# 1) Intro biped

**Commando's:**

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

**Demo:**

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

# Egel project

for
MSP430G2553
on Launchpad or Egel Kit

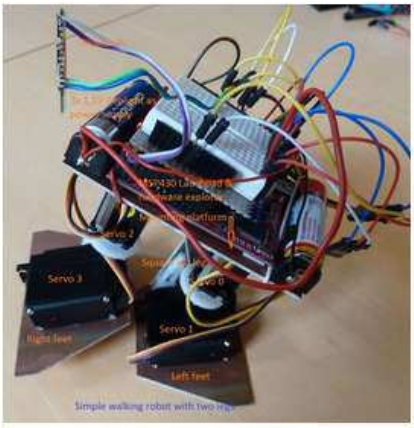
Willem Ouwerkerk with help from Albert Nijhof
juli 2016

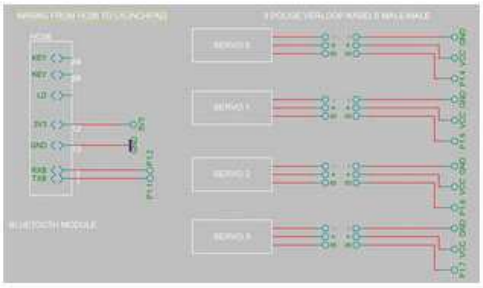
# 101
# Walking biped

**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

The original Egel werkboek was written in 1997 for the 8051 microcontrollers by a group of Dutch Forth-gg members. Later it was translated to the AVR, and now refreshed and improved for the MSP430 from 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 routine in Timer-A0 vector
```

# 2) Why Forth

Example:

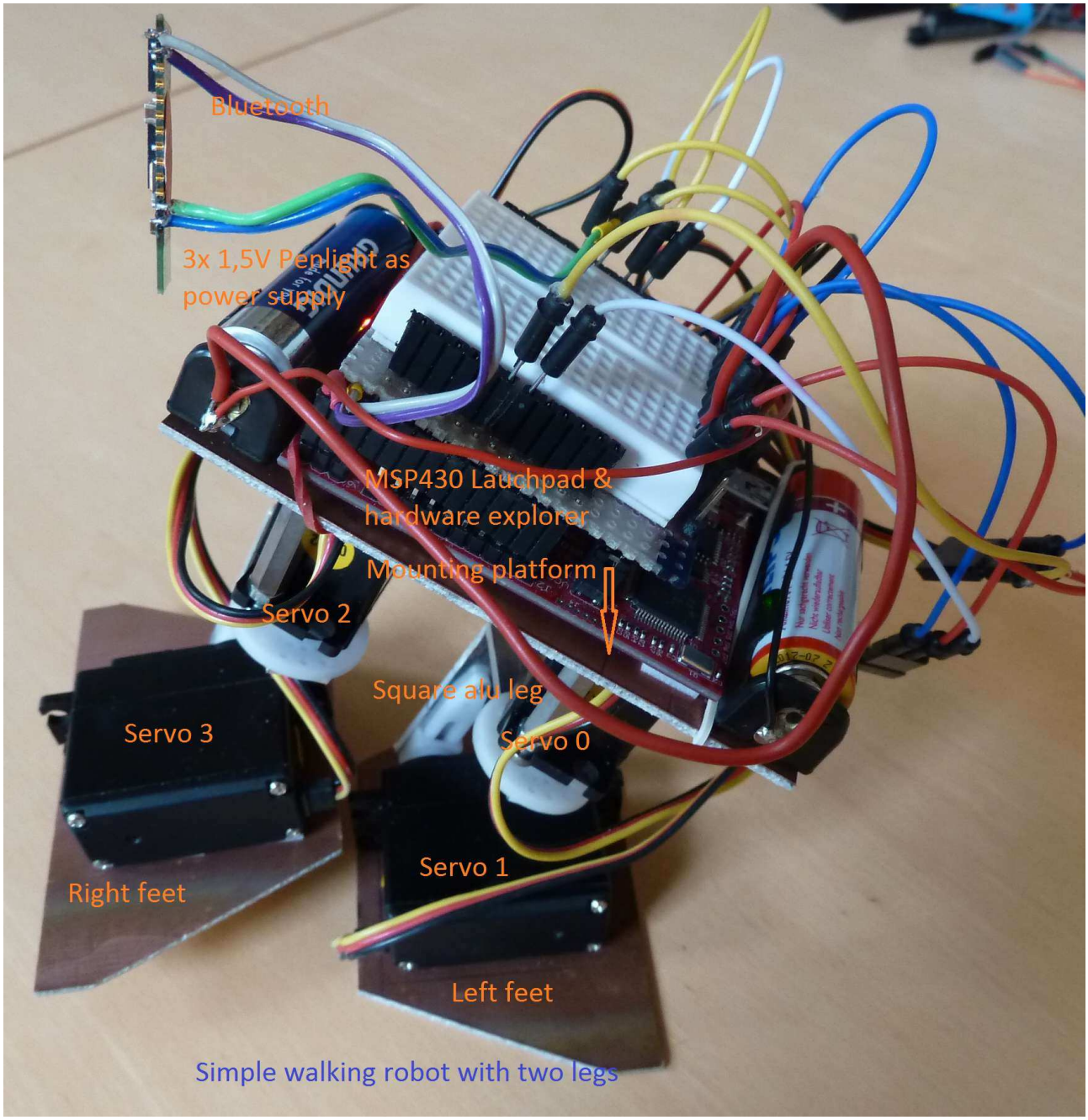
```
    RIGHT-UP  RIGHT-FORW  DOWN


: 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 ;
```

Used 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 & Hexapod comparision




Simple biped balancing on one leg.

| Biped: | Hexapod: |
|---|---|
| e110 - autonomous walking biped.f | – noforth-asm.f<br>– rs232 usb.f |
| Without assembler, but with:<br>– RC-servo motor interrupt<br>– piliplop6c<br>– US distance meter<br>– Sounds<br>– Walking and other movements<br>– Autonomous locomotion<br>– Single key remote control | – i2c-24c64a.f<br>– 2x10 servo interrupt 1a.f<br>– random6b.f<br>– piliplop6c.f<br>– Legs5b.f<br>– ext-Legs.f<br>– servotester1.f |

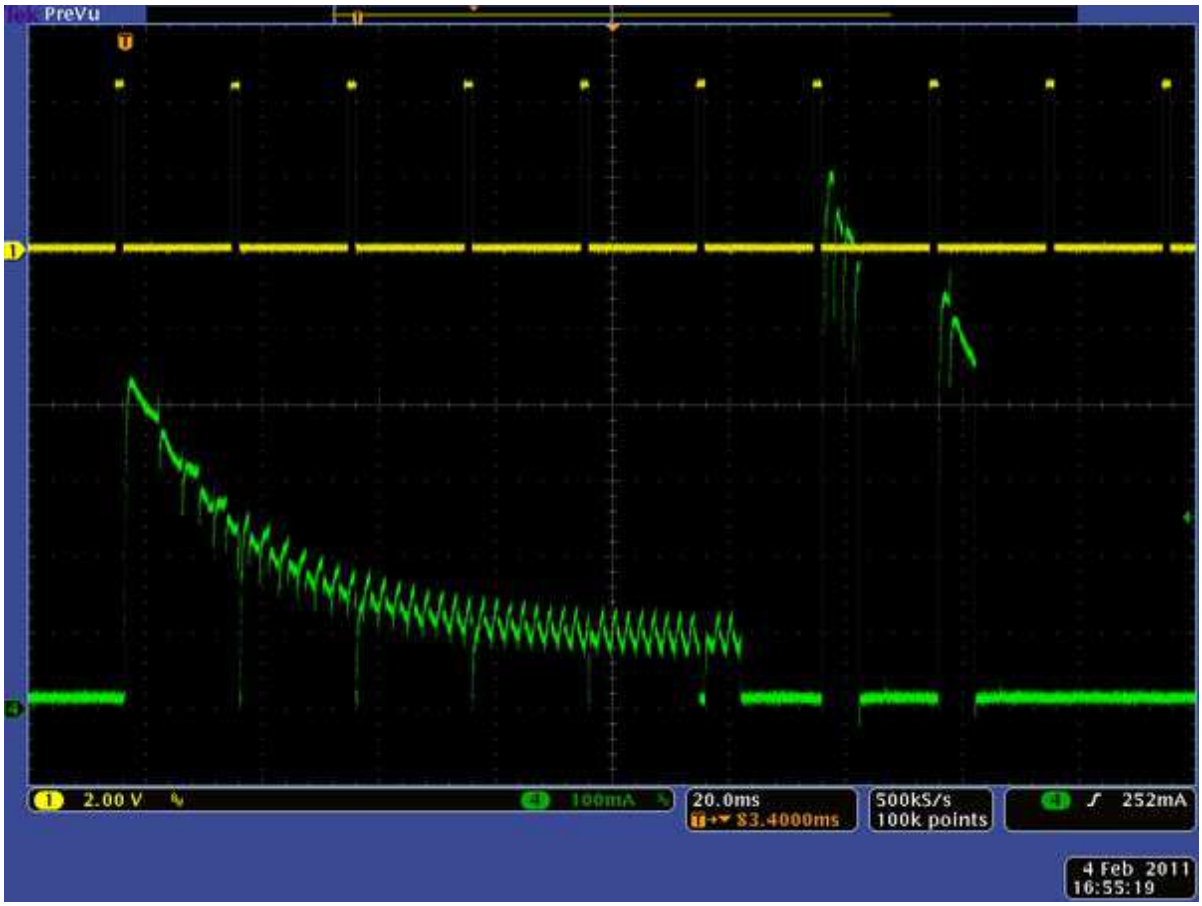
Motor limits:

```
\ Measured limits for each MG90 servo!

ecreate #BEGIN
  029E e,  029E e,          \ Head
  0271 e,  02DA e,  0320 e, \ Leg-4 = 1
  029E e,  028A e,  02A8 e, \ Leg-5 = 2
  028A e,  028A e,  029E e, \ Leg-6 = 3
  02D0 e,  02EE e,  0276 e, \ Leg-1 = 4
  028A e,  02E9 e,  02EE e, \ Leg-2 = 5
  02E4 e,  02BC e,  02BC e, \ Leg-3 = 6

ecreate #END
  0988 e,  0988 e,          \ Kop
  08E8 e,  09E2 e,  0988 e, \ Leg-4 = 1
  0924 e,  091F e,  0988 e, \ Leg-5 = 2
  0942 e,  0910 e,  08FC e, \ Leg-6 = 3
  0988 e,  0988 e,  0924 e, \ Leg-1 = 4
  092E e,  09CE e,  09F6 e, \ Leg-2 = 5
  09A6 e,  094C e,  0988 e, \ Leg-3 = 6
hex
```

Controlling each leg or group of legs:

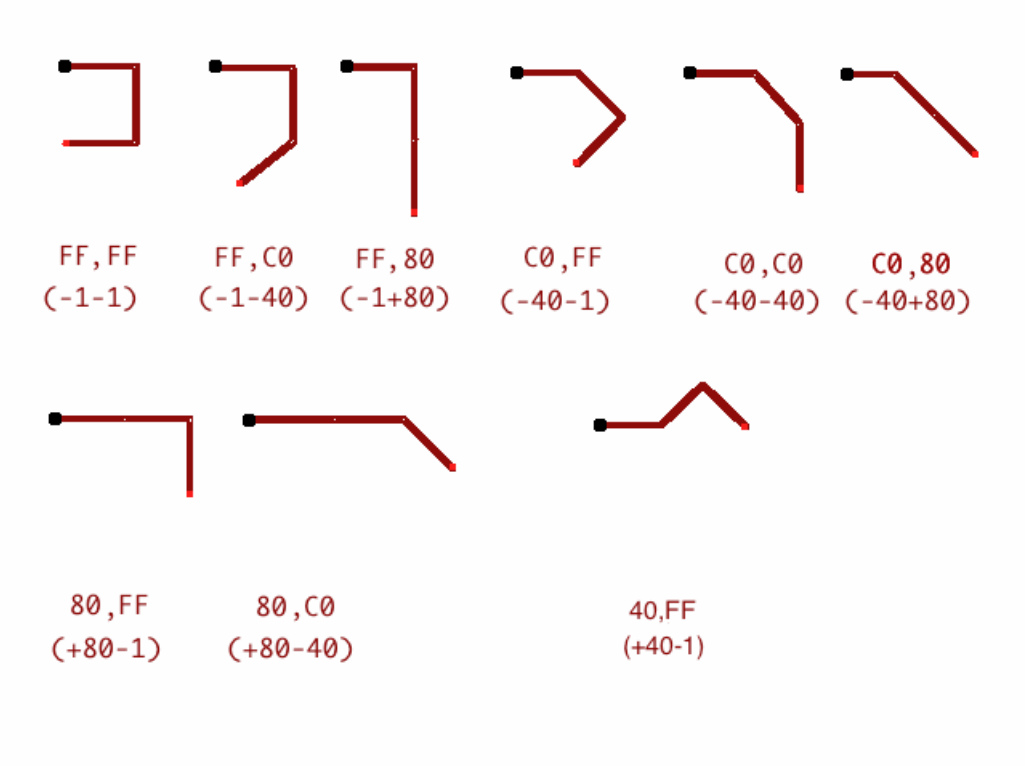
```
ecreate NORM      00 ec, 60 ec, D0 ec,

: LEG1    ( hor. shoulder elbow -- )
     02 (JOINT) 03 (JOINT) 0 hor 04 (JOINT) ;

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

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

: START   ( -- )
     0 to pos   norm   >all ; \ Legs in the basic position
```

Electrical surge:

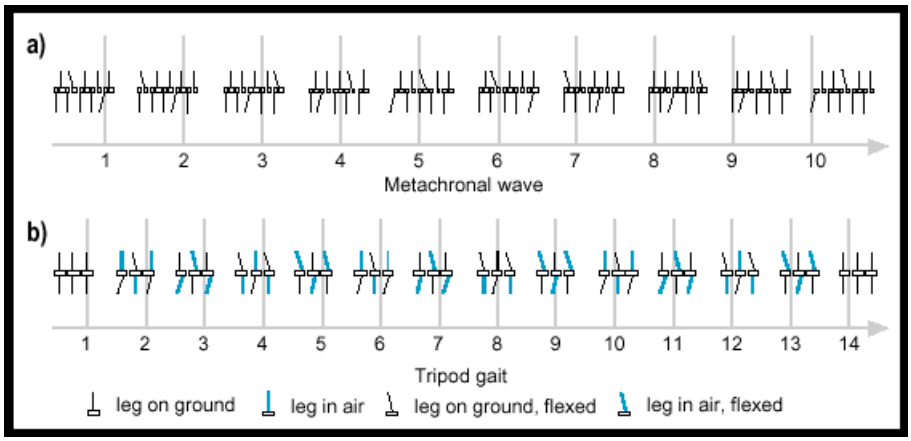


# 4) Develop methods of locomotion

Think logically:

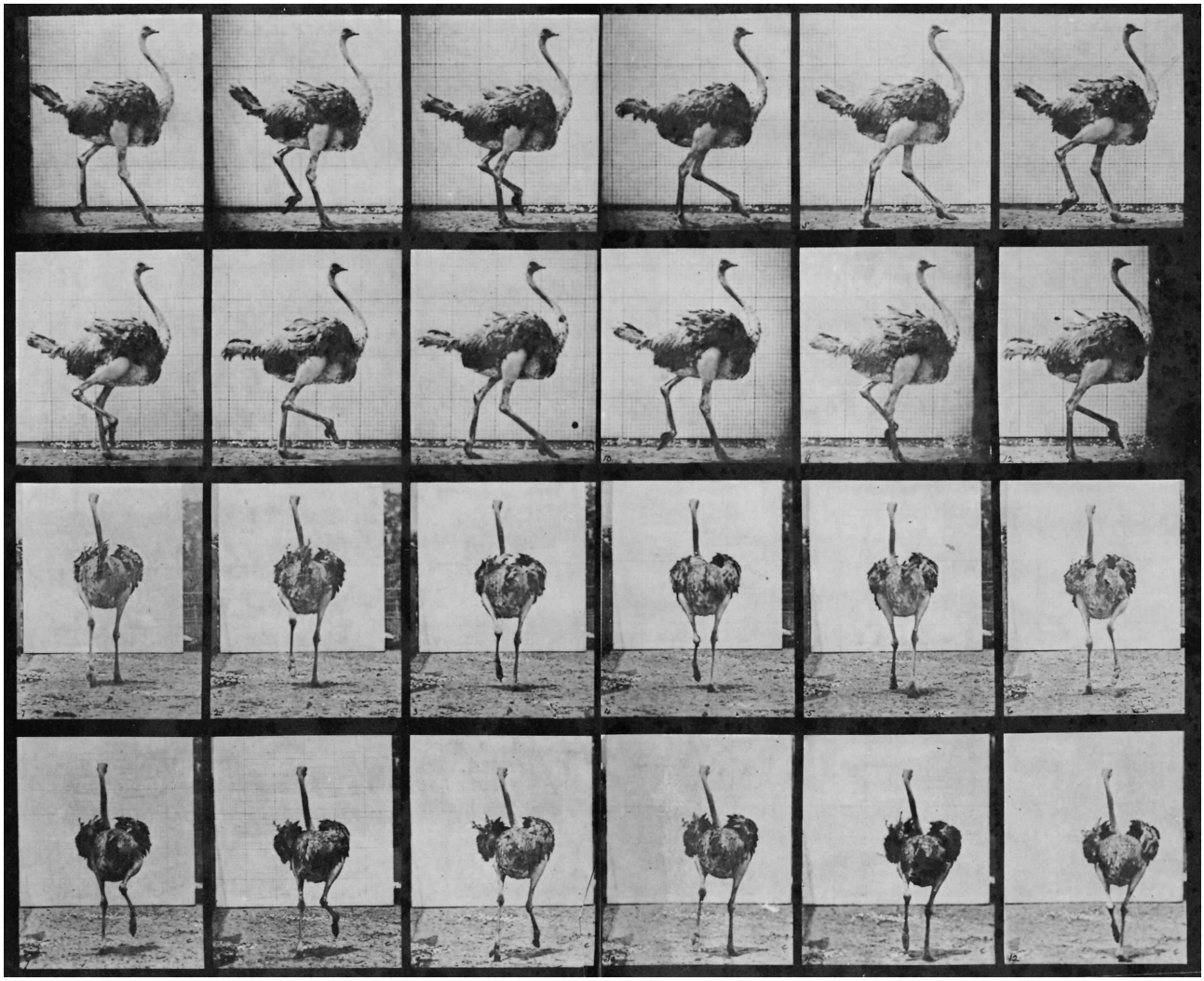


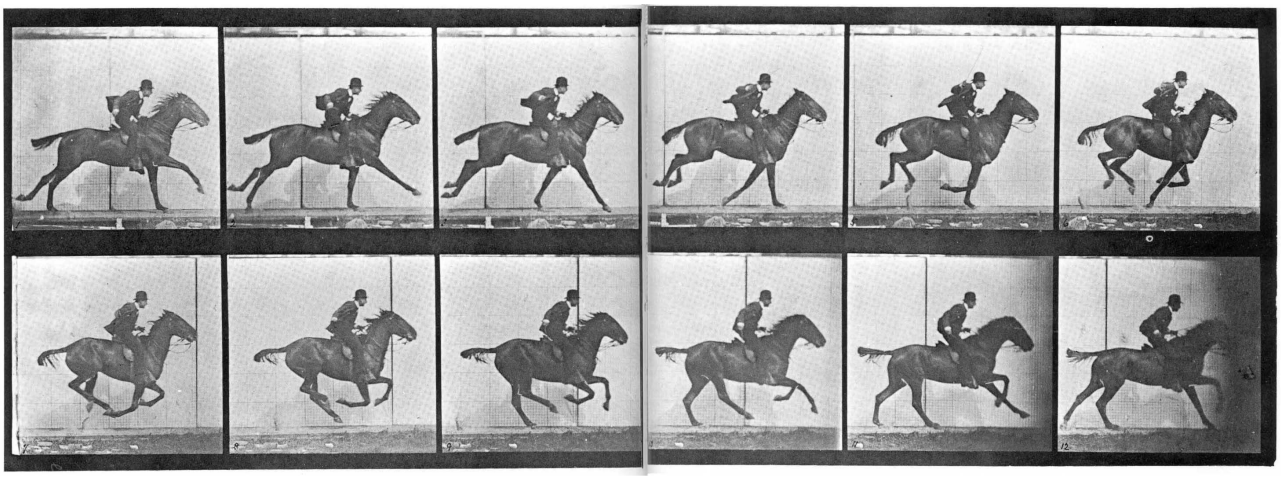

Research on the internet:



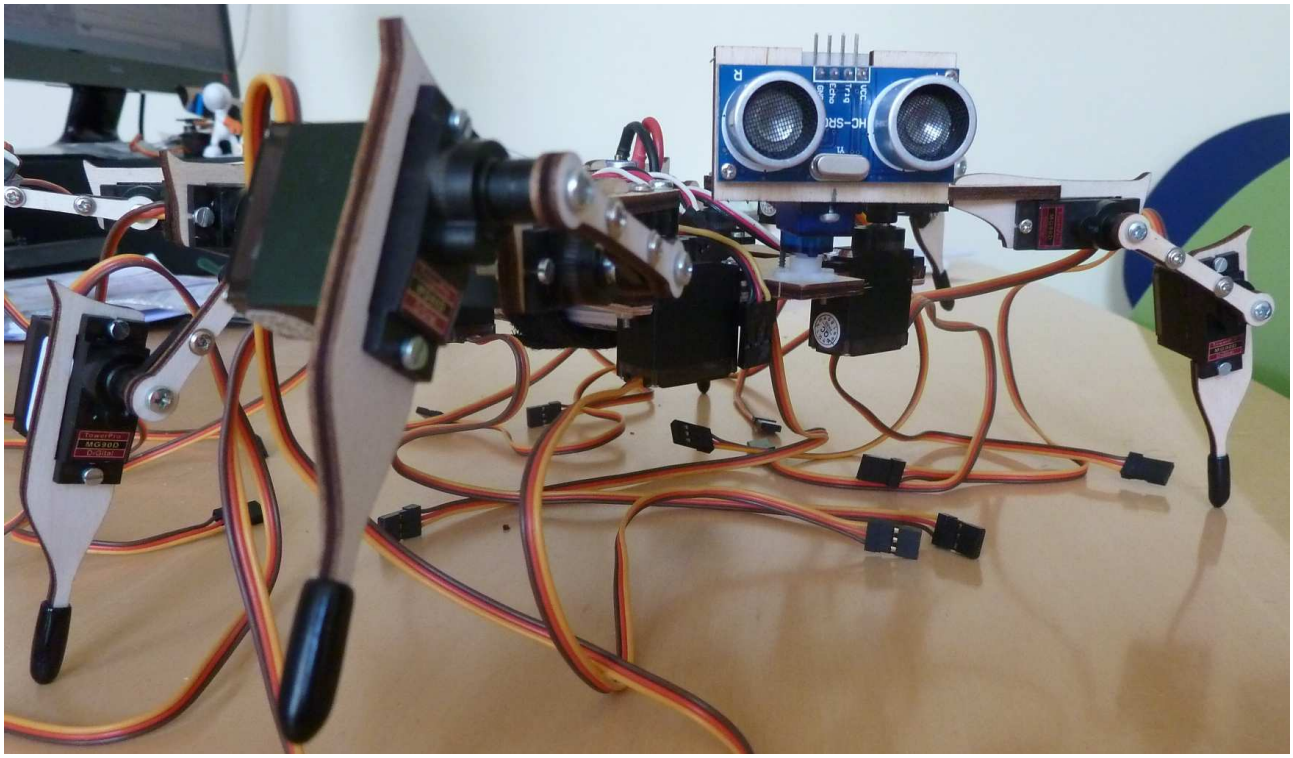

**http://cronodon.com/BioTech/Insect_locomotion.html**

Photo studies from Eadweard Muybridge:





Experiments:



```
16 ------- 11    Leg - positions
   |       |
   |       |
15 |       | 12
   |       |
   |       |
14 ------- 13
```

Leg to basic position:

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

Leg - up:

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

# 5) Materials and construction

## Buy a hexapod kit:

Motors:

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

**Additional Specifications**

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

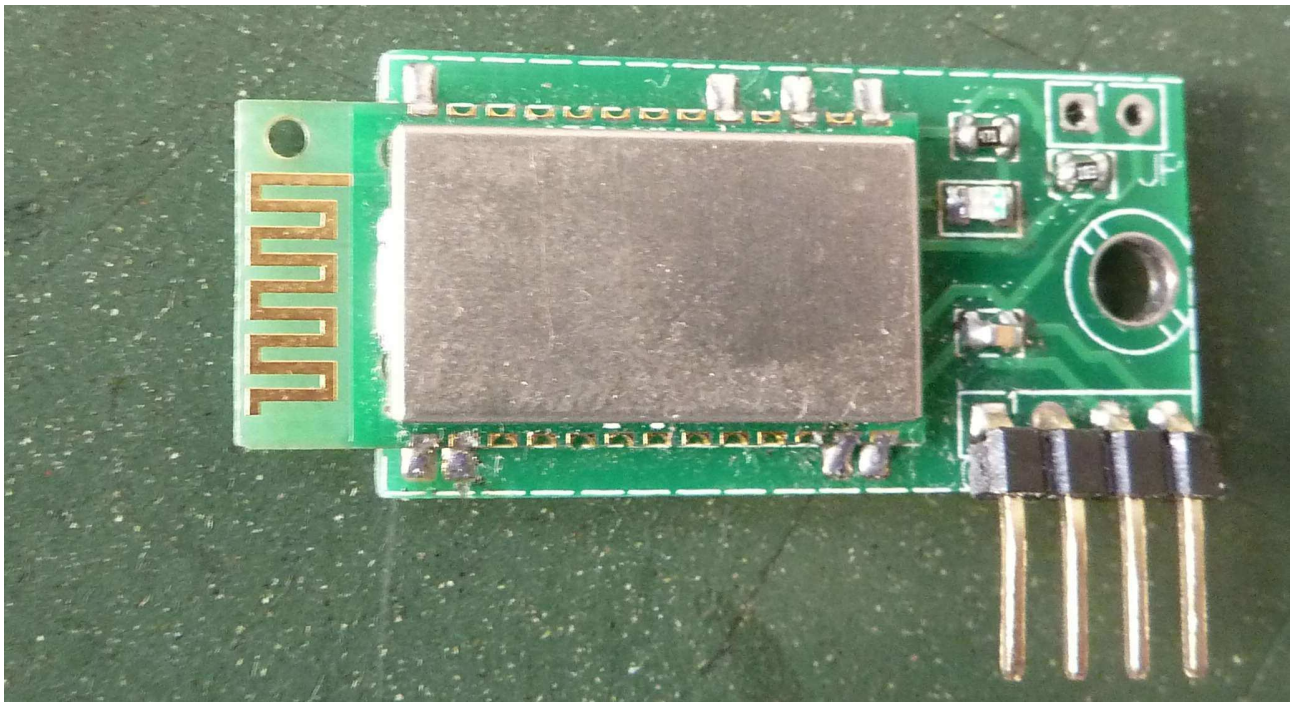



| | |
|---|---|
| Brand: | Tower pro |
| Product Number: | ? (add) |
| Suggested Retail: | ? (add) |
| Street Price: | 9.69 USD |
| Compare: | add |

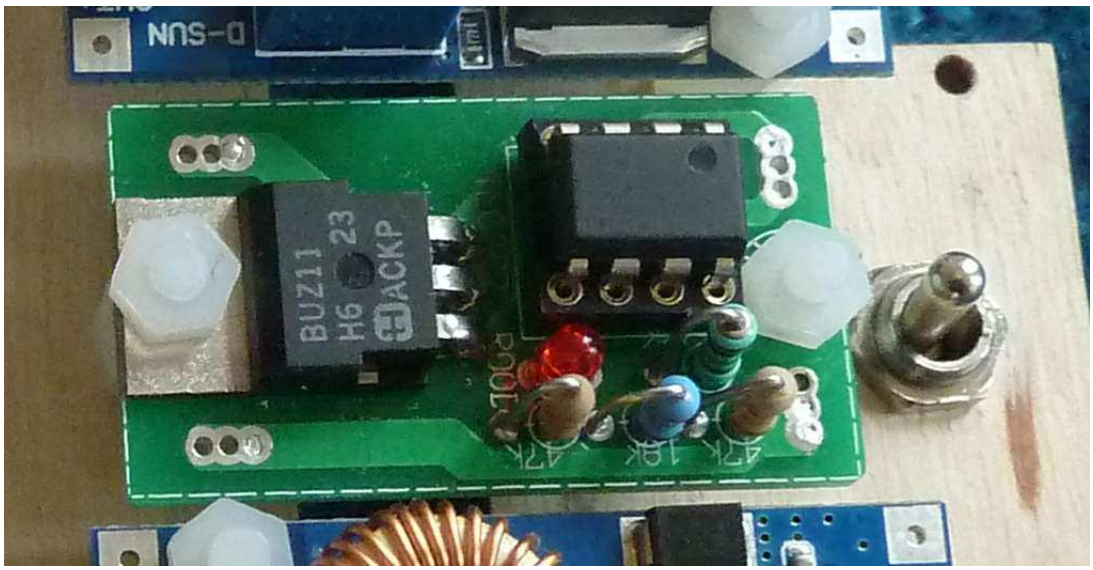
**User Reviews**

| | |
|---|---|
| Number of Reviews: | 12 |
| Average Rating: | 3.3 / 5.0 |

HC-06 Bluetooth communication module:

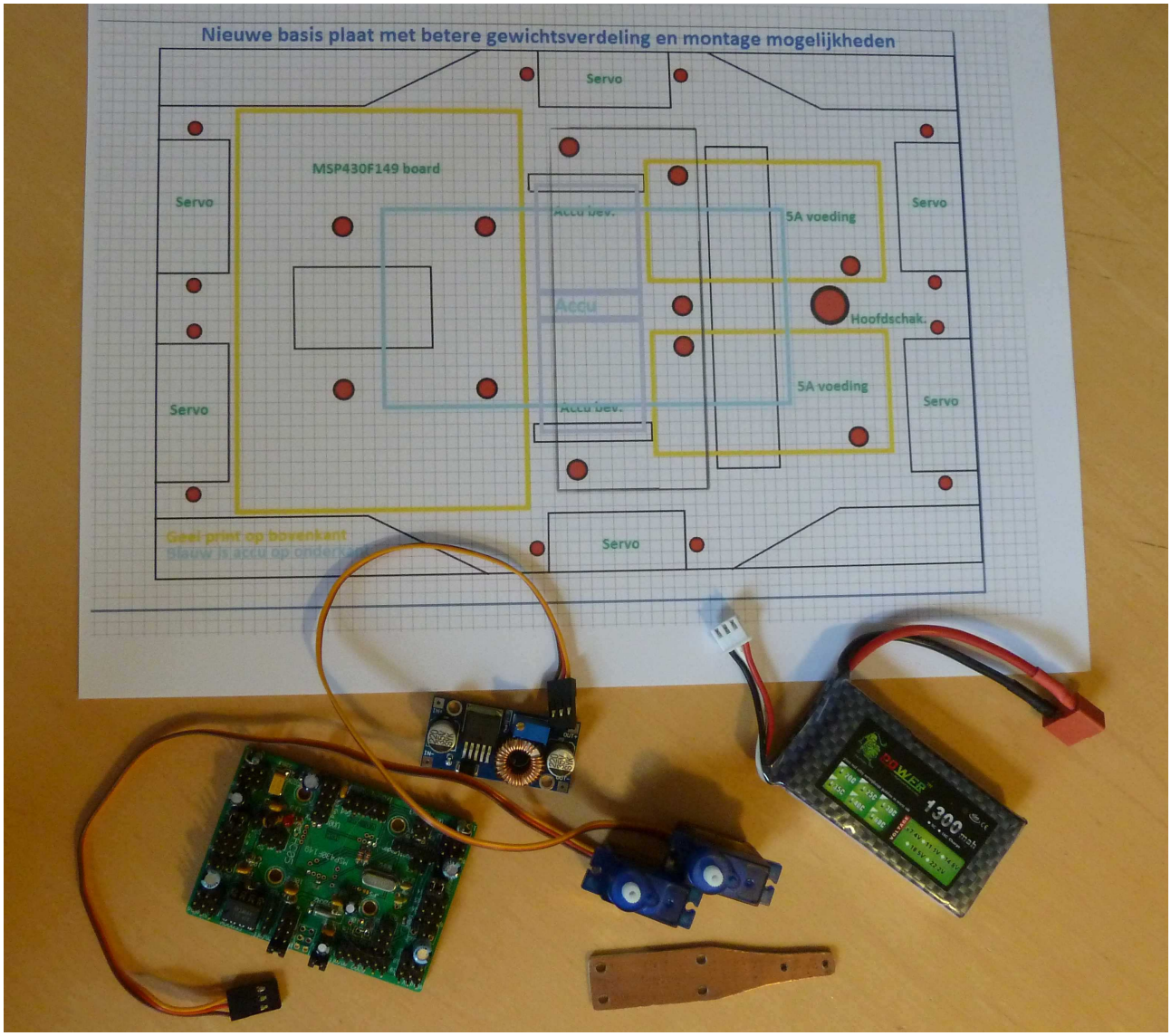


LiPo battery discharge protection:



**Own design:**

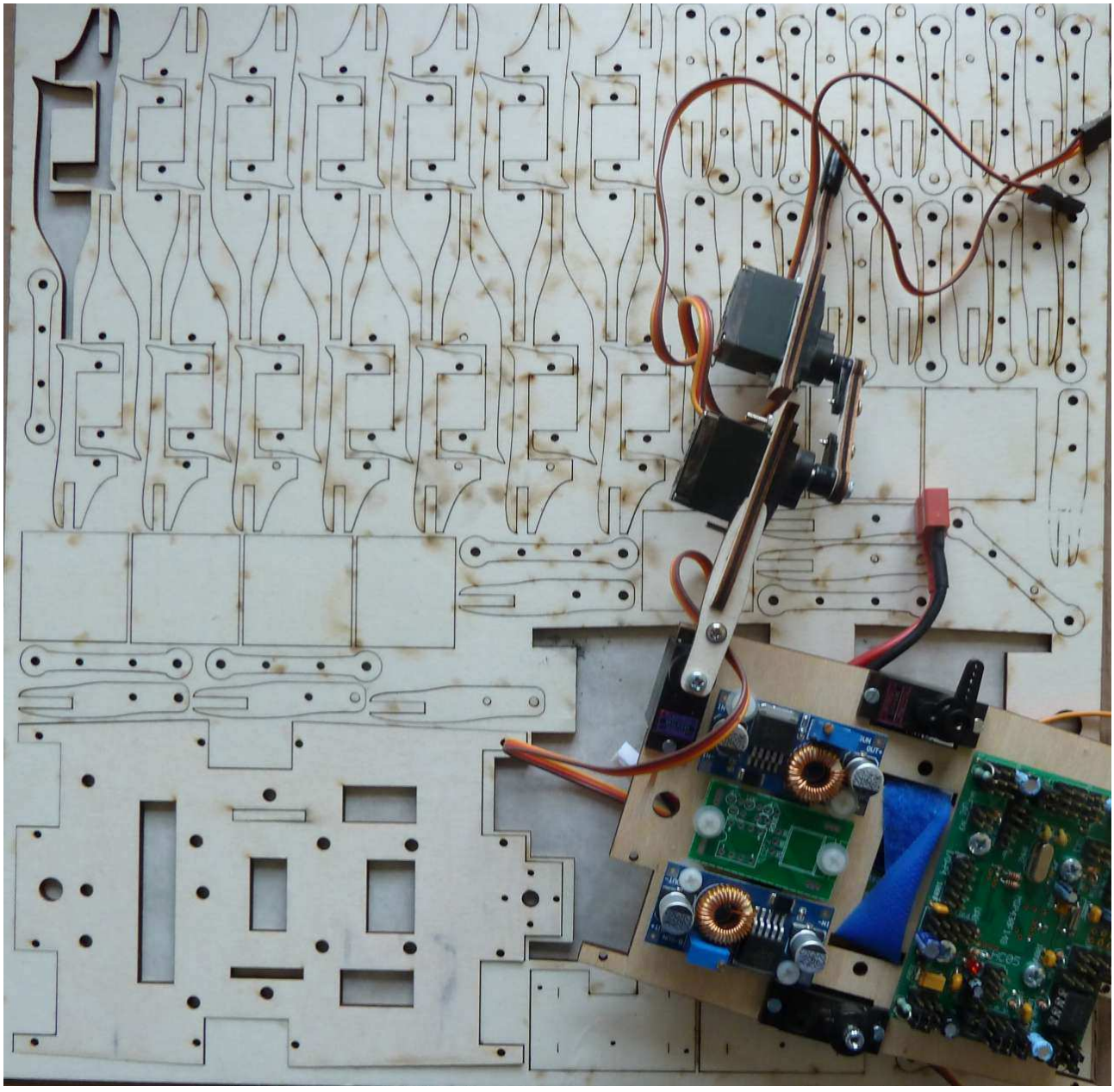
Sketch:




Attention has been paid to:

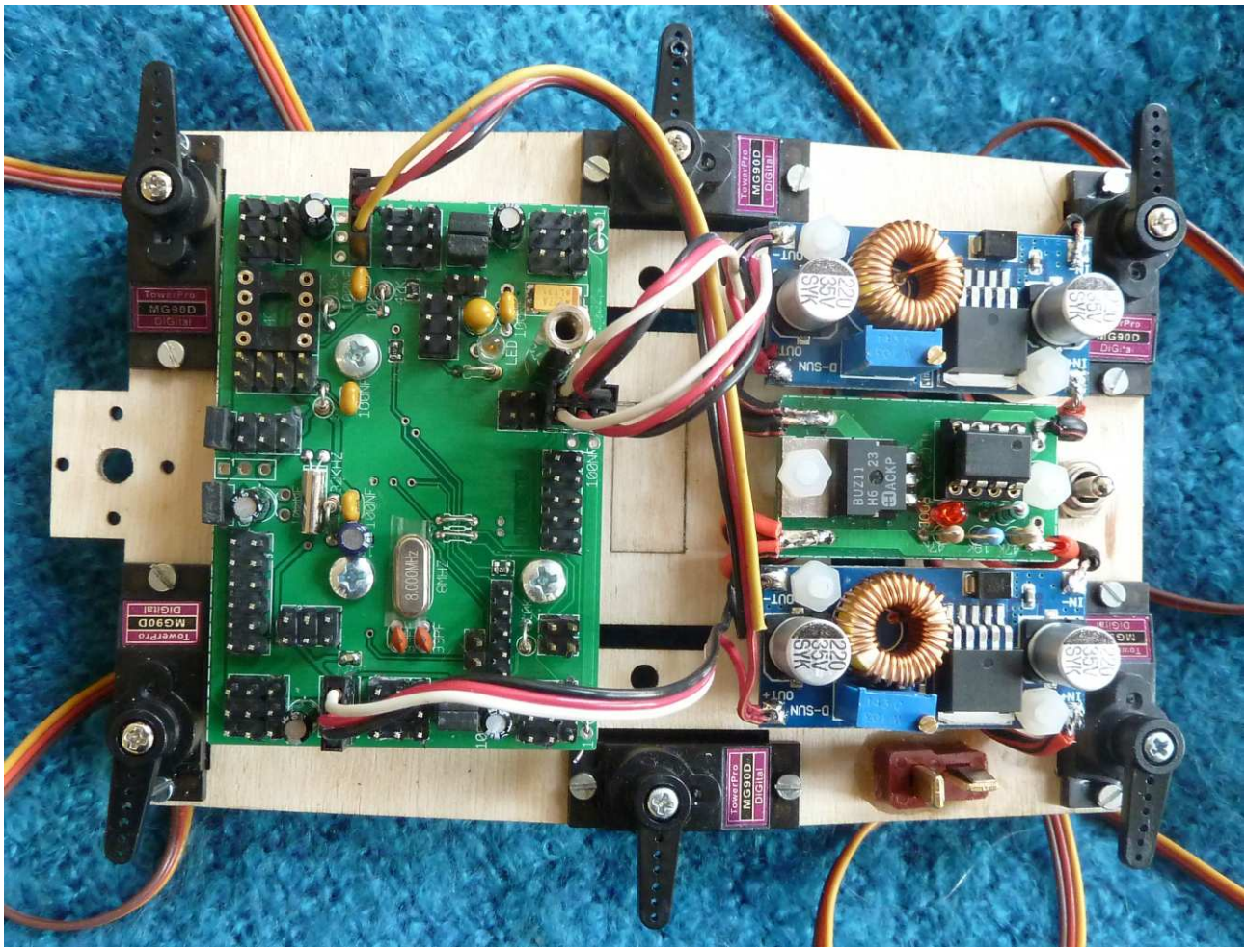
- A better weight distribution.
- Efficient placement of components
- Sufficient strength and rigidity

Laser cutted plywood or perspex:



One Leg:

The torso:



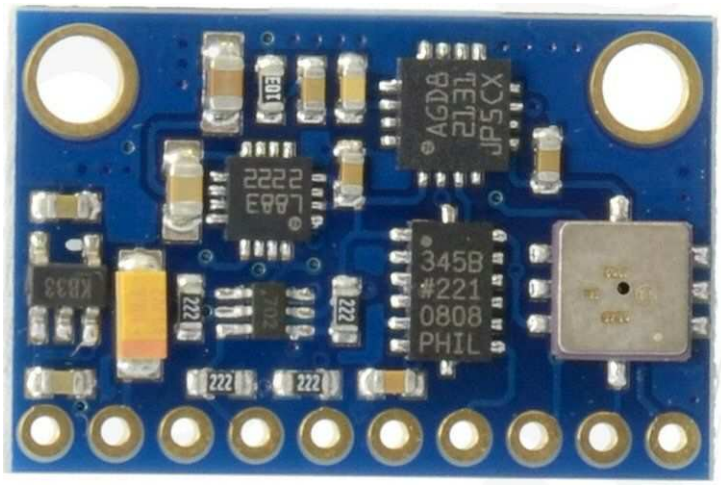
# 6) Sensors

Object detection:


Touch:

- Feelers (Antennas)
- Pressure on Legs

Ballance and coordination:




- Acceleration
- Gyroscope
- Compass
- Pressure

Internal state:

- Using the ADC of the microcontroller to measure the accu condition and temperature.

# 7) Applications

Experiments:

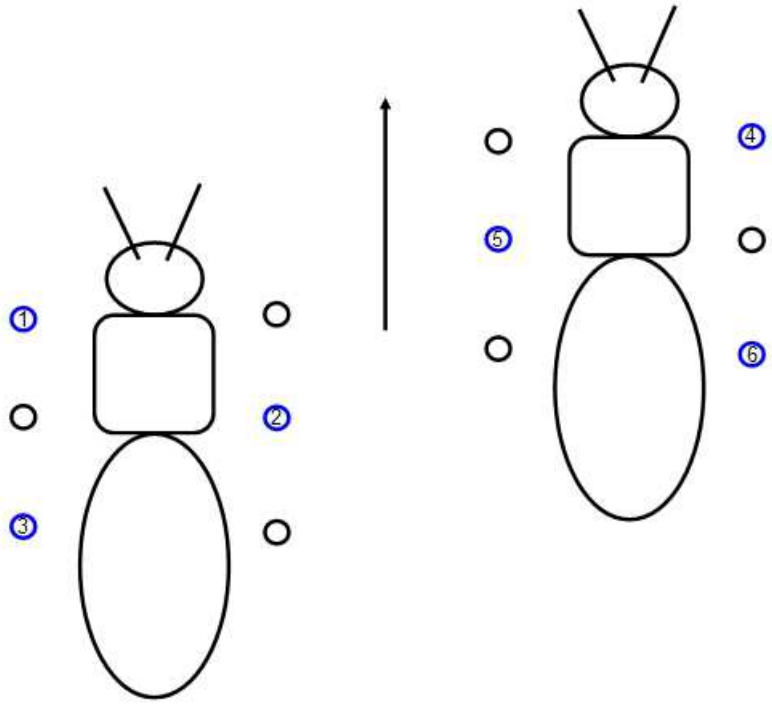
- Software implementations:
  - Absolute movement patterns
  - Relative movement patterns
  - Feedback

- Movement patterns

- Behaviour

- Sensors and the integration in software

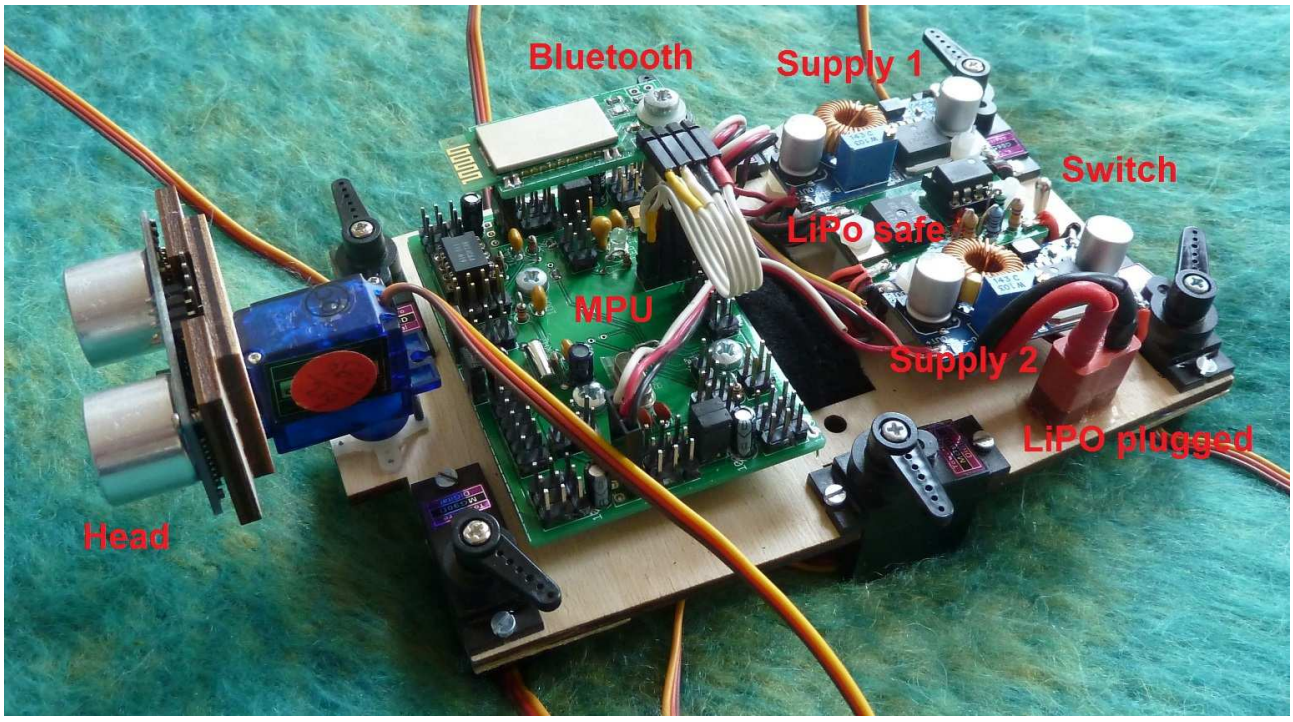
- Locomotion on non-planar surface



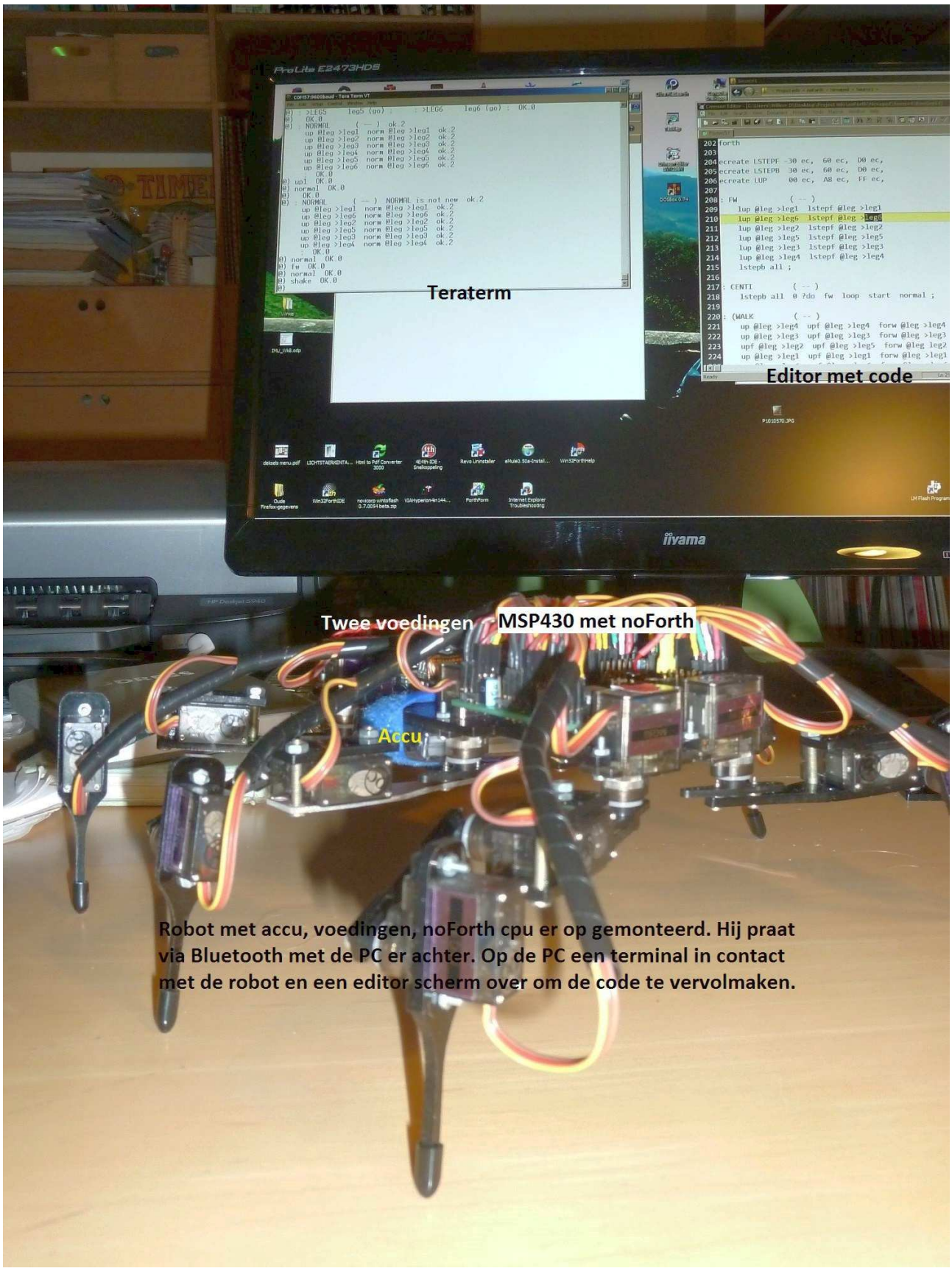
# 8) Hexapod demo

Demo commands:

a) Activate Hexapod and connect to it
b) `READY` - Slowly wake up and stand up
c) `WALK/BACKW` - Walk 'u' steps forward or backward
d) `LTURN/RTURN` - Turn 'u' steps to the left or right
e) `.SPEED` - Show current motion delay
f) `10 SPEED 5 WALK` - Walk using Piliplop with a delay of 10
g) `-40 SPEED 5 WALK` - Walk with a gesture delay of 40
h) `RCRAB/LCRAB` - Crab like walk, 'u' steps
i) `ANT` - Simulate an ant like walk
j) `LOW/HIGH` - Pushups
k) `REST2` - To rest position 2




**Hexapod-2 body**




**Hexapod with active communication and editor window on monitor.**

## Ant simulation:

```
\ ANT simulation routine by Gerard Vriens, translated to hexapod
value CHANCE    \ Random range
value ANGLE     \ Maximum angle
value STEPS     \ Forward steps

\ The scratch variant has a range: chance - angle to chance + angle
\ That is in the case of chance = 15 and angle = 1 from -14 tot 16
\ The simulation has an inclination to right rather then left
\ This variant is completely balanced. It uses antennas and a build
\ in reflex movement to avoid obstacles:
\ -chance - angle to chance + angle is -16 to 16
: GET-ANGLE     ( -- n )
    chance angle + 2* 1+ choose \ Choose angle
    chance angle +  -           \ Determine turning direction
    2/ 2/ ;                     \ A quarter is enough for hexapod

: SENS?         ( -- )      10 ms  01 01C bit* 0= ; \ Antenna?

: AVOID         ( -- )          \ Dodge with reflex movement
    sens? if
        even  -40 speed  2 backw  3 rturn  10 speed
    then ;

: FORW          ( s -- )         \ Do S steps forward
    0 ?do
        avoid  1 walk  ch . emit \ Step forward with escape
    loop ;

: TURN          ( -- )
    get-angle  dup .  ?dup 0=   \ New angle, angle = 0 ?
    if  even 1 forw  exit  then \ Yes, go forward and ready!
    dup 0 >                     \ Angle positive?
    if  right else left  then   \ Yes: turn right, No: turn left
    abs 0 ?do                   \ Take 1 or more steps to left or right
        avoid  1 walk           \ Step with escape
    loop ;

: ANT)          ( step angle chance -- )  \ Example:  1 8 15 ant)
    FE 01E c!  0 01D c!         \ Initialise input
    setup-random  even  up1     \ Hexapod stands up
    to chance to angle to steps \ Set help variables
    begin
        turn  even steps forw   \ Simulate ANT-like walk
    key? until  rest2 ;         \ Ready, go rest

: ANT           ( -- )     0 speed  1 8 0F ant) ; \ Ant demo

shield ANT\  freeze
```