Universitatea “Politehnica” din București

Facultatea de Electronică, Telecomunicații și Tehnologia Informației

**Robot autonom cu braț mecanic și cameră video**

Prezentat ca cerință parțială pentru obținerea titlului de

Inginer în domeniul *Electronică, Telecomunicaţii şi Tehnologia Informaţiei*

programul de studii de licență *Tehnologii și Sisteme de Telecomunicații*

Conducător științific Absolvent

Conf.Dr.Ing. Octaviana DATCU Teodor LINA

București 2021

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | Universitatea “Politehnica” din Bucureşti Facultatea de Electronică, Telecomunicaţii şi Tehnologia Informaţiei Program de studiu **TST** |  | |
|  |
| **TEMA PROIECTULUI DE DIPLOMĂ** |
| a studentului **LINA E. Teodor , 442C-TST** |
|  |
| **1. Titlul temei:** Robot autonom cu braț mecanic și cameră video  **2. Descrierea temei și a contribuției personale a studentului (în afara părții de documentare):** Robotul care este proiectat poate fi utilizat pentru orice aplicație care presupune recunoaștere video și ridicarea unor componente pentru a le depozita în altă locație decât cea în care se află la momentul inițial. Spre exemplu, sortarea unor componente pasive aflate într-un laborator de măsurări.  Studentul: (1) realizează schemele bloc pentru robot și pentru telecomanda cu care se controlează robotul. (2) alege modulele de comunicație radio în funcție de specificațiile necesare temei (raza de acțiune, cantitatea de informație ce se poate transmite într-un pachet) (3) alege motoarele DC corespunzătoare, astfel ca robotul să se poată deplasa conform cu specificațiile. (4) alege servo-motoarele necesare deplasării brațului robotic in 4 direcții. (5)alege un modul de cameră compatibil cu placa de dezvoltare Raspberry Pi. (6) proiectează PCB-ul (printed circuit board) pentru telecomandă. (7) scrie software-ul pentru robot si telecomandă - implementează funcțiile de mișcare a robotului, comunicația wireless între dispozitive folosind modulele de comunicație radio alese. (8) concluzionează asupra funcționalității robotului proiectat și propune direcții de dezvoltare ulterioară.  **3. Discipline necesare pt. proiect:** Programarea calculatoarelor, Microcontrolere, Tehnici CAD, IRPSE (Inițiere în realizarea practică a schemelor electronice)  **4. Data înregistrării temei:**2020-11-28 23:30:27 |
| |  |  |  | | --- | --- | --- | | |  |  | | --- | --- | | **Conducător lucrare,** Conf.Dr.Ing. Octaviana DATCU | **Student,** LINA E. Teodor | | | |  | | --- | |  | | | |  |  | | --- | --- | | **Director departament,** Conf. dr. ing. Șerban OBREJA | **Decan,** Prof. dr. ing. Mihnea UDREA | | |
| |  | | --- | | Cod Validare: **d920929d55** | |

**Declarație de onestitate academică**

Prin prezenta declar că lucrarea cu titlul “ *Robot autonom cu braț mecanic și cameră video*”, prezentată în cadrul Facultății de Electronică, Telecomunicații și Tehnologia Informației a Universității “Politehnica” din București ca cerință parțială pentru obținerea titlului de Inginer în domeniul Electronică, Telecomunicaţii şi Tehnologia Informaţiei, programul de studii Tehnologii și Sisteme de Telecomunicații este scrisă de mine și nu a mai fost prezentată niciodată la o facultate sau instituție de învățămînt superior din țară sau străinătate.

Declar că toate sursele utilizate, inclusiv cele de pe Internet, sunt indicate în lucrare, ca referințe bibliografice. Fragmentele de text din alte surse, reproduse exact, chiar și în traducere proprie din altă limbă, sunt scrise între ghilimele și fac referință la sursă. Reformularea în cuvinte proprii a textelor scrise de către alți autori face referință la sursă. Înțeleg că plagiatul constituie infracțiune și se sancționează conform legilor în vigoare.

Declar că toate rezultatele simulărilor, experimentelor și măsurătorilor pe care le prezint ca fiind făcute de mine, precum și metodele prin care au fost obținute, sunt reale și provin din respectivele simulări, experimente și măsurători. Înțeleg că falsificarea datelor și rezultatelor constituie fraudă și se sancționează conform regulamentelor în vigoare.

București, 26.06.2021

Absolvent *Teodor LINA*



(semnătura în original)

# CUPRINS

[CUPRINS 7](#_Toc75972529)

[LISTA ACRONIMELOR 9](#_Toc75972530)

[LISTA FIGURILOR 11](#_Toc75972531)

[LISTA TABELELOR 13](#_Toc75972532)

[INTRODUCERE 15](#_Toc75972533)

[CAPITOLUL 1 DESCRIERE HARDWARE 17](#_Toc75972534)

[1.1 BOM 17](#_Toc75972535)

[1.2 Programul EAGLE 18](#_Toc75972536)

[1.3 Scheme bloc și *layout* realizate în programul EAGLE 19](#_Toc75972537)

[1.4 Raspberry Pi 22](#_Toc75972538)

[1.4.1 Descriere generală 22](#_Toc75972539)

[1.4.2 Configurarea Raspberry Pi 23](#_Toc75972540)

[1.4.3 Accesul de la distanță la Raspberry Pi cu ajutorul RealVNC Viewer 24](#_Toc75972541)

[1.5 Arduino și Microcontrolerul ATmega328P 25](#_Toc75972542)

[1.6 Modulele NRF24l01+ 26](#_Toc75972543)

[1.6.1 Descriere generală 26](#_Toc75972544)

[1.6.2 Modul de funcționare 27](#_Toc75972545)

[1.7 Motoare 28](#_Toc75972546)

[1.7.1 Motoare stepper DC 3V 28](#_Toc75972547)

[1.7.2 Micro servo-motoare SG90 29](#_Toc75972548)

[1.8 Puntea H duală L298N 29](#_Toc75972549)

[1.9 Modul cameră compatibil cu Raspberry Pi 31](#_Toc75972550)

[1.10 Coborâtor de tensiune LM2596 32](#_Toc75972551)

[1.11 Joystick biaxial 33](#_Toc75972552)

[1.12 Senzorul DHT 11 33](#_Toc75972553)

[1.13 LCD alfanumeric 16x02 caractere 34](#_Toc75972554)

[1.14 Acumulatori Li-ion 18650 34](#_Toc75972555)

[CAPITOLUL 2 DESCRIERE SOFTWARE 37](#_Toc75972556)

[2.1 Generalități limbaje de programare 37](#_Toc75972557)

[2.2 C++ 37](#_Toc75972558)

[2.2.1 Generalități 37](#_Toc75972559)

[2.2.2 Biblioteci folosite 38](#_Toc75972560)

[2.3 Python 40](#_Toc75972561)

[2.3.1 Generalități 40](#_Toc75972562)

[2.3.2 Biblioteci folosite 40](#_Toc75972563)

[2.4 HTML 41](#_Toc75972564)

[CAPITOLUL 3 CONSTRUCȚIA ȘI MODUL DE FUNCȚIONARE AL ROBOTULUI 43](#_Toc75972565)

[3.1 Construcția telecomenzii 43](#_Toc75972566)

[3.2 Construcția brațului mecanic 43](#_Toc75972567)

[3.2.1 Prima parte a brațului mecanic 44](#_Toc75972568)

[3.2.2 A doua parte a brațului mecanic 45](#_Toc75972569)

[3.2.3 A treia parte a brațului mecanic 45](#_Toc75972570)

[3.3 Construcția robotului 47](#_Toc75972571)

[3.3.1 Asamblarea kit-ului robotului 47](#_Toc75972572)

[3.3.2 Conectarea motoarelor DC la puntea H L298N 49](#_Toc75972573)

[3.3.3 Realizarea conectării componentelor robotului 49](#_Toc75972574)

[3.4 Modul de funcționare 51](#_Toc75972575)

[CONCLUZII ȘI VIITOARE ÎMBUNĂTĂȚIRI 57](#_Toc75972576)

[BIBLIOGRAFIE 59](#_Toc75972577)

[ANEXA 1 COD SURSĂ 63](#_Toc75972578)

# LISTA ACRONIMELOR

MCU – Microcontroller Uni – Unitatea Microcontrolerului

MPU – Central Processing Unit – Unitatea Microprocesorului

PCB – Printed Circuit Board – Placă cu Circuit Imprimat

HDMI – High Definition Multimedia Interface – Interfață Multimedia de Înaltă Definiție

AV – Audio Visual – Audio Vizual

USB – Universal Serial Bus – Magistrală Serială Universală

LCD – Liquid Crystal Display – Ecran cu Cristale Lichide

BOM – Bill of Materials – Listă Completă de Componente

EDA – Electronic Design Automation – Automatizarea Proiectării Electronice

CAM – Computer Aided Manufacturing – Fabricare Asistată de Calculator

CAD – Computer Aided Design – Proiectare Asistată de Calculator

SBC – Single Board Computer – Computer cu o Singură Placă

CPU – Central Processing Unit – Unitate Centrală de Prelucrare

CSI – Camera Serial Interface – Interfață Serială pentru Cameră

SD – Secure Digital – Siguranță Digitală

RAM – Random Access Memory – Memorie cu Acces Aleator

GPIO – General Purpose Input/Output – Pini de Intrare/Ieșire de Uz General

I/O – Input/Output – Intrare/Ieșire

VNC – Virtual Network Computing – Calcul de Rețea Virtuală

SRAM – Static Random Access Memory – Memorie Statică cu Acces Aleator

EEPROM – Electrically Erasable Programable Read-Only Memory – Memorie Electrică ROM Reprogramabilă

ROM – Read Only Memory – Memorie Pentru Citire

IDE – Integrated Development Environment – Mediu de Dezvoltare Integrat

RF – Radio Frequency – Radio Frecvență

ISM – Industrial, Scientific, and Medical – Industrial, Știinșific și Medical, Bandă de frecvență ISM

GFSK – Gaussian Frequency Shift Keying – Modulație în Frecvență de tip GFSK

PA/LNA – Power Amplifier/ Low Noise Amplifier – Amplificator de Putere/ Amplificator cu Zgomot Redus

ESB – Enhanced Shock Burst – Protocol Sporit de Șoc

ACK – Acknowledgement – Recunoaștere

DC – Dirrect Current – Curent Continuu

PWM – Pulse Width Modulation – Modulația Lățimii Impulsului

NTC – Negative Temperature Coefficient – Coeficient Negativ al Temperaturii

I2C – Eye-Squared-C – Interfață de Comunicație Serială I2C

SPI – Serial Peripheral Interface – Interfață Serială Periferică

PIP – Package Manager – Manager de Pachete/Biblioteci

HTTP – Hypertext Transfer Protocol – Protocol de Transfer Hipertext

HTML – Hypertext Markup Language – Limbaj de Marcare Hipertext

RPM – Revolutions Per Minute – Rotații pe Minut

MJPEG – Motion Joint Photographic Experts Group

THT – Through Hole Technology – Tehnologie prin Găurire

WWW/WEB – World Wide Web – Internet

HTTPS – Hyper Transfer Protocol Secure – Protocol de Transfer Hipertext Sigur

MPEG – Moving Picture Experts Group

JPEG – Joint Photographic Experts Group

WI-FI – Wireless Fidelity – Internet WI – FI

ASCII – American Standard Code For Information Interchange – Codul American Standard pentru Schimbul de Informații

ERC – Electric Rules Check – Verificarea Regulilor Electrice

DRC – Design Rules Check – Verificarea Regulilor de Proiectare

# 

# LISTA FIGURILOR

[Fig. 1. 1 Componente Robot. 17](#_Toc76046809)

[Fig. 1. 2 Componente Telecomandă. 18](#_Toc76046810)

[Fig. 1. 3Programul CAM/CAD EAGLE. [11] 19](#_Toc76046811)

[Fig. 1. 4 Schemă Bloc Robot din Programul EAGLE. 20](#_Toc76046812)

[Fig. 1. 5Schemă Bloc Telecomandă din programul EAGLE. 21](#_Toc76046813)

[Fig. 1. 6*Layout* Telecomandă din programul EAGLE. 21](#_Toc76046814)

[Fig. 1. 7Raspberry Pi model 3A+. [17] 22](#_Toc76046815)

[Fig. 1. 8 Afișare mesaj Hello World! în Thonny IDE 24](#_Toc76046816)

[Fig. 1. 9Interfețe Raspberry Pi. 24](#_Toc76046817)

[Fig. 1. 10Autentificare Raspberry Pi. 25](#_Toc76046818)

[Fig. 1. 11Placă de dezvoltare Arduino Nano. [20] 26](#_Toc76046819)

[Fig. 1. 12 Transmițător radio NRF24L01+. [21] 26](#_Toc76046820)

[Fig. 1. 13Protocolul ESB. [21] 27](#_Toc76046821)

[Fig. 1. 14Principiu funcționare motor DC. [22] 28](#_Toc76046822)

[Fig. 1. 15 Servo–motor SG90.[23] 29](#_Toc76046823)

[Fig. 1. 16Semnale de tip PWM.[24] 30](#_Toc76046824)

[Fig. 1. 17Punte H. 30](#_Toc76046825)

[Fig. 1. 18Verificare funcționare modul de cameră. 32](#_Toc76046826)

[Fig. 1. 19Coborâtor de tensiune LM2596. 32](#_Toc76046827)

[Fig. 1. 20 Modul Joystick Biaxial. [29] 33](#_Toc76046828)

[Fig. 1. 21 Mod de funcționare senzor DHT11. [30] 34](#_Toc76046829)

[Fig. 1. 22Acumulatori Li-ion 18650 3.7V. 35](#_Toc76046830)

[Fig. 2. 1C++ [33] 37](#_Toc75632863)

[Fig. 2. 2Deschidere manager de biblioteci în Arduino Software(IDE) 38](#_Toc75632864)

[Fig. 2. 3Instalarea unei biblioteci în Arduino Software(IDE) 39](#_Toc75632865)

[Fig. 2. 4Adăugarea unei biblioteci în Arduino Software(IDE) manual 39](#_Toc75632866)

[Fig. 2. 5Python 3 [34] 40](#_Toc75632867)

[Fig. 2. 6Exemplu format HTML [36] 42](#_Toc75632868)

[Fig. 3. 1Telecomanda robotului 43](#_Toc75680906)

[Fig. 3. 2Structură braț mecanic 44](#_Toc75680907)

[Fig. 3. 3 Bază braț mecanic 44](#_Toc75680908)

[Fig. 3. 4Centru braț mecanic 45](#_Toc75680909)

[Fig. 3. 5Clește braț mecanic 46](#_Toc75680910)

[Fig. 3. 6Brațul mecanic complet 46](#_Toc75680911)

[Fig. 3. 7Kit robot 47](#_Toc75680912)

[Fig. 3. 8Asamblare kit robot. [37] 48](#_Toc75680913)

[Fig. 3. 9Conectare motoare DC la puntea H [24] 49](#_Toc75680914)

[Fig. 3. 10Robot complet asamblat 50](#_Toc75680915)

[Fig. 3. 11 Datele transmise de ATmega328P în monitorul serial din Arduino Software(IDE) 53](#_Toc75680916)

[Fig. 3. 12Datele recepționate și transmise de Raspberry Pi în consola din Thonny IDE 54](#_Toc75680917)

[Fig. 3. 13Afișare măsurători temperatură și umiditate pe ecranul LCD 54](#_Toc75680918)

[Fig. 3. 14Captura unui cadru din *stream-ul* video transmis de cameră 55](#_Toc75680919)

# LISTA TABELELOR

[Tabel I Listă componente folosite 17](#_Toc75633412)

[Tabel II Specificații SG90[23] 29](#_Toc75633413)

# 

# INTRODUCERE

Motivație

Aplicațiile realizate cu robotica mobilă au permis de-a lungul anilor multor industrii să profite de beneficiile pe care automatizarea a adus-o liniilor de producție. Robotica mobilă a permis crearea de spații de lucru mult mai sigure și mai eficiente, reușind să asigure în același timp o productivitate mai ridicată. Aplicațiile care folosesc robotica mobilă au ca scop optimizarea proceselor care s-au dovedit a fi periculoase sau repetitive pentru om. Oamenii pot lucra în siguranță într-un spațiu de lucru comun cu roboții, deoarece fiecare robot este echipat cu senzori de detecție a mișcării, care detectează când un om este în apropiere. Chiar dacă sunt roboți autonomi care se operează de la distanță, procesul pe care aceștia îl parcurg este controlat de oamenii care îi programează. Sunt numeroase zone în care robotica mobilă a adus o îmbunătățire considerabilă cum ar fi: în procesele de mentenanță, în horticultură, în îmbunătățirea calității vieții persoanelor cu dizabilități, în probleme de logistică etc.

Având în vedere dezvoltarea continuă în robotica mobilă, motivația lucrării a fost în primul rând, realizarea unui robot mobil autonom care să se poată deplasa în locuri greu accesibile omului. În acest fel se pot evita accidentele și punerea în pericol a oamenilor. Robotul poate fi operat de la o distanță de maxim 100 m și poate reda în timp real imagini de la fața locului. Brațul mecanic poate fi folosit pentru a apuca obiecte din zonele prin care se deplasează robotul. În al doilea rând, acest proiect poate fi folosit în scop didactic în lucrări de laborator, pentru familiarizarea studenților cu plăci de dezvoltare, cum ar fi Raspberry Pi, cu microcontrolere, cum ar fi ATmega328P, cu interfețe de comunicație, motoare, pe scurt cu ideea de robotică. Pentru dezvoltarea acestui proiect au fost consultate inițial cinci lucrări științifice din domeniu, cu temă asemănătoare.

Lucrări de referință din care a fost dezvoltat proiectul

În lucrarea [1] a fost dezvoltat un robot mobil format dintr-un kit cu patru roți și patru motoare, comandat cu ajutorul plăcilor de dezvoltare Raspberry Pi și Arduino Uno. Robotul conține de asemenea și un braț mecanic capabil de deplasare pe o axă. Proiectul dezvoltat în această lucrare este alcătuit dintr-un kit cu două roți și două motoare, este comandat cu ajutorul plăcii de dezvoltare Raspberry Pi și a microcontrolerului ATmega328P și conține un braț mecanic capabil de deplasare pe două axe. Brațul poate apuca și muta obiecte ușoare.

Lucrările [2] și [3] prezintă dezvoltarea un braț robotic capabil să sorteze obiecte. Proiectele au în componență și un modul de cameră. Atât brațul, cât și camera sunt comandate de o placă de dezvoltare Raspberry Pi. Modulul de cameră este folosit pentru a captura imagini ale obiectelor. Imaginile sunt procesate cu ajutorul limbajului de programare GNU Octave, în lucrarea [2] și cu ajutorul limbajului Matlab, în lucrarea [3], pentru a determina forma și culoarea obiectelor. În actualul proiect, modulul de cameră este folosit pentru a realiza un *stream* video ce poate fi accesat într-o pagină de *web*. Prelucrarea cadrelor și redarea *stream-ului* video se realizează în limbajul de programare Python 3.

Lucrarea [4] realizează controlul robotului de la distanță cu ajutorul unei aplicații Android. Se folosește Wi-Fi pentru stabilirea comunicației între robot și *smart phone*. În proiectul prezent, controlarea de la distanță se realizează cu ajutorul unor transmițătoare radio, pentru stabilirea comunicației între robot și telecomandă.

În lucrarea [5] robotul dezvoltat poate fi acționat până la o distanță de maxim 20 de metri și poate funcționa autonom timp de 30 de minute.Robotul dezvoltat în prezentul proiect poate fi operat până la o distanță de maxim 90–100 de metri și poate funcționa autonom pentru o perioadă de aproximativ 4 ore.Proiectul din această lucrare conține atât o parte de hardware, cât și o parte de software, iar modul în care acestea sunt împărțite este explicat în cele ce urmeză.

Structura proiectului

Acest proiect este împărțit în trei capitole, completate de introducere, concluzii și direcții viitoare de dezvoltare. În primul capitol se prezintă în detaliu componentele folosite la dezvoltarea proiectului, schemele bloc pentru robot și pentru telecomandă realizate în programul de CAD EAGLE, *layout*-*ul* pentu telecomandă realizat tot în EAGLE. În al doilea capitol sunt descrise limbajele de programare folosite în realizarea codului sursă pentru robot și pentru telecomandă, împreună cu bibliotecile care s-au folosit în fiecare limbaj de programare. Se folosesc limbajele de programare C++, Python 3 și limbajul de marcare HTML, care se folosește strict pentru partea de *stream* a camerei video. În capitolul trei se prezintă modul în care a fost asamblat robotul, construcția brațului mecanic, modul în care a fost asamblat cablajul imprimat al telecomenzii și modul de funcționare al acestora.

Proiectul este alcătuit din două circuite, care împreună realizează produsul final. Primul circuit este telecomanda, cu care se controlează funcțiile ce trebuie îndeplinite de robot. Acest circuit are la bază un MCU și anume placa de dezvoltare Arduino Nano, care conține un microcontroler ATmega328P. Pentru telecomandă s-a realizat un PCB. PCB-ul are rolul de a susține mecanic și de a conecta electric o serie de componente pentru a realiza un produs final funcțional. Acest lucru este important în electronică, deoarece oferă un design compact, ușurință în testare, se evită scurtcircuitele, se evită golurile, oferă o protecție mult mai bună împotriva zgomotului și a interferențelor, deoarece nu există jumpere sau fire și nu în ultimul rând în acest mod se crește fiabilitatea circuitului. Al doilea circuit este robotul propriu zis, care are la bază un MPU și anume placa de dezvoltare Raspberry Pi 3A+. Pentru a evidenția diferența între un microcontroler și un microprocesor, adică între ATmega328P și între Raspberry Pi, se fac următoarele observații:Raspberry Pi are nevoie de un sistem de operare pentru a rula, se comportă ca un computer, permite conectarea a numeroase dispozitive periferice, cum ar fi monitor (cu port HDMI sau AV), tastatură și mouse (prin USB), o cameră printr-o interfață dedicată, poate fi conectată la internet (prin cablu Ethernet sau prin Wi-Fi); ATmega328P nu necesită un sistem de operare pentru a rula, are nevoie doar de codul binar rezultat în urma compilării codului sursă, după cum s-a precizat în [6]-[10].

Contribuții

În acest proiect de diplomă, contribuțiile studentului constau în interconectarea și asamblarea unor dispozitive, pentru realizarea unor sisteme *hardware* și realizarea unor programe *software* care determină un răspuns al acestor sisteme *hardware*. Prin programele *software* se realizează toate funcționalitățile proiectului, cum ar fi comunicația *wireless* între telecomandă și robot sau funcțiile de deplasare ale robotului și ale brațului mecanic.

# CAPITOLUL 1 DESCRIERE HARDWARE

## BOM

Lista componentelor necesare în dezvoltarea proiectului:

Tabel Listă componente folosite



În Fig. 1.1 se pot observa componentele folosite la dezvoltarea robotului mobil.

Fig. 1. Componente Robot.

În Fig. 1.2 se pot observa componentele folosite pentru dezvoltarea telecomenzii robotului.



Fig. 1. Componente Telecomandă.

## 1.2 Programul EAGLE

EAGLE aste o aplicație de tip EDA care cuprinde numeroase instrumente necesare în dezvoltarea/crearea unui PCB funcțional. Printre aceste instrumente se numără: editor de design schematic al circuitului, editor de design *layout* al circuitului, auto-rutare și o multitudine de caracteristici CAM. În prima etapă din crearea unui PCB se utilizează editorul de design schematic care salvează fișiere cu extensia .SCH.Componentele utilizate sunt definite în biblioteci specifice care conțin dispozitivele respective cu extensia .LBR. Într-un fișier schematic creat cu acest editor se selectează din biblioteci componentele necesare,se alege un *foot-print* pentru fiecare componentă în parte, se interconectează între ele componentele și se verifică funcționarea circuitului din punct de vedere electric, prin rularea unei verificări ERC din paleta de instrumente a programului EAGLE. În a doua etapă se utilizeaza editorul de design *layout*, care salvează fișiere cu extensia .BRD. În acest editor se face practic design-ul PCB-ului, se plasează componentele în aria de lucru, se poate modifica dimensiunea plăcii, dimensiunile pentru pad-uri, găuri, se rutează traseele între componente, se modifică grosimea traseelor,se poate crea un plan de masă, un plan de alimentare, se selectează numărul de *layere* etc. După ce a doua etapă este completă și PCB-ul se află în forma finală se va genera fișierul Gerber, care se trimite ulterior la un producător pentru realizarea PCB-ului în formă fizică. Fișierul Gerber reprezintă în electronică, făcând o analogie, echivalentul pdf-ului pentru word. Un fișier Gerber este tratat ca o imagine care reprezintă un *layer* al PCB-ului, care trebuie fabricat. Se trimite fișierul Gerber și nu fișierul cu extensia .BRD pentru a nu oferi toată informația necesară pentru asamblarea PCB-ului. În acest mod nu se poate fura o idee de circuit, conform [11]-[14]. În Fig. 1.3 se poate vizualiza sigla programului de CAD EAGLE.

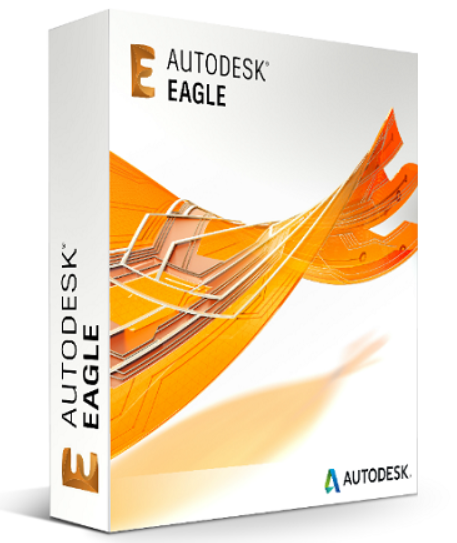


Fig. 1. Programul CAM/CAD EAGLE. [11]

## 1.3 Scheme bloc și *layout* realizate în programul EAGLE

Atât pentru robot cât și pentru telecomandă au fost realizate cu ajutorului programului de CAD, EAGLE, scheme bloc care reprezintă interconectarea componentelor electronice pentru realizarea unui circuit funcțional. Pentru telecomandă s-a realizat și un PCB conform etapelor explicate la secțiunea 1.2 din primul capitol. Toate componentele folosite fac parte din biblioteci predefinite în EAGLE sau din biblioteci externe atașate ulterior în proiect, conform [15]. După terminarea proiectării schemelor bloc, se verifică dacă acestea au greșeli din punct de vedere electric, dacă există goluri sau scurtcircuite. Verificarea din punct de vedere electric se realizează foarte ușor în EAGLE prin rularea unui ERC. În cazul PCB-ului telecomenzii, după terminarea *layout-ului*, se face o verificare de tipul DRC, în EAGLE. Prin verificarea de tipul DRC se identifică posibilele erori de spațiere între trasee sau între componente. DRC este foarte importantă în cazul circuitelor integrate, din cauză că pot avea geometrii la o scară foarte mică, de ordinul µm sau nm. În Fig. 1.4 se poate observa schema bloc a robotului realizată în EAGLE. În Fig. 1.5, respectiv Fig. 1.6 se poate vizualiza schema bloc a telecomenzii, respectiv *layout-ul* telecomenzii, realizate în EAGLE.

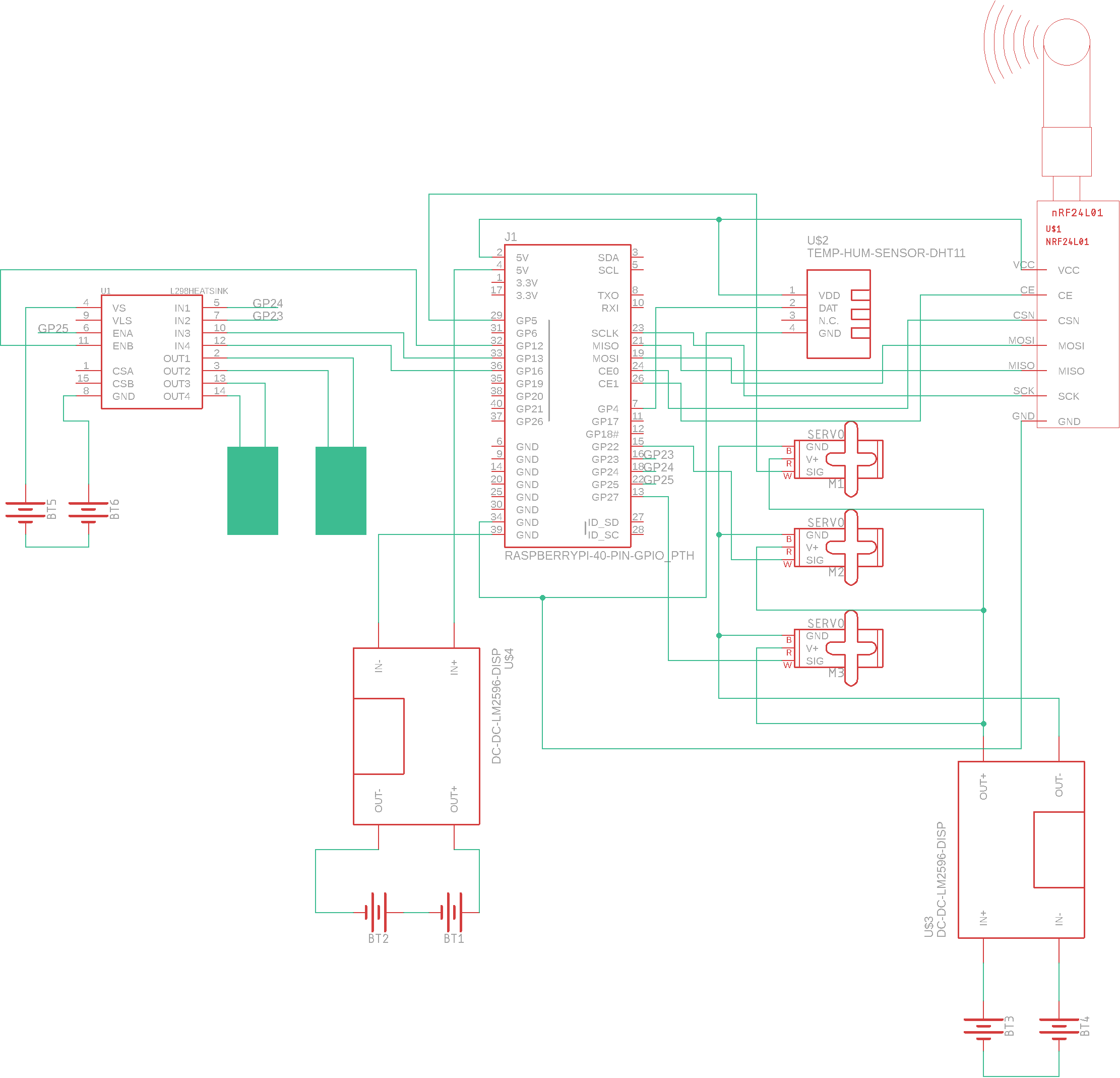


Fig. 1. Schemă Bloc Robot din Programul EAGLE.

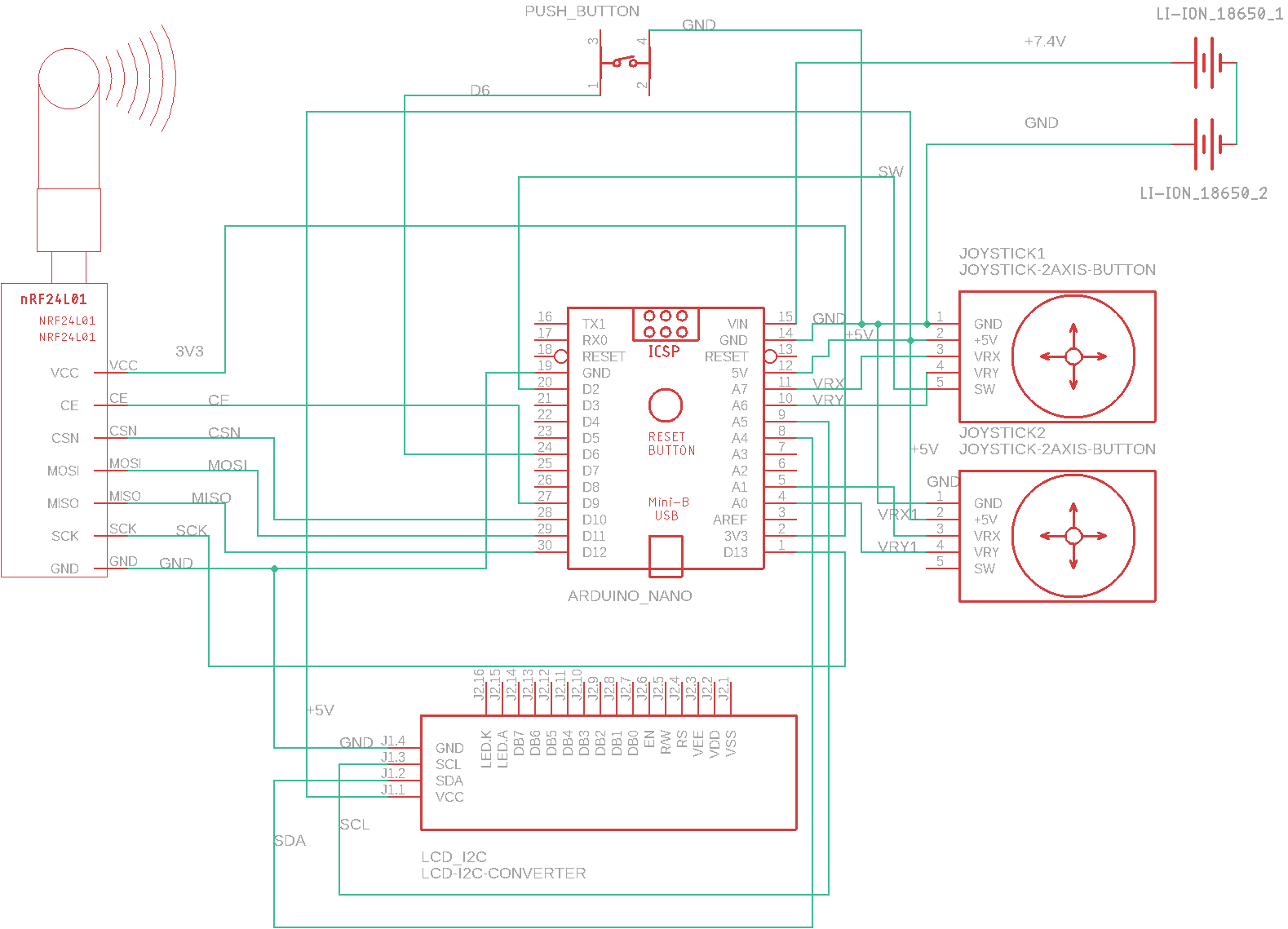
****

Fig. 1. Schemă Bloc Telecomandă din programul EAGLE.

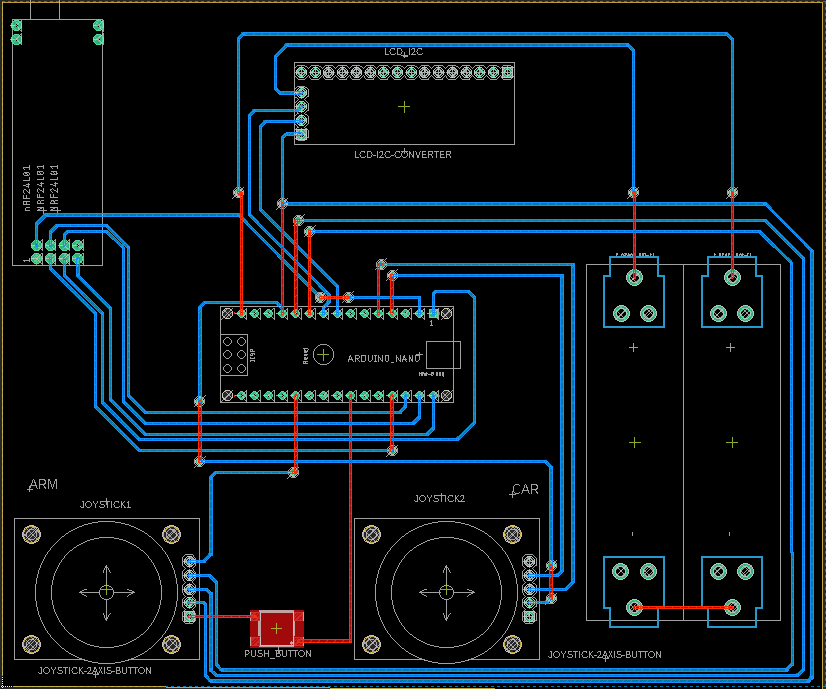
****

Fig. 1. *Layout* Telecomandă din programul EAGLE.

## 1.4 Raspberry Pi

### 1.4.1 Descriere generală

Raspberry Pi este un computer de dimensiuni și costuri reduse, care se poate conecta la un monitor sau un TV și utilizează o tastatură și un mouse standard. Deși este un dispozitiv mic, poate fi folosit pentru o multitudine de aplicații și poate fi programat folosind limbaje de scriptare precum Scratch sau Python. Este capabil să realizeze lucrurile pe care le face un computer normal, de la navigarea pe internet și redarea de video, până la procesarea de text și jucarea de jocuri video, conform [16].

În acest proiect s-a folosit modelul Raspberry Pi 3A+. Acest model a fost lansat pe piață în anul 2018. Se face remarcat prin faptul că are dimensiuni reduse, este ușor și are de asemenea un cost redus, caracteristici ce îl fac potrivit pentru acest proiect. Privind specificațiile tehnice pentru modelul 3A+, cele mai importante sunt:CPU de 64 biți cu 4 nuclee, port USB, port HDMI, conector micro USB, conector jack 3,5 mm, interfață CSI pentru cameră video, *slot* pentru card micro SD, o memorie RAM de 512 MB și 40 de pini GPIO de I/O ce pot fi controlați de utilizator în timpul rulării[17]. Placa de dezvoltare Raspberry Pi 3A+ poate fi vizualizată în Fig. 1.7.

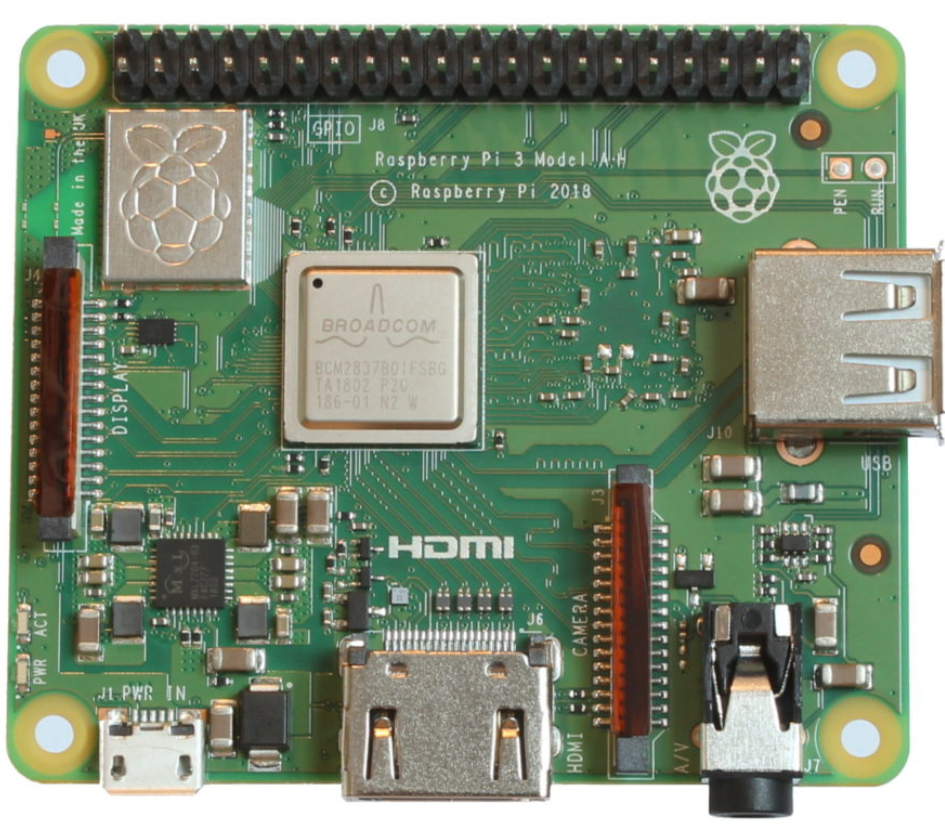


Fig. 1. Raspberry Pi model 3A+. [17]

### 1.4.2 Configurarea Raspberry Pi

Înainte de a putea folosi o placă de dezvoltare Raspberry Pi, aceasta trebuie configurată urmând niște pași. În prima etapă se realizează formatarea cardului micro SD, pe care se va instala ulterior sistemul de operare, Raspbian, necesar pentru funcționarea Raspberry Pi. Instalarea sistemului de operare se realizează cu ajutorul Raspberry Pi Imager, instalat pe un laptop/computer, deoarece este cea mai ușoară metodă de a instala sistemul de operare pe cardul micro SD. Se introduce cardul în laptop/computer în *slot-ul* pentru carduri, este necesar un cititor de card și se rulează programul Raspberry Pi Imager. Cardul se formatează, adică se șterg toate datele de pe card ireversibil. Din programul Raspberry Pi Imager utilizatorul va alege cardul pe care dorește să instaleze sistemul de operare și sistemul de operare dorit, deoarece există mai multe opțiuni. După alegerea cardului și a sistemului de operare se selectează butonul de *WRITE*. După terminarea procesului se va afișa un mesaj pentru scoaterea cardului din cititor.

În a doua etapă se introduce cardul micro SD în *slot-ul* de la Raspberry Pi, se conectează tastatura și *mouse-ul* prin porturi USB, se conectează monitorul prin portul HDMI, se conectează sursa de alimentare prin portul de micro USB. Opțional se poate conecta o pereche căști sau microfon prin conectorul de tip jack 3.5 mm. După ce toate conexiunile sunt realizate, Raspberry Pi va porni operația de *boot* și va apărea pe monitor un *desktop* grafic.

În ultima etapă, pentru a finaliza configurarea, trebuie selectate opțiunile din aplicația de bun venit Raspberry Pi, care apare la prima pornire.Se vor seta țara, limba, fusul orar, parola și se va realiza conexiunea la internet prin Wi-fi, dacă modelul de Raspberry Pi folosit are conexiune wireless. Conexiunea la internet se poate face și prin conectarea unui cablu Ethernet, care se conectează la un port Ethernet de pe placă. În final se afișează mesajul conform căruia configurarea este completă. În cazul în care s-au instalat actualizări pentru sistemul de operare va fi necesară și o operație de *reboot*, după cum s-a precizat în [18].

După ce placa de dezvoltare a executat operația de *reboot*, în cazul în care este necesară, Raspberry Pi este gata pentru a putea fi folosită de utilizator. Sistemul de operare Raspbian conține deja un IDE, pentru a putea rula programe în Python și anume Thonny IDE, vizibil în Fig. 1.8. În acest proiect se folosește Thonny IDE, deoarece conține Python 3.7 încorporat. Un avantaj al acestui IDE îl reprezintă *debugger-ul* simplist, care poate fi activat foarte ușor, prin apăsarea *Ctrl+F5* de la tastatură. În acest mod se poate parcurge un program pas cu pas, pentru a vedea exact unde este o eroare, în caz că aceasta apare în faza de dezvoltare *software*. Pentru a putea trece de la o instrucțiune la următoarea instrucțiune se apasă *F7*, de la tastatură. Un alt avantaj important al acestui IDE este reprezentat de sublinierea în cod a eventualelor erori de sintaxă.



Fig. 1. Afișare mesaj Hello World! în Thonny IDE

### 1.4.3 Accesul de la distanță la Raspberry Pi cu ajutorul RealVNC Viewer

Pentru a putea accesa într-un mod mai eficient *desktop-ul* de la Raspberry Pi, se va folosi aplicația RealVNC Viewer pe computerul de pe care se dorește să se acceseze Raspberry Pi. În prima etapă va trebui activată interfața VNC. Acest lucru se face prin comanda: *sudo raspi-config*, în linia de comandă. Se va deschide un meniu din care se vor selecta opțiunile de interfață și se activează interfața VNC, cum se poate observa în Fig. 1.9.

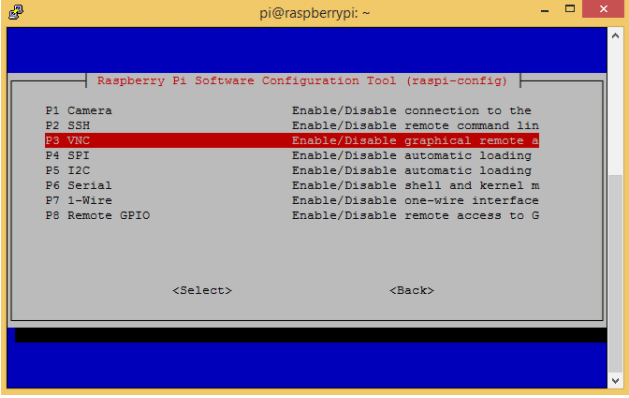


Fig. 1. Interfețe Raspberry Pi.

Pentru a completa această etapă este necesar un *reboot*, prin folosirea comenzii: *sudo reboot*, în linia de comandă. În a doua etapă trebuie instalată aplicația RealVNC Viewer pe computerul de pe care se dorește accesarea *desktop-ului* de la Pi. În următoarea etapă trebuie aflată adresa IP a Raspberry Pi. Adresa se află prin executarea comenzii: ifconfig, în linia de comandă. După aflarea adresei IP, aceasta se introduce în bara de adrese a programului RealVNC Viewer. În final se va deschide o fereastră unde trebuie introduse parola și numele de utilizator de la Raspberry Pi, cum este arătat în Fig. 1.10.

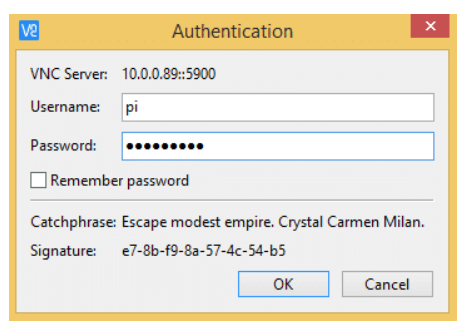


Fig. 1. Autentificare Raspberry Pi.

*Desktop-ul* de la Raspbian se va porni într-o fereastră de VNC Viewer, conform [19].

## 1.5 Arduino și Microcontrolerul ATmega328P

Arduino este o companie *open-source* care produce plăci de dezvoltare bazate pe microcontrolere și partea de *software* necesară pentru programarea 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. În acest proiect s-a folosit modelul Arduino Nano, cum se poate vedea și în Fig. 1.11. Acestă placă are dimensiuni și costuri reduse și este bazată pe un microcontroler ATmega328P. Cu privire la specificațiile tehnice, cele mai importante sunt: arhitectura procesorului AVR, tensiunea de operare de 5V, 8 pini analogici de I/O, 22 de pini digitali de I/O, memorie de tip Flash, SRAM, EEPROM. Scrierea și încărcarea programelor pe plăcile Arduino se realizează folosind Arduino Software (IDE). Acest *soft* este scris cu funcții din C și C++. IDE-ul Arduino acceptă limbajele de programare C și C++.[20]

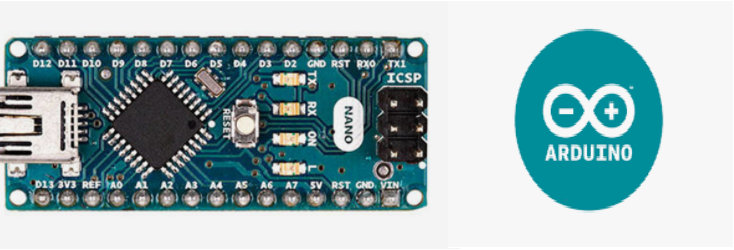
****

Fig. 1. Placă de dezvoltare Arduino Nano. [20]

## 1.6 Modulele NRF24l01+

### 1.6.1 Descriere generală

Modulele NRF24l01+, așa cum se pot observa în Fig. 1.12, sunt folosite pentru a realiza comunicația *wireless* între Raspberry Pi al robotului și Arduino Nano al telecomenzii. Aceste module sunt niște transmițătoare RF care funcționează în banda de frecvență ISM de 2.4 GHz și folosesc pentru transmisia de date o modulație de tip GFSK. Rata de transfer poate fi selectată între 250 kbps, 1Mbps sau 2Mbps. Tensiunea de funcționare al acestor module este cuprinsă între 1.9 – 3.6 V, dar pinii au o logică de 5V, deci pot fi conectați la orice microcontroler cu logică de 5V, fără a fi nevoie de un convertor de nivel logic. Există două versiuni pentru modulele NRF și anume cele cu antenă încorporată pe placă, NRF24l01+ și cele care conțin un amplificator de putere, NRF24l01+ PA/LNA. Diferența între cele două versiuni o face distanța pe care se poate realiza transmisiunea și anume 100 m pentru prima opțiune, respectiv 1000m pentru a doua opțiune. În acest proiect s-a folosit versiunea fără amplificator de putere. Se ia în considerare faptul că distanțele reale până la care se poate realiza transmisiunea vor fi mai mici decât cele menționate mai sus din cauza obstacolelor, cum ar fi zidurile.



Fig. 1. Transmițător radio NRF24L01+. [21]

Se poate observa că acest modul are 8 pini. Primul pin este cel de masă GND. Al doilea pin este pinul de alimentare VCC de 3.3V în current continuu. Al treilea pin este CE,*chip enable*, care este folosit pentru a activa interfața de comunicație serială SPI. Al patrulea pin este CSN, *chip select not*, care trebuie menținut mereu în starea de *HIGH* pentru că altfel se va dezactiva interfața de comunicație SPI. Pinul cinci este SCK, *serial clock*, care furnizează semnalul de tact necesar pentru funcționarea interfeței SPI. Pinii șase și șapte sunt MOSI, *master out – slave in*, respectiv MISO, *master in - slave out*. MOSI este folosit pentru a putea primi date de la microcontrolerul ATmega328P, MCU–ul pe care este bazată placa de dezvoltare Arduino Nano. MISO este folosit pentru a putea trimite date către microcontrolerul ATmega328P. Ultimul pin este IRQ, pinul folosit în caz că este necesară generarea unei întreruperi. În configurația acestui proiect, acest pin nu este folosit pentru nici unul din cele două transmițătoare radio NRF24L01+.

Din cauză că nici ATmega328P și nici Raspberry Pi nu pot furniza întotdeauna suficientă energie transmițătoarelor NRF24L01+, pentru a remedia acest lucru au fost folosite două condensatoare de 100uF. Condensatoarele vor fi legate între pinul de alimentare VCC și pinul de împământare GND. Rolul acestor condensatoare este să înmagazineze energie electrică și să o elibereze atunci când este necesar.

### 1.6.2 Modul de funcționare

Din punct de vedere al modului de funcționare, modulele NRF24l01+ emit și recepționează date pe o anumită frecvență numită canal. Pentru ca două module să comunice între ele trebuie să se afle pe același canal. Deoarece lățimea de bandă ISM de 2.4 GHz este de 125 MHz și fiecare canal ocupă 1MHz, vom avea 125 de canale posibile, care oferă posibilitatea de a avea o rețea de 125 de module, care funcționează independent în același loc. Acest lucru nu este posibil dacă se alege o rată de transfer de 2Mbps, din cauza suprapunerii canalelor. Pentru a evita acest lucru, pentru o rată de 2Mbps se va lăsa o spațiere de 2MHz între două canale vecine. Datele se trimit sub formă de pachete. Modulele NRF24l01+ folosesc ca structură pentru pachetele de date protocolul ESB, cum se poate observa în Fig. 1.13.

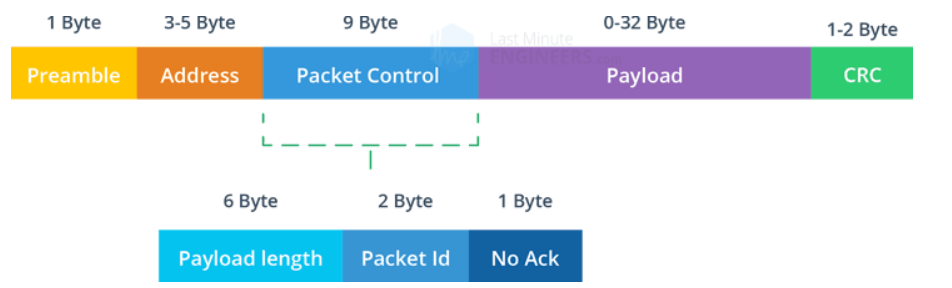


Fig. 1. Protocolul ESB. [21]

Structura pachetului este prezentată în Fig. 1.13. Putem observa faptul că *payload-ul* este variabil, dimensiunea lui poate lua valori între 1 și 32 byte. Dimesiunea maximă a unui pachet care poate fi trimis cu NRF24l01+ este de 32 byte. Pentru a evita pierderea de pachete în momentul trasnmisiunii se folosește pachetul ACK. În cazul transmiterii corecte al unui pachet, după 130us receptorul va trimite pachetul de ACK, iar după recepționarea acestuia de către emițător se va trimite următorul pachet. Pot exista și pachete pierdute în momentul transmiterii sau pachete de ACK pierdute. Aceste cazuri sunt nedorite și pot afecta transmisiunea de date de la un modul la altul, conform [21]. Tipul de comunicație folosit în acest proiect este *half-duplex*. Cu aceste module NRF24l01+ se mai pot realiza comunicații de tip simplex sau full-duplex.

## 1.7 Motoare

### 1.7.1 Motoare stepper DC 3V

Motorul *stepper*, din Fig. 1.14, este un motor electric a cărui principală caracteristcă este faptul că se rotește prin efectuarea de pași, adică prin deplasarea cu o cantitate fixă de grade. Se poate determina poziția doar prin numărarea pașilor efectuați. Aceste motoare sunt alcătuite dintr-o parte statică și o parte care se mișcă, rotorul. Partea statică are niște cătări pe care sunt înfășurate fire și se comportă ca niște bobine, în timp ce rotorul este un magnet permanent sau un miez de fier cu reluctanță variabilă. Principiul de funcționare al motorului *stepper* se bazează pe alimentarea uneia sau a mai multor faze a părții statice. Prin alimentare se generează un câmp magnetic de către curentul care trece prin bobină, iar rotorul se aliniază cu acest câmp. Prin furnizarea a mai multor faze în succesiune, rotorul poate fi rotit până atinge poziția dorită, după cum se precizează în [22]. Aceste motoare au două terminale nepolarizate. Prin schimbarea pinului de alimentăre cu cel de masă, motorul se va roti în sensul acelor de cearsornic sau în sens trigonometric.

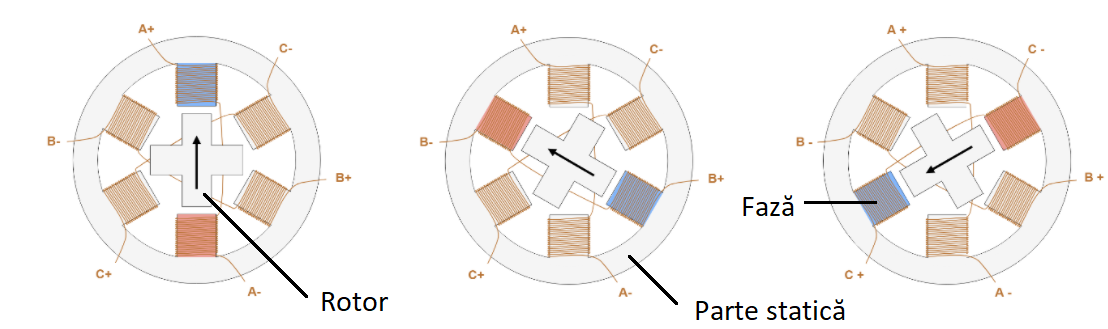


Fig. 1. Principiu funcționare motor DC. [22]

### 1.7.2 Micro servo-motoare SG90

Servo–motoarele, din Fig. 1.15, permit un control precis asupra poziției unghiulare, accelerației și vitezei. Prezintă niște caracteristici pe care un motor obișnuit nu le are. Îmbină un motor obișnuit cu un senzor pentru controlul poziției. Un servo–motor este alcătuit din un motor DC sau AC, un potențiometru, un echipament de angrenaj și un circuit de control. Echipamentul de angrenaj este folosit pentru a reduce RPM și pentru a crește cuplul servo– motorului.În acest proiect s-au folosit servo-motoare SG90, deoarece au dimensiuni reduse, sunt ușoare și au o putere de ieșire ridicată. Se pot roti aproximativ 180 de grade, 90 de grade în fiecare direcție. Aceste motoare au trei pini: pinul de alimentare, pinul de masă și un pin digital pentru controlul mișcării, prin specificarea unghiului la care trebuie să se deplaseze motorul, conform [23]. Specificațiile servo-motorului SG90 se pot vizualiza în Tabelul II.

Tabel Specificații SG90[23]

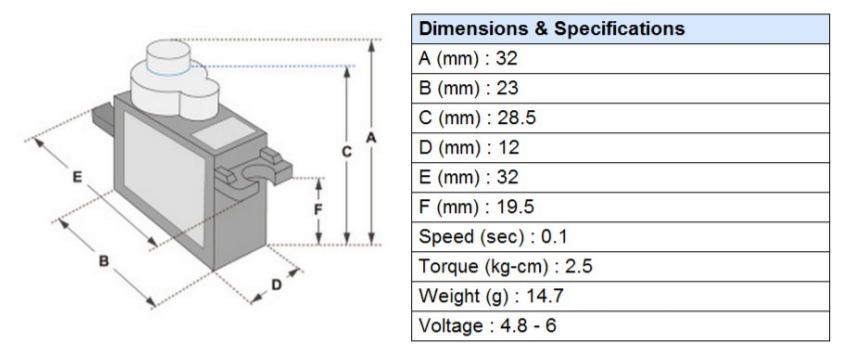


Fig. 1. Servo–motor SG90.[23]

## 1.8 Puntea H duală L298N

Puntea H L298N este folosită pentru a controla direcția și viteza de rotație a motoarelor stepper DC de 3V. Suportă o tensiune de alimentare de maxim 46 V și un current de până la 4 A. Conține 4 pini de *input*, 4 pini de *output*, 2 pini de *enable*, 1 pin de alimentare, 1 pin de masă și 1 pin de 5V. Controlarea vitezei de rotație a motoarelor DC se realizează prin controlarea tensiunii de intrare în motoare, iar cea mai întâlnită metodă pentru a face acest lucru este folosind semnale de tip PWM, cum se poate observa în Fig. 1.16.

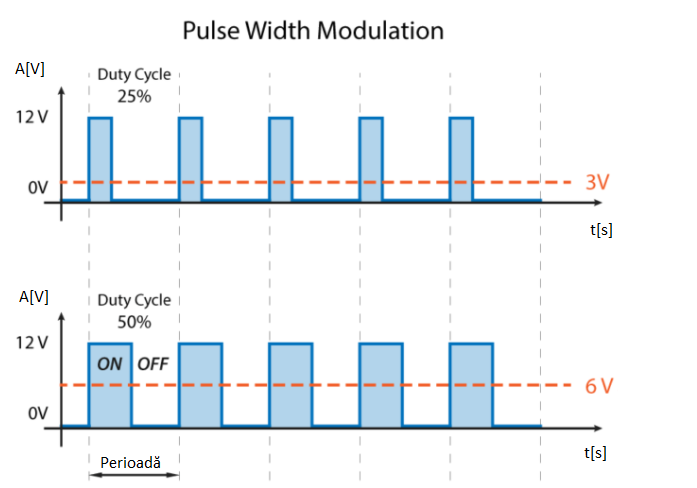


Fig. 1. Semnale de tip PWM.[24]

Se poate observa în figura de mai sus dependența tensiunii de factorul de umplere, adică de timpul cât semnalul este *ON* în comparație cu timpul în care semnalul este *OFF*, într-o singură perioadă. Cu cât tensiunea de intrare este mai mare, cu atât factorul de umplere este mai mare și în consecință viteza de rotație a motoarelor va fi mai ridicată, după cum se precizează în [24]. În Fig. 1.17 este prezentată structura punții H folosită în acest proiect.

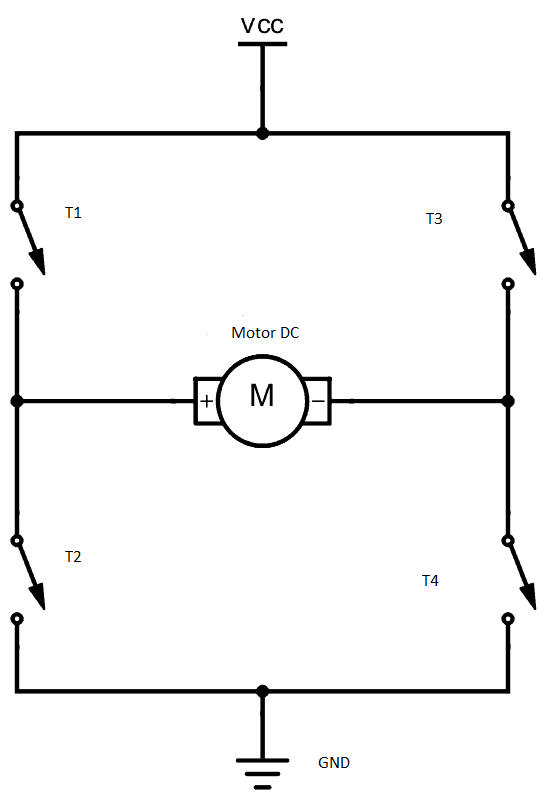


Fig. 1. Punte H.

Pentru controlul direcției de rotație a motoarelor trebuie inversată direcția fluxului de curent prin motor. Pentru acest lucru se folosesc cele 4 elemente de *switch* ale punții H și anume cele 4 tranzistoare, având în mijloc motorul. Din această cauză se spune că formează o configurație în H. Prin activarea simultan a două elemente de *switch* se poate schimba fluxul curentului prin motor și în consecință se va schimba și direcția de rotație a motorului. Așadar modulul L298N combină tehnica pentru semnalele PWM cu o punte H și poate controla simultan viteza și direcția de deplasare pentru cele două motoare *stepper* DC, conform [24].

## 1.9 Modul cameră compatibil cu Raspberry Pi

Modulul de cameră folosit în acest proiect este compatibil cu orice versiunea de Raspberry Pi. Dintre specificațiile importante ale acestui modul fac parte: rezoluția de 5MP, modul fotosensibil OV5647, apertura de 1.8, distanța focală ajustabilă de 3,6 mm, sprijină accesul la lumina infraroșu sau la lumina de umplere, conform [25]. Camera se conectează prin interfața specială a Raspberry Pi. Din meniul de configurare al Raspberry Pi se selectează opțiunea de interfețe și din noua fereastră deschisă se activează interfața de cameră, după care se realizează operația de *reboot* a plăcii Raspberry Pi. Se testează funcționarea camerei prin deschiderea liniei de comandă a Raspberry Pi și prin rularea comenzii: raspistill –o Desktop/image.jpg, cum se poate observa în Fig. 1.18. Această comandă are ca scop deschiderea camerei pentru câteva secunde și realizarea unei fotografii care va fi salvată pe *desktop-ul* Raspberry Pi, conform [26]. Imaginile vor fi transmise cu ajutorul formatului de compresie video MJPEG, dimensiunea imaginilor va fi setată din codul sursă la valoarea de 640x480 pixeli, iar rata de redare va fi setată la 24 de cadre/secundă. Pentru a calcula câtă memorie ocupă *stream-ul* video se folosesc următoarele ecuații:

Numărpixeli = Numărpixeli orizontală x Numărpixeli verticală [pixeli] (1)

Dimensiunecadru = Numărpixeli x Numărcanale x Numărbiți/pixel [biți] (2)

Dimensiunetotală stream video = Dimensiunecadru xNumărcadre/secundăxDuratăstream [biți] (3)

Din cauză că pentru *stream-ul* video s-a ales cea mai mică rezoluție suportată, adică 640x480, utilizând formulele de mai sus s-au obținut:

Numărpixeli = 640 x 480 = 307200 pixeli, va avea fiecare cadru al *stream-ului* video

Dimensiunecadru = 307200 x 3 x 8 = 7372800 biți, va fi dimensiunea în biți a fiecărui cadru al *stream-ului* video. Deoerece imaginile înregistrate sunt de tipul RGB, vom avea 3 canale și fiecare pixel va fi reprezentat pe 8 biți.

Dimensiunetotală stream video = 7372800 x 24 x 60 = 10616832000 biți = 1327104000 bytes = 1327.104 Megabytes, ar ocupa *stream-ul* video necomprimat cu durata de 1 minut, dacă ar fi stocat în memorie.

Comitetul JPEG a definit, documentat, simulat, testat și validat soluțiile și variantele pentru a asigura compresia imaginilor fixe. MJPEG, Motion JPEG, este o adaptare video a standardului de compresie JPEG, pentru imagini fixe. Există peste 44 de versiuni acceptate de JPEG. Avantajele folosirii standardului JPEG sunt simplitatea algoritmului de codare și asigurarea unui raport de compresie decent. MJPEG este practic o extensie a standardului JPEG de la un cadru, la o serie de cadre. Codarea folosind standardul MJPEG este intracadru și de aceea raportul de compresie este de maxim 1:20, comparativ cu alte standarde care folosesc codarea intercadru, cum ar fi MPEG1, care au rapoarte de compresie de 1:50 sau mai mari, după cum se precizează în [27]. Prin urmare dimensiunea totală a *stream-ului* video după compresia cu standardul MJPEG va fi de 66.35 Megabytes, folosind un raport de compresie de 1:20.

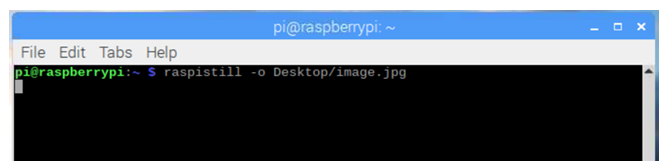


Fig. 1. Verificare funcționare modul de cameră.

## 1.10 Coborâtor de tensiune LM2596

LM2596, vizibil în Fig. 1.19, este un circuit integrat ideal pentru a coborî nivelul tensiunii.Tensiunea poate fi ajustată între valorile 1,23 – 37 V. Poate suporta un curent de până la 3 A cu o excelentă reglare a sarcinii. Este alcătuit din doi pini de intrare, prin care se primește tensiunea ce trebuie reglată și doi pini de ieșire care furnizează valoarea dorită pentru tensiune, după ce a fost reglată, conform [28]. În acest proiect au fost folosite două astfel de coborâtoare, unul pentru a limita tensiunea de la acumulatorii Li-ion la 5V, pentru a alimenta placa Raspberry Pi și celălalt pentru a limita tensiunea de alimentare la 6V, pentru cele 3 servo-motoare care alcătuiesc brațul mecanic. Tensiunea se reglează ușor folosind o șurubelniță și învârtind de potențiometrul dispozitivului LM2596, până se atinge valoarea tensiunii dorită. Vizualizarea valorii tensiunii se poate face folosind un multimetru digital.

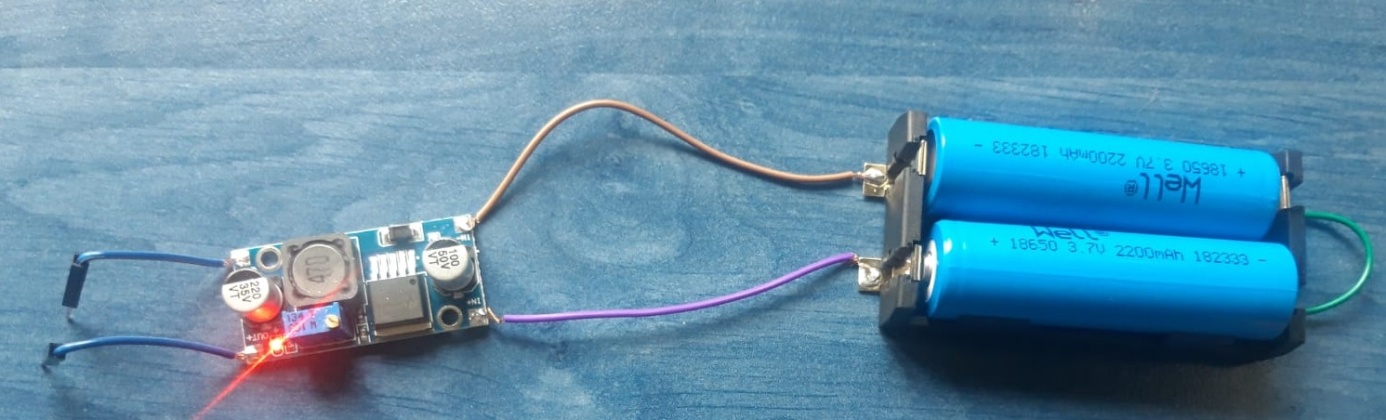


Fig. 1. Coborâtor de tensiune LM2596.

## 1.11 Joystick biaxial

Modulele joystick biaxial sunt folosite pentru a trimite coordonatele necesare deplasării motoarelor *stepper* și a servo-motoarelor. Primul modul joystick este folosit pentru controlarea mișcării robotului, iar celălalt este folosit pentru controlarea mișcării brațului mecanic. Un joystick biaxial are cinci pini: un pin de alimentare de 5V, un pin de masă GND, un pin VRx care controlează potențiometrul de pe axa Ox, un pin VRy care controlează potențiometrul de pe axa Oy și un pin SW care controlează un *push-button*. Valorile atât de pe axa Ox, cât și de pe axa Oy, dintr-un capăt în celălalt, pot varia între 0 – 1023, după cum se poate observa în Fig. 1.20 și după cum s-a precizat în [29]. Aceste valori au fost scalate la 0 – 99 pentru motoarele *stepper*, 0 însemnând că motoarele nu se mișcă și 99 însemnând viteză maximă. În cazul servo–motoarelor, aceste valori au fost scalate între 2 – 12, 2 însemnând că motorul se află la 0 grade și 12 însemnând că motorul se află la 180 de grade.

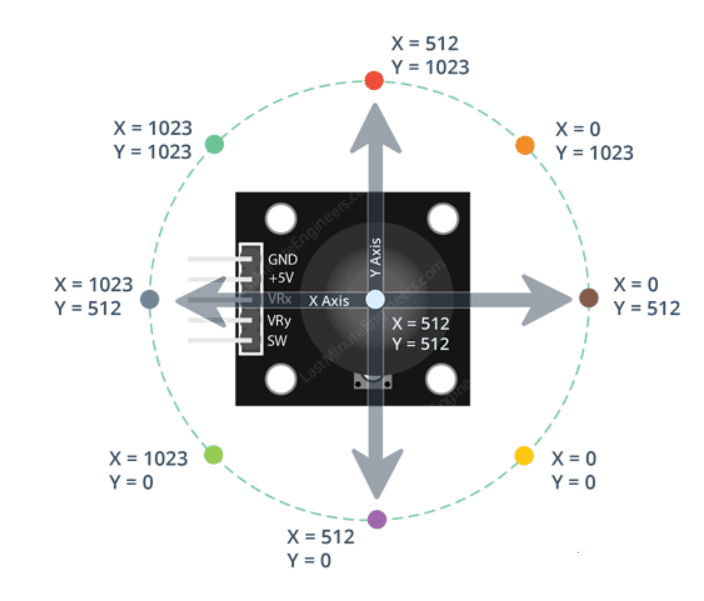
****

Fig. 1. Modul Joystick Biaxial. [29]

## 1.12 Senzorul DHT 11

Senzorul DHT 11 este un senzor digital pentru detectarea temperaturii și umidității. Este potrivit pentru acest proiect deoarece are un cost redus. Folosește un NTC pentru a măsura aerul din apropiere și furnizează un semnal digital pe pinul de date. Înafară de pinul de date mai este prevăzut cu un pin de alimentare de 5V și un pin de masă GND. Se pot citi noi date de pe senzor la fiecare două secunde. Raspberry Pi va avea un comportament de tip *Master*, iar senzorul DHT 11 va avea un comportament de tip *Slave*. În Fig. 1.21 este prezentat modul de funcționare al senzorului.

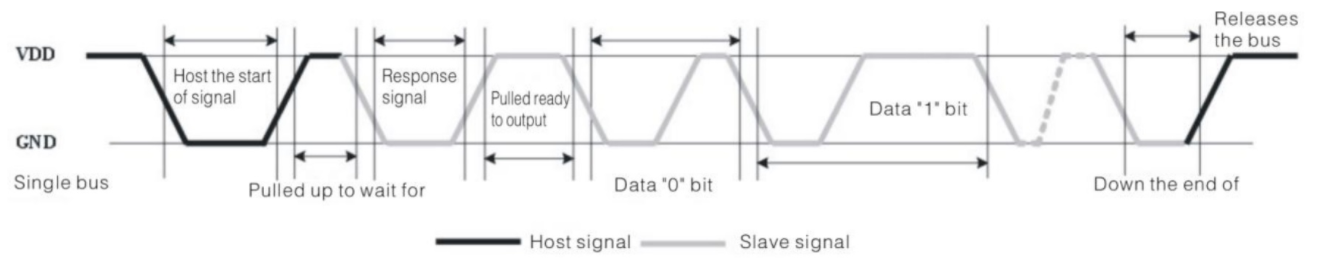
****

Fig. 1. Mod de funcționare senzor DHT11. [30]

Raspberry Pi este setat ca *output* și va pune linia de date în starea de *low* pentru un timp minim de 18 ms după care va elibera linia. După acest timp Raspberry Pi este făcut *input*. Pinul de date al DHT 11, care este *input*, citește starea de *low* făcută de Raspberry Pi și acționează ca un pin *output* și trimite un răspuns de semnal *low* pe linia de date pentru aproximativ 80us, iar apoi ridică linia pentru încă 80us. După această etapă, senzorul DHT 11 trimite 40 de biți de date cu logica ‘0’, fiind o combinație de 50us stare *low* și 28us stare *high* și logica ‘1’, fiind o combinație de 50us stare *low* și 80us stare *high*. După transmiterea celor 40 de biți pinul de date al DHT 11 rămâne *low* pentru încă 50us și apoi își schimbă starea la *input* pentru a accepta o nouă cerere de la Raspberry Pi, conform [30].

## 1.13 LCD alfanumeric 16x02 caractere

LCD-ul folosit în acest proiect are două linii cu câte șaisprezece caractere fiecare. Acest dispozitiv folosește interfața de comunicare I2C. Are patru pini și anume: un pin de alimentare 5V, un pin de masă GND, un pin SDA pentru linia de date și un pin SCL pentru linia de ceas serial. Contrastul LCD-ului este ajustabil prin reglarea potențiometrului pe interfața I2C, cum este precizat în [31].

## 1.14 Acumulatori Li-ion 18650

Alimentarea tuturor circuitelor din acest proiect se realizează cu ajutorul acestui tip de acumulator.Printre specificațiile importante ale acestor acumulatori se numără capacitatea nominală minimă de 2200mAh și tensiunea nominală de 3.7 V. În acest proiect a fost nevoie de 8 astfel de acumulatori, înseriați câte doi. Din cele patru grupuri de acumulatori înseriați câte doi, primul grup a fost folosit pentru alimentarea telecomenzii robotului. Al doilea grup a fost folosit pentru alimentarea celor două motoare DC de 3V și a punții duale H L298N. Al treilea grup a fost folosit pentru alimentarea celor trei servo–motoare ale brațului mecanic. Al patrulea grup s-a folosit pentru alimentarea plăcii de dezvoltare Raspberry Pi. Pentru a pune doi acumulatori în serie s-au folosit suporți pentru acumulatori. La fiecare suport au fost lipite trei fire: un fir care leagă terminalul de plus al primului acumulator cu terminalul de minus al celui de-al doilea acumulator, un fir care v-a fi folosit ca terminal de alimentare și un fir care v-a fi folosit ca terminal de masă. În Fig. 1.22 se pot vizualiza doi acumulatori Li-ion 18650 înseriați, cu ajutorul unui support și a firelor de legătură.

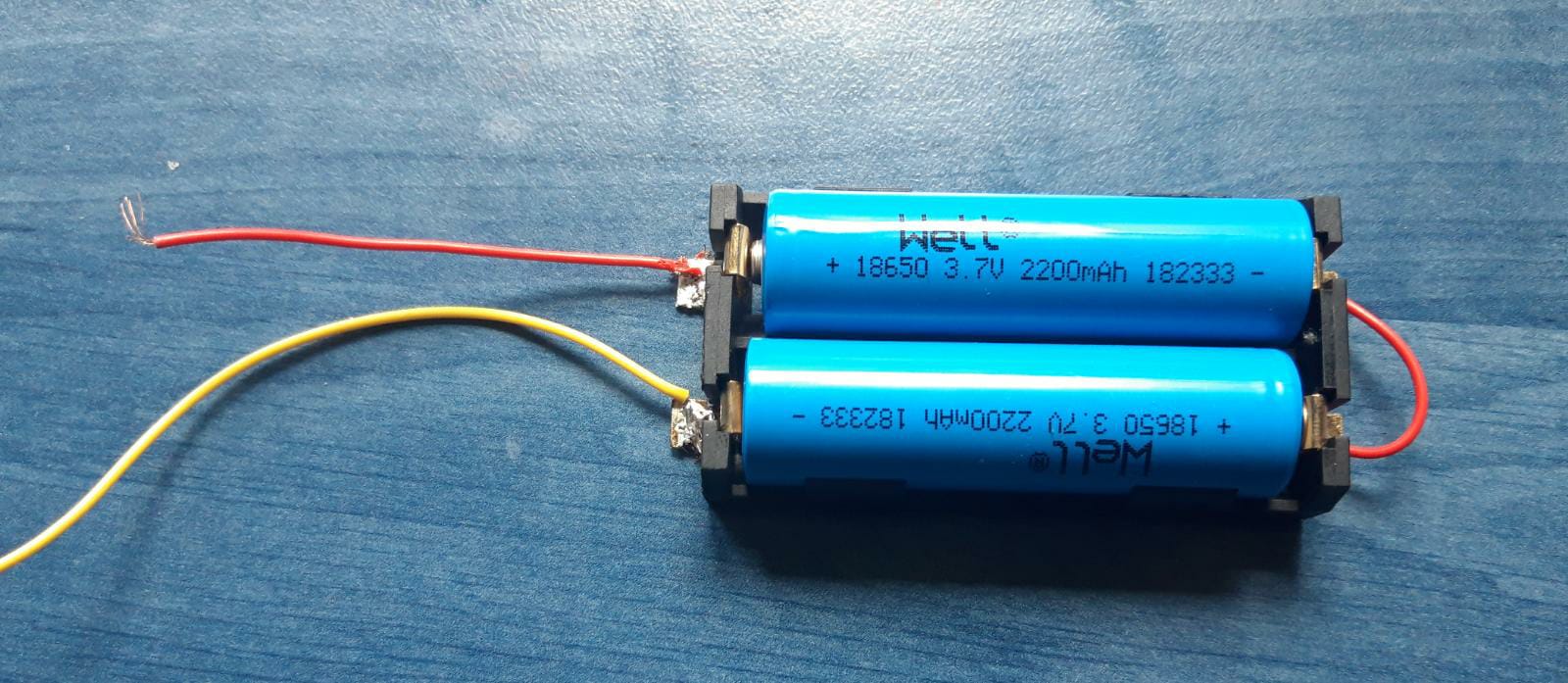


Fig. 1. Acumulatori Li-ion 18650 3.7V.

# 

# 

# CAPITOLUL 2 DESCRIERE SOFTWARE

## 

## 2.1 Generalități limbaje de programare

Un limbaj de programare este o grupare de expresii și reguli valide de formulare a instrucțiunilor pentru un computer. Un limbaj de programare are definită o sintaxă care trebuie urmată de programatori pentru a scrie programele necesare. Practic limbajul de programare oferă programatorului posibilitatea de a specifica în mod clar acțiunile pe care trebuie să le execute un calculator, în ce ordine și cu ce date. După paradigma programării limbajele pot fi:ezoterice, procedurale, funcționale, pentru programarea logică sau mixte. După nivelul de abstractizare limbajele pot fi: cod-mașină, de asamblare, de nivel înalt, neprocedurale sau limbaje utilizate în domenii precum inteligența artificial, conform [32]. În acest proiect limbajele de programare folosite sunt C++ și Python 3.

## 2.2 C++

### 2.2.1 Generalități

C++, sigla vizibilă în Fig. 2.1, este un limbaj de programare de uz general creat de informaticianul Bjarne Stroustrup în anul 1985 și este o extensie a limbajului de programare C. Se mai numește și C cu clase. Ca stil folosit, acest limbaj de programare este un limbaj procedural de nivel înalt. C++ se concentrează pe programarea orientată pe obiect și pe facilități de manipulare a memoriei de nivel scăzut. Este de obicei implementat ca un limbaj ce trebuie compilat și există multe companii care furnizează compilatoare C++, cum ar fi Microsoft, Oracle sau Intel, deci este disponibil pe o gamă largă de platforme. C++ a fost conceput pentru programarea sistemelor embedded, fiind constrâns de software și sisteme largi să fie performant, eficient, să aibă flexibilitate, așa cum subliniază design-ul său, conform [33].



Fig. 2. C++. [33]

### 2.2.2 Biblioteci folosite

Includurea tuturor structurilor, precum și a funcțiilor ce permit manipularea și prelucrarea lor sunt realizate cu ajutorul bibliotecilor. O bibliotecă C++ este compusă din unul sau mai multe fișiere *header* și o bibliotecă obiect. Fișierele *header* furnizează clasa și alte definiții necesare accesării funcțiilor bibliotecii. Biblioteca de obiecte oferă funcții și date compilate necesare pentru a produce un program executabil.

Deoarece telecomanda realizată pentru controlul robotului este un circuit ce are la bază Arduino Nano, programarea acesteia s-a realizat în Arduino Software (IDE) folosind biblioteci scrise în limbajul C++. Bibliotecile folosite în realizarea programului pentru telecomandă sunt: RF24, SPI, DigitalIO, LiquidCrystal\_I2C și Wire.RF24 este biblioteca folosită pentru a inițializa un obiect de tip radio, pentru modulul NRF24l01+ de pe telecomandă. Cu ajutorul acestei biblioteci se poate programa transmițătorul, pentru a transmite informații către robot. SPI este folosită de microcontrolere pentru a comunica cu unul sau mai multe dispozitive periferice în mod rapid și pe distanțe scurte. De obicei microcontrolerul, în cazul de față Arduino Nano, este dispozitivul *Master*, iar celelalte dispozitive periferice se comportă ca *Slave*. DigitalIO furnizează metode ușoare de accesare a pinilor I/O ai microcontrolerului, cum ar fi digitalRead() sau digitalWrite(). Wire este folosită pentru a comunica cu dispozitive care au interfața de comunicație I2C, în cazul de față cu afișajul LCD.LiquidCrystal\_I2C este biblioteca folosită pentru a defini un obiect de tip afișaj LCD cu două linii a câte șaisprezece caractere, care folosește interfața de comunicație I2C.

Pentru a instala o bibliotecă externă, care nu este deja predefinită în Arduino Software (IDE), se va deschide IDE-ul și se va selecta din bara de opțiuni aflată în partea de sus Sketch, apoi Include Library, urmat de Manage Libraries, după cum se poate vedea în Fig. 2.2.

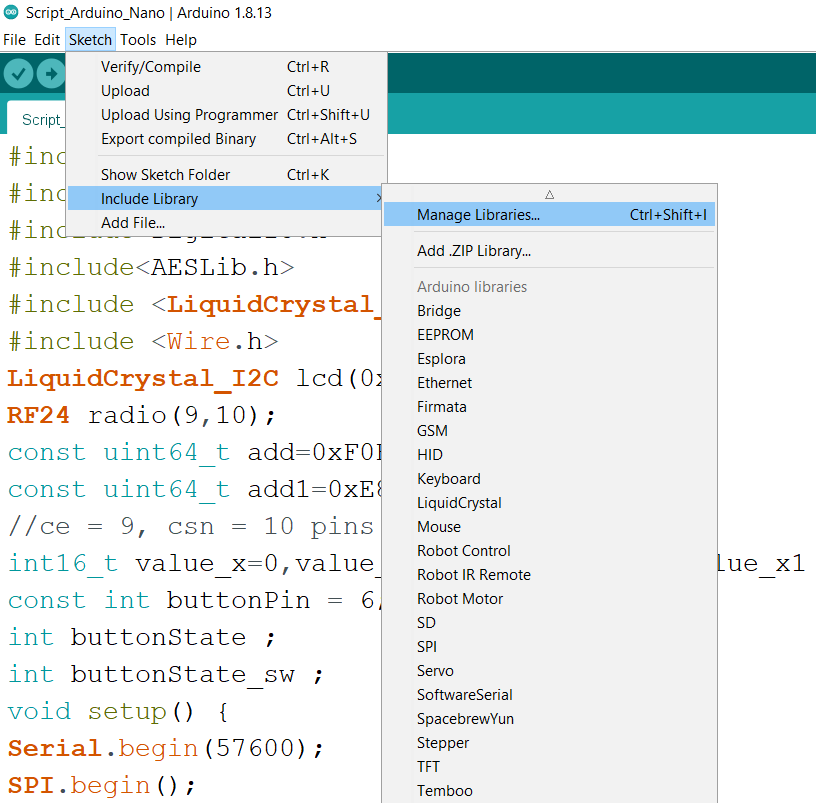


Fig. 2. Deschidere manager de biblioteci în Arduino Software(IDE).

După selectarea Manage Libraries se va deschide o fereastră unde se poate căuta biblioteca dorită. După selectarea bibliotecii și a versiunii se va apăsa butonul de Install. După terminarea instalării se verifică dacă biblioteca a fost adăugată, prin selectarea din meniu a Sketch -> Include Library.Acest lucru se poate vedea în Fig. 2.3.

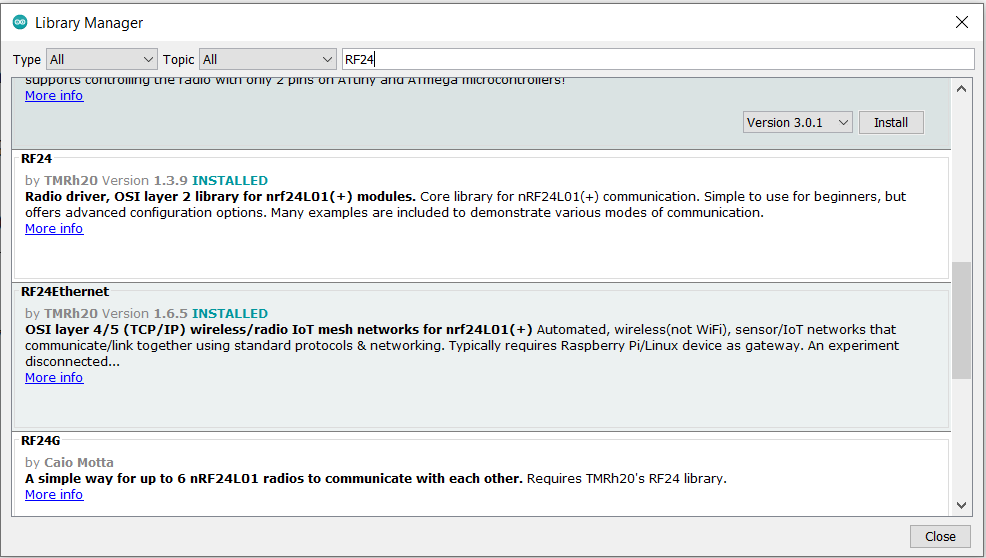


Fig. 2. Instalarea unei biblioteci în Arduino Software(IDE).

De asemenea se poate adăuga și manual o bibliotecă. Pentru asta trebuie descărcată biblioteca sub formă de fișier arhivă .ZIP. După descărcare se dezarhivează fișierul .ZIP și se copiază *folder-ul* dezarhivat în folder-ul ,,libraries’’ din Sketchbook. În ultima etapă se pornește Arduino Software (IDE) și se verifică dacă biblioteca adăugată se află în lista de biblioteci din program, tot prin selectarea Sketch –>Include Library din bara de comandă din program. Acest lucru se poate vedea în Fig. 2.4.

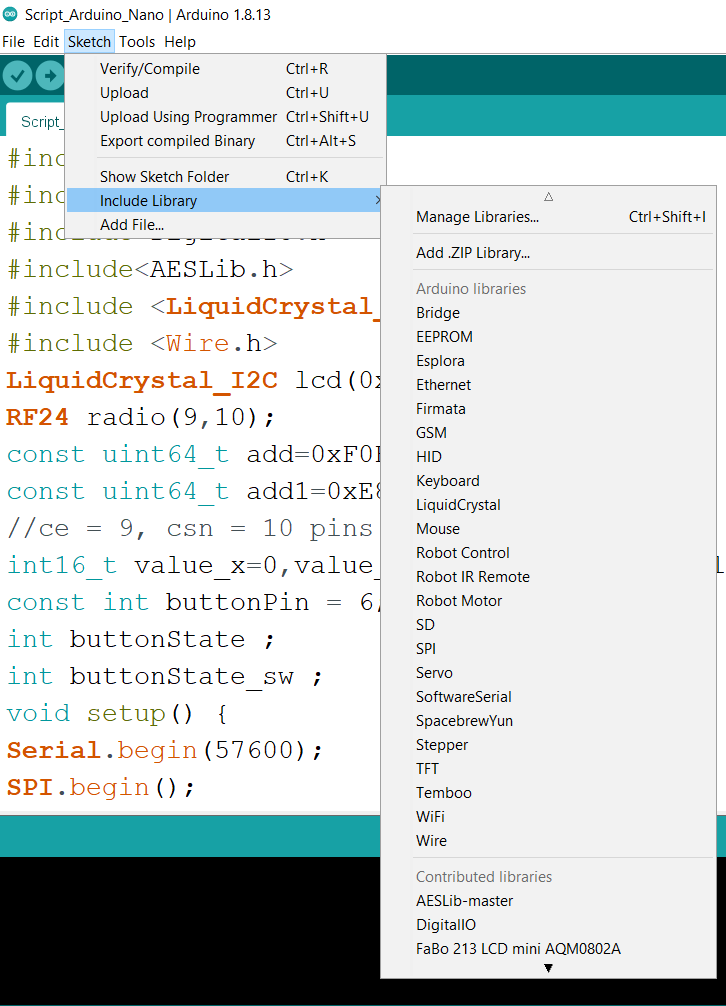


Fig. 2. Adăugarea unei biblioteci în Arduino Software(IDE) manual.

## 2.3 Python

### 2.3.1 Generalități

Python, sigla vizibilă în Fig. 2.5, este un limbaj de programare procedural, de nivel înalt, pentru uz general creat de Guido van Rossum în 1989. Este un limbaj de programare orientată pe obiect care urmărește lizibilitatea codului prin folosirea indentării. Sintaxa simplistă, precum și abordarea orientată pe obiecte își propun să ajute programatorii să scrie coduri clare și logice pentru proiecte la scară mică și mare. În ceea ce privește paradigma de programare, pe lângă utilizarea pentru *software* de tipul programare orientată pe obiect, permite și programarea imperativă, funcțională sau procedurală. Sistemul de tipizare este dinamic și administrarea memoriei se face în mod automat prin intermediul unui serviciu denumit *’’garbage collector’’*. Implementarea Python este făcută în C/C++, deci Python va avea un mod de execuție mai lent, timpul necesar pentru a rula un program va fi mai mare și va folosi mai multă memorie decât C/C++. Un alt avantaj al limbajului Python este, pe lângă simplitatea sintaxei, existența unei biblioteci standard de metode amplă, conform [34].



Fig. 2. Python 3. [34]

### 2.3.2 Biblioteci folosite

Bibliotecile folosite sunt necesare pentru a putea programa toate dispozitivele care alcătuiec robotul. Deoarece robotul este un circuit bazat pe un SBC și anume Raspberry Pi, programarea lui s-a realizat în Thonny IDE, folosind biblioteci concepute pentru limbajul de programare Python 3. Bibliotecile folosite în realizarea programului pentru robot sunt: lib\_nrf24, RPi.GPIO, pigpio, spidev și Adafruit\_DHT. Bibliotecile necesare pentru a putea programa modulul de cameră sunt:IO, picamera, logging, socketserver, threading și http.

Biblioteca lib\_nrf24 este folosită pentru a crea un obiect de tip radio pentru modulul NRF24l01+ de pe robot. Cu această bibliotecă se poate programa transmițătorul, pentru a recepționa informații de la modulul NRF24l01+ de pe telecomandă și de asemenea pentru a trimite informații la modulul de pe telecomandă. Biblioteca spidev este folosită pentru comunicarea între dispozitive, Raspberry Pi se comportă ca un *Master*, iar celelalte dispozitive periferice se comportă ca *Slave*. RPi.GPIO este folosită pentru controlarea pinilor I/O ai Raspberry Pi. Biblioteca pigpio este folosită pentru controlul servo-motoarelor. La fel ca în cazul RPi.GPIO este folosită pentru a controla pinii I/O ai Raspberry Pi, cu diferența că impulsurile la pigpio sunt în micro secunde, față de mili secunde la RPi. GPIO.A fost necesară folosirea pigpio deoarece impulsurile sunt mult mai precise și așa se elimină mișcarea nedorită a servo-motoarelor. Acest lucru nu era posibil prin folosirea bibliotecii RPi.GPIO, din cauza mișcării haotice, dezordonată, pentru servo-motoarele ce alcătuiesc brațul mecanic. Biblioteca Adafruit\_DHT a fost folosită pentru a controla activitatea senzorului digital de temperatură și umiditate DHT11. Biblioteca picamera furnizează o interfață de Python pură pentru modulul de cameră Raspberry Pi. Biblioteca IO permite gestionarea operațiilor de intrare și ieșire legate de fișier. Prin ultilizarea bibliotecii IO funcțiile și clasele disponibile permit extinderea funcționalității, pentru a permite scrierea de date unicode. Logging este o metodă de urmărire a evenimentelor care se întâmplă atunci când rulează un *software*. Socketserver este folosită pentru a putea conecta două noduri ale unei rețele, astfel încât să comunice între ele. Threading este folosită pentru a putea rula mai multe sarcini, apeluri ale funcțiilor simultan. Threading se folosește în cazul în care executarea unei sarcini în Python implică o așteptare. Http este folosit deoarece colectează mai multe module pentru lucrul cu protocolul de transfer Hyper Text.

Instalarea bibliotecilor se realizează din linia de comandă a Raspberry Pi folosind PIP. Pentru a instala PIP se va rula în linia de comandă a Raspberry Pi instrucșiunea: *sudo apt-get install python3-pip*. Pentru a actualiza bibliotecile se va rula: *sudo apt-get update* sau *sudo apt-get upgrade*. Pentru instalarea unei biblioteci se va rula: *sudo apt-get install* sau *sudo pip3 install*, urmat de numele bibliotecii. Mai jos este prezentat un exemplu, cu setul de instrucțiuni care trebuie rulate în linia de comandă, pentru a instala biblioteca necesară, pentru a interfața senzorul digital DHT11 cu Raspberry Pi[35]:

*sudo apt-get install git-core*

*git clone* <https://github.com/adafruit/Adafruit_Python_DHT.git>

*cd Adafruit\_Python\_DHT*

*sudo apt-get install build-essential python-dev*

*sudo python setup.py install*

## 2.4 HTML

HTML este un limbaj de marcare folosit pentru a realiza pagini web ce pot fi afișate într-un *browser*, navigator. Marcarea se referă la faptul cum sunt folosite etichetele pentru a defini aspectul paginii și elementele din pagină. Internetul a trecut de-a lungul anilor prin multe schimbări, dar HTML a fost mereu limbajul fundamental folosit în crearea paginilor *web*. În timp ce *site-urile web* au devenit din ce în ce mai avansate, HTML a devenit mai simplu. HTML a fost conceput petru a putea fi citit și editat utilizând un editor de text simplu. Totuși scrierea și modificarea paginilor astfel solicită cunoștințe solide de HTML.[36]

Multe *site-uri web* dinamice generează pagini web folosind un limbaj de scriptare pe partea de server, cum ar fi PHP sau ASP. Cu toate acestea, chiar și paginile dinamice trebuie formatate folosind HTML. Prin urmare, limbajele de scriptare generează codul HTML care este trimis *browser-ului web*.

Paginile HTML sunt formate din etichete și au extensia .html sau .htm. De obicei etichetele sunt pereche, una de deschidere <eticheta> și una de închidere </eticheta>. Există cazuri în care etichetele nu se închid și se folosește <eticheta/>. Navigatorul *web* interpretează aceste etichete. Acest limbaj nu este *case-sensitive*, adică nu face diferența între litere majuscule și minuscule. Pagina principală pentru un *site web* se numește de obicei index.html sau index.htm. Această pagină este afișată automat la vizitarea *site-ului*. Un exemplu de structură al formatului HTML este vizibil în Fig. 2.6.

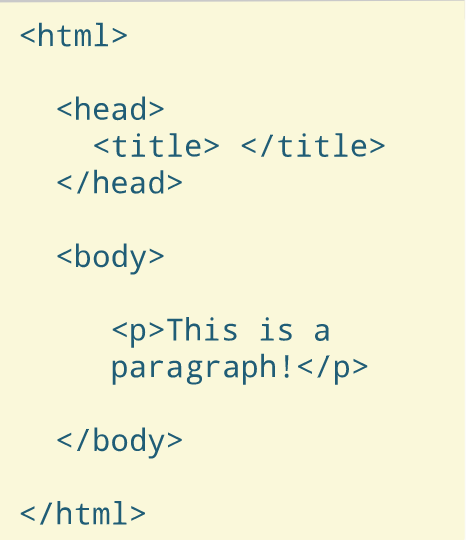


Fig. 2. Exemplu format HTML. [36]

# 

# CAPITOLUL 3 CONSTRUCȚIA ȘI MODUL DE FUNCȚIONARE AL ROBOTULUI

## 3.1 Construcția telecomenzii

Construcția telecomenzii a constat în lipirea componentelor pe PCB-ul proiectat conform schemei bloc și a *layout-ului*, Fig. 1-5, respectiv Fig. 1-6, din primul capitol. Majoritatea conexiunilor între componente au fost trasate pe *layer-ul* de *BOTTOM*. Traseele de pe *layer-ul* de *TOP*, în general treceri drepte, au fost realizate prin lipirea unor cabluri de cupru. Atât componentele, cât si cablurile au fost lipite utilizând o stație letcon, fludor și pastă decapantă flux. Cablajul complet asamblat se poate vizualiza în Fig. 3.1.

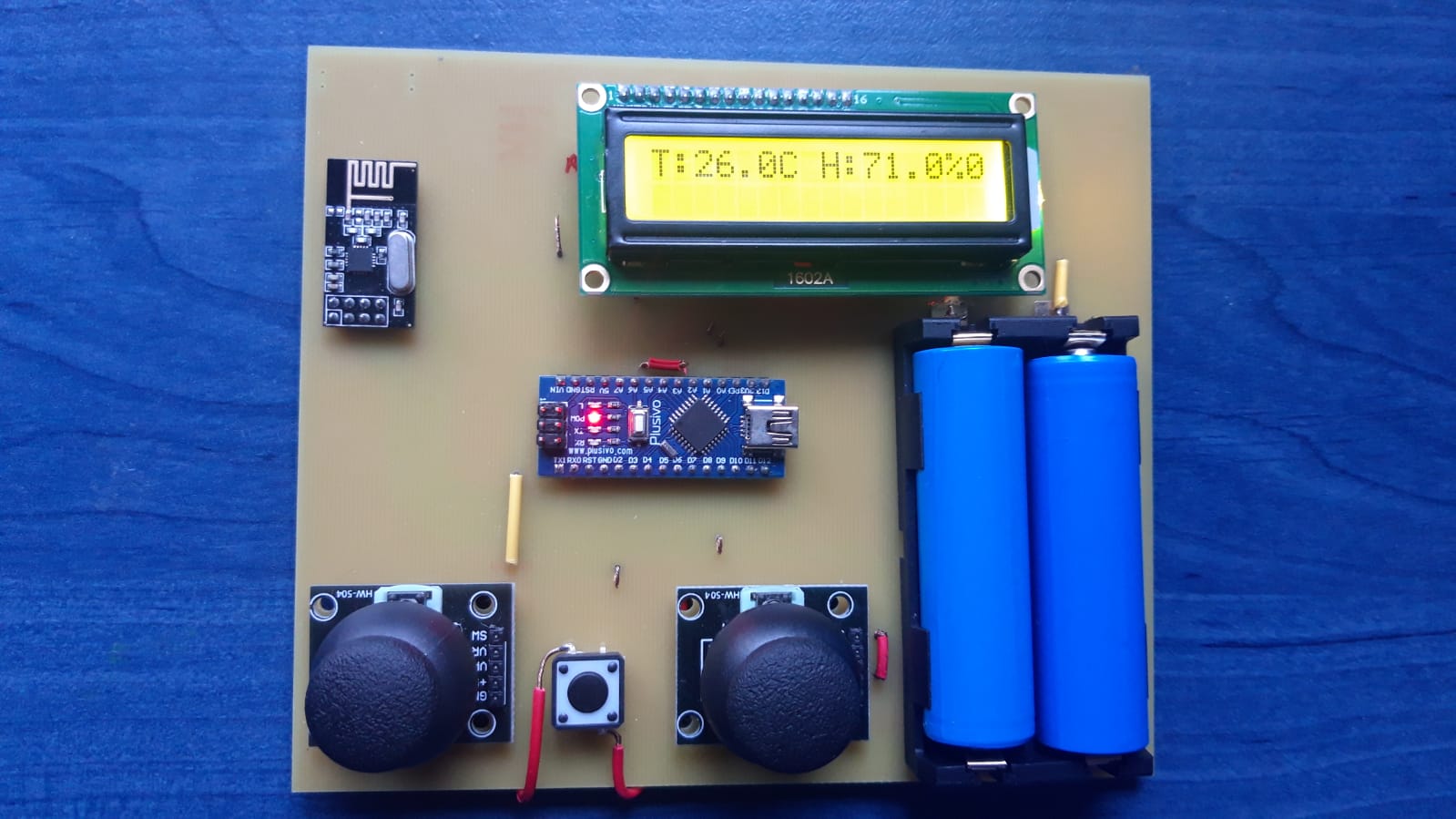


Fig. 3. Telecomanda robotului.

## 3.2 Construcția brațului mecanic

Pentru construcția brațului mecanic au fost necesare trei servo-motoare micro SG90 și părți metalice pentru a susține servo-motoarele. Brațul mecanic se poate deplasa pe două axe și anume pe axa orizontală, în direcțiile stânga-dreapta și pe axa verticală, în direcțiile sus–jos. Cu brațul mecanic se pot apuca obiecte cu greutate ușoară și se pot transporta dintr-o parte în alta.

Structura brațului mecanic și locul în care este plasat fiecare servo–motor se poate vizualiza în Fig. 3.2.

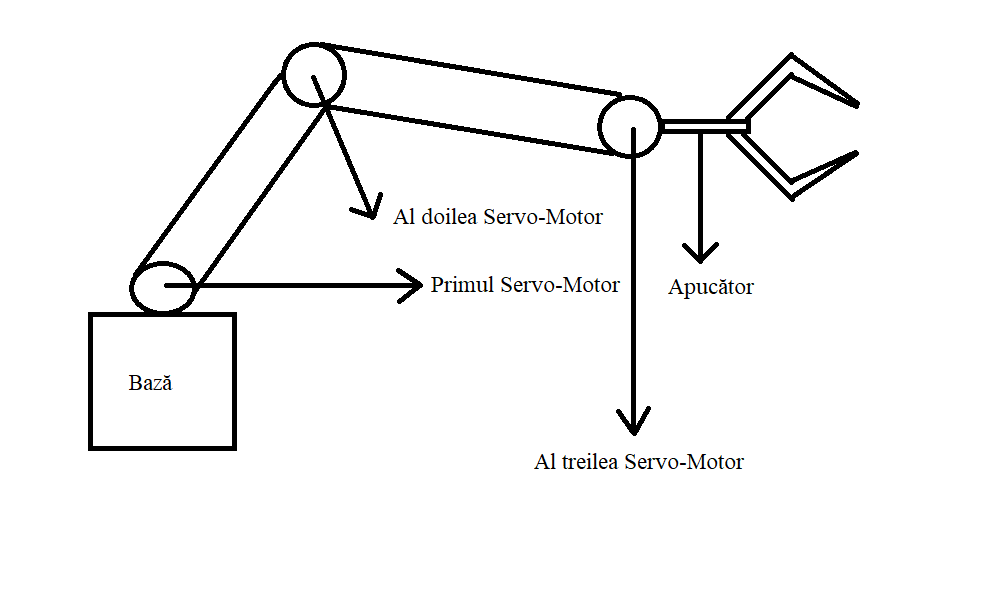


Fig. 3. Structură braț mecanic.

Brațul mecanic a fost structurat în trei părți. Prima parte asigură deplasarea pe axa orizontală. A doua parte asigură deplasarea pe axa verticală. Partea a treia asigură apucarea obiectelor prin acționarea cleștelui(apucătorului).

### 3.2.1 Prima parte a brațului mecanic

Această parte este folosită pentru a deplasa întregul braț mecanic pe axa orizontală. Este alcătuită din un servo–motor micro SG90 și suportul metalic pentru acesta. Deplasarea se poate realiza între 0 și 180 de grade. Se consideră că atunci când brațul este în poziția de mijloc,servo-motorul are unghiul de deplasare setat în poziția de 90 de grade. Atunci când servo-motorul este setat în poziția de 0 grade, se consideră ca brațul este deplasat la maxim spre stânga. Atunci când servo–motorul este setat în poziția de 180 grade, se consideră că brațul este deplasat la maxim spre dreapta. Baza brațului mecanic este prezentată în Fig. 3.3.

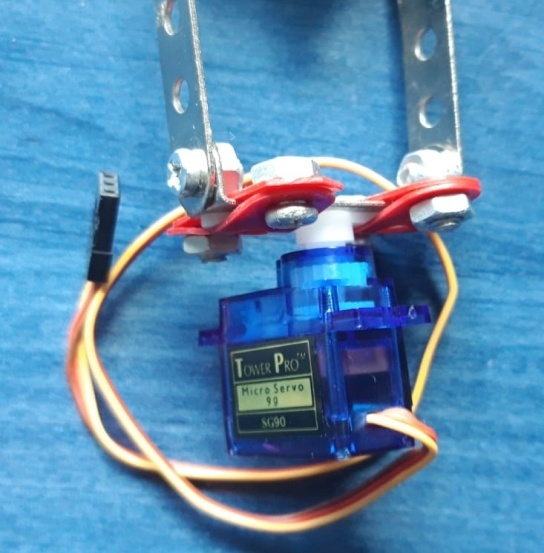


Fig. 3. Bază braț mecanic.

### 3.2.2 A doua parte a brațului mecanic

Această parte este folosită pentru a deplasa partea de sus a brațului mecanic, inclusiv cleștele, pe axa verticală. Este alcătuită din un servo–motor micro SG90 și suportul metalic pentru acesta. Deplasarea se poate realiza între 0 și 180 de grade. Se consideră că atunci când brațul este în poziția de mijloc, servo-motorul are unghiul de deplasare setat în poziția de 90 de grade. Atunci când servo-motorul este setat în poziția de 0 grade, se consideră ca brațul este deplasat la maxim în jos. Atunci când servo-motorul este setat în poziția de 180 grade, se consideră că brațul este deplasat la maxim în sus. A doua parte a brațului mecanic se poate observa în Fig. 3.4.

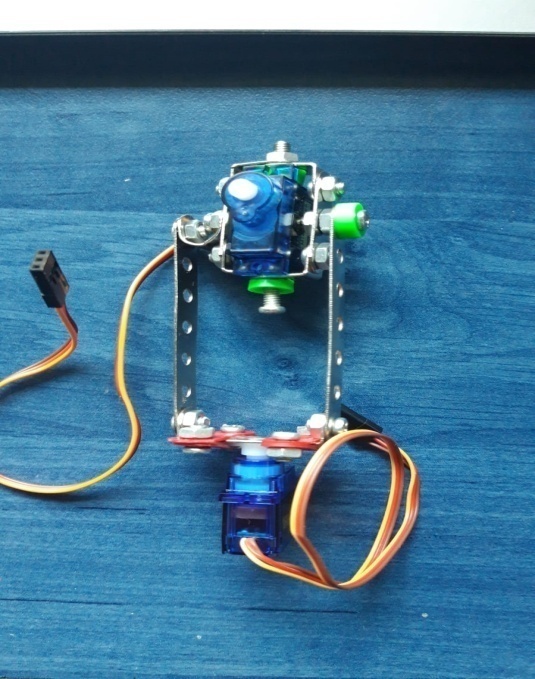


Fig. 3. Centru braț mecanic.

### 3.2.3 A treia parte a brațului mecanic

Această parte este folosită pentru a acționa cleștele brațului mecanic. În acest fel este posibilă apucarea și mutarea obiectelor. Este alcătuită din un servo–motor micro SG90, suportul metalic pentru acesta, din părțile metalice din care a fost alcătuit cleștele și din două sârme metalice folosite pentru acționarea cleștelui. Deplasarea servo–motorului din această parte este setată între 0 și 90 de grade. Cele două sârme se leagă de elicea servo–motorului și de cele două părți ale cleștelui. Când servo–motorul este setat în poziția de 0 grade cleștele este deschis. Atunci când servo–motorul este setat în poziția de 90 de grade, din cauza rotației sârmele se vor deplasa și în acest fel cleștele se va închide. În acest mod se realizează apucarea obiectelor. Atunci când servo–motorul revine în poziția de 0 grade cleștele se va deschide și în acest mod se va elibera obiectul, după ce a fost așezat în poziția dorită. A treia parte a brațului mecanic este prezentată în Fig. 3.5. Brațul mecanic complet asamblat se poate vizualiza în Fig. 3.6.



Fig. 3. Clește braț mecanic.

Prin interconectarea celor 3 părți va rezulta brațul mecanic capabil să se deplaseze pe axa orizontală și pe axa verticală. Un servo–motor sicro SG90 acceptă o tensiune de alimentare cuprinsă între 4.8 și 6 V de curent continuu. Servo–motorul SG90 consumă aproximativ 10 mA atunci când este în repaus și circa 250 mA atunci când este comandat să se deplaseze la un anumit unghi, într-o anumită direcție. Pe durata verificării funcționării motoarelor a fost suficientă alimentarea acestora din Arduino Nano, care la rândul său a fost alimentat de la laptop folosind un cablu microUSB-USB. Capătul microUSB a fost introdus în portul de microUSB al plăcii de dezvoltare Arduino Nano și celălalt capăt a fost introdus într-un port USB de la laptop, care poate furniza o tensiune de 5V în curent continuu. Deoarece în acest proiect este propusă autonomia brațului mecanic, cele trei motoare care alcătuiesc brațul vor fi alimentate cu o tensiune de 6V în curent continuu, folosind doi acumulatori Li - ion 18650 înseriați. Din cauză că tensiunea celor doi acumulatori înseriați depășește limita de 6V, cu care pot fi alimentate motoarele, a fost necesară folosirea unui coborâtor de tensiune LM2596 pentru a limita tensiunea de alimentare a motoarelor la 6V.

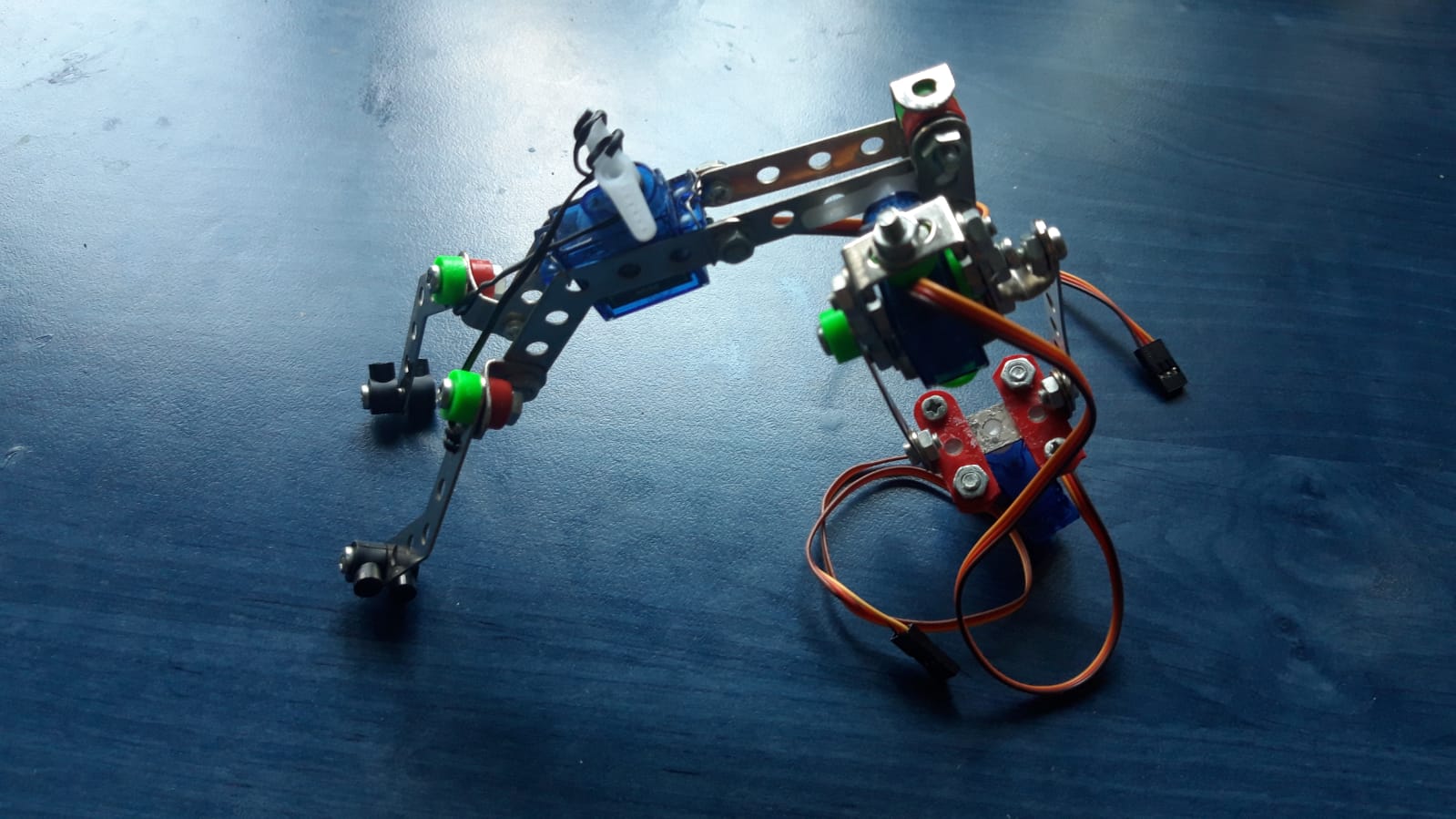


Fig. 3. Brațul mecanic complet.

## 3.3 Construcția robotului

Pentru robot au fost necesare: un kit cu două motoare DC de 3V, un *breadboard*, placa de dezvoltare Raspberry Pi modelul 3A+, fire de legătură, o punte H duală L298N, un senzor de temperatură și umiditate DHT11, un modul NRF24L01+, folosit pentru transmiterea informației *wireless* și un braț mecanic realizat din părți metalice și trei servo–motoare micro SG90.

### 3.3.1 Asamblarea kit-ului robotului

Kit-ul ales pentru acest proiect este unul foarte simplu, flexibil și practic. Datorită celor două motoare de curent continuu, robotul se poate deplasa în direcțiile față, dreapta, stânga, spate, având tracțiune spate. Toate aceste elemente fac acest kit potrivit pentru acest tip de aplicație.

Acest kit conține:

* un șasiu;
* două roți de cauciuc;
* două motoare de 3V ce pot fi dotate cu sistem de *encoder*;
* o a treia roată ce nu este acționată de motor și este folosită pentru deplasarea pe axa orizontală;
* suporți pentru motoare;
* șuruburi și piulițe necesare asamblării;

Dimensiunile acestui kit sunt de 15 cm x 20 cm. Kit-ul asamblat este vizibil în Fig. 3.7.

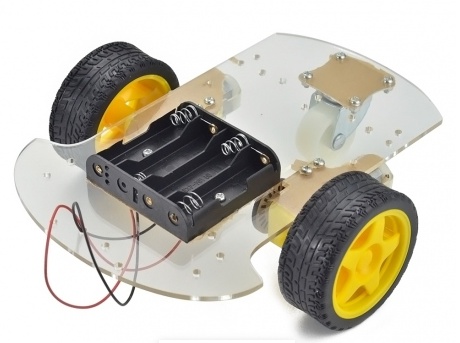


Fig. 3. Kit robot.

În prima etapă din asamblarea kit-ului se îndepărtează hârtia auto-adezivă de pe componentele acrilice ale kit-ului și anume șasiul și suporții pentru motoare. În a doua etapă se vor monta rotițele negre folosite pentru sistemele de numărare a impulsurilor și de asemenea se verifică funcționarea motoarelor. Deoarece aceste motoare au terminale care nu sunt polarizate, verificarea funcționării lor este foarte ușor de făcut. Se ia o sursă de alimentare și se conectează terminalul de masă al sursei la un terminal al motorului, iar pinul de alimentare al sursei se conectează la celălalt terminal al motorului. Dacă motorul funcționează corect, după conectarea sursei de alimentare ar trebui să înceapă să se învârtă la viteză maximă. Pentru a verifica dacă motorul se poate învârti și în sens opus, se vor conecta terminalele sursei de alimentare invers la terminalele motorului. După verificarea funcționării motoarelor urmează lipirea firelor de legătură la motoare. Se lipesc două fire de legătură, unul pentru terminalul de alimentare și celălalt pentru terminalul de masă. În ultima etapă se vor monta motoarele pe șasiu cu ajutorul suporților, șuruburilor și a piulițelor. De asemenea se vor monta roțile de cauciuc din spate și roata liberă din față. Acest kit conține și un suport pentru 4 baterii de tip AA de 1.5 V, dar acest suport nu a fost folosit, pentru că motoarele sunt controlate de o punte H duală L298N, care la rândul ei va consuma o tensiune de aproximativ 1.4V. În acest caz tensiunea minimă necesară pentru a controla motoarele este de 7.4 V. Această tensiune este atinsă folosind doi acumulatori 186500 în serie, fiecare având o tensiune cuprinsă între 3.7 – 4.2V. Etapele asamblării kit-ului se pot observa și în Fig. 3.8.

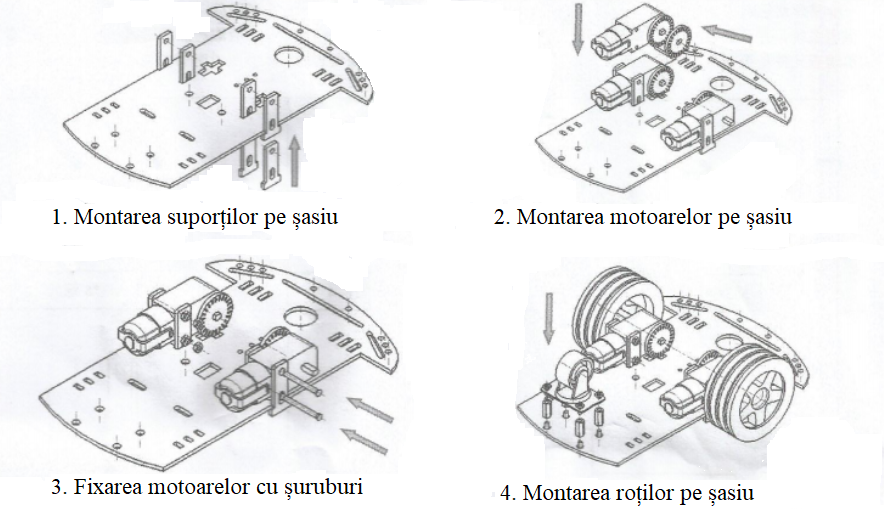


Fig. 3. Asamblare kit robot. [37]

### 3.3.2 Conectarea motoarelor DC la puntea H L298N

Puntea L298N poate controla atât direcția în care se deplasează cele două motoare, cât și viteza acestora de deplasare. În Fig. 3.9 este prezentat modul de conectare al motoarelor DC la puntea H.

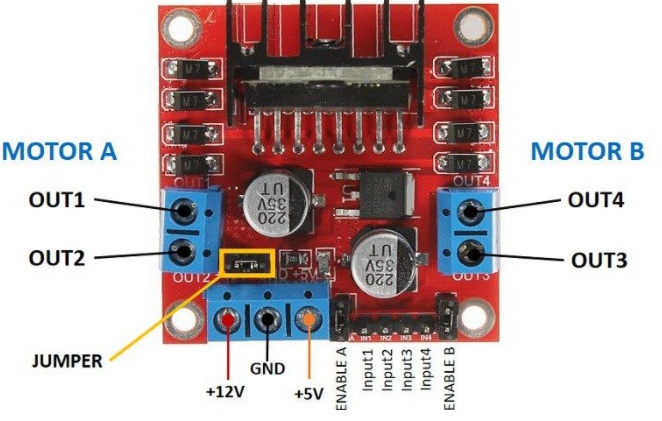


Fig. 3. Conectare motoare DC la puntea H.[24]

Motorul A se va conecta la terminalele *OUT1* și *OUT2* ale punții H. Motorul B se va conecta la terminalele *OUT3* și *OUT4*. Pinii *ENABLE* A și *ENABLE* B sunt pini de tipul PWM și sunt folosiți pentru controlul vitezei motoarelor. Pinul *ENABLE* A va fi folosit pentru motorul A, în timp ce pinul *ENABLE* B va fi folosit pentru motorul B. Pinii de *input* se vor conecta la placa de dezvoltare Raspberry Pi. Pinii de *input* sunt folosiți pentru a controla direcția de deplasare a motoarelor. Pinii *Input1* și *Input2* vor fi folosiți pentru a controla direcția de deplasare a motorului A, în timp ce pinii *Input3* și *Input4* vor fi folosiți pentru a controla direcția de deplasare a motorului B. Atunci când *jumper-ul* este pus, va fi activ regulatorul de 5V al punții. Pinul de +12V este folosit pentru a alimenta puntea și cele două motoare. Pinul de GND este pinul de masă la care se va conecta pinul de masă de la cei doi acumulatori în serie 18650, cu care se face alimentarea. Pinul de 5V nu este folosit, el poate furniza o tensiune continuă de 5V.

### 3.3.3 Realizarea conectării componentelor robotului

Pentru a putea realiza conexiunile între Raspberry Pi și celelalte componente ale robotului s-a folosit un *breadboard* și fire de legătură mamă–mamă, tată–tată sau mamă–tată. Raspberry Pi controlează și coordonează întreaga activitate a robotului. Raspberry Pi este dispozitivul principal al circuitului, el dictează activitatea celorlalte dispozitive periferice. Cu ajutorul transmițătorului radio NRF24L01+, Raspberry Pi primește informația de *input* de la telecomandă, o prelucrează, iar cu rezultatul obținut va putea comanda activitatea motoarelor, a senzorului de temperatură și umiditate DHT11 și a modului de cameră. Toate conexiunile între Raspberry Pi și celelalte componente ale robotului s-au realizat conform Fig. 1.4 din primul capitol. În Fig. 3.10 se poate vedea robotul complet asamblat, cu brațul mecanic și camera montate pe kit-ul de robot.

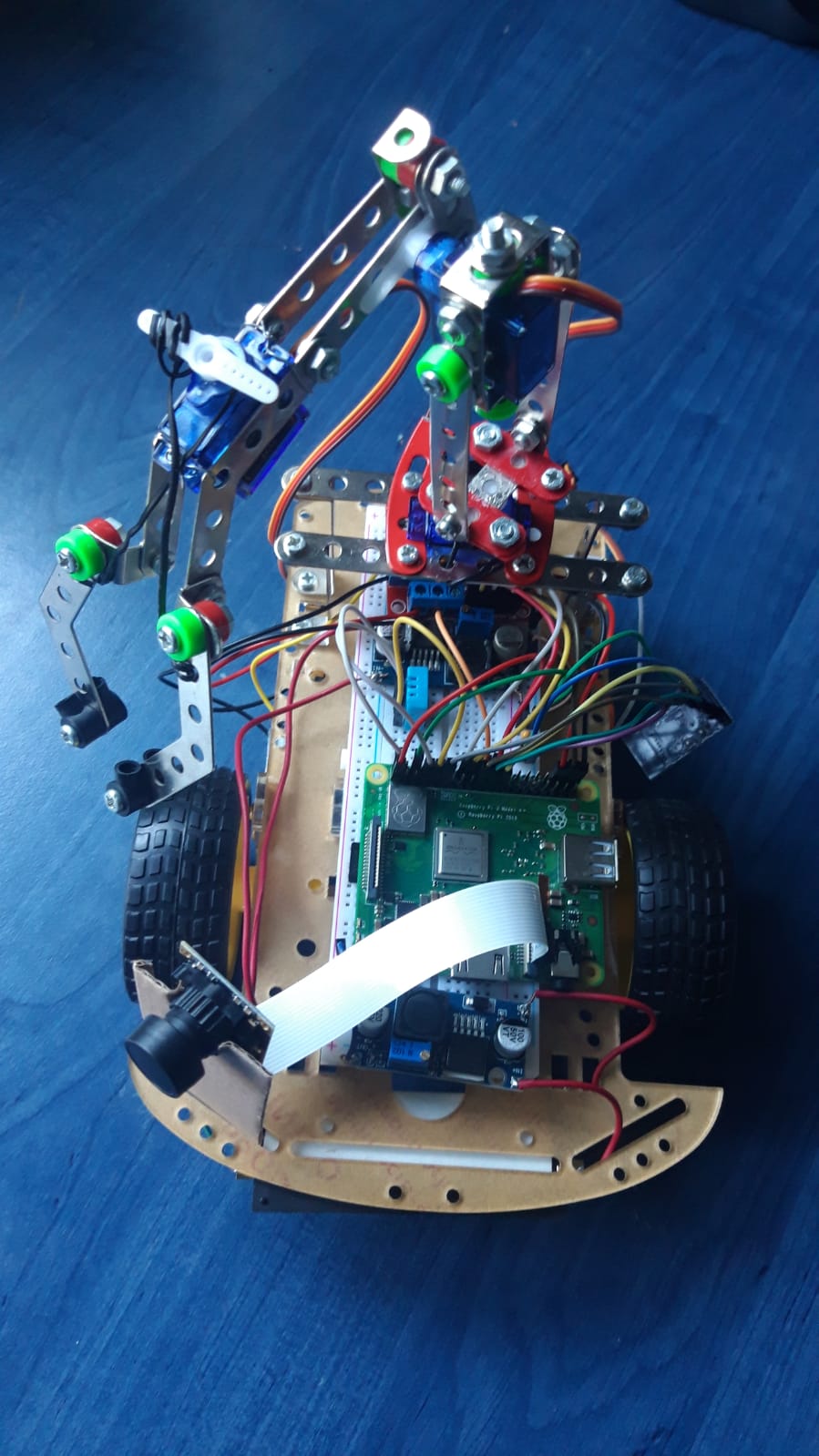


Fig. 3. Robot complet asamblat.

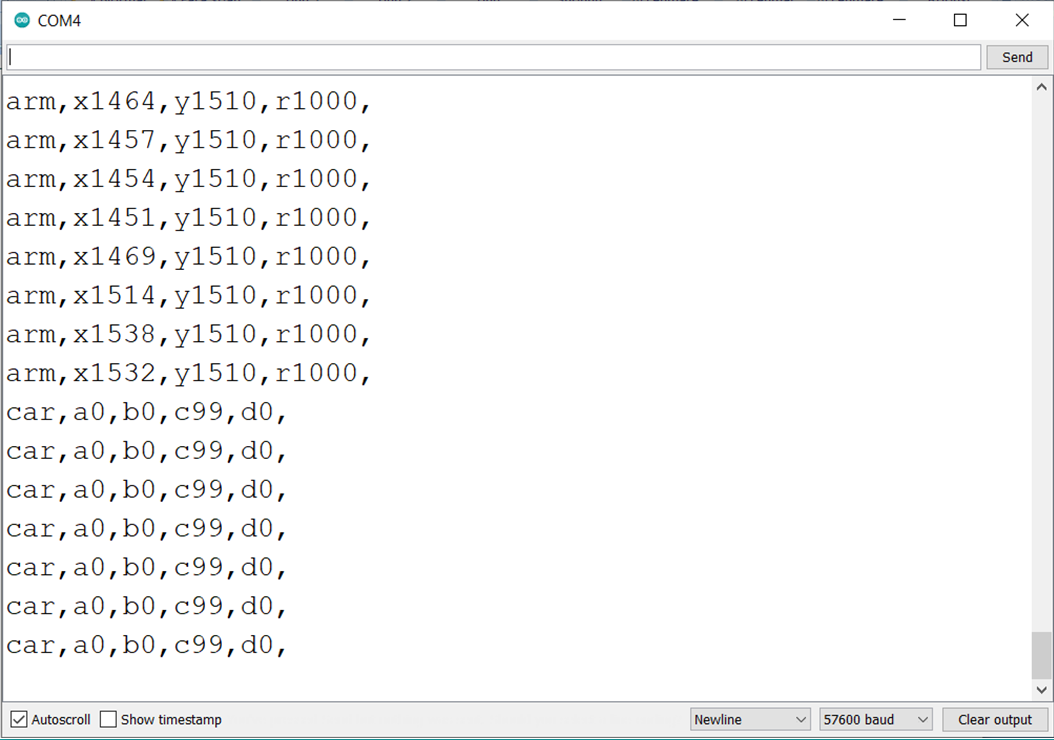
## 3.4 Modul de funcționare

Robotul mobil autonom este comandat prin telecomandă. Transmisia de informații între robot și telecomandă este una de tip *half-duplex*. Telecomanda și robotul comunică cu ajutorul celor două transmițătoare radio NRF24L01+ montate atât pe robot, cât și pe telecomandă. Când unul dintre *joystick-urile* de pe telecomandă este mișcat într-o direcție, robotul se va deplasa în direcția indicată de *joystick*. Când *joystick-ul* revine în poziția inițială, motoarele DC sau servo –motoarele vor intra în repaus. Pe telecomandă există două *joystick-uri*. Primul *joystick* este folosit pentru a controla motoarele DC de 3V, cu care se deplasează robotul mobil. Al doilea *joystick* este folosit pentru a controla servo–motoarele, cu care se deplasează brațul mecanic. Butonul de pe telecomandă are un rol esențial în trasnmisia informațiilor între telecomandă și robot. Butonul are două stări, starea de *HIGH*, adică butonul este neapăsat și starea de *LOW*, adică butonul este apăsat. Când butonul se află în starea de *HIGH*, datele se vor transmite într-o buclă infinită de la telecomandă și vor fi recepționate de către robot. Când butonul se află în starea *LOW*, se inversează sensul de transmitere al datelor. În acest caz, se vor transmite date privind temperatura și umiditatea mediului ambiental, de la robot, și vor fi ulterior recepționate de către telecomandă și afișate pe ecranul LCD. În tot acest timp se poate viziona în timp real traseul parcurs de robot cu ajutorul camerei compatibile cu Raspberry Pi, montată pe robotul mobil. Condiția pentru vizionarea traseului este ca dispozitivul de pe care se efectuează acest lucru să aibă o conexiune la internet și să se găsească în aceeași rețea cu placa de dezvoltare Raspberry Pi.

Pentru că toată activitatea telecomenzii este controlată de Arduino Nano și toată activitatea robotului este controlată de Raspberry Pi, a fost necesar crearea unui cod sursă pentru fiecare în parte.

În primul rând, pentru codul sursă al Arduino Nano realizat în C++, se includ librăriile necesare. În următoarea etapă este necesară definirea unui obiect de tipul LiquidCrystal\_I2C pentru LCD și un obiect de tipul RF24 pentru transmițătorul radio NRF24L01+. În continuare trebuie setate adresele pentru scriere și citire. Se setează adresa de scriere add = 0xF0F0F0F0E1LL și adresa de citire add1 = 0xE8E8F0F0E1LL. Aceste adrese vor fi setate în același mod și în codul sursă al lui Raspberry Pi. Dacă adresele de la Arduino diferă de cele de la Raspberry Pi nu se poate realiza comunicația. Se va seta canalul pe care se realizează comunicația radio.setChannel(0x76). După cum a fost explicat în primul capitol, există 125 de canale posibile pe care se poate face o comunicație. Se va seta viteza de transmisie a datelor la 1Mbps. De asemenea, se va activa opțiunea radio.enableDynamicPayloads(). Această opțiune este folositore, pentru că în acest mod datele se pot transmite mai rapid. Prin modulele NRF24L01+ se pot transmite pachete de date cu dimensiunea de maxim 32 bytes. Dacă nu se folosește opțiunea pentru *payload* dinamic, atunci se va aloca pentru fiecare pachet 32 bytes, chiar dacă pachetul are o dimensiune mai redusă. Prin folosirea opțiunii de *payload* dinamic pachetele de date vor avea o dimensiune variabilă între 1–32 bytes și astfel se vor putea transmite mai rapid. Se vor seta pinii analogici și cei digitali folosiți ca *INPUT* sau ca *OUTPUT*, de exemplu pinMode(A6, INPUT). În bucla infinită void loop() se inițializează variabile care să citească valorile pinilor analogici și digitali, care vor fi controlați din mișcarea *joystick-urilor*. Inițial la alimentarea telecomenzii, transmițătorul radio NRF24L01+ este pe post de emițător, radio.stopListening() și butonul este în starea de *HIGH*, adică neapăsat. Cât timp butonul este neapăsat se vor transmite în mod continuu pachete cu informații, coordonatele de deplasare pentru robotul mobil sau pentru brațul mecanic, de la telecomandă la robot. Acest lucru se poate observa în Fig. 3.11. Se pot transmite ori coordonate pentru deplasarea robotului mobil, ori unghiuri la care trebuie să se deplaseze servo–motoarele brațului mecanic, dar nu în același timp ambele. *Joystick-ul* se consideră în poziția de mijloc, nemișcat, atunci când pe axa orizontală și pe axa verticală se află între valorile 470–550, conform Fig. 1.20 din primul capitol. Dacă *joystick-ul* pentru controlul motoarelor robotului mobil se deplasează înafara valorilor 470–550, atunci se va forma într-o variabilă string un mesaj cu coordonatele de deplasare pentru motoare. Mesajul va avea o formă de genul car,a99,b0,c40,d0. Se poate observă că separatorul este virgula. Literele reprezintă direcția de deplasare: a este folosit pentru deplasarea în față, b este folosit pentru deplasarea în spate, c este folosit pentru deplasarea la dreapta și d este folosit pentru deplasarea la stânga. Numerele după litere reprezintă viteza cu care se vor deplasa motoarele în direcția respectivă, 0 reprezintă faptul că motorul este în repaus și 99 reprezintă faptul că motorul se deplasează cu viteză maximă. Din exemplul de mai sus a99,b0,c40,d0, robotul va deplasa primul motor înainte cu viteza maximă 99 și al doilea motor tot înainte, dar cu viteza 40. În acest mod se efectuează un viraj la dreapta. Dacă *joystick-ul* pentru controlul mișcării brațului mecanic se deplasează înafara valorilor 470–550, atunci se va forma într-o variabilă string un mesaj cu unghiurile de deplasare ale brațului mecanic. Mesajul va avea o formă de genul arm,x1528,y1512,b1500. Se poate observa că separatorul este tot virgula. Literele reprezintă servo–motorul care este acționat. Litera x va acționa primul servo–motor, litera y va acționa al doilea servo–motor și ultima literă este folosită pentru acționarea celui de-al treilea servo– motor. Ultima literă poate să fie b sau r. Când ultima literă este b, servo–motorul este rotit la 90 de grade și cleștele se închide. Când ultima literă este r, servo–motorul este setat în poziția de 0 grade și cleștele se deschide, eliberând obiectul după ce acesta a fost mutat. Numerele de după litere reprezintă unghiul la care se vor roti servo–motoarele. Aceste numere, când sunt recepționate de robotul mobil vor fi scalate în felul următor: valoarea 1000 reprezintă o rotire a servo–motorului la 0 grade, valoarea 1500 reprezintă o rotire a servo–motorului la 90 de grade și valoarea 2000 reprezintă o rotire a servo–motorului la 180 de grade. Toate celelalte valori între aceste interval vor fi de asemenea scalate, astfel încât servo–motoarele să se poată roti la orice unghi cuprins între 0–180 de grade, mai puțin servo–motorul care acționează cleștele, care este setat la o rotație maximă de 90 de grade din codul sursă. Din exemplul arm,x1528,y1512,b1500 cele trei servo–motoare vor fi în poziția de mijloc, aproximativ 90 de grade, iar cleștele va fi închis. Mesajele transmise de telecomandă robotului pot fi vizualizate în interfața serială a *software-ului* Arduino IDE, prin activarea acesteia folosind instrucțiunea Serial.begin(57600). Numărul 57600 reprezintă rata de simbol utilizată. Rata de simbol determină viteza comunicației prin canalul de date. Dacă butonul este trecut în starea de *LOW*, adică este apăsat,se va transmite într-o variabilă de tip char mesajul stop, urmând ca transmițătorul de pe telecomandă să treacă de la rolul de emițător la rolul de receptor, radio.startListening(). De asemenea se vor afișa pe ercanul LCD informațiile legate de umiditatea și temperatura mediului ambiental, pe care le va recepționa acum telecomanda de la robotul mobil.

Pentru codul sursă al Raspberry Pi realizat în Python 3, se importă librăriile necesare pentru programarea componentelor periferice:modul cameră, transmițător radio NRF24L01+, senzor digital DHT11. În următoarea etapă se definesc pinii de uz general GPIO ai Raspberry Pi, care vor fi folosiți pentru controlul motoarelor DC și a servo–motoarelor și se setează ca *INPUT* sau ca *OUTPUT*. Există două moduri de numerotare a pinilor GPIO. Primul mod BCM, păstrează numărul pinilor GPIO așa cum au fost ei definiți de producător. La al doilea mod, *BOARD*, numerotarea pinilor se face conform locației fizice a acestora pe placa de dezvoltare Raspberry Pi. În acest proiect s-a folosit modul BCM. În continuare se inițializează un obiect de tip radio pentru modulul NRF24L01+, un obiect de tip senzor pentru DHT11, clasele necesare pentru a putea realiza *streaming* video în formatul MJPEG, cu 24 de cadre/secundă, rezoluția 640x480 și pagina în html, care poate fi accesată pentru a putea vizualiza *stream-ul* video în timpul rulării codului sursă al Raspberry Pi. Accesarea paginii se va face utilizând adresa IP a Raspberry Pi 192.168.1.114 și portul 8000. Din testele efectuate în faza de testare a funcționării modului de cameră s-a constatat că atunci când rezoluția e mai mare, *stream-ul* video este mai greoi și se creează congestie și întârzieri. De aceea a trebuit să se găsească un echilibru între rezoluția imaginii și calitatea *stream-ului* video. În continuare se setează canalul, rata de transfer a datelor și se activează opțiunea de *payload* dinamic pentru transmițătorul radio NRF24L01+. Se setează pinii GPIO de tip PWM, care sunt folosiți pentru a controla viteza motoarelor DC. În bucla infinită atunci când nu se recepționează nici un mesaj se va executa instrucțiunea time.sleep(0.01). Când în sfârșit se recepționează un mesaj există trei opțiuni. Dacă mesajul recepționat este stop, transmițătorul, care inițial avea rolul de receptor își va schimba rolul în emițător. Se înregistrează o citire a temperaturii și umidității de la senzorul digital DHT11 într-un pachet de date, care este trimis către telecomandă pentru a fi recepționat și afișat pe ecranul LCD, după cum se poate vedea în Fig. 3.13. După acest pas transmițătorul de pe robot iși va schimba din nou rolul din emițător în receptor, pentru a putea recepționa următorul mesaj. În cazul în care mesajul recepționat nu este stop, înseamnă că s-a recepționat un mesaj cu coordonate pentru deplasarea motoarelor DC ale robotului sau pentru deplasarea servo–motoarelor brațului mecanic. Acest lucru se poate observa în Fig. 3.12. Mesajul va fi împărțit într-o variabilă de tip listă, cu ajutorul separatorului virgulă. Dacă primul element al listei este ,,car’’, atunci mesajul recepționat reprezintă coordonate pentru deplasarea motoarelor DC. Dacă primul element al listei este ,,arm’’, atunci mesajul recepționat reprezintă unghiurile la care trebuie să se deplaseze brațul mecanic. Codul sursă atât pentru Raspberry Pi, cât și pentru Arduino Nano,ce au fost comentate în această secțiune, pot fi vizualizate în Anexa 1 de la finalul proiectului. În figurile de mai jos se poate vizualiza modul în care se afișează coordonatele de deplasare pentru robot și pentru brațul mecanic în portul serial de la Arduino și de la Raspberry Pi. De asemenea se poate vizualiza modul în care se afișează pe ecranul LCD măsurătorile de temperatură și umiditate de la senzorul digital DHT11 și cum arată *stream-ul* video realizat de modulul de cameră compatibil cu Raspberry Pi.

**** Fig. 3. Datele transmise de ATmega328P în monitorul serial din Arduino Software(IDE).

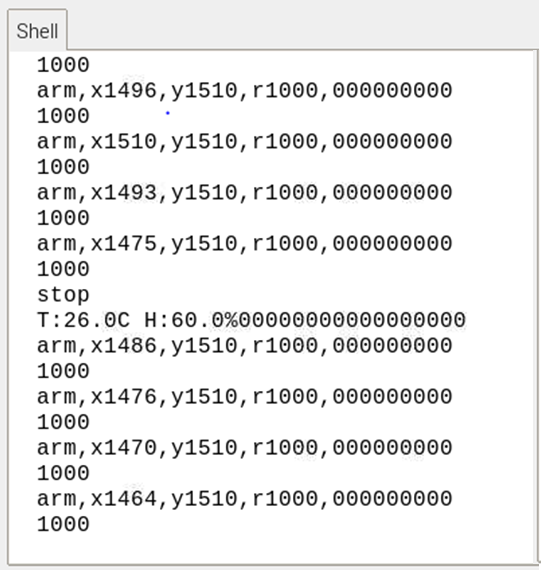
****

Fig. 3. Datele recepționate și transmise de Raspberry Pi în consola din Thonny IDE.

****

Fig 3. Afișare măsurători temperatură și umiditate pe ecranul LCD.

Un cadru al *stream-ului* video care este disponibil atât timp cât rulează codul sursă al plăcii de dezvoltare Raspberry Pi, se poate vizualiza în Fig. 3.14.

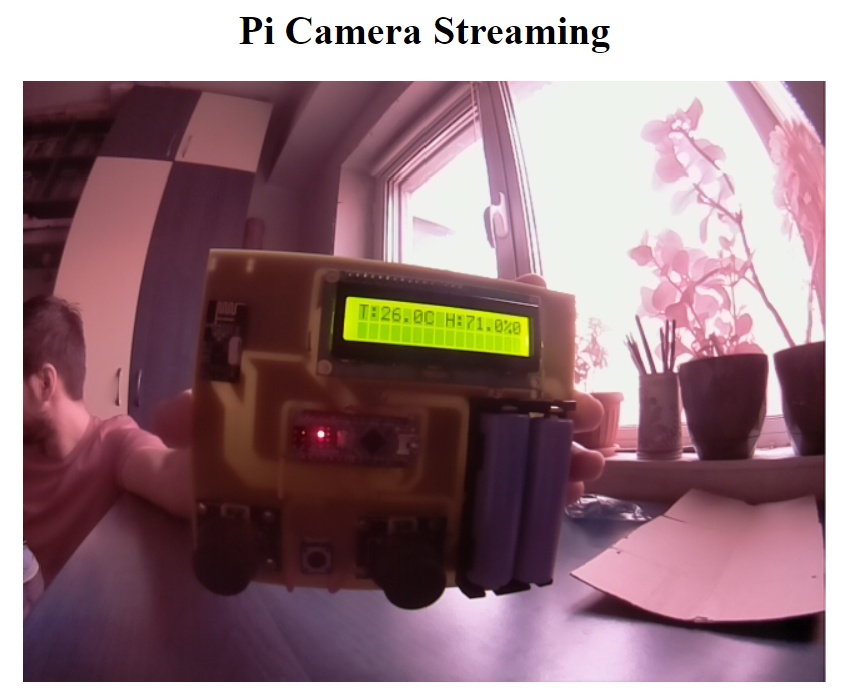


Fig 3. Captura unui cadru din *stream-ul* video transmis de cameră.

# 

# 

# 

# CONCLUZII ȘI VIITOARE ÎMBUNĂTĂȚIRI

A fost creat un robot mobil autonom, care poate îndeplini funcționalitățile propuse inițial pentru proiect. Robotul este comandat de la o distanță maximă de aproximativ 90-100 de metri. Robotul este capabil să recepționeze pachete de date și să transforme aceste informații în acțiuni fizice, ce determină mișcarea motoarelor DC ale robotului sau mișcarea brațului mecanic. Pentru a putea vizualiza de pe un dispozitiv(telefon, laptop sau tabletă) *stream-ul* video generat de camera robotului, ele trebuie să împartă aceeași rețea și aceeași conexiune la internet cu placa de dezvoltare Raspberry Pi.

* La proiectarea PCB-ului pentru telecomandă a apărut o eroare de proiectare privind dimensiunea alocată pentru suportul de acumulatori Li-ion 18650. Dimensiunea *footprint-ului* ales în program, a fost mai mică decât dimensiunea reală a suportului. Inițial suportul pentru acumulatori trebuia montat pe *layer-ul* de *BOTTOM*, dar în urma acestei erori a fost montat pe *layer-ul* de *TOP* cu bandă dublu adezivă, iar conexiunile au fost realizate cu fire de legătură din cupru lipite cu o stație letcon și cositor. De asemenea a apărut o altă eroare de proiectare privind dimensiunile pentru unele găuri pentru pinii componentelor THT, de pe PCB–ul telecomenzii. Dimensiunea de găurire a fost setată la 28mils(0.7 mm), iar dimensiunea necesară pentru pini este de 36mils(0.9 mm). Această eroare a fost corectată, pentru a putea insera componentele pe PCB, cu ajutorul unui burghiu de 0.9 mm.
* Servo-motoarele care alcătuiesc brațul mecanic sunt acționate de impulsuri fine, de ordinul micro secundelor, pentru că se folosește biblioteca pigpio din Python 3 pentru controlul servo-motoarelor, dar chiar și așa uneori pot avea mici zvâcniri. Din această cauză mișcarea brațului mecanic poate părea uneori că nu este foarte fluentă. Mișcarea servo–motoarelor este limitată de producător la 180 de grade. Al treilea capitol rezolvă problema deplasării brațului mecanic pe două axe, dar pentru o mobilitate mai bună se pot modifica servo–motoarele astfel încât să poată efectua o rotație completă de 360 de grade, prin scoaterea din potențiometrul motoarelor a opritorului pentru rotirea mai mare de 180 de grade. De asemenea va fi nevoie de două rezistoare de 1kΩ legate în serie, ce trebuie conectate între firul de la potențiometru și la *driver-ul* de motor. Se mai poate adăuga încă un servo–motor brațului mecanic pentru a mări raza de acțiune și calitatea mișcării. Aceste lucruri reprezintă direcții viitoare de dezvoltare.
* Realizarea comunicației *half-duplex* între transmițătoarele radio poate fi problematică, dacă nu sunt setate corespunzător adresele de scriere și citire, canalul radio pe care se realizează comunicația, rata de transfer a datelor, puterea de emisie a antenelor transmițătoarelor, sau dacă există incompatibilități între bibliotecile pentru Arduino și Raspberry Pi, folosite pentru configurarea transmițătoarelor. Pentru a putea comunica, transmițătoarele trebuie să aibă toate aceste setări făcute în mod identic. Pe viitor se poate proiecta un PCB și pentru robot pentru a evita conexiunile de proastă calitate realizate cu firele de legătură și *breadboard*. Transmițătorul radio NRF24L01+ este cel mai afectat de conexiunile de proastă calitate și comunicația între robot și telecomandă se poate întrerupe relativ ușor. Transmițătorul de pe PCB-ul telecomenzii este mult mai stabil din acest punct de vedere.
* *Stream-ul* video realizat cu ajutorului modulului de cameră are niște întârzieri, calitatea imaginii și rezoluția nu sunt de foarte bună calitate. Ca și direcție viitoare de îmbunătățire se poate implementa un algoritm mai performant, utilizând un standard care realizează codarea intercadru, cum ar fi MPEG1 și poate asigura un raport de compresie mai bun decat standardul MJPEG, folosit în acest proiect. Dacă întârzierile devin mai mici se poate încerca o rezoluție mai mare. În acest proiect a fost aleasă o rezoluție mică, de 640x480, pentru a evita întârzieri și mai mari.
* Pentru a mări distanța de la care se poate controla robotul, se pot folosi module radio NRF24L01+PA+LNA. Aceste module au în plus un amplificator de putere ce poate mări raza de transmisie considerabil, ajungând la o valoare de 1000 de metri într-un spațiu fără obstacole sau alte fenomene perturbatoare. Transmițătoarele radio NRF24L01+PA+LNA reprezintă o îmbunătățire semnificativă ce poate fi adusă acestui proiect, deoarece se extinde raza de acțiune de aproximativ 10 ori.
* Robotul mobil este autonom, dar nu are un sistem de încărcare pentru acumulatori montat pe el. Pentru încărcarea acumulatorilor aceștia trebuie scoși și încărcați separat, folosind un încărcător de tipul TP4056. Ca viitoare îmbunătățire se poate monta un sistem capabil să încarce acumulatorii, fără a fi nevoie ca aceștia să fie scoși, încărcați pe rând și montați la loc pe robotul mobil.
* O altă viitoare îmbunătățire o poate reprezenta montarea unei carcase pe robotul mobil, pentru a acoperi firele de legătură, placa de dezvoltare Raspberry Pi, senzorul DHT și transmițătorul radio NRF24l01+. Pentru acest lucru trebuie făcute măsurători, pentru a vedea în ce fel este afectată transmisiunea radio a antenei modulului NRF24L01+ de carcasă. De asemenea trebuie verificat dacă măsurătorile senzorului digital DHT11 vor fi afectate de carcasă.

Acest proiect a fost important pentru mine deoarece am dobândit experiență în proiectarea unui cablaj imprimat folosind un program de tip CAD. De asemenea am dobândit cunoștințe despre interfețe de comunicație serială cum ar fi SPI sau I2C, am dobândit cunoștințe despre realizarea unei comunicații de tip *half-duplex*, utilizând transmițătoare radio care pot avea atât rolul de emițător, cât și de receptor, deci comunicația este bidirecțională. Nu în ultimul rând am învățat un nou limbaj de programare și anume Python și am învățat să lucrez cu o placă de dezvoltare de tipul Raspberry Pi.

# 

# 

# BIBLIOGRAFIE

[1] Stelian-Emilian Oltean, ’’Mobile Robot Platform with Arduino Uno and Raspberry Pi for Autonomous Navigation,’’ 2019 12th International Conference Interdisciplinarity in Engineering, 2019, Online la: <https://www.sciencedirect.com/science/article/pii/S2351978919302896>, accesat la data de 13.10.2020.

[2] Viren Pereira, ’’Low Cost Object Sorting Robotic Arm using Raspberry Pi,’’ 2014 IEEE GHTC – SAS, 2014, Online la: <https://ieeexplore.ieee.org/abstract/document/6967550>, accesat la data de 13.10.2020.

[3] Hemalatha Pologanti, ’’Real time Image Processing based Robotic Arm Control Standalone System using Raspberry Pi,’’ 2015 SSRG – IJECE, 2015, Online la: <https://www.researchgate.net/publication/287641055_Real_time_Image_Processing_based_Robotic_Arm_Control_Standalone_System_using_Raspberry_pi>, accesat la data de 13.10.2020.

[4] Sachin S. Pujari, ’’Remotely Controlled Autonomous Robot using Android Application,’’ 2017 International Conference on I-SMAC, 2017, Online la: <https://ieeexplore.ieee.org/abstract/document/8058248>, accesat la data de 14.10.2020.

[5] Gökay Berkay, ’’Mobile robotic arm with real time video streaming,’’ 2019, Online la: [https://openaccess.maltepe.edu.tr/xmlui/handle/20.500.12415/3556](http://openaccess.maltepe.edu.tr/xmlui/handle/20.500.12415/3556), accesat la data de 14.10.2020.

[6] ’’What is MCU and How do Microcontroller Units Work,” 2021, Online la: <https://resources.pcb.cadence.com/blog/2020-what-is-an-mcu-and-how-do-microcontroller-units-work>, accesat la data 07.05.2021.

[7] En.wikichip.org, ’’Microprocessor(MPU),” 2021, Online la: <https://en.wikichip.org/wiki/microprocessor>, accesat la data de 07.05.2021.

[8] Ravi Engineer, ’’What are the differences between Raspberry Pi and Arduino,” Electronics Hub, 2021, Online la: <https://www.electronicshub.org/raspberry-pi-vs-arduino/>, accesat la data de 07.05.2021.

[9] Ro.wikipedia.org, ’’Circuit imprimat,” 2021, Online la: <https://ro.wikipedia.org/wiki/Circuit_imprimat>, accesat la data de 07.05.2021.

[10] Rush PCB, ’’IMPORTANCE OF PCBS AND PCB DESIGN,” 2021, Online la: <https://rushpcb.com/importance-of-pcbs-and-pcb-design/>, Accesat la data de 10.05.2021.

[11] En.wikipedia.org, ’’EAGLE(program),” 2021, Wikipedia, Online la: [https://en.wikipedia.org/wiki/EAGLE\_(program)](https://en.wikipedia.org/wiki/EAGLE_(program)%20), Accesat la data de 10.05.2021.

[12] Cadence, ’’Gerber vs. CAD,” 2021, Online la: <https://resources.pcb.cadence.com/blog/gerber-vs-cad-pcb-file-formats-demystified-2>, Accesat la data de 10.05.2021.

[13] Prof. Norocel Codreanu, Tehnici CAD de Realizare a Modulelor Electronice – notițe de

curs și laborator 2018-2019, Online la: <https://www.cetti.ro/v2/tehnicicad.php>.

[14] Instructables.com, ’’PCB Creation With Eagle for Beginners,” 2021, Online la: <https://www.instructables.com/PCB-Creation-with-Eagle-for-Beginners/>, Accesat la data de 15.03.2021.

[15] Diymodules.org, ’’DIYmodules,” 2021, Online la: <https://www.diymodules.org/>, Accesat la data de 15.03.2021.

[16] Raspberry Pi, ’’What is a Raspberry Pi,” 2021 Online la: <https://www.raspberrypi.org/help/what-%20is-a-raspberry-pi/>, Accesat la data de 18.05.2021.

[17] Raspberry Pi Spy, ’’Introducing the Raspberry Pi 3 Model A+,” 2021, Online la: <https://www.raspberrypi-spy.co.uk/2018/11/introducing-raspberry-pi-3-model-a-computer/>, accesat la data de 18.05.2021.

[18] Projects.raspberrypi.org, ’’Getting started with Raspberry Pi,” 2021, Online la: <https://projects.raspberrypi.org/en/projects/raspberry-pi-getting-started>, accesat la data de 18.05.2021.

[19] Scott Campbell, ’’Remote Desktop Connection,” 2021, Circuit Basics, Online la: <https://www.circuitbasics.com/access-raspberry-pi-desktop-remote-connection/>, accesat la 18.05.2021.

[20] Arduino.cc 2021, Datasheet Arduino – ArduinoBoardNano, 2020, Online la: <https://www.arduino.cc/en/pmwiki.php?n=Main/ArduinoBoardNano>, accesat la data de 13.10.2020.

[21] Lastminuteengineers.com, ’’nRF24L01+,” 2020, Online la: <https://lastminuteengineers.com/nrf24l01-arduino-wireless-communication/>, accesat la data de 13.10.2020.

[22] Monolithicpower.com, ’’Stepper Motors: Types, Uses and Working Principle,” 2021, Online la: <https://www.monolithicpower.com/en/stepper-motors-basics-types-uses>, accesat la data de 19.05.2021.

[23]Ee.ic.ac.uk, Datasheet SG90, 2021, Online la: <http://www.ee.ic.ac.uk/pcheung/teaching/DE1_EE/stores/sg90_datasheet.pdf>, accesat la data de 27.04.2021.

[24] HowToMechatronics, ’’L298N Motor Driver – Arduino Interface, How It Works, Codes, Schematics,” 2021, Online la: <https://howtomechatronics.com/tutorials/arduino/arduino-dc-motor-control-tutorial-l298n-pwm-h-bridge/>, accesat la data de 25.03.2021.

[25] Media.Digikey.com, Datasheet Cameră pentru Raspberry Pi cu vedere pe timp de noapte și focalizare reglabilă, 2021, Online la: <https://media.digikey.com/pdf/Data%20Sheets/DFRobot%20PDFs/SEN0184_Web.pdf>, accesat la 27.03.2021.

[26] Projects.raspberrypi.org, ’’Getting started with the Camera Module,” 2021, Online la <https://projects.raspberrypi.org/en/projects/getting-started-with-picamera/3>, accesat la data de 27.03.2021.

[27] Prof. Mircea Răducanu, Tehnici de compresie a semnalelor multimedia – notițe de curs 2020-2021

[28]Onsemi.com, Datasheet LM2596, 2021, producător ON Semiconductor, Online la: <https://www.onsemi.com/pdf/datasheet/lm2596-d.pdf> , accesat la data de 27.03.2021

[29] Lastminuteengineers.com, ’’2-Axis Joystick,” 2021, Online la: <https://lastminuteengineers.com/joystick-interfacing-arduino-processing/>, accesat la data de 28.03.2021.

[30] Electronics Hub, ’’Raspberry Pi DHT11 Humidity and Temperature Sensor Interface,” 2021, Online la: <https://www.electronicshub.org/raspberry-pi-dht11-humidity-temperature-sensor-interface/>, accesat la data de 28.03.2021.

[31] Opencircuit.shop, Datasheet LCD cu interfață I2C și backlight galben-verde, Online la: <https://opencircuit.shop/resources/file/da88acc1702a90667728fcf4ac9c75c455475706466/I2-LCD-interface.pdf>, accesat la data de 30.03.2021.

[32]Ro.wikipedia.org , ’’Limbaj de programare,” 2021, Wikipedia, Online la: <https://ro.wikipedia.org/wiki/Limbaj_de_programare>, accesat la data de 22.05.2021.

[33] En.wikipedia.org, ’’C++,” 2021, Wikipedia, Online la: <https://en.wikipedia.org/wiki/C%2B%2B>, accesat la data de 22.05.2021.

[34] En.wikipedia.org, ’’Python (programming language),” 2021, Wikipedia, Online la: <https://en.wikipedia.org/wiki/Python_(programming_language)>, accesat la data de 22.05.2021.

[35] Pantech Blog, ’’Installing Library package in Raspberry Pi-Chapter 2,” 2020, Online la: <https://www.pantechsolutions.net/blog/installing-library-packages-in-raspberry-pi/>, accesat la data de 13.10.2020.

[36] Joni Trythall, ’’Web Design: Structura HTML,’’ 2015, Online la: <https://webdesign.tutsplus.com/ro/tutorials/web-design-for-kids-html-structure--cms-23763>, accesat la data de 22.05.2021.

**[**3**7]** Optimus Digital, Kit Robot cu 2 Motoare pentru Arduino / Raspberry PI DIY, 2021, Online la: <https://www.optimusdigital.ro/ro/robotica-kit-uri-de-roboti/140-kit-robot-2-motoare.html?search_query=kit+robot+&results=67>, accesat la data de 05.06.2021.

# ANEXA 1 COD SURSĂ

Codul sursă pentru ATmega328P:

//includere biblioteci

#include <RF24.h>

#include<SPI.h>

#include<DigitalIO.h>

#include <LiquidCrystal\_I2C.h>

#include <Wire.h>

// Inițializare obiecte și variabile folosite

LiquidCrystal\_I2C lcd(0x27,16,2);

RF24 radio(9,10);

const uint64\_t add=0xF0F0F0F0E1LL;

const uint64\_t add1=0xE7E7E7E7E7LL;

//ce = 9, csn = 10 pins

int16\_t value\_x=0,value\_y=0,value\_sw = 0, value\_x1 = 0, value\_y1 = 0, value\_sw1 = 0;;

const int buttonPin = 4;

int buttonState ;

int buttonState\_sw ;

// Setarea pinilor folosiți ca I/O, setarea transmițătorului radio și a ecranului LCD.

void setup() {

Serial.begin(57600);

SPI.begin();

pinMode(A6, INPUT);

pinMode(A7, INPUT);

pinMode(A0, INPUT);

pinMode(A1, INPUT);

pinMode(2, INPUT\_PULLUP);

pinMode(8, INPUT\_PULLUP);

pinMode(buttonPin, INPUT\_PULLUP);

digitalWrite(buttonState, HIGH);

radio.begin();

radio.setPALevel(RF24\_PA\_MIN);

radio.setChannel(0x76);

radio.setDataRate(RF24\_1MBPS);

radio.setAutoAck(false);

radio.openWritingPipe(add) ;

radio.openReadingPipe(1,add1);

radio.enableDynamicPayloads();

radio.powerUp();

lcd.init();

lcd.backlight();

lcd.setCursor(0,0); }

// Intrarea în instrucțiunea loop() infinită

void loop() {

radio.stopListening();

float pos = 1000;

delay(100);

//Citirea valorilor de la pinii analogici și digitali

value\_x = analogRead(A6);

value\_y = analogRead(A7);

value\_sw1 = digitalRead(2);

value\_x1 = analogRead(A0);

value\_y1 = analogRead(A1);

String text,s1,s2,s3,s4,a1,a2,a3;

//Variabile pentru citirea stării butonului de pe telecomandă și a switch-ului de la joystick-ul pentru brațul mecanic.Când switch–ul este apăsat, cleștele brațului apucă obiectul.

buttonState = digitalRead(buttonPin);

buttonState\_sw = digitalRead(2);

// Ce se întâmplă dacă butonul este neapăsat

if(buttonState==HIGH){

//dacă joystick-ul pentru deplasarea robotului este în poziție centrală, nemișcat, atunci se va forma un pachet de date cu coordonatele de deplasare ale brațului mecanic.

if(value\_x>470 && value\_x<550 && value\_y>470 && value\_y<550)

{a1="x"+String(int(value\_x1+pos))+",";

a2="y"+String(int(value\_y1+pos))+",";

if(buttonState\_sw==LOW)

a3="b"+String(1500)+",";

else

a3="r"+String(1000)+",";

// Formarea pachetului cu coordonatele de deplasare ale brațului mecanic

text="arm,"+a1+a2+a3;

char mesaj[32];

text.toCharArray(mesaj,32);

Serial.println(mesaj);

// Transmiterea pachetului de date format către robot

radio.write(&mesaj, sizeof(mesaj));

delay(500); } else{

//În cazul în care joystick-ul pentru deplasarea brațului mecanic este în poziție centrală, nemișcat, atunci se va forma un pachet de date cu coordonatele de deplasare ale robotului mobil.

if(value\_x<470){

s3="c0";

s4="d"+String(int(abs(value\_x-470)/4.71));

}else if(value\_x>550) {

s3="c"+String(int((value\_x-550)/4.74));

s4="d0";

}else{

s3="c0";

s4="d0"; }

if(value\_y<470){

s1="a0";

s2="b"+String(int(abs(value\_y-470)/4.71));

}else if(value\_y>550){

s1="a"+String(int((value\_y-550)/4.74));

s2="b0";

}else{

s1="a0";

s2="b0"; }

// Formarea pachetului cu coordonatele de deplasare ale robotului mobil

text="car,"+s1+","+s2+","+s3+","+s4+",";

char mesaj[32];

text.toCharArray(mesaj,32);

Serial.println(mesaj);

// Transmiterea pachetului de date format către robot

radio.write(&mesaj, sizeof(mesaj));

delay(500); }}else{

//Ce se întâmplă dacă butonul este apăsat

char a[32]="stop";

//Trimiterea mesajului "stop" către robot

radio.write(&a, sizeof(a));

delay(200);

radio.startListening();

delay(200);

if(radio.available()){

while(radio.available()){

char text1[32]={0};

// Recepționarea pachetului de date cu informațiile pentru temperatură și umiditate de la senzorul DHT11 al robotului

radio.read(&text1, sizeof(text1));

Serial.println(text1);

lcd.clear();

//Afișarea temperaturii și umidității pe ecranul LCD al telecomenzii

lcd.print(text1);

radio.stopListening(); } }

delay(500);

}}

Codul sursă pentru Raspberry Pi:

# Importarea bibliotecilor folosite

import RPi.GPIO as GPIO

from lib\_nrf24 import NRF24

import pigpio

import time

import spidev

import Adafruit\_DHT

import io

import picamera

import logging

import socketserver

from threading import Condition

from http import server

# Crearea paginii Web pentru vizualizarea stream-ului video cu modulul de cameră compatibil cu Raspberry Pi, cu ajutorul limbajului HTML.

PAGE="""\

<html>

<head>

<title>Pi Camera Streaming</title>

<style>

body {

background-image: url(https://lh3.googleusercontent.com/proxy/p0rdn1qcalA3TIi9QkxOx-CgL\_3bsjvRByOqI7Ynn6tYa7DCsIxex0uzd5smXW4p-ZASQ-xjiSUtp-tKVCuKe0epo5sPCJmWOOFRlRPyjzp\_HMIhnfwzix6fouLpWQ995ixPiQxBjQ);

background-repeat: no-repeat;

background-attachment: fixed;

background-size: 100% 100%;}

</style>

</head>

<body>

<center><h1>Pi Camera Streaming</h1></center>

<center><img src="stream.mjpg" width="640" height="480"></center>

</body>

</html>

"""

# Crearea clasei StreamingOutput

class StreamingOutput(object):

def \_\_init\_\_(self):

self.frame = None

self.buffer = io.BytesIO()

self.condition = Condition()

def write(self,buf):

if buf.startswith(b'\xff\xd8'):

self.buffer.truncate()

with self.condition:

self.frame = self.buffer.getvalue() self.condition.notify\_all()

self.buffer.seek(0)

return self.buffer.write(buf)

#Crearea clasei StreamingHandler prin care se realizează conexiunea

class StreamingHandler(server.BaseHTTPRequestHandler):

def do\_GET(self):

if self.path == '/':

self.send\_response(301) self.send\_header('Location','/index.html')

self.end\_headers()

elif self.path == '/index.html':

content = PAGE.encode('utf-8')

self.send\_response(200)

self.send\_header('Content-Type', 'text/html')

self.send\_header('Content-Length', len(content))

self.end\_headers()

self.wfile.write(content)

elif self.path == '/stream.mjpg':

self.send\_response(200)

self.send\_header('Age',0)

self.send\_header('Cache-Control', 'no-cache, private')

self.send\_header('Pragma', 'no-cache')

self.send\_header('Content-Type', 'multipart/x-mixed-replace; boundary=FRAME')

self.end\_headers()

try:

while True: with output.condition: output.condition.wait()

frame = output.frame self.wfile.write(b'--FRAME\r\n') self.send\_header('Content-Type', 'image/jpeg') self.send\_header('Content-Length', len(frame)) self.end\_headers() self.wfile.write(frame)

self.wfile.write(b'\r\n')

except Exception as e:

logging.warning(

'Removed streaming client %s: %s', selfclient\_address, str(e))

else:

self.send\_error(404)

self.end\_headers()

#Crearea clasei StreamingServer, în această clasă se setează rezoluția imaginii, numărul de cadre/secundă, formatul, adresa la care va fi accesat site-ul.

class StreamingServer(socketserver.ThreadingMixIn, server.HTTPServer):

allow\_reuse\_address = True

daemon\_threads = True

with picamera.PiCamera(resolution = '640x480', framerate=24) as camera:

output = StreamingOutput()

#camera.rotation = 90

camera.start\_recording(output, format='mjpeg')

try:

address = ('192.168.1.114',8000)

server = StreamingServer(address, StreamingHandler)

server.serve\_forever()

finally:

camera.stop\_recording()

#Setarea pinilor folosiți în proiect ca I/O, inițializarea senzorului DHT11 și a bibliotecii necesare pentru controlul servo – motoarelor brațului mecanic.

sensor = Adafruit\_DHT.DHT11

DHT11\_pin = 4

pi = pigpio.pi()

in1 = 24

in2 = 23

enA = 25

in3 = 20

in4 = 16

enB = 21

GPIO.setwarnings(False)

GPIO.setmode(GPIO.BCM)

GPIO.setup(in1,GPIO.OUT)

GPIO.setup(in2,GPIO.OUT)

GPIO.setup(enA,GPIO.OUT)

GPIO.setup(in3,GPIO.OUT)

GPIO.setup(in4,GPIO.OUT)

GPIO.setup(enB,GPIO.OUT)

GPIO.output(in1,GPIO.LOW)

GPIO.output(in2,GPIO.LOW)

GPIO.output(in3,GPIO.LOW)

GPIO.output(in4,GPIO.LOW)

# Setarea transmițătorului radio NRF24L01+

pipes = [[0xE8, 0xE8, 0xF0, 0xF0, 0xE1], [0xF0, 0xF0, 0xF0, 0xF0, 0xE1]]

radio = NRF24(GPIO, spidev.SpiDev())

radio.begin(0, 17)

radio.setPayloadSize(32)

radio.setChannel(0x76)

radio.setDataRate(NRF24.BR\_1MBPS)

radio.setPALevel(NRF24.PA\_MAX)

radio.setAutoAck(False)

radio.enableDynamicPayloads()

radio.enableAckPayload()

radio.openWritingPipe(pipes[0])

radio.openReadingPipe(1, pipes[1])

radio.startListening()

#Setarea pinilor PWM pentru controlul vitezei motoarelor DC

p=GPIO.PWM(enA,255)

q=GPIO.PWM(enB,255)

# Intrarea în instrucțiunea while infinită

while True:

ackPL = [1]

# Se așteaptă până se recepționează un pachet de date

while not radio.available(0):

time.sleep(0.01)

receivedMessage = []

# Se recepționează un pachet de date

radio.read(receivedMessage, radio.getDynamicPayloadSize())

string = ""

#Se transformă din codul ASCII în caractere

for n in receivedMessage:

if (n >= 32 and n <= 126) :

string +=chr(n)

print(string)

#Dacă se recepționează mesajul "stop" se va transmite către telecomandă un pachet de date cu o măsurătoare a temperaturii și umidității de la senzorul digital DHT11

if(string=="stop"):

radio.stopListening()

time.sleep(0.2)

humidity, temperature = Adafruit\_DHT.read\_retry(11, DHT11\_pin)

string1="T:"+str(temperature)+"C H:"+str(humidity)+"%"

radio.write(string1[0:15])

print(string1)

#Dacă mesajul recepționat nu este "stop" atunci acesta este prelucrat

radio.startListening()

else:

#Se împarte mesajul într-un vector cu ajutorul delimitatorului virgulă.

x = string.split(",")

#Dacă primul element al vectorului este "car", atunci pachetul recepționat reprezintă coordonatele de deplasare pentru robotul mobil

if(x[0]=="car"):

# Servo – motoarele brațului mechanic sunt puse în repaus

pi.set\_servo\_pulsewidth(27,1500)

pi.set\_servo\_pulsewidth(22,1500)

pi.set\_servo\_pulsewidth(5,1000)

s1 = x[1]

s2= x[2]

s3 = x[3]

s4 = x[4]

s1 = s1[1:]

s2 = s2[1:]

s3 = s3[1:]

s4 = s4[1:]

grad1 = int(s1)

grad2 = int(s2)

grad3 = int(s3)

grad4 = int(s4)

if(grad1>0):

#Deplasare motoare DC în față

if(grad3 == 0 and grad4 == 0):

p.start(grad1)

q.start(grad1)

GPIO.output(in1,GPIO.HIGH)

GPIO.output(in3,GPIO.HIGH)

GPIO.output(in2,GPIO.LOW)

GPIO.output(in4,GPIO.LOW)

#Deplasare motoare DC în față și la dreapta

elif(grad3 > 0):

p.start(grad3)

GPIO.output(in1,GPIO.HIGH)

GPIO.output(in3,GPIO.LOW)

GPIO.output(in2,GPIO.LOW)

GPIO.output(in4,GPIO.LOW)

#Deplasare motoare DC în față și la stânga

elif(grad4>0):

q.start(grad4)

GPIO.output(in1,GPIO.LOW)

GPIO.output(in3,GPIO.HIGH)

GPIO.output(in2,GPIO.LOW)

GPIO.output(in4,GPIO.LOW)

elif(grad2>0):

#Deplasare motoare DC în spate

if(grad3 == 0 and grad4 == 0):

p.start(grad2)

q.start(grad2)

GPIO.output(in2,GPIO.HIGH)

GPIO.output(in4,GPIO.HIGH)

GPIO.output(in1,GPIO.LOW)

GPIO.output(in3,GPIO.LOW)

#Deplasare motoare DC în spate și la dreapta

elif(grad3>0):

p.start(grad3)

GPIO.output(in2,GPIO.HIGH)

GPIO.output(in4,GPIO.LOW)

GPIO.output(in1,GPIO.LOW)

GPIO.output(in3,GPIO.LOW)

#Deplasare motoare DC în spate și la stânga

elif(grad4>0):

q.start(grad4)

GPIO.output(in2,GPIO.LOW)

GPIO.output(in4,GPIO.HIGH)

GPIO.output(in1,GPIO.LOW)

GPIO.output(in3,GPIO.LOW)

else:

#Motoarele DC sunt în repaus.

GPIO.output(in1,GPIO.LOW)

GPIO.output(in3,GPIO.LOW)

GPIO.output(in2,GPIO.LOW) GPIO.output(in4,GPIO.LOW)

p.stop()

q.stop()

radio.startListening()

#Dacă primul element al vectorului este "arm", atunci pachetul recepționat reprezintă coordonatele de deplasare pentru brațul mecanic

if(x[0]=="arm"):

#Motoarele DC sunt

GPIO.output(in1,GPIO.LOW)

GPIO.output(in3,GPIO.LOW)

GPIO.output(in2,GPIO.LOW)

GPIO.output(in4,GPIO.LOW)

a1=x[1]

a2=x[2]

a3=x[3]

a1=a1[1:]

a2=a2[1:]

a3=a3[1:]

print(a3)

#Deplasarea brațului mecanic pe orizontală

pi.set\_servo\_pulsewidth(27,int(a1))

#Deplasarea brațului mecanic pe verticală

pi.set\_servo\_pulsewidth(22,int(a2))

#Servo – motorul care acționează cleștele brațului mecanic

pi.set\_servo\_pulsewidth(5,int(a3))

#pi.set\_servo\_pulsewidth(27,0)

#pi.set\_servo\_pulsewidth(22,0)

#pi.set\_servo\_pulsewidth(5,0)

time.sleep(0.2)

radio.startListening()

# 

# 