TELEGRAF VIRTUAL

UTILIZAND NI DAQ MX SI LABVIEW

Codreanu Dan

Cuprins:

1. Documentare privind partea hardware

Unul dintre cele mai vechi şi cele mai cunoscute coduri din lume a fost creat în urmă cu 187 de ani pentru transmiterea informaţiei cu ajutorul telegrafului electric. Invenţia i-a adus creatorului ei, Samuel Morse, o avere uriaşă pe care acesta a împărţit-o artiştilor săraci şi talentaţi, el însuşi fiind la bază pictor.

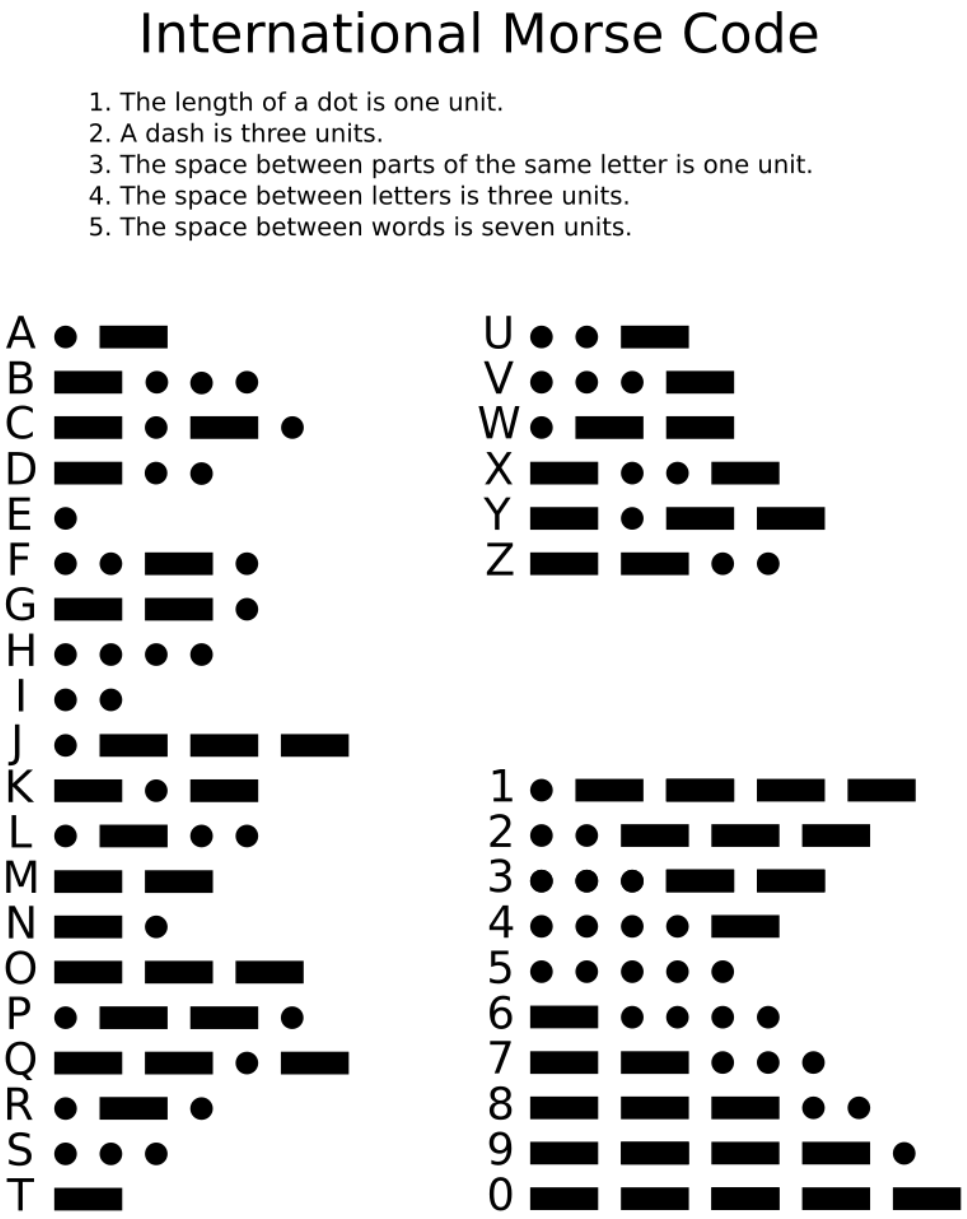
Inventat la mijlocul anilor 1830, Codul Morse a avut cea mai lungă viaţă dintre toate sistemele electronice de codificare, fiind folosit până în anul 1997. Aşa cum este definit astăzi, Alfabetul sau Codul Morse este o metoda de transmitere a informatiei folosind secvente standardizate de semne sau pulsaţii scurte şi lungi cunoscute în mod comun ca puncte şi linii - pentru litere, cifre şi caracterele speciale specifice oricarui mesaj.

Iniţial, codul Morse a fost folosit pentru a transmite informaţia prin telegraf. Când codul a fost adaptat pentru comunicarea radio, oamenii şi-au dat seama că ar putea învăţa mai uşor codul Morse ca o limbă auzită decât o limbă scrisă. Pentru a reproduce sunetele scoase de un receptor Morse, operatorii au vocalizat punctul ca "dit", şi linia ca "dah". Punctele ce nu se aflau la finalul unei secvenţe, au devenit "di" (figura 2).

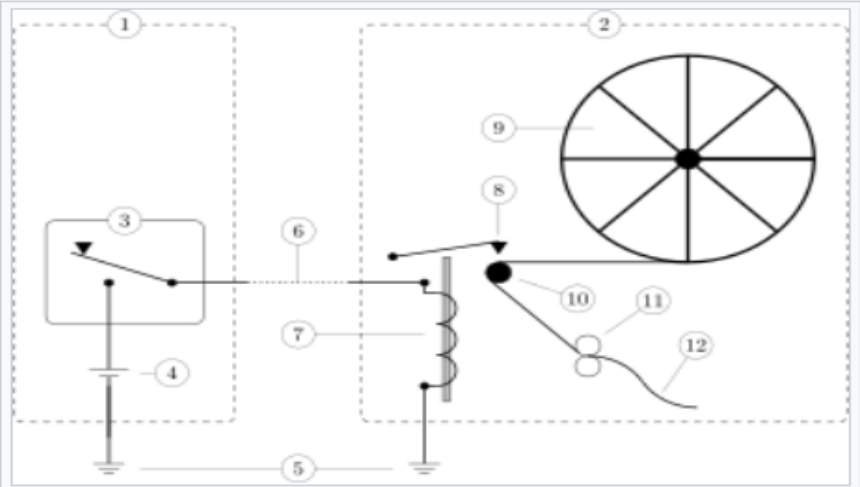
În jurul anului 1920, codul a început să fie folosit ca bază de comunicare în aviaţie. Începând cu 1930, piloţilor militari dar şi celor civili li se cerea să cunoască codul Morse. Codul Morse a fost vital în cel de-al doilea Război Mondial, pentru transmiterea de mesaje între nave de război şi bazele navale ale Marinei Regale Britanice, Marinei imperiale japoneze,canadiene , ausraliene şi americane.

Codul Morse a fost folosit până în 1999, când a fost înlocuit de către GMDSS (sisteme de comunicare marină de mare frecvenţă). Când Marina franceză a încetat folosirea codului Morse în 1997, ultimul mesaj transmis a fost: "*Către toţi. Acesta e strigătul ultim înainte de tăcerea noastră veşnică.*"[1]

Codul Morse sau alfabetul Morse este o metodă de transmitere a informației folosind secvențe standardizate de semne sau pulsații scurte și lungi - cunoscute în mod comun ca „puncte” și „linii” - pentru litere, cifre și caracterele specialespecifice oricărui mesaj (figura 1).



Fig, 1 Codifcarea caracterelor in cod Morse



Telegraf (schemă): 1.stația de transmisie, 2.stația de recepție, 3.manipulator, 4.baterie (sursă de curent), 5.împământare (masă), 6.linie de telegraf (cablu), 7.electromagnet, 8.știft de imprimare, 9.rolă cu banda de hârtie, 10.cilindru de imprimare, 11.role de avans, 12.banda de hârtie imprimată.

Fig. 2 Telegraf – schema de principiu

Cum functioneaza un telegraf? Un telegraf functioneaza prin transmiterea semnalelor electrice prin fire. Un telegraf are atat un transmitator, cat si un receptor. Transmitatorul este telegraful sau cheia de transmisie.

Firele conecteaza transmitatorul si receptorul. Aceste fire formeaza un circuit in serie. Curentul electric este furnizat de o baterie. Butonul de pe cheia telegrafului actioneaza ca un intrerupator.

Cand intrerupatorul este apasat, se face contact cu baza si circuitul se inchide. Curentul electric poate apoi sa circule spre receptor. Cand butonul este eliberat, intrerupatorul se deschide si circuitul se intrerupe.

Receptorul contine un electromagnet. Cand electromagnetul primeste un impuls de electricitate, acesta misca un brat conectat la un cilindru cu cerneala. Cilindrul marcheaza o banda de hartie. (figura 3) [2]

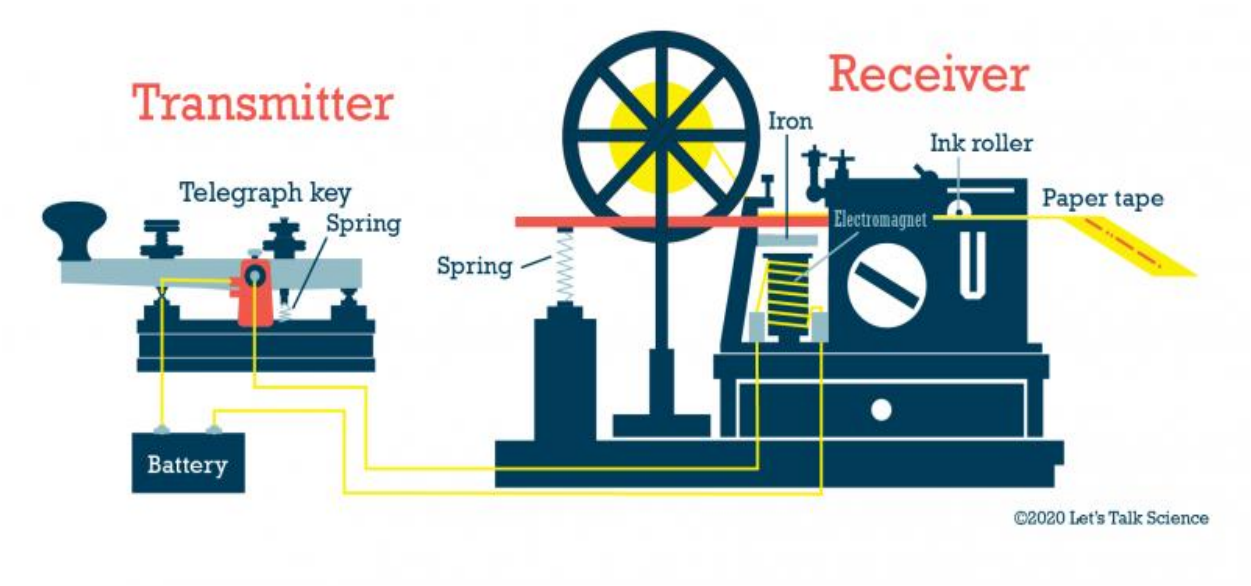


Fig. 3 Telegraf – trasnmitator si receptor

Cerinta proiect:

"Generator si decodificator de semnale Morse cu cartela de achizitii de date"

Abordarea aleasa pentru dezovoltarea acestui proiect este realizarea unui instrument de telegrafie tip teleprinter, pentru partea de ***trasnsmisie*** si afisare a caracterelor codificare in cod Morse pentru partea de ***receptie***.

Principiu de functionare:

Modul de lucru pentru acest dispozitiv va fi urmatorul:

1. Intrumentul virtual de trasnmisie va citi mesajul de input ce este sub forma de string si il va converti fiecare caracter sub forma de bool, in concordanta cu lungimea unei unitati de timp stabilite ca unitate de baza.

Legend

Parametru	Valoare
Durata DOT (•)	0.1 s (100 ms)
Durata DASH (–)	0.3 s (3 DOT)
Pauza simboluri	0.1 s
Pauza litere	0.3 s
Pauza cuvinte	0.7 s (7 DOTs)
Amplitudine semnal	0-5 V

Fig. 4 Tabelul cu duratele alese pentru semnale

- Semnale linie si punct vor fi scrise ca TRUE de lungimi dfierite, punctul de o unitate de timp = ***1 TRUE*** ai linia de 3 unitati de timp ***= 3 TRUE***;
- Pauza intre simboluri un *ut* = ***1 FALSE***, intre litere 3 *ut* = ***3 FALSE*** si intre cuvinte 7 *ut* = ***7 FALSE***.

2. In continueare acest input de valori boolene va fi convertit intr-un semnal sinusoidal cu o frecventa variabila si lungimea unei unitati de timp de apoximativ *100ms* si o amplitudine a semanlului de *5V* dupa cum se observa si in figura 4.
3. Datele sunt trimise cu ajutorul DAQmx task prin iesirea analogica catre receptor.
4. Instrumentul virtual folosit pentru partea de recptie achiztioneaza semnaul pe intrarea analogica.
5. Dupa care este calculata lungimea perioadei semnalui care este adaugata intr-un buffer de citire.
6. Pentru a stabili exact lungimea unui estantion de mesaj se foloseste un buffer intermediar pentru a se stabili cand avem o trasnimise de mesaj.
7. Mesajul este apoi introdus intr-un alt buffer und este convertit in semnal logic.
8. Urmatorul pas este codificarea lui in “-” pentru semnal HIGH si “.” pentru semnal LOW.
9. Pasul urmator este calcularea lungimii acestor secventa pentru a determina daca am recptionat linie sau punct:
    - Se calculeaza si se returneaza un array cu lungimile secventelor;
    - Se returneaza lungimea secventei utile.
10. Ultimul pas este reconstructia semnalului primit si afisarea acestuia sun forma de cod Morse.
    - Se filtreaza elementele parasite si se returneaza un array de caractere ASCII;
    - Se returneaza lungimea secventei utile;
    - Se reface semanul sub forma de cod Morse.

In figura 5 este prezentata schema de principiu a telegrafului fiind prezentat ca un sistem intre doua cartele de achizitie date NI DAQmx USB 6008 ce comunica la mare distanta folosind repetoare de semnal pe firul de date.

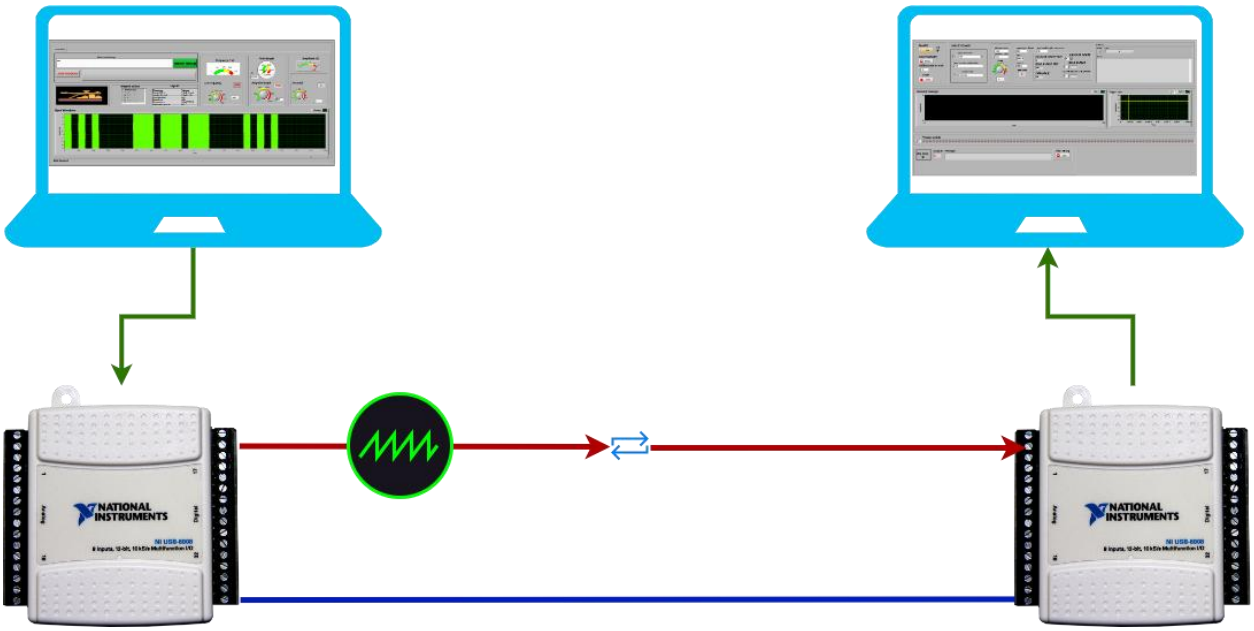


Fig. 5 Schema de principiu a telegrafului cu Labview si cartele de achitie date

2. Conceptia partii hardware a proiectului

Partea hardware este compusa din placa de achizitii date NI DAQmx USB 6281.

Mai jos este prezentat un tabel cu specificatiile Analog Input si Analog Output necesare pentru proiect[3]:

Analog Input:

Specificație	Valoare
Număr canale	2
Rezoluție DAC	16 biți
Rată actualizare max. (1 canal)	2.86 MS/s
Rată actualizare max. (2 canale)	2.00 MS/s per canal
Gama de ieșire (calibrată)	±1, ±2, ±5, ±10 V
Impedanță ieșire	0.2 Ω
Curent maxim ieșire	±5 mA
Supratensiune ieșire	±25 V
Glitch la pornire	2.3 V vârf timp de 1.2 s
Timp de stabilizare	3 µs pentru pas de amplitudine maximă
Slew rate	20 V/µs
FIFO de ieșire	8.191 eșantioane (partajat)
Moduri de regenerare AO	Non-periodic, regenerare din FIFO, regenerare din buffer host

Analog Output:

Specificație	Valoare
Număr canale	*2*
Rezoluție DAC	*16 biți*

Specificație	Valoare
Rată actualizare max. (1 canal)	*2.86 MS/s*
Rată actualizare max. (2 canale)	*2.00 MS/s per canal*
Gama de ieșire (calibrată)	*±1, ±2, ±5, ±10 V*
Impedanță ieșire	*0.2 Ω*
Curent maxim ieșire	*±5 mA*
Supratensiune ieșire	*±25 V*
Glitch la pornire	*2.3 V vârf timp de 1.2 s*
Timp de stabilizare	*3 µs pentru pas de amplitudine maximă*
Slew rate	*20 V/µs*
FIFO de ieșire	*8.191 eșantioane (partajat)*
Moduri de regenerare AO	Non-periodic, regenerare din FIFO, regenerare din buffer host

Conexiunea si configuratiile utilizate sunt urmatoarele:

1. Pentru transmsie:
    - Portul Analog Output 0 – ***ao0***;
    - Output terminal configuration: RSE Referenced Single-Ended, referinta fiind AO GND;
    - Generarea se face cu un numar finit de esnatioane si o frecventa de esantaionare de 10 kHz.
    - Semnalul de iesire este de tipul *Analog Wfm 1 Chan NSamp*.
2. Pentru receptie:
    - Portul Analog Input 0 – ***ai0***;
    - Input terminal configuration: RSE Referenced Single-Ended, referinta fiind AI GND;
    - Frecventa de esantaionare de 5 kHz.

- Semnalul de iesire este de tipul *Analog 2 DBL NChan NSamp* pentru a putea avea access la forma de unda si a putea calcula perioada unui impuls.
- 50 de estantioane per canal pentru a putea achizitiona cat mai multe esantioane intr-un timp cat mai scurt:

$$T_{scan} = \frac{\text{Samples per Channel}}{\text{Sample Rate}} = \frac{50}{5000} = 0.01 \text{ sec} = 10 \text{ ms}$$

Semnalul de intrare este un semnal sinusoidal cu amplitudinea intre 1 si 5 V si o frceventa variabila selectata de utilizator. Pentru un semnal Morse cu unitatea de timp egala cu 100ms setarile sunt urmatoarele (fig 6. fig.7):

1. 1000 de cicluri
2. 950 de esantionare

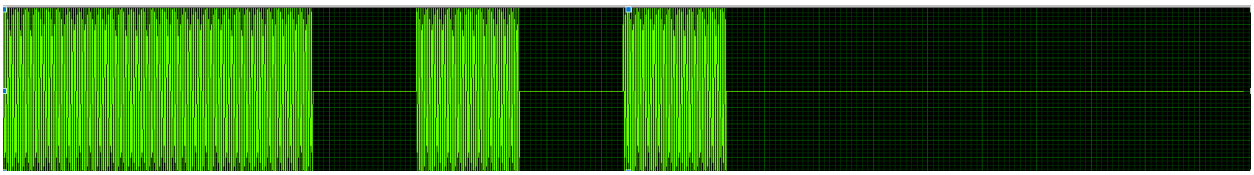


Fig. 6 Semnal generat pentru litera “D” → Morse: “-..”

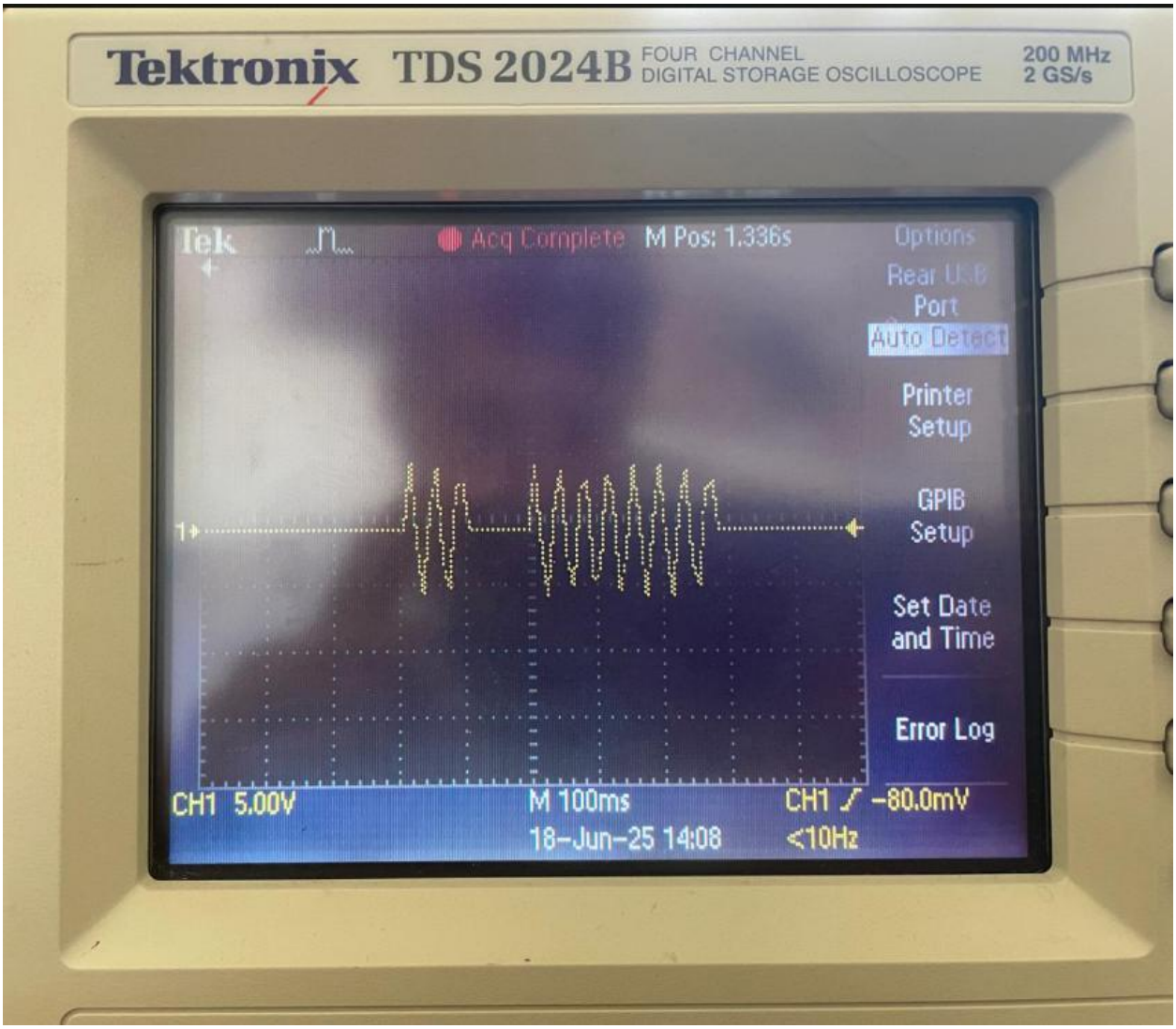


Fig. 7 – Litera A (.-): Punct = 1ut~*100ms* si Linia = 3ut~*300ms*

Achzitia semnalului se face odata la 10ms si ceea ce este necesar este calcularea perioadelor din semnalul de mesaj. Astfel din semnalul achiztionat rezulta un array ideal de forma:

[*per>0*, *per>0*, *per>0*, *0*, *per>0*, *0*, *0*, *0*]

Care poate fi interpretat dupa prelucrare ca "-." → Litera **N**.

Mai jos este prezentata diagrama hardware pe placa de achiztie date si setup-ul de test:

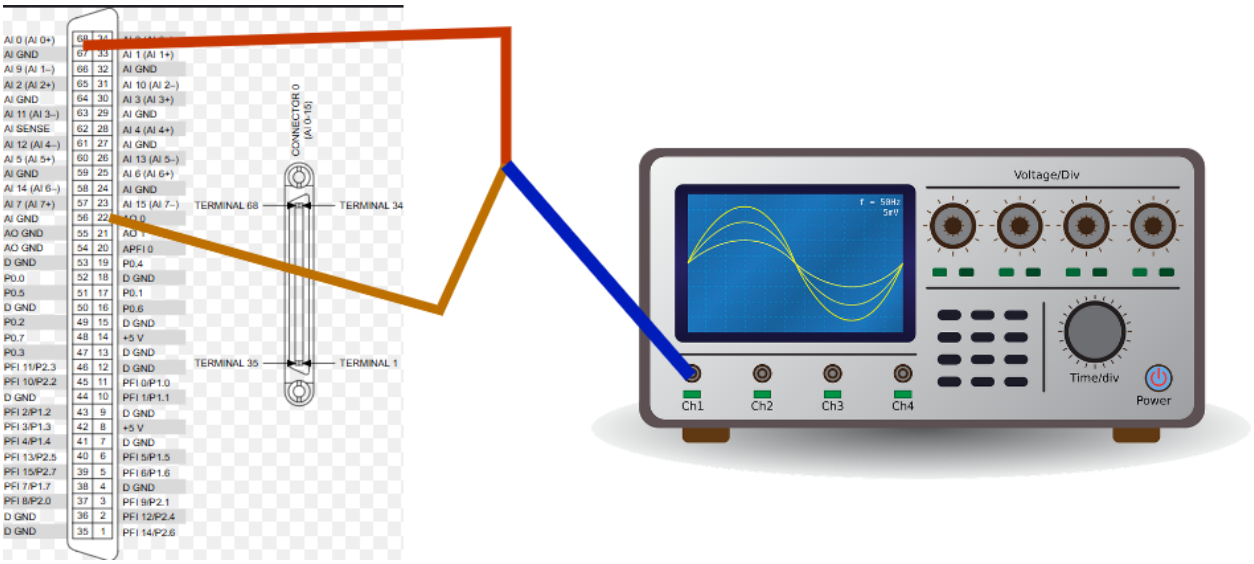


Fig. 8 Setup ul de test

3. Dezvoltarea aplicatiei Labview

Aplicatia Labview este compusa din doua intrumente virtuale:

1. Partea de transmisie: ***citire → codificare → transmisie***
2. Partea de recteptie: ***achizitie semnal → prelucrare → afisare***

Transmisia datelor

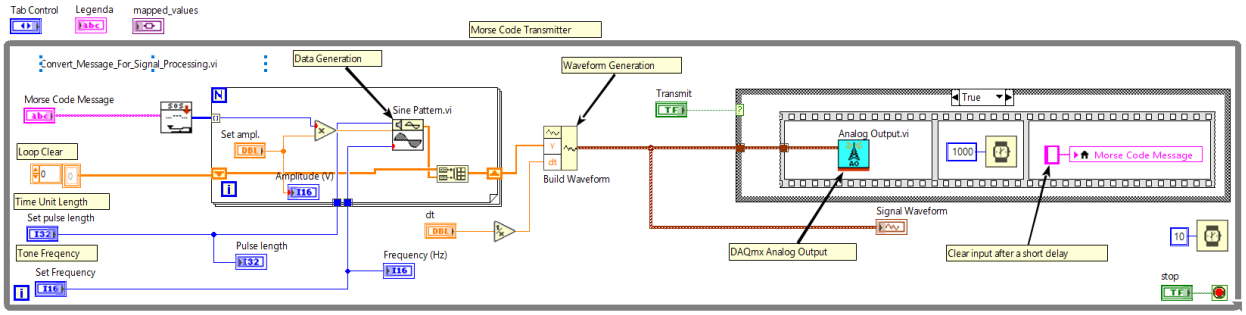


Fig. 9 Diagrama bloc pentru Transmitator

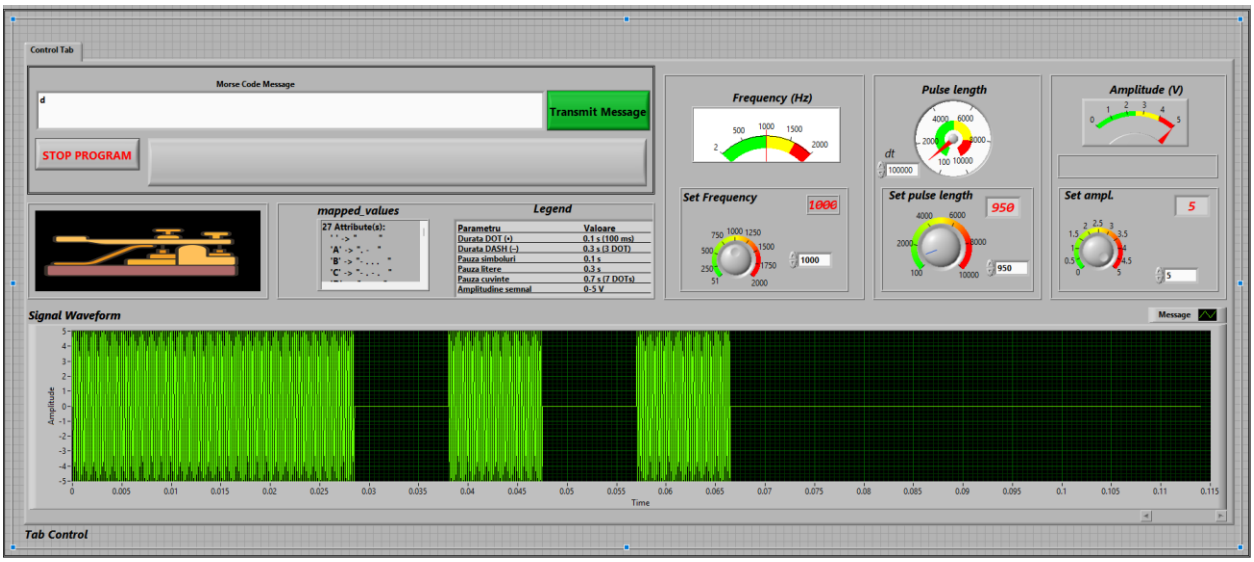


Fig. 10 Panoul frontal al transmitatorului

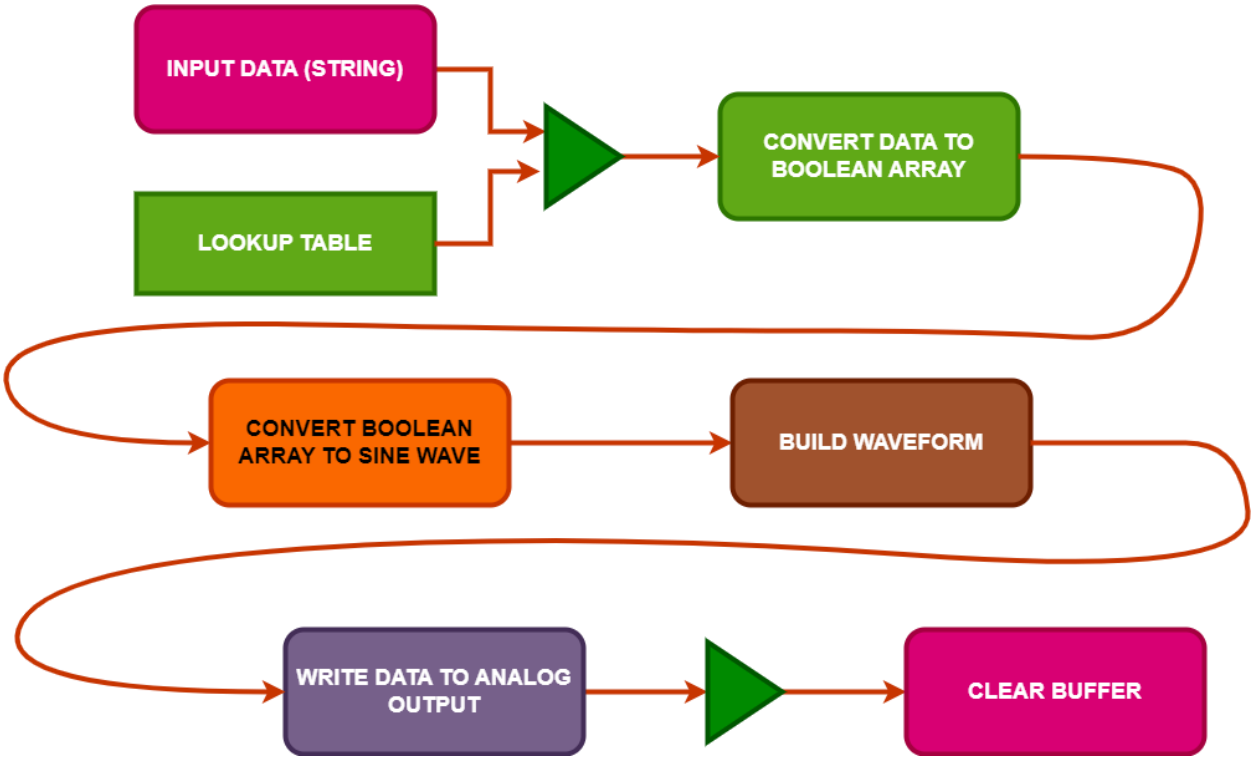


Fig. 11 Flow lochart transmisie

Conversia din string in semnal logic se face comparand fiecare caracter cu simbolurile mapate in ASCII si corespondenta lor in format Boolean, exemplu: pentru litera ‘A’:

- Codul ASCII: | 65 | 101 | 41 | 01000001 | A | A | | Uppercase A |
- Format Morse: “.-”
- Format Boolean: *TFTTTF*

Se parcurge tot sirul de input si se convertesc semnalele astfel ca va rezulta un array de date boolene care vor defini impulsurile sinusoidale ale semnalului ce urmeaza a fi trimis.

Tabelul de codificare este prezentat in figura 12 si modul de codficare in figura 13.

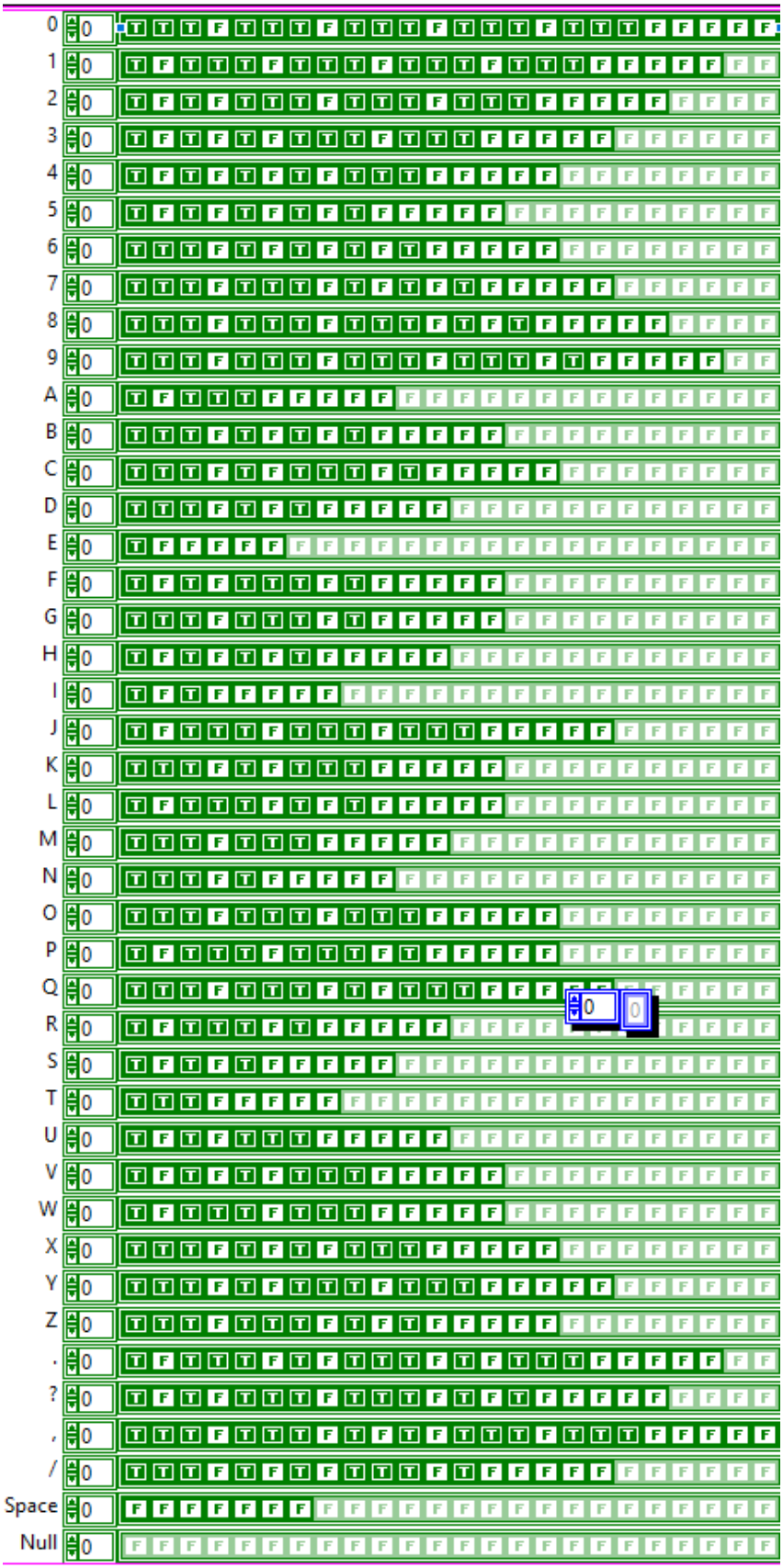


Fig. 12 Tabelul de codficare

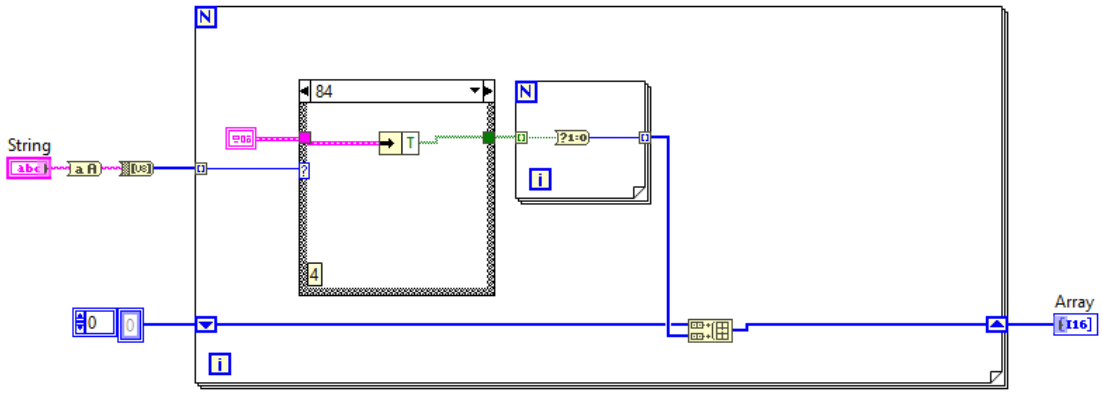


Fig. 13 Codificarea mesajului

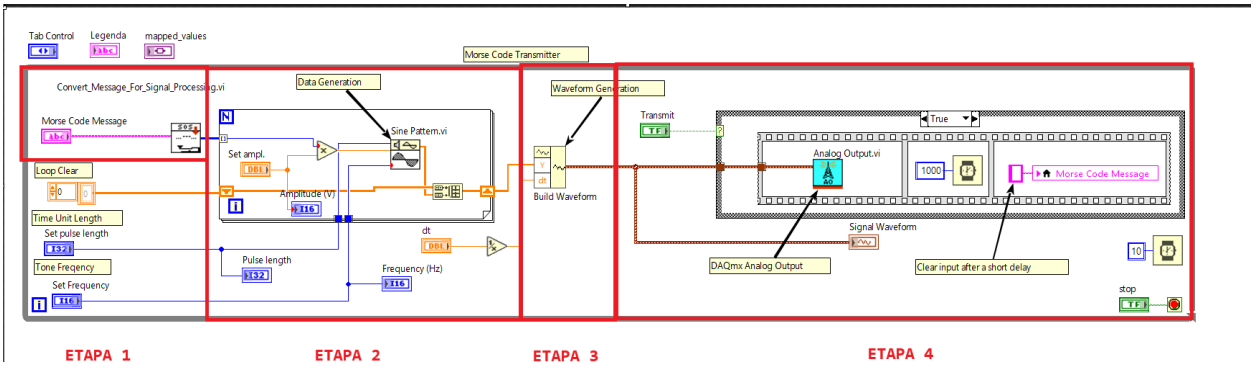


Fig. 14 Etapele transimisei de date

Transmisia datelor se face generand semnalul cu ajutorul DAQmx prin portul ao0 din Analog Outputs cu specificatiile mentionate anterior:

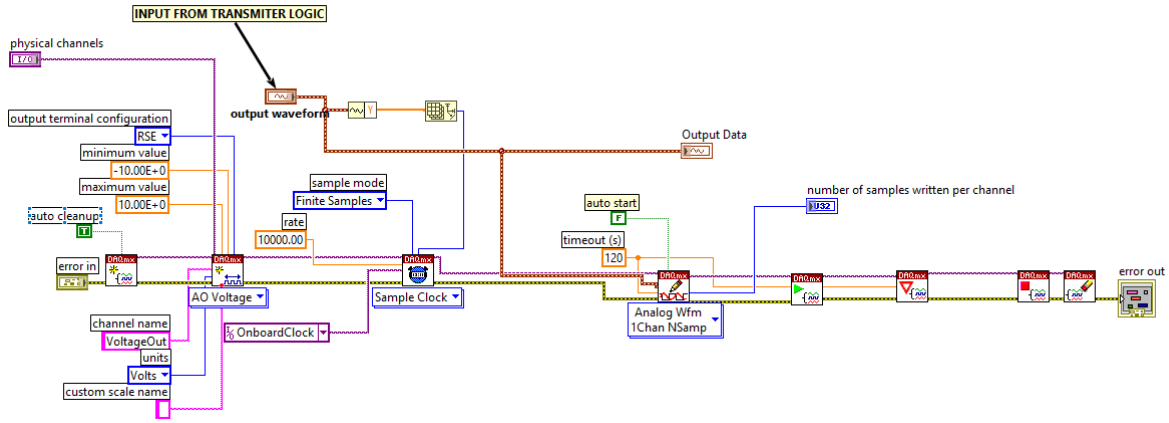


Fig. 15 Transmisia datelor utilizand DAQmx

Receptionarea mesajului

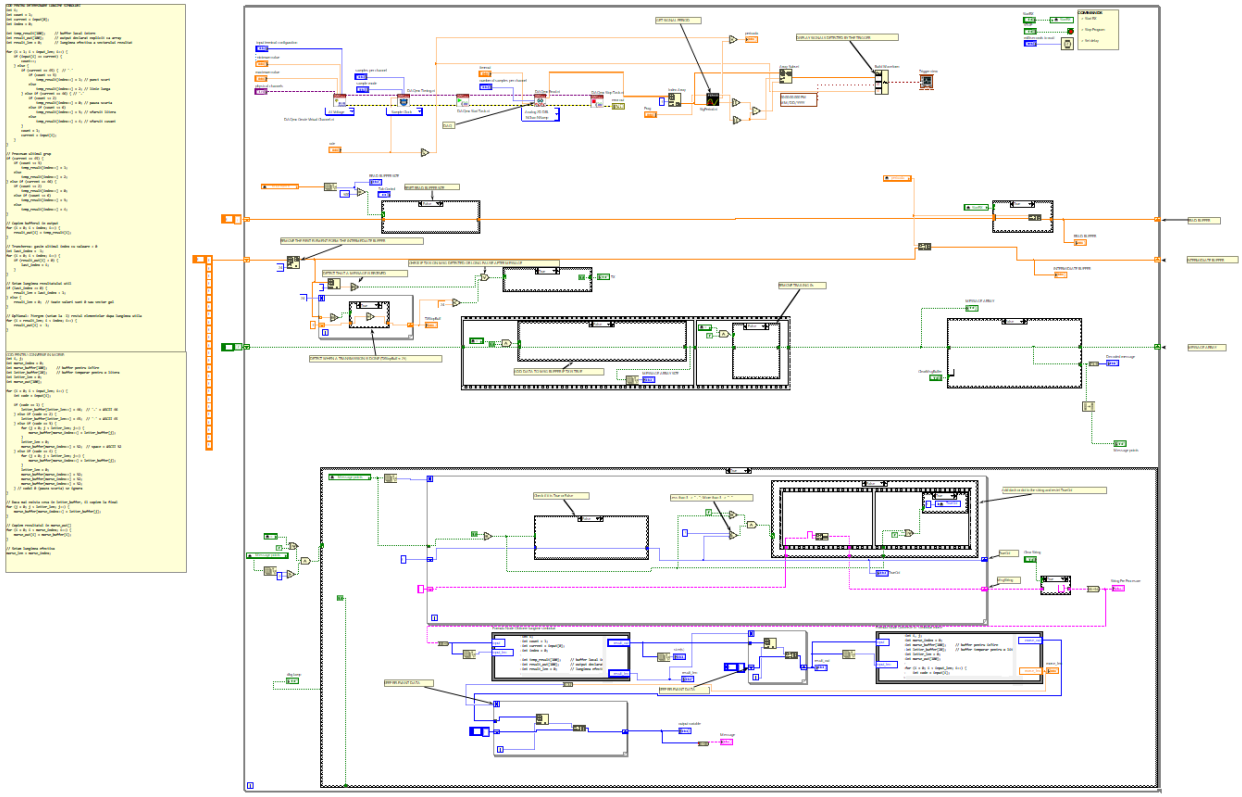


Fig. 16 Schema bloc a receptorului

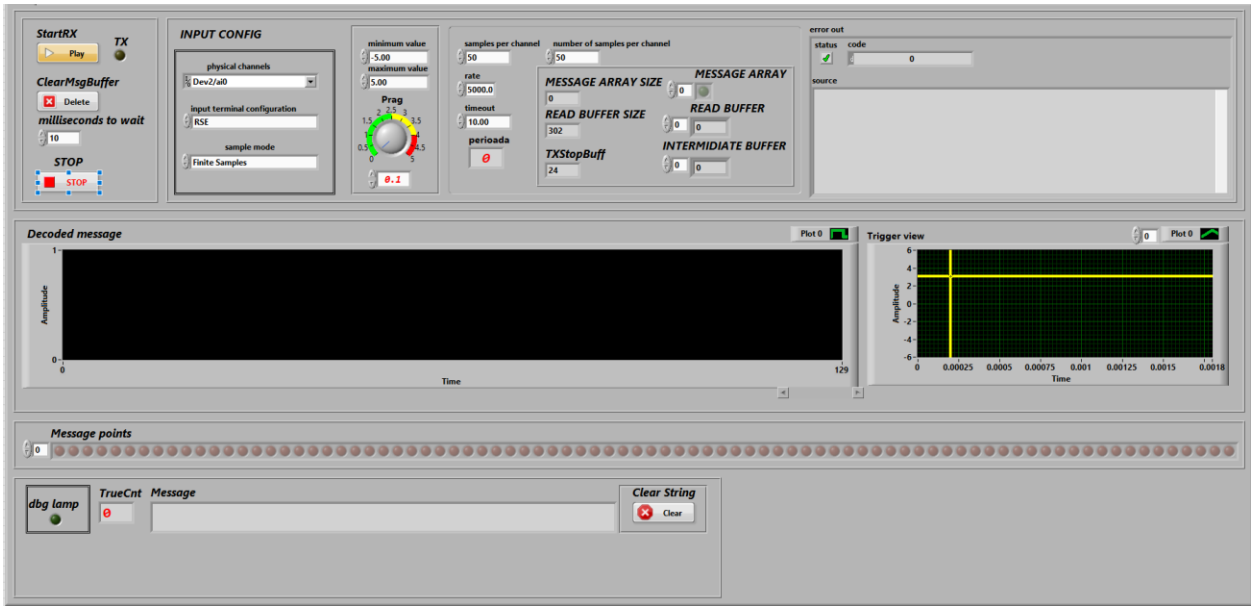


Fig. 17 Panoul frontal al receptorului

Flow chart-ul este prezentat in figura 18. Receptie este compusa din urmatoarele etape:

1. Achitia semnalului cu tigger software cu prag setat;
2. Calcularea lungimii unei perioade de semnal + afisare snapshot
3. Adaugarea acestor date intr-un buffer;
4. Utilizarea unui buffer intermediar pentru a detecta receptia unui mesaj → Daca primim mesaj il salvam intr-un buffer separat;
5. La terminarea trasnsimisiei se decodifica mesajul:
   a. Se determina lungimea secventelor de perioade inlantuite, de exemplu max 3 elemente > 0 (TTT) inseamna punct si ce e peste linie;
   b. Se convertesc datele in string dupa care sunt afisate in limbaj Morse.

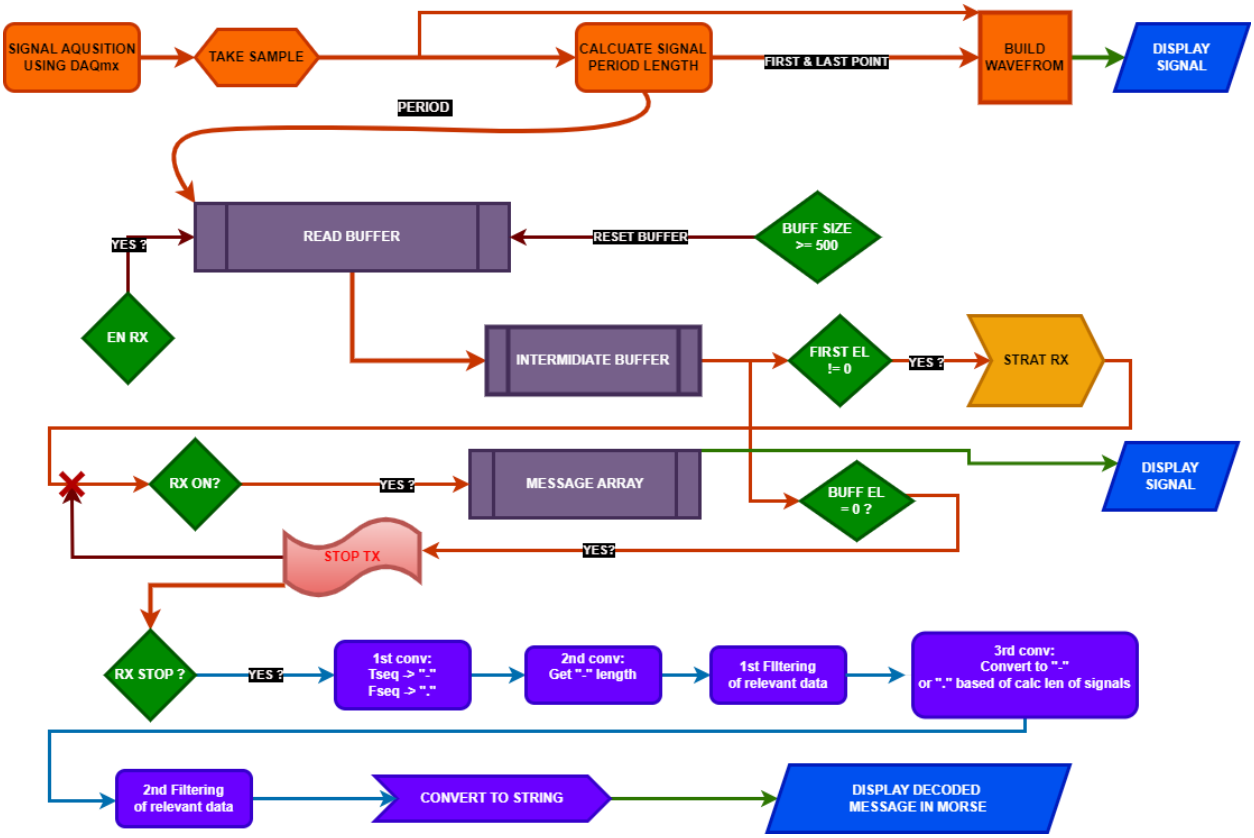


Fig. 18 Structura instrumentului receptor

Achizitia semnalului se face cu o rata de esantionare de 5Khz si 50 de esantioane pe canal rezulta un esation la fiecare 10ms (fig 19 si 20):

$$T_{scan} = \frac{\text{Samples per Channel}}{\text{Sample Rate}} = \frac{50}{5000} = 0.01 \text{ sec} = 10 \text{ ms}$$

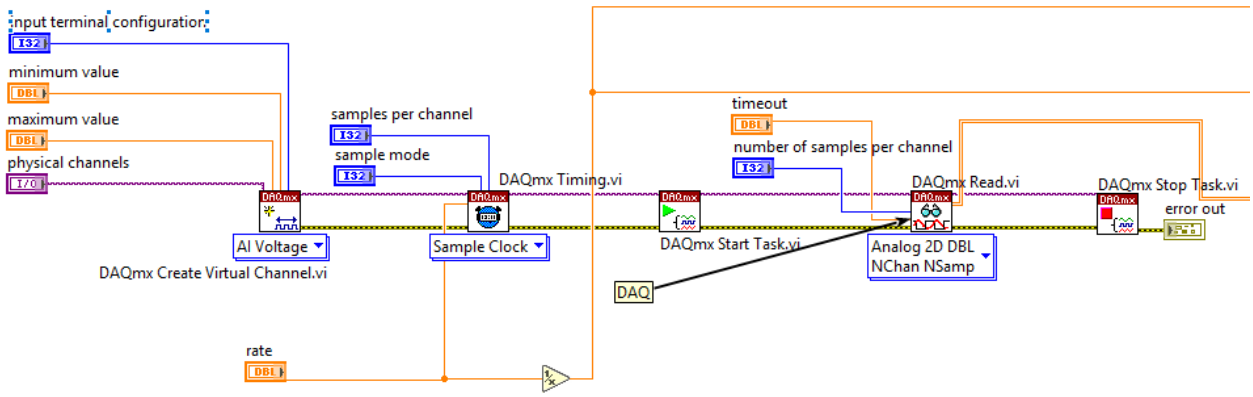


Fig. 19 Schema bloc DAQ

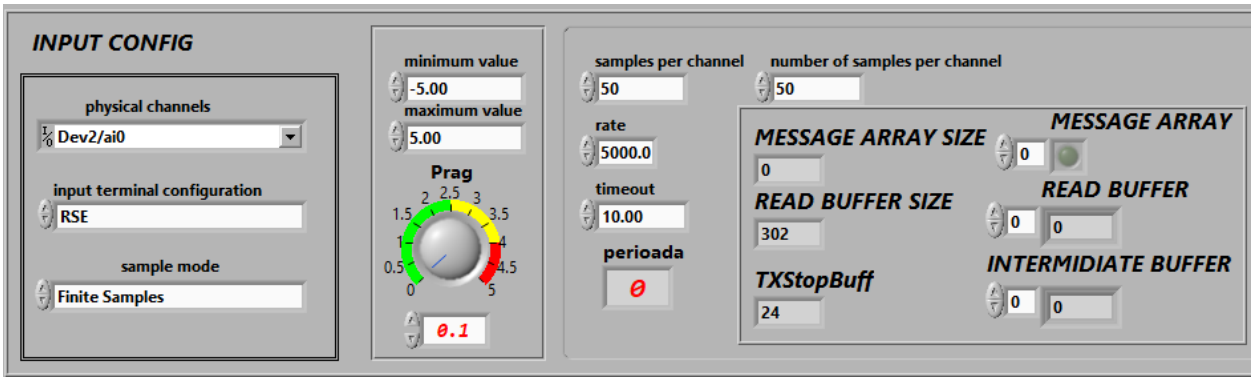


Fig. 20 Setari pe panou frontal:

Calcul lungimii perioadei se face folosind urmatorul algoritm prezentat in figura 21 iar in figura 22 este prezentata legatura acestuia in schema bloc.

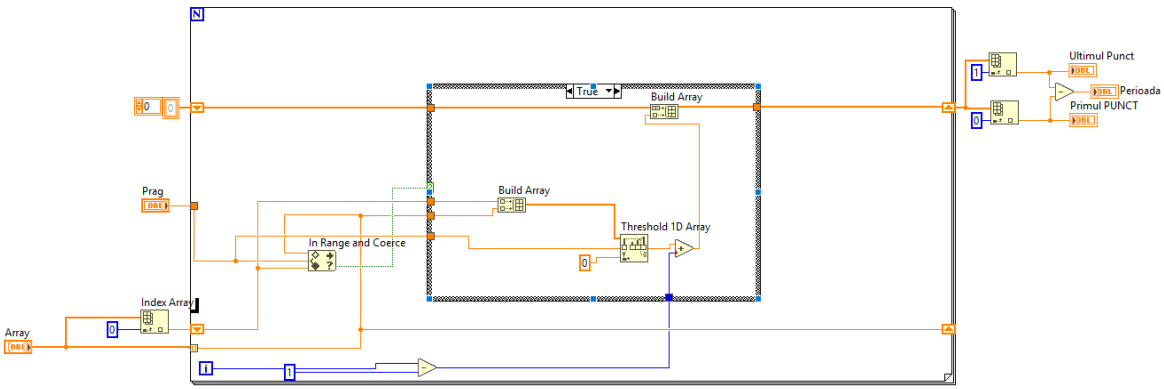


Fig. 21 Algortimul de calcul pentru perioada

Perioada este calculata ca fiind diferenta dintre primul si ultimul punct pe un esantion de semnal.

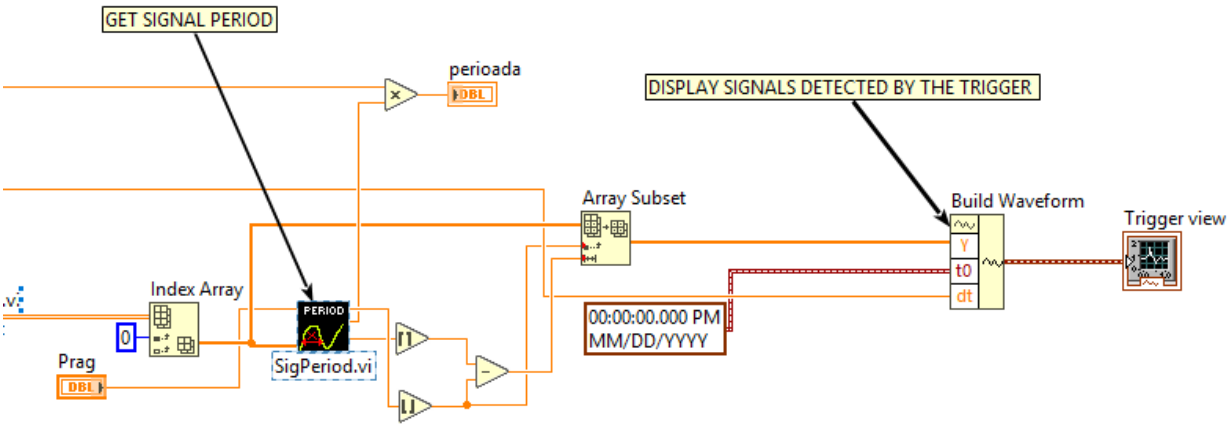


Fig. 22 Legatura blocului in schema bloc

Buffer-ul de citire se resteaza daca lugimea sa depaseste 500 de elemente si de asemenea poate fi pornit sau oprit (fig. 23 si 24)

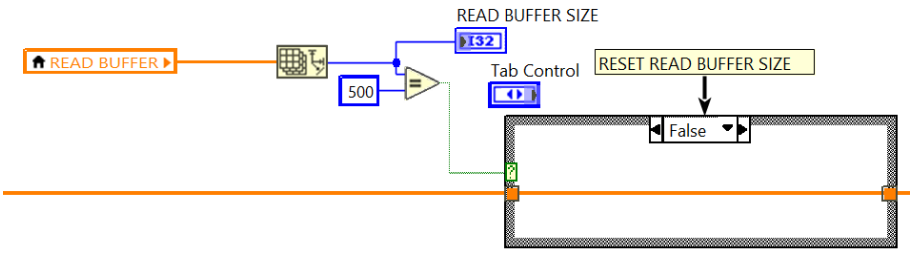


Fig. 23 Resetare buffer

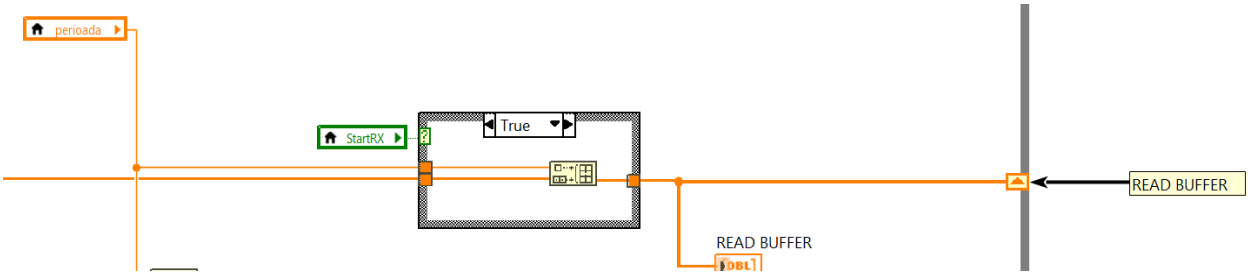


Fig. 24 Activare/Dezactivare buffer

Etapa urmatoare este utilzarea buffer-ului intermediar pentru a putea salva mesajul atunci cand acesta este transmis (fig. 25). Buffer-ul se actualizeaza constant eliminand primul element si adaugand alt element, un fel de FIFO. Daca ultimul element intrat in buffer este mai mare ca 0 atunci TX a inceput si putem salva mesajul. Daca buffer-ul are toate elementele 0 atunci transmisia s-a incheiat si putem incepe prelucrarea acestuia.

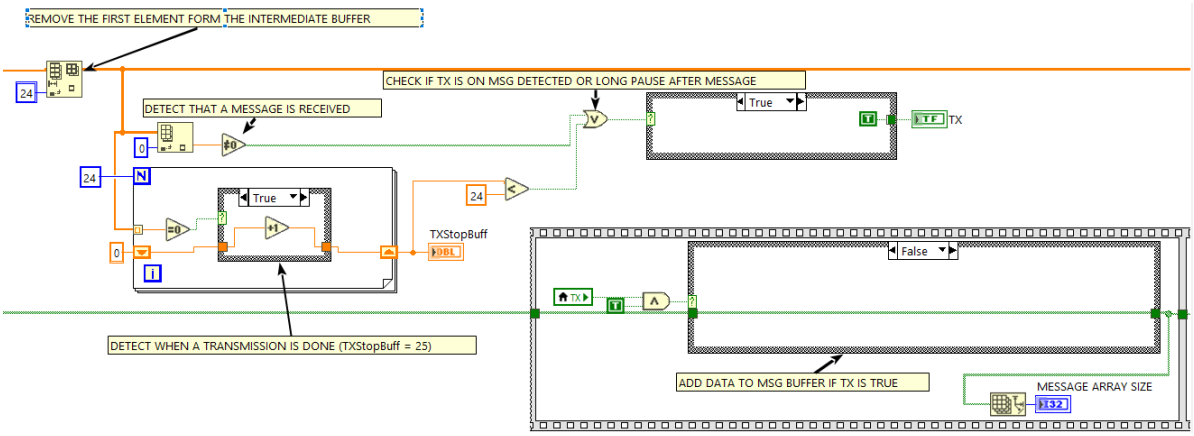


Fig. 25 Operatiuni buffer intermediar

Buffer-ul de mesaje poate fi sters la comanda, de asemnea datele sunt afisate pe grafic in ordinea aparitiei si intr-un array de tip boolean pentru a putea vizualiza si lungimea efectiva a impulsurilor detectate. (fig. 26)

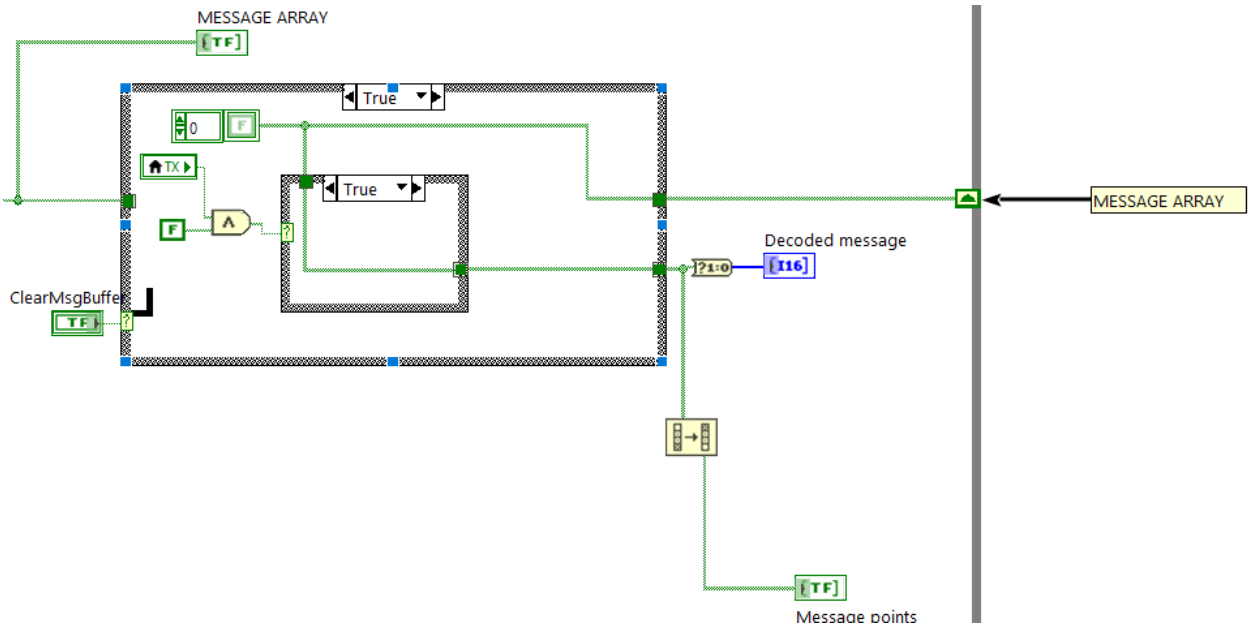
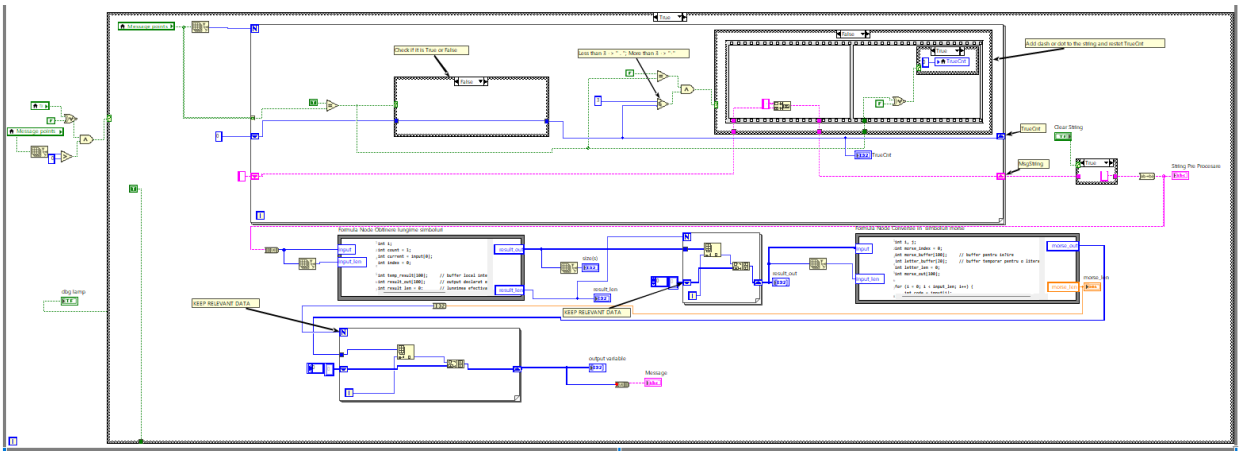


Fig. 26

Decodificarea Mesajului



Lantul de date pana la decodificare se prezinta in modul urmator la **modul ideal**:

Trimitem litera ***N***:

`Semnal → Calcul perioada → rezulta un array de tip (excluzand interferentele):`

[0.18, 0.18, 0.18, 0, 0.18]

`→ acesta este transformat in (excluzand interferentele):`

[T, T, T, F, T]

→ acesta este transformat in (excluzand interferentele):

[-,-,-,.,-,]

→ acesta este transformat dupa **filtrare** in(excluzand interferentele):

[3,0,1]

→ acesta este transformat dupa **filtrare** in (excluzand interferentele):

[45, 32, 56] (ASCII) → -. (Morse)

Pentru decodificarea mesajului prima data semnalele de tip True si False sunt codificate in “-” pentru True si “.” (figura 27):

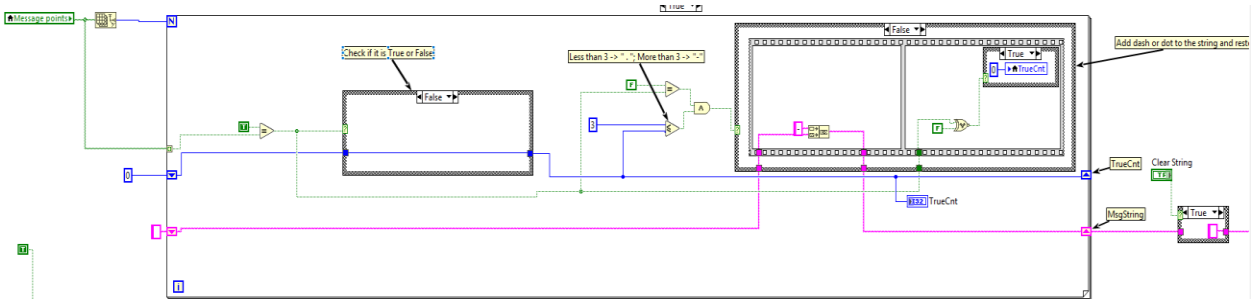


Fig. 27

Este prima forma de filtrare se cauta gruparea secventelor de perioade dupa lungimea lor.

Pasul urmator este sa cream un array ce contine lungimea fiecarei secvente. Aceasta se realizeaza cu un formula node (fig. 29) si cu urmatorul cod in C dupa care sunt eliminate elementele inutile deoarece buffer ul intern are o dimesiune fixa (fig. 29).

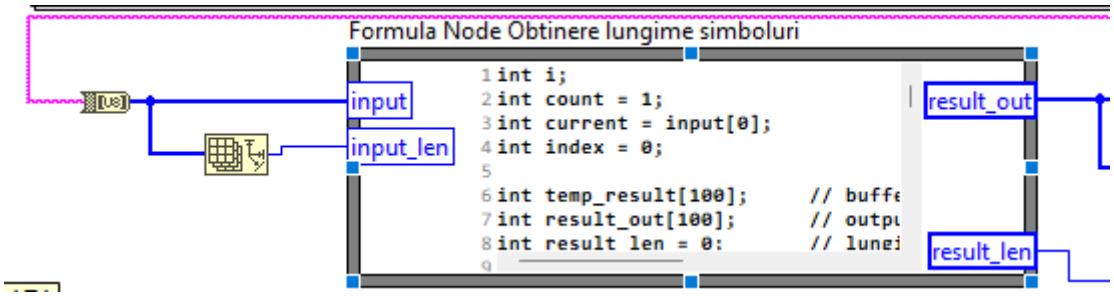


Fig. 28

```c
int i;
int count = 1;
int current = input[0];
int index = 0;

int temp_result[100];      // buffer local intern
int result_out[100];       // output declarat explicit ca array
int result_len = 0;        // lungimea efectiva a vectorului rezultat

for (i = 1; i < input_len; i++)
{
    if (input[i] == current)
    {
        count++;
    }
    else
    {
        if (current == 45){  // '-'
            if (count <= 3)
                temp_result[index++] = 1; // punct scurt
            else
                temp_result[index++] = 2; // linie lunga
        } else if (current == 46) { // '.'
            if (count <= 2)
                temp_result[index++] = 0; // pauza scurta
            else if (count <= 6)
                temp_result[index++] = 3; // sfarsit litera
            else
                temp_result[index++] = 4; // sfarsit cuvant
        }
        count = 1;
        current = input[i];
    }
}

// Procesam ultimul grup
if (current == 45) {
```

```
    if (count <= 3)
        temp_result[index++] = 1;
    else
        temp_result[index++] = 2;
}
else if (current == 46)
{
    if (count <= 2)
        temp_result[index++] = 0;
    else if (count <= 6)
        temp_result[index++] = 3;
    else
        temp_result[index++] = 4;
}

// Copiem bufferul in output
for (i = 0; i < index; i++) {
    result_out[i] = temp_result[i];
}

// Truncherea: gasim ultimul index cu valoare > 0
int last_index = -1;
for (i = 0; i < index; i++) {
    if (result_out[i] > 0) {
        last_index = i;
    }
}

// Setam lungimea rezultatului util
if (last_index >= 0) {
    result_len = last_index + 1;
} else {
    result_len = 0;  // toate valori sunt 0 sau vector gol
}

// Optional: stergem (setam la -1) restul elementelor dupa lungimea utila
for (i = result_len; i < index; i++) {
    result_out[i] = -1;
}
```

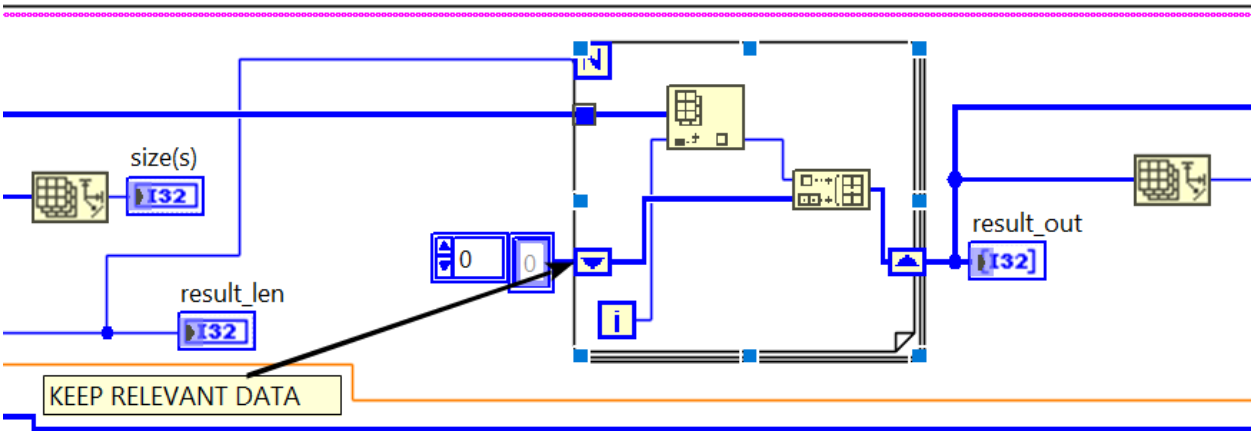


Fig. 29 Filtrarea rezultatelor utile

Urmatorul pas este convertirea datelor in ASCII. Daca elemntul de exemplu este mai mare ca 3 de exmplu este convertit in ***ASCII 46 adica linie***, invers in ***ASCII 45 adica punct*** si daca este 0 in ***ASCII 32 adica spatiu***. Aceasta se realizeaza cu un formula node (fig. 30) si cu urmatorul cod in C dupa care sunt eliminate elementele inutile deoarece buffer ul intern are o dimesiune fixa (fig. 31).

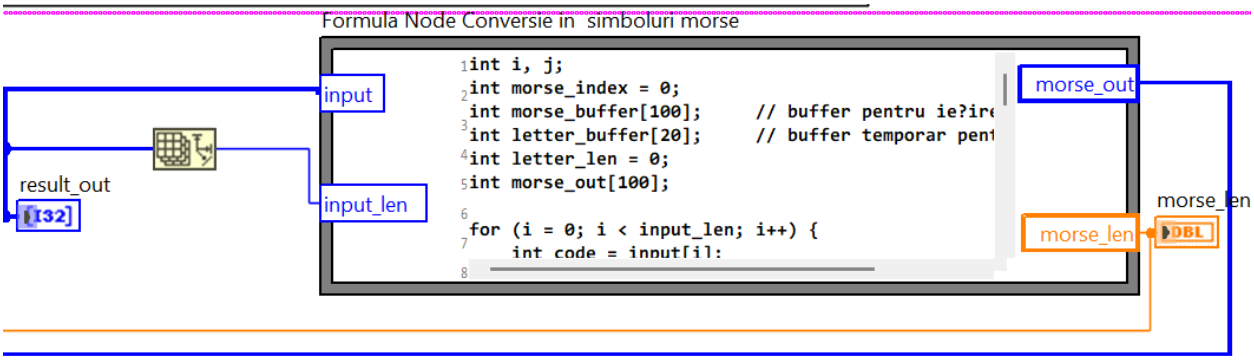


Fig. 30

```
int i, j;
int morse_index = 0;
int morse_buffer[100]; // buffer pentru iesire
int letter_buffer[20]; // buffer temporar pentru o litera
int letter_len = 0;
int morse_out[100];

for (i = 0; i < input_len; i++)
{
    int code = input[i];

    if (code == 1)
    {
```

```
        letter_buffer[letter_len++] = 46; // '.' = ASCII 46
    }
    else if (code == 2)
    {
        letter_buffer[letter_len++] = 45; // '-' = ASCII 45
    }
    else if (code == 3)
    {
        for (j = 0; j < letter_len; j++)
        {
            morse_buffer[morse_index++] = letter_buffer[j];
        }
        letter_len = 0;
        morse_buffer[morse_index++] = 32; // space = ASCII 32
    }
    else if (code == 4)
    {
        for (j = 0; j < letter_len; j++)
        {
            morse_buffer[morse_index++] = letter_buffer[j];
        }
        letter_len = 0;
        morse_buffer[morse_index++] = 32;
        morse_buffer[morse_index++] = 32;
        morse_buffer[morse_index++] = 32;
    } // codul 0 (pauza scurta) se ignora
}

// Daca mai exista ceva în letter_buffer, il copiem la final
for (j = 0; j < letter_len; j++)
{
    morse_buffer[morse_index++] = letter_buffer[j];
}

// Copiem rezultatul în morse_out[]
for (i = 0; i < morse_index; i++)
{
    morse_out[i] = morse_buffer[i];
}

// Setam lungimea efectiva
morse_len = morse_index;
```

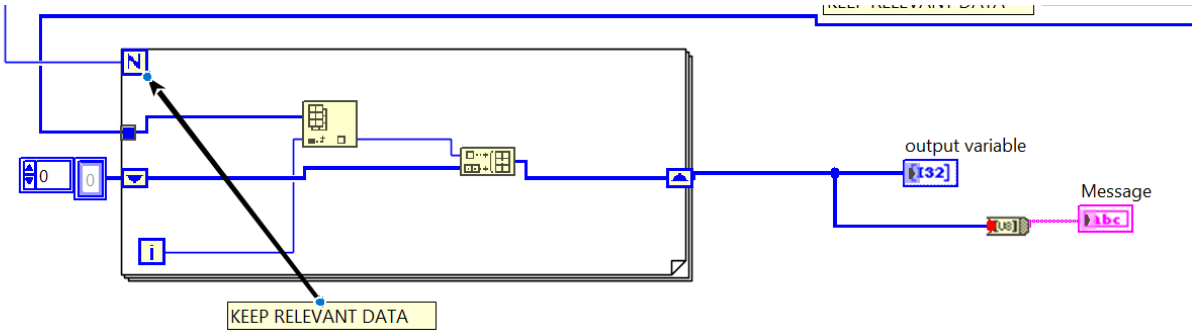
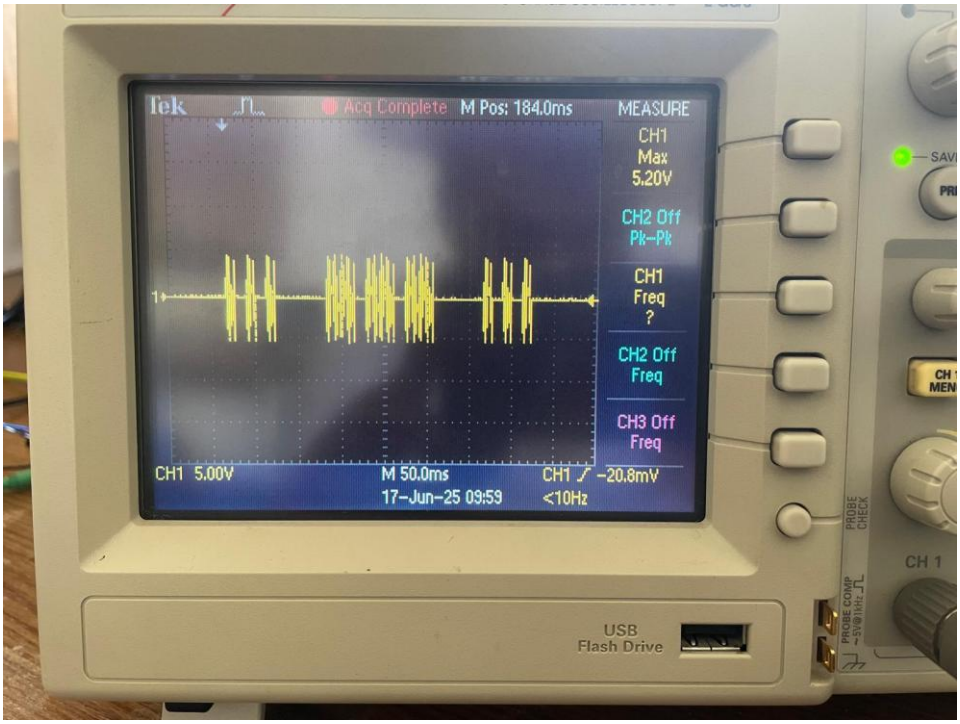
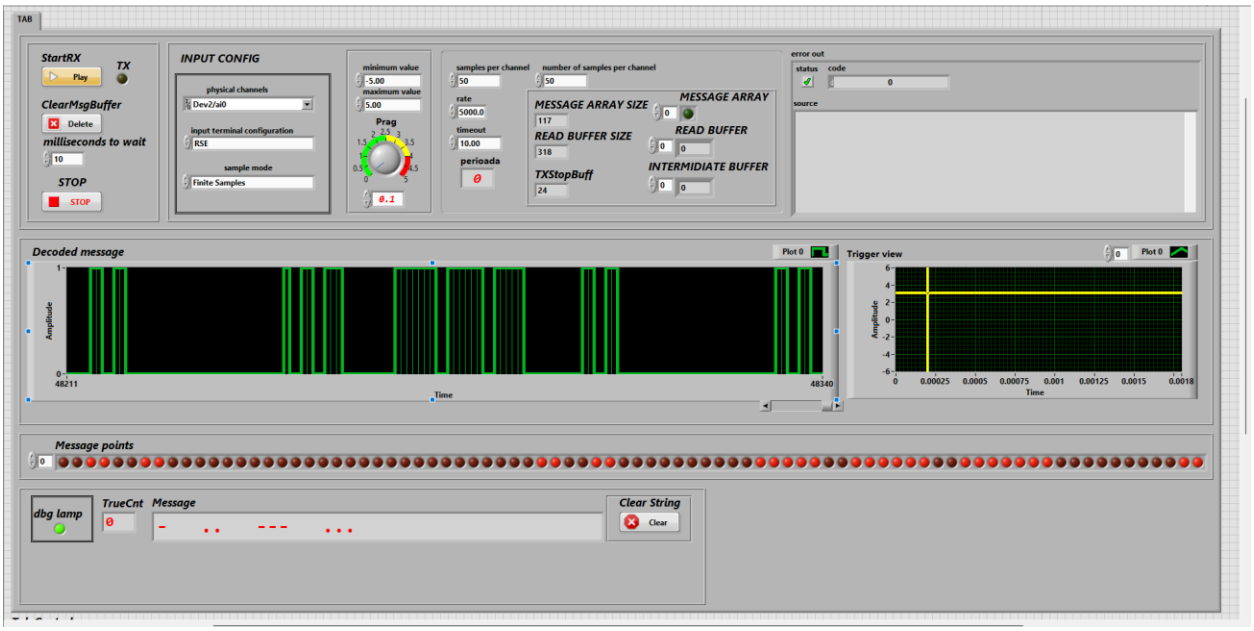


Fig. 31 Filtrarea rezultatelor utile si conversia in string

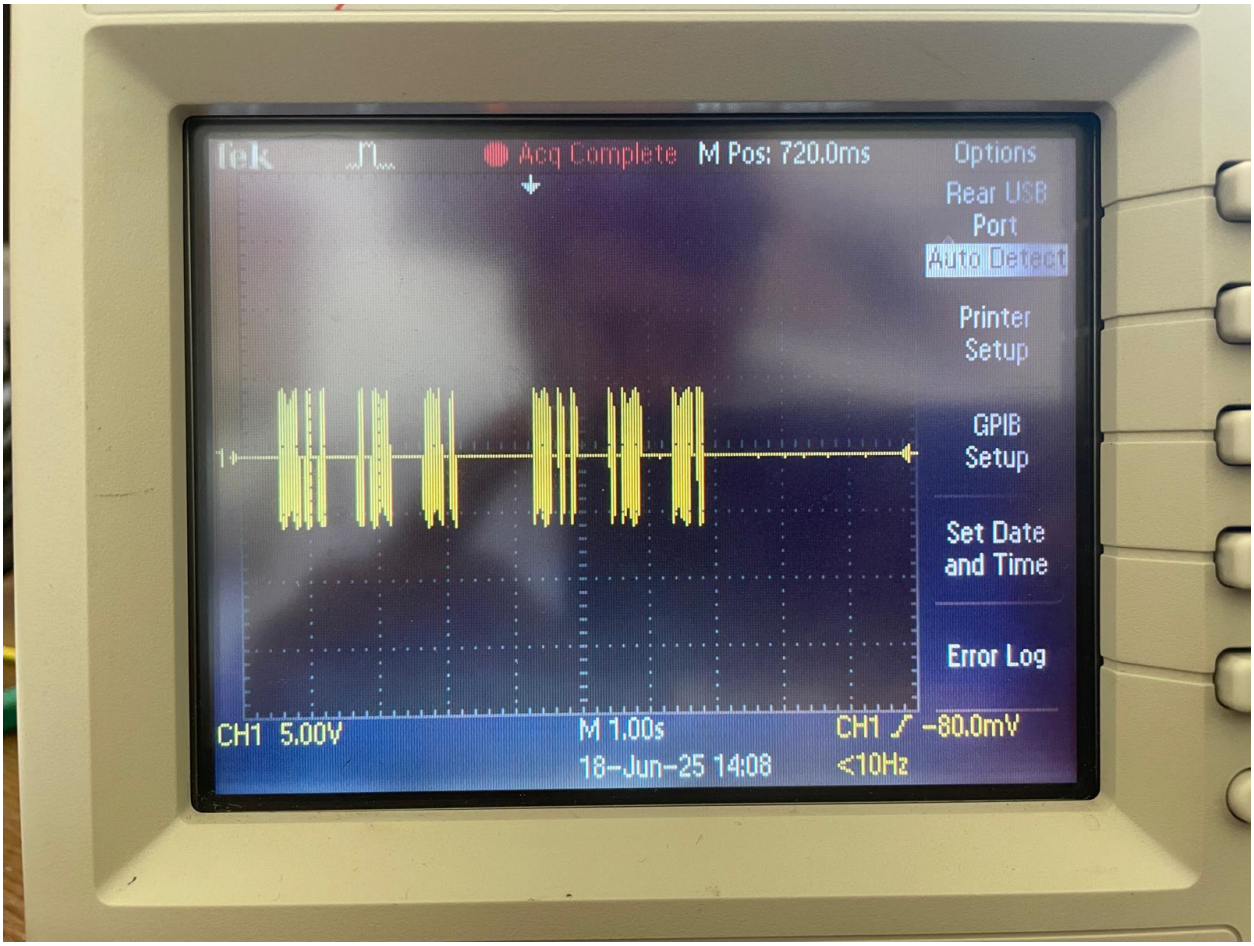
4. Testarea sistemului

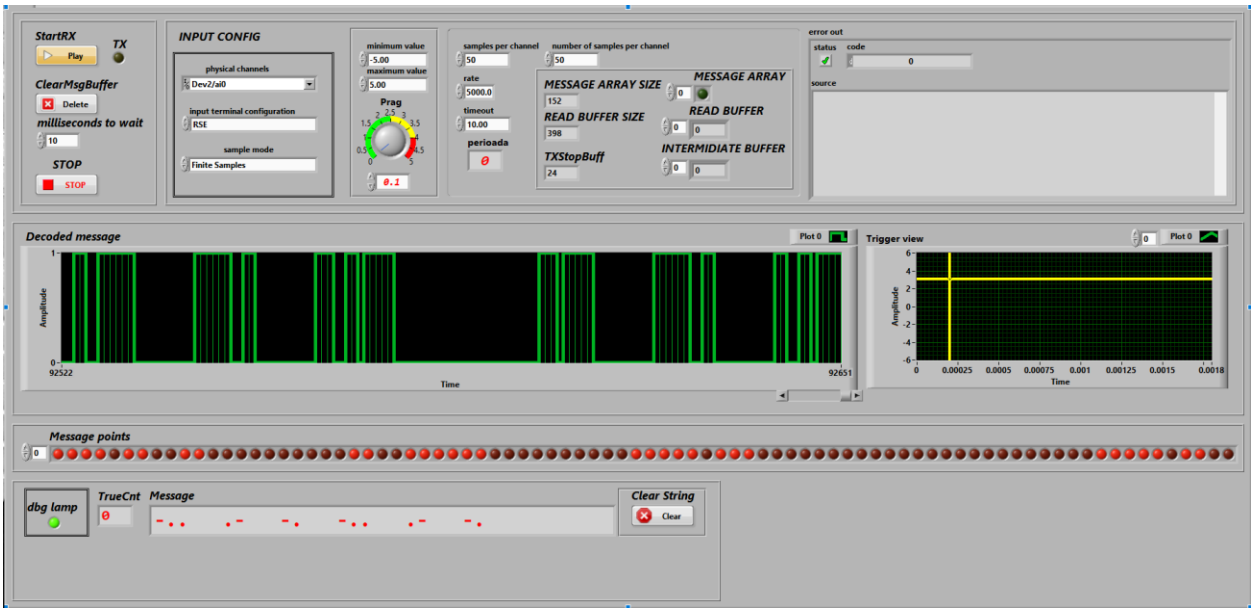
Test nr.1 “SOS”: *VI TX; Osciloscop; VI RX*



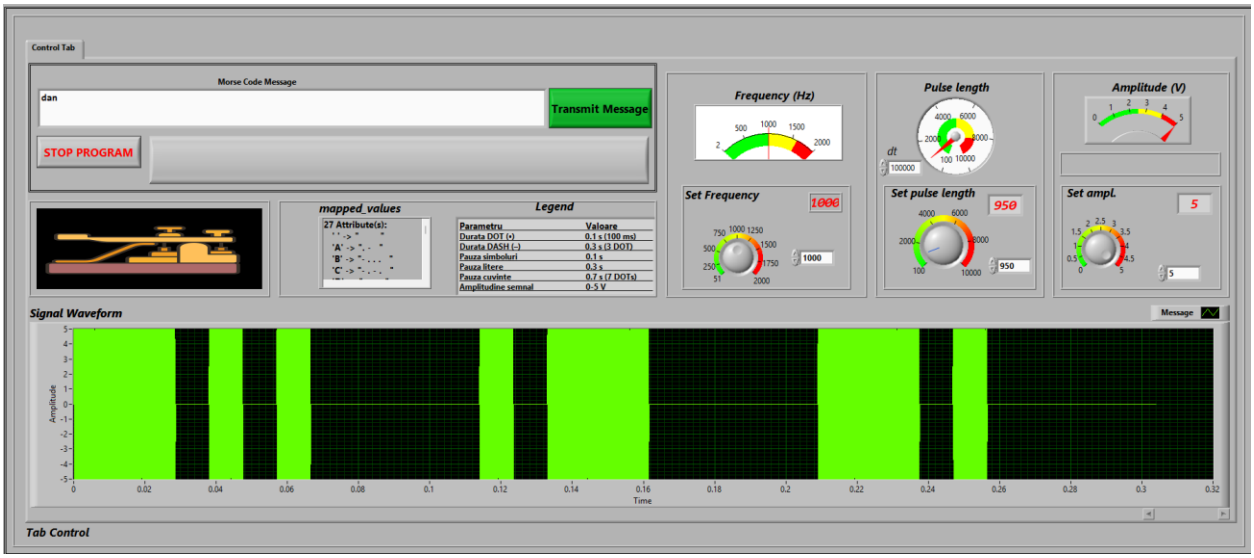


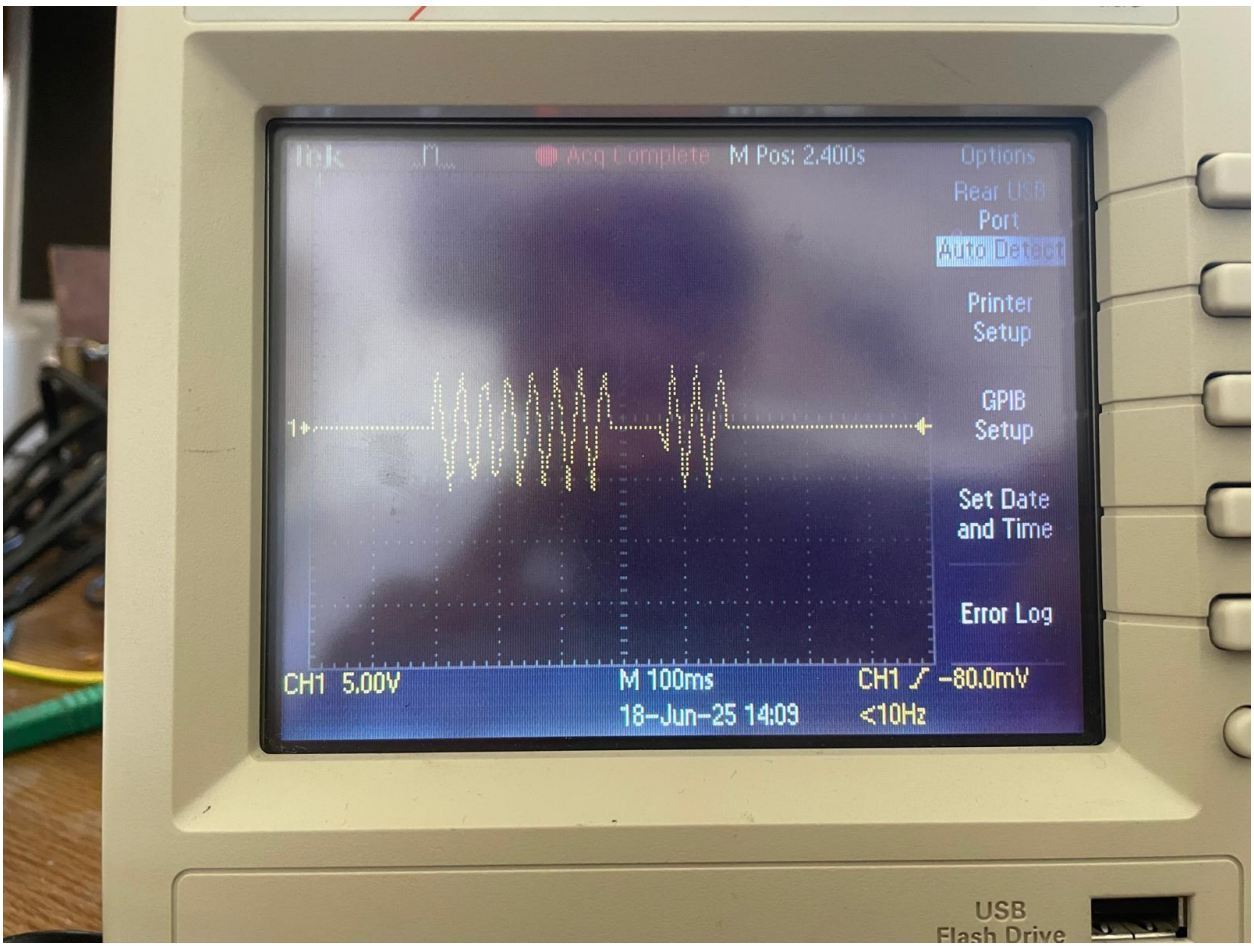
Test nr.2 “Dan Dan”: *VI TX; Osciloscop; VI RX*

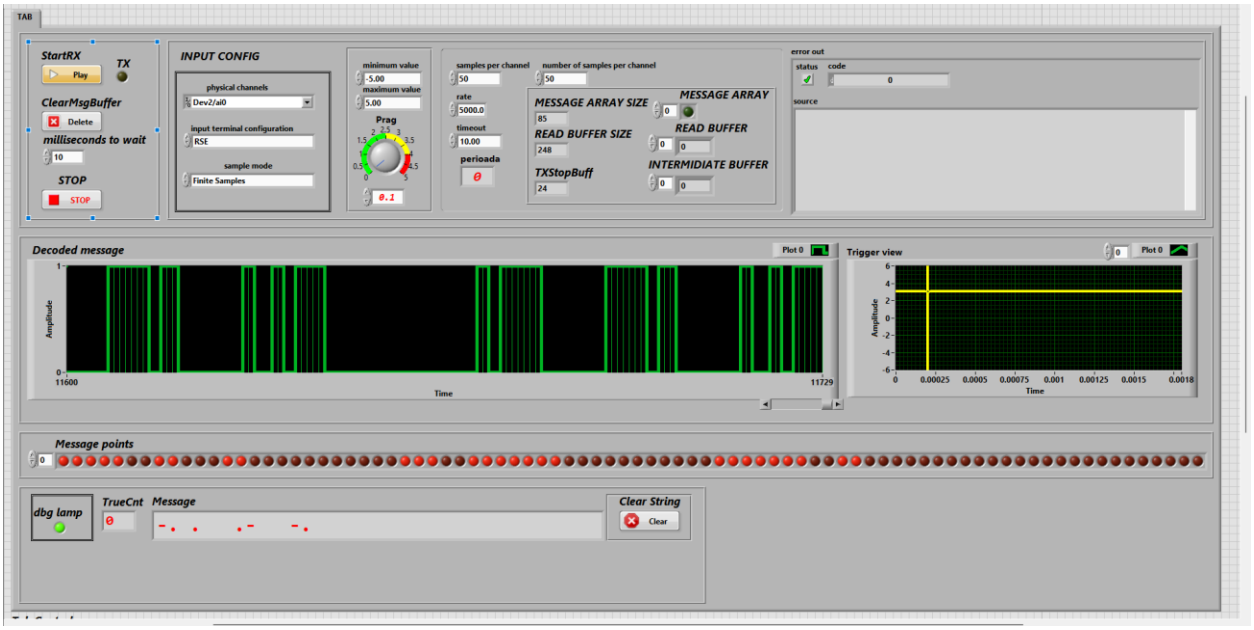

Tek
Acq Complete
M Pos: 720.0ms
Options
Rear USB Port
Auto Detect
Printer Setup
GPIB Setup
Set Date and Time
Error Log
CH1 5.00V
M 1.00s
18-Jun-25 14:08
CH1 -80.0mV
<10Hz


StartRX
Play
TX
ClearMsgBuffer
Delete
milliseconds to wait
10
STOP
STOP
INPUT CONFIG
physical channels
Dev2/ai0
input terminal configuration
RSE
sample mode
Finite Samples
minimum value
-5.00
maximum value
5.00
Prag
0.1
samples per channel
50
number of samples per channel
50
rate
5000.0
timeout
10.00
perioada
0
MESSAGE ARRAY SIZE
152
MESSAGE ARRAY
READ BUFFER SIZE
398
READ BUFFER
TXStopBuff
24
INTERMIDIATE BUFFER
error out
status
code
source
Decoded message
Plot 0
Amplitude
Time
92522
92651
Trigger view
Message points
dbg lamp
TrueCnt
Message
Clear String
Clear

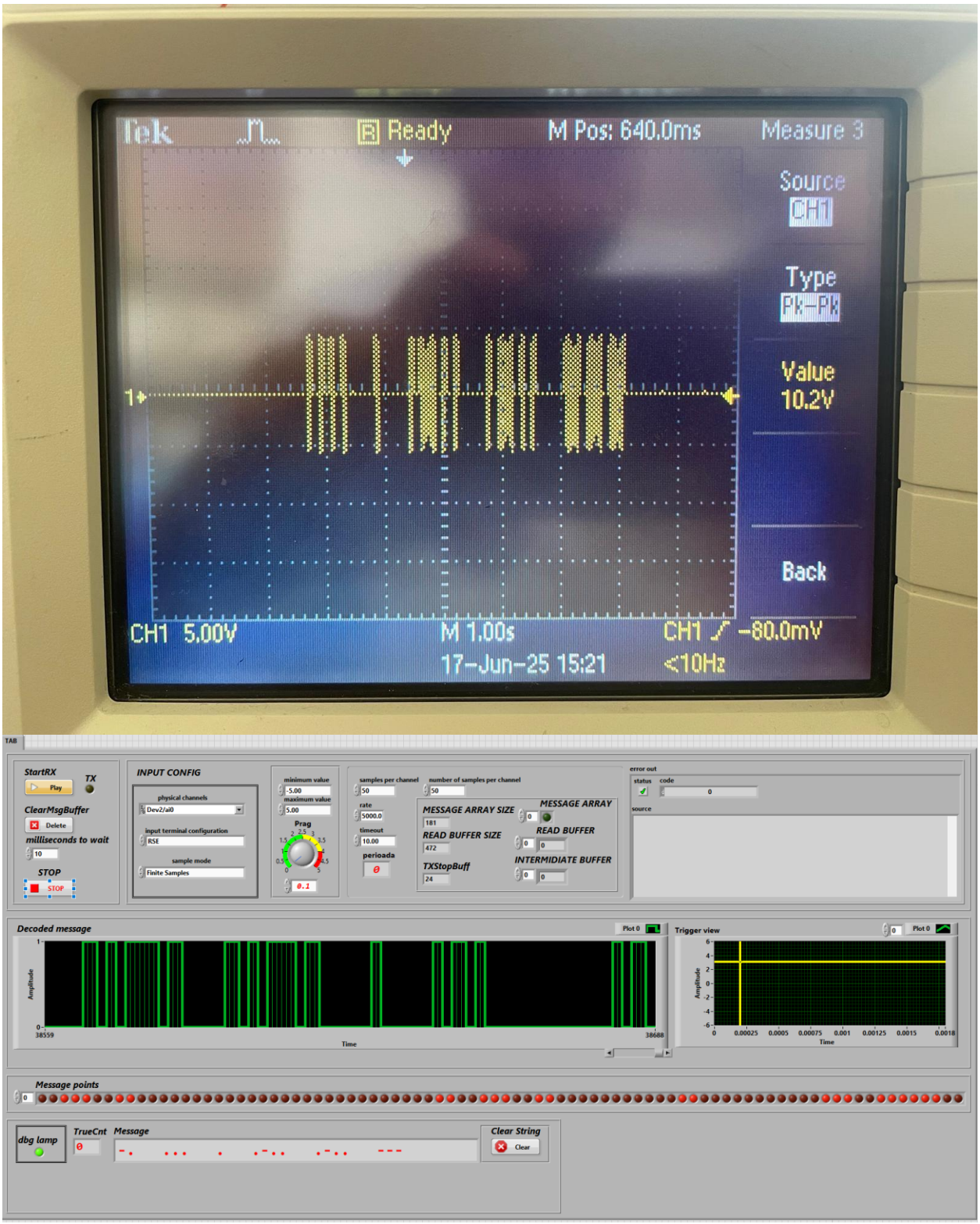
Test nr.3 “Dan”: *VI TX; Osciloscop litera N; VI RX*







Test nr.4 “Hello”: *VI TX; Osciloscop ; VI RX*


Tek
Ready
M Pos: 640.0ms
Measure 3
Source
CH1
Type
Value
10.2V
Back
CH1 5.00V
M 1.00s
CH1 -80.0mV
17-Jun-25 15:21
<10Hz
TAB
StartRX
Play
TX
ClearMsgBuffer
Delete
milliseconds to wait
10
STOP
STOP
INPUT CONFIG
physical channels
Dev2/ai0
input terminal configuration
RSE
sample mode
Finite Samples
minimum value
-5.00
maximum value
5.00
Prag
0.1
samples per channel
50
number of samples per channel
50
rate
5000.0
timeout
10.00
perioada
0
MESSAGE ARRAY SIZE
181
MESSAGE ARRAY
READ BUFFER SIZE
472
READ BUFFER
TXStopBuff
24
INTERMIDIATE BUFFER
error out
status
code
0
source
Decoded message
Plot 0
Amplitude
Time
38559
38688
Trigger view
Plot 0
Amplitude
Time
Message points
dbg lamp
TrueCnt
0
Message
-. ... . .-.. .-.. ---
Clear String
Clear

5. Bibliografie

[1]: Istoria Codului Morse. Care a fost primul mesaj transmis prin intermediul celui mai cunoscut limbaj cifrat

[2]: Telegraph Key - Let's Talk Science

[3]: NI 6281 Specifications - NI