Controlul prin SMS al unui robot

Miron Cezar Andrei

Pentru realizarea proiectului, am folosit următoarele componente :

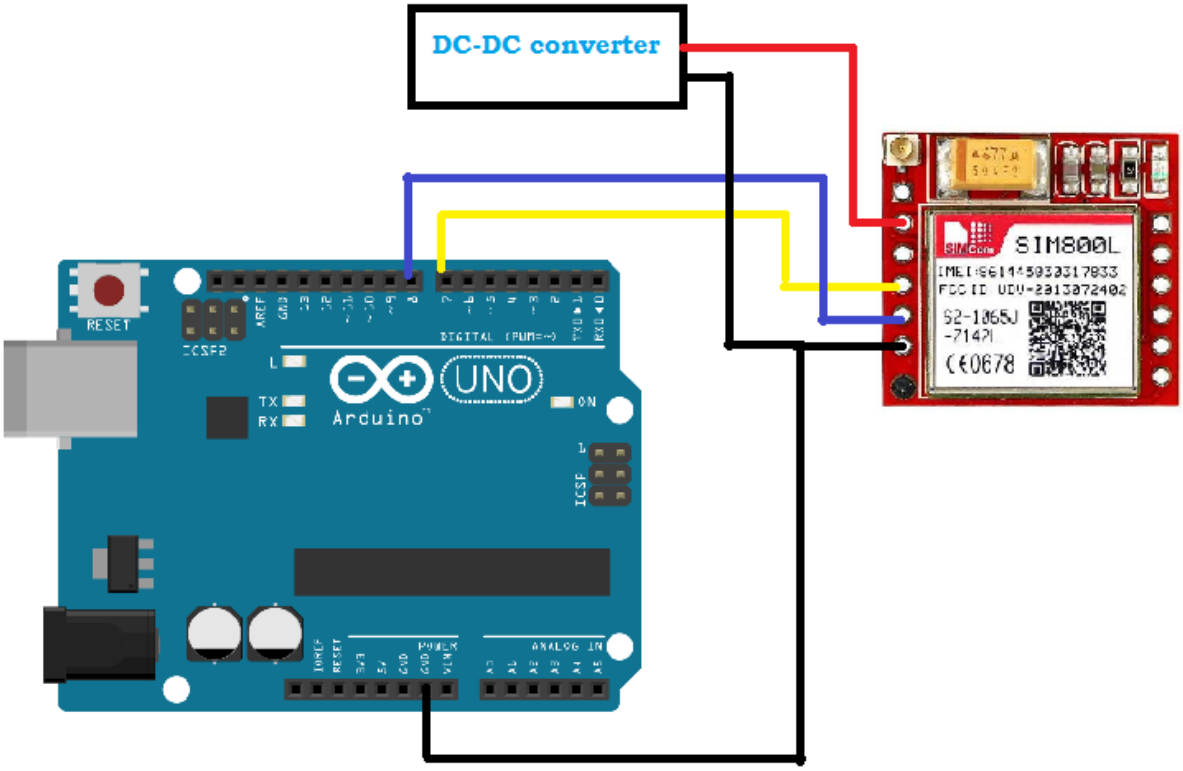
- Modul GSM – Sim800l
- Cutie de viteze dublă Tamiya
- Sursă reglabilă step down Lm2596
- Shield Ardumoto
- Arduino Uno
- 8 Baterii tip AA/R6
- Rezistențe, fire de conexiune și un condesator de 2200nF

Etape de proiectare :

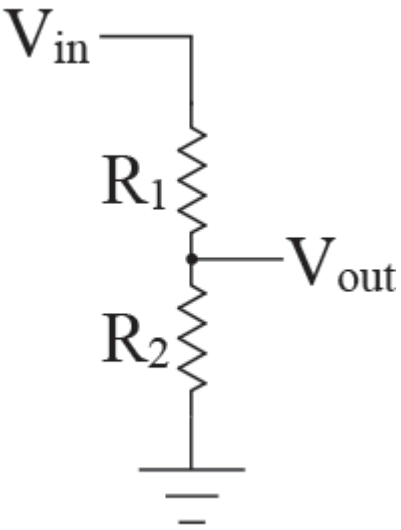
1. Comunicarea Modulului GSM cu placa de dezvoltare Arduino Uno

   Modului GSM (Sim800L) funcționează în parametrii optimi la o tensiune cuprinsă între 3.4V și 4.4V, avand nevoie (în anumite momente) de un curent mare. (2A). Avand în vedere aceste aspecte, pentru a conecta corect modului, am folosit o sursă reglabilă de tensiune (LM2596) pe care o alimentăm cu 4 baterii AA de 1.5V, iar output-ul sursei este de 4V.

   Pentru a asigura necesitatea de curent, intre 4V+ și GND-ul sursei (output) am pus un condensator de 2200nF ce suportă o tensiune maximă de 16V.



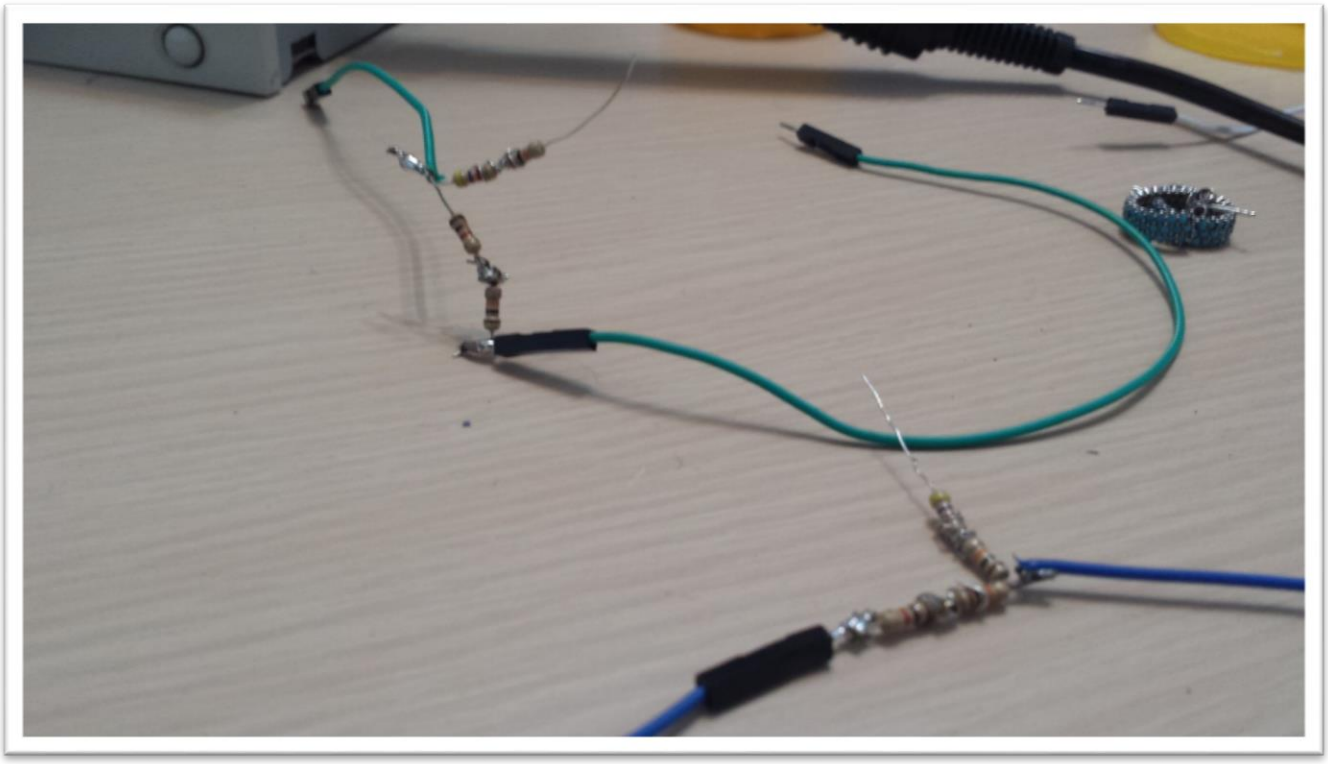
Nivelele logice ale modului GSM funcționează la 2.8. De aceea, am fost nevoiți să creăm un divizor de tensiune, ce limitează tensiunea la valoarea așteptată.



Unde R1 = 1100

R2 = 1470

Pentru un mai bun management al firelor, am ales să realizăm divizorul de tensiune direct la capetele firelor de legătura. Am înseriat 2 rezistețe de 1k, 10k și respectiv 10k, 4.7k.

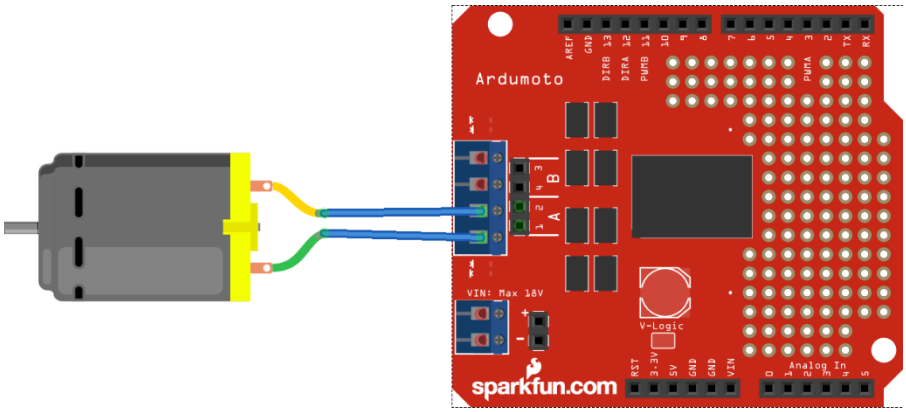


Pentru a verifica conexiunea și funcționalitatea modulului, ne-am folosit de bibilioteca SoftwareSerial și de comenzile AT disponibile în manualul modului GSM

Cele mai folosite comenzi:

Comandă	**Răspuns**
AT	Verificare conexiune
AT+CPIN	Pin cartela sim
AT+COPS	Verificare operator telefonie
AT+CSQ	Verificare semnal antenă
AT+CMEE	Activare mesaje erori
AT+CFUN	Resetare modul
AT+CNMI	Afișează pe serială ce primește modulul

2. Conectarea motoarelor cu Ardumoto



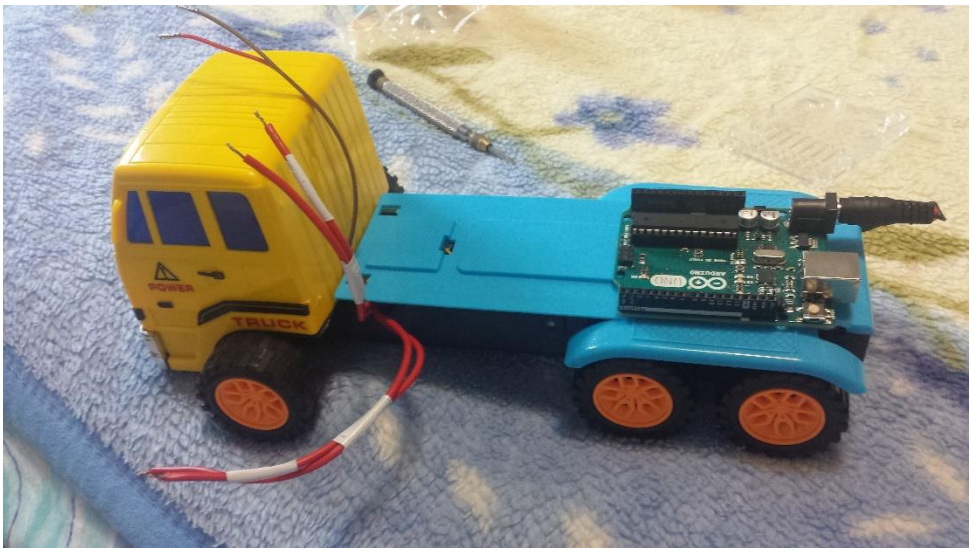
Am ales ca funcționarea mașinii să se poată face și prin alimentarea separată a shield-ului ardumoto (putere mare) dar și prin alimentarea shield-ului doar de la placuța Arduino Uno (putere mică).

Pentru că folosim pinul digital 11 pentru comunicarea modulului GSM cu arduino, am ales ca Ardumoto să fie poziționat independent față de arduino, interconectându-le prin fire și nu prin shield.

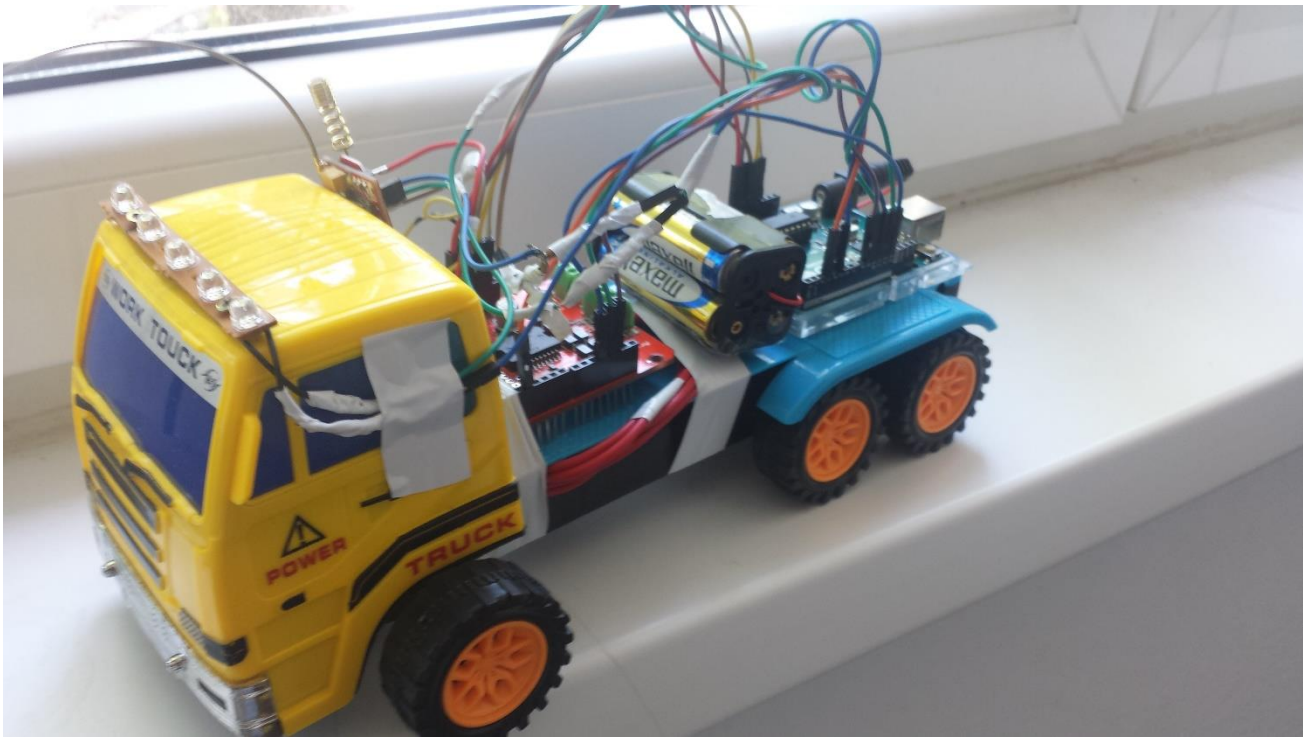
Verificarea motoarelor am realizat-o folosind exemplul de cod al celor de la sparkfun.

3. Integrarea tuturor componentelor și realizarea conexiunilor între ele

Pentru acest pas, am modificat o betonieră (jucărie).



Am integrat toate componentele și am ajuns la următorul rezultat :



4. Programarea robotului

Pentru a programa mașinuța, ne-am folosit de biblioteca SIM800L, disponibilă aici : https://github.com/cristiansteib/Sim800l

Principiu de funcționare :

Dacă robotul primește mesajul F urmat de o cifra de la 1 la 9, acesta se va deplasa în față un numar de secunde egal cu cifra introdusă. Similar pentru mesajul S precedat de o cifra, acesta va merge în spate. Celalalte alte mesaje vor fi ignorate.

```
#include <Sim800l.h>
#include <SoftwareSerial.h>//is necesary for the library!!

#define CW  0
#define CCW 1

#define MOTOR_A 0
#define MOTOR_B 1

Sim800l Sim800l;  //to declare the library
String text;     // to storage the text of the sms
uint8_t index = 1;   // to indicate the message to read.
char timp; //time in seconds
char stare; //state, F or S
String s_timp;
uint8_t i;


const byte PWMA = 3;  // PWM control (speed) for motor A
const byte PWMB = 5; // PWM control (speed) for motor B
const byte DIRA = 12; // Direction control for motor A
const byte DIRB = 9; // Direction control for motor B


void setup(){

  Serial.begin(9600); // only for debug the results .
  pinMode(LED_PIN, OUTPUT);
  Sim800l.begin(); // initializate the library.
```

```
  Sim800l.delAllSms();
  Serial.println(text);
  setupArdumoto();

}

void loop(){

   digitalWrite(LED_PIN, LOW);

  text=Sim800l.readSms(index);

  if (text.indexOf("OK")!=-1) {

     text.toUpperCase();
     Serial.println(text);

     if(text.indexOf("F") != -1) {

        digitalWrite(LED_PIN, HIGH);

        index = text.indexOf("F");
        stare = text[index];
        timp = text[index+1];
        Serial.print(stare);
        Serial.println(timp);

        s_timp = timp;
        s_timp = s_timp  + "000";
```

```
    driveArdumoto(MOTOR_A, CW, 127);  // Motor A at max speed.
    driveArdumoto(MOTOR_B, CW, 127);  // Motor B at max speed.

    //Serial.println(s_timp);
    delay(s_timp.toInt());
    Serial.println("done_delay");

    stopArdumoto(MOTOR_A);
    stopArdumoto(MOTOR_B);

    digitalWrite(LED_PIN, LOW);
}

else if(text.indexOf("S") != -1) {

    index = text.indexOf("S");
    stare = text[index];
    timp = text[index+1];
    Serial.print(stare);
    Serial.print(timp);

    s_timp = timp;

    driveArdumoto(MOTOR_A, CCW, 127);  // Motor A at max speed.
    driveArdumoto(MOTOR_B, CCW, 127);  // Motor B at max speed.

    for(i = 0; i<= s_timp.toInt(); i++)
    {
```

```
          digitalWrite(LED_PIN, HIGH);
          delay(500);
          digitalWrite(LED_PIN, LOW);
          delay(500);

        }

        stopArdumoto(MOTOR_A);
        stopArdumoto(MOTOR_B);

        digitalWrite(LED_PIN, LOW);

     }

        else{
          Serial.println("Not Compatible ...sorry.. :D");
         }
     Sim800l.delAllSms();
     index = 1;

   } //ceva
}

void stopArdumoto(byte motor)
{
  driveArdumoto(motor, 0, 0);
}

// setupArdumoto initialize all pins
```

```
void setupArdumoto()
{
  // All pins should be setup as outputs:
  pinMode(PWMA, OUTPUT);
  pinMode(PWMB, OUTPUT);
  pinMode(DIRA, OUTPUT);
  pinMode(DIRB, OUTPUT);

  // Initialize all pins as low:
  digitalWrite(PWMA, LOW);
  digitalWrite(PWMB, LOW);
  digitalWrite(DIRA, LOW);
  digitalWrite(DIRB, LOW);
}

// driveArdumoto drives 'motor' in 'dir' direction at 'spd' speed
void driveArdumoto(byte motor, byte dir, byte spd)
{
  if (motor == MOTOR_A)
  {
    digitalWrite(DIRA, dir);
    analogWrite(PWMA, spd);
  }
  else if (motor == MOTOR_B)
  {
    digitalWrite(DIRB, dir);
    analogWrite(PWMB, spd);
  }
}
```