MOTOR SPEED CONTROL

Prof.dr.ing. Chiriță Doinița

Noțiuni generale despre Arduino

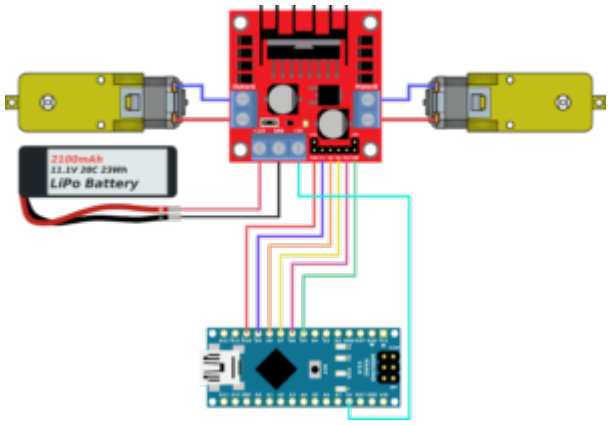
Arduino este o companie open-source care produce atât plăcuțe de dezvoltare bazate pe microcontrolere, cât și partea de software destinată funcționării și programării acestora. Pe lângă acestea include și o comunitate uriașă care se ocupă cu creația și distribuirea de proiecte care au ca scop crearea de dispozitive care pot sesiza și controla diverse activități sau procese în lumea reală.

Proiectul este bazat pe designul plăcilor cu microcontroler produse de câțiva furnizori, folosind diverse tipuri de microcontrolere. Aceste plăci pun la dispoziția utilizatorului pini I/O, digitali și analogici, care pot fi interfațați cu o gamă largă de plăcuțe numite scuturi (shield-uri) și/sau cu alte circuite. Plăcile au interfețe de comunicații seriale, inclusiv USB pe unele modele, pentru a încărca programe din calculatorele personale. Pentru programarea microcontrolerelor, Arduino vine cu un mediu de dezvoltare integrat (IDE) bazat pe proiectul Processing, care include suport pentru limbaje de programare ca C și C++.

Un microcontroler instalat pe Arduino vine preprogramat cu un bootloader care simplifică încărcarea programelor pe memoria flash a cipului, în comparație cu alte dispozitive care necesită programatoare externe. Acest aspect face Arduino o soluție simplă, permițând programarea de pe orice computer ordinar. În prezent, bootloader-ul optiboot este bootloader-ul implicit instalat pe Arduino NANO.

Controlul vitezei motoarelor

1.1 Schema montajului

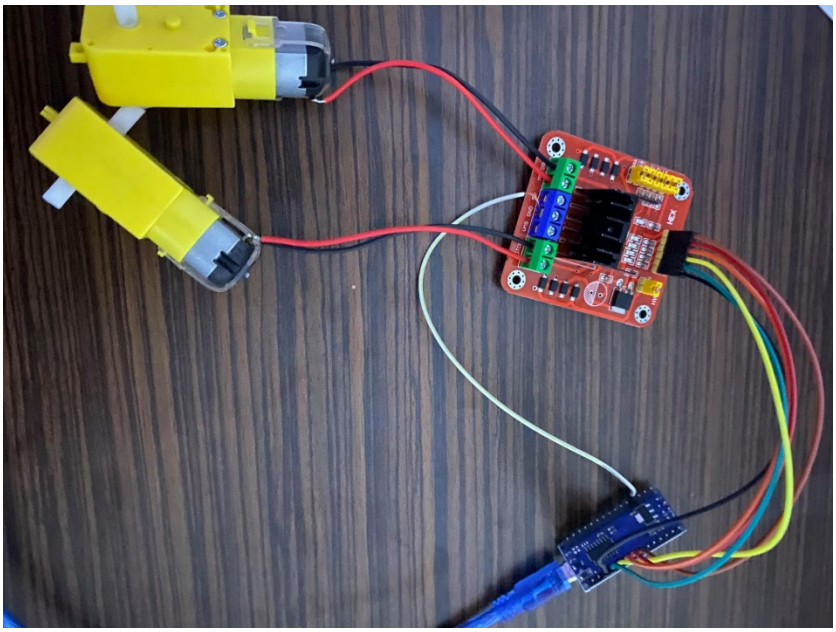


Elemente componente:

- Arduino Nano
- Motoare (2 bucati)
- L298N Modul

- Fire de legatura de tip tata – tata
- Fire de legatura de tip mama - mama
- Baterie LiPo 11.1V

1.2 Montajul Practic



1.3 Elemente componente:

- Arduino Nano
- X2 Motoare
- L298 Modul
- fire de legatura
- Baterie

1.4 **Principiul de functionare:**

Circuitul corespunzator temei de proiect alese functioneaza in felul urmator:

Cele 2 motoare sunt conectate la modulul L298N deoarece motoarele necesita multa putere, modulul este conectat la Arduino Nano prin intermediul firelor de legatura. Modulul necesita si o baterie de 11.1V pentru alimentarea motoarelor.

Astfel dupa scrierea codului in Arduino IDE, microcontroller-ul va transmite comanda in intreg circuitul. In caz de scrierea codului gresit, software-ul va ajuta la gasire erorii, putand sa fie rezolvata cu usurinta. Pentru ca totul sa functioneze corespunzator, mai intai, Arduino Nano trebuie testat prin conectarea unui LED intr-un breadboard si introducera unui cod simplu, care prin anumite indicatii va arata daca totul functioneaza perfect. Iar dupa ce ati facut acest lucru, se poate incepe montarea si introducerea codului in Arduino Nano.

Codul are rolul de a utiliza motoarele in toate modurile posibile, prin intermediul său putem ajusta viteza motoarelor individual sau in acelasi timp dar si polaritatea lor

1.5 Program: Arduino IDE

Programele Arduino pot fi scrise în orice limbaj de programare cu un compilator capabil să producă un cod mașină binar. Atmel oferă un mediu de dezvoltare pentru microcontrolerele sale, AVR Studio și mai nou, Atmel Studio.

Proiectul Arduino oferă un mediu integrat de dezvoltare (IDE), care este o aplicație cross-platform, scrisă în Java. Acesta își are originile în mediul de dezvoltare pentru limbajul de programare Processing și în proiectul Wiring. Este proiectat pentru a introduce programarea în lumea artiștilor și a celor nefamiliarizați cu dezvoltarea software. Include un editor de cod cu funcții ca evidențierea sintaxelor, potrivirea acoladelor și spațierea automată și oferă mecanisme simple cu un singur click, pentru a compila și a încărca programele în plăcuța Arduino. Un program scris în IDE pentru Arduino se numește *sketch*.

Arduino IDE suportă limbajele de programare C și C++ folosind reguli speciale de organizare a codului. Arduino IDE oferă o librărie software numită Wiring, din proiectul Wiring, care oferă multe proceduri comune de intrare și ieșire. Un sketch tipic Arduino scris în C/C++ este compus din două funcții care sunt compilate și legate cu un ciot de program ***main()***, într-un program executabil cu o execuție ciclică:

- *setup()*: o funcție care este rulată o singură dată la începutul programului, când se inițializează setările.
- *loop()*: o funcție apelată în mod repetat până la oprirea alimentării cu energie a plăcuței.

După compilarea și legarea cu GNU toolchain inclus, de asemenea, în IDE, mediul de dezvoltare Arduino trimite comandă către programul avrdude pentru a converti codul executabil într-un fișier text codat hexazecimal, care poate fi încărcat în placa Arduino de un program de încărcare.

1.6 Codul realizat in Arduino IDE pentru functionarea proiectului:

```
/********* Connections **********/

// Motor connections
```

```
#define   IN1   9
#define   IN2   8
#define   IN3   7
#define   IN4   6
#define   ENA   10
#define   ENB   5

/******** Variables ********/
// Direction variables
#define   F     1
#define   FR    2
#define   R     3
#define   BR    4
#define   B     5
#define   BL    6
#define   L     7
#define   FL    8
#define   STOP  0
#define   SPEED_CONSTANT  0.2

void setup() {
  for (int i = 5; i <= 10; i++) {
    pinMode(i, OUTPUT);
```

```
  }
}

void motor(int dir, int speed);
void loop() {
  // 1000ms forward @150 speed
  motor(F, 150);
  delay(1000);
  // 500ms right @200 speed
  motor(R, 200);
  delay(500);
  // 1000ms forward @150 speed
  motor(F, 150);
  delay(1000);
  // 1000ms backward @150 speed
  motor(B, 150);
  delay(1000);
  // 500ms left @200 speed
  motor(L, 200);
  delay(500);
  // 1000ms backward @150 speed
  motor(B, 150);
  delay(1000);
```

```
}

/*
  dir:
  F- Forward
  FR- Forward Right
  R- Right
  BR- Backward Right
  B- Backward
  BL- Backward Left
  L- Left
  FL- Forward Left
  STOP- Stop

  speed:
  min: 0, max: 255
*/
void motor(int dir, int speed) {
  if (dir == F) {
    // Left motor forward with full speed
    digitalWrite(IN1, HIGH);
    digitalWrite(IN2, LOW);
    analogWrite(ENA, speed);
```

```
  // Right motor forward with full speed
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
  analogWrite(ENB, speed);
} else if (dir == FR) {
  // Left motor forward with full speed
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  analogWrite(ENA, speed);
  // Right motor forward with speed*SPEED_CONSTANT
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
  analogWrite(ENB, speed * SPEED_CONSTANT);
} else if (dir == R) {
  // Left motor forward with full speed
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  analogWrite(ENA, speed);
  // Right motor backward with full speed
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
  analogWrite(ENB, speed);
} else if (dir == BR) {
```

```
  // Left motor backward with full speed
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  analogWrite(ENA, speed);
  // Right motor backward with speed*SPEED_CONSTANT
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
  analogWrite(ENB, speed * SPEED_CONSTANT);
} else if (dir == B) {
  // Left motor backward with full speed
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  analogWrite(ENA, speed);
  // Right motor backward with full speed
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
  analogWrite(ENB, speed);
} else if (dir == BL) {
  // Left motor backward with speed*SPEED_CONSTANT
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  analogWrite(ENA, speed * SPEED_CONSTANT);
  // Right motor backward with full speed
```

```
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
  analogWrite(ENB, speed);
 } else if (dir == L) {
  // Left motor forward with full speed
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  analogWrite(ENA, speed);
  // Right motor backward with full speed
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
  analogWrite(ENB, speed);
 } else if (dir == FL) {
  // Left motor forward with speed*SPEED_CONSTANT
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  analogWrite(ENA, speed * SPEED_CONSTANT);
  // Right motor forward with full speed
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
  analogWrite(ENB, speed);
 } else if (dir == STOP) {
  // Left motor stop
```

```
    analogWrite(ENA, 0);
    // Right motor stop
    analogWrite(ENB, 0);
  }
}
```