Andrei CADOR (78016) - MoveIT (Masinuta teleghidata controlata cu miscarile mainii)

Autorul poate fi contactat la adresa: **Login pentru adresa**

Introducere

Salut !

Contrar multor pareri conform carora jucariile copilariei raman abandonate/uitate in vreun colt de sertar (au fost unele, nu neg), o buna parte s-au stricat dupa nu foarte mult timp de distractie. Mie imi placeau foarte mult masinutele teleghidate, dar o data la 2 luni telecomanda, facuta cu niste joystick-uri deplorabila, cu un plastic mai ieftin decat un sul de hartie igienica, nu mai functiona. Electronist nu eram in tinerete, deci telecomanda ajungea, impreuna cu masinuta, in cosul de gunoi. Naspa.

In plus, parca nu esti chiar in mijlocul actiunii, stand in spatele unui controller cu o antena de 3,71 metri, dand de niste butoane. Sigur, asa am crescut toti, dar exista alternative mai amuzante! De asta m-am gandit la **MoveIT**. **MoveIT** este o masinuta inteligenta a carui ghidaj este realizat prin miscarile mainii.

Cum merge? Foarte simplu. Vei purta o manusa care are atasat un accelerometru, iar prin intermediul transmisiei Radio (ca la o masina teleghidata clasica), un decodificator iti va interpreta comenzile si va controla cele doua motoare ale masinii ca sa obtii efectul dorit(sau, cel putin, cel indicat).

Asta te scuteste de o telecomanda plictisitoare si asigura ca o sa te distrezi copios, pentru ca, desi pare ceva mai naturala decat miscarea unui knob/joystick, este o miscare pe care n-ai mai intalnit-o si-ti va lua ceva sa te prinzi cum merge smecheria!

```
                                                                    Spor la
joaca!
```

Descriere generală

```
            Cum prezentarea este inclusa intr-un proiecte ingineresc si cum
toata lumea iubeste diagramele, am inclus un model grafic al lui MoveIT :
```

MoveIT include:

- colectarea datelor despre inclinatia mainii se face utilizand accelerometrul
- bineinteles, alimentarea se face utilizand baterii portabile(9V pentru Transmitator, 5V pentru Receptor)
- procesarea pozitiei, interpretarea comenzilor si comandarea motoarelor se face utilizand MCU din familia ATMega
- transmiterea si receptia au nevoie de un mecanism de encoding-decoding
- comenzile vor ajunge la masinuta prin transmiterea radio, folosind 2 module de comunicatie radio(RFTransmitter si RFReceiver), inclusiv cate o antena pentru fiecare parte a comunicatiei.
- Driver-ul este folosit pentru a comanda complet si corect directia de deplasare(prin interemediul microcontrollerului, va fi ajustata si viteza).

Hardware Design

Aici gasiţi tot ce ţine de hardware design:

- ATMega 324A - masina
- ATMega 328p - manusa
- Accelerometru MPU6050
- 2x RF transceiver nRF24L01
- 2x Driver Dual

Piesa	Cantitate	Pret(lei)
Condensator Polarizat 10uF	1	1.20
Condensator Polarizat 1uF	1	1
Condensator Ceramic 0.1pF	1	0.20
Fire Breadboard	65	9.99
Transceiver nRF24L01	2	6.99
Accelerometru MPU6050	1	17.99
Driver Motor DRV8833	2	19.99
Arduino Nano (chinezarie)	1	19.99
DC Motor	4	4.99
Suport 4 baterii AA	1	2.99
Baterii AA	4	2.99
Baterie 9V + suport	1	7.99

Software Design

- IDE : AVR Studio - pentru ATMega324A (placuta PM)
- IDE : Arduino Studio - pentru ATMega328P → Arduino Nano (desi codul proiectului din Arduino Studio este scris in C, nu in C++)
  - Pentru evitarea Arduino Studio se poate folosi IDE-ul CodeBlocks Arduino, dar nu ofera SerialMonitor(bun pt debugging) si are o interfata mai prost construita.

```
                                                                Functionalitatea
codului este descrisa mai jos:
```

Inclinatia mainii reprezinta principalul obiect al controlului masinii. Pentru a comanda motoarele este nevoie de 2 lucruri : construirea unui canal de comunciatii si alegerea unei modalitati de gestionare a pozitiei mainii. Inclinatia mainii este determinata cu ajutorul unui ansamblu Accelerometru-Giroscop, montat pe manusa.

Procesarea datelor brute are loc pe microcontroller-ul ATMega328p aflat pe modulul Arduino Nano, utilizand o biblioteca dedicata acestui accelerometru (apartinand lui jrowberg). Datele sunt stocate intr-un tablou unidimensional ale carui elemente vor fi inlocuite cu valori logice, relevante pentru receptor (e inutil sa trimit valori float cand la receptie vrem sa aflam doar directia de rotatie a rotilor).

Transmisia datelor de pe “manusa” la procesorul aflat pe Masinuta se face utilizand un modul Wireless(nRF24L01), modul pentru care poti gasi foarte multe informatii si exemple pe Google si pe Git. Interfatarea cu modului Wireless se face, desi usor laborios, destul de facil daca urmaresti un tutorial decent. In principal, comunicarea presupune schimbarea a 3-5 registre in functie de forma mesajului transmis. De incapsulare, de CRC si de AutoACK se ocupa nRF.

La receptie, dupa cum am amintit si mai sus, primim doar valori logice (porneste motorul 1, cu sensul de rotatie pentru avans - reprezentat printr-o valoare mare, pozitiva, de 2 cifre). Urmeaza comandarea motoarelor in functie de caracterul miscarii impus de transmitatorul de manusa, implicit de inclinatia mainii.

Fluxul de date este de aproximativ 10pachete/secunda. Mai mult de 10 pachete/secunda, motoarele devin greu de controlat(trebuie sa ai miscari extrem de ferme ale mainii), iar o viteza mai mica implica o latenta in miscarea masinii.

```
                                                                Bucati de cod
semnificative pentru modulul radio:
```

```
                                                                Initializarea
modului nRF24L01 pentru receptie:



void nrf_init(uint8_t inittype){

uint8_t val[5];
uint8_t i;

val[0]=0x01;
nrf_writeregister(EN_AA,val,1); // enable la Pipe-ul 1
```

```
val[0]=0x01;

// activam prima adresa de receptie
nrf_writeregister(EN_RXADDR,val,1);


for(i=0;i<5;i++)
{
        val[i]=0x3C;
}
nrf_writeregister(RX_ADDR_P0,val,5); // setam adresa de receptie = 60


for(i=0;i<5;i++)
{
        val[i]=0x3C;
}
nrf_writeregister(TX_ADDR,val,5); // setam adresa de transmisie = 60


val[0]=0x05; // 5 bytes pe transmisie
nrf_writeregister(RX_PW_P0,val,1);


if(inittype==INIT_TX)
{
        val[0]=0x0E;
}
if(inittype==INIT_RX)
{
        val[0]=0x0F;
}
nrf_writeregister(CONFIG,val,1); // 0 = receptor , 1 = transmitator


// canalul 40 - 2,440 GHz
val[0] = 0x00 ;

// canalul = adaos de frecventa la 2.4 GHz
nrf_writeregister( RF_CH , val , 1);
 }




                                                                Receptia
Mesajului:


void nrf_rx(){
```

```
CSN_low; // activam modulul
SPI_exchange(FLUSH_RX); // stergem stiva de receptie
CSN_high; // Nu mai avem de dat comenzi
CE_high; // asteptam mesaj
_delay_ms(50);
CE_low; // am primit mesaj
}
```

Stocarea Mesajului:

```
uint8_t* nrf_readdata()      {

int i;
static uint8_t recvdata[5];
CSN_low; // activam modulul
SPI_exchange(R_RX_PAYLOAD); // trimitem numarul de bytes asteptat
for(i=0;i<5;i++)
{
        // trimitem dummy bytes pentru a primi continutul mesajului
        recvdata[i]=SPI_exchange(NOP);
}
CSN_high; // nu mai avem nimic de transmis
return recvdata;
    }
```

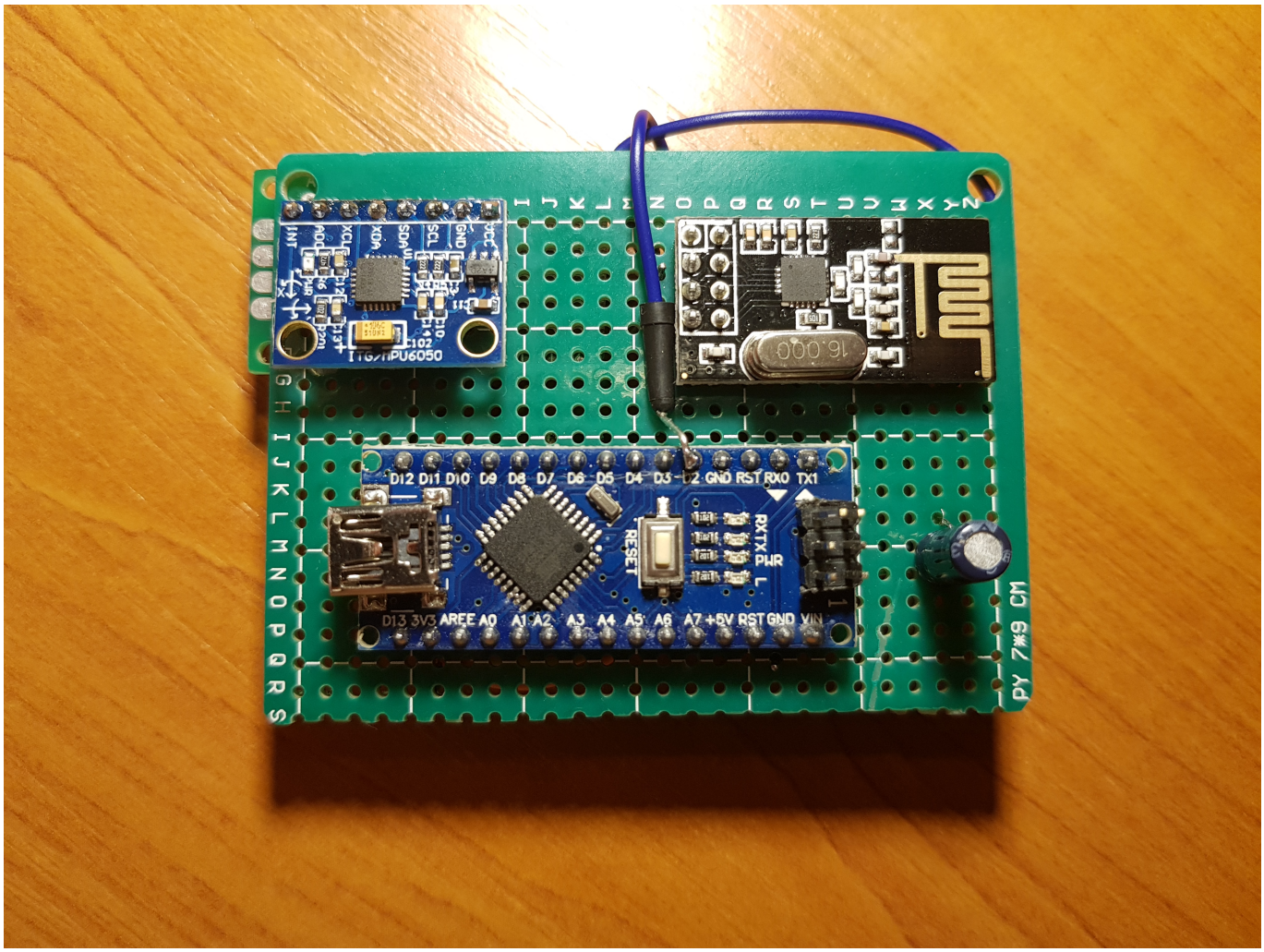
Exemplu de controlare a motoarelor

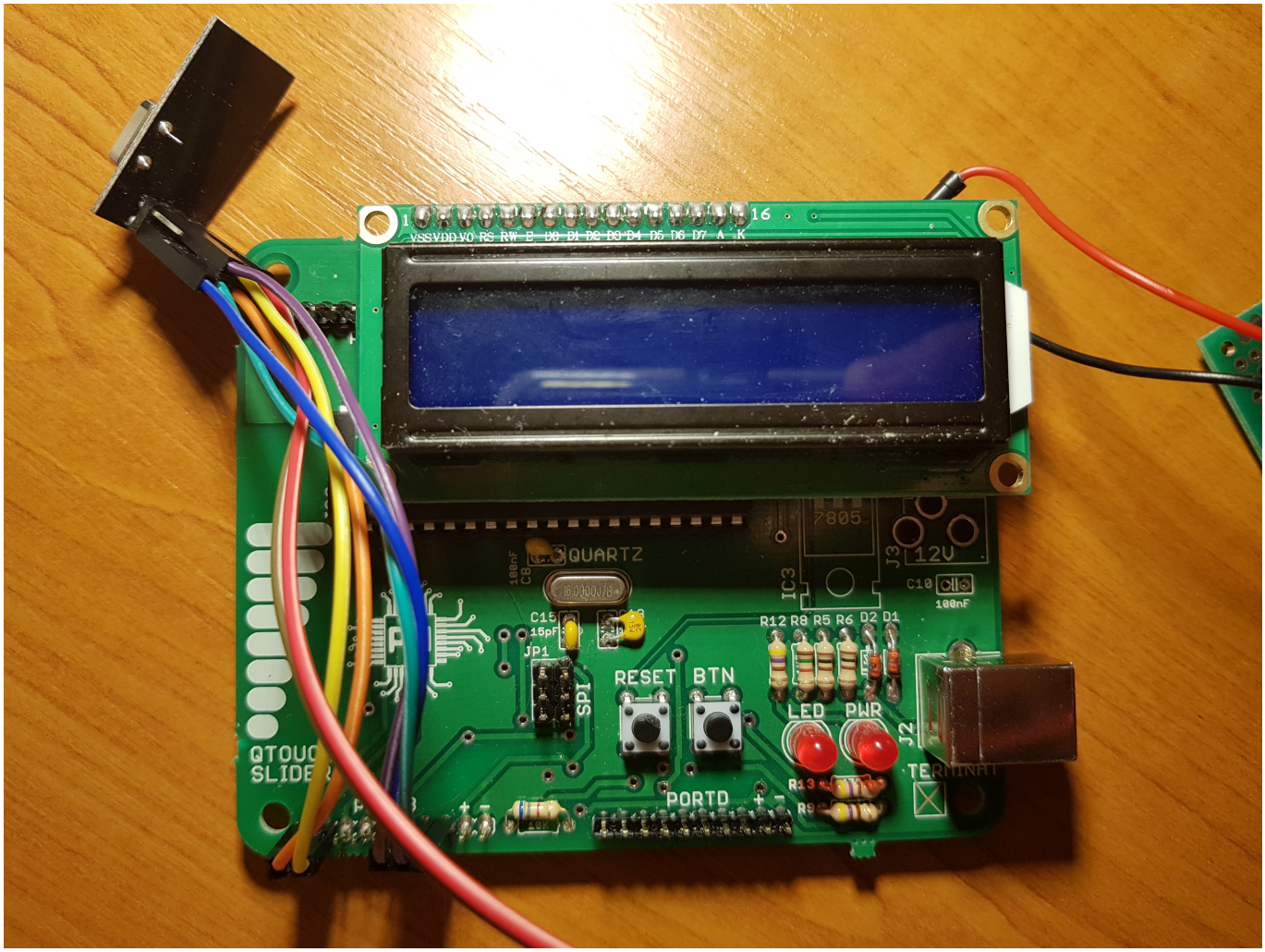
```
    if ( recvdata[0] == 1 && recvdata[3] == 4 )// stanga-fata
{
                      // MxA = rotatie pentru avans, MxB pentru sensul opus
                      //acceleram doar motorul din dreapta-fata ca sa
deplasam masina spre stanga. Toate celelalte motoare sunt pe 0.

                      PORTD &= ~( 1<< M1A) ;
                PORTD &= ~( 1<< M2A) ;
                PORTD &= ~( 1<< M3A) ;
                      PORTB |=  ( 1<< M4A) ;//motorul de actionat

                PORTD &= ~( 1<< M1B) ;
                PORTD &= ~( 1<< M2B) ;
                PORTD &= ~( 1<< M3B) ;
                PORTB &= ~( 1<< M4B) ;
```

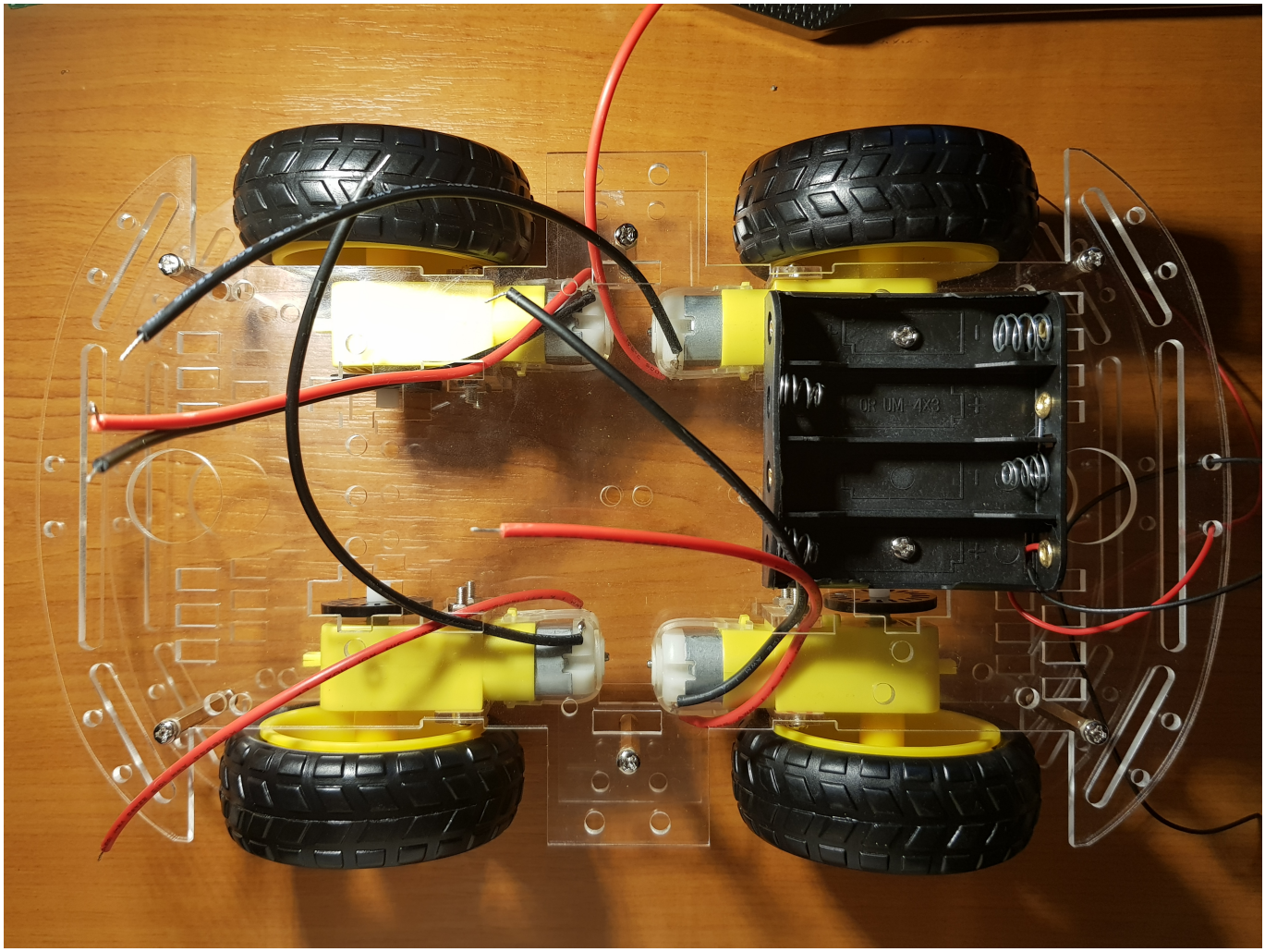
```
}
```

Rezultate Obţinute

Mai jos sunt atasate pozele din timpul constructiei proiectului. Comenzile corecte au fost interpretate conform modulelului impus, cele incoerente/gresite au fost respinse. Controlul este destul de natural, fara intarzieri semnificative si fara sensibilitate excesiva a directiei de deplasare.




VSSVDDVO RS RW E D0 D1 D2 D3 D4 D5 D6 D7 A K
7805
QUARTZ
12V
C15
15pF
JP1
SPI
RESET
BTN
LED
PWR
PORTD
J2
J3
IC3
C10
100nF



Aici puteti gasi un video cu functionarea Masinii . Enjoy! https://youtu.be/BHSE6Lwej5o

Concluzii

Proiectarea, Asamblarea, Testarea si Imbunatatirea(niciodata etapa de Proiectare nu se face suficient de bine) din cadrul ProiectuluiPM au fost de departe cele mai intrigante etape ale vreunui proiect univarsitar. A fost complex, cu cerinte reale si cu un deadline realist. Am fost in fiecare stadiu intermediar de pe scara entuziasm → “crapa-l cu un ciocan”. Varianta finala a proiectu a fost, oricum, extrem de satisfacatoare.

Download

Descarcarea arhivei proiectului :moveit.zip

Jurnal

1. Comandarea pieselor pentru comunicatia wireless (doar trimiterea si receptionarea unor siruri de caractere)
2. Stabilirea protocolului si programarea MCU pentru comunicatia radio → cea mai lunga parte
3. Comandarea pieselor de actionarea (Accelerometru, Motoarele, Driverele)
4. Programarea MCU doar pentru interactiunea cu mediul fizic (programarea specifica elementelor de mai sus)
5. Asamblarea celor doua bucati de proiect
6. Stabilirea sensibilitatii accelerometrului si a sensibilitatii actionarii motoarelor
7. Stabilirea vitezei si a incarcaturii optime pentru transmisia wireless
8. Debugging
9. Testing
10. Si mai mult Debugging + Testing

Bibliografie/Resurse

Listă cu documente, datasheet-uri, resurse Internet folosite:

- Biblioteca de MPU6050 : https://github.com/jrowberg/i2cdevlib/tree/master/Arduino/MPU6050
- Biblioteca de mapare a registrelor nRF24L01 : https://github.com/kehribar/nrf24L01_plus/blob/master/nRF24L01.h
- Datasheet nRF24L01 : https://www.sparkfun.com/datasheets/Components/SMD/nRF24L01Pluss_Preliminary_Product_Specification_v1_0.pdf
- Datasheet MPU6050 : https://www.invensense.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf

- Documentația în format PDF
- Schema electrica în format PDF: pm-sch-el.pdf