Laborator 6

SISTEME INFORMATIZATE DE BORD CU TRANSFER DE DATE UTILIZÂND PROTOCOLUL CAN

1. Scurt istoric

În industria de automobile la ora actuală se utilizează o magistrală serială utilizată cu scopul de a asigura comunicarea între mai multe microcontrolere fără utilizarea unui calculator-gazdă. Acesta magistrală se numește CAN (Controller Area Network).

Magistrala a fost dezvoltată inițial în 1983 de către firma Bosh, dar a fost lansată prima versiune (versiunea CAN 1.2) în anul 1986. A fost înregistrată și standardizată sub denumirea ISO 11898. Ulterior diverși producători de componente electronice printre care Intel, Philips, Infineon, Texas Instruments. Motorola au început să producă periferice ce aveau implementat protocolul CAN. A doua versiune a protocolului fost lansată în 1991 sub denumirea de CAN 2.0.

Diferența majoră între cele două versiuni ale protocolului constă în creșterea domeniului de adresare a nodurilor. Mai exact, CAN 1.2 definește doar un singur tip de mesaj (mesaj standard) având lungimea câmpul de identificare a nodului (Id) de 11 biți, pe când versiunea CAN 2.0 mai introduce, pe lângă tipul de mesaj definit anterior și un mesaj cu lungimea Id-ul de 29 de biți numit mesaj extins [1].

În anul 2003 standardul ISO 11898 apar o serie de versiuni ale standardului ISO 11898-1, și ISO 11898-2. În anul 2012 Bosh a lansat versiunea CAN FD 1.0 care are rată flexibilă de date. În anul 2015 se standardizează protocolul CAN FD sub denumirea ISO 11898-1. În anul 2016 s-a modificat nivelul fizic pentru a suporta viteze de transfer de date până la 5 Mb/s și a fost standardizat sub denumirea de ISO 11898-2. Astăzi, protocolul CAN este un echipament standard în toate vehiculele (mașini, camioane, autobuze, tractoare) - precum și nave, avioane, mașini industriale și multe altele.

Avantajele versiunii CAN FD:

- permite transmiterea datelor până la 8 Mbit / s - cu mult peste 1 Mbit / s din CAN classical

- permite pachete de date de 64 de octeți (în loc de 8 octeți), îmbunătățind eficiența protocolului [2].

2. Noțiuni teoretice despre magistrala CAN

Schema generală a unei rețele CAN este prezentată în figura 1.

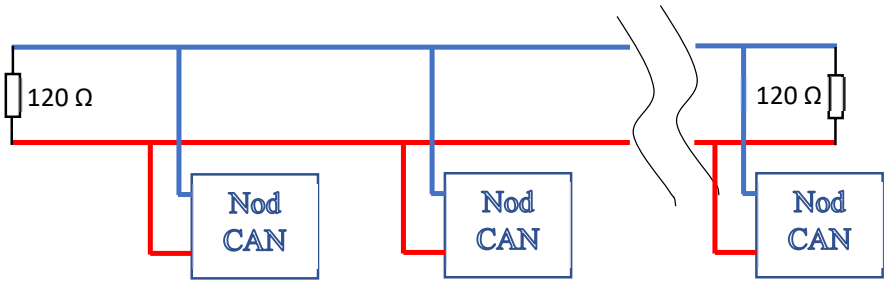

Fig. 1. Schema generală a unei rețele CAN.

Protocolul magistralei CAN

Concepte CAN de bază

Conform cu modelul OSI, protocolul CAN are multiple nivele:

1. Nivel aplicație.
2. Nivel obiect:
   - manipulare stări şi mesaje;
   - filtrare mesaje.
3. Nivel transfer:
   - confirmare mesaje;
   - detectare şi semnalare erori;
   - viteza de transfer şi sincronizare;
   - limitare erori;
   - arbitrare;
   - încadrare mesaje;
   - validare mesaje.

Nivelul de transfer constituie nucleul CAN. El prezintă mesajele recepţionate nivelului obiect şi acceptă de la acesta mesajele de transmis. Nivelul de transfer este responsabil cu detectarea ,sincronizarea biţilor, confirmări,încadrarea mesajelor, arbitrări, semnalarea şi limitarea erorilor.

4. Nivel fizic:
   - nivel semnal şi reprezentare biţi;
   - mediu de transmisie.

Din punct de vedere software, datele CAN sunt simple. Interfețele fizice CAN și transmițătorul (transceiver) hardware gestionează automat protocoalele de transmisie și recepție CAN la nivel de biți. Acest lucru lasă software-ul de nivel superior doar să gestioneze identificatorul mesajului, lungimea datelor și octeții de date. Acestea sunt deseori expuse programelor prin Interfața de Programare a Aplicației (API) a unui dispozitiv CAN conectat la portul USB al unui computer. În mod alternativ, un program terminal sau un software personalizat citește datele CAN dintr-un port serial folosind un dispozitiv de interfață CAN, inclusiv interfețele CAN bazate pe Bluetooth și WiFi.

Pachetul de mesaje CAN constă din:

- Identificator 0-2047 (11 biți) standard sau 0 - 2^{29} -1 (29 biți) extins
- Codul lungimii datelor (Data Length Code DLC) 0 până la 8 pentru numărul de octeți de date prezenți sau pentru CAN FD valoarea 8 = 8 octeți, 9 = 12 octeți, 10 = 16 octeți, 11 = 20, 12 = 24, 13 = 32, 14 = 48 și 15 = 64 octeți
- Bytes de date de la 0 la 8, sau pentru CAN FD de la 0 la 64 (folosind lungimi definite de DLC).

Diagrama CAN Standard este dată în figura 2.

Spațiu Inter-Cadre	**Cadrul de Date**							Spațiu Inter-Cadre
	Începutul cadrului	Câmpul de Arbitrare	Câmpul de Control	Câmpul de Date	Câmpul CRC	Câmpul de ACK	Sfârșitul Cadrului	

Fig. 2. Diagrama CAN Standard.

Unde

Începutul cadrului - marchează începutul unui cadru de date sau de cerere de date și va consta într-un singur bit "dominant" (de nivel 0 logic). Pentru a începe o transmisie, un nod trebuie să verifice dacă magistrala este în starea inactiv (**bus idle**). În acest cadru celelalte noduri se vor sincroniza la apariția bitului de start pe frontul crescător.

Câmpul de arbitrare – pentru un cadru standard este compus din câmpul de Identificare a nodului de 11 biți, (**Id**) și din bitul de Remote Transmission Request (**RTR)**.

Câmpul de control - este format din șase biți dintre care primii doi biți sunt rezervați (r0 și r1) iar restul de 4 biți formează Codul de Lungime a Datelor (**Data Length Code**, DLC).

Câmpul de CRC - este format din două părți:

a. Secventa de CRC - câmp de 15 biți ce reprezintă restul împărțirii șirului de biți format din câmpurile SOF, arbitrare, control și date la polinomul-generator CRC: X^15 (+) X^14 (+) X^10 (+) X^8 (+) X^7 (+) X^4 (+) X^3 (+) 1 (cod BCH)
b. Delimitatorul de CRC - format dintr-un singur bit "recesiv"

Câmpul de Confirmare (ACK) - are lungimea de 2 biți și constă:

a. în slotul de ACK (ACK Slot)
b. delimitatorul de ACK (ACK Delimiter) de valoare "recesiv" (bit 1).

Sfârșitul cadrului - (EOF) este compus din șapte biți recesivi și este specific cadrelor de Date și de Cerere de date [3].

CAN_HS poate avea viteza de transfer a datelor de 125, 250, 500 sau 1000 kb/s. Datorită vitezei mari de transfer a datelor este utilizat cu precădere pentru motor, cutie de viteze și sistemele de siguranță activă (ABS, ESP).

CAN_LS are viteza de transfer între 40 și 125 kbps. Protocolul CAN LS are avantajul că este tolerant la erori (fault tolerant). În cazul în care unul din cele două fire este întrerupt comunicația se realizează pe un singur fir. Acest tip de protocol CAN este utilizat cu precădere la închiderea centralizată și la imobilizator, datorită funcționării și în regim de avarie.

Lungimea maximă a magistralei poate să fie de 250 m (CAN_HS) sau de 50 m (CAN_LS). Numărul de calculatoare care pot fi conectate la magistrală variază în funcție de viteza și de numărul parametrilor ce trebuie transmiși. O rețea CAN poate suporta până la 50 de calculatoare interconectate. În capetele magistralei sunt prevăzute rezistențe electrice de aproximativ 120 Ω care au rolul de a crește impedanța rețelei, în scopul eliminării fenomenului de reflexie a semnalelor.

Magistrala CAN conține două fire numite CAN_H (High voltage) și CAN_L (Low voltage). Pe firul CAN_H tensiunea electrică poate avea două niveluri: 2.5 și 3.5 V. Pe firul CAN_L tensiunea electrică poate fi de 1.5 și 2.5 V. Aceste semnale sunt prezentate în figura 3.

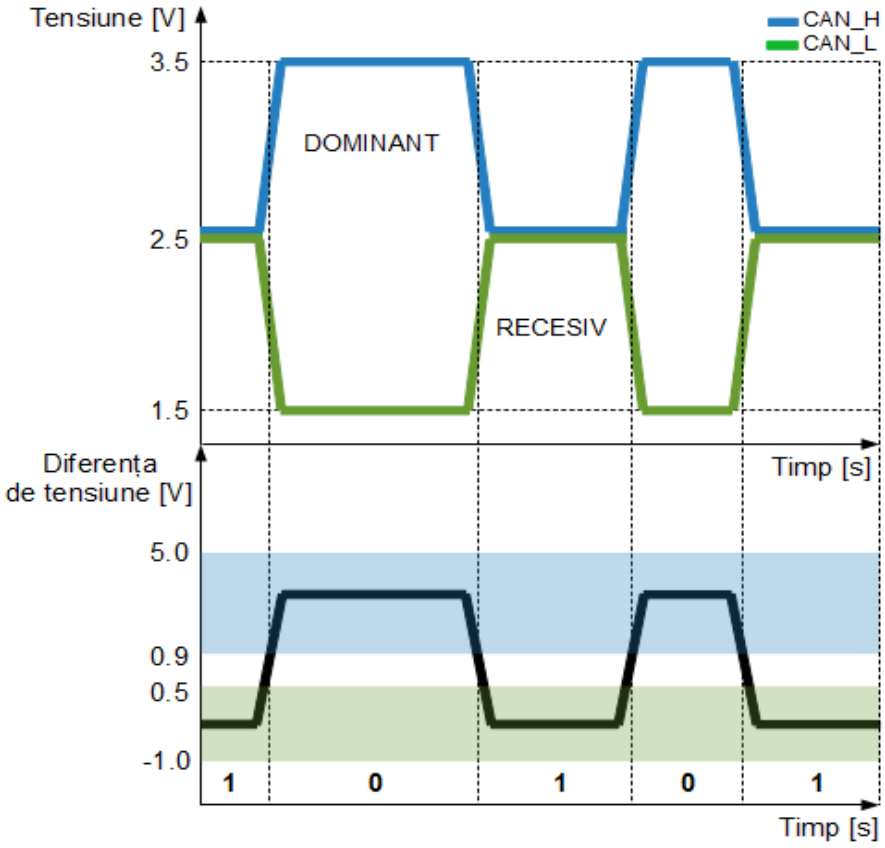

Fig. 3. Semnalele electrice utilizate pe magistrala CAN.

Semnalele de tensiune pe cele două fire au ambele valoarea minimă de 2.5 V sau valoarea maximă 3.5 V pe CAN_H și valoarea minimă 1.5 V sau valoarea maximă 2.5 V pe CAN_L. Traducerea acestor valori de tensiune în semnal digital se face prin diferența celor două tensiuni. Când tensiunea pe cele două fire este de 2.5 V (deci pe CAN_H are valoare minimă iar CAN_L are valoare maximă) diferența este de 0 V, când cele două tensiuni au 3.5 și 1.5 V, diferența este de 2 V. Semnalul de tensiune ce are valori de 0 și 2 V reprezintă valori digitale de 0 și 1.

Atenție! Cele două valori digitale nu sunt reprezentata exact de valori fixe de tensiune. Datorită eventualelor perturbații aceste valori pot varia între anumite limite. Astfel, valoarea digitală de 0 poate fi reprezentată de o tensiune între -1.0 și 0.5 V iar valoarea digitală 1 înseamnă o tensiune între 0.9 și 5.0 V [4].

Atenție: Să nu se facă confuzie între CAN HS (High Speed) și CAN_H (High voltage). Primul reprezintă viteza de transfer a datelor iar a doua tensiune electrică din fir. Aceeași observație este valabilă și pentru CAN LS (Low Speed) și CAN_L (Low voltage). Ambele versiuni de viteză conțin cele două fire CAN_H și CAN_L!

Schema conexiunilor între placa de dezvoltare Arduino UNO și modulul CAN 2515, precum și între cele două module CAN este prezentată în figura 4.

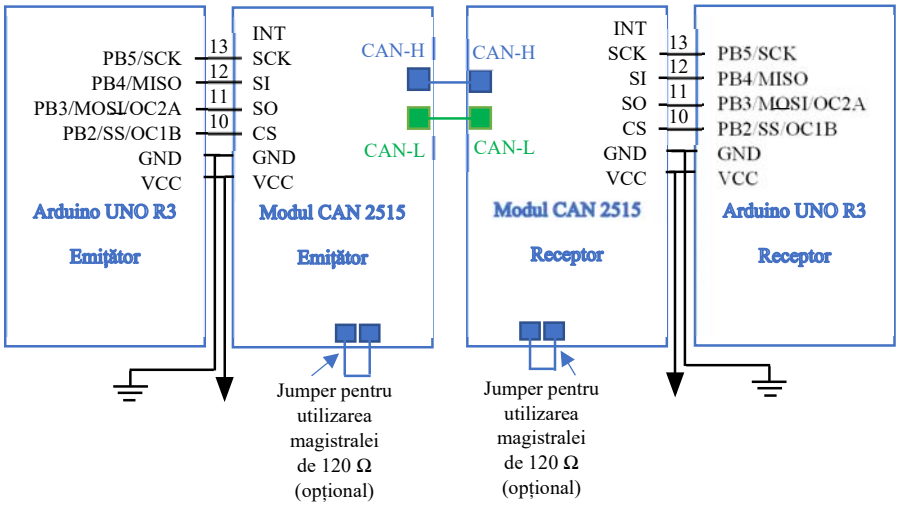

Fig. 4. Conexiunile între placa de dezvoltare Arduino UNO și modulul CAN 2515.

Pentru afișarea datelor transmise pe interfața CAN la modulul receptor se va atașa un ecran alfanumeric 2X16 caractere.

Schema conexiunilor dintre modulul Arduino UNO R3 și ecranul alfanumeric este prezentată în figura 5. În codul scris pentru receptor se va include librăria <LiquidCrystal.h> [5] necesară display-ului. De asemenea se vor declara pinii display-ului împreună cu conexiunea corespunzătoare de la placa de dezvoltare Arduino UNO R3 ca în exemplul următor:

```
const int rs = 7, en = 6, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7); //Definirea pinilor
                                           //RS, E, D4, D5, D6, D7
```

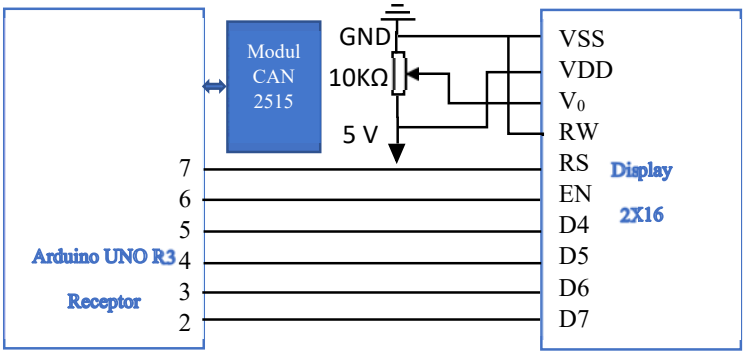

Fig. 5. Schema conexiunilor dintre modulul Arduino UNO R3 și ecranul LCD alfanumeric.

Modulul CAN 2515 este prezentat în figura 6 cu librăria <mcp2515.h> [6], iar funcțiile pinilor sunt prezentate în tabelul 1.

Numele pinului	Utilizare
VCC	Pin de alimentare 5V
GND	Pin de masă
CS	Pin de selecție SPI SLAVE (Active low)
SO	Intrare master SPI, ieșire slave
SI	Ieșire master, intrare slave SPI
SCLK	Pin de tact SPI
INT	Pin de întrerupere MCP2515

Tabel 1. Funcțiile pinilor modulului CAN MCP2515.

Fig. 6. Prezentarea modulului CAN MCP2515.

În cele ce urmează sunt prezentate exemple de cod pentru modulul transmițător respectiv modulul receptor. În acest exemplu se va transmite prin interfața CAN 2515 temperatura măsurată cu ajutorul unui senzor analogic de tip LM 35.

În figura 7 se prezintă montajul ce trebuie realizat conform schemelor din figurile 4 și 5.

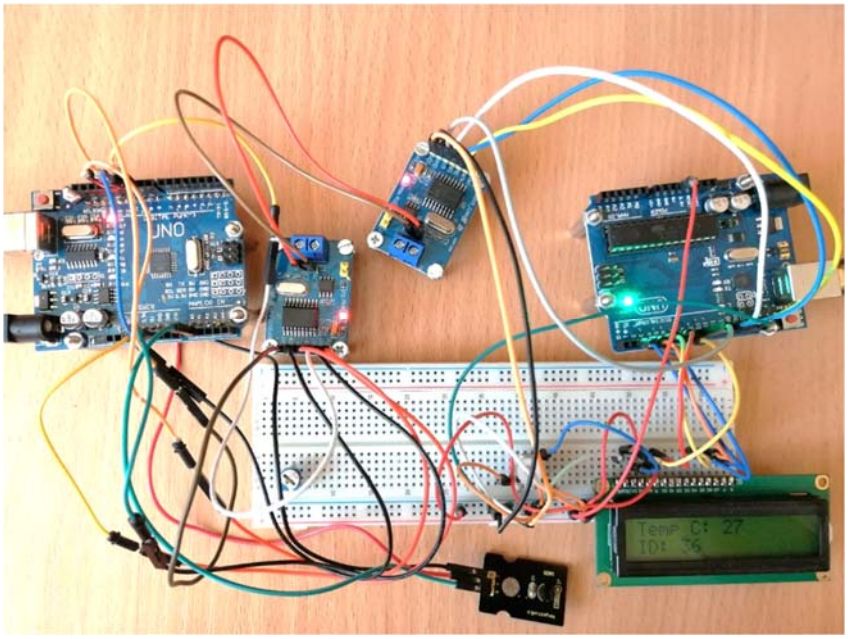

Fig. 7. Realizarea montajului cu ajutorul breadboard-ului.

În figura 8 se poate vedea mesajul ce apare pe ecranul LCD în momentul inițializării (pornirii sau resetării) montajului. Acest text este scris și se poate modifica din codul corespunzător receptorului.

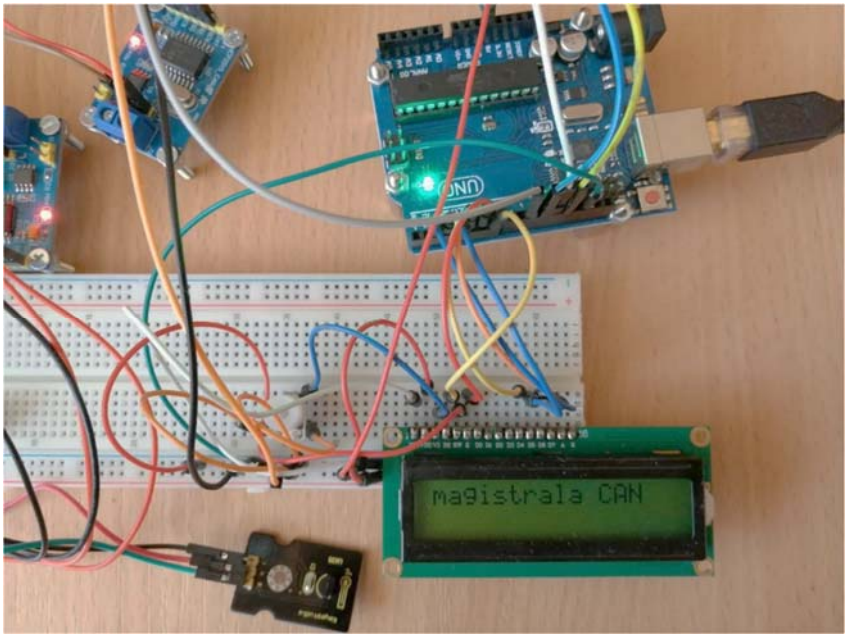

Fig. 8. Mesajul ce apare pe afișajul LCD în momentul inițializării.

După inițializare dacă nu au apărut erori, montajul emițător va începe transmiterea datelor, iar receptorul va primi informațiile transmise prin interfața CAN.

În figura 9 este prezentat montajul în momentul în care se recepționează informația. În acest caz se transmite ID-ul transmițătorului și temperatura măsurată de către senzorul LM 35.

În codul transmițătorului s-a setat modulul CAN ca având adresa ID 36. Ea se poate modifica din cod. Fiecare modul CAN va avea o adresă astfel încât modulul receptor (în cazul autoturismelor, ECU, unitatea centrală) să știe de unde și ce informație primește pentru a putea fi prelucrată corespunzător și a lua decizia corectă. În cazul nostru dacă temperatura este mai mare decât cea permisă, avaria va fi semnalizată prin indicatorul luminos din bordul autovehiculului.

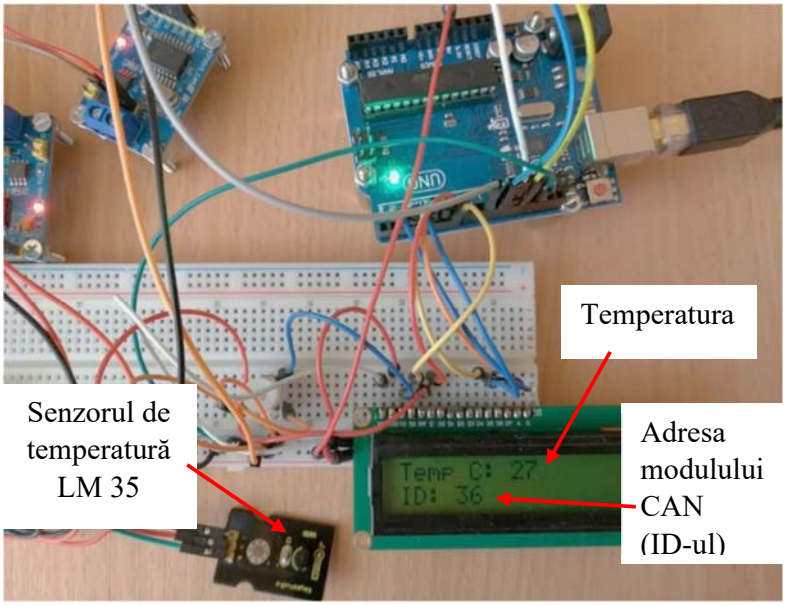

Fig. 9. Mesajul ce apare pe ecranul LCD în timpul transmiterii informației (temperatura de 27 grade Celsius și adresa modulului CAN care este 36).

În figura 10 se prezintă simularea modificării temperaturii prin atingerea cu mana a senzorului de temperatură.

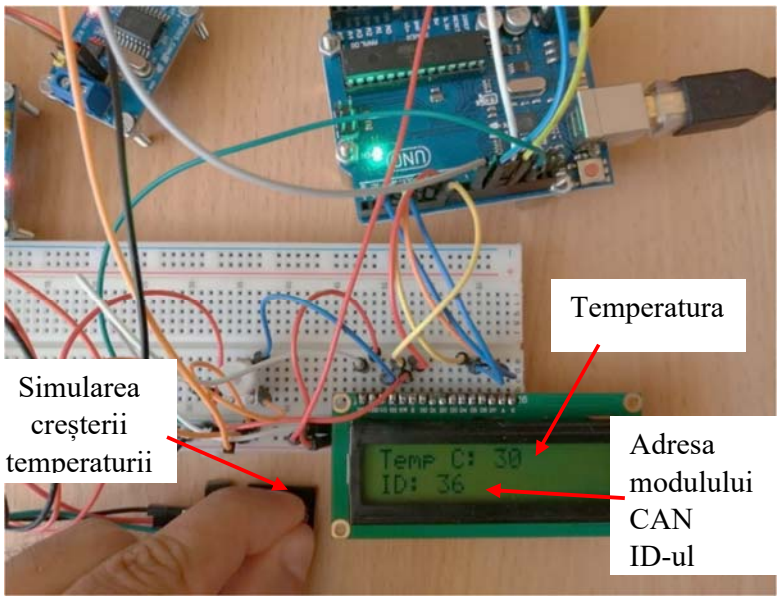

Fig. 10. Mesajul ce apare pe ecranul LCD cu modificarea temperaturii (temperatura de 30 grade Celsius și adresa modulului CAN care este 36).

1. Codul pentru transmițător

```
//Cod pentru transmițător CAN:

#include <SPI.h>//Librăria necesară folosirii protocolului de comunicație SPI
#include <mcp2515.h>//Librăria pentru protocolul de comunicație CAN

// declararea variabilelor
const int sensor=A1;//Setarea pinului A5 ca intrare pentru senzor LM 35
float tempc; //variablă pentru stocarea valorii temperaturii in grade Celsius
float tempf; //variabilă pentru stocarea temperaturii în grade Fahreinheit
float vout; //variabilă pentru stocarea valorii tensiunii de ieșire de la LM 35

struct can_frame canMsg; //declararea mesajului transmis prin interfața CAN
//ca fiind de tip structura (struct)
MCP2515 mcp2515(10); //Setarea pinului 10 al modulului Arduino UNO R3
//ca fiind pinul de selecție (CS) pentru modulul CAN MCP 2515

void setup() //funcție pentru setarea porturilor
{
  while (!Serial); //se așteaptă conexiunea pe portul serial
  Serial.begin(115200); //se setează viteza portului serial la 115200 baud rate
  pinMode(sensor,INPUT); // Configurarea pinului definit anterior “sensor”
//ca intrare
  SPI.begin(); //Inițializarea comunicației pe SPI

  mcp2515.reset(); //resetarea modulului CAN
```

```
 mcp2515.setBitrate(CAN_500KBPS,MCP_8MHZ);  //Setarea modulului
//CAN la viteza de 500KBPS și clock 8MHz
  mcp2515.setNormalMode();//configurarea modulului pentru modul de lucru
//normal
}

void loop() //funcția principală ce va fi executată de microcontroler
{
vout=analogRead(sensor); //Se citește valoarea de la senzor
vout=(vout*500)/1023;//se face scalarea la 5V și la numărul de biți ai
//convertorului analog numeric
tempc=vout; //Memorarea valorii în grade Celsius
tempf=(vout*1.8)+32; // Convertirea în grade Fahrenheit

  canMsg.can_id  = 0x036; //Setarea adresei modulului CAN ID 0x036
  canMsg.can_dlc = 8; //setarea lungimii datelor la 8 caracter
  canMsg.data[0] = tempc;//în caracterul [0] se trimite valoarea în gr. Celsius
  canMsg.data[1] = tempf;//în caracterul [1] se trimite valoarea în gr.
                         //Fahrenheit
  canMsg.data[2] = 0x00; //pentru restul caracterelor (de la 2 la 8) se trimite
                        //valoarea 0
  canMsg.data[3] = 0x00;
  canMsg.data[4] = 0x00;
  canMsg.data[5] = 0x00;
  canMsg.data[6] = 0x00;
  canMsg.data[7] = 0x00;
```

```
  mcp2515.sendMessage(&canMsg);     //Se trimite mesajul CAN
  Serial.println(tempc); //se trimite valoarea în grade Celsius și pe portul serial
                         //pentru monitorizare pe linie noua
  Serial.println(canMsg.can_id); ); //se trimite valoarea ID a modulului pe
                                    //portul serial pentru monitorizare pe linie noua
  Serial.println(tempf); //se trimite valoarea în grade Fahrenheit și pe portul
                         //serial pentru monitorizare pe linie noua
  delay(1000); // se adaugă o întârziere de 1000 milisecunde
}// se închide bucla loop
```

2. Codul pentru receptor

```
//Codul pentru modulul receptor:

#include <SPI.h>            //Librărie pentru comunicație pe SPI
#include <mcp2515.h>        // Librărie pentru comunicație CAN
#include <LiquidCrystal.h>  //Librărie pentru utilizarea display-ul LCD

const int rs = 7, en = 6, d4 = 5, d5 = 4, d6 = 3, d7 = 2; // declararea pinilor
//pentru displayul LCD
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);  //Definirea pinilor pentru display-ul
//LCD (RS, E, D4, D5, D6, D7)

struct can_frame canMsg; //declararea mesajului transmis prin interfața CAN
//ca fiind de tip structura (struct)
```

```
MCP2515 mcp2515(10);  //Setarea pinului 10 al modulului Arduino UNO R3
//ca fiind pinul de selecție (CS) pentru modulul CAN MCP 2515



void setup() { //funcția pentru inițializare
  lcd.begin(16,2); //Setarea display-ului ca fiind de tip 16x2 caractere
  lcd.setCursor(0,0); //Setarea cursorului pe coloana 0 și linia 0
  lcd.print("magistrala CAN");//afișarea mesajului dintre ghilimele începând
//cu pozițiile setate anterior
  lcd.setCursor(0,1);// setare cursorului pe poziția coloana 0 și linia 1
   delay(3000);//se adaugă o întârziere de 3000 ms
  lcd.clear();//se șterge tot de pe ecran


  SPI.begin();//Inițializarea comunicației SPI


  Serial.begin(115200);//Inițializarea comunicației pe portul serial la viteza de
//115200 baud rate
    mcp2515.reset(); //resetarea modulului CAN
  mcp2515.setBitrate(CAN_500KBPS,MCP_8MHZ);  //Setarea  modulului
CAN la viteza de 500KBPS și clock 8 MHz
  mcp2515.setNormalMode(); //Setarea modulului CAN în mod normal
}


void loop() //funcția principală ce va fi executată
{
  if  (mcp2515.readMessage(&canMsg)  ==  MCP2515::ERROR_OK)  //Se
//citesc datele de pe portul serial (prin sondaj, Poll Read)
```

```
  {
    int x = canMsg.data[0];//se declară variabila x în care se va stoca caracterul
//de pe poziția [0]
    int y = canMsg.data[1]; ];//se declară variabila y în care se va stoca
caracterul //de pe poziția [0]
    int z; // declararea variabilei z de tip intreg

     lcd.setCursor(0,0); //setarea cursorului pe coloana 0 și linia 0
     lcd.print("Temp C: "); //afișarea pe display a textului dintre ghilimele
     lcd.print(x);//afișarea pe display a valorii stocate în variabila x
     lcd.setCursor(0,1); //setarea cursorului pe coloana 0 și linia 1
 /*   lcd.print("Temp F: ");//afișarea pe display a textului dintre ghilimele
     lcd.print(y); //afișarea pe display a valorii stocate în variabila y */
     lcd.print("ID: ");//afișarea pe display a textului dintre ghilimele
     lcd.print(canMsg.can_id, HEX);//afișarea pe display a adresei ID a
//modulului CAN 2515, în format HEX
//trimiterea pe portul serial a valorilor și afișarea lor în mediul de dezvoltare
//ARDUINO
     Serial.println(x);
     Serial.println(y);
     Serial.println(canMsg.can_id, HEX);
     delay(1000);//se adaugă o întârziere de 1000 milisecunde
     lcd.clear(); // se șterge tot ce este afișat pe display pentru o noua recepție
//a datelor
   }
}
```

3. Lucrarea practică

1. Se va studia programul.
2. Se va adăuga un nou senzor LM 35 și se va adapta programul pentru a trimite valoarea temperaturii pe poziția 6 în grade Celsius și pe poziția 7 în grade Fahrenheit
3. Se va schimba adresa ID a dispozitivului în 0x042 și se vor trimite datele sub această adresă.
4. Se va adăuga hardware și software un nou modul CAN cu adresa ID 0X062 și se va transmite temperatura de la un alt treilea senzor LM 35 la modulul receptor pe poziția opt a mesajului.

Bibliografie

[1] https://ro.wikipedia.org/wiki/Controller_Area_Network
[2] https://www.csselectronics.com/screen/page/simple-intro-to-can-bus/language/en#CAN-Bus-Intro-Dummies-Basics
[3] https://ro.wikipedia.org/wiki/Controller_Area_Network#Arhitectura_re%C8%9Belei_CAN
[4] http://www.e-automobile.ro/categorie-electronica/74-protocol-can-auto.html
[5] librăria <LiquidCrystal.h>
[6] librăria <mcp2515.h>