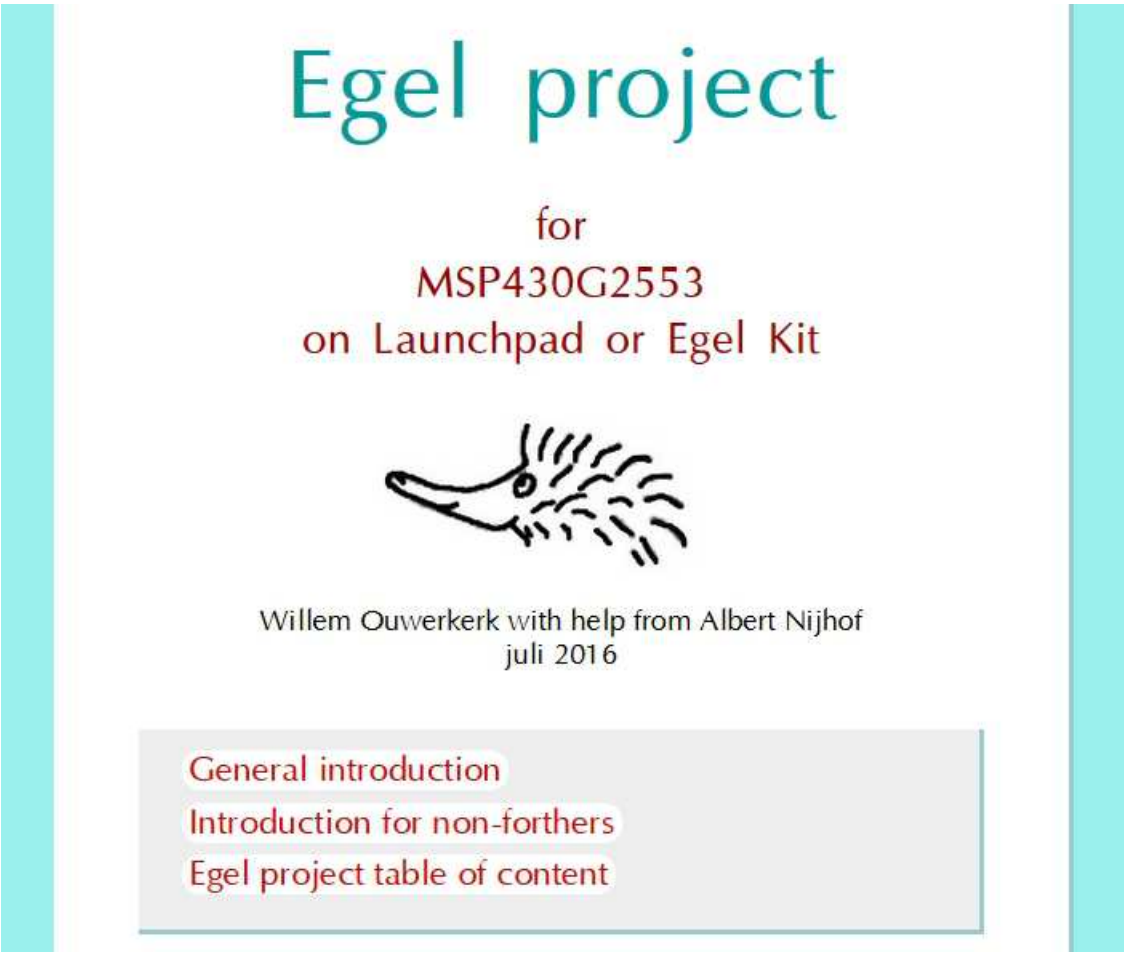
1) Intro biped

Commando's:

?(help)	B(ack)	F(orward	Ll(eft)
R(ight)	H(ello)	S(tamp)	1(rtwist)
2(wiggle)	W(alk autonoom)		N(oForth)
T(wist)	P(osition)	+(faster)	-(slower)

Demo:

H(ello) F(orward) B(ackward
1(twist) 2(wiggle) N(oForth)

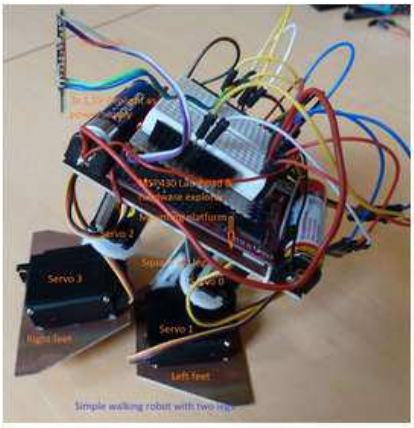


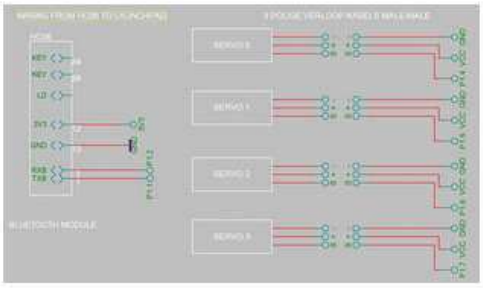
101
Walking biped

Simple walking robot

Materials:
e101 Biped BOM
Info on a standard servo,
and HC06 Bluetooth module.

e101a noForth program
e101b noForth program
Biped building plan as PDF





The small walking robot has two legs, using four servo's. It is wireless controlled thru bluetooth. The code comes from the first eight chapters. The result is a cute walking robot.

The e101a version has dead simple code, in just a few lines the robot really does something.

http://noforth.bitbucket.org/site/egel%20for%20launchpad.html

Het Egel project is afkomstig van de een Forth voor de oude 8051 microcontroller. Daarna vertaald naar de AVR en nu opgefrist en verbeterd voor de MSP430 van Texas Instruments.

Listing Biped E101a software:

```
Hex
04 constant #SRV  ( Four servo outputs )

\ Space for #srv servos PWM values and pause period
create SERVOS  #srv 1+ cells allot

\ I/O-bits for each output, the last cell is 0 output for pause period
\ With this version of the software the maximum is eight servo's
CREATE #BITS  10 c, 20 c, 40 c, 80 c, 0 c, align

: SET-PAUSE     ( -- )
    dm 20000  servos #srv cells bounds
    do i @ -  cell +loop  servos #srv cells + ! ;

\ Set servo position in steps from 0 to 200
: SERVO         ( u +n -- )
    >r  dm 5 * dm 1000 + dm 2000 umin
    r> [ #srv 1- ] literal  umin cells servos + !  set-pause ;

\ This interrupt gives 1 to 2 millisec. pulses at 50 Hz
\ Register R11 (xx) can not be used for something else!!!!
routine PULSES  ( -- )      \ 6 - interrupt call
    day push                \ 3 - Save original r8
    servos # day mov        \ 2 - Load address pointer
    xx day add              \ 1 - Calc. address of next period
    xx day add              \ 1 - One cell!
    day ) 172 & mov         \ 5 - TA0CCR0  Set next period
    #bits # day mov         \ 2 - Load bit-table pointer
    xx day add              \ 1 - Calculate next bit
    day ) 021 & .b bis      \ 5 - P1OUT  Set bit on (P1)
\ The piece that resets previous servo pulse
    #0 xx cmp               \ 1 - Is it the first bit?
    =? if,                  \ 2 - Yes
        #4 day add          \ 1 - Set bit pointer on de pause position
    then,
    #-1 day add             \ 1 - To next bit
    day ) 021 & .b bic      \ 5 - P1OUT  Reset previous bit (P1)
\ To next servo
    #1 xx add               \ 1 - To next servo
    #srv 1+ # xx cmp        \ 2 - Hold pointer in valid range
    =? if, #0 xx mov then,
    rp )+ day mov           \ 3 - Restore originele r8
    reti                    \ 5 -
end-code

code INTERRUPT-ON      ( -- )    #0 xx mov  #8 sr bis  next  end-code
code INTERRUPT-OFF     ( -- )    #8 sr bic  next  end-code

value L/R       \ 0 = rest-position, 1 = right up, -1 = left up
value WAIT      \ Step duration ins MS
```

```
\ Activate 4 servo's at P1,4 etc.
: BIPED-ON      ( -- )
    0F0 022 *bis            \ P1DIR   Bit P1.4 to P1.7 outputs
    0 160 !                 \ TA0CTL  Stop timer-A0
    dm 1000 172 !           \ TA0CCR0 First interrupt after 1 ms
    02D4  160 !             \ TA0CTL  Start timer
    0010 162 !              \ TA0CCTL0  Set compare 0 interrupt on
    #srv 0 do 64 i servo loop \ Default pulse lenght is 1,5 ms
    150 to wait             \ Wait time 340 ms
    interrupt-on ;          \ Activate

: BIPED-OFF     ( -- )
    0 160 !                 \ TA0CTL  Stop timer-A0
    010 162 **bic           \ TA0CCTL0  Interrupts off
    interrupt-off ;

decimal  \ basic biped posture routines
: W             wait ms ;
: REST          #srv 0 do 100 i servo loop  w  0 to l/r ;
: RIGHT-UP      150 1 servo  150 3 servo  w  1 to l/r ;
: LEFT-UP       050 3 servo  050 1 servo  w  -1 to l/r ;
: RIGHT-FORW    060 0 servo  060 2 servo  w ;
: LEFT-FORW     140 2 servo  140 0 servo  w ;
: DOWN          100 1 servo  100 3 servo  w ;
: WAVE          040 3 servo  w  150 3 servo  w ;
: TOES          160 3 servo  040 1 servo  w ;


\ Legs to rest position, real biped movements
: >REST         ( -- )
    l/r 0= if  exit  then
    l/r 0< if  left-up  rest exit  then
    right-up  rest ;

\ Small dance s times
: WOBBLE        ( s -- )
    0 ?do
        right-up  w  left-up  w
    loop  down ;

\ Walk s steps forward
: WALK          ( s -- )
    0 ?do
        right-up  right-forw  down
        left-up   left-forw   down
    loop
    w  >rest ;

\ Say hello to viewers
: HELLO         ( -- )
    toes  w  rest  w  right-up  w
    5 0 ?do  wave  loop  w  rest ;

hex  pulses  FFF2 vec!  freeze  \ Install pulses in Timer-A0 vector
```

2) Waarom Forth

Voorbeeld:

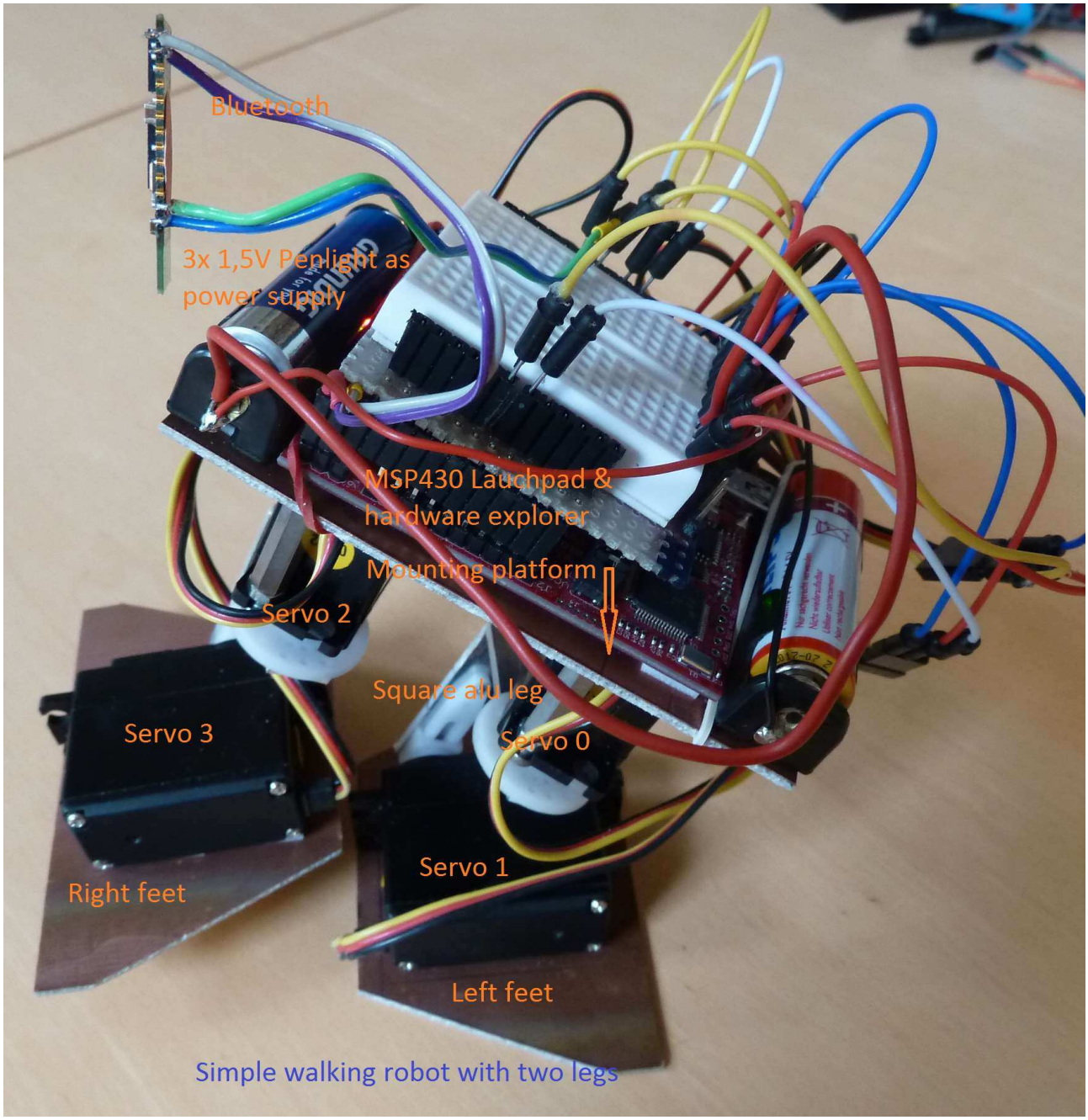
```
    RIGHT-UP  RIGHT-FORW  DOWN


: RIGHT-UP  ( -- )
    150 1 servo  150 3 servo  w  1 to l/r ;

: RIGHT-FORW    ( -- )
    060 0 servo  060 2 servo  w ;

: DOWN        ( -- )
    100 1 servo  100 3 servo  w ;
```

Benodigde code:

```
noforth asm.f
e101a – walking biped robot-1.f
```

```
: WALK          ( s -- )
    0 ?do
        right-up  right-forw  down
        left-up   left-forw   down
    loop
    w  >rest ;
```

3)Biped en Hexapod vergeleken



De eenvoudige biped balanceert op een been.

Biped:	Hexapod:
e110 - autonomous walking biped.f	– noforth-asm.f
	– rs232 usb.f
Zonder assembler, maar wel:	– i2c-24c64a.f
– servo motor interrupt	– 2x10 servo interrupt 1a.f
– piliplop6c	– random6b.f
– US distancemeter	– piliplop6c.f
– geluidjes	– pootjes5b.f
– lopen en andere bewegingen	– ext-pootjes.f
– autonome voortbeweging	– servotester1.f
– één toets afstand besturing	

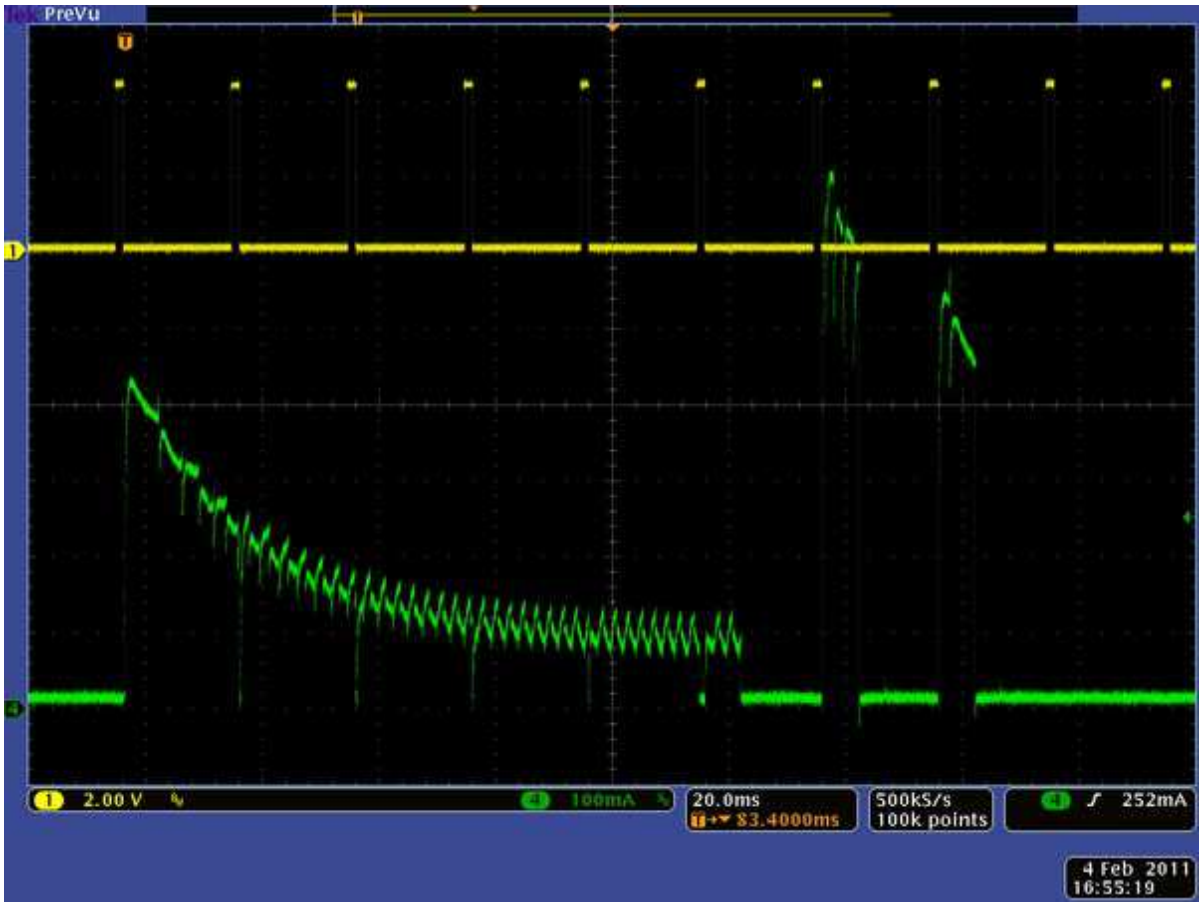
Motor limieten:

```
\ Gemeten limieten voor elke MG90 servo!

ecreate #begin
  029E e,  029E e,          \ Kop
  0271 e,  02DA e,  0320 e, \ Poot4 = 1
  029E e,  028A e,  02A8 e, \ Poot5 = 2
  028A e,  028A e,  029E e, \ Poot6 = 3
  02D0 e,  02EE e,  0276 e, \ Poot1 = 4
  028A e,  02E9 e,  02EE e, \ Poot2 = 5
  02E4 e,  02BC e,  02BC e, \ Poot3 = 6

ecreate #end
  0988 e,  0988 e,          \ Kop
  08E8 e,  09E2 e,  0988 e, \ Poot4 = 1
  0924 e,  091F e,  0988 e, \ Poot5 = 2
  0942 e,  0910 e,  08FC e, \ Poot6 = 3
  0988 e,  0988 e,  0924 e, \ Poot1 = 4
  092E e,  09CE e,  09F6 e, \ Poot2 = 5
  09A6 e,  094C e,  0988 e, \ Poot3 = 6
hex
```

Aansturing per poort of groep poten:

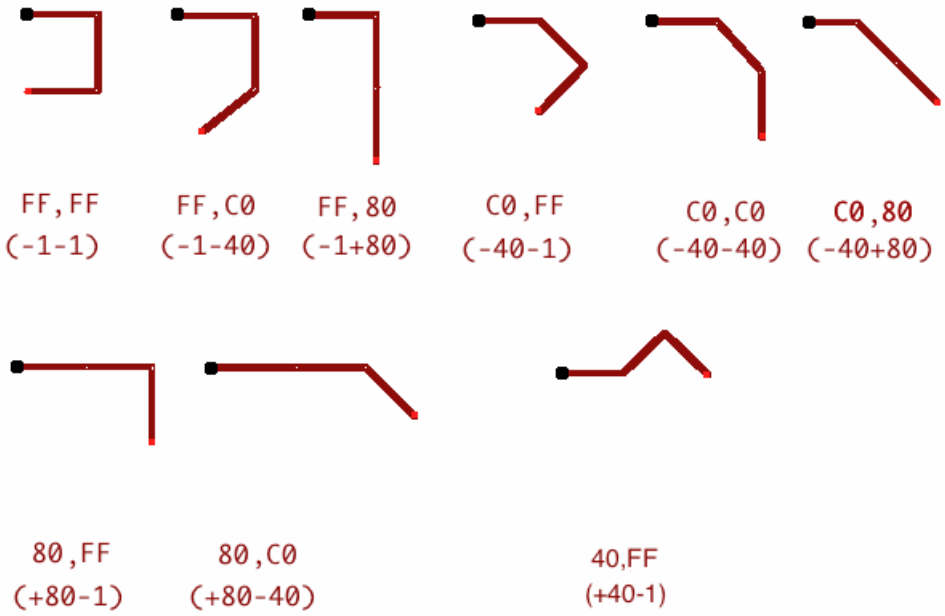
```
ecreate NORM      00 ec, 60 ec, D0 ec,

: LEG1   ( hor. schouder elleboog -- )
    02 (JOINT) 03 (JOINT) 0 hor 04 (JOINT) ;

: RLEGS  ( houding -- )
    dup @leg leg6  dup @leg leg4  @leg leg2 ;

: >ALL   ( houding -- )
    dup llegs >rlegs ;

: START  ( -- )
    0 to pos  norm  >all ; \ Poten in basis positie
```

Piekstroom:

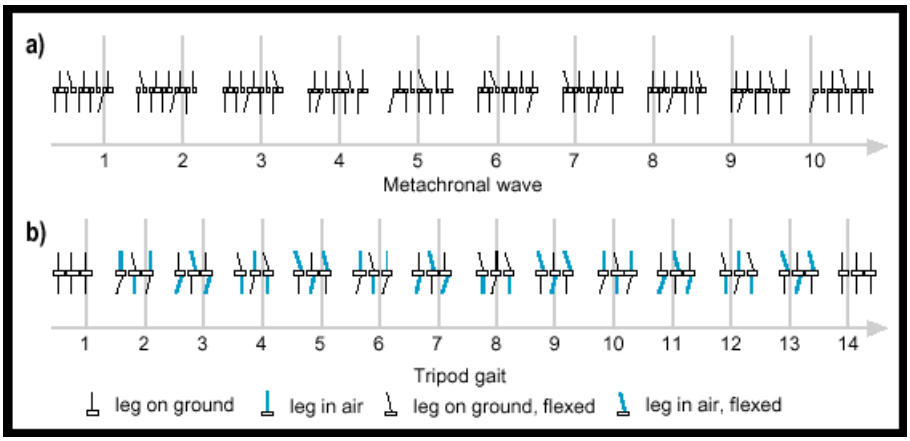


4) Uitwerken loopjes

Logisch nadenken:

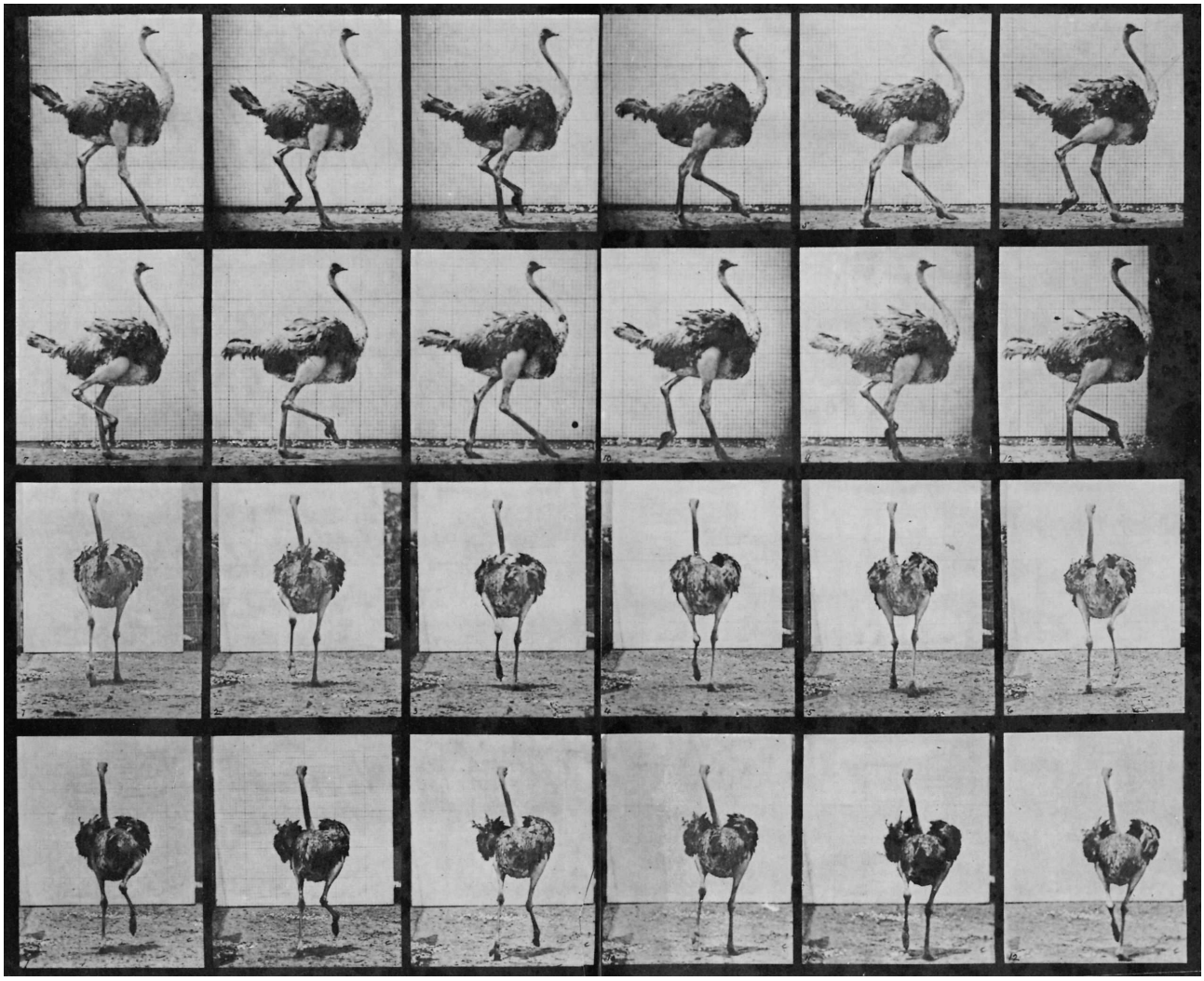


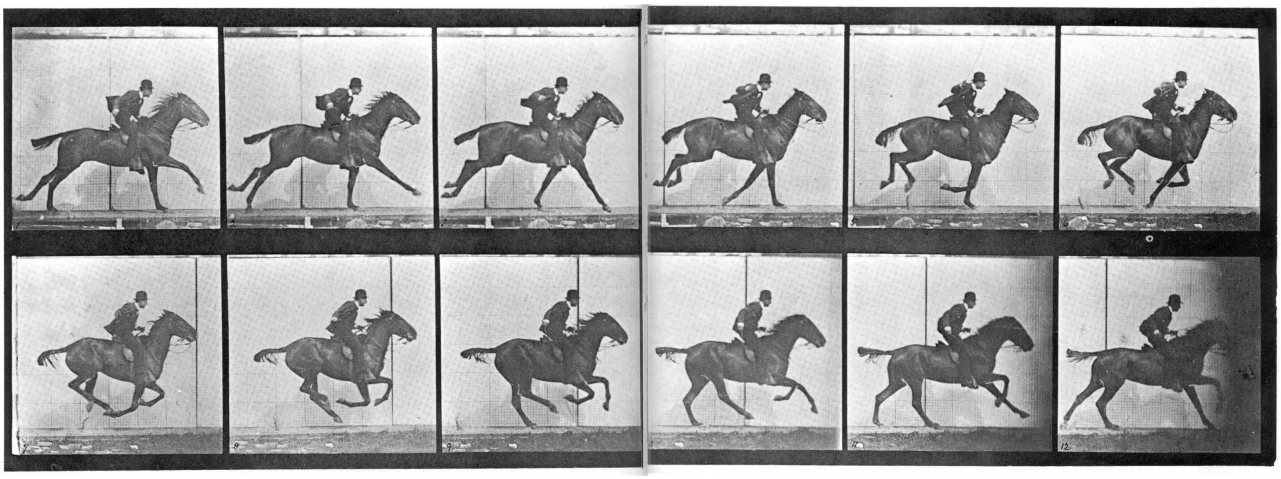
Onderzoek op internet:



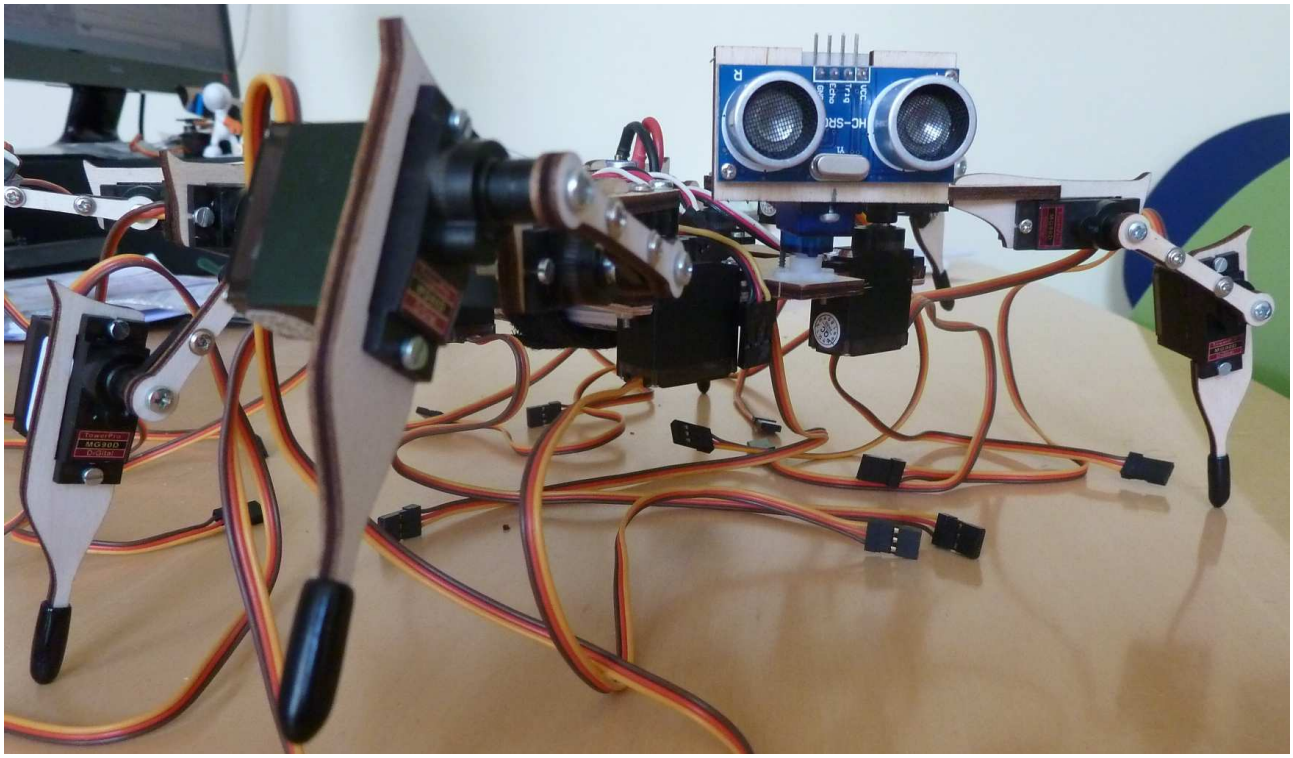
http://cronodon.com/BioTech/Insect_locomotion.html

Foto studies van Eadweard Muybridge:





Experimenten:



```
p6 ------- p1    Poot posities
   |       |
   |       |
p5 |       | p2
   |       |
   |       |
p4 ------- p3
```

Poot in basis positie:

```
ecreate NORM      00 ec, 60 ec, D0 ec,
```

Poot omhoog:

```
ecreate UP        00 ec, A8 ec, FF ec,
```

5) Materialen en constructie

Hexapod kit kopen:

Motoren:

Find a servo: Google™ Custom Search | Search | Advanced Search | Your comparison engine (0)

Servo Database › TowerPro Servos › MG90

TowerPro MG90 - Micro Servo

Basic Information

Modulation:	Analog
Torque:	**4.8V:** 30.6 oz-in (2.20 kg-cm) **6.0V:** 34.7 oz-in (2.50 kg-cm)
Speed:	**4.8V:** 0.11 sec/60° **6.0V:** 0.10 sec/60°
Weight:	0.49 oz (14.0 g)
Dimensions:	Length: 0.91 in (23.1 mm) Width: 0.48 in (12.2 mm) Height: 1.14 in (29.0 mm)
Motor Type:	? (add)
Gear Type:	Metal
Rotation/Support:	Dual Bearings

Brand:	Tower pro
Product Number:	? (add)
Suggested Retail:	? (add)
Street Price:	9.69 USD
Compare:	add

Additional Specifications

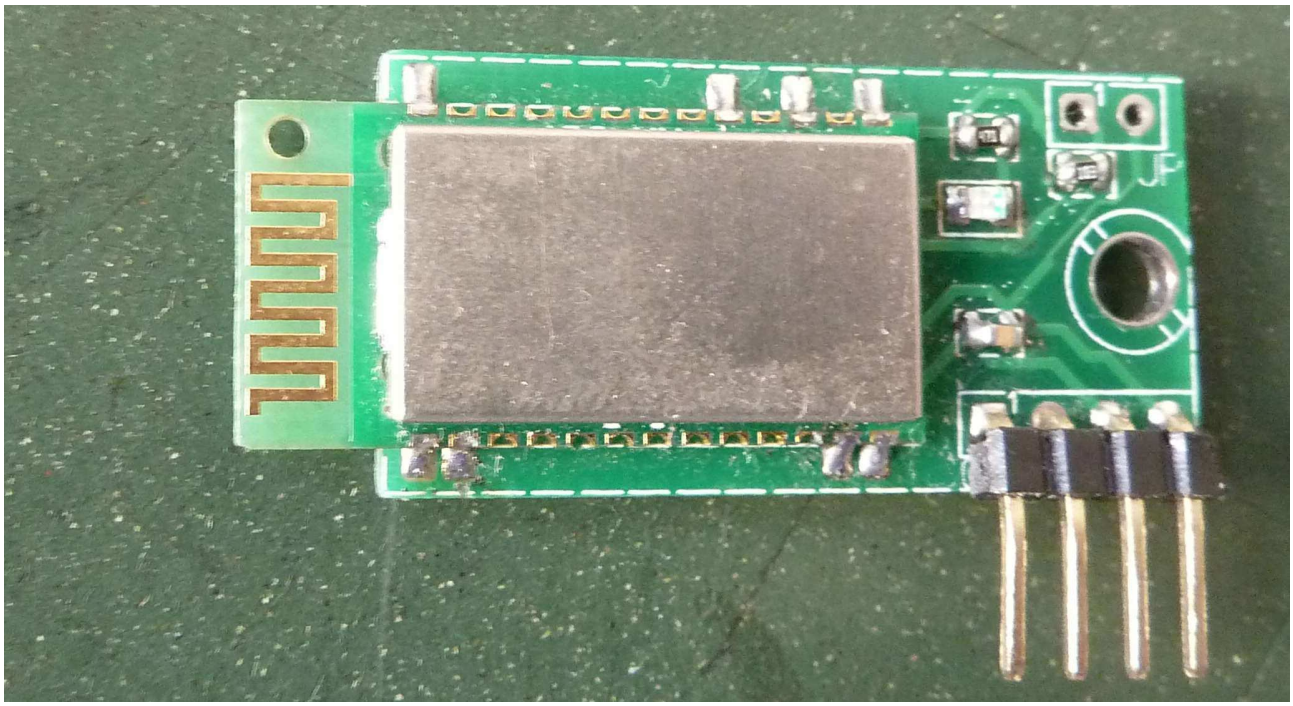
Rotational Range:	180°
Pulse Cycle:	20 ms
Pulse Width:	400-2400 µs
Connector Type:	? (add)

User Reviews

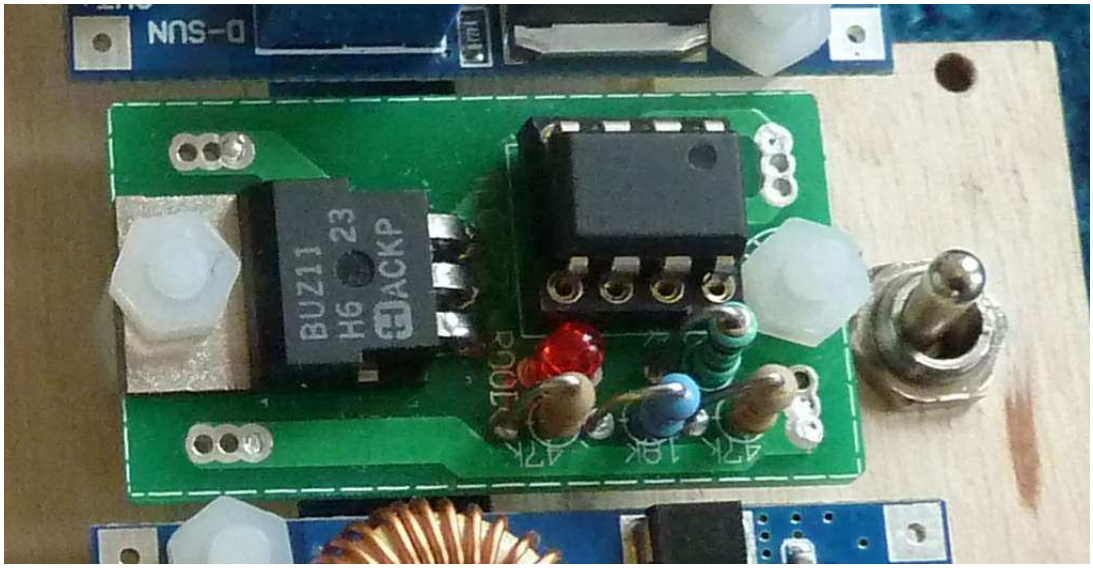
Number of Reviews:	12
Average Rating:	3.3 / 5.0

Hobbyking Rc
Online Store

HC-06 Bluetooth communicatie module:

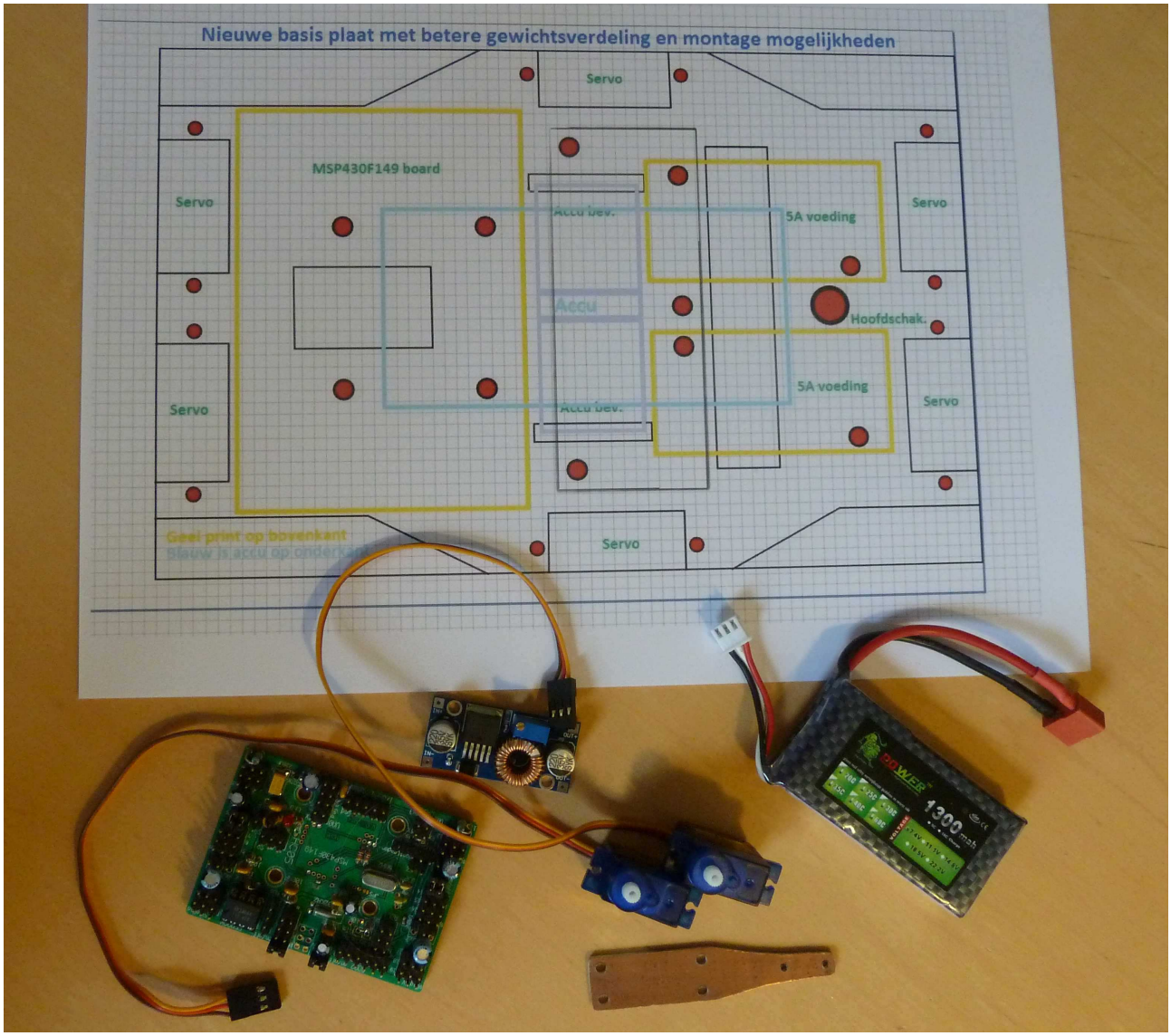


LiPo leegloop beveiliging:



Eigen ontwerp:

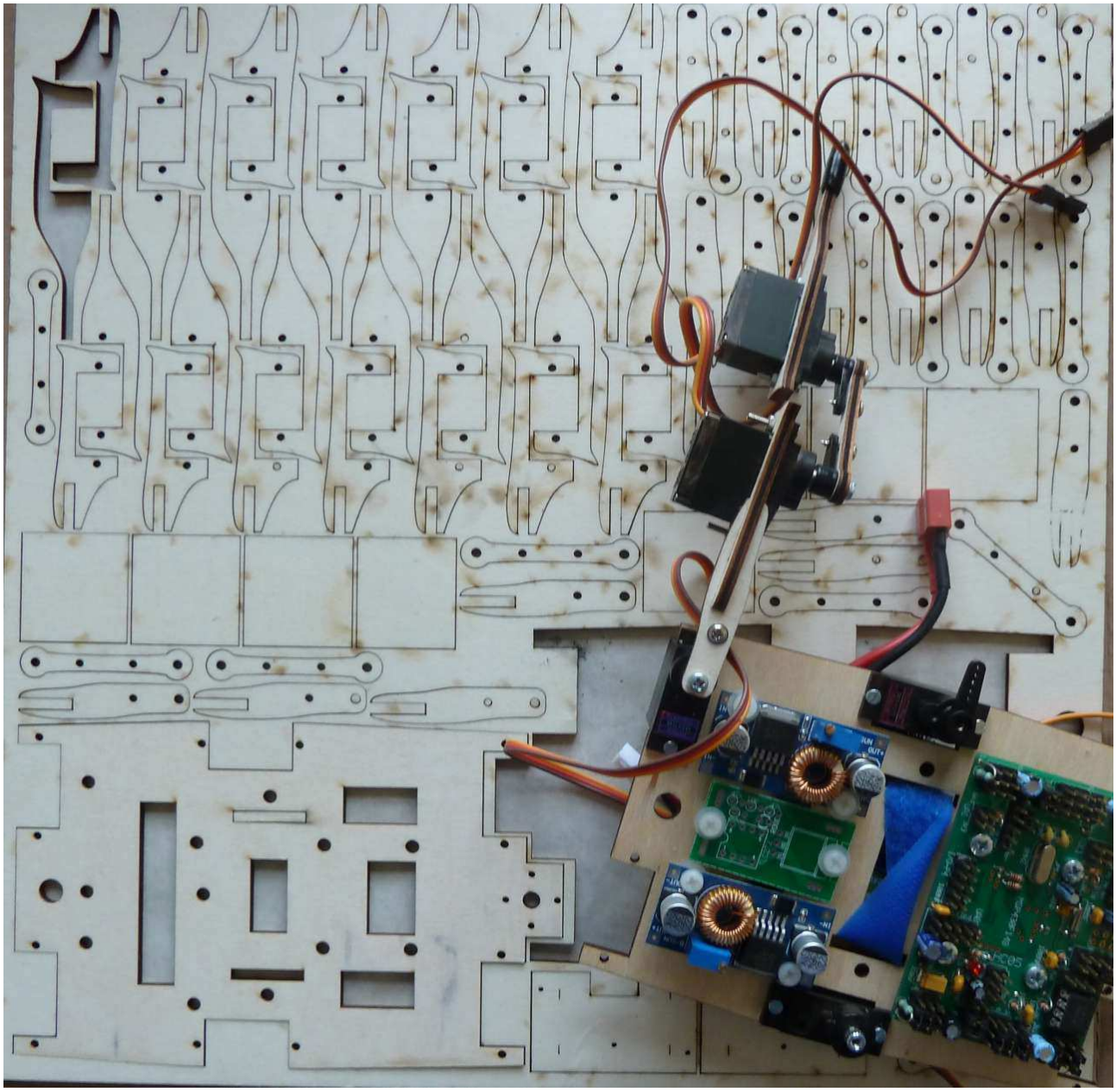
Schets:



Er is vooral aandacht besteed aan:

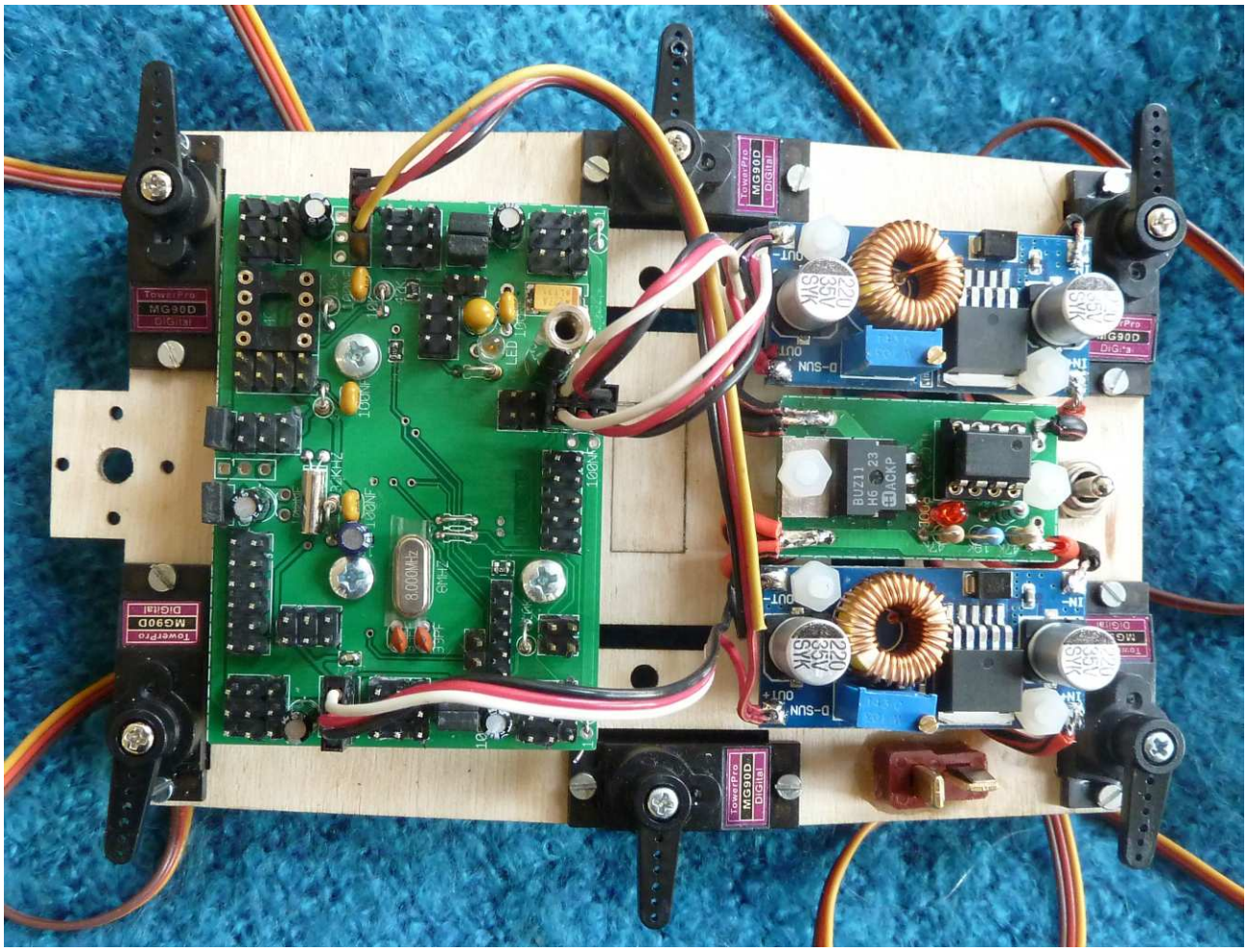
- Een betere gewichts verdeling.
- Efficiënte plaatsing van de componenten
- Voldoende sterkte en stijfheid

Laser gesneden triplex of plexiglas:



Een pootje:

De romp:



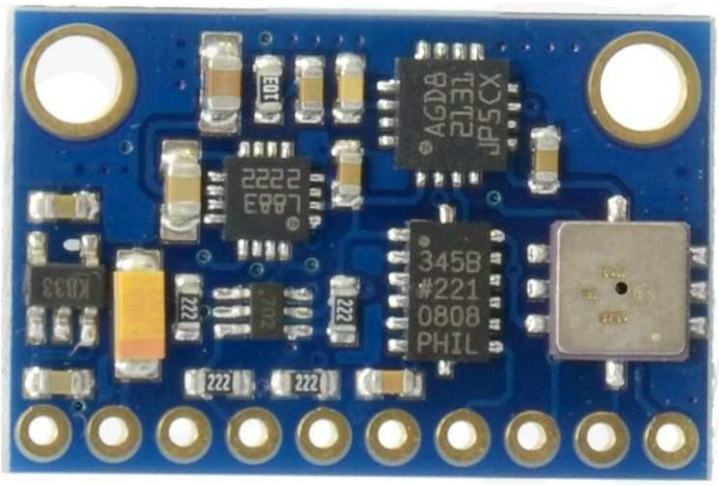
6) Sensoren

Object detectie:

Tast:

- Voelsprieten
- Druk op pootjes

Evenwicht en coördinatie:



- Versnelling
- Gyroscoop
- Kompas
- Druk

Interne toestand:

- ADC van microcontroller voor accustatus en temperatuur

7) Toepassingen

Experimenten:

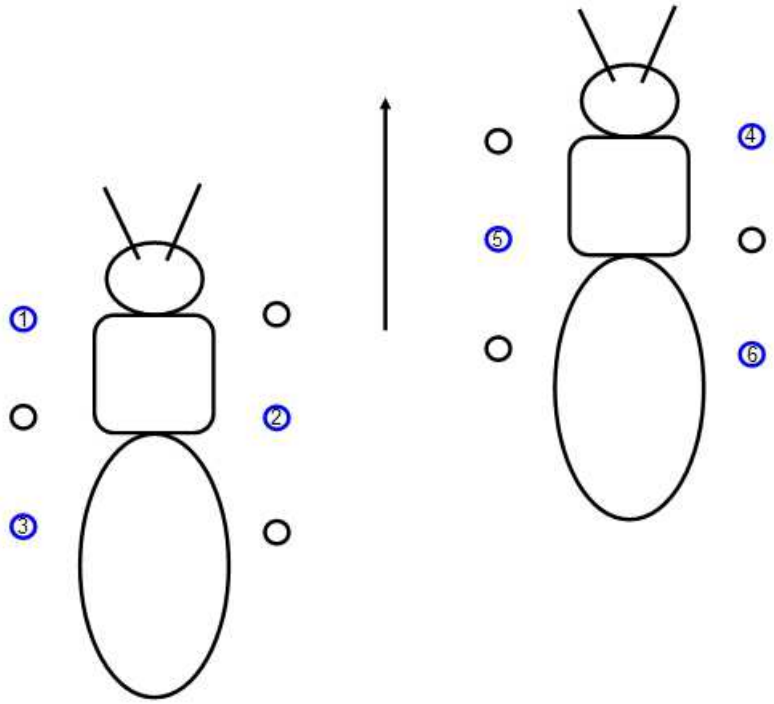
- Software implementaties:
  - Absolute bewegingspatronen
  - Relatieve bewegingspatronen
  - Terugkoppeling

- Bewegingspatronen

- Gedrag

- Sensoren en software integratie daarvan

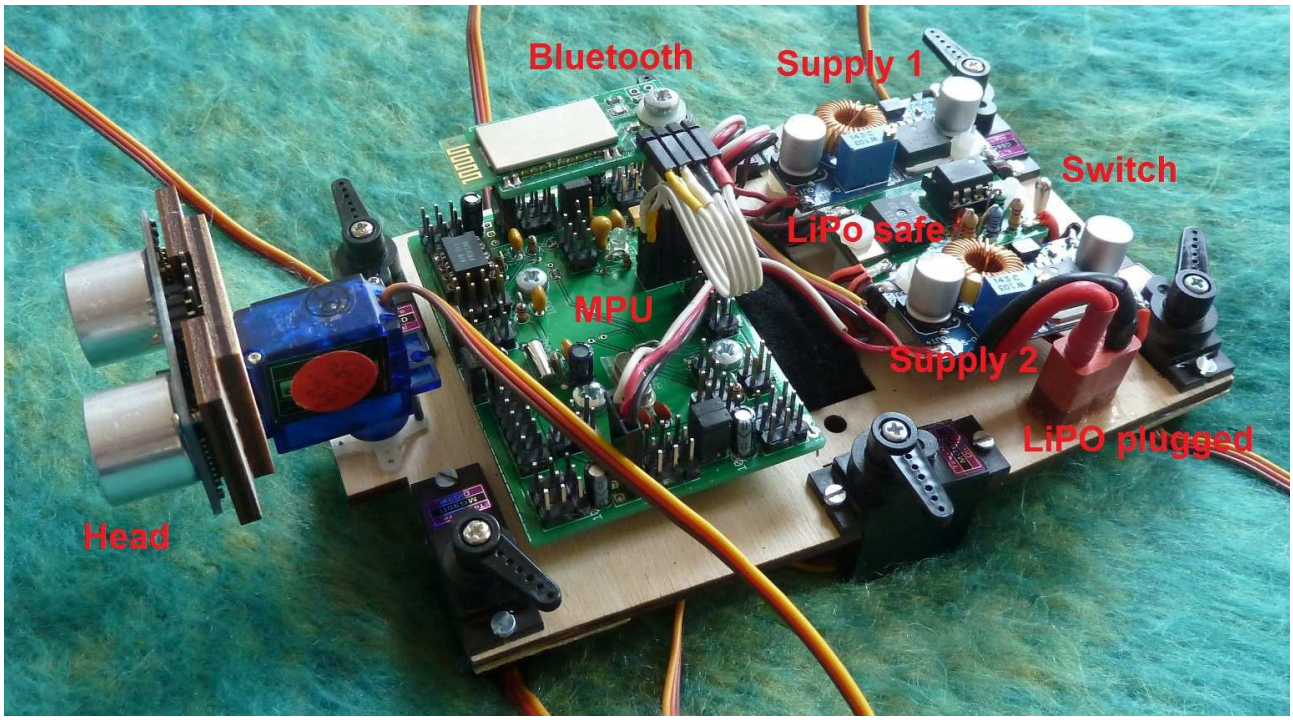
- Voortbewegen op niet vlakke ondergrond

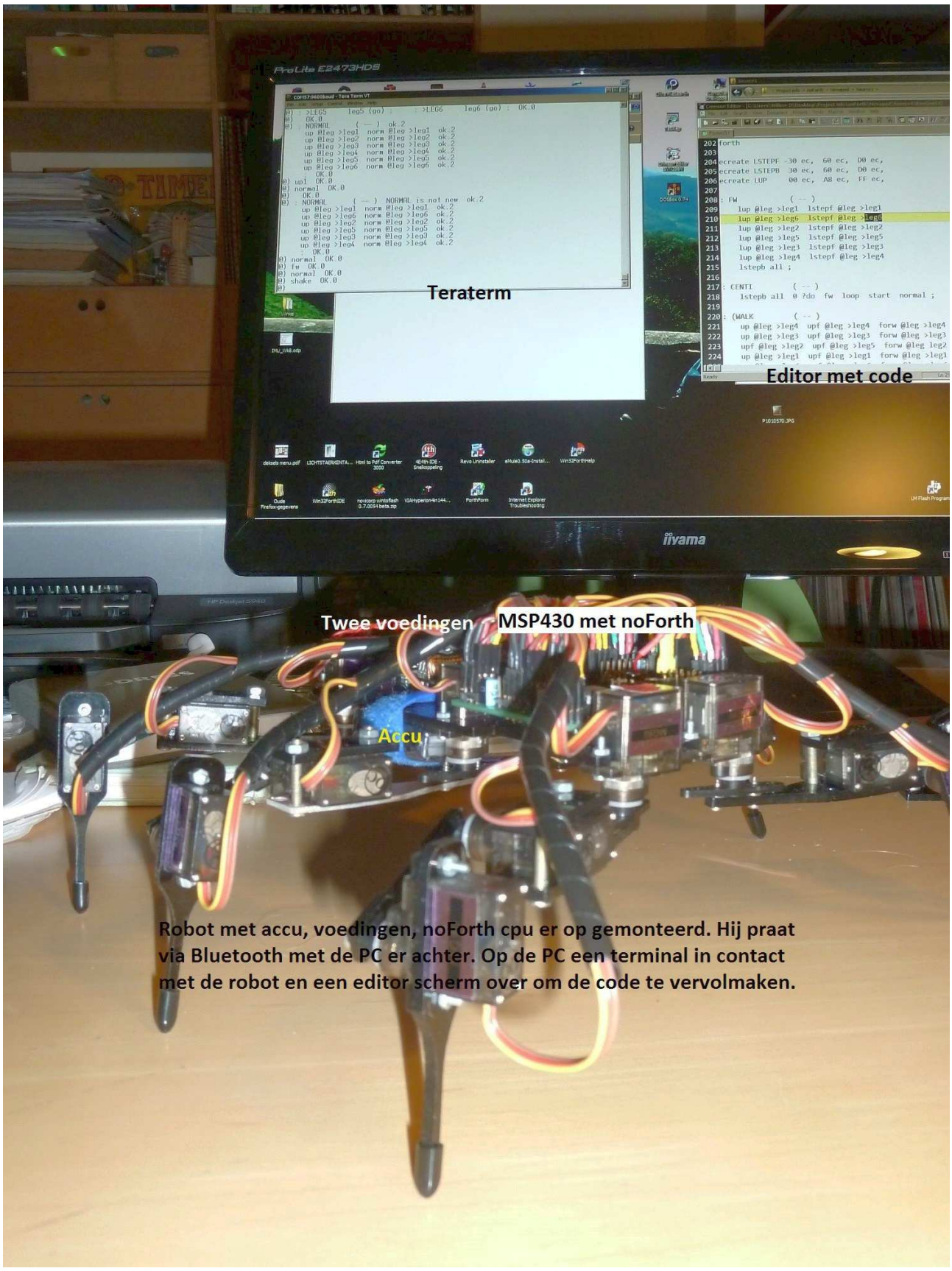


8) Demo hexapod

Onderdelen:

a) Hexapod aanzetten en contact maken
b) `READY` - De robot start in fasen op en staat op
c) `WALK/BACKW` - Loop 'u' stappen voor en achteruit
d) `LTURN/RTURN` - Draai 'u' stappen links en rechts
e) `.SPEED` - Toon ingestelde snelheid
f) `10 SPEED 5 WALK` - Wandel via Piliplop met stap vertraging 10
g) `-40 SPEED 5 WALK` - Wandel zonder Piliplop met vertraging 40
h) `RCRAB/LCRAB` - Crab loopjes 'u' stappen
i) `ANT` - Simuleer een mierenloopje
j) `SLEEP/WAKEUP` - Slapen als een kat
k) `REST2` - In ruststand 2





Robot met accu, voedingen, noForth cpu er op gemonteerd. Hij praat via Bluetooth met de PC er achter. Op de PC een terminal in contact met de robot en een editor scherm over om de code te vervolmaken.

Mieren simulatie:

```
\ ANT simulatie routine, vertaald naar hexapod

value CHANCE    \ Random range
value ANGLE     \ Maximum angle
value STEPS     \ Forward steps

\ Bereik van de scratch variant is: chance - angle to chance + angle
\ Dat is in het geval van chance = 15 en angle = 1 van -14 tot 16
\ Een grotere neiging naar rechts dan naar links
\ Onderstaande variant is geheel in evenwicht:
\ -chance - angle tot chance + angle is -16 tot 16

: GET-ANGLE     ( -- n )
\   chance 2* choose  chance angle -  - 2/ ;  \ Kies hoek, scratch var.
    chance angle + 2* 1+ choose \ Kies hoek
    chance angle +  -           \ Bepaal draairichting
    2/ 2/ ;                     \ Een kwart is genoeg voor de hexapod

: SENS?         ( -- )      10 ms  01 01C bit* 0= ; \ Voelspriet?
: AVOID         ( -- )      sens? if  even  2 backw  3 rturn  then ;

: FORW          ( s -- )         \ Doe S stappen
    0 ?do
        avoid  1 walk  ch . emit \ Stap met ontsnap
    loop ;

: TURN          ( -- )
    get-angle  dup .  ?dup 0=   \ Nieuwe hoek, hoek = 0 ?
    if  even 1 forw  exit  then \ Ja, gewoon rechtdoor en klaar!
    dup 0 >                     \ Hoek positief?
    if  right else left  then   \ Ja: rechtsaf, Nee: linksaf
    abs 0 ?do                   \ Neem 1 of meer stappen links of rechts
        avoid  1 walk           \ Stap met ontsnap
    loop ;

\ ANT simulatie variant  Voorbeeld:  1 8 15 ant
: ANT           ( step angle chance -- )
    FE 01E c!  0 01D c!         \ Initialiseer input
    setup-random  even  up1     \ Hexapod staat op
    to chance to angle to steps \ Zet hulpvariabelen
    begin
        turn  even steps forw   \ Simuleer mierenloopje
    key? until  rest2 ;         \ Klaar, ga rusten

\ End
```