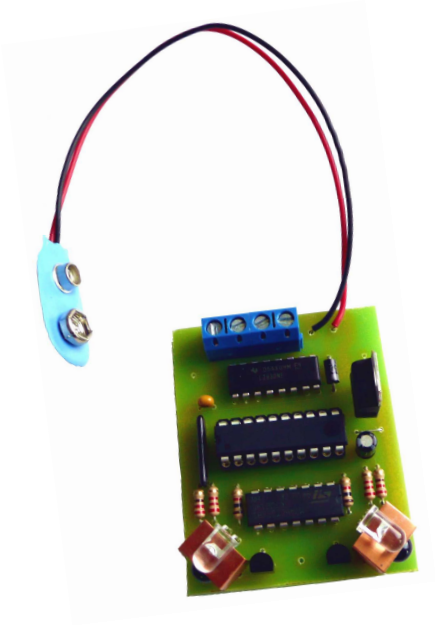
EPSICOM

Ready Prototyping

Colecția Roboți

EP 0060



Cuprins

ROBOT CU ATTINY2313

- Avantaj Pret/Calitate
- Livrare rapida
- Design Industrial
- Proiecte Modificabile
- Adaptabile cu alte module
- Module usor de asamblat
- Idei Interesante

Idei pentru afaceri

Hobby & Proiecte Educationale

www.epsicom.com/kits.php

a division of **EPSICO Manufacturing**

Un robot care, funcţie de cât de ingenios reuşim să realizăm mecanica, va rula, paşi sau se va târî. Prin modificarea mecanicii acesta ia funcţii diverse şi este amuzant la culme în deplasarea sa precum şi în modul în care evită obstacolele. Este uşor de realizat: cu numai doi senzori IR, un procesor şi două motoare cu reductor.

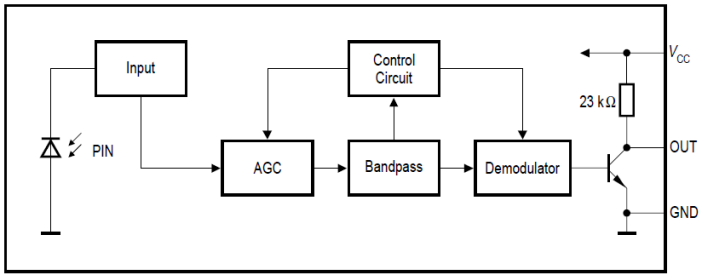
Caracteristici:

- Nu este perturbat de variaţii lumină/întuneric.
- Sesizează obstacole la distanţe de cca. 10cm
- Tensiune alimentare 9V

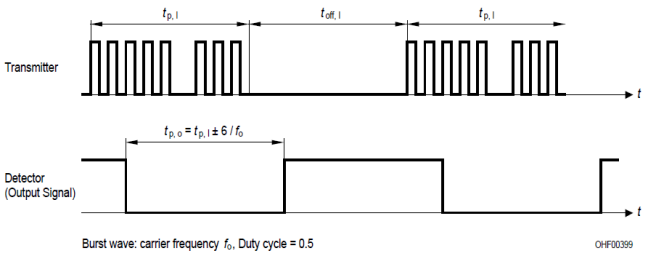
Funcţionare

Sistemul se bazează pe transmiterea unor impulsuri luminoase prin Led-urile IR D1 și D2 și captarea acestora cu receptorii în infraroșu IC3 și IC4.
Semnalele, așa cum se observă în schemă, sunt generate de un controller ATTINY2313 prin porturile PD2 și PD3, sunt aplicate pe bazele tranzistorilor Q1 și Q2 în colectoarele cărora sunt conectate Led-urile D1 și D2. Semnalele reflectate sunt captate de receptorii în infraroșu SFH511036, sunt demodulate, selectate cu semnalele din PD3 si PD4 prin porțile IC6C și IC6D și aplicate portului PD0/RXD.

Receptorul IR pe 940nm, cu filtru de lumină (daylight filter) detectează lumina în domeniul infraroșu şi cuprinde, pe langă fotodiodă, un preamplificator cu control automat al amplificării, al benzii de frecvenţă și un demodulator. Sensibilitatea receptorului este foarte mare astfel încât există posibilități mărite de modificare a distanței față de obstacol.



Frecvenţa de modulaţie a semnalului luminos este de 36KHz, corespunzătoare frecvenţei receptorului IR

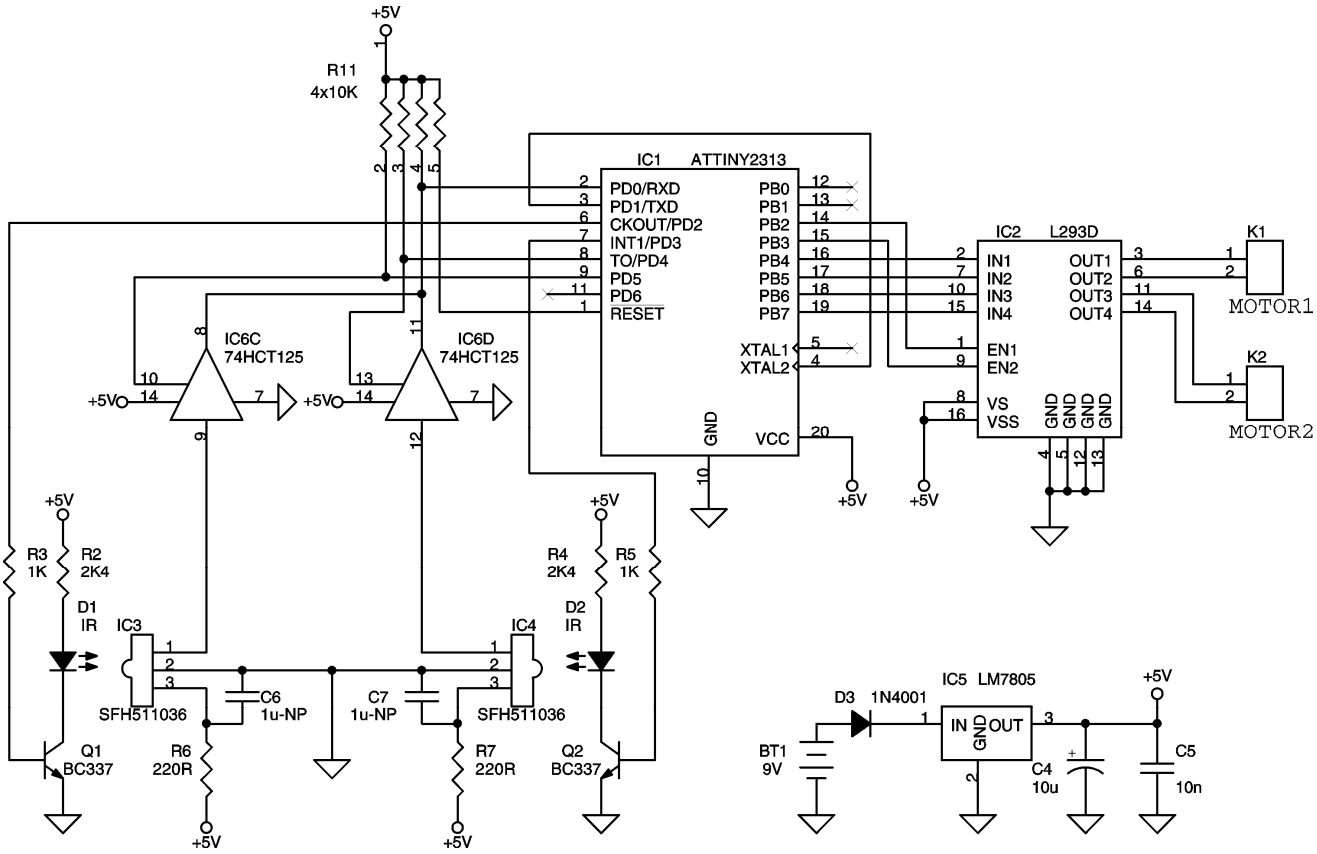


SFH511036. Pentru alt tip de receptor va trebui să fie modificată frecvenţa de modulaţie a semnalului luminos în programul controllerului.

Comanda motoarelor se face printr-un circuit driver de tip L293D cu semnalele primite de pe pinii porturilor PB4, PB5, PB6 si PB7. Circuitul L293D conţine diode de protecţie pe ieșiri şi elimină folosirea unei punți H cu tranzistoare. Motoarele se rotesc cu 44rpm la 3,2V.

Totul este plantat pe un circuit imprimat pe care se prinde şi ansamblul motoare + baterie de 9V, fixate pe o bucată de plastic sau aluminiu ce reprezintă „corpul” la care se ataşează rotiţe de la o jucărie sau două picioruşe cu lungime de cca. 20mm, prevăzute la capete cu bucaţele din cauciuc pentru a mări aderenţa la deplasare.

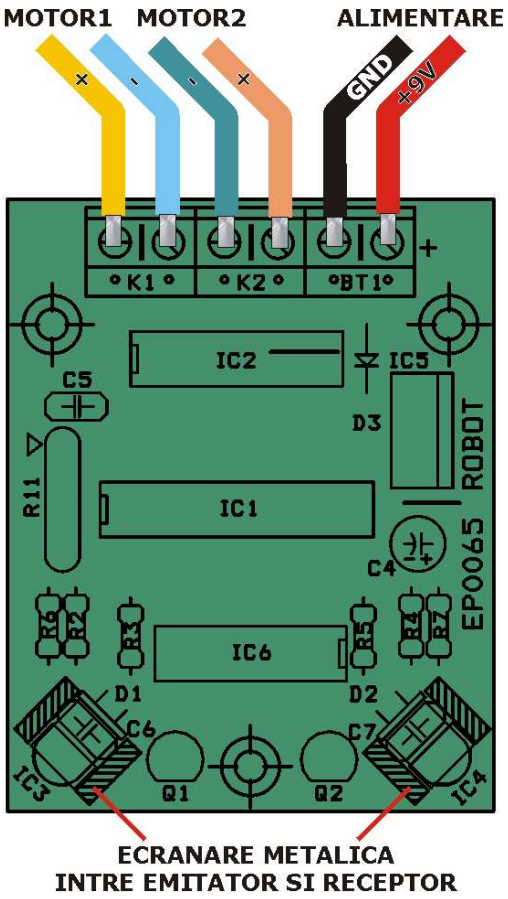
Soft-ul este astfel realizat încât permite rotirea motoarelor în ambele sensuri, respectiv deplasarea înainte sau înapoi a robotului.
La alimentare, robotul se deplasează înainte 5 secunde (inversați conectorii de la motor/motoare dacă observați o altă deplasare). Dacă întalnește un obstacol, se retrage 5 secunde și asteaptă trei secunde. Dacă ambii senzori detectează obstacol se retrage 3 secunde și isi roteste axa 4 secunde.



Schema electrică

Lista de componente

Nr.Crt.	Componenta	Denumire	Valoare	Cant
1	BT1	Baterie	9V	1
2	C4	Condensator pol	10µF	1
3	C5	Condensator np	10nF	1
4	C6,C7	Condensator np	1µF-NP	2
5	D1,D2	Led emițător	IR	2
6	D3	Diodă	1N4001	1
7	IC1	C.I.	ATTINY2313	1
8	IC2	C.I.	L293D	1
9	IC4,IC3	Receptor IR	SFH511036	2
10	IC5	Regulator tensiune	LM7805	1
11	IC6	C.I.	74HCT125	1
12	K2,K1	Conector	CON2	2
13	Q2,Q1	Tranzistor	BC337	2
14	R4,R2	Rezistență	2,4KΩ	2
15	R3,R5	Rezistență	1KΩ	2
16	R6,R7	Rezistență	220Ω	2
17	R11	Rezistență SIR	4x10KΩ	1



Amplasarea componentelor

Bibliografie

1. Multi PIC Programmer 5 Ver.2
2. WinPicProg 1.91
3. http://www.microengineeringlabs.com
4. http://www.techsystemsembedded.com
5. http://www.microchip.com
6. http://melabs.com

Acest produs se livrează în varianta circuit imprimat, circuit imprimat + componente sau în varianta asamblată în scopuri educaționale și va fi însoțit de documentația completă de asamblare pe CD.

```
.include "tn2313def.inc"

.equ    clock       = 8000000 ;10MHZ clock frequency
.equ    baudrate    = 9600     ;baudrate is 9600
.equ    baudconstant= (clock/(16*baudrate))-1

.equ    TSTOP   = 0            ;Stop Timer/Counter


.equ    STACKTOP    = RAMEND - 100

;***********************************************
;* Registers used                              *
;***********************************************

.def    Done        = r10      ;used in GetROMs
.def    ROMCount    = r11      ;number of devices found
.def    SaveSREG    = r12      ;used in Delay
.def    BitNum      = r13      ;used in ROMsrch
.def    LastDis     = r14      ;used in ROMsrch
.def    LasDisCpy   = r15      ;used in ROMsrch

.def    AReg        = r16      ;working reg, like the 68HC11 A register
.def    BReg        = r17      ;working reg, like the 68HC11 B register
.def    ScratchL    = r18
.def    ScratchH    = r19
.def    Tick        = r20
.def    status      = r21
.def    DelayVal_H  = r22
.def    DelayVal_L  = r23
.def    Dir         = r31                  //0 = left, 1 = right



.cseg
.org    $000                ;Reset Vector
        rjmp    Reset

 ;Interrupt vectors

.org      OVF1addr
                rjmp Timer1_Int
.org    URXCaddr
        rjmp URXC_Int


;***********************************************
;* Timer1 overflow interrupt handler           *
;***********************************************
Timer1_Int:
        cli                             ;disable interrupts
        ldi             Tick, 0x0
        ldi     ScratchH, TSTOP         ;Stop Timer 0
        out     TCCR1B, ScratchH
        sei                             ;re-enable interrupts
        reti


URXC_Int:
                cli
                sbi             PORTD, 4
                sbi             PORTD, 5
                push    AReg
                in              AReg, UDR
```

```
                cpi             AReg, 0xaa
                brne    exit_u
                cpi     Dir, 0x0
                brne    test0
                ldi     status, 0x1
                ldi             Tick, 0x0
exit_u:
                pop             AReg
                sei
                reti
test0:
                cpi             status, 0x1
                brne    test1
                ldi             status, 0x3
                ldi             Tick, 0x0
                rjmp    exit_u
test1:
                ldi             status, 0x2
                ldi             Tick, 0x0
                rjmp    exit_u


;***********************************************
;* Delay   time in us based on CLK value      *
;* DelayVal holds the delay value.            *
;* ScratchH has the prescaler value           *
;***********************************************
Delay:
        out     TCNT1L, DelayVal_L
                out     TCNT1H, DelayVal_H
        ldi             Tick, 0x1
        out     TCCR1B, BReg
Dwait:
        cpi             Tick, 0x1
                breq    Dwait
        ret

Delay_ext:
                out     TCNT1L, DelayVal_L
                out     TCNT1H, DelayVal_H
        ldi             Tick, 0x1
        out     TCCR1B, BReg
                ret

Send_byte:
                out       UDR, AReg
                ret

Pulse_right:
                sbic      PINA, 1
                rjmp    exit_test_r
                sbi             PORTD, 2
                ldi       R24, $26
WGLOOP0:
                dec       R24
        brne      WGLOOP0
                nop
                nop
                cbi             PORTD, 2
                ldi       R24, $26
WGLOOP1:
                dec       R24
        brne      WGLOOP1
                nop
                nop
```

```
exit_test_r:
                cpi             Tick, 0x1
                breq    Pulse_right
                ret

Pulse_left:
                sbic    PINA, 1
                rjmp    exit_test_l
                sbi             PORTD, 3
                ldi     R24, $26
WGLOOP2:
                dec     R24
        brne    WGLOOP2
                nop
                nop
                cbi             PORTD, 3
                ldi     R24, $26
WGLOOP3:
                dec     R24
        brne    WGLOOP3
                nop
                nop
exit_test_l:
                cpi             Tick, 0x1
                breq    Pulse_left
                ret


Process_status:
                //cli
                cpi             status, 0x0
                brne    Ps0
                ldi             AReg, 0xac              //INAINTE
                out             PORTB, AReg
                //sei
                ret
Ps0:
                cpi             status, 0x1
                brne    Ps1
                ldi             AReg, 0x5c              //INAPOI
                out             PORTB, AReg
                ldi             BReg, 0x5               //0.5s
                ldi         DelayVal_L, 0xbd
                ldi         DelayVal_H, 0xf0
                rcall   Delay
                ldi             AReg, 0x88
                out             PORTB, AReg
                ldi             BReg, 0x5               //0.5s
                ldi         DelayVal_L, 0xbd
                ldi         DelayVal_H, 0xf0
                rcall   Delay
                ldi             AReg, 0xac
                out             PORTB, AReg
                ldi     status, 0x0
                ret
Ps1:
                cpi             Status, 0x2
                brne    Ps2
                ldi             AReg, 0x5c
                out             PORTB, AReg
                ldi             BReg, 0x5               //0.5s
                ldi         DelayVal_L, 0xbd
                ldi         DelayVal_H, 0xf0
                rcall   Delay
                ldi             AReg, 0x24
```

```
                out             PORTB, AReg
                ldi             BReg, 0x5           //0.5s
                ldi         DelayVal_L, 0xbd
                ldi         DelayVal_H, 0xf0
                rcall   Delay
                ldi             AReg, 0xac
                out             PORTB, AReg
                ldi    status, 0x0
                ret
Ps2:
                cpi             Status, 0x3
                brne     Ps3
                ldi             AReg, 0x5c
                out             PORTB, AReg
                ldi             BReg, 0x5           //1s
                ldi         DelayVal_L, 0x7b
                ldi         DelayVal_H, 0xe1
                rcall   Delay
                ldi             AReg, 0x6c
                out             PORTB, AReg
                ldi             BReg, 0x5           //0.5s
                ldi         DelayVal_L, 0xbd
                ldi         DelayVal_H, 0xf0
                rcall   Delay
Ps3:
                ldi             AReg, 0xac
                out             PORTB, AReg
                ldi    status, 0x0
                ret



Setup:
                ldi    AReg, 0x0
                out    DDRA, AReg
                ldi    AReg, 0x3
                out    PORTA, AReg

                ldi    AReg, 0x7e
                out    DDRD, AReg
                ldi    AReg, 0x73
                out    PORTD, AReg

                ldi    AReg, 0xfc
                out    DDRB, AReg
                ldi    AReg, 0xac
                out    PORTB, AReg

                ldi    AReg, 0x0
                out    UCSRA, AReg
                ldi         AReg, 0x98
                out    UCSRB, AReg
                ldi         AReg, 0x6
                out    UCSRC, AReg
                ldi         AReg, 0x0
                out    UBRRH, AReg
                ldi         AReg, 0xff
                out    UBRRL, AReg

                ldi    AReg, 0x0
                out    TCCR1A, AReg
      out    TCCR1B, AReg
                out    TCCR1C, AReg
                out    TCNT1H, AReg
                out    TCNT1L, AReg
```

```
                out   ICR1H,  AReg
                out   ICR1L,  AReg
                out   OCR1AH, AReg
                out   OCR1AL, AReg
                out   OCR1BH, AReg
                out   OCR1BL, AReg
                ldi   AReg, 0x80
                out   TIMSK, AReg

                ldi   AReg,TSTOP
        out   TCCR0B,AReg

                sei

        ret



Reset:
        ldi   AReg,low(STACKTOP)
        out   SPL,AReg
        rcall Setup
                ldi         AReg, 0xac          //INAINTE
                out         PORTB, AReg

Loop:
                ldi   status, 0x0
                ldi   Dir, 0x0
                ldi         Tick, 0x1
                ldi         BReg, 0x1
                ldi         DelayVal_L, 0xff
                ldi         DelayVal_H, 0x82
                rcall Delay_ext

                ldi         AReg, 0xaa
                cbi   PORTD, 4                  //STINGA
                rcall Send_byte

                rcall Pulse_left

                ldi         Tick, 0x1
                ldi         BReg, 0x1
                ldi         DelayVal_L, 0xff
                ldi         DelayVal_H, 0x82
                rcall Delay

                sbi         PORTD, 4
                cpi         status, 0x0
                breq  Loop1
                rcall Process_status
Loop1:
                ldi         Tick, 0x1
                ldi   Dir, 0x1
                ldi         BReg, 0x1
                ldi         DelayVal_L, 0xff
                ldi         DelayVal_H, 0x82
                rcall Delay_ext

                ldi         AReg, 0xaa
                cbi   PORTD, 5                  //DREAPTA
                rcall Send_byte
                rcall Pulse_right

                ldi         Tick, 0x1
                ldi         BReg, 0x1
```

```
ldi         DelayVal_L, 0xff
ldi         DelayVal_H, 0x82
rcall  Delay

sbi         PORTD, 5
cpi    status, 0x0
breq   Loop
rcall  Process_status
rjmp   Loop
```