

**KAUNO TECHNOLOGIJOS UNIVERSITETAS**

|  |
| --- |
| **ELEKTROS IR ELEKTRONIKOS FAKULTETAS** |

**Povilas Rūgys**

**Vilius Zobėla**

**Robotizuotų sistemų modeliavimas**

(T125B151)

**IRC5 valdiklio sąsajos su PC platformą sudarymas naudojant WindowsForms aplikaciją**

Projektas

|  |  |
| --- | --- |
|  | **Vadovas**  Lekt. G. Narvydas |

**KAUNAS, 2019**

Turinys

[1. Įvadas 2](#_Toc5792282)

[2. ABB IRB 120 manipuliatorius 2](#_Toc5792283)

[3. IRC 5 Valdiklis 3](#_Toc5792286)

[4. FlexPendant Valdymo Operatorius 4](#_Toc5792287)

[5. WinForms vartotojo sąsaja su IRC5 6](#_Toc5792288)

[6. Kodo apžvalga 7](#_Toc5792289)

[7. Naujų komandų aprašymas 9](#_Toc5792290)

[8. Rezultatai 11](#_Toc5792291)

[9. Priedas A 12](#_Toc5792292)

# **Įvadas**

Pagrindinis darbo tikslas įgauti pagrindinių IRB 120 manipuliatoriaus valdymo žinių prgramuojant RAPID kalba RobotStudio Aplinkoje ir C# kalba, sudarant WinForms aplikaciją, Visual Studio aplinkoje.

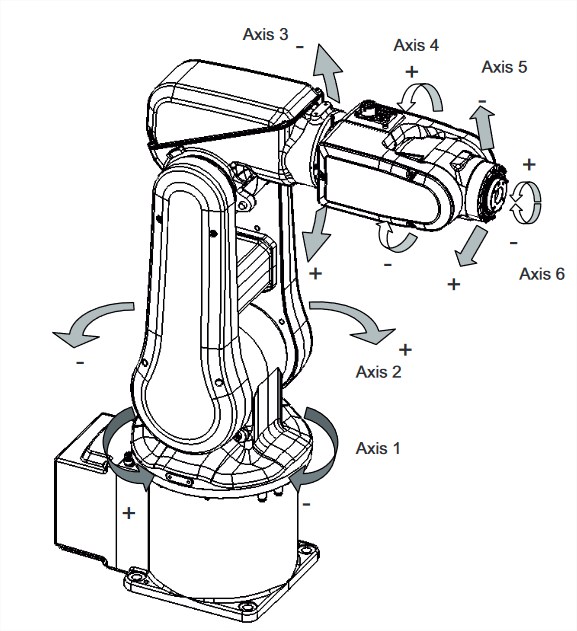
Darbo užduotis:

1. Susipažinti su ABB IRB 120 manipuliatoriumi ir ABB IRC 5 valdikliu.
2. Susipažinti su ABB FlexPendant rankiniu operatoriumi.
3. Susipažinti su ABB RobotStudio projektavimo aplinka.
4. Susipažinti su RAPID programavimu.
5. Sudaryti .NET platformos sąsają su IRC 5 valdikliu.
6. Parašyti C# ir RAPID programas, kurios leistų vartotojui valdyti manipuliatorių.

Priedą A sudarys naudojamas RAPID kodas.

# ABB IRB 120 manipuliatorius

IRB 120 yra 6 ašių manipuliatorius. Jo maksimalus siekis yra 580mm, krovinio svoris gali būti iki 3 kg.



pav. 1



# IRC 5 Valdiklis

IRC 5 valdiklis vykdo komandas judindamas IRB 120 manipuliatorių. Jį galima valdyti FlexPendant rankiniu operatoriumi arba prisijungus ethernet kabelį į PC naudojantis RobotStudio programavimo ir projektavimo aplinka. Jis įjungiamas sukamo kontaktoriaus įjungimu. Esant pavojingai situacijai, valdiklis gali būti išjungiamas raudono mygtuko paspaudimu. Norint paleisti programą, turi būti įjungtas automatinis režimas pasukant raktą į kairiąją poziciją ir paspaustas baltas mygtukas, kad būtų įjungti varikliai. Norint valdyti manipuliatorių per FlexPendant operatorių turi būti įjungiamas rankinis režimas pasukant raktą į dešinę poziciją.



D

C

A

B

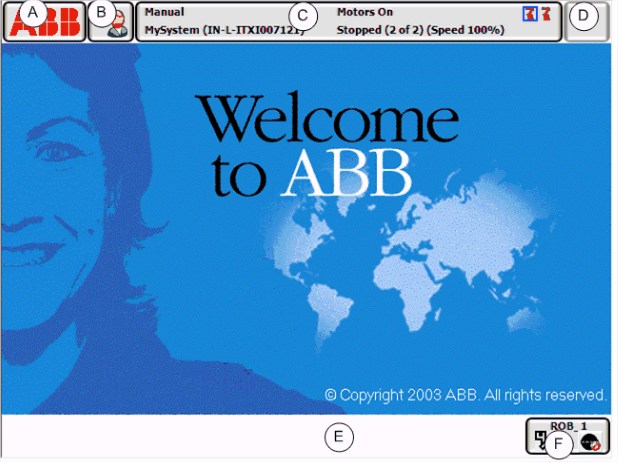
pav. 2 IRC 5 Valdiklis

# FlexPendant Valdymo Operatorius

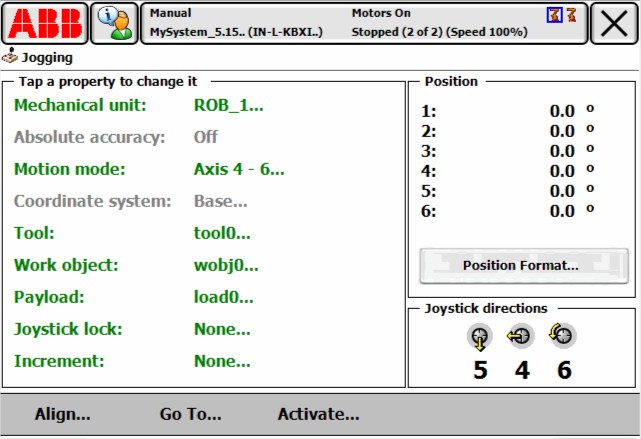
FlexPendant yra rankomis laikomas valdymo prietaisas naudojamas atlikti daugumą užduočių, kontroliuojant robotų sistemą: pradėti programoms, valdyti manipuliatorių, modifikuoti RAPID programas, keisti parametrus. FlexPendant susideda iš techninės ir programinės įrangos ir pats yra kaip atskiras kompiuteris. Jis yra neatskiriama IRC5 valdiklio dalis ir yra sujungtas su valdikliu per kabelį.

Norint išbandyti operatorių reikia pirmiausiai įjungti IRC5 valdiklį (Pav.2 A) ir kai pasikraus valdiklio operacinė sistema įjungti rankinį režimą pasukus raktą pagal laikrodžio rodyklę (Pav.2 D).

Tuomet reikia nueiti į ABB meniu (Pav.4 A), paspausti skiltį „Jogging“ ir nustatyti „Motion mode“ į „Linear“ (Pav.5 A), o „Coordinate system“ į Base (Pav.5 B). Ties laukeliu „Joystick directions“ galima pamatyti kaip yra valdomas manipuliatorius valdymo lazdele. Rodyklės nurodo manipuliatoriaus galinio įtaiso judėjimą teigiama ašies puse pasirinktoje koordinačių sistemoje.



pav. 3 Pradinis langas



B

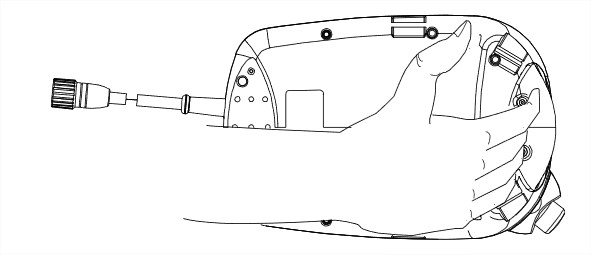
A

C

pav. 4 Jogging parinktys

Norint atlikti judėjimo operacija reikia įjungti saugikli FlexPendant šone, kaip parodyta.(Pav 6)

**Pastaba: Paspaudus saugiklį per stipriai, jis neįsijungs.**



pav. 5 Saugiklio įjungimas

Išgirsite kaip įsijungia varikliai ir judinant valdymo lazdelę.(Pav 7 A)

C



A

B

pav. 6 FlexPendant

Roboto judėjimo greitis priklausys nuo to, kiek pasukta ar palenkta lazdelė.

**Ispėjimas: Patartina iš pradžių menkai pasukti lazdelę, kadangi manipuliatorius gali greitai pajudeti ir įsirėžti, jeigu aplinkoje yra objektų. Esant pavojingai situacijai išjungti valdiklį galima su „Emergency Stop“ mygtuku (Pav 7 C).**

Ant roboto esantis darbo įrankis gali būti perorientuojamas. Jeigu „Tools“ skiltyje yra parinktas „tool0“ (Pav.5 C), kurio koordinatės yra ties 6-ja roboto ašimi, įrankis jį galima sukinėti įjungus reorentacinį režimą „Motion mode“ skiltyje (Pav.5 A) arba FlexPendant pulte paspaudus mygtuką, kuris keičia tarp „Linear“ ir „Reorient“ režimų (Pav.7 B).

# WinForms vartotojo sąsaja su IRC5

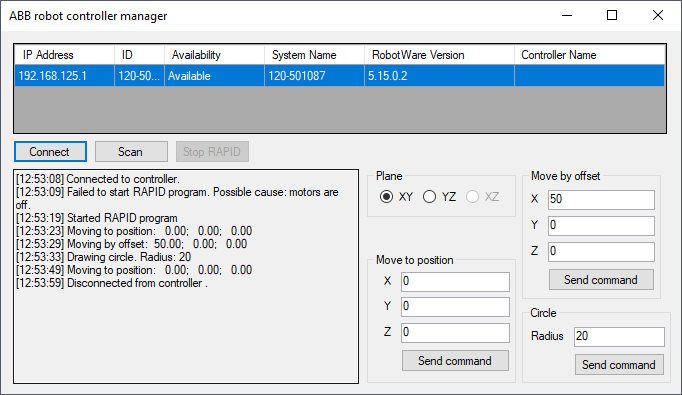
Paleidus programą bus ieškoma prijungtų valdiklių. Prijungus valdiklį per ethernet ir paleidus vartotojo sąsają arba paspaudus „Scan“ mygtuką prie įrenginių turėtų pasirodyti prijungtas valdiklis. Pasirinkus norimą valdiklį ir paspaudus „Connect“, bus prisijungta prie valdiklio. Norint pradėti valdymą, reikia paleisti RAPID kodą su „Start RAPID“ mygtuku. Kol nepaleistas šis kodas, tol komandų robotas nepriims.

Šiuo metu įdėtos 3 galimos komandos:

1. Judėjimas pagal absoliučias koordinates, kur (0, 0, 0) yra HOME pozicija.
2. Judėjimas pagal reliatyvias koordinates, kur (0, 0, 0) yra esama pozicija.
3. Apskritimo piešimas pagal pasirinktą plokštumą.

Visos koordinatės ir ilgiai yra milimetrais.

Komandoms išsiųsti naudojami „Send command“ mygtukai.



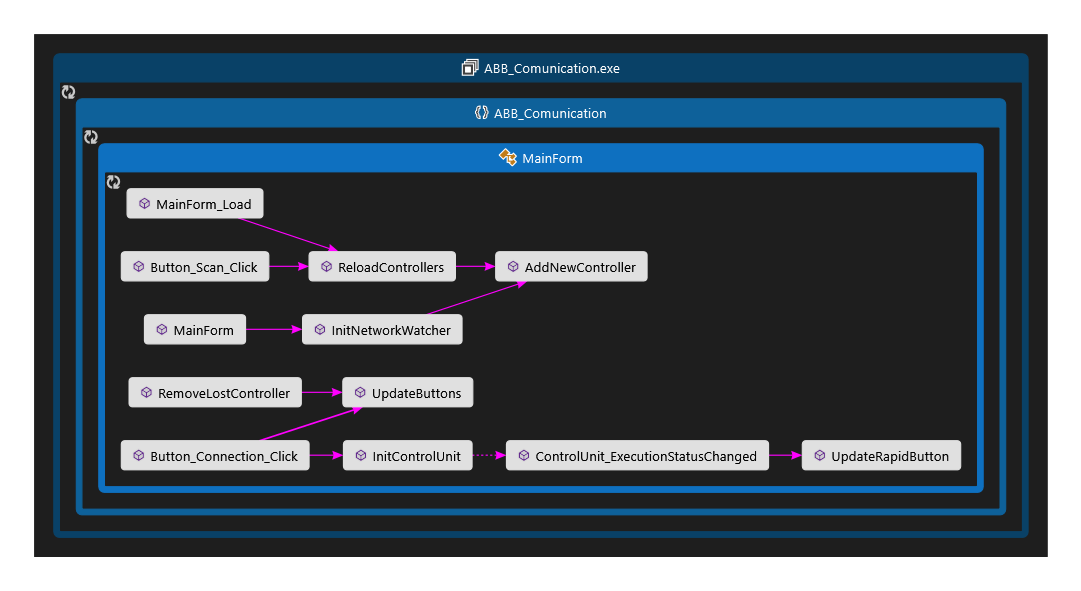
Pavyzdiniame lange buvo vykdomos tokios vartotojo komandos:

1. Prisijungimas prie valdiklio.
2. Bandymas pradėti vykdyti RAPID kodą.
3. Įjungus motorus ir vėl pabandžius buvo pradėtas vykdyti RAPID kodas.
4. Komanda „Move to position“, judėjimas į (0, 0, 0) koordinates.
5. Komanda „Move by offset“, judėjimas į (50, 0, 0) nuo buvusios (0, 0, 0) koordinatės. Prie pozicijos (20, 20, 20) tai pajudintų manipuliatorių į (70, 20, 20).
6. Komanda piešti 20mm spindulio apskritimą XY plokštumoje.
7. Atsijungimas nuo valdiklio.

# Kodo apžvalga

Vartotojo sąsajos kodas susideda iš formos logikos ir ControlUnit klasės atsakingos už IRC5 valdiklio komunikaciją.

Formos logika:



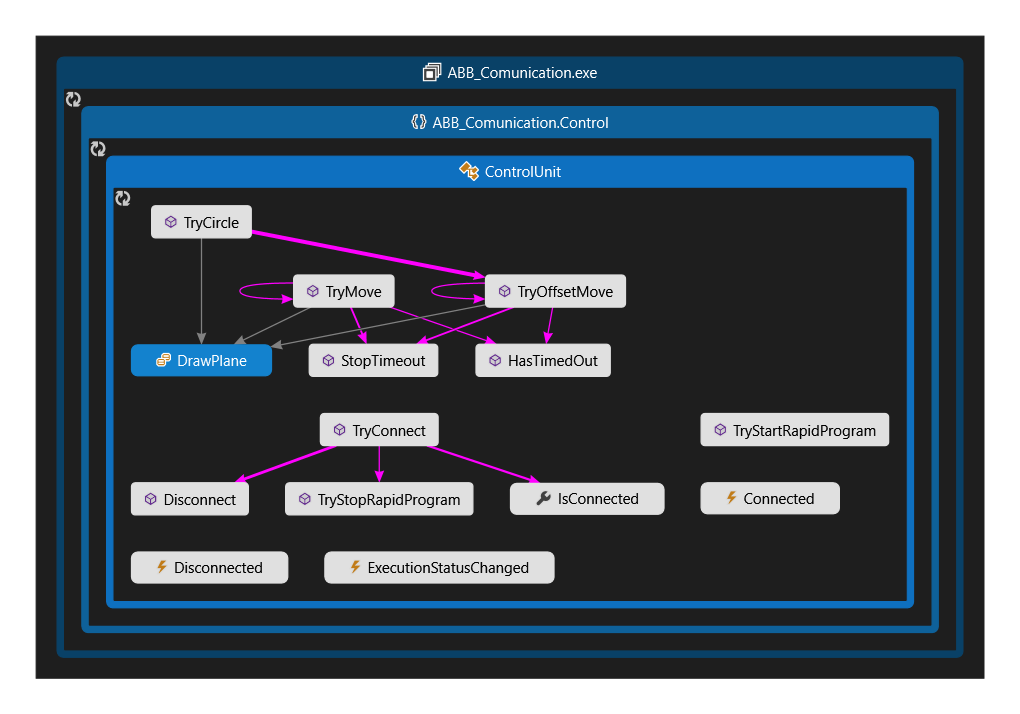
Šiame code map pateikta pagrindinė sąsajos logika.

Inicializavus formą yra inicializuojamas NetworkWatcher objektas, kurio dėka randami nauji valdikliai ar pašalinami seni valdikliai, kurie yra prijungiami ar atjungiami. Šiam objektui yra priskiriami „AddNewController“ ir „RemoveLostController“ metodai prie atitinkamų events.

Užkrovus formą ar vėliau paspaudus “Scan“ mygtuką vykdoma valdiklių paieška.

Visi rasti valdikliai yra atnaujinami viršuje esančioje lentelėje. Prie „Connect” mygtuko yra priskirtas „Button\_Connection\_Click“ event, atsakingas už prisijungimą prie valdiklio.

„UpdateButtons” ir „UpdateRapidButton“ metodai yra atsakingi už mygtukų turinio atnaujinimą.



Šiame code map yra pateikta ControlUnit logika.

Events „Connected“, „Disconnected“ ir „ExecutionStatusChanged“ yra vykdomi atitinkamai prisijungus, atsijungus ir pasikeitus RAPID kodo vykdymui.

Metodas „Disconnect” atsijungia nuo valdiklio.

Metodai „TryStartRapidProgram“ ir „TryStopRapidProgram” atitinkamai pradeda ir sustabdo RAPID kodo vykdymą.

Kintamasis „IsConnected“ grąžina prisijungimo prie valdiklio būklę.

Metodas „TryConnect“ prisijungia prieš tai atsijungdamas, jei buvo prisijungęs, bei sustabdo RAPID kodo vykdymą, jei prisijungus šis yra vykdomas.

Metodai „TryMove“ ir „TryOffsetMove“ yra bazinės roboto judesio komandos, naudojamos visų naujų komandų kūrimui. Judesio vykdymui trukus per ilgai, „HasTimedOut“ metodas grąžina „true“ir komandos vykdymas yra nutraukiamas. Komandoms taip pat gali būti naudojamas enum „DrawPlane“, nurodantis, kurioje plokštumoje turėtų būti vykdomos komandos.

Metodas „TryCircle“ yra nauja komanda, kuri naudojasi „TryOffsetMove“, ir piešia nurodyto spindulio apskritimą nurodytoje plokštumoje.

# Naujų komandų aprašymas

Siekiant aprašyti naujas komandas reikia sukurti naują metodą norimai komandai ControlUnit klasėje ir iškviesti jį mygtuko paspaudimu iš vartotojo sąsajos. Pavyzdys pasitelkiant „TryCircle“ metodą:

public bool TryCircle(DrawPlane plane, double radius)

Viduje yra vykdoma:

Logger.InvokeLog($"Drawing circle. Radius: {radius}");

int stepCount = 64;

double circleLength = radius \* 2 \* Math.PI;

double stepLength = circleLength / stepCount;

return

    stepOut()

    && drawCircle()

    && stepIn();

„stepOut“ metodas, kuris pajudina manipuliatorių prie apskritimo krašto nepiešiant:

switch (plane)

{

    case DrawPlane.XY:

        if (!TryOffsetMove(plane, 0, 0, 5.0, false)

            || !TryOffsetMove(plane, -radius, 0, 0, false)

            || !TryOffsetMove(plane, 0, 0, -5.0, false))

        {

            return false;

        }

        break;

    case DrawPlane.YZ:

        if (!TryOffsetMove(plane, -5.0, 0, 0, false)

            || !TryOffsetMove(plane, 0, 0, -radius, false)

            || !TryOffsetMove(plane, 5.0, 0, 0, false))

        {

            return false;

        }

        break;

    case DrawPlane.XZ:

        if (!TryOffsetMove(plane, 0, -5.0, 0, false)

            || !TryOffsetMove(plane, -radius, 0, 0, false)

            || !TryOffsetMove(plane, 0, 5.0, 0, false))

        {

            return false;

        }

        break;

}

return true;

„drawCircle“ metodas, piešiantis apskritimą:

for (int i = 0; i < stepCount; i++)

{

    double angle = 2 \* Math.PI \* i / stepCount;

    switch (plane)

    {

        case DrawPlane.XY:

            if (!TryOffsetMove(plane, Math.Sin(angle) \* stepLength, Math.Cos(angle) \* stepLength, 0, false))

            {

                return false;

            }

            break;

        case DrawPlane.YZ:

            if (!TryOffsetMove(plane, 0, Math.Cos(angle) \* stepLength, Math.Sin(angle) \* stepLength, false))

            {

                return false;

            }

            break;

        case DrawPlane.XZ:

            if (!TryOffsetMove(plane, Math.Sin(angle) \* stepLength, 0, Math.Cos(angle) \* stepLength, false))

            {

                return false;

            }

            break;

    }

}

return true;

„stepIn“ metodas, kuris pajudina manipuliatorių atgal į apskritimo centrą nepiešiant:

switch (plane)

{

    case DrawPlane.XY:

        if (!TryOffsetMove(plane, 0, 0, 5.0, false)

            || !TryOffsetMove(plane, radius, 0, 0, false)

            || !TryOffsetMove(plane, 0, 0, -5.0, false))

        {

            return false;

        }

        break;

    case DrawPlane.YZ:

        if (!TryOffsetMove(plane, -5.0, 0, 0, false)

            || !TryOffsetMove(plane, 0, 0, radius, false)

            || !TryOffsetMove(plane, 5.0, 0, 0, false))

        {

            return false;

        }

        break;

    case DrawPlane.XZ:

        if (!TryOffsetMove(plane, 0, -5.0, 0, false)

            || !TryOffsetMove(plane, radius, 0, 0, false)

            || !TryOffsetMove(plane, 0, 5.0, 0, false))

        {

            return false;

        }

        break;

}

return true;

„Button\_Circle\_Click“ metodas, kviečiamas mygtuko paspaudimu:

private void Button\_Circle\_Click(object sender, EventArgs e)

{

    \_controlUnit?.TryCircle(\_plane, NumericBox\_CircleRadius.Value);

}

Tokiu būdu aprašius komandą galima paprastai aprašyti naują norimą judesį.

# Rezultatai

# Priedas A

MODULE naujas

VAR num X1 ;

VAR num Y1 ;

VAR num Z1;

VAR bool flagas;

VAR bool flagas2;

VAR num size;

VAR robtarget p1;

PROC naujas\_main()

p1 := CRobT(\Tool:=clean);

IF flagas THEN newSquare p1, size;

ENDIF

IF flagas2 THEN newMoveCommand;

ENDIF

ENDPROC

PROC newSquare(robtarget target, num size)

MoveL Offs(target,0,0,-20),v20,fine,clean;

MoveL Offs(target,size,0,-20),v20,fine,clean;

MoveL Offs(target,size,size,-20),v20,fine,clean;

MoveL Offs(target,0,size,-20),v20,fine,clean;

MoveL Offs(target,0,0,-20),v20,fine,clean;

MoveL target,v20,fine,clean;

flagas := FALSE;

ENDPROC

PROC newMoveCommand()

MoveL Offs(p1,X1,Y1,Z1), v50,fine,clean;

flagas2 := FALSE;

ENDPROC

ENDMODULE