2. Structura generală și principiile generale de funcționare ale aplicației informatice

2.1 Componentele hardware utilizate în aplicația informatică

Aplicația informatică și algoritmul de deplasare pentru robotul cu mișcare prin târâre pot fi înțelese mai bine după cunosterea completă a componentelor hardware pe care aplicația informatică le utilizează.

Compomnenta hardware cea mai importantă pentru realizarea aplicației informatice este microcontrolerul, utilizat pentru controlul tututor proceselor din aplicație. Am folosit placa de dezvoltare Arduino Uno R3 Mini, care are prezintă microcontrolerul ATmega328P, fiind cea mai bună opțiune pentru aplicația aceasta. Prezintă suficienți pini pentru a putea acționa 9 servomotoate, un senzor utrasonic și viitoare componente care vor fi adăugate pentru îmbunătățirea robotului, precum un modul bluetooth HC-05/HC-06 și o cameră pentru înregistrare video OV2640. De asemenea prezintă suficientă memorie pentru rularea aplicației. Comunicarea dintre placă de dezvoltare și computerul personal de pe care se va inițializa aplicația va fi USB-Serial, folosindu-ne de chip-ul USB CH304.

Fig. 1.1.1 Placa de dezvoltare Arduino Uno R3

Tabelul 2.1.1 Date tehnice Arduino Uno R3

Placa de dezvoltare	Arduino Uno R3 Mini	
Microcontroler	ATmega328P	
Conexiune USB	Micro USB-B	
Pini	Pini led incorporați	13
	Pini digitali I/O	14
	Pini analog intrare	6
	Pini PWM	6
Comunicare	UART	Da
	I2C	Da
	SPI	Da
Putere	Tensiune de operare	5V
	Tensiune de intrare	7-12V
	Curent Continuu pe pin I/O	20mA
	Conectare alimentare	Barrel Plug
Viteza ceas	Procesor principal	ATmega328P 16 MHz
	Procesor USB-Serial	ATmega16U2 16 MHz
Memorie	ATmega328P	2KB SRAM, 32KB FLASH, 1KB EEPROM
Dimensiuni	Masa	25 g
	Latimea	53.4 mm
	Lungimea	68.6 mm

În continuare, am folosit placa de expansiune Sensor Shield V5.0, fiind o opțiune populară și des folosită în programarea plăcilor Arduino Uno, Mega 2560. Această placă de expansiune oferă o modalitate convenabilă de a conecta mai ușor senzorii, servomotoarele și alte periferice. Prezintă multe opțiuni pentru conectarea interfețelor de comunicare.

Fig. 2.1.2 Sensor Shield V5.0

Tabelul 2.1.2 Date tehnice Sensor Shield V5.0

Placa de expansiune	Sensor Shield V5.0	
Conectori terminali cu șurub	GND	1
	VCC=5V	1
Pini digitali I/O	GND	16
	VCC	16
	Digital I/O	16
Pini analog I/O	GND	5
	VCC=5V	5
	Digital I/O	5
Interfată Card SD	VCC	1
	GND	1
	D11=SPI MOSI	1
	D10=SD Card select	1
	D12=SPI MISO	1
	D13=SPI SCK	1
URF01(interfata ultrasonic)	VCC=5V	1
	A0	1
	A1	1
	GND	1
Bluetooth	VCC=5V	1
	GND	2
	D1=Serial Port RX	1
	D0=Serial Port TX	1
	3V3	1
APC220(interfata wireless)	GND	1
	VCC=5V	1
	N/C	2
	D1=Serial Port RX	1
	D0=Serial Port TX	1
I2C Interfată	SCL = A5 / I2C SCL	1
	SDA = A4 / I2C SDA	1
	'-' = Ground	1
	'+' = 5V	1
COM Interfată	TX = D1 / Serial TX	1
	RX = D0 / Serial RX	1
	'-' = Ground	1
	'+' = 5V	1
LCD Serial	VCC=5V	1

	GND	1
	Digital I/O	3
LCD Parallel	VCC=5V	1
	GND	1
	Digital I/O	12
Dimensiuni placa L*l	57*57mm	

Servomotoarele utilizate pentru deplasarea robotului sunt MG995 si sunt in numar de 9 unitati. Acestea se pot rotii aproximativ 180 de grade, 90 de grade in fiecare directie. Sunt rezistente la soc cu un design cu rulment dublu cu bile.

Tabelul 2.1.3 Date tehnice MG995

Servomotor	MG995	
Cuplu de blocare	8.5 kgf*cm(4.8V)	10kgf*cm(6V)
Viteza de operare	0.2 s/60º(4.8V)	0.16s/60 º(6V)
Tensiune de operare	4.8 – 7.2V	
Latimea benzii moarte	5 µs	
Dimensiuni	40,7 x 19,7 x 42,9 mm	
Temperatura de functionare	0 ºC – 55 ºC	

Fig. 2.1.3 Servomotor MG995

Pentru citirea distanței și evitarea obstacolelor este folosit un senzor ultrasonic HC-SR04. Acest senzor prezintă o tensiune de operare de 5V, un curent de alimentare de 15mA și o frecvență de lucru de 40Hz. Distanță maximă de citire este de 20cm cu un unghi de măsurare de 15 grade, iar semnalul Trigger este de 10µS pulsuri. Dimensiunile senzorului sunt de 40x20x15mm.

Fig. 2.1.4 HC-SR04

2.2 Prezentarea generală a algoritmului

Algoritmul de deplasare și aplicația informatică sunt realizate în LabVIEW utilizând bibloteca MakerHub Linx. Aplicația se poate rula din LabVIEW sau din aplicația web.

Deschidem VI-ul în care s-a programat algoritmul de deplasare și de citire a senzorului ultrasonic și aplicația web, odată aflat în aplicația web, putem da start aplicației unde vom fi întâlniți cu o interfața care conține 3 butoane de tip Dial pentru controlul servomotoarelor în diferite stări precum cele de deplasare înainte/înapoi, dreapta/stânga, respectiv rostogolire dreapta/stânga. De asemenea, în interfață sunt prezente 2 indicatoare, de LoopRate Hz pentru a avea o cunoaștere permanentă de rulare a parametrilor în stare corectă și respectiv, de citire distanță senzor ultrasonic, care ne arăta distanță citită în centrimetrii. Mai avem 3 elemente de control pentru alegerea canalelor servo respective în aplicație, un element de control pentru portul serial folosit și un buton de stop pentru a opri aplicația.

În această situație, putem controla robotul, prin acționarea primului dial prezentat, servomotoarele poziționate pe verticală pentru mișcarea de înainte și înapoi vor acționa așa cum au fost parametrizate în algoritm. Mișcarea de înainte și înapoi sunt aceeași funcție, dar în ordine inversă. Pentru crearea mișcării de înainte și dreapta, spre exemplu, vom acționa primul dial și

cel de al doilea, combinând cele două mișcări. Dial-ul pentru mișcarea de dreapta/stânga acționează servomotoarele poziționate pe orizontală și numai acestea. Astfel pentru crearea mișcării de înainte, înapoi, stânga, dreapta și toate combinațiile acestora, se vor folosii cele două dial-uri pentru controlul deplasării robotului.

În dreapta aplicației avem al 3-lea dial, pentru rostogolirea robotului, această funcție este creată în cazul blocării robotului pe teren. Această funcție de deplasare este independentă de cele două stări prezentate, aceasta este activată după ce cele două stări anterioare sunt oprite. Funcția de rostogoloire va acționa toate servomotoarele aflate pe verticală și orizontală, astfel creeand o mișcare de răsucire/rostogolire care poate ajuta robotul a se debloca și a se repoziționa corect. Fiecare stare controlată de cele 3 dial-uri prezinta o pozitie neutră, care setează servomotoarele într-o pozitie de 90 de grade, astfel îndreptând corpul robotului. Această funcție a fost creată pentru evitarea erorilor și pentru poziționarea corectă. Acționarea a mai multor stări incompatibile crează erori, servomotoarele se blochează. Așa cum este și logic, pe un motor nu poți acționa stânga și dreapta în același timp, astfel starea de neutru are rolul de a elimina acest tip de erori. Odată satisfăcuți cu deplasarea robotului, putem opri aplicația folosind butonul stop.

În tot acest timp, senzorul ultrasonic citește distanța în centrimetrii, așa cum putem vedea în indicatorul prezent în aplicația web. Această citire ne oferă mai multe informații legate de mediul în care se deplasează robotul și, de asemenea, ne poate ajuta la evitarea obstacolelor, având o mai bună cunoaștere a distanței dintre robot și obstacole.

Funcționalitatea algoritmului explicată până acum, așa cum este văzut de operatorul uman. poate fi explicată mai ușor folosindu-ne de schema logică de comandă care apare în figura următoare.

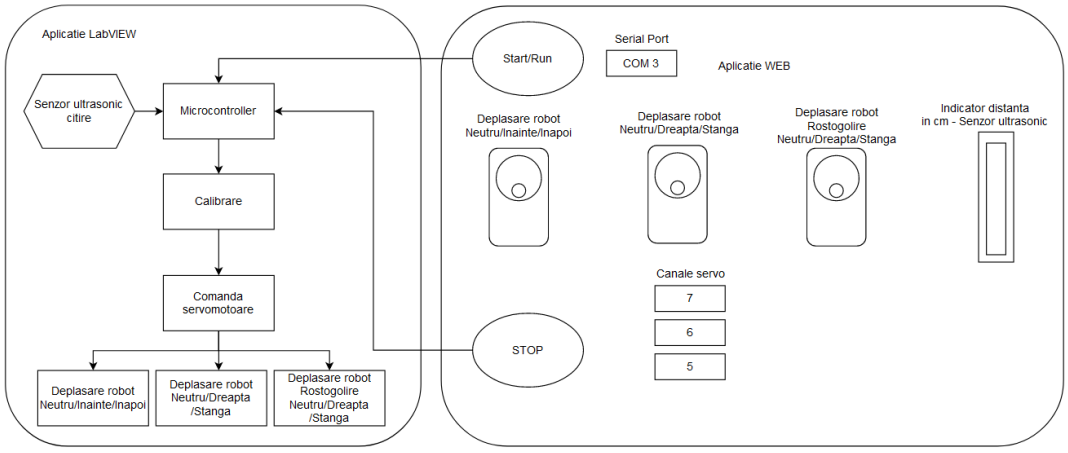

Fig. 2.2.1 Schema logică de comandă

În afara buclei while avem funcția de OpenSerial.vi, care deschide portul serial pentru a putea rula aplicația, după, această informație trece mai departe în ServoOpenOneChannel.vi, responsabil pentru deschiderea canalelor servomotoarelor, fără deschiderea și precizarea corectă a numărului canalului, servomotarele nu vor fi acționate.

Vi-ul aplicației este format într-o buclă While, pentru a putea rula aplicația continuu. De asemenea s-au folosit structuri case pentru a putea specifica stările de funcționalitate. În fiecare structure case, este specificat un timp de deplasare, dial-urile sunt conectate la structurile case respective, astfel când acționăm un dial, numai o subdiagramă a structurii case va fi executată.

Mișcarea dial-ului va selecta altă stare, astfel altă subdiagrama care va fi acționată, în care este scris algoritmul respectiv pentru funcția respectivă.

În fiecare structure case, sunt create flat sequence-uri care acestea ajută la controlul rând pe rând sau când se vrea al servomotoarelor. Servomotoarele pot fi acționate în același timp, dar pentru diferite deplasări, precum cea de înainte, înapoi, se vrea o acționare secvențială a servomotoarelor pentru a avea o mișcare mai bună.

În cadrul buclei while și a structurilor case, sunt folosite funcția de Wait (ms). Această funcție oprește rularea unei anumite acțiuni cu valoarea în milisecunde precizată de operatorul uman. Această funcție a fost integrată pentru a evita erorile legate de memorie, portul serial COM3 folosit în aplicație poate transfera o anumită cantitate de memorie, cu o anumită viteză, dacă nu se respectă aceste chestiuni, aplicația nu va rula sau va rula, dar doar pentru foarte puțin timp rezultând într-o eroare de tip timeout. Eroare de tip timeout reprezintă că durează prea mult pentru a răspunde sau, în cazul de față, este corespunzătoare faptului că prea multă memorie la o prea mare viteză este transmisă microcontrolerului. Chip-ul USB CH304 este responsabil pentru comunicarea pe serială în cazul plăcii Arduino Uno. Tot în interiorul buclei while, dar separat de structurile case se află funcția de citire senzor ultrasonic, formată din UltrasonicRead.vi.

Urmărind informația și ieșind din nou din bucla while, informația este conectată la ServoCloseOneChannel.vi, pentru închiderea canalelor servo la oprirea aplicației, și Close.vi, pentru închiderea dispozitivului de tip Linx și orice sursă de tip I/O.

Aplicația informatică, de asemenea, prezintă și un algoritm de citire pentru senzorul ultrasonic. În afară buclei while, avem funcția OpenSerial.vi, care deschide portul serial, după informația este conectată la UltrasonicRead. VI, care se afla în interiorul buclei while, pentru citirea semnalului primit. Ieșirea acestei funcții este în continuare conectată la Close.vi pentru închiderea dispozitivului de tip Linx și orice sursă de tip I/O.

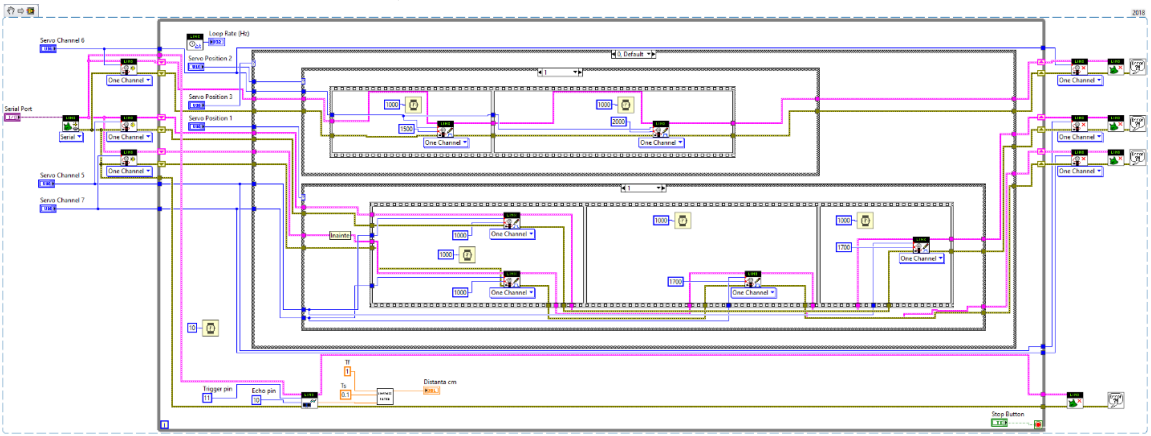

Fig. 2.2.1 VI Block Diagram – Screenshot al algoritmului pentru poziția de mers inainte-dreapta și pentru algoritmul de citire senzor ultrasonic

2.3 Componentele principale ale aplicației

Aplicația informatică este formată dintr-un sistem principal, acesta fiind format din alte 2 subsisteme. Sistemul principal este cel de Rostogolire Neutru/Dreapta/Stângă. Cele 2 subsisteme, așa cum au fost menționate în subcapitolul anterior sunt formate din cele 2 elemente de control de tip Dial, acestea fiind: Deplasare Neutru/Înainte/Înapoi, Deplasare Neutru/Dreapta/Stângă.

Sistemul de Rostogolire Neutru/Dreapta/Stânga are o importanță ierarhică mai mare decât celelalte, deoarece, ca celelalte subsisteme să funcționeze corect, trebuie setat poziția de Rostogolire Neutru, această funcție determină faptul că sistemul robotic nu se rostogolește, iar servomotoarele nu sunt acționate în nicio altă acțiune, astfel putând fii acționate de sistemele de Deplasare Neutru/Înainte/Înapoi, Deplasare Neutru/Dreapta/Stângă. În figura următoare, va fi prezentat sistemul principal când se află în poziție neutră, iar celelalte 2 subsisteme sunt tot în poziție neutră.

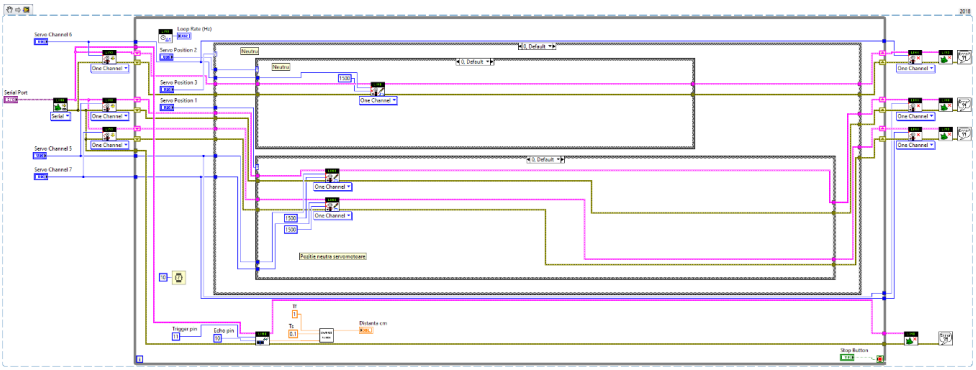

Fig. 2.3.1 Rostogolire Neutru – Neutru – Neutru

Primul subsistem este cel pentru Deplasarea Neutru/Înainte/Înapoi, prezintă 3 stări. Prima stare este cea pentru poziționarea servomotoarele pe verticală în poziție neutră, pentru îndreptarea robotului și pentru revenirea la poziția inițială. A doua stare este cea de deplasare înainte, activat, servomotoarele acționează conform algoritmului pentru a crea mișcare liniară și pentru a crea deplasare. A treia stare este cea de deplasare înapoi, creat la fel ca subsistemul pentru deplsare înainte, dar funcția fiind scrisă invers. Mai jos, va fi prezentată o imagine când sistemul principal se afla în poziție neutră, subsistemul de Deplasare Neutru/Înainte/Înapoi se afla în poziția înainte, iar subsitemul de Deplasare Neutru/Dreapta/Stânga se afla în poziție neutră.

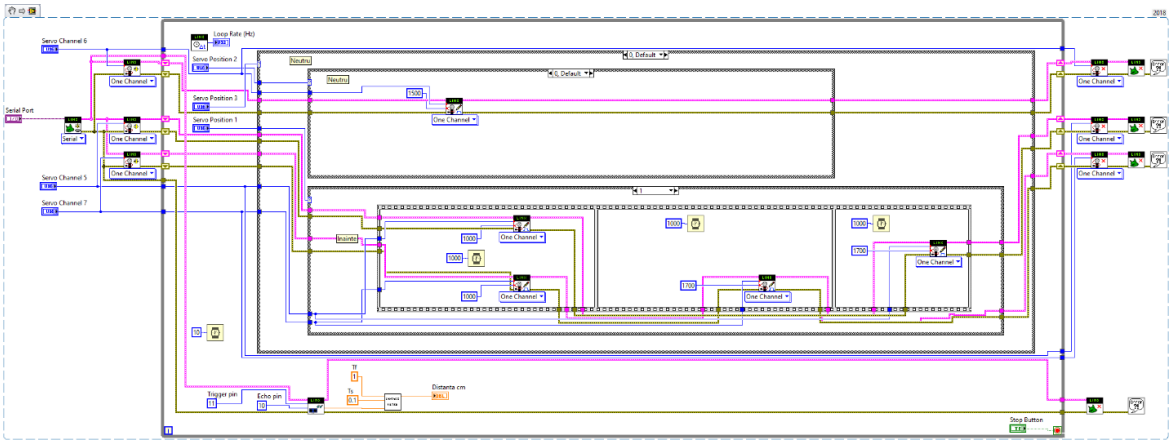

Fig. 2.3.2 Rostogolire Neutru – Inainte – Neutru

Al doilea subsistem este cel pentru Deplasarea Neutru/Dreapta/Stânga, prezintă 3 stări. Prima stare este cea pentru poziționarea servomotoarele pe orizontală în poziție neutră, pentru îndreptarea robotului și pentru revenirea la poziția inițială. A doua stare este cea de deplasare dreapta, activat, servomotoarele acționează conform algoritmului pentru a crea mișcare unghiulară și pentru a crea deplasare. A treia stare este cea de deplasare stânga, creat la fel ca subsistemul pentru deplsare dreapta, dar funcția fiind scrisă invers. Mai jos, va fi prezentată o

imagine când sistemul principal se afla în poziție neutră, subsistemul de Deplasare Neutru/Înainte/Înapoi se afla în poziția înainte, iar subsitemul de Deplasare Neutru/Dreapta/Stânga se afla în poziția dreapta.

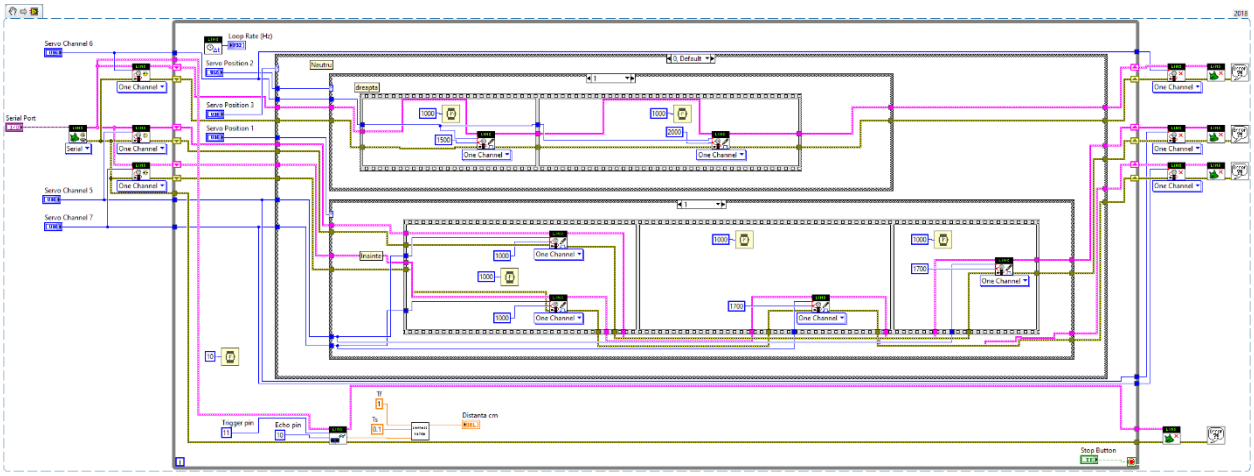

Fig. 2.3.4 Rostogolire Neutru – Inainte – Dreapta

Sistemul de Rostogolire Neutru/Dreapta/Stanga, cand este pus in pozitia de rostogolire dreapta sau rostogolire stanga, celelalte subsisteme nu pot modifica actiunile sistemului principal. Astfel, daca sistemul este setat in cele doua stari de Rostogolire Dreapta sau Stanga, nu va putea realiza combinatii de miscari cu celelalte subsisteme, fiind independent de acestea. Mai jos va fi prezentata o imagine cand sistemul principal se afla in starea de Rostogolire Dreapta.

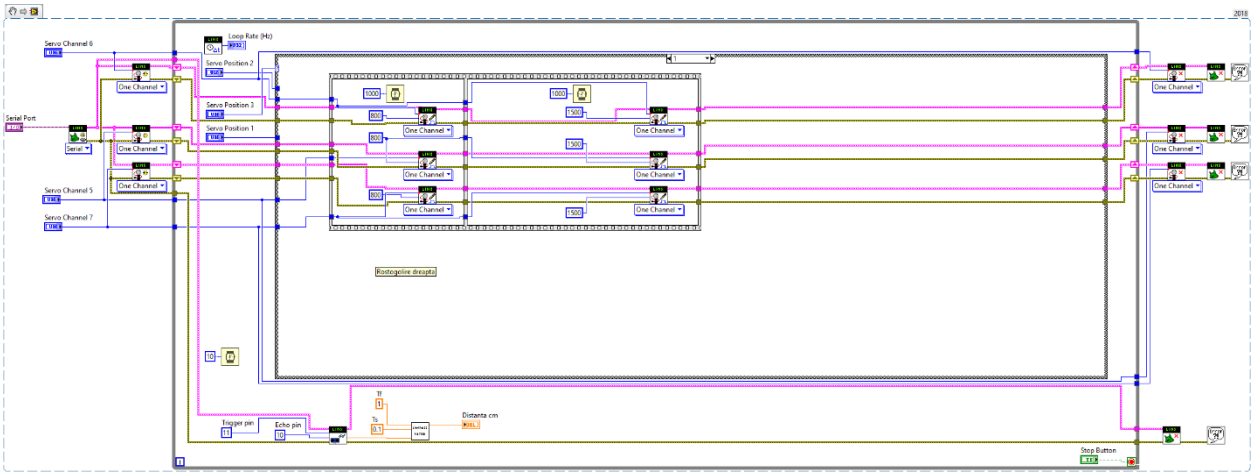

Fig. 2.3.5 Rostogolire Dreapta

Sistemele prezentate, care fac parte din algoritmul de deplasare cu mișcare prin târâre îndeplinesc funcțiile dorite. Avem subsisteme care sunt dependete de sistemul principal și un sistem independent. Informația, pe care aplicația o schimbă cu exterioul, se află în operatorul uman, care teleghidează robotul și pe baza terenului și a răspunsurilor primite din deplasarea robotului, dar cât și a semnalului ultrasonic primit de la senzor, poate lua decizii și influența mișcarea prin târâre în continuare.

Un alt sistem, care este separat de algoritmul de deplasare și care a mai fost menționat, este cel de citire senzor ultrasonic. Senzorul transmite prin aer un sunet la o freventa de 40kHz (trigger), care nu face parte din spectrul de auz al oamenilor, și dacă găsește un obiect, sunetul ricoșează înapoi la senzor. Receiverul ultrasonic preia sunetul reflectat și îl citește cu ajutorul algoritmului creat (echo). Senzorul ultrasonic schimbă informații cu exteriorul, pe care după noi le citim și le interpretăm.

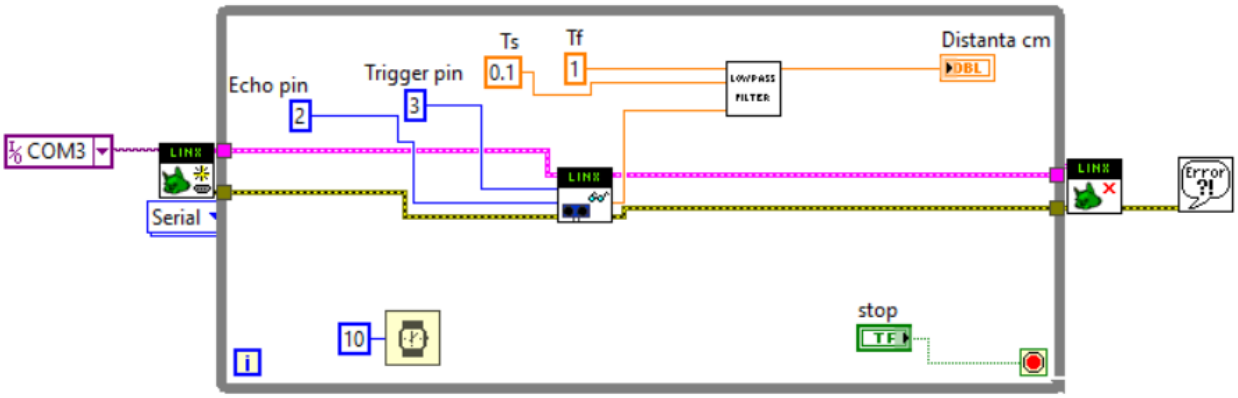

Fig. 2.3.6 Algoritm citire senzor ultrasonic

Sistemul schimbă, de asemenea, informații cu exteriorul prin aplicația Web, care permite controlul programului de tip VI.

2.4 Aplicatia Web

Aplicatia va fi publicata pe web sub forma unui fisier de tip HTML utilizand serverele Web LabView. Cu serverul web LabVIEW, nu există nimic de programat pe diagrama bloc, toate detaliile sunt gestionate de LabVIEW. VI-ul care urmează să fie afișat într-un browser web trebuie să fie deja încărcat în memorie (în LabVIEW) pentru a fi servit de serverul web LabVIEW. Serverul Web Labview poate publică VI-ul în trei moduri diferite: snapshot (static), monitor (o imagine a VI-ul care este configurat să își dea auto-refresh la un număr anume de secunde, sau embedded (o versiune controlabilă a panoului frontal, opțiunea utilizează un plug-in pentru a afișa programul în timp real și pentru a lăsa operatorul uman să controleze aplicația. Am ales să public VI-ul ca embedded, pentru a avea un control continuu, receptiv, fără probleme. Pentru realizarea aplicației Web, voi prezența următorii pași pe care i-am luat în construcția și publicarea VI-ului.

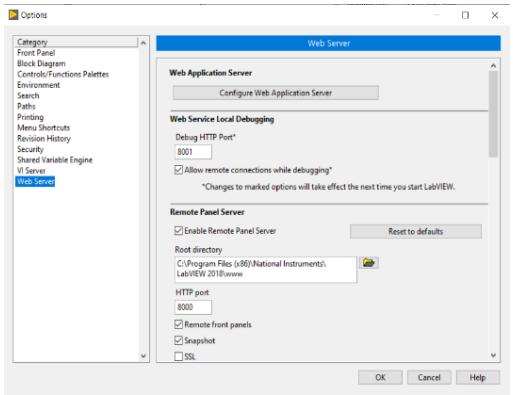

Fig. 2.4.1 Tools – Options – Web Server

Am deschis VI-ul, aplicația nu poate fi rulată în Web dacă nu este deschisă și aplicația principală. Din meniul Tools, accesăm opțiunile, bifăm Enable Remote Panel Server pentru a putea rula aplicația în Web.După vom folosi Web Publishing Tool, din meniul Tools. Vom alege opțiunea Embedded și vom bifa Request control și Enable IMAQ support.

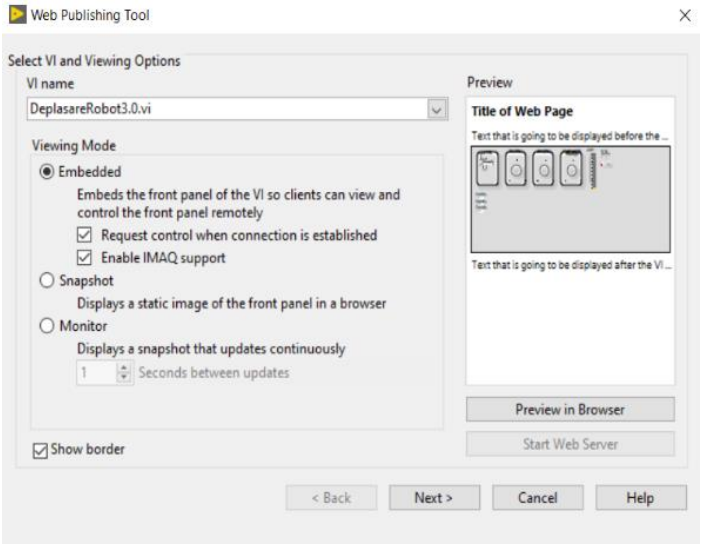

Fig. 2.4.2 Tools – Web Publishing Tool

Următoarea fereastră va fii pentru editarea titlului paginii Web, a headerul-ui și a footer-ului. Ultima fereastă va fi pentru a selecta unde pe disk să salvezi fișierul HTML creat, de luat în vedere faptul că trebuie verificat dacă portul și IP-ul sunt corecte. Vom utiliza port 8000, care este des utilizat ca port HTTP și pentru administrare Web. URL-ul va fi format din IP-ul PC-ului și numele fișierului HTML. Folosim IP-ul PC-ului pe care am salvat VI-ul, pentru că îl vom folosi pentru crearea server-ului web. URL aplicație Web: http://192.168.0.109:8000/DeplasareRobot3.0.html.

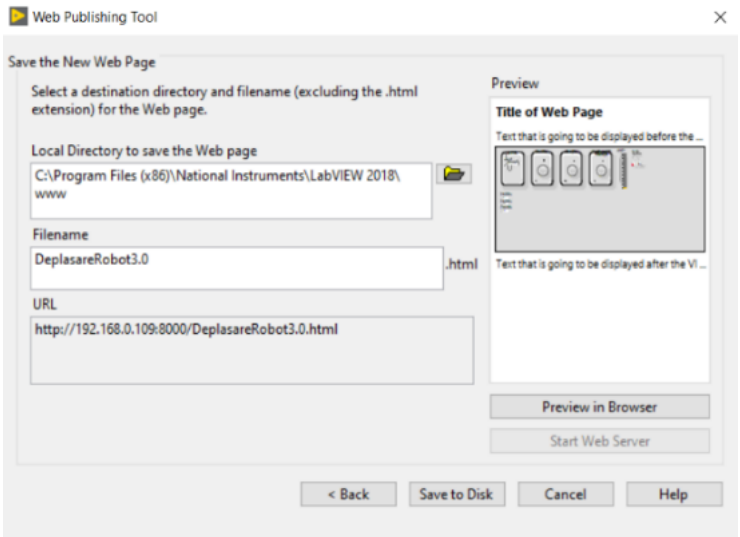

Fig. 2.4.3 Web Publishing Tool – URL

În figura următoare, avem o prezentare a interfeței aplicației Web, partea de editare și formatare a paginii a fost realizată utilizând cunostințele dobândite la materia de studiu “Interfețe și Servicii Web”. Am folosit structuri de tip html, css și javascript.

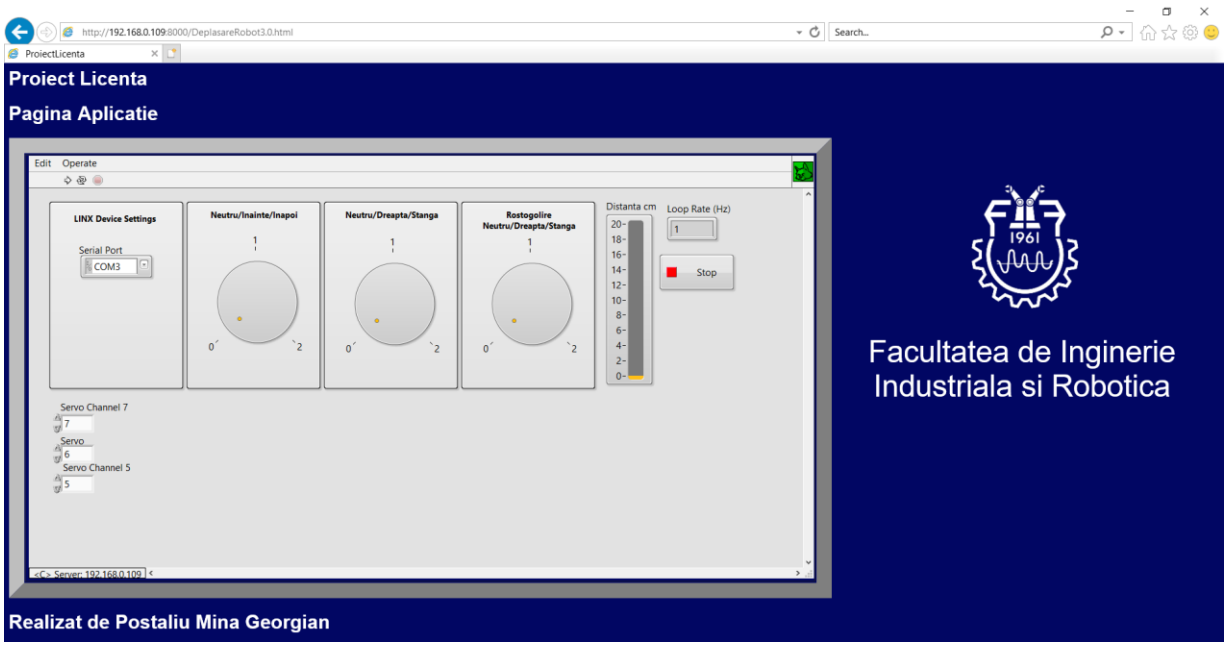

Fig. 2.4.4. Interfața aplicației Web

2.5 Justificare alegerii uneltelor software pentru realizarea algoritmului

Am ales LabView și bibloteca MakerHub Linx pentru că oferă cea mai bună interacțiune cu platforme încorporate precum Arduino. Acestea prezintă VI-uri speciale, deja construite pentru controlul și citirea senzorilor și a plăcilor de dezvoltare. Oferă o interfața grafică, astfel putând vizualiza programul în timp real, mult mai accesibil și rapid, decât folosind IDE-uri și programare clasică. De asemenea, se află într-un singur pachet, compact, simplificând procesul de creare al algoritmiilor. LabVIEW oferă multe opțiuni pentru crearea serviciilor Web, precum partea de hosting server, creare fișiere HTML.

SolidWorks 3D CAD a fost ales pentru proiectarea asistată pe calculator și modelarea 3D a solidelor, prezintă multe opțiuni în privința modelării solidelor, analiza și simulările utilizând principiile metodei elementului finit, studiu topologic, proprietăți material, randări 3D foto-realistice, animații

Z-suite a fost ales ca program de slicing pentru parametrizarea viitoarelor repere (module) ce vor urmă a fii fabricate aditiv utilizând imprimantele Zortrax M300+. Acești parametrii fiind spre exemplu: viteza de printare, volumul de material extrudat de duză, temperatura de topire pentru material, crearea de suporți pentru zonele care necesită, creare cod G, exportarea fișierului pentru utilizarea acestuia pe imprimantă.

3. Proiectarea detaliată a aplicației informatice

3.1 Prezentarea detaliată a algoritmului

În acest capitol, vom prezența în detaliu algoritmii creați și utilizați pentru aplicația informatică. Acest lucru, îl vom face într-o ordine logică, așa cum apare în Block Diagram, urmărind sensul informației transmise, tipul acesteia, rolul VI-urilor și a subVI-urilor folosite și cum este tradus acest algoritm către operatorul uman și alte elemente exterioare.

Deschidem VI-ul principal, care conține algoritmii de deplasare și de citire senzor ultrasonic. Deschidem meniul Tools> MakerHub> LINX> LinxFirmware Wizard, se va deschide o fereastră pop-up. Alegem placă de dezvoltare pe care o utilizăm, în cazul nostru Arduino Uno și tipul de upload al firmware-ului. Vom alege tipul de port pe care îl vom folosi pentru descărcarea firmware-ului. În continuare vom apăsă next pentru a descărca pe placa de dezvoltare software-ul Linx. Este necesar să facem acești pași preliminări pentru a putea folosi funcțiile de tip Linx din MakerHub pentru LabVIEW împreună cu placă noastră de dezvoltare.

Fig. 3.1.1. Pasi preliminari descarcare firmware Linx

Revenind la Block Diagram, pentru coumunicarea dintre programul VI realizat pe PC și microcontroler am utilizat OpenSerial.vi, care deschide comunicarea pe serială cu Arduino Uno folosindu-ne de chip-ul USB CH304 care se află pe placă. Comunicarea se realizează prin COM3 (USB), astfel am conectat la VI un element de control pentru alegerea portului. O altă dată de intrare pentru OpenSerial.vi este Baud Rațe Override, care este folosit pentru dispozitive hardware care au comunicarea pe serială, reprezintă rata la care informația este transmisă, dacă nu punem un elemnt de control, va fi luat de program cu o rată default de 9600, ceea ce înseamnă o rată de transfer de 9600 biți pe secundă, care sunt suficienți pentru această aplicația, deci nu vom modifica nimic. Datele de ieșire pentru OpenSerial.vi va fi Linx Resources, acest tip de informație String reprezintă un strat abstract, care ne ajută să avem o singură interfața pentru toate dispozitivele și senzorii utilizați, și Error Out, care are rolul de semnalarea unei erori în cazul rulării programului, acesta va fi conectat în toate VI-urile care urmează într-o ordine logică, mai departe în VI va fi conectat SimpleErrorHandler.vi care va arăta o căsuță de tip dialog când o eroare se întâmplă. Informația Linx Resources data de OpenSerial.vi va fi conectată, în continuare, la trei ServoOpenOneChannel.vi, acestea au rolul de a deschide canalul servo specific la care este conectat fiecare servomotor pe placa de dezvoltare. ServoOpenOneChannel.vi mai prezintă două date de intrare, unul este pentru specificarea scanalului servo, vom folosi un element de control pe care îl vom conecta, și o dată de Error In,

menționat anterior, pentru semnalrea unei erori în cadrul VI-ului, Data de ieșire va fi Linx Resources care va fi conectată la ServoSetPulseWidthOneChannel.vi.

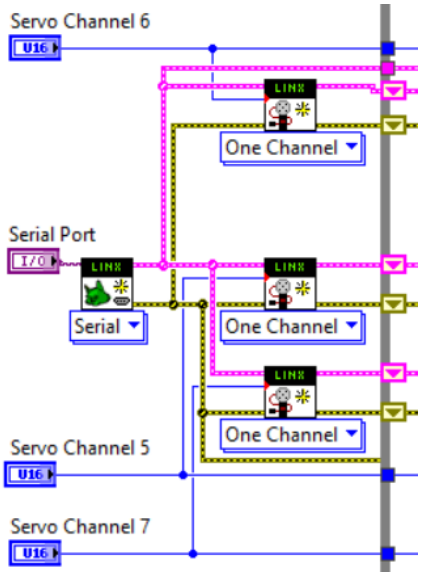

Fig. 3.1.2. OpenSerial.vi > ServoOpenOneChannel.vi

A fi luat în considerare faptul că sunt doar trei ServoOpenOneChannel.vi, chiar dacă algoritmul este realizat pentru nouă servomotoare care ar avea nevoie de nouă servo canale. Acționarea servomotoarelor este împărțită în trei tipuri de mișcări în algoritmul de deplasare: mișcare de tragere, mișcare de împingere, mișcare unghiulară; acestea formează mișcarea de târâre. Astfel la fiecare ServoOpenOneChannel, elementul de control pentru canalul servo îl vom transforma într-o matrice, fiecare matrice va avea inclus canalele servo respective tipului de mișcare. Putem controla nouă servomotoare utilizând doar trei ServoOpenOneChannel.vi, dar făcând acest lucru, proprietatiile VI-ului se vor schimbă puțin, în mod automat, și se va transforma în ServoOpenNChannels.vi, destinat pentru o matrice de servomotoare.

Datele de ieșire ale ServoOpenOneChannel.vi vor fi conectate la buclă while utilizând shift register. Acesta este folosit pentru a stoca date în timpul rulării buclei, te lasă să accesezi informația și să returnezoi valoarea iterației anterioare, de asemenea este folosit pentru evitarea erorilor și pentru o tranziție mai bună a informațiilor, fiind comparabil cu o variabilă locală.

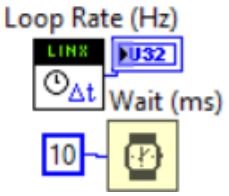

Fig. 3.1.3. Loop Rate și Wait

În interiorul buclei While, am adăugat LoopFrequency.vi pentru a calcula rata fiecărei bucle, în cazul unei erori va întoarce valori incorecte, în caz optim de funcționare va afișa o valoare corectă în hertz. De asemenea, am adăugat și funcția de Wait (ms) pentru 10ms, ca să nu existe erori de timeout legate de memorie când aplicația rulează.

În continuare, folosim o buclă whille pentru rularea continuă a aplicației. În interiorul buclei am introdus un Case Structure, care conține mai multe subdiagrame care sunt executate într-o ordine prestabilită conform elementelor de control numeric de tip Dial. Elementele de control numeric sunt conectate la selectorii case, astfel, începând de la o valoare 0, default până la o valoare 2, putem selecta subdiagrama/subcase-urile pe care vrem să le acționăm în deplasarea robotului.

În interiorul Structururilor Case avem SetPulseWidthOneChannel.vi, funcție care este responsabilă de lățimea pulsui pentru canalul servo specific, astfel putem alege o valoare fixă de pulsații. 1500 µs esre corespondent unei poziții a servomotoarelor la 90 de grade, această valoare este folosită des pentru poziționarea neutră a servomotoarelor în sistemele și subsistemele prezentate. Datelel de intrare sunt: ServoChannel, setat din afara buclei while; Linx Resources pentru a face conexiunea informației cu OpenSerial.vi, care acesta face comunicarea dintre aplicație și placa de dezvoltare; și PulseWidth (µs) care este setat în interiorul Structurii Case respectiv. Datele de ieșire ale SetPulseWidthOneChannel.vi sunt: Linx Resources și Error Out.

Dial-ul Servo Position 3, este responsabil pentru deplasarea de Rostogolire Neutru/Dreapta/Stânga. Acest element de control determină starea de deplasare a sistemului principal. Dacă setăm pe 0, vom vedea un Strucutre Case în care se află alte două Structuri Case. Poziția 0 a sistemului principal determină faptul că robotul nu realizează o mișcare de rostogolire, astfel celelalte două Structuri Case responsabile pentru Deplasarea Neutru/Înainte/Înapoi și

Deplasarea Neutru/Dreapta/Stânga pot fi acționate. Fiecare din cele două Structuri Case ale celor două subsisteme au un element numeric de control, începând de la o valoare 0, default până la o valoare 2, putem selecta subdiagrama/subcase-urile pe care vrem să le acționăm în deplasarea robotului.

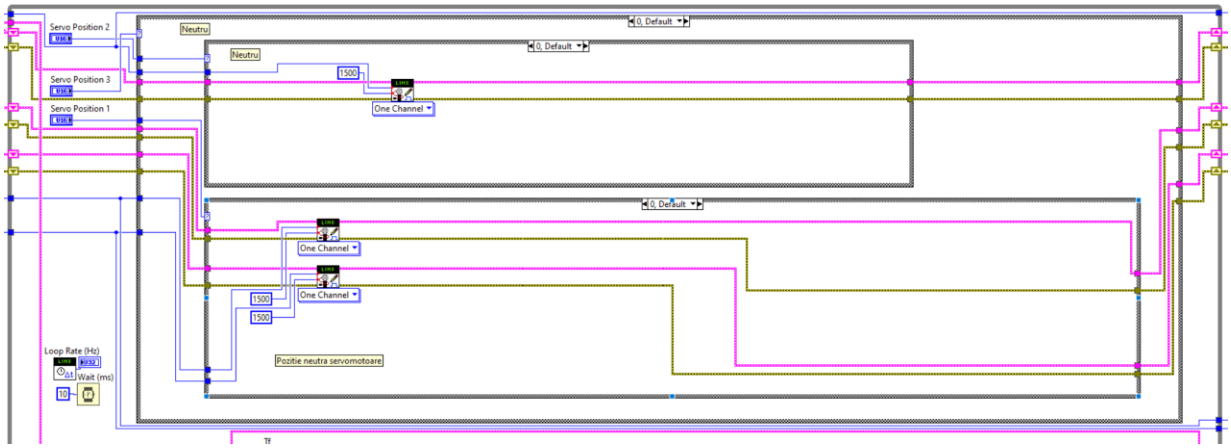

Fig. 3.1.4. Poziție Dial Rostogolire Neutru/Dreapta/Stanga 3

În cazul Deplasării Neutru/Înainte/Înapoi, poziția de 0 a Dial-ului setează servomotoarele atribuite funcții în poziție neutră, la 90 de grade cu o valoare constanța setată de 1500 µs. Poziția 1, setează servomotoarele atribuite funcții în poziția de deplasare Înainte.

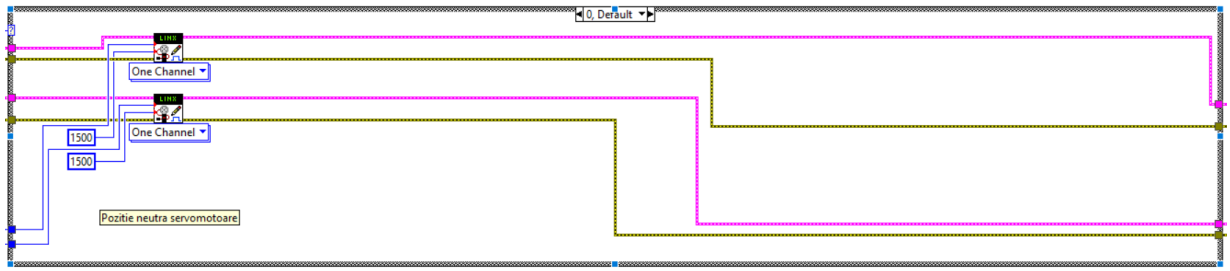

Fig. 3.1.5. Deplasare Neutru/Inainte/Inapoi , poziția 0

Poziția 1 este formată dintr-un Sequence Flat, care este format din mai multe subdiagrame care inițiază secvențial acționarea servomotoarelor în diferite poziții.

Prima secvență, prezintă două SetPulseWidthOneChannel.vi responsabile de punerea servomotoarelor la o pulsație de 1000 µs, care reprezintă o poziție a servomotoarelor la 45 de grade. Această mișcare face robotul să se incovoiasca asemănător unei rame.

A două secvență pune servomotoarele care au realizat mișcare de tragere să se retragă la o pulsație de 1700 µs, care reprezintă o poziție a servomotoarelor la 108 de grade. Servomotoarele care realizează mișcarea de tragere, au deplasat robotul înainte, acum sunt poziționate să nu mai aibă contact cu solul ca servomotoarele care realizează mișcarea de împingere să realizeaze această acțiune și, ca prin urmare, robotul se deplaseze și mai mult înainte.

A treia secvență pune servomotoarele care au fost puse la o pulsație de 1000 µs, să realizeze mișcarea de împingere și să se retragă la o pulsație de 1700 µs, care reprezintă o poziție a servomotoarelor la 108 de grade.

În fiecare secvență a structurii Sequence Flat, a fost adăugat o funcție de Wait (ms) de 1000ms, pentru ca să existe o pauză de 1 secundă între fiecare secvență, eliminând erorile de tip timeout și pentru a avea o deplasare mai controlată a robotului.

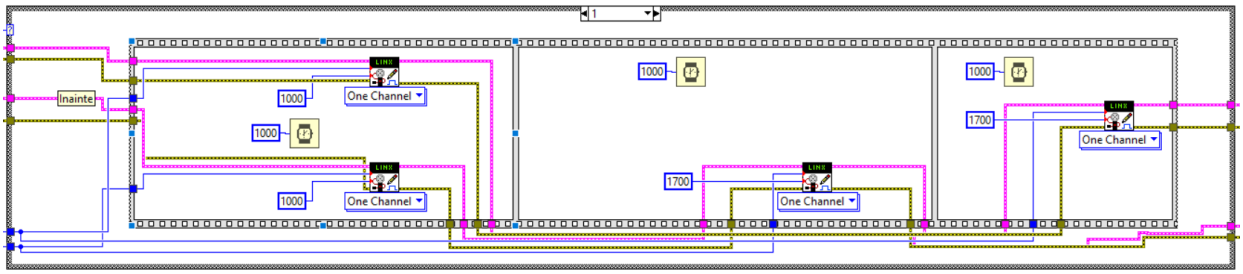

Fig. 3.1.6. Deplasare Neutru/Inainte/Inapoi , pozitia 1

Datorită faptului că structura de tip Sequence Flat se afla în buclă while, secvențele se vor repeta, astfel având o deplasare continuă. Prima secvență va rula, urmată de o pauză de 1 secundă, a doua secvență, o pauză de 1 secundă, a treia secvență, o pauză de 1 secundă, ca în final să revenim la prima secvență, de unde se va repeta iar strucutura până când operatorul uman oprește acțiunea.

Poziția 2, corespunzătoare de realizarea deplasării înapoi, este formată dintr-un Sequence Flat, care este format din mai multe subdiagrame care inițiază secvențial acționarea servomotoarelor în diferite poziții. Structure Case-ul corespunzător deplasării înapoi va avea același algoritm că deplasarea înainte, doar că funcția servomotoarelor care trag va fi inversată cu a servomotoarelor care împing.

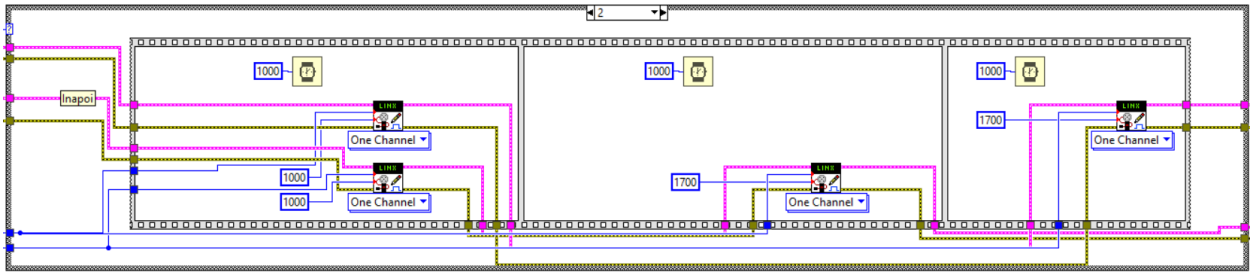

Fig. 3.1.7. Deplasare Neutru/Inainte/Inapoi , poziția 2

În cazul Deplasării Neutru/Dreapta/Stângă, poziția de 0 a Dial-ului setează servomotoarele atribuite funcții în poziție neutră, la 90 de grade cu o valoare constanța setată de 1500 µs. Poziția 1, setează servomotoarele atribuite funcții în poziția de mișcare unghiulară pentru a realiza întoarcerea la dreapta.

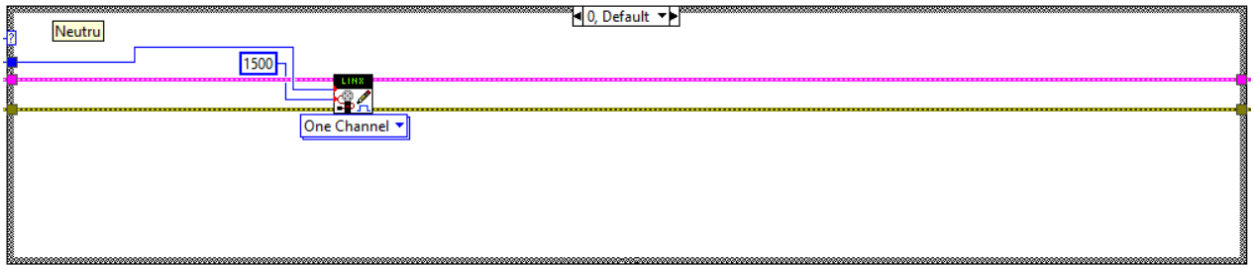

Fig. 3.1.8. Deplasare Neutru/Dreapta/Stanga, poziția 0

Poziția 1 este formată dintr-un Sequence Flat, care este format din mai multe subdiagrame care inițiază secvențial acționarea servomotoarelor în diferite poziții.

Prima secvență, prezintă un SetPulseWidthOneChannel.vi responsabil de punerea servomotoarelor la o pulsație de 1500 µs, care reprezintă o poziție a servomotoarelor la 90 de grade, poziție neutră.

A două secvență pune servomotoarele la o pulsație de 2000 µs, care reprezintă o poziție a servomotoarelor la 135 de grade.

În fiecare secvență a structurii Sequence Flat, a fost adăugat o funcție de Wait (ms) de 1000ms, pentru ca să existe o pauză de 1 secundă între fiecare secvență, eliminând erorile de tip timeout și pentru a avea o deplasare mai controlată a robotului.

Datorită faptului că structura de tip Sequence Flat se afla în buclă while, secvențele se vor repeta, astfel având o deplasare continuua. Prima secvență va rula, urmată de o pauză de 1 secundă, a două secvență, o pauză de 1 secundă, ca în final să revenim la prima secvență, de unde se va repetă iar strucutura până când operatorul uman oprește acțiunea.

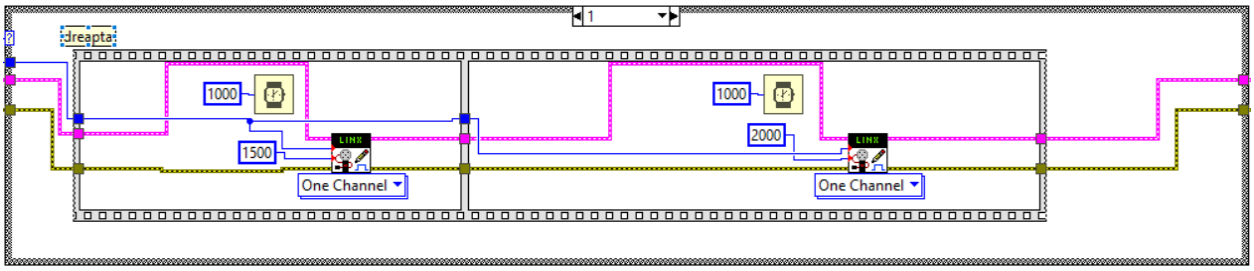

Fig. 3.1.9. Deplasare Neutru/Dreapta/Stanga, pozitia 1

Poziția 2, corespunzătoare de realizarea deplasării de întoarcere la stânga, este formată dintr-un Sequence Flat, care este format din mai multe subdiagrame care inițiază secvențial acționarea servomotoarelor în diferite poziții. Structura Case corespunzător deplasării de întoarcere la stânga va avea același algoritm că deplasarea înainte, doar că funcția va începe prima secvență cu o pulsație de 1500 µs, că după să fie urmată de o a două secvență cu o pulsație de 1000 µs.

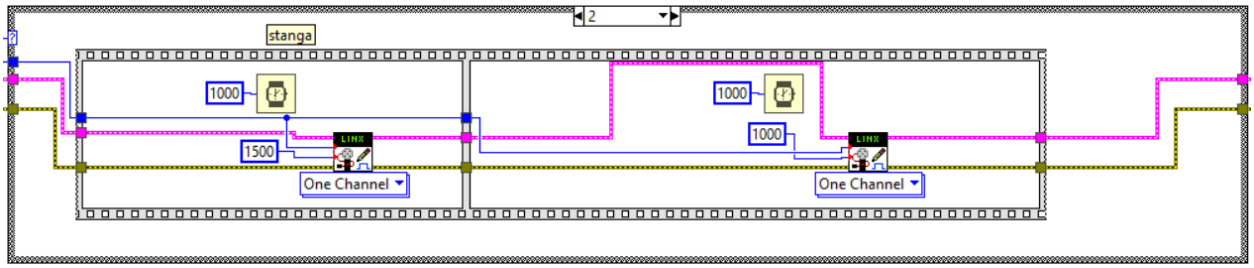

Fig. 3.1.10. Deplasare Neutru/Dreapta/Stanga, poziția 2

În cazul deplasării de Rostogolire Neutru/Dreapta/Stângă, Structure Case-ul corespunzător poziției 1 prezintă o structura Sequence Flat, formată din două secvențe.

Prima secvență prezintă 3 SetPulseWidthOneChannel.vi, care vor seta servomotoarele la o pulsație de 800 µs, care reprezintă o poziție a servomotoarelor la 27 de grade.

A două secvență prezintă 3 SetPulseWidthOneChannel.vi, care vor retrage servomotoarele la o pulsație de 1500 µs, care reprezintă o poziție a servomotoarelor la 90 de grade, poziție neutră.

În fiecare secvență a structurii Sequence Flat, a fost adăugat o funcție de Wait (ms) de 1000ms, pentru ca să existe o pauză de 1 secundă între fiecare secvență, eliminând erorile de tip timeout și pentru a avea o deplasare mai controlată a robotului.

Datorită faptului că structura de tip Sequence Flat se afla în buclă while, secvențele se vor repeta, astfel având o deplasare continuua. Prima secvență va rula, urmată de o pauză de 1 secundă, a două secvență, o pauză de 1 secundă, ca în final să revenim la prima secvență, de unde se va repetă iar strucutura până când operatorul uman oprește acțiunea.

Deplasarea de Rostogolire Dreapta are ca scop punerea în poziție corectă a robotului prin mișcare de răsucire, în caz de blocare a acestuia pe partea dreaptă.

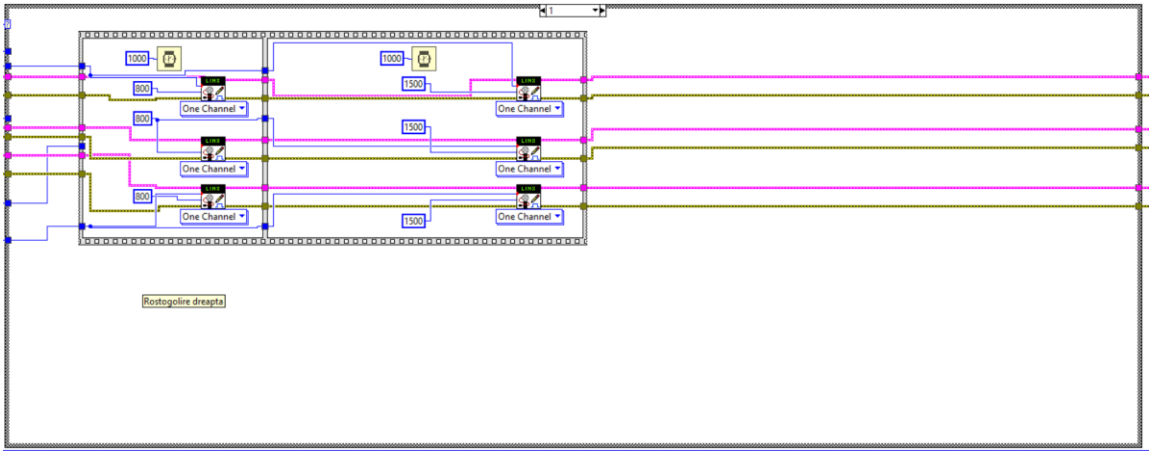

Fig. 3.1.11. Deplasare Rostogolire Neutru/Dreapta/Stanga, pozitia 1

În cazul deplasării de Rostogolire Neutru/Dreapta/Stângă, Structure Case-ul corespunzător poziției 2 prezintă o structura Sequence Flat, formată din două secvențe.

Prima secvență prezintă 3 SetPulseWidthOneChannel.vi, care vor seta servomotoarele responsabile pentru mișcarea de împingere și tragere a robotului la o pulsație de 800 µs, care reprezintă o poziție a servomotoarelor la 27 de grade, iar servomotoarele responsabile de mișcarea de întoarcere la stânga și la dreapta vor fi setate la o pulsație de 2100 µs, care reprezintă o poziție a servomotoarelor la 145 de grade.

A două secvență prezintă 3 SetPulseWidthOneChannel.vi, care vor retrage servomotoarele la o pulsație de 1500 µs, care reprezintă o poziție a servomotoarelor la 90 de grade, poziție neutră.

În fiecare secvență a structurii Sequence Flat, a fost adăugat o funcție de Wait (ms) de 1000ms, pentru ca să existe o pauză de 1 secundă între fiecare secvență, eliminând erorile de tip timeout și pentru a avea o deplasare mai controlată a robotului.

Datorită faptului că structura de tip Sequence Flat se afla în buclă while, secvențele se vor repeta, astfel având o deplasare continuă. Prima secvență va rula, urmată de o pauză de 1 secundă, a două secvență, o pauză de 1 secundă, ca în final să revenim la prima secvență, de unde se va repetă iar strucutura până când operatorul uman oprește acțiunea.

Deplasarea de Rostogolire Stânga are ca scop punerea în poziție corectă a robotului prin mișcare de răsucire, în caz de blocare a acestuia pe partea stângă.

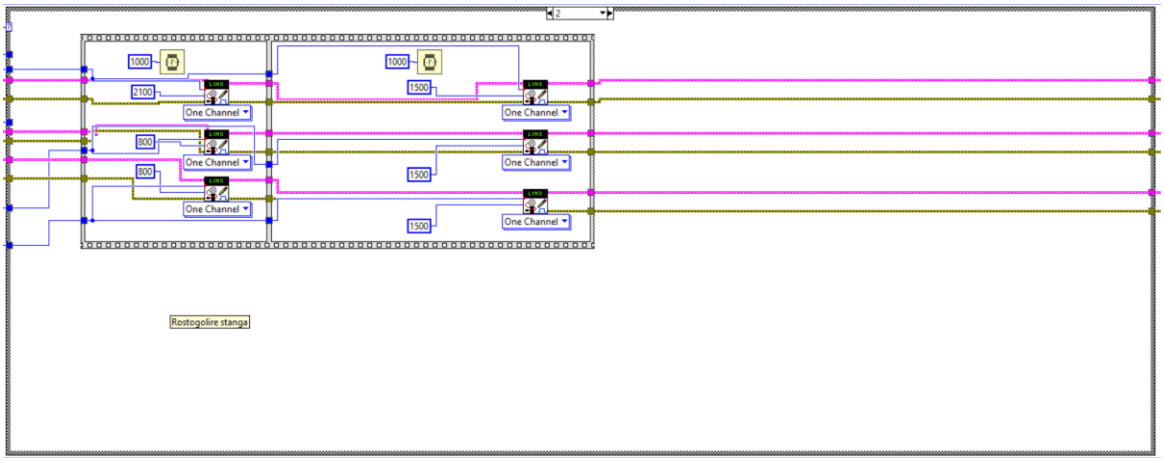

Fig. 3.1.12. Deplasare Rostogolire Neutru/Dreapta/Stanga, pozitia 2

Datele de iesire ale structurii Case a sistemului principal vor fii Linx Resource si Error Out, acestea vor devenii date de intrare pentru ServoCloseOneChannel.vi, care se afla in afara buclei while. ServoCloseOneChannel.vi are ca rol inchiderea canalelor servo, data de iesire a VI-ului va fii in continuare conectata la Close.vi, care are rol de a inchide comunicarea dintre program si placa de dezvoltare pe portul serial cand operatorul uman isi doreste acest lucru.

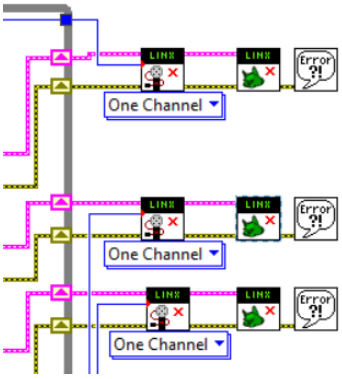

Fig. 3.1.13 ServoCloseOneChannel.vi > Close.vi

În afara structurii case principle, în buclă while, se afla algoritmul pentru citirea senzorului ultrasonic. Are că date de intrare Linx Resource creat din OpenSerial.vi și două elemente de control numeric, care vor să reprezintă numărul pinului respectiv de pe placa de dezvoltare pentru Trigger și Echo respectiv. Ca date de ieșire avem doi indicatori pentru citirea în inch și centrimetrii. Pentru a avea o citire corectă a semnalului, fără zgomot, vom folosii un filtru LowPass. Filtrul LowPass este un filtru care transmite semnalul la o frecvența mai mică decât cea inițiată de dispozitiv și care este atenuat cu o frecvența mai mare de tăiere. Astfel, avem parte de o citire mai bună a semnalului. În final, ne vom folosi de un Close.vi pentru a închide comunicarea dintre program și placa de dezvoltare atunci când operatoul uman dorește acest lucuru.

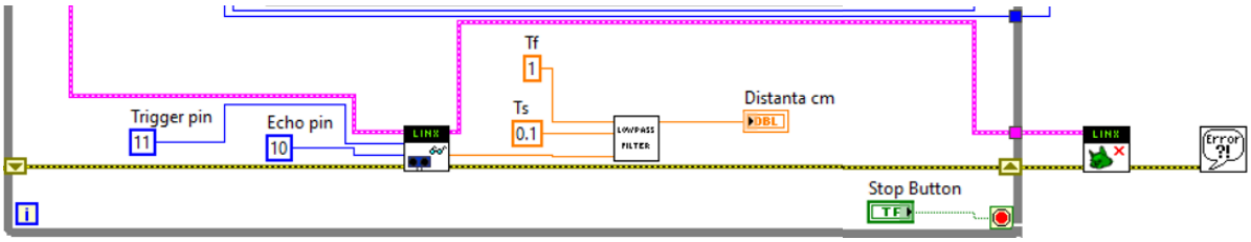

Fig. 3.1.14 Algoritm de citire senzor ultrasonic

4. Realizarea și testarea aplicației informatice

4.1 Asamblarea componentelor aplicației informatice. Verifiări inițiale. Rezultate obținute

Aplicația informatică prezintă mai multe componente care asamblate în împreună o creează. Prima componentă inițială creată pentru aplicația informatică a fost cea de control manual de către operatorul uman a unui singur servomotor, această testare intermediară a fost realizată cu placă de dezvoltare Arduino Uno, placa de expansiune Sensor Shield V5.0 și un servomotor SG90. Aplicația finală prezintă de asemena control din partea operatorului uman prin interfața aplicației, unde poți alege tipul de deplasare pe care vrei să îl realizeze robotul, dar odată ales tipul de deplasare, robotul poate funcționa fără a mai fii influențați parametrii de deplasare. Parametrii de deplasare sunt deja definitizati, astfel oferind câteva caracteristici autonome robotului, fiind mult mai ușor de utilizat. Pe când, în verificarea inițială, am folosit un element de control de tip Dial pentru a modifica numărul de pulsuri dat servomotorului, fiind necesar controlul manual al operatorului în mod constat, dacă operatorul uman nu mișcă dial-ul, nici servomotorul nu se va roți, astfel oferind un control mult mai dificil și nu foarte exact. În algoritmul folosit în verificarea inițială a servomotorului, viteza, timpi de așteptare, executarea comenzii la un timp dat și setarea poziției servomotorului la un număr de impulsuri exact nu este posibilă. Totuși, verificarea inițială a avut rolul de a înțelege mai bine funcționalitatea servomotoarelor, cât și funcționalitatea bibliotecii MakerHub Linx, care oferă funcții specializate pentru controlul servomotoarelor.

Verificarea inițială, a fost realizată utilizând o buclă while, pentru a putea controla în timp real servomotorul. În afară buclei while avem funcția de OpenSerial.vi, pentru a putea comunica pe portul serial al plăcii de dezvoltare, data de ieșirea a acestuia Linx Resource va fi conectată la ServoOpenOneChannel.vi, unde vom specifica și canalul servo la care este conectat servomotorul, În interiorul buclei while avem o funcție de SetPulseWidthOneChannel.vi, unde vom conecta Linx Resource, elementul de control numeric de tip Dial, care va fii responsabil de controlul servomotorului și elementul de control numeric pentru canlul servo. Datele de ieșire ale funcției SetPulseWidthOneChannel.vi vor deveni date de intrare pentru ServoCloseOneChannel, aflat în afară buclei while, responsabil pentru închiderea canalului servo și în continuare com conecta la Close.vi, care are rol de a închide comunicarea dintre program și placa de dezvoltare pe portul serial când operatorul uman își dorește acest lucru. În aplicație, au fost folosite funcțiile de Wait (ms) setat pentru 10ms și LoopRate (Hz).

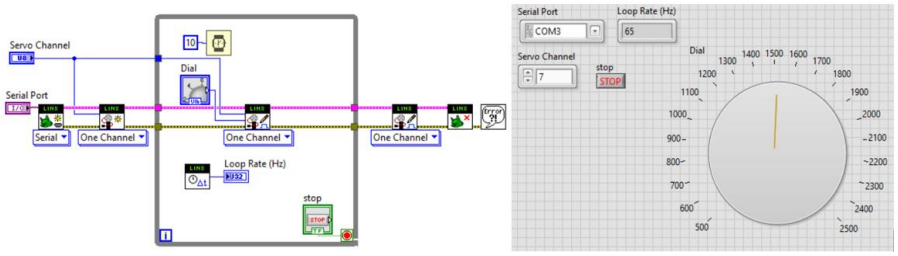

Fig. 2.1.1 Block Diagram și Front Panel pentru Verificare initială servomotor

Rezultatele obținute de pe urmă verificării au fost pozitive, am aflat cum se controlează un servomotor prin impulsuri, cum reacționează servomotorul la comandă oferită. Din aceasta prima verificare am învățat de erorile de tip timeout și că trebuie să ofer mai mult timp pentru executarea comenzilor, astfel folosind funcțiile de tip Wait (ms).

A doua verificare făcută a fost pentru algoritmul de deplasare cu mișcare prin târâre. Am folosit cunoștințele dobândite din prima verificare și am realizat un VI pentru testarea ipotezelor.

Această verificare este făcută pentru deplasarea cu mișcare prin târâre a trei servomotoare. Deschiderea portului serial și a canalului servo sunt la fel ca în prima verificare, dar elementul de control numeric pentru canalul servo este o matrice. În interiorul matricei sunt incluse canalele servo respective aplicației. Punând într-o matrice canalele servo poate reprezenta un avantaj și un dezavantaj. Utilizând ServoOpenNChannels oferă o funcție simplă de scris și de acționat, dar nu oferă la fel de multe opțiuni precum oferă aplicația informatică finală. În această aplicație, servomotoarele sunt acționate în același timp, nu în mod secvențial. Se poate realiza deplasare cu mișcare prin târâre acționând servomotoarele în același timp dacă schimbi vitezele servomotoarelor în mod individual, dar din testele pe care le am realizat, am constatat că acționarea servomotoarelor poate deveni neregulată, ducând la erori ale aplicației, cât și erori de direcție a deplasării, pe când dacă servomotoarele sunt acționate secvențial, deplasarea robotului este mai bună că timpi, distanța parcursă și direcție de deplasare.

În interiorul buclei while, am construit un structure case pentru sistemul principal de deplasare Rostogolire Neutru/Dreapta/Stângă, când acționăm dial-ul corespunzător structurii case în poziția 0, subsistemele de Deplasare Neutru/Înainte/Înapoi și de Deplasare Neutru/Dreapta/Stânga pot fii acționate. În poziția 1, se realizează doar funcția de rostogolire dreapta și în poziția 2, se realizează doar funcția de rostogolire stânga.

În interiorul strucutre case-ului, folosind case selectorii cu valoare de adevărat și fals, am putut crea o perioadă de pulsații, unde servomotorul poate să acționeze în limitele impuse, folosindu-ne de logică matematică pot scrie în program cu ce viteză servomotoarele se pot deplasa. Astfel de este impus din program că servomotorul să se deplaseze de la o poziție de 1500 µs la una de 2000 µs, voi selecta viteza specificând cu câte impulsuri trebuie să crească până la valoarea dorită.

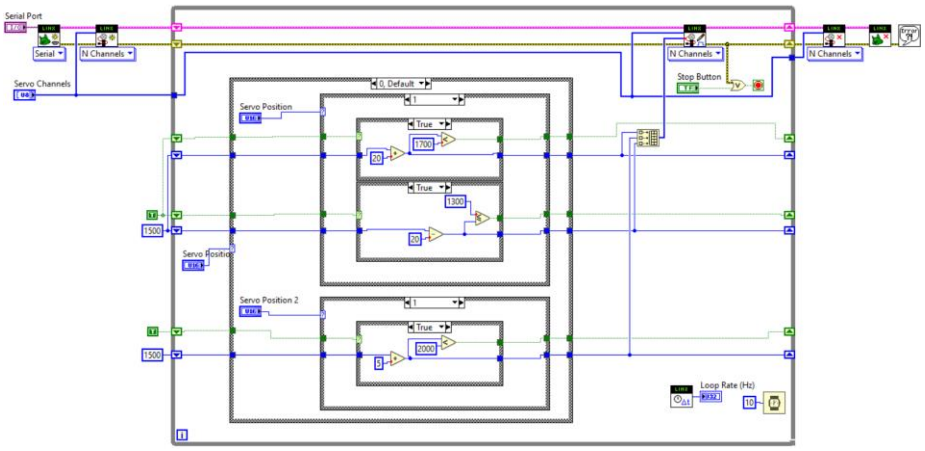

Fig. 4.1.2 Block Diagram, Verificare intială algoritm deplasare

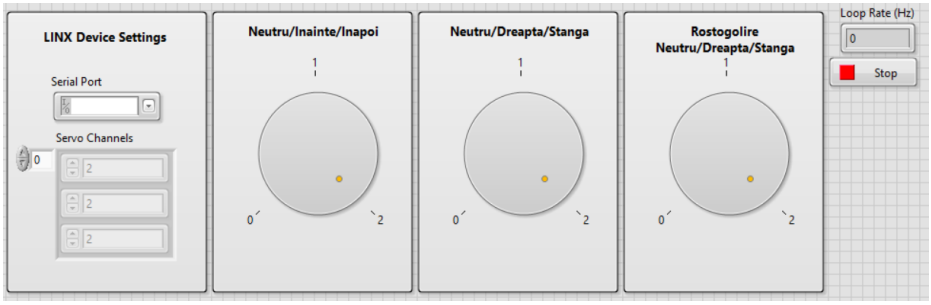

Fig. 4.1.2 Front Panel, Verificare intială algoritm deplasare

După realizarea unui algoritm inițial de deplasare a unui robot mobil cu mișcare prin târâre, am realizat algoritmul final utilizând cunoștințele dobândite din programul de experimente și verificări inițiale.

Algoritmul de citire senzor ultrasonic l-am creat separat inițial, că după să îl integrez în VI-ul principal pentru aplicația informatică. Astfel, componentele aplicației informatice sunt: algoritmul de deplasare, format din sistemele și subsistemele pe care le prezintă, și algoritmul de citire senzor ultrasonic.

4.2 Rezultate obtinute la testarea algoritmului final

Testarea algoritmului final de deplasare a fost realizat utilizând trei module și arăta faptul că viteza de deplasare înainte și înapoi a robotului este de 1 metru pe minut. A se lua în considerare, că viteza de deplasare a robotului poate varia în funcție de coeficientul de frecare, robotul se deplasează mult mai eficient și rapid pe suprafețe cu un coeficient mare frecare. Pentru a combate pierderea de cuplu mecanică care ajută la deplasare, pe zona de contact a modulelor cu solul am suprafețe de cauciuc care vor crea o frecare mai bună cu terenul. Testul a fost realizat pe o suprafață de lemn (parchet), iar modulele robotului sunt realizate din HIPS (High Impact Polystyrene). Înainte de a aplică suprafețele de cauciuc, robotul alunecă pe suprafața de lemn laminat, dar după aplicarea garniturilor robotul se putea deplasa pe suprafață. Coeficientul de frecare dintre materialele plastice și lemn este de 0.4, pe când coeficientul de frecare dintre cauciuc este de 0.7-0.95, fiind mai mare și recomandat pentru aplicația noastră.

De asemenea, am testat citirea senzorului ultrasonic și distanță pe care o poate citii. Astfel senzorul poate citii până la o distanță de 20 cm.

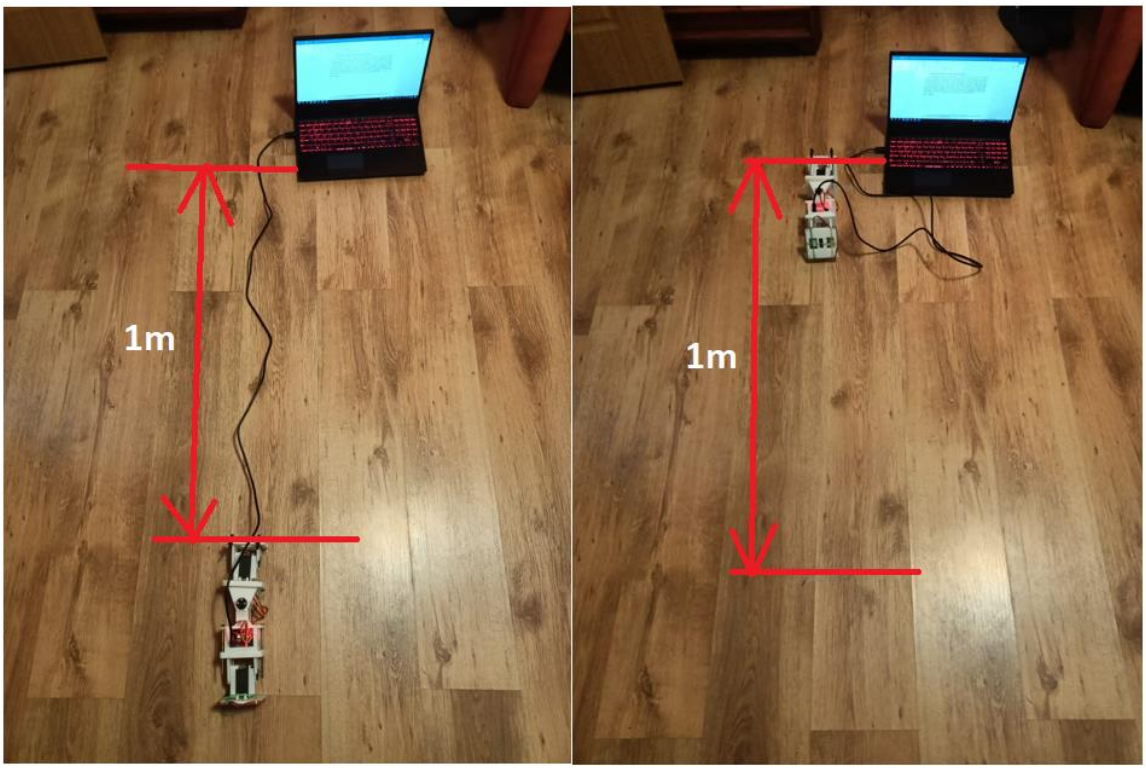

Fig. 3.2.1 Testare deplasare robot

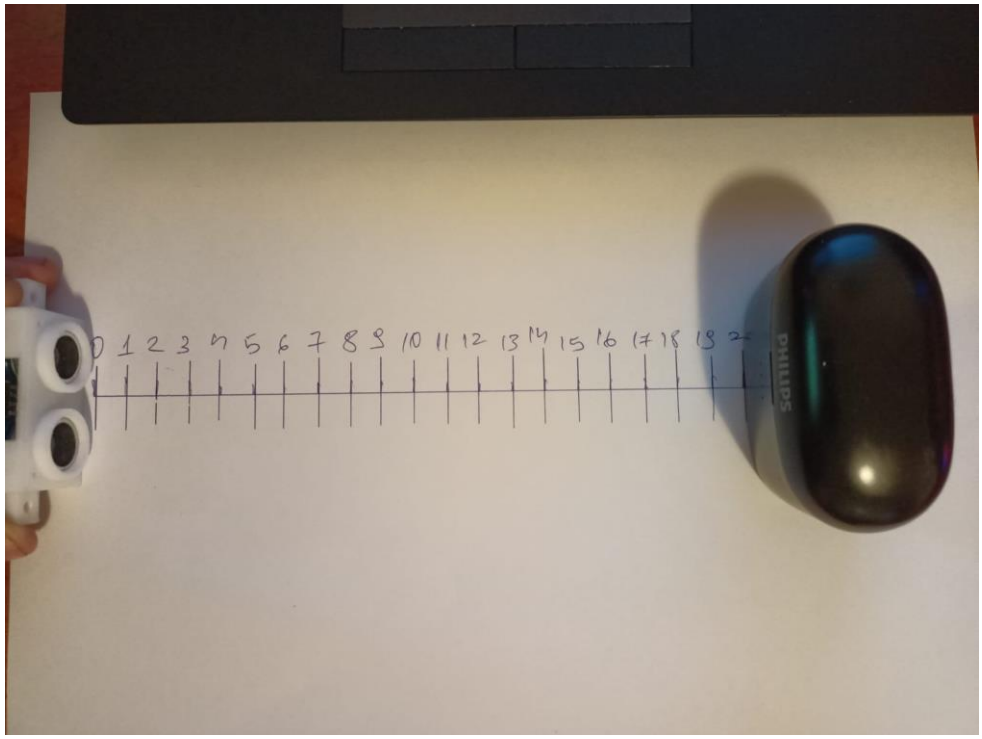

Fig. 4.2.2. Testarea reușita de citire cu senzor ultrasonic a unui obiect la o distantă de 20 cm