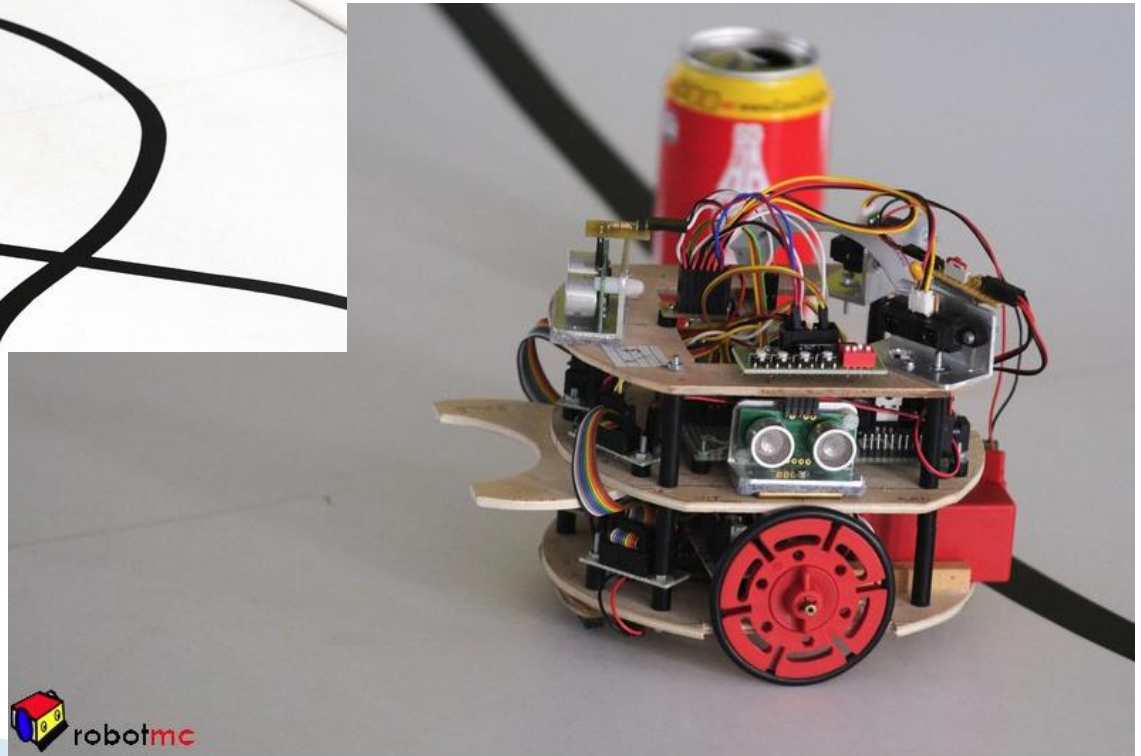
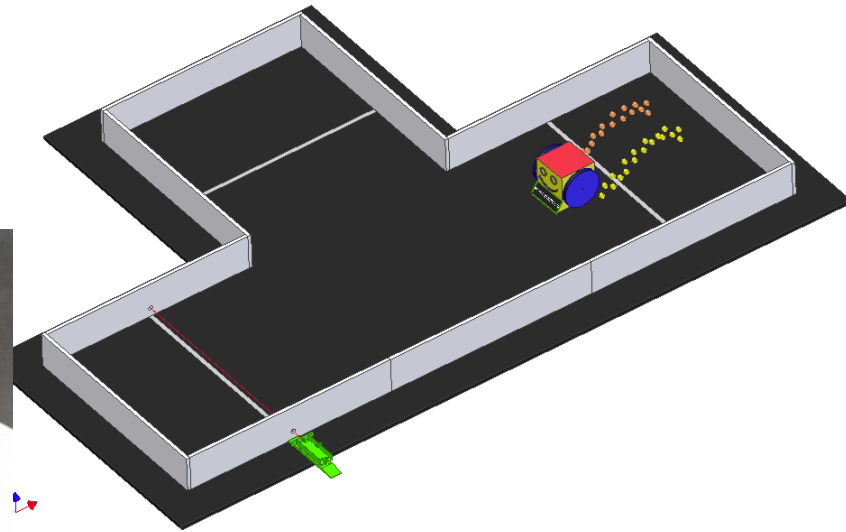
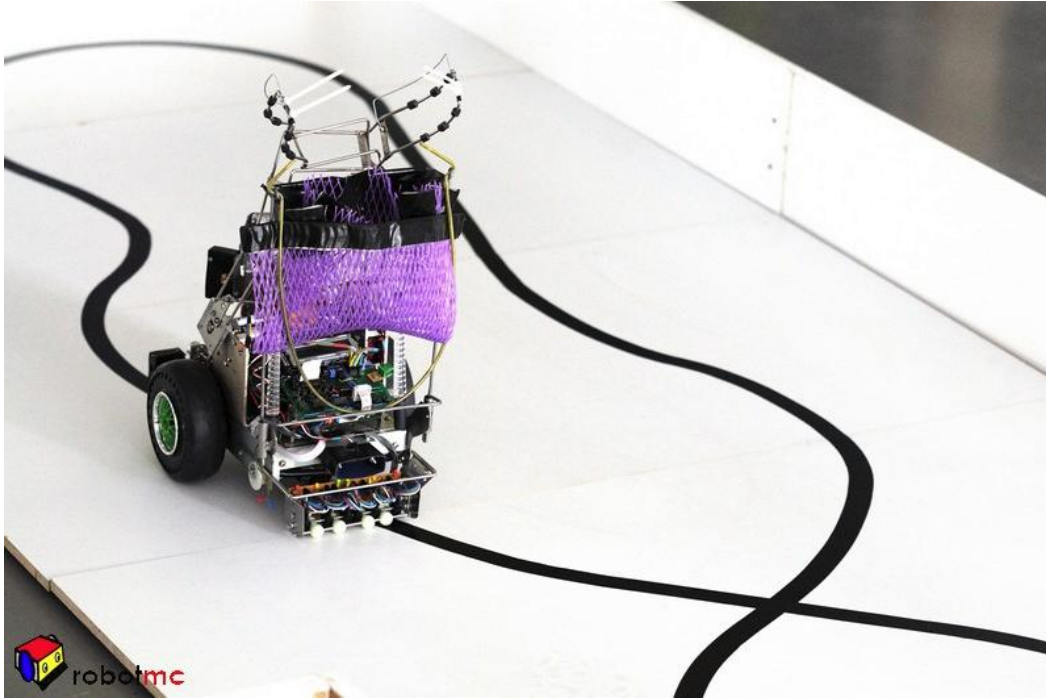
};
```

Belangrijkste leerpunten (Robots)

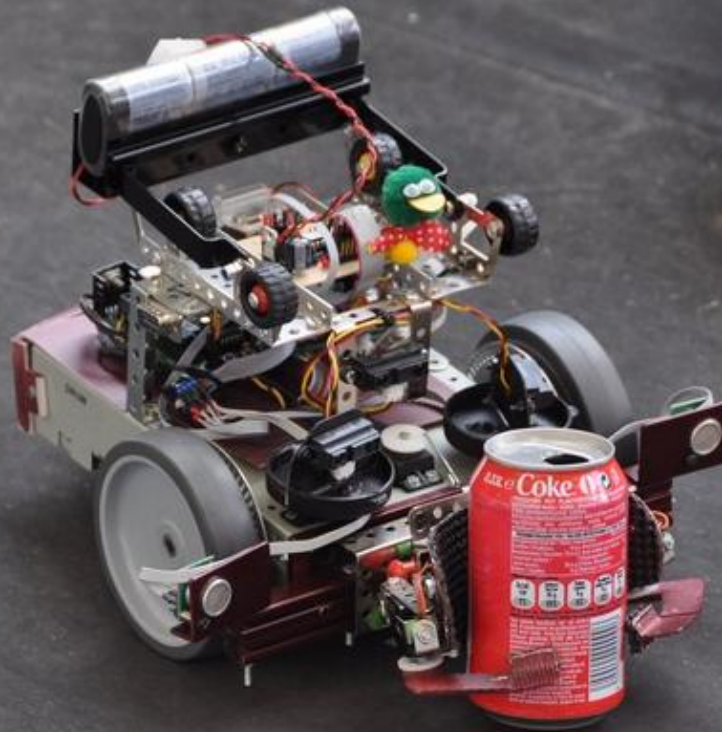
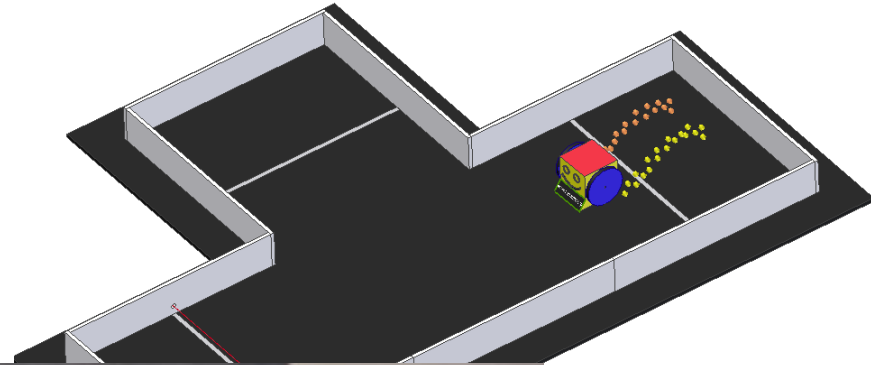
- ▶ Hoofdlus met strakke timing
 - ▶ Taken bouwen met state machines
 - ▶ Serieële poort voor debuggen (zien wat er gebeurt)
 - ▶ Harde aansturing van motoren

 - ▶ *Patronen*
 - ▶ *Regellus (P-regelaar)*
 - ▶ *Laagdoorlaatfilter*
 - ▶ *Encoders*
- 

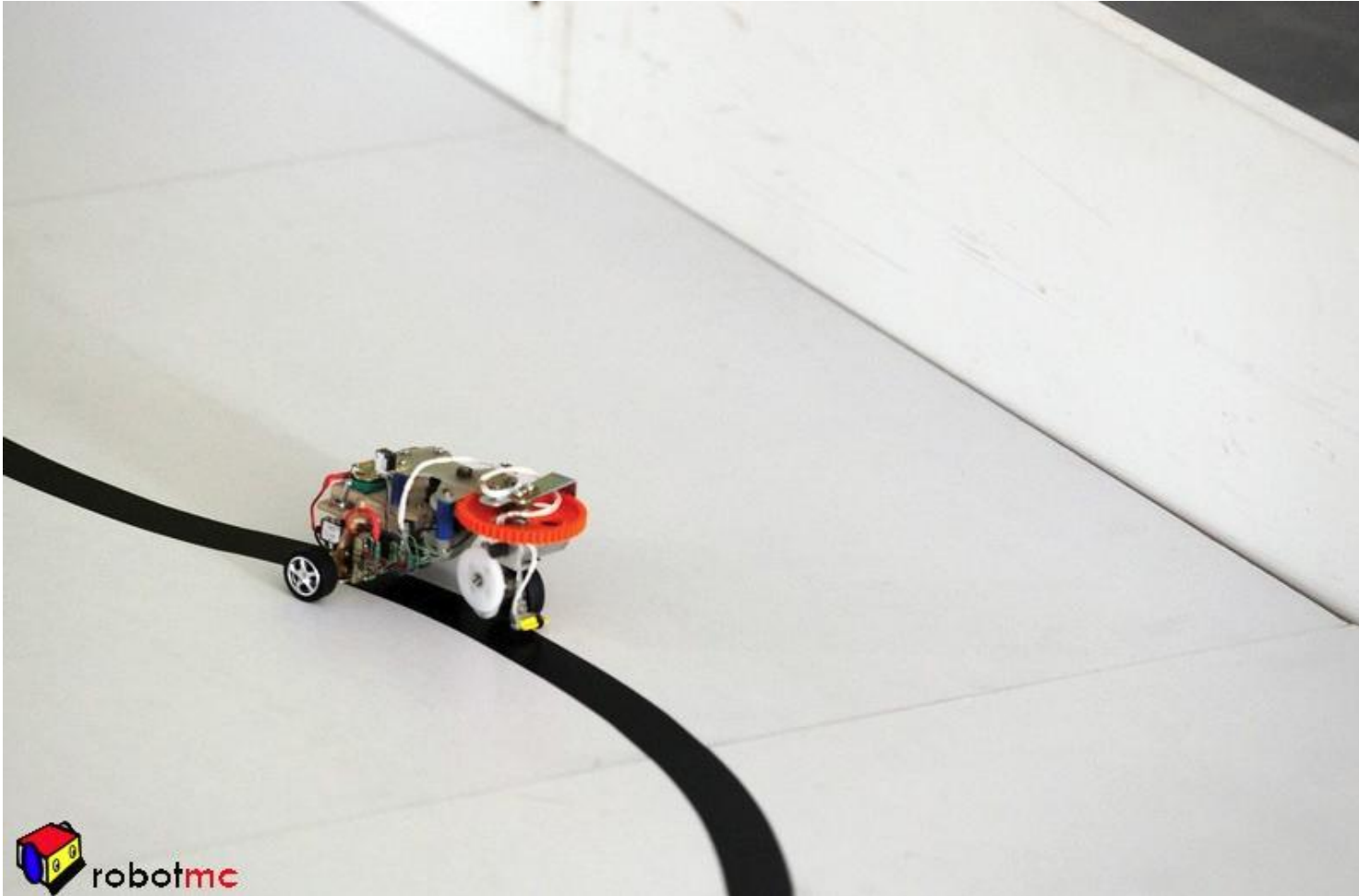
Roborama



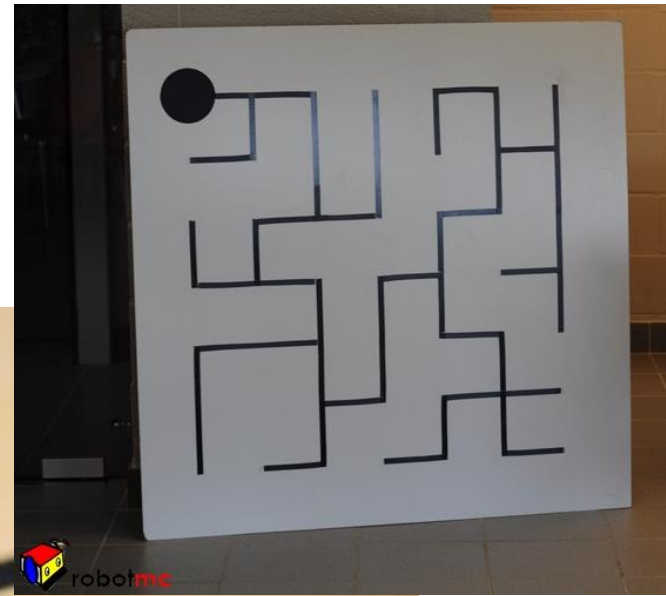
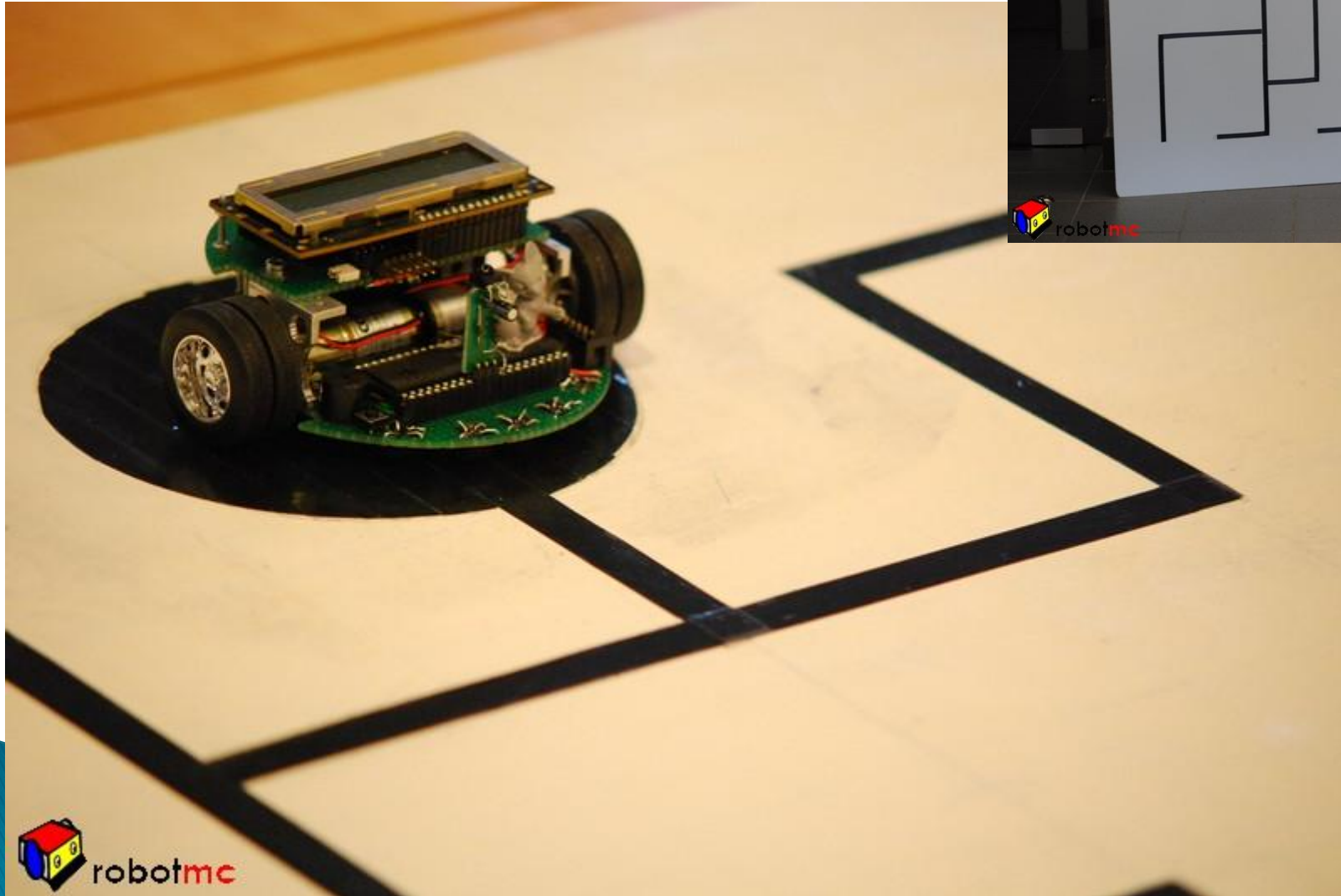
Roborama



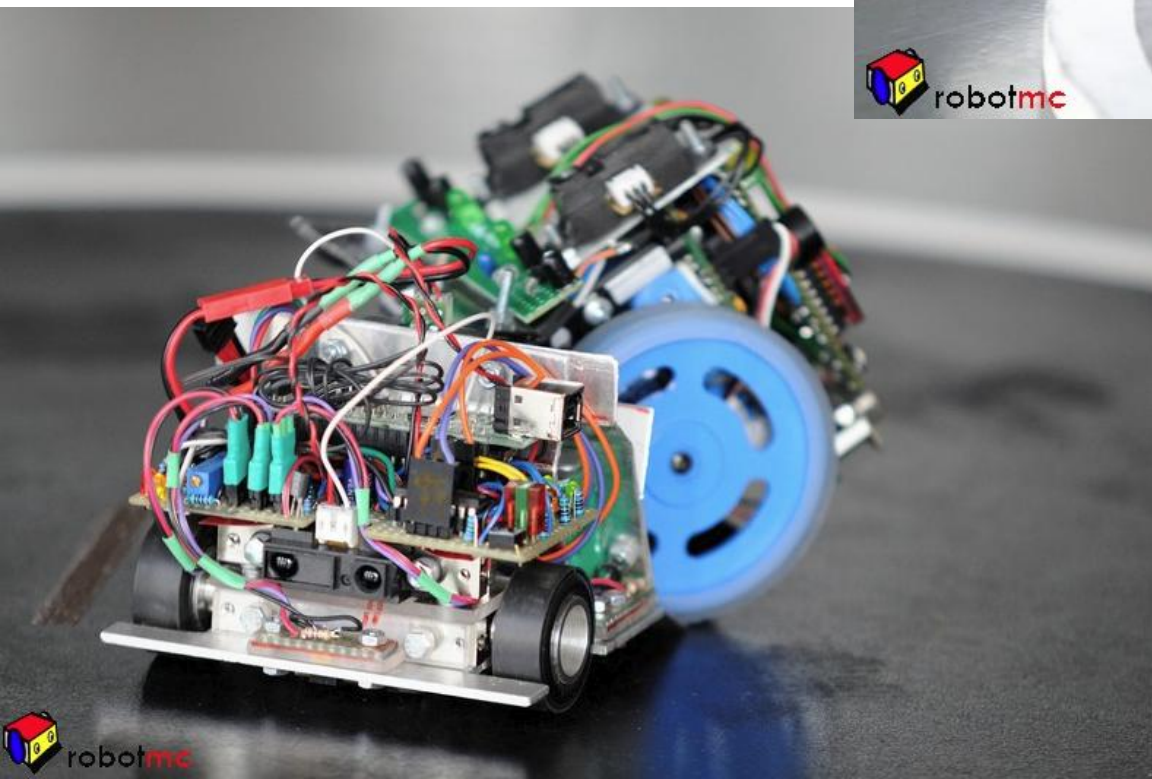
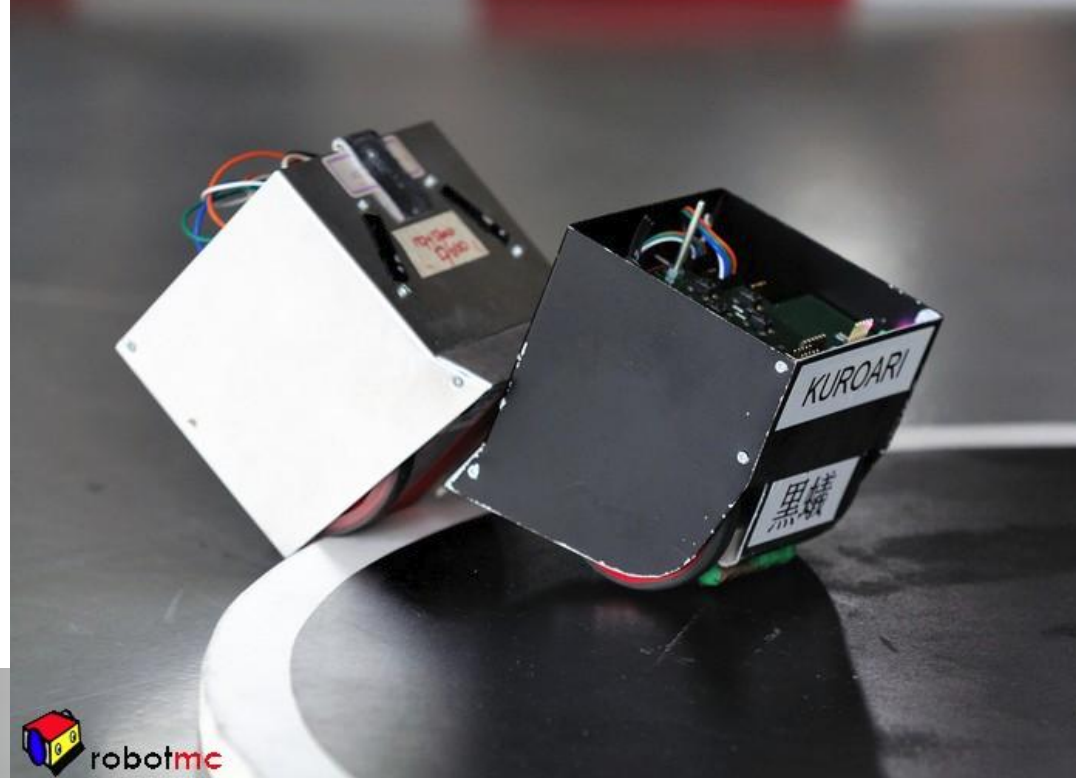
Lijnvolg race



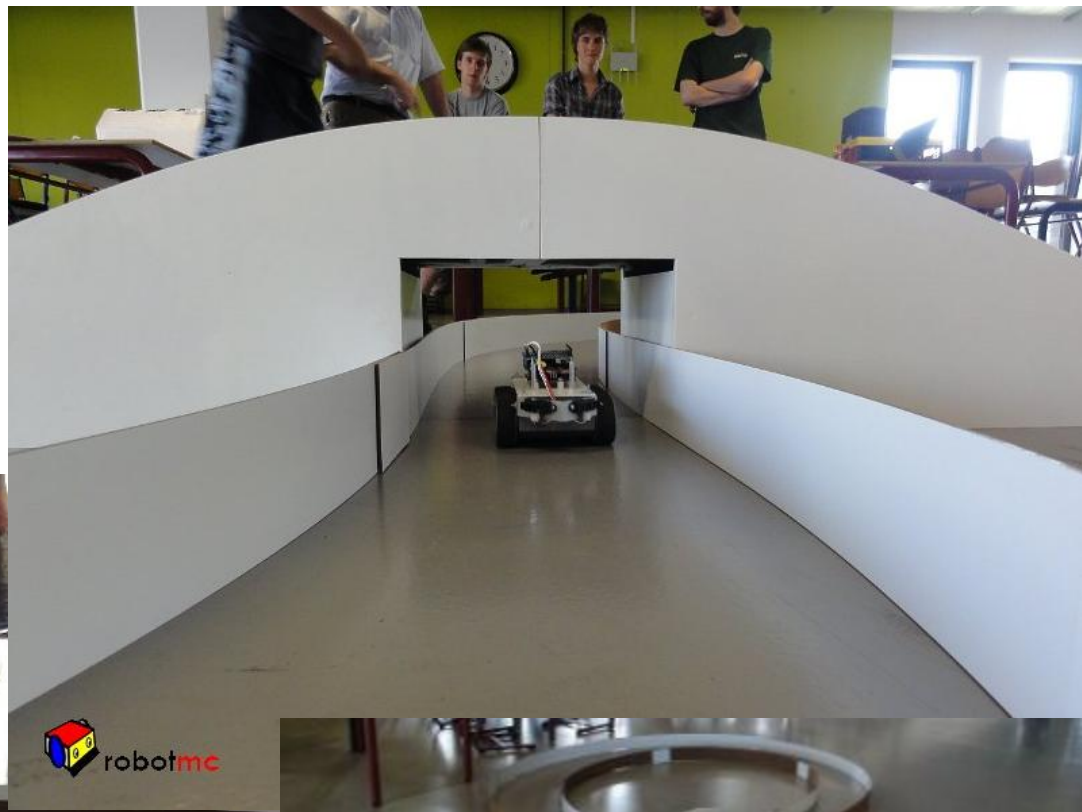
Line maze



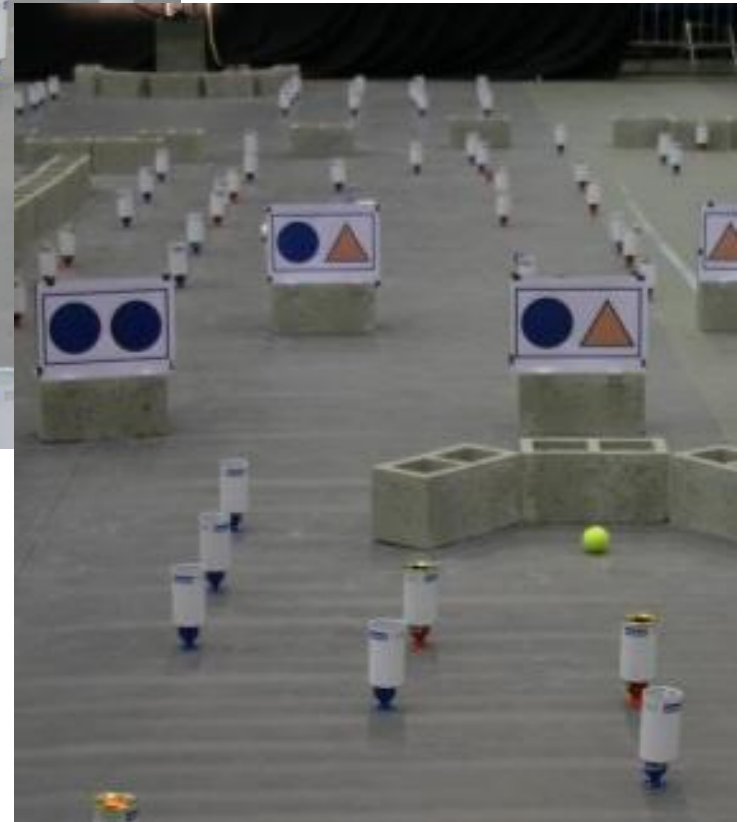
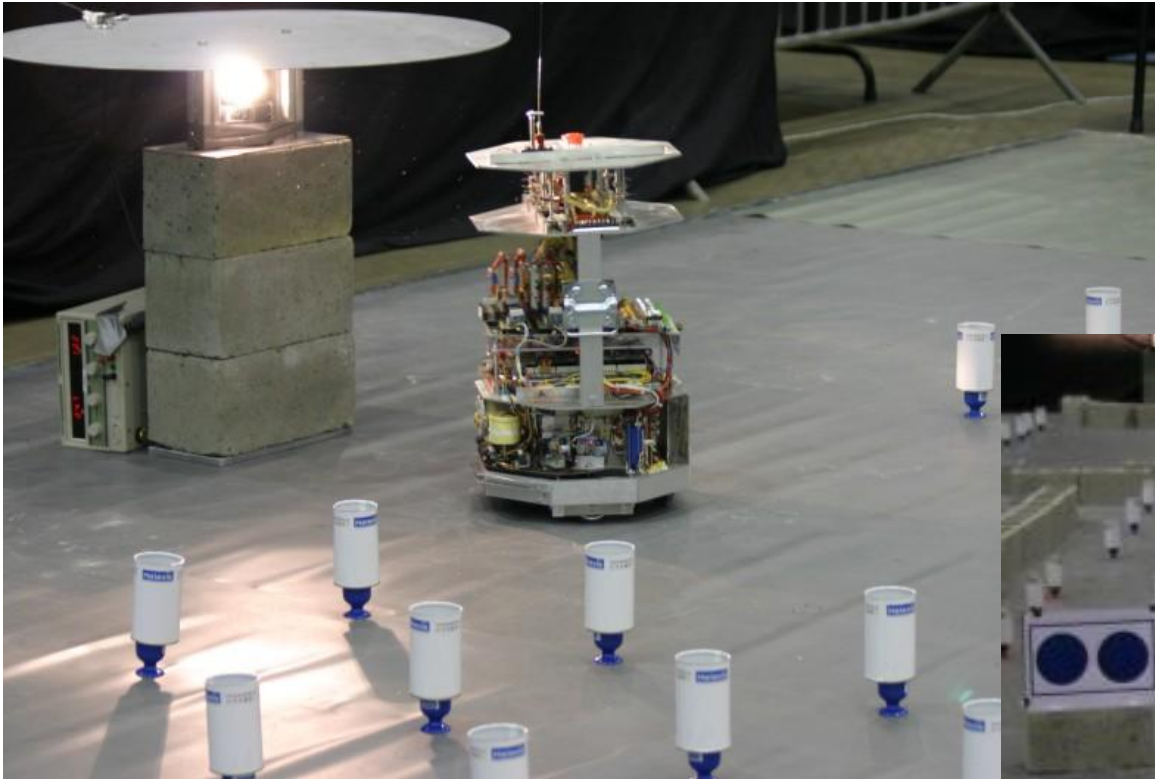
Mini sumo



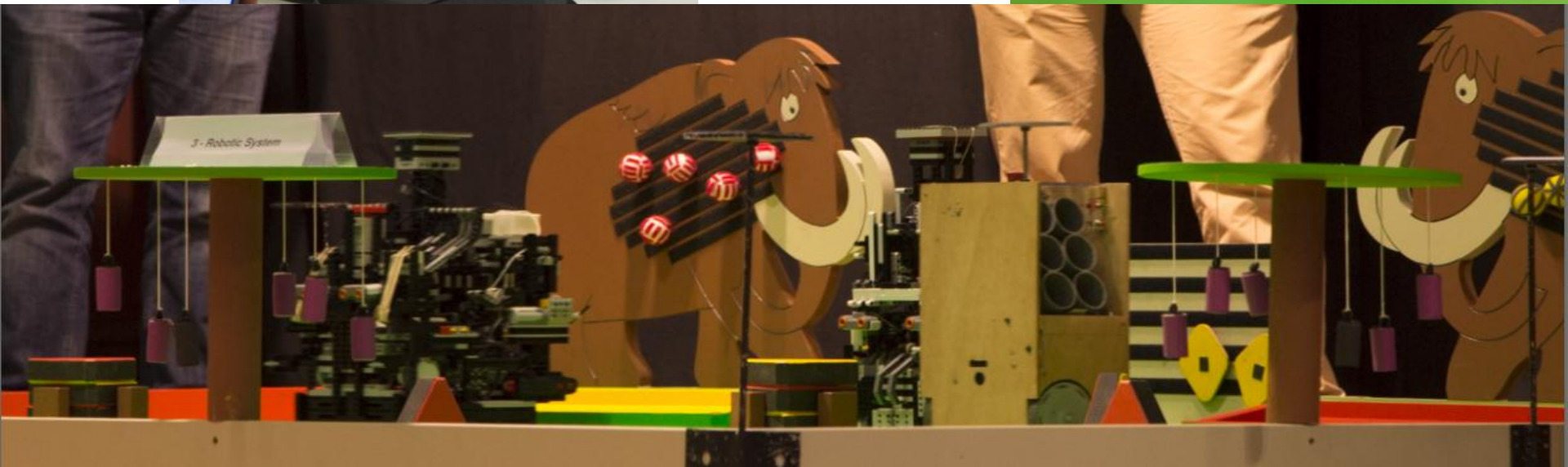
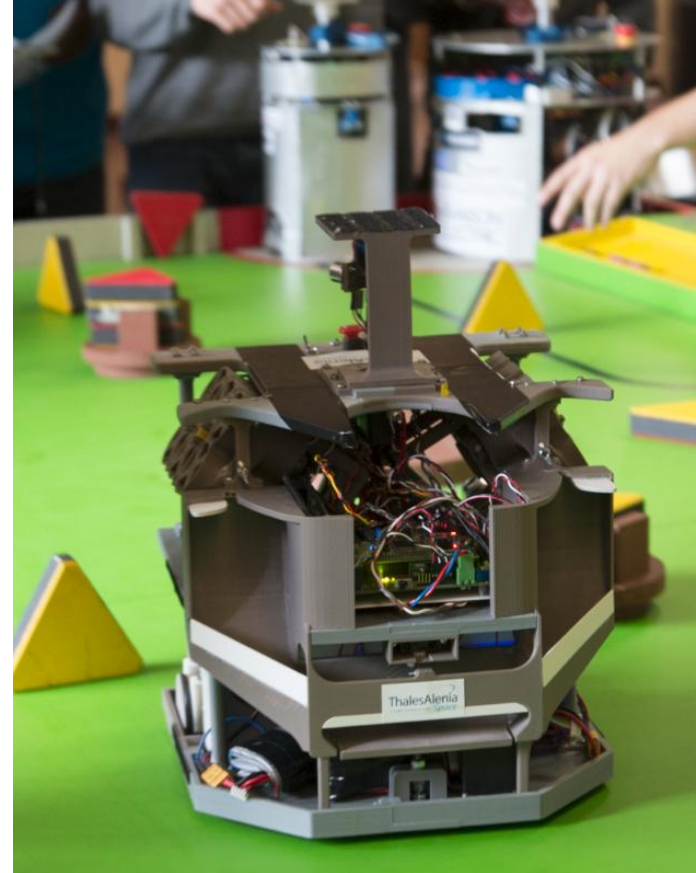
De Bergse brug



Doelzoeken & obstacels vermijden



Eurobot





The DARPA logo is located in the top right corner of the image. It consists of the word "DARPA" in white, bold, sans-serif capital letters, enclosed within a dark blue oval with a slight 3D effect.The background of the top section is a stylized, colorful illustration of a city skyline. It features various skyscrapers in shades of brown, tan, and grey, interspersed with green palm trees. A small white bird is flying in the light blue sky. The overall style is reminiscent of mid-century modern graphic design.

SAVE THE DATE

DARPA ROBOTICS CHALLENGE FINALS

Fairplex
Pomona, California
June 5-6, 2015

Finals Countdown
187 Days Remaining

The DARPA Robotics Challenge

   #DARPADRC

The DRC is a competition of robot systems and software teams vying to develop robots capable of assisting humans in responding to natural and man-made disasters. It was designed to be extremely difficult. Participating teams, representing some of the most advanced robotics research and development organizations in the world, are collaborating and innovating on a very short timeline to develop the hardware, software, sensors, and human-machine control interfaces that will enable their robots to complete a series of challenge tasks selected by DARPA for their relevance to disaster response.

The DRC Finals will take place from June 5-6, 2015 at Fairplex in Pomona, California. The DRC Finals will require robots to attempt a circuit of consecutive physical tasks, with degraded communications between the robots and their operators; the winning team will receive a \$2 million prize.

Technologies resulting from the DRC will transform the field of robotics and catapult forward development of robots featuring task-level autonomy that can operate in the hazardous, degraded conditions common in disaster zones.