# SISTEME DE POZIȚIONARE A ȚINTELOR PENTRU EXPERIMENTE DE ACCELERARE CU LASER A FASCICULELOR DE ELECTRONI ȘI PROTONI

# TARGET POSITIONING SYSTEMS FOR LASER ACCELERATION EXPERIMENTS OF ELECTRON AND PHOTON BEAMS

MIHALCEA Alexandru
Facultatea: IIR, Specializarea: NSN, Anul de studii: IV, e-mail: mihalceaalex@yahoo.com

Conducător ştiinţific: Conf.dr.ing. **Elena LĂCĂTUȘ**

*ABSTRACT: The aim of this paper is to configure the motorized target positioning system for laser accelerated electron and proton beam experiments with PW class lasers. Since any electronic system is subjected to ionizing radiation exposure during the experiments, as well as sensitive to intense electromagnetic pulses, the control unit for the motorized stages was conceived as small as possible, based on Raspberry PI single board computer, in order to optimize the footprint and its electromagnetic shielding. Consequently, the command control software was adapted to the hardware limitations and it was developed based on Python 3 scripts in order to remotely control the target positioning.*

*CUVINTE CHEIE: sistem de poziționare, poziționare ținte, Python*

## 1. Introducere

Lucrarea are ca scop prezentarea unor soluții originale pentru optimizarea sistemului de pozitionare a tintelor in experimentele de acceleare cu laser a particulelor, electroni si protoni. In majoritatea laboratoarelor de cercetare programele software de comanda si control sunt realizate în LabView, program prezent pe platforma Windows ce ruleză in general pe statii ce lucru de tip PC-uri. Utilizarea unei statii PC impune utilizarea unor solutii pentru ecranarea echipamentelor electronice, expuse la pulsuri electromagnetice (EMP) din timpul experimentelor de accelerare laser. Utilizarea echipamentelor si dispozitivelor elecronice complexe in proximitatea incintei experimentale cu laser de clasa PW duce la întreruperi in funcționarea sistemului și conexiunea cu calculatorul de comandă. Acest lucru îngreunează pornirea și rularea sistemului limitând numărul de pulsuri laser pe oră. Acest lucru duce la creștea costurilor de întreținere și folosire a întregii infrastructuri de accelerare cu laser.

Obiectivul acestei lucrări de cercetare este de a configura sistemul de poziționare motorizată a țintelor pentru a eficientiza întregul proces și pentru a reduce costul de întreținere și de funcționare.

Acest obiectiv a fost atins realizând un program similar celui ce funcționează până în momentul de față în institut, în Python, limbaj de programare prietenos cu utilizatorul și înlocuirea calculatorului care rula programul de poziționare cu calculator single-board, platforma aleasă fiind Raspberry Pi.

## 2. Stadiul actual

În momentul de față sistemul de poziționare al țintelor este în fază experimentală, întrucât nu a fost testată funcționarea acestuia în vid. Programul este funcțional, lucrându-se la optimizarea acestuia și la adăugarea funcției de aliniere automată a țintelor în fasciculul laser, iar momentan în atmosferă obișnuită.

Motivația provine din faptul că aceste experimente de accelerare cu laser a fasciculelor de electroni și protoni se realizează într-o incintă vidată la circa $10^{-6}$ mbar, condiție în care sistemul de poziționare și de diagnosticare trebuie automatizat complet.

**Tabelul 1. Axe motorizare S2C-ROBOT-PWLaser**

| Axele motorizate folosite | Rezoluție pas | Determinări | |
|---|---|---|---|
| | | Analitic | Experimental |
| 8MT160V-300XY | 2.5 µm | 2.5 µm | 2.5 µm |
| | | | |
| 8MT175V-200-VSS42 | 2.5 µm | 2.5 µm | 2.5 µm |
| 8MT173V-30-VSS42 | 1.25 µm | 1.25 µm | 1.25 µm |
| 8MR190V-2-VSS42 | 0.01° | 0.01° | 0.01° |

Atât axele motorizate cât și suporturile sunt fabricate astfel încât sa poată funcționa în parametri optimi în condițiile experimentale de vid. Axele fiind achiziționate de pe site-ul producătorului Standa.

Fiecare axă motorizată folosește la bază principiul șurub-piulită, transformând mișcarea de rotație a șurubului micrometric în mișcare de translație. Șurubul micrometric face parte din rotorul motoarelor pas cu pas. Regimul de alimentare al acestora fiind de maxim 42V la 1.2A.

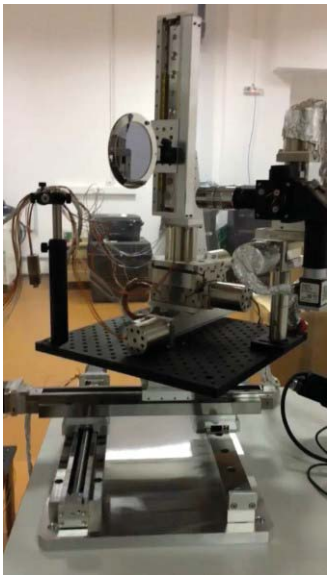

Fig. 1. Robot poziționare ținte

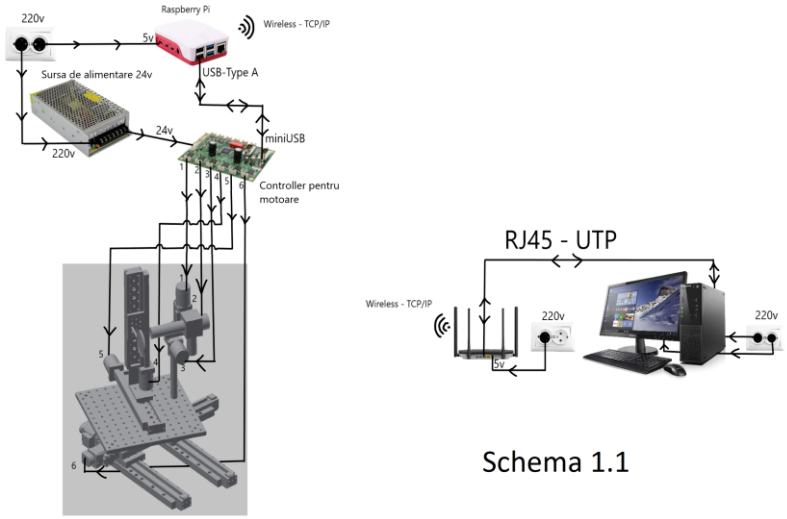


Fig. 2. Schemă experimentală Robot PWLaser

Întreg ansamblul de comandă este compus din:

- Controller motoare pas cu pas TMCM 6110 ce poate comanda până la 6 motoare, având în vedere faptul ca robotul foloseste 11 axe motorizate, se vor folosi 2 controllere pentru utilizarea intregului robot.
- Un calculator single-board Raspberry Pi, calculator unde va rula programul realizat in urma stagiului de practica
- O sursa de alimentare de 24V pentru controllerul TMCM
- O sursa de alimentare 5V pentru Raspberry Pi

Toate acestea sunt configurate conform schemei experimentale din figura 2.

Raspberry Pi-ul funcționează sub forma unui server unde va rula programul la pornire, însa acesta trebuie în primul rând să primească toate comenzile de poziționare din partea unui calculator de comandă unde se alfă interfața cu utilizatorul. Legatura între Raspberry Pi și calculatorul de comandă se realizează prin intermediul protocolului TCP/IP, mai precis prin intermediul unei conexiuni directe către rețeaua internă prin wi-fi, fie direct prin cablu ethernet.

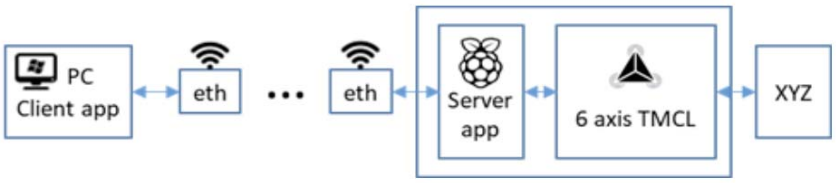


Fig. 3. Principiul de funcționare al sistemului de poziționare

## 3. Realizarea programului de comandă cu ajutorul limbajului de programare Python

Programul de comandă realizat în LabView a fost conceput având la baza funcții simple precum:

- Comandă de mișcare relativă
- Comandă de mișcare absolută
- Comandă pentru oprirea tuturor axelor motorizate
- Comandă pentru returnarea poziției actuale
- Setarea unei noi viteze de funcționare (viteza de lucru)
- Precum și o comandă pentru calibrarea fiecărei axe motorizare cu ajutorul unei funcții « home »

Producătorul controllerului TMCM a furnizat o librărie pentru comenzile ce pot fi adresate acestuia, însa aceasta este funcțională numai pe o versiune anterioară Python, lucru care necesită a fi remediat pentru a rula pe ultima versiune : Python 3.

Comenzile echivalente programului LabView:

- Stop
- Move_r
- Move_a
- Home
- Set_vel
- Get_pos

## 4. Implementarea comenzilor

Principiul de funcționare al acestui program constă în realizarea unei funcții pentru fiecare comandă enumerată anterior în Capitolul 3. Pe lângă acestea, o funcție « citește » verifică în permanență

protocolul TCP/IP și citește orice comandă primită, o interpretează și o execută în conformitate cu mesajul primit de la panoul de comandă.

Funcția « stop », este realizată pentru a întrerupe rularea oricărei alte comenzi deja executate astfel prevenind rularea unei comenzi introdusă greșit.

```
if command == 'stop':
    controller = StepRocker(24, port='/dev/ttyACM0')
    controller.TMCL.mst(0)
    controller.TMCL.mst(1)
    controller.TMCL.mst(2)
    controller.TMCL.mst(3)
    controller.TMCL.mst(4)
    controller.TMCL.mst(5)
    print('Motoarele au fost oprite!')
    return
```

Fig. 4. Comanda de oprire

Funcțiile « move_r » și « move_a » constau în mișcarea relativă și respectiv absolută, astfel se poate în orice moment monitoriza poziția fiecărei componente și să se analizeze datele obținute.

```
if command=='move_r':
    global reply
    arg1=dataMessage[3]
    if int(arg1)==1:
        nr_tmcl=0
    elif int(arg1)==2:
        nr_tmcl=1
    else:
        print('Numarul controllerelui nu este corect!')
        reply=('Numarul controllerelui nu este corect!')
        conn.send(bytes(reply, "utf-8"))
        return

    if len(data_list[0])==len(data_list[1]):
        for i in data_list[0]:
            index_motor=data_list[0].index(i)
            valoare=data_list[1][index_motor]
            controller = StepRocker(24, port='/dev/ttyACM'+str(nr_tmcl))
            controller.TMCL.mvp(i-1,'REL', valoare)
            print('Valoarea motorului '+str(i)+' este: ' + str(valoare))
            reply=('Valoarea motorului '+str(i)+' este: ' + str(valoare)+'\n')
            conn.send(bytes(reply, "utf-8"))
```

Fig. 5. Comanda pentru mișcare relativă

Funcția home constă în parcurgerea întregii curse a fiecărei axe motorizate până la capătul acesteia urmând ca mai apoi programul să memoreze acea poziție drept punctul de plecare (poziția 0).

```
elif command == 'home':
    if arg_nr==3:
        arg1=dataMessage[3]
        if int(arg1)==1:
            nr_tmcl=0
        elif int(arg1)==2:
            nr_tmcl=1
        else:
            print('Numarul controllerelui nu este corect! (3 argumente)')
            reply=('Numarul controllerelui nu este corect! (3 argumente)')
            conn.send(bytes(reply, "utf-8"))
            return
        if len(data_list[0])==len(data_list[1]):
            for i in data_list[0]:
                index_motor=data_list[0].index(i)
                valoare=data_list[1][index_motor]
                home(i,valoare,nr_tmcl)
        else:
            print('Valorile nu corespund numarului de motoare!')
            reply=('Valorile nu corespund numarului de motoare!')
            conn.send(bytes(reply, "utf-8"))
    if arg_nr==2:
        arg1=dataMessage[2]
        if int(arg1)==1:
            nr_tmcl=0
        elif int(arg1)==2:
            nr_tmcl=1
        else:
            print('Numarul controllerelui nu este corect! (2 argumente)')
            reply=('Numarul controllerelui nu este corect! (2 argumente)')
            conn.send(bytes(reply, "utf-8"))
            return
        for t in data_list[0]:
            mnl=t
            side=0
            home(mnl,side,nr_tmcl)
```

Fig. 6. Comanda pentru reinițializarea axelor

comanda nr. axei nr. controller

```
'get_pos [1,2,3,4,5,6] 1'
```

Fig. 7. Comanda pentru returnarea poziției axelor

comanda nr. axa viteza motor nr. controller

```
'set_val [1,2] [1000,1000] 1'
```

Fig. 8. Configurarea vitezei pentru fiexare axa motorizată

Folosind panoul de comandă se realizează întâi de toate, conexiunea către Raspberry Pi urmând ca numai după deschiderea unui canal între cele doua dispozitive, să se poată raliza transferal de date.


Fig. 9. Interfața de conectare TCP/IP panou comandă

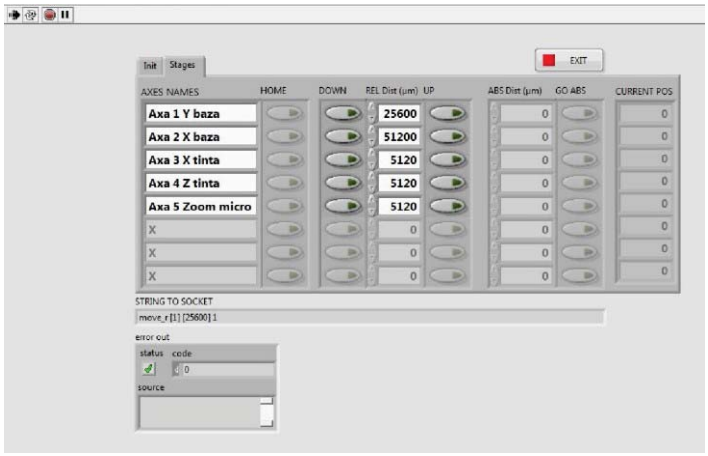


Fig. 10. Interfața de comunicare panou comandă

Interfața grafică realizată cu ajutorul programului LabView, ce rulează pe calculatorul de comandă poate fi folosiă și pentru transmiterea comenzilor către Raspberry Pi și pot fi interpretate în conformitate pentru poziționarea țintei în cadrul experimentelor de accelerare de electroni și protoni.

## 5. Tipuri de ținte folosite

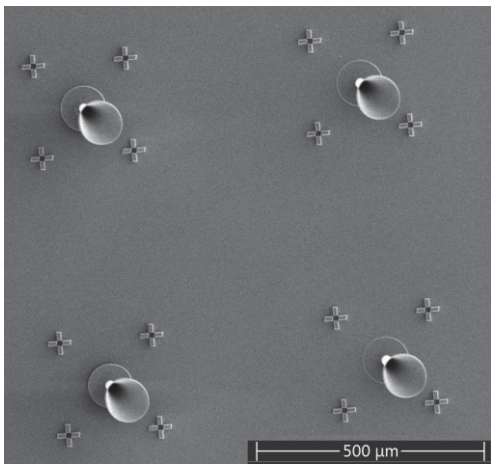


Fig. 11. Suport de siliciu cu repere tip cruce și tinte laser cu forma conică pentru direcționarea fasciculului laser

Acest tip de ținte sunt folosite deoarece subiectul experimentului se poate realiza cu mare precizie la baza conului, folosind cele patru repere de tip « cruce ». În centrul acestora se află o membrană, cu grosime de ordinul a zeci de micrometri. Pentru concentrarea fasciculi laser către această membrană se folosesc construcții de tip con, înclinate conform unghiului de atac al fasciculului laser pentru a direcționa fasciculul către aceasta fereastră.

## 6. Concluzii

Implementarea software este un proces complex, însă datorită aspectului modular al programului, acesta poate fi foarte ușor adaptat, îmbunătățit și optimizat.

Procesul de implementarea a unui program Python substituent pentru cel realizat cu ajutorul programului LabView conduce către scăderea considerabilă a timpului de pornire, de executare și de răspuns între calculatorul de comandă și robot-ul propriu-zis. Pe lângă acest lucru, ecranarea unui dispozitiv de dimensiunile acestui calculator single-board se poate realiza mult mai ușor și conduce către menținerea funcționării echipamentului de control și în timpului experimentelor, spre deosebire de folosirea unui calculator clasic, unde înainte de realizarea fiecărui impuls laser, trebuie închis întregul echipament de comandă, având posibilitate numai de a observa experimentul nu și de a interveni asupra acestuia.

## 8. Bibliografie

[1]. Python, librării și exemple https://docs.python.org/3/library/index.html

## 7. Notații

Următoarele prescurtări sunt utilizate în cadrul lucrării:

S2C-ROBOT-PWLaser – Sistem de Comandă și Control – ROBOT- poziționare-ținte-PWLaser