Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

MARKUS METSO

Mitsubishi-robottisolun parantaminen

SÄHKÖ- JA AUTOMAATIOTEKNIIKAN
TUTKINTO-OHJELMA
2025

TIIVISTELMÄ

Metso, Markus: Mitsubishi-robottisolun parantaminen
Opinnäytetyö, AMK
Sähkö- ja automaatiotekniikka
Toukokuu 2025
Sivumäärä: 46

Tässä opinnäytetyössä tutkittiin, kuinka Mitsubishi RV-6S -robotille voitaisiin kytkeä kenttäväyliä ja ohjelmoitavaa logiikkaa hyödyntäen Robotiq 2F-85 -älytarttujaa. Robotti oli yli 20 vuotta vanha ja sitä käytetään RoboAI-laboratoriossa erilaisten demonstraatioiden tekemiseen, joten kenttäväylien ja ohjelmoitavan logiikan lisääminen laitteistoon laajentaa mahdollisuuksia demonstraatioiden toteuttamiseen.

Työssä suunniteltiin, millainen laitteisto tarvitaan ja asennettiin se. Tarttujan sähköjohtojen asennusta varten suunniteltiin ja toteutettiin robottia pitkin kulkeva kaapelikouru 3D-tulostusta hyödyntäen. Fyysisten asennustöiden jälkeen sekä robotille että ohjelmoitavalle logiikalle kirjoitettiin ohjelmat, jotta ne saataisiin toimimaan keskenään. Tarttujan käytöstä robotilla kirjoitettiin yksinkertainen käyttöohje ja sähköasennuksista tehtiin piirustukset.

Opinnäytetyön tavoitteena oli saada käyttöön kaikki älytarttujan ominaisuudet hyödynnettäväksi robotin ohjelmoinnissa. Tavoitteessa onnistuttiin ja robotin sähköasennuksista ja käytöstä saatiin valmiiksi käyttökelpoinen dokumentaatio.

Avainsanat: Mitsubishi, automaatio, robotiikka, tarrain, logiikkaohjelmointi

ABSTRACT

Metso, Markus: Improvement of a Mitsubishi robot cell
Bachelor's thesis
Electrical and automation engineering
May 2025
Number of pages: 46

In this thesis it was studied how it would be possible to improve the capabilities of a Mitsubishi RV-6S robot by attaching a Robotiq 2F-85 smart gripper tool to it using a programmable logic controller (PLC) and serial buses as an interface. The robot was over 20 years old, and it is used in the RoboAI laboratory for different kinds of demonstrations, so adding fieldbuses and a PLC expands the possibilities for implementing demonstrations.

The plans and hardware installations for the robot and PLC were made. For the wiring of the gripper, a cable channel that runs up the chassis of the robot was designed and 3D printed. After the physical installation of the required hardware, software for the robot's controller and the PLC were designed and implemented to enable them to communicate with each other. Instructions were written for the usage of the gripper when programming the robot and schematics for the installed hardware were drawn.

The objective of this thesis was to be able to make use of all the features of the smart gripper when programming the robot. The objective was achieved, and the schematics and instructions were completed.

Keywords: Mitsubishi, automation, robotics, gripper, logic programming

SISÄLLYS

1 JOHDANTO

Automaatiossa tarvitaan jatkuvasti erilaisia ratkaisuja, joilla vanhaa laitteistoa saadaan päivitettyä uudempien laitteiden tarpeisiin. Vaikka vanhemmista laitteista puuttuu usein sellaisia kommunikaatioprotokollia, joita uudemmissa laitteissa käytetään, on ne silti mahdollista saada toimimaan keskenään.

Vaihtoehto laitteiston modernisoinnille on ostaa uudet kalliit robotit tilalle, mikä ei usein ole taloudellisesti järkevä sijoitus. Varsinkin suurempien robottien hankinta-, asennus- ja käyttöönottokustannukset voivat hyvinkin nousta kuusinumeroiseksi luvuksi. Useammista roboteista puhuttaessa hinta voi nousta miljooniinkin euroihin.

Työn tarkoituksena oli saada RoboAI-laboration yli 20 vuotta vanhalle Mitsubishi-robotille kokonaisuus, jossa robottiin saadaan käyttöön moderni tarttujatyökalu kaikkine ominaisuuksineen. Robottia käytetään yleisesti erilaisten demonstraatioiden toteuttamiseen, joten robotin käytettävyyden parantaminen oli todella tärkeässä osassa. Valmiissa ratkaisussa tarttujalle voidaan määritellä ohjelmassa tarttujan nopeus, haluttu paikkapiste ja tarttujan käyttämä voima.

2 TYÖN TOIMEKSIANTAJA

Opinnäytetyön toimeksiantaja on Satakunnan ammattikorkeakoulu, SAMK ja RoboAI-laboratorio. SAMK on suomalainen korkeakoulu, joka työllistää yli 500 ihmistä ja jossa opiskelee vuosittain noin 6700 opiskelijaa. Opinnäytetyö kehittää koulun edellytyksiä tarjota laadukasta opetusta opiskelijoille ja laajentaa yhteistyömahdollisuuksia paikallisten tekniikan alan yritysten kanssa.

3 LAITTEISTON KUVAUS

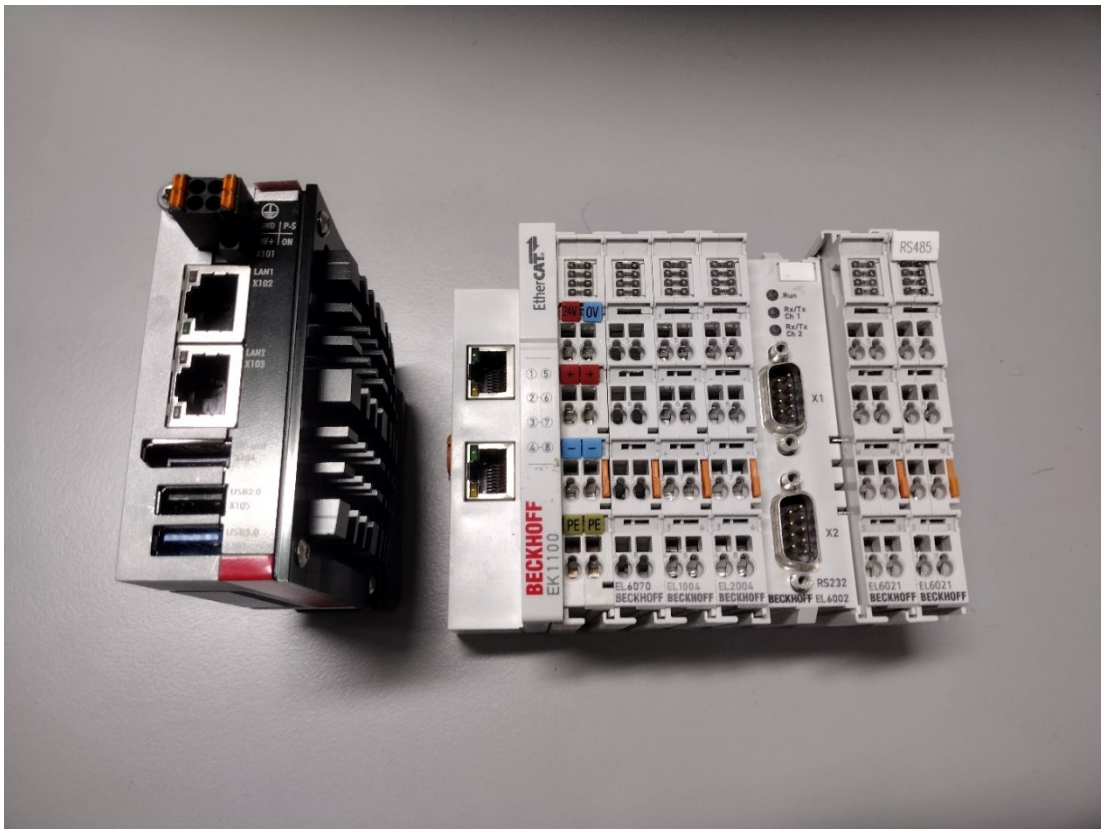
3.1 Robotti

Opinnäytetyössä käytettiin Mitsubishin RV-6S-robottia. Robotin kontrolleri eli ohjausyksikkö on mallia CR2B-574. Robotti on kuusiakselinen käsivarsirobotti ja teollisuusrobotti.

Kontrollerin käsiohjain eli opetuspendantti toimii kontrollerin käyttöliittymänä ja on mallia R46TB. Pendantin kosketusnäyttö helpottaa robotin ohjelmointia huomattavasti verrattuna vanhempaan pendanttiin R28TB. Pendantti on varustettu kuolleenmiehenkytkimellä eli sallintalaitteella, hätäseispainikkeella ja USB-portilla ohjelmien varmuuskopiointiin.

3.2 Beckhoff C6015 ja kortit

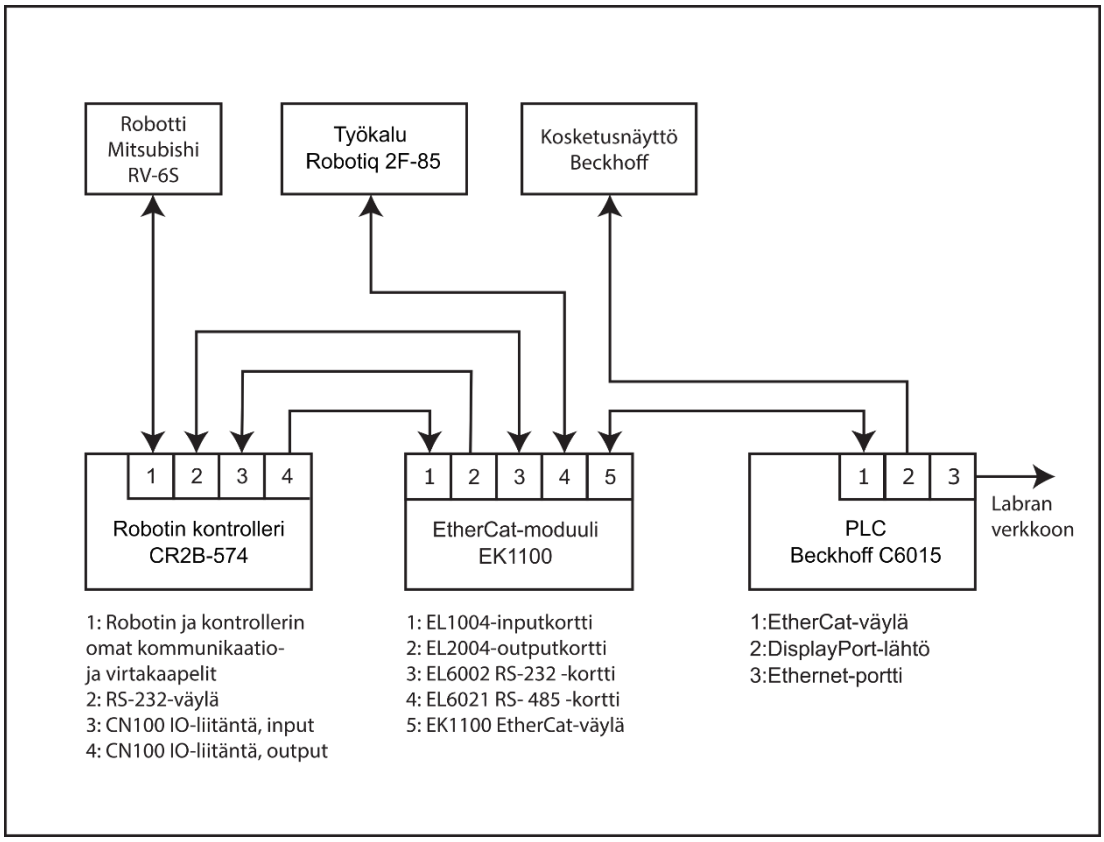
Työssä käytettävä ohjelmoitava logiikka (PLC) on Beckhoffin C6015-kompakti PLC (kuva 1). Koska PLC:ssä ei ole laajennuskorteille paikkoja, otettiin käyttöön myös EtherCat-moduuli EK1100, johon laajennuskortit voitiin asentaa. Moduuliin liitettiin lisenssikortti EL6070, digitaalinen input-kortti EL1004, digitaalinen output-kortti EL2004, RS232-kortti EL6002 ja RS485-kortti EL6021. PLC ja moduuli liitettiin toisiinsa Cat5e-patchkaapelilla EtherCat-portteihin.



Kuva 1. Beckhoff C6015 -kompakti PLC ja EK1100-moduuli laajennuskorteilla.

Laitteistoon asennettiin lisenssikortti EL6070 sen takia, että projektin tarvitsemat lisenssit pysyvät tallessa sellaisessa tapauksessa, jossa logiikka halutaan tai tarvitsee vaihtaa. Laitteistosta saadaan näin toimintavarmempi. Lisenssit on mahdollista tallentaa myös logiikalle, mutta tällaisessa tapauksessa lisenssejä ei pysty käyttämään enää, jos logiikka esimerkiksi hajoaa.

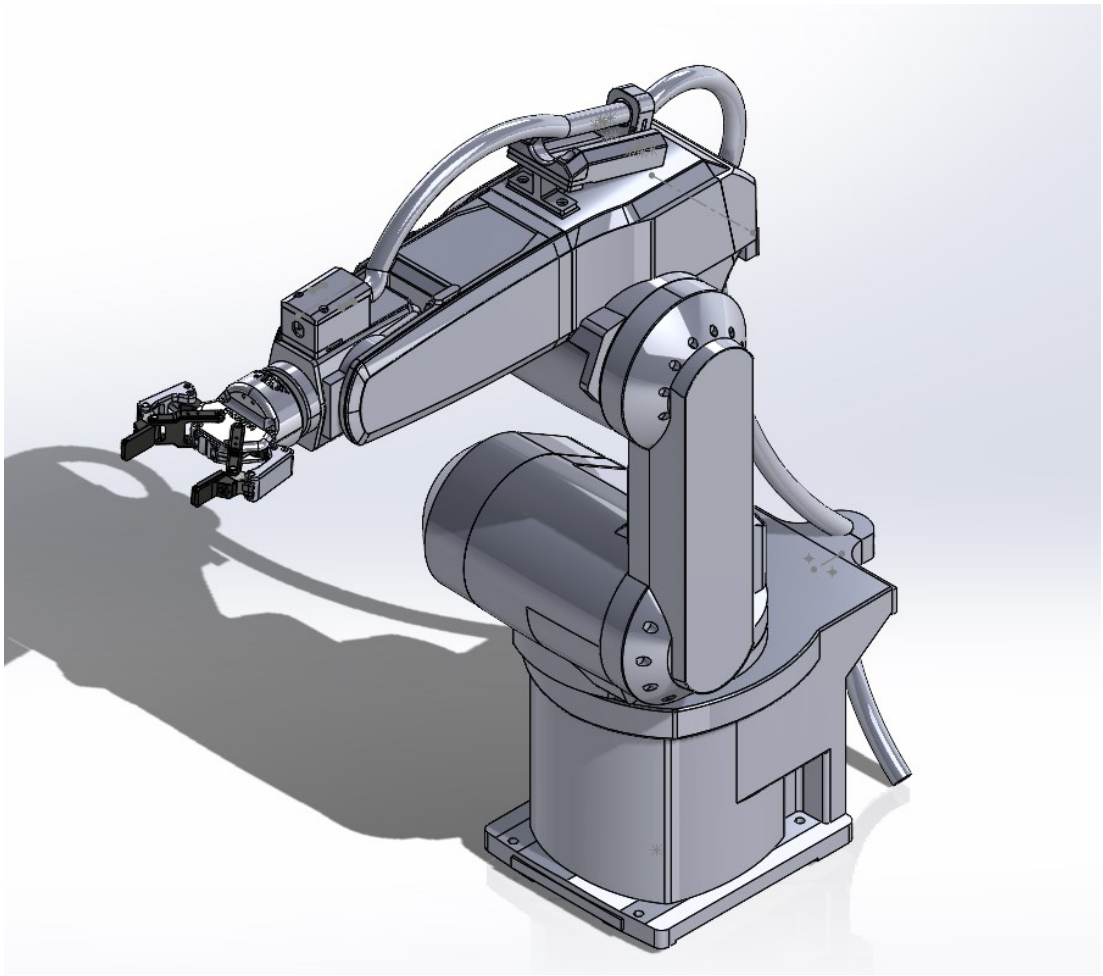
Laitteistoon kuuluu myös Beckhoffin kosketusnäyttö, jolle voidaan tehdä erilaisia visualisointeja logiikan ohjaamiseen. Kosketusnäytön käyttö kuitenkin rajattiin ulos opinnäytetyön laajuudesta. Laitteiston osista tehtiin kaaviokuva, joka yksinkertaistaa kokonaisuuden hahmottamista (kuva 2).



Kuva 2. Kaavio laitteiston osista.

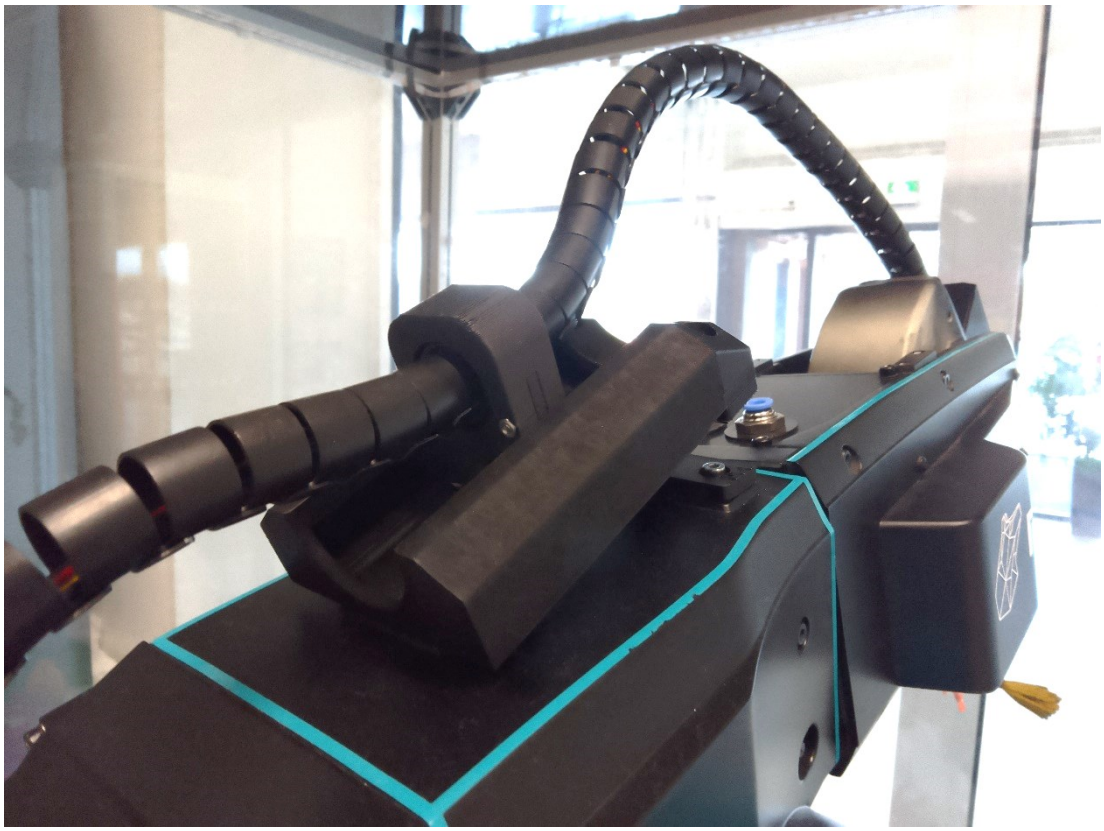
4 ASENNUKSET

Robotille asennettiin sähkökaapeli, jotta tarttuja saataisiin kytkettyä. Koska robotin sisäiset kaapelit todettiin liian ohuiksi tarttujan tarvitsemalle virralle, suunniteltiin ulkoinen kaapelointi. Netistä haettiin robotin 3D-malli, jonka päälle suunniteltiin Solidworks-ohjelmalla osat, joita pitkin voitaisiin asentaa kaapelikouru (kuva 3).



Kuva 3. Kaapelointi Solidworks-ohjelmassa.

Suunnitellut osat tulostettiin PLA-muovista Prusa MK4S -3D-tulostimella. Ongelmaksi havaittiin, että robotin päällä kulkeva kaapeli menee liian mutkalle robottia liikutellessa, joten siihen suunniteltiin liikkuva kisko (kuva 4), jonka avulla kaapelikouru sai liikkumavaraa ongelmakohdassa.



Kuva 4. Kisko, jolla kaapelikourulle saatiin liikkumavaraa.

Kaapelikourun päähän tehtiin tarttujan 5-pinnistä M12-urosliitintä vastaava naarasliitin, johon tarttuja voitiin kytkeä. Näin tarttuja voidaan vaihtaa helposti tarvittaessa.

Ohjelmoitava logiikka C6015, laajennusmoduuli EK1100 ja sen lisäkortit sekä muuntaja, jolta laitteisto saa 24 V käyttöjännitteen asennettiin robottisolussa kontrollerin viereen asennettuun sähkökaappiin. Sähkökytkennöistä piirrettiin sähkökuva, josta ilmenee kojeiden paikat ja sähkökytkennät (Liite 1).

5 DIGITAALINEN I/O KONTROLLERIN JA PLC:N VÄLILLÄ

Beckhoff-logiikan digitaaliset input- ja output-kortit kytkettiin robotin kontrollerin IO-porttiin CN100. Korteilla saatiin käyttöön yhteensä 4 inputtia ja 4 outputtia robotin kontrollerin ja PLC:n välillä. Kytkentää varten tehtiin kaapeli, jonka toiseen päähän tehtiin kontrollerin IO-porttiin sopiva Centronics 50-pin urosliitin ja toiseen päähän pääteholkitetut erilliset johtimet (kuva 5).

Kuva 5. Opinnäytetyötä varten valmistettu Centronics 50-pin -kaapeli.

Koska inputit 0-5 ja outputit 0-3 on varattu kontrollerin omaan sisäiseen käyttöön, valittiin kytkettäviksi signaaleiksi selkeyden vuoksi molempiin suuntiin signaalit 8-11 (kuva 5). Näin varmistettiin, että kontrollerin toiminta ei mene sekaisin. Outputeille kytkettiin 24 V käyttöjännite ja nolla, jotka ne tarvitsevat toimiakseen. Kaapeliin kytkettiin myös FG (frame ground) eli maadoitus. Kytkennässä olennaista oli se, että inputtien COM1-liitäntä tarvitsee kontrollerin tyypistä riippuen joko 0 V tai 24 V potentiaalin, joka oli tässä tapauksessa 0 V, koska kontrolleri on Source-tyyppinen (kuva 6).

(2) Pin numbers of standard parallel input/output cards and signal assignment(CR2B-574)

Table 3-12 : Standard parallel I/O interface CN100pin No. and signal assignment list (2A-CBL □□)

Pin No.	Line color	Function name		Pin No.	Line color	Function name	
		General-purpose	Dedicated/power supply, common			General-purpose	Dedicated/power supply, common
1	Orange/Red A		FG	26	Orange/Blue A		FG
2	Gray/Red A		0V:For pins 4-7	27	Gray/Blue A		0V:For pins 29-32
3	White/Red A		12V/24V:For pins 4-7	28	White/Blue A		12V/24V:For pins 29-32
4	Yellow/Red A	General-purpose output 0	Running	29	Yellow/Blue A	General-purpose output 4	
5	Pink/Red A	General-purpose output 1	Servo on	30	Pink/Blue A	General-purpose output 5	
6	Orange/Red B	General-purpose output 2	Error	31	Orange/Blue B	General-purpose output 6	
7	Gray/Red B	General-purpose output 3	Operation rights	32	Gray/Blue B	General-purpose output 7	
8	White/Red B		0V:For pins 10-13	33	White/Blue B		0V:For pins 35-38
9	Yellow/Red B		12V/24V:For pins 10-13	34	Yellow/Blue B		12V/24V:For pins 35-38
10	Pink/Red B	General-purpose output 8		35	Pink/Blue B	General-purpose output 12	
11	Orange/Red C	General-purpose output 9		36	Orange/Blue C	General-purpose output 13	
12	Gray/Red C	General-purpose output 10		37	Gray/Blue C	General-purpose output 14	
13	White/Red C	General-purpose output 11		38	White/Blue C	General-purpose output 15	
14	Yellow/Red C		COM0:For pins 15-22 Note1)	39	Yellow/Blue C		COM1:For pins 40-47 Note1)
15	Pink/Red C	General-purpose input 0	Stop(All slot) Note2)	40	Pink/Blue C	General-purpose input 8	
16	Orange/Red D	General-purpose input 1	Servo off	41	Orange/Blue D	General-purpose input 9	
17	Gray/Red D	General-purpose input 2	Error reset	42	Gray/Blue D	General-purpose input 10	
18	White/Red D	General-purpose input 3	Start	43	White/Blue D	General-purpose input 11	
19	Yellow/Red D	General-purpose input 4	Servo on	44	Yellow/Blue D	General-purpose input 12	
20	Pink/Red D	General-purpose input 5	Operation rights	45	Pink/Blue D	General-purpose input 13	
21	Orange/Red E	General-purpose input 6		46	Orange/Blue E	General-purpose input 14	
22	Gray/Red E	General-purpose input 7		47	Gray/Blue E	General-purpose input 15	
23	White/Red E		Reserved	48	White/Blue E		Reserved
24	Yellow/Red E		Reserved	49	Yellow/Blue E		Reserved
25	Pink/Red E		Reserved	50	Pink/Blue E		Reserved

Note1)Sink type:24V/12V(COM), Source type:0V(COM)
Note2)The assignment of the dedicated input signal "STOP" is fixed.

Kuva 6. IO-portin CN100 signaalitaulukko (Mitsubishi, 2019, s. 459). Kytketyt johtimet merkitty keltaisella.

Robotin kontrollerin CN100-portin (kuva 7) input-signaalit kytkettiin logiikan output-korttiin EL2004 ja kontrollerin output-signaalit kytkettiin logiikan input-korttiin EL1004. Kytkentä tehtiin näin, koska digitaaliset IO-signaalit ovat yksisuuntaisia ja signaali kulkee aina lähettävän laitteen output-terminaalista vastaanottavan laitteen input-terminaaliin.

Kuva 7. Kontrollerin IO-portit. Kontrollerin tyyppi (Source) on merkitty liittimen alapuolelle. Missään muualla ei lue tätä tietoa.

6 KOMMUNIKAATIO RS232-VÄYLÄSSÄ

6.1 RS232-väylä

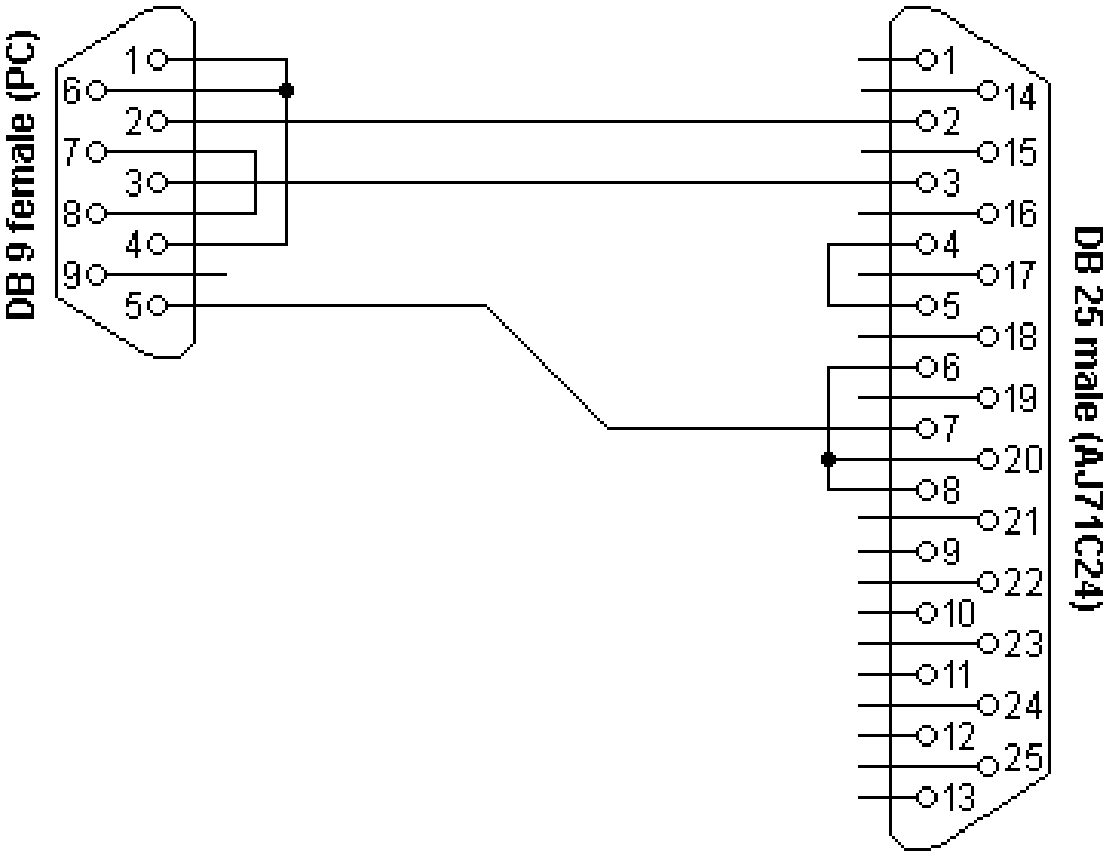
RS232 (Recommended Standard 232) -sarjaliikenneväylä on vuonna 1960 kehitetty kommunikaatioprotokolla erilaisten sähkölaitteiden keskinäiseen tiedonvaihtoon. Nykyään käytössä on vuonna 1969 määritelty RS232C-standardi, jossa määritellään signaalien sähköiset ominaisuudet, liittimien speksit ja sähköisten piirien käyttötarkoitukset. Käytännössä monet laitteet eivät kuitenkaan seuraa standardia ja eri laitevalmistajien liittimet ja signaalit voivat olla jotenkin muuten konfiguroitu. (Reynders ym., 2005, s. 71-72.)

6.2 DTE ja DCE

RS232-laitteita on kahta tyyppiä: DTE (Data Terminal Equipment) ja DCE (Data Communications Equipment). DTE-laitteet ovat päätelaitteita, jotka lähettävät ja vastaanottavat dataa. DCE-laitteet ovat modeemeja tai muita tiedonsiirtolaitteita, jotka vastaanottavat dataa DTE-laitteilta ja siirtävät sitä eteenpäin muille DCE-laitteille (Reynders ym., 2005, s. 71-72.)

Laitteen tyypin voi tarkistaa katsomalla yleismittarilla, mihin liittimen pinneihin tulee jännite, kun laite on päällä ja ei lähetä dataa. Liittimestä mitataan signaalimaapinniä vasten jännite Receive Data (RD)-pinnistä ja Transmit Data (TD)-pinnistä. Jos liittimen maapinnin ja RD-pinnin välillä on negatiivinen jännite, on kyseessä DCE-laite. Jos taas signaalimaapinnin ja TD-pinnin välillä on negatiivinen jännite, kyseessä on DTE-laite. (Advantech, 2018.)

Mitsubishi CR2B-574 -kontrollerin ja EL6002-kortin RS232-liittimet todettiin näin DTE-laitteina toimiviksi. Yksinkertaisin tapa liittää RS232-DTE-laitteita keskenään on käyttää ns. nollamodeemikaapelia (kuva 8). Nollamodeemikaapelissa laitteiden lähetys- ja vastaanottojohtimet on kytketty keskenään ristiin, jolloin laitteet voivat lähettää ja vastaanottaa toistensa dataa. Laitteiden signaalimaajohtimet tulee kytkeä silti yhteen, jotta signaalien potentiaali pysyy samana.



Kuva 8. Nollamodeemikaapeli RS-232-väylään (Bies, 2021).

Nollamodeemikaapelia käytettäessä laitteiden väliset kättelysignaalit voidaan joko kytkeä laitteiden välillä samoin tavoin ristiin tai kytkeä takaisin laitteelle. Yksinkertaisin tapa on kytkeä signaalit takaisin, jolloin laitteiden väliseen kommunikaatioon tarvitaan vain kolme johdinta. Kommunikaatio on tällöin asynkronista. (Dawoud & Dawoud, 2020, s. 19.)

Sarjaliikenneväylän parametrien tulee täsmätä laitteiden välillä, jotta kommunikaatio toimii. Väärin parametroitu väyläliikenne joko ei toimi ollenkaan, kärsii datahäviöstä tai hitaista tiedonsiirtonopeuksista. (Reynders ym., 2005, s. 86.)

6.3 Baud rate

Baud rate on sarjaliikenneväylän tiedonsiirtonopeus. Määritelmällisesti baud rate on seuraavanlainen:

```
Baud rate = 1/bit time = 1 / TB (bits/ second)
```

jossa TB on aika, joka kuluu yhden bitin siirtämiseen (Dawoud & Dawoud, 2020, s. 12). Esimerkiksi 9600 baudin tiedonsiirtonopeudella TB olisi siis 1/9600 s = 0,0001042 s eli 104,2 µS. 9600 baudin nopeudella voidaan siis siirtää 9600 bittiä sekunnissa. Tämä tiedonsiirtonopeus sisältää siirrettävien databittien lisäksi datakehyksen bitit, joten todellinen siirrettävän datan määrä on pienempi.

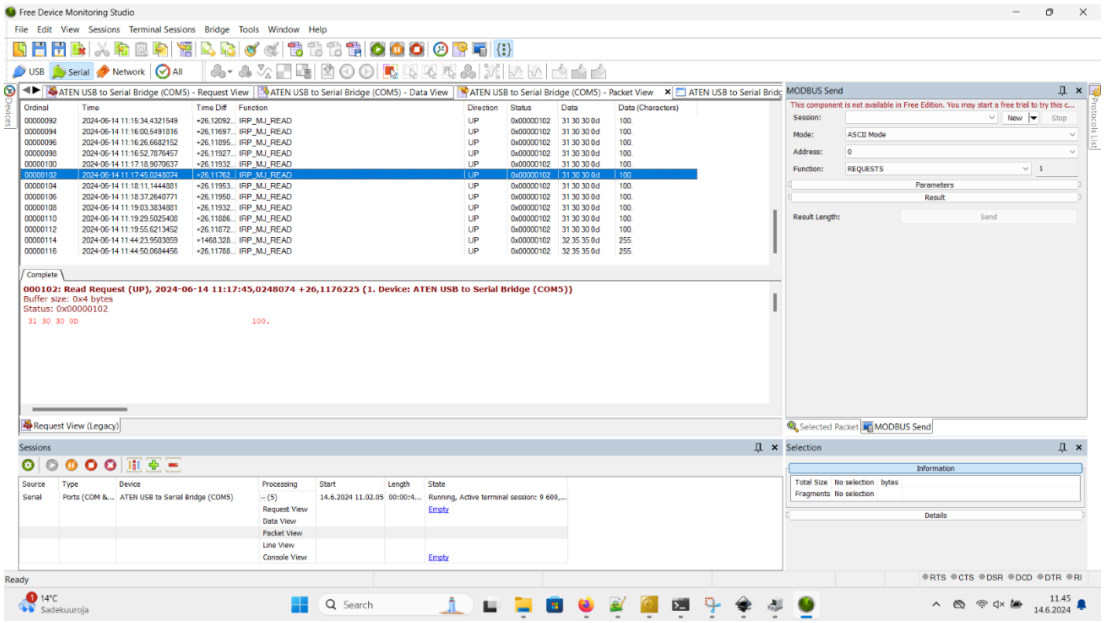
6.4 Synkronisaatio ja datakehys

RS232-kommunikaatio voidaan toteuttaa joko asynkronisoituna tai synkronisoituna. Tässä opinnäytetyössä on käytetty asynkronista tietoliikennettä, jossa data siirretään data framen eli datakehyksen sisällä ilman erillistä kellopulssia, joka määräisi datansiirron tahdin (Dawoud & Dawoud, 2020, s. 19). Tahdin määräävät siis vain kehyksen aloitus- ja lopetusbitit sekä ennalta sovittu yhteinen tiedonsiirtonopeus.

RS232-sarjaliikenteessä kehys koostuu alkubitistä, joka on aina 0. Sen jälkeen siirretään databitit, joita voi olla sovelluksesta riippuen 5-9. Databittien jälkeen tulee valinnainen parity-bitti ja sen jälkeen stop-bitti, joka on aina 1. Parity-bittiä käytetään virheentarkastukseen. (Dawoud & Dawoud, 2020, s.20.)

Robotin kontrollerilta lähtevä sarjaväyläliikenne on vakiolla määritetty lähteväksi 8 data bitin, 1 parity bitin (tässä tapauksessa Even) ja 2 stop bitin muotoisena bittijonona (Mitsubishi, 2019, s.351). Tämän kaltaista datakehystä kutsutaan yleisesti 8E2-koodatuksi.

Koska tässä vaiheessa oli tiedossa kontrollerin RS232-väylän parametrit, kontrolleri asetettiin lähettämään dataa jatkuvasti ja kytkettiin nollamodeemikaapelilla tietokoneen sarjaväyläporttiin. Tämän jälkeen CAS Modbus Scanner -sarjaporttiskanneriohjelmaa käyttämällä päästiin tarkastelemaan minkä muotoista dataa kontrolleri lähettää (kuva 9).



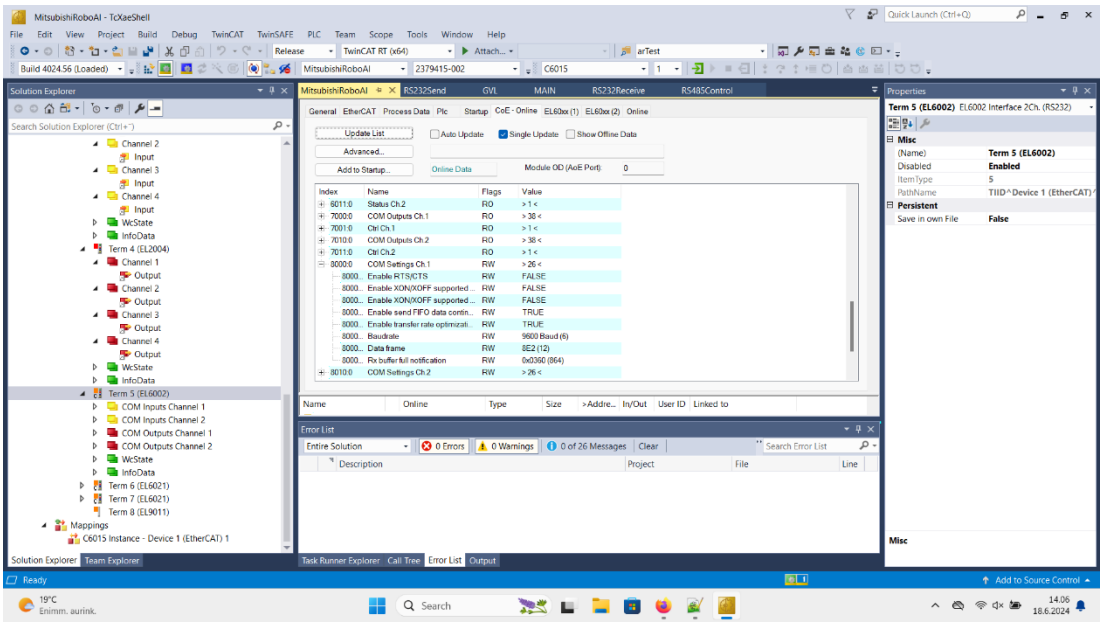
Kuva 9. RS232-väylän seurantaa CAS Modbus Scanner -ohjelmalla.

Kontrollerilta lähetettiin viestejä ”100” ja ”255” koska ne olivat sellaisella alueella, jota haluttiin käyttää ohjaukseen. Koska ”100” tuli muodossa ”31 30 30 0d”, voitiin todeta, että data on ASCII-koodattua. Koska kontrollerin data on 8 bittiä eikä 7 bittiä pitkä, niin kyse on Extended ASCII-koodista. 0d koodin päässä merkitsee ASCII-merkkiä ”Carriage Return”, joka merkitsee datan päättymistä (kuva 10).

Control Characters				Graphic Symbols											
Name	Dec	Binary	Hex	Symbol	Dec	Binary	Hex	Symbol	Dec	Binary	Hex	Symbol	Dec	Binary	Hex
NUL	0	0000000	00	space	32	0100000	20	@	64	1000000	40	'	96	1100000	60
SOH	1	0000001	01	!	33	0100001	21	A	65	1000001	41	a	97	1100001	61
STX	2	0000010	02	"	34	0100010	22	B	66	1000010	42	b	98	1100010	62
ETX	3	0000011	03	#	35	0100011	23	C	67	1000011	43	c	99	1100011	63
EOT	4	0000100	04	$	36	0100100	24	D	68	1000100	44	d	100	1100100	64
ENQ	5	0000101	05	%	37	0100101	25	E	69	1000101	45	e	101	1100101	65
ACK	6	0000110	06	&	38	0100110	26	F	70	1000110	46	f	102	1100110	66
BEL	7	0000111	07	'	39	0100111	27	G	71	1000111	47	g	103	1100111	67
BS	8	0001000	08	(	40	0101000	28	H	72	1001000	48	h	104	1101000	68
HT	9	0001001	09	)	41	0101001	29	I	73	1001001	49	i	105	1101001	69
LF	10	0001010	0A	*	42	0101010	2A	J	74	1001010	4A	j	106	1101010	6A
VT	11	0001011	0B	+	43	0101011	2B	K	75	1001011	4B	k	107	1101011	6B
FF	12	0001100	0C	,	44	0101100	2C	L	76	1001100	4C	l	108	1101100	6C
CR	13	0001101	0D	–	45	0101101	2D	M	77	1001101	4D	m	109	1101101	6D
SO	14	0001110	0E	.	46	0101110	2E	N	78	1001110	4E	n	110	1101110	6E
SI	15	0001111	0F	/	47	0101111	2F	O	79	1001111	4F	o	111	1101111	6F
DLE	16	0010000	10	0	48	0110000	30	P	80	1010000	50	p	112	1110000	70
DC1	17	0010001	11	1	49	0110001	31	Q	81	1010001	51	q	113	1110001	71
DC2	18	0010010	12	2	50	0110010	32	R	82	1010010	52	r	114	1110010	72
DC3	19	0010011	13	3	51	0110011	33	S	83	1010011	53	s	115	1110011	73
DC4	20	0010100	14	4	52	0110100	34	T	84	1010100	54	t	116	1110100	74
NAK	21	0010101	15	5	53	0110101	35	U	85	1010101	55	u	117	1110101	75
SYN	22	0010110	16	6	54	0110110	36	V	86	1010110	56	v	118	1110110	76
ETB	23	0010111	17	7	55	0110111	37	W	87	1010111	57	w	119	1110111	77
CAN	24	0011000	18	8	56	0111000	38	X	88	1011000	58	x	120	1111000	78
EM	25	0011001	19	9	57	0111001	39	Y	89	1011001	59	y	121	1111001	79
SUB	26	0011010	1A	:	58	0111010	3A	Z	90	1011010	5A	z	122	1111010	7A
ESC	27	0011011	1B	;	59	0111011	3B	[	91	1011011	5B	{	123	1111011	7B
FS	28	0011100	1C	<	60	0111100	3C	\	92	1011100	5C	|	124	1111100	7C
GS	29	0011101	1D	=	61	0111101	3D	]	93	1011101	5D	}	125	1111101	7D
RS	30	0011110	1E	>	62	0111110	3E	^	94	1011110	5E	~	126	1111110	7E
US	31	0011111	1F	?	63	0111111	3F	_	95	1011111	5F	Del	127	1111111	7F

Kuva 10. Ascii-taulukko (VLSIFacts, 2023).

Koska siirrettävän baud rate, datakehys ja enkoodaus oli tässä vaiheessa tiedossa, sarjaväyläkaapeli kokeiltiin kytkeä logiikan sarjaväyläkorttiin EL6002. Tiedossa olevat parametrit asetettiin TwinCat-ohjelmassa kortille (kuva 11).

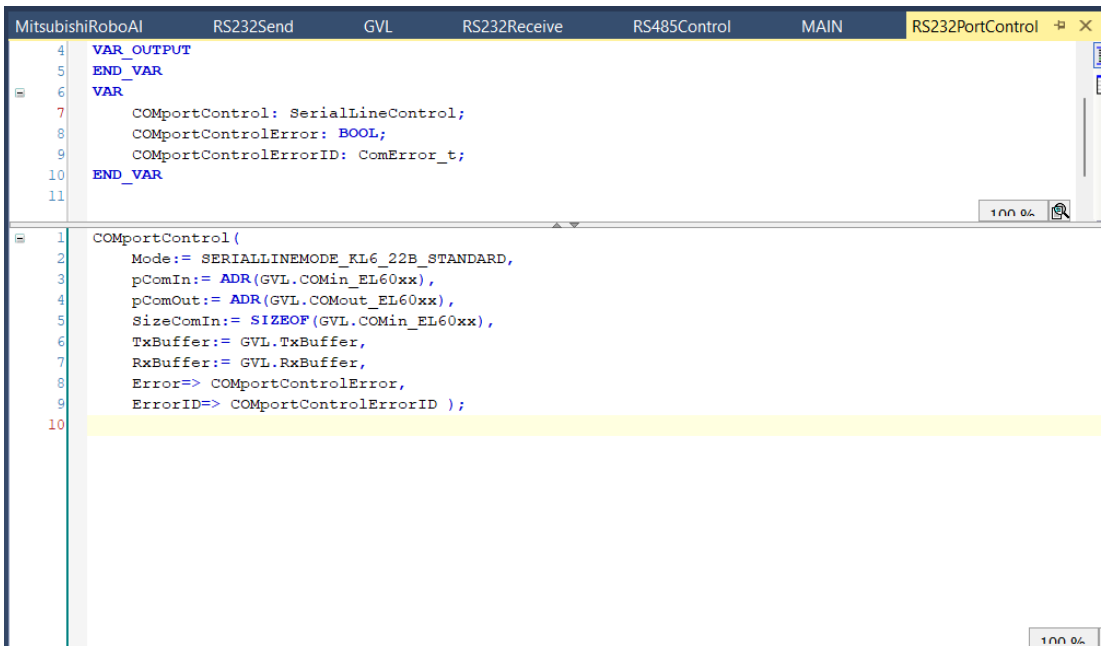


Kuva 11. RS232-parametrit asetettu TwinCat-ohjelmassa.

Baud rateksi valittiin sama 9600 Bps kuin kontrollerille oli asetettu. Data Frameksi valittiin samoin 8E2. Koska kaapeli oli tehty loopback-tyyppiseksi eli ilman kättelysignaaleja, laitettiin Enable RTS/CTS -kohtaan False. Koska kontrolleri ei tue XON/XOFF-ohjelmistokättelyä, nämäkin laitettiin Falseksi.

6.5 Logiikan ohjelmointi

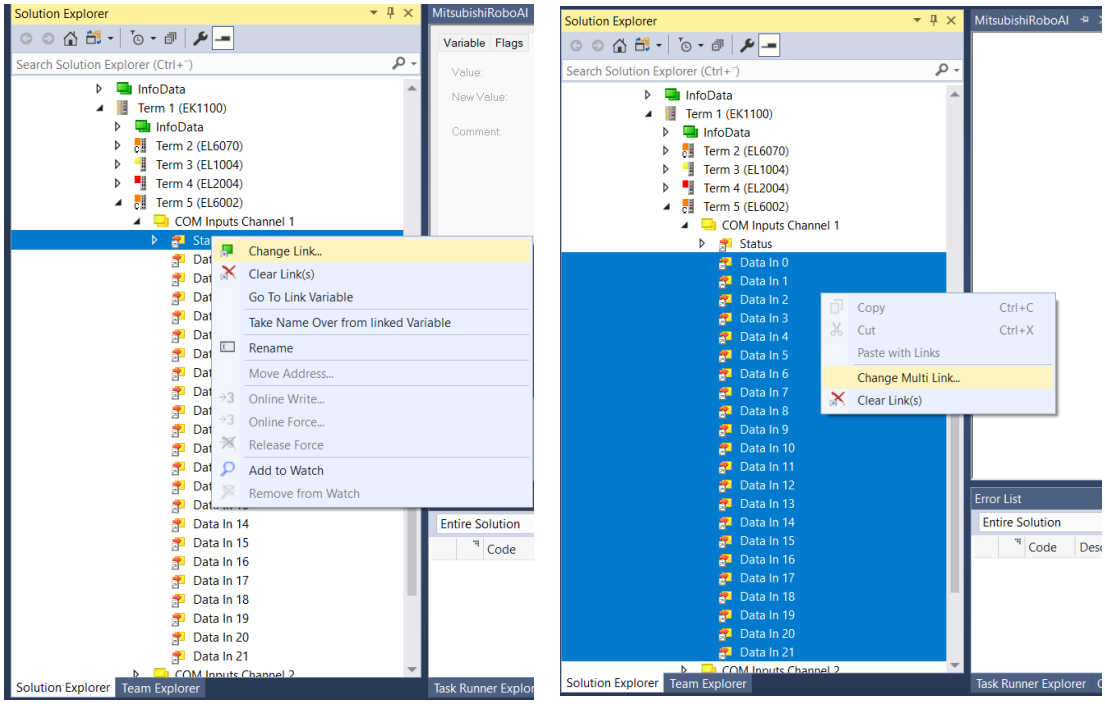
Beckhoffin logiikat toimivat siten, että ohjelmaan täytyy luoda funktio, johon liitetään kortin IO-muuttujat. Logiikalle luotiin funktiolohko nimeltä RS232PortControl, jotta kortin EL6002 sarjaporttiliitintä saadaan käytettyä ohjelmassa (kuva 12). Sarjaportin konfiguroimista varten luotiin funktiolohkon tarvitsemat muuttujat globaaliin muuttujalistaan (GVL), jotta ne olisivat sekä sarjaväyläkortin että funktiolohkon käytettävissä (kuva 13). Muuttujat linkitettiin sekä input- että output-puolella RS-232-korttiin EL6002 (kuva 14).



Kuva 12. Funktiolohko RS232PortControl ja sen sisällä oleva funktio SerialLineControl.

```
15
16  //RS232-yhteyden muisti
17  COMin_EL60xx     AT %I*   : KL6InData22B;
18  COMout_EL60xx    AT %Q*   : KL6OutData22B;
19  RxBuffer: ComBuffer;
20  TxBuffer: ComBuffer;
```

Kuva 13. Funktion tarvitsema muuttujalista globaalissa muuttujalistassa (GVL).



Kuva 14. Muuttujien linkitys kortille.

6.5.1 Datan vastaanottaminen

Logiikkaan luotiin aliohjelma, jolla vastaanotetaan dataa RS-232-väylää pitkin. Lohko nimettiin kuvaavasti RS232Receive. Aliohjelmaan lisättiin ReceiveString-tyyppinen funktiolohko, jolle annettiin nimi Receive (kuva 15).

```
VAR
    Receive: ReceiveString;
    receiveData: STRING;
    dataReceived: BOOL;
    receiveBusy: BOOL;
    receiveErrorID: ComError_t;
    receiveTimeout: BOOL;

    newData:R_TRIG;
    lastData : STRING;
```

Kuva 15. Funktion ReceiveString-muuttujat.

Suffix-kohtaan laitettiin ’$0D’, koska kontrollerilta tulevassa datassa käytettiin samaa merkkiä datan lopettamiseen. ’$0D’ vastaa ASCII-merkkiä ”Carriage Return”. RXBuffer, joka on datan vastaanottamiseen käytetty datatyyppi, liitettiin aikaisemmin luotuun saman muotoiseen datatyyppiin GVL:ssä, jotta funktio olisi yhteydessä EL6002-korttiin (kuva 16).

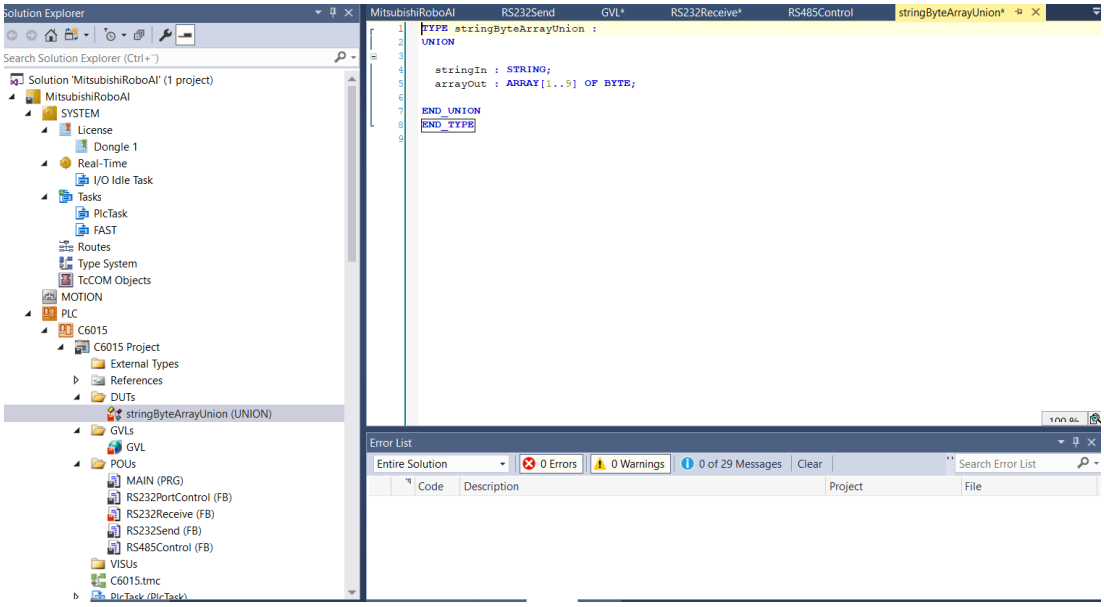
```
Receive(
    Suffix:= '$0D',
    Timeout:= T#1s,
    ReceivedString:= receiveData,
    RXbuffer:= GVL.RxBuffer,
    StringReceived=> dataReceived,
    Busy=> receiveBusy,
    Error=> receiveErrorID,
    RxTimeout=> receiveTimeout );

newData(CLK  := Receive.StringReceived );
IF newData.Q THEN
    lastData := receiveData;
END_IF
```

Kuva 16. Funktio Receive.

Dataa alkoi tämän jälkeen ilmestyä muuttujaan receiveData. Koska data on ASCII-koodattua, jouduttiin tekemään pieniä muunnosoperaatioita, jotta sitä voidaan käyttää ohjelmassa. Koska tarttujaa haluttiin ohjata kolmella arvolla väliltä 0-255, tarttujan ohjaamiseen käytettäväksi muuttujaksi valittiin yhdeksän merkin pituinen merkkijono- eli string-tyyppinen muuttuja. Merkkijono muodostuu kolmesta kolmen merkin pituisesta arvosta välillä 000-255, joista ensimmäinen ohjaa tarttujan sormien asentoa, toinen tarttujan liikkumanopeutta ja kolmas tarttujan käyttämää voimaa.

Koska data tuli yhtenä stringinä PLC:lle, se täytyi ensin hajottaa tavuiksi, jotta yksittäisiä merkkejä voidaan hyödyntää. Tämä toteutettiin luomalla dataunioni, joka yhdistää stringin 9 merkin pituiseen tavutaulukkoon (kuva 17).



Kuva 17. Dataunioni stringByteArrayUnion.

Käytännössä dataunioni ottaa stringin, joka tulee muuttujaan receiveData ja jakaa sen unionissa olevaan 9 merkin tavutaulukkoon. Tämä erottaa merkit toisistaan. Tämän jälkeen ASCII-koodatut merkit voidaan muuttaa PLC:ssä hyödyntämiskelpoisiin arvoihin. Koska ASCII-koodissa merkit 0-9 ovat alueella 48-58, muunnos tehtiin vähentämällä arvoista 48 (kuva 18). Tämän jälkeen merkit muunnettiin kolminumeroisiksi arvoiksi kertomalla ensimmäinen merkki sadalla ja toinen merkki kymmenellä ja lisäämällä ne yhteen yhdeksi muuttujaksi. Näin saatiin aikaan kolme muuttujaa, joiden arvot voivat vaihdella välillä 0-255, joita pystyttiin käyttämään suoraan tarttujan ohjaukseen.

```
23
24  //string rs232Data  kirjoitetaan unioniin byte arrayn kanssa jotta tavut saadaan erotettua toisistaan
25  stringToByte.stringIn := receiveData;
26
27  //Tarttujan aktivointi Mitsubishillä. General Output 8 pitää aktivoida kontrollerilla
28  //jotta tarttujaa voidaan käyttää.
29  IF GVL.input8 THEN
30  //ASCII-koodatussa datassa merkit 0-9 ovat alueella 48-59.
31  gripperControlArray[1] := stringToByte.arrayOut[1] - 48 ;
32  gripperControlArray[2] := stringToByte.arrayOut[2] - 48 ;
33  gripperControlArray[3] := stringToByte.arrayOut[3] - 48 ;
34  gripperControlArray[4] := stringToByte.arrayOut[4] - 48 ;
35  gripperControlArray[5] := stringToByte.arrayOut[5] - 48 ;
36  gripperControlArray[6] := stringToByte.arrayOut[6] - 48 ;
37  gripperControlArray[7] := stringToByte.arrayOut[7] - 48 ;
38  gripperControlArray[8] := stringToByte.arrayOut[8] - 48 ;
39  gripperControlArray[9] := stringToByte.arrayOut[9] - 48 ;
40
41  GVL.posData := gripperControlArray[1] *100 + gripperControlArray[2] *10 + gripperControlArray[3] ;
42  GVL.speedData := gripperControlArray[4] *100 + gripperControlArray[5] *10 + gripperControlArray[6] ;
43  GVL.forceData := gripperControlArray[7] *100 + gripperControlArray[8] *10 + gripperControlArray[9] ;
44
45  END_IF
46
```

Kuva 18. Muuttujien muunto.

6.5.2 Datan lähettäminen

Datan lähettämistä varten luotiin aliohjelma nimeltään RS232Send. Aliohjelmaan luotiin SendString-tyyppinen funktio, nimeltään send (kuva 19), joka lähettää string-tyyppistä dataa eli merkkijonoja RS232-väylää pitkin. Muuttujaan sendData (kuva 20) kirjoitettu merkkijono lähetetään jatkuvasti väylää pitkin tällä funktiolla.

```
1
2   send(
3       SendString:= sendData,
4       Busy=> ,
5       Error=> ,
6       TXbuffer:= GVL.TxBuffer
7   );
8
```

Kuva 19. Funktio send.

```
6   VAR
7   send : SendString ;
8   sendData : STRING ;
```

Kuva 20. Funktion send-muuttujat.

Kontrolleri ei hyväksynyt ihan minkä tahansa muotoista dataa. PLC:ltä lähetettävän datan ja kontrollerin vastaanottaman datan täytyi olla samanmuotoista ennen, kuin lähetys toimi. Datan lähetykseen valittiin kolmen merkkijonon mittainen datamuoto. Datan lähetyksessä täytyi huomioida, että merkkijonojen määrä on molemmilla puolilla sama, merkkijonot on erotettu pilkuilla ja ASCII-merkki ”$0D” on datan lopussa, jotta kontrolleri tietää, mihin data päättyy. Kun kontrolleri lukee kolmea merkkijonoa komennolla INPUT #1,C1$,C2$,C3$ niin PLC:n puolelta täytyy lähettää merkkijono ”Esimerkki1,Esimerkki2,Esimerkki3$0D” ja pilkuilla erotetut merkkijonot tulevat omiin muuttujiinsa robotin kontrollerille.

7 ROBOTIN TYÖKALUN KÄYTTÖ RS485-VÄYLÄSSÄ

7.1 ModBus RTU -kommunikaatio RS-485-väylän yli

Opinnäytetyössä käytetty Robotiq 2F-85 -tarttuja kommunikoi RS-485-väylän yli Modbus RTU -protokollalla. Tarttujan M12-liitintä ei saanut liitettyä suoraan PLC:n EL6021-sarjaliikenneväyläkorttiin, ja koska kaapelointia robotille ei oltu tässä vaiheessa vielä tehty, tarttuja liitettiin 3D-tulostusta hyödyntämällä valmistetun adapterin kautta (kuva 21).

Kuva 21. Tarttuja ja adapteri.

RS-485-väylä liitetään siten, että molempien laitteiden samannimiset liitännät kytketään yhteen. Toisin kuin esimerkiksi RS-232-väylässä, johtimia ei siis tarvitse kytkeä ristiin keskenään. Käyttöjännite tarttujalle otettiin 24 V jännitelähteeltä (kuva 22).

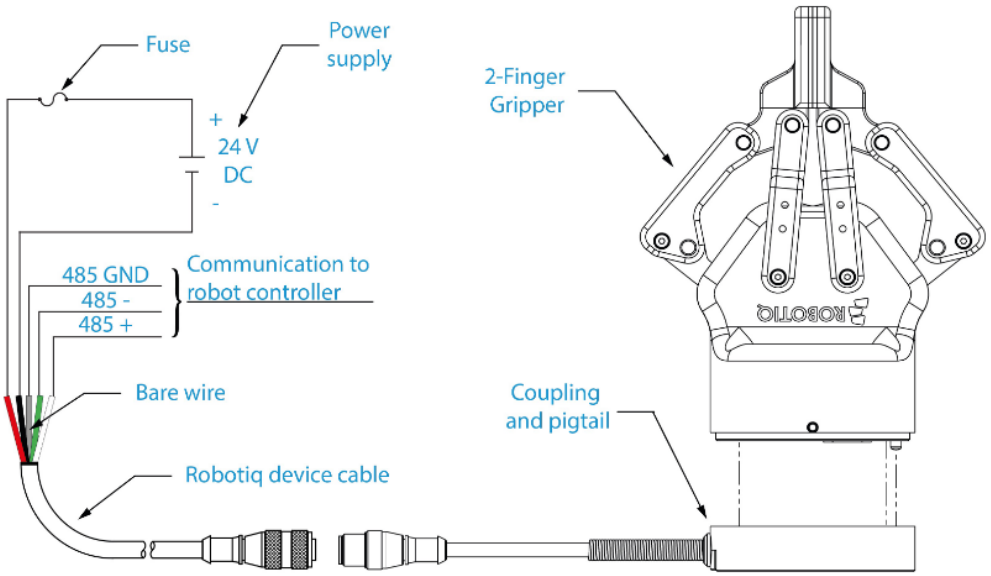
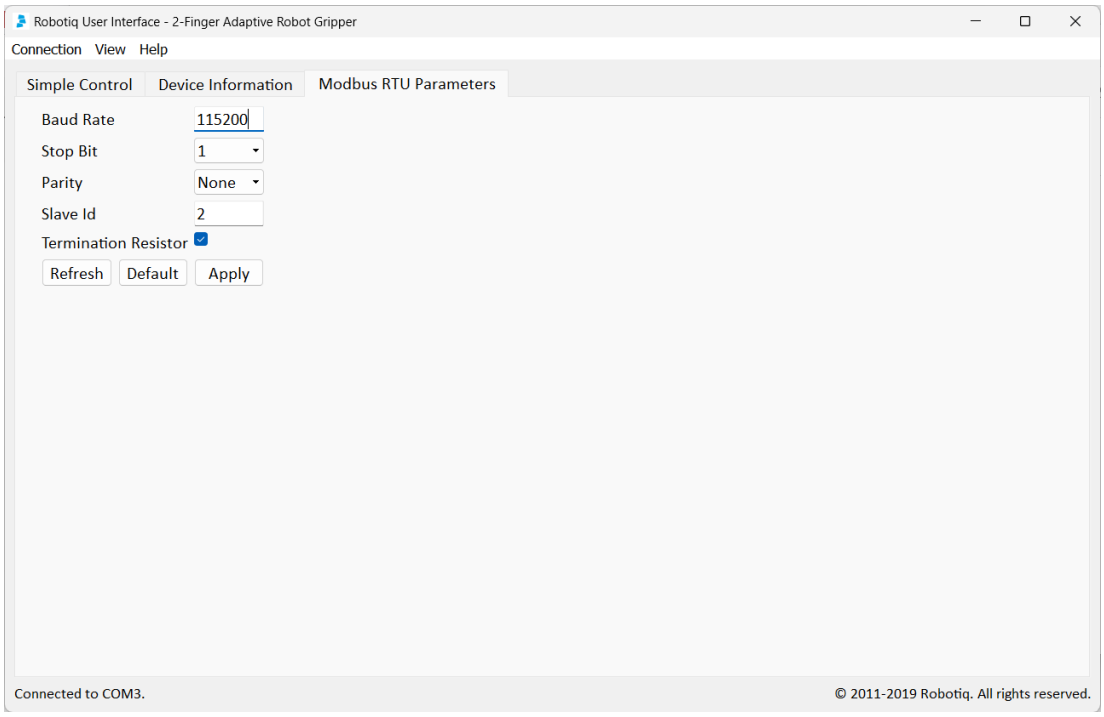


Fig. 3-9: Robotiq 2-Finger with pigtail cable and device cable wiring schematic.

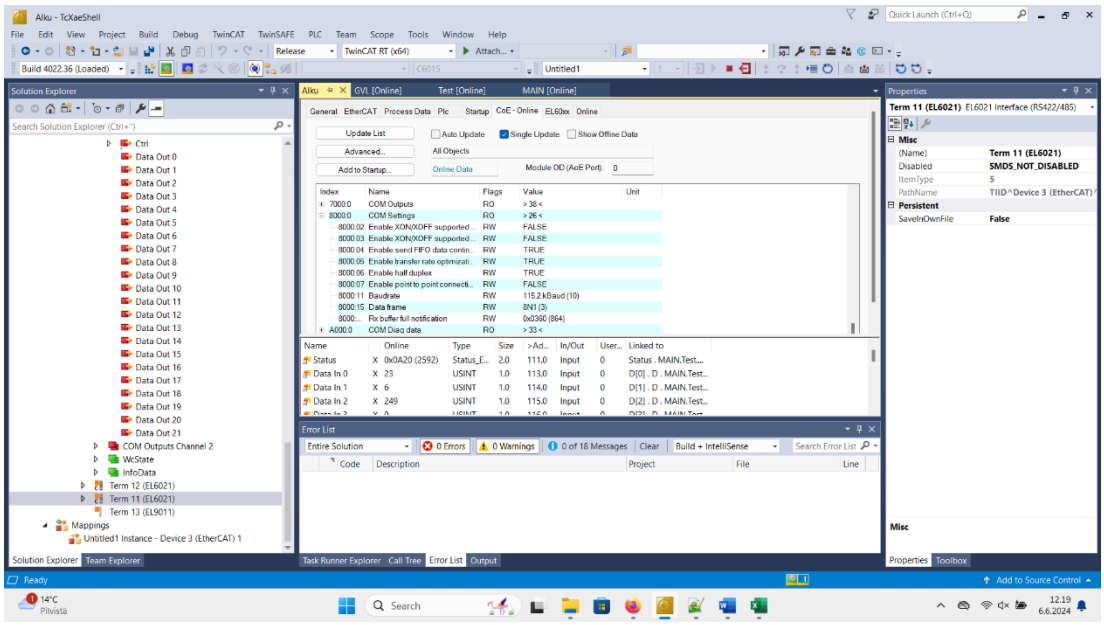
Kuva 22. Robotiq-tarttujan sähköiset liitännät (Robotiq, 2021, s. 26).

Kun tarttuja oli kytketty USB-RS485-adapterin kautta tietokoneeseen, saatiin Robotiqin ohjelmistoa käyttäen parametroitua tarttuja. Baud rate on datan siirtonopeus, jonka nopeudeksi valittiin suurin tuettu 115200 bps (bits per second). Stop-bitti ja parity-bitti määrittelevät data framen enkoodauksen. Slave Id on laitteen numero väylässä, ja sen pitää olla jokaisella laitteella uniikki. Enkoodaukseksi valittiin Robotiqin vakioenkoodaus, jossa ei ole pariteettibittiä ja 1 stop-bitti. Data frameksi muodostuu siis 8N1 (8 databittiä, N= None pariteettibitti, 1 stop-bitti (kuva 23).



Kuva 23. Robotiqin tarttujan parametrointi USB-yhteyden yli.

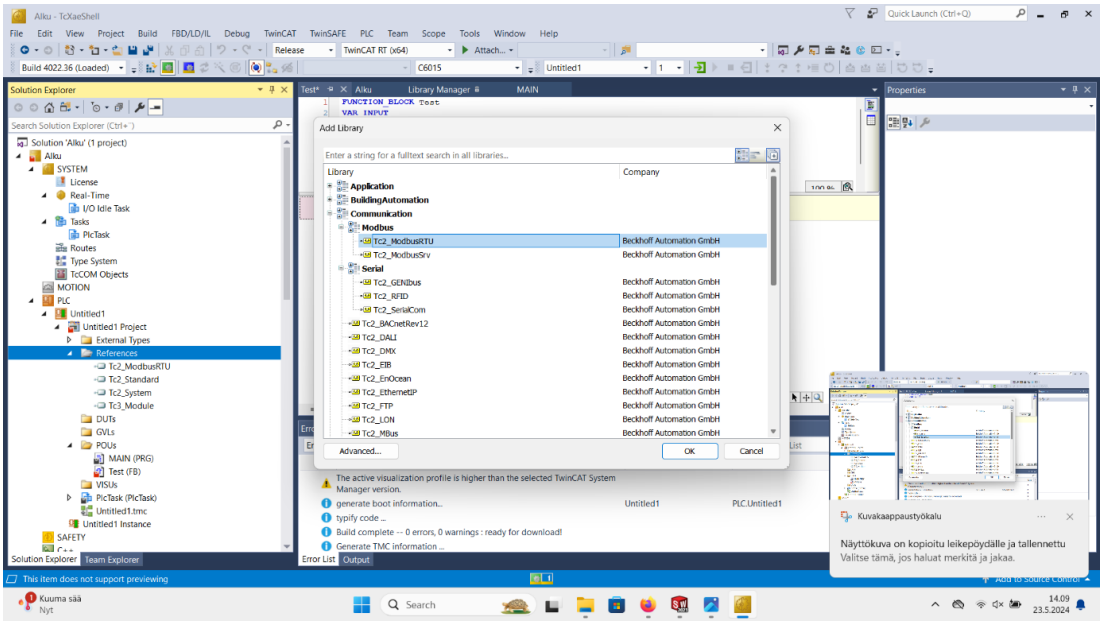
RS-485-väylää käytettäessä tärkeää on asettaa molempien toistensa kanssa keskustelevien laitteiden parametrit oikein. Koska tarttujalle oli tässä vaiheessa asetettu enkoodaukseksi 8N1, sama tieto asetettiin PLC:n puolella kohtaan Dataframe. Baud rateksi oli asetettu tarttujalle 115200 bps, joten se asetettiin PLC:lle samaksi (kuva 24).



Kuva 24. Beckhoffin PLC:n parametrointi Twincat-ohjelmassa.

EL6021-korttia on mahdollista käyttää joko RS422 tai RS485-väylänä. Jotta kortti toimisi RS485-väylänä, parametrin 8000:07 Enable point to point connection on oltava pois päältä. Kortin parametroinnin jälkeen siirryttiin tarttujan ohjaukseen tarvittavan ohjelman luomiseen.

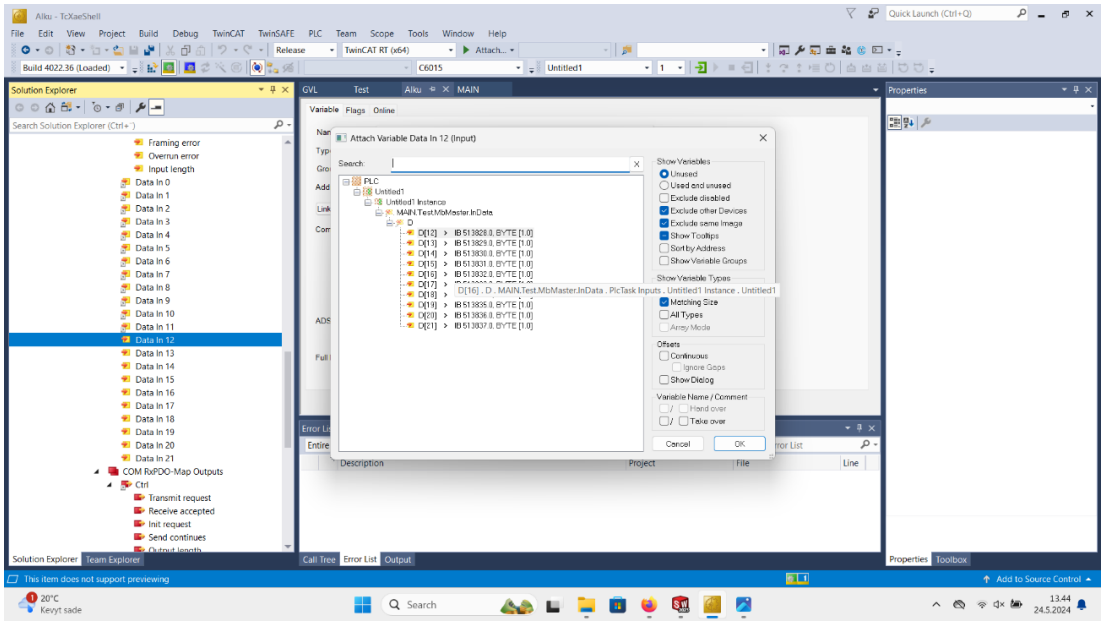
Twincat-ohjelmassa on valmiina kirjasto Modbus RTU -väylän ohjaukseen. Kirjasto Tc2_ModbusRTU tuotiin projektiin klikkaamalla hiiren toisella näppäimellä References-välilehteä ja valitsemalla Add Library. Sarjaväyläohjaukseen tarvittiin myös kirjasto Tc2_SerialCom, joka tuotiin projektiin samalla tavalla (kuva 25).



Kuva 25. Kirjaston tuominen Twincat-ohjelmaan.

Modbus RTU -kirjastosta löytyy funktio ModbusMasterV2_KL6x22B. Tällä funktiolla EL6021-kortti saatiin käyttäytymään Modbus-masterina. Master on Modbus-väylässä muita laitteita ohjaava laite. Funktio täytyi parametroida oikein, jotta sen kautta pystyy siirtämään dataa PLC:ltä tarttujalle. Olennainen funktion parametri on tarttujalle aikaisemmin asetettu Slave Id, joka asetettiin arvoon 2. Tämä arvo asetettiin funktion muuttujaan UnitID. Tämän lisäksi tulee tietää tarttujan rekisterien koko ja niiden osoite. Funktio nimettiin MbMaster-nimiseksi.

Jotta funktio toimisi EL6021-kortin kautta, funktion käyttämät muuttujat tarvitsi liittää ohjelmassa EL6021-kortin muuttujiin. Kun ModbusMasterV2_KL6x22B on olemassa ja projekti on aktivoitu kertaalleen, ohjelmaan ilmestyy muuttujat, jotka voidaan liittää korttiin. Terminaalin alta löytyy ComTxPDO-Map Inputs-välilehti, jonka alta löytyy Status-sana ja 22 datasanaa. Sanaa klikkaamalla toisella hiiren näppäimellä ja valitsemalla Change Link päästään yhdistämään sanat keskenään. Saman nimiset sanat liitettiin keskenään toisiinsa. Sama toimenpide tehtiin välilehdellä ComRxPDO-Map Outputs (kuva 26).



Kuva 26. Kortin muuttujien liittäminen funktion muuttujiin.

Tässä vaiheessa täytyi tietää, mitkä tarttujan rekisterit ovat kirjoitus- ja lukusuuntaan. Robotiqin spesifikaation mukaan kirjoitusrekisteri ja lukurekisteri koostuvat kolmesta sanasta, joissa on jokaisessa kaksi tavua (Robotiq, 2021, s. 31, 49, 54). Kirjoitusrekisteri on osoitteessa 1000 ja lukurekisteri osoitteessa 2000. Funktiolle MbMaster luotiin kuudesta tavusta koostuvat taulukot (array) tätä dataa varten (Kuva 27). PLC:n ja tarttujan välillä liikkuu siis molempiin suuntiin kuusi tavua dataa (Kuva 28).

MitsubishiRoboAI.C6015.MAIN.RS485Control

Expression	Type	Value	Prepared value	Address	Comm
DataRead	ARRAY [1..6] OF BYTE				Tarttuja
DataWrite	ARRAY [1..6] OF BYTE				
DataWrite[1]	BYTE	0			
DataWrite[2]	BYTE	1			
DataWrite[3]	BYTE	0			
DataWrite[4]	BYTE	0			
DataWrite[5]	BYTE	0			
DataWrite[6]	BYTE	0			

Kuva 27. Muuttujat RS-485-väylässä kulkevalle datalle.

Register	Robot Output / Functionalities	Robot Input / Status
Byte 0	ACTION REQUEST	GRIPPER STATUS
Byte 1	RESERVED	RESERVED
Byte 2	RESERVED	FAULT STATUS
Byte 3	POSITION REQUEST	POS REQUEST ECHO
Byte 4	SPEED	POSITION
Byte 5	FORCE	CURRENT
Byte 6 to 15	RESERVED	RESERVED

Kuva 28. Robotiq-tarttujan IO-rekisteri (Robotiq, 2021, s. 31).

Ohjelmaan tehtiin aluksi tarttujan initialisaatio eli alustus. Alustuksessa tarttujalle kirjoitetaan ensin pelkkiä nollia sisältävät taulukot, jotta tarttujan rekisterit ovat tyhjät. Tämän jälkeen bittiin rACT kirjoitetaan 1, jolla tarttuja aktivoituu (kuva 29).

4.4 Robot output registers & functionalities

Register: **ACTION REQUEST**
Address: **Byte 0**

Bits	7	6	5	4	3	2	1	0
Symbols	Reserved		**rARD**	**rATR**	**rGTO**	Reserved		**rACT**

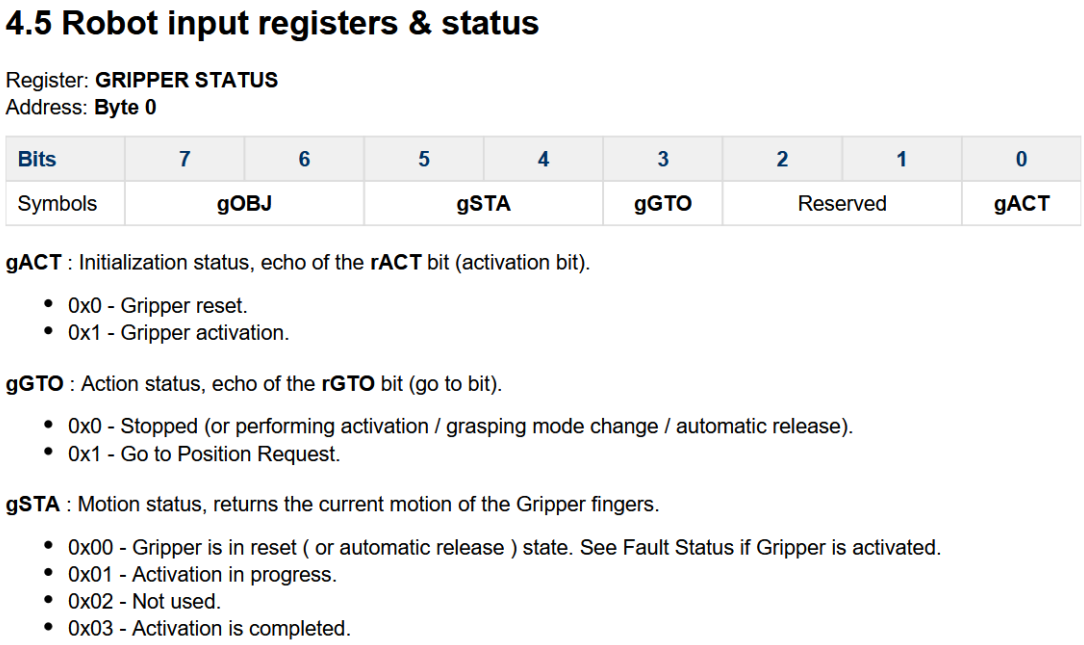
rACT: First action to be made prior to any other actions, **rACT** bit will initialize the Adaptive Gripper. Clear **rACT** to reset the Gripper and clear fault status.

- 0x0 - Deactivate Gripper.
- 0x1 - Activate Gripper (must stay on after activation routine is completed).

Kuva 29. Robotiq-tarttujan kirjoitusrekisterin tavu 0 (Robotiq, 2021, s. 32).

Aktivoinnin jälkeen tarttujalta luetaan tavusta 0 bitit 4 ja 5, jotka kertovat tarttujan tilan (kuva 30). Jos molemmat bitit ovat 1, on tarttujan aktivoituminen valmis. Rekisterin sanasta 1 on siis luettava bittijono 0011 0000 0000 0000,

joka vastaa desimaaliarvoa 12544. Tämän jälkeen tarttuja on valmis ohjattavaksi.



4.5 Robot input registers & status

Register: **GRIPPER STATUS**
Address: **Byte 0**

Bits	7	6	5	4	3	2	1	0
Symbols	**gOBJ**		**gSTA**		**gGTO**	Reserved		**gACT**

gACT : Initialization status, echo of the **rACT** bit (activation bit).

- 0x0 - Gripper reset.
- 0x1 - Gripper activation.

gGTO : Action status, echo of the **rGTO** bit (go to bit).

- 0x0 - Stopped (or performing activation / grasping mode change / automatic release).
- 0x1 - Go to Position Request.

gSTA : Motion status, returns the current motion of the Gripper fingers.

- 0x00 - Gripper is in reset (or automatic release) state. See Fault Status if Gripper is activated.
- 0x01 - Activation in progress.
- 0x02 - Not used.
- 0x03 - Activation is completed.

Kuva 30. Robotiq-tarttujan lukurekisterin tavu 0 (Robotiq, 2021, s. 36).

Tarttujan ohjaamiseen käytetään bittiä rGTO. Tarttujaa ohjattaessa myös aktivointibitin rACT täytyy pysyä päällä. Rekisterin sanaan 1 täytyy siis kirjoittaa bittijono 0000 1001 0000 0000, joka vastaa desimaalimuotoista lukua 2304. Rekisterin sanassa 2 kerrotaan paikkatieto, johon tarttujan halutaan liikkuvan. Koska tavu 3 ei ole käytössä, tarttujan paikkatieto annetaan käytännössä lukuarvolla 0-255, jossa 255 on täysin kiinni ja 0 on täysin auki. Rekisterin sanaan 3 kirjoitetaan tarttujan haluttu nopeus ja voima-anturin herkkyys. Ensimmäinen tavu on nopeusohjetta varten ja toinen tavu voima-anturin ohjetta varten. Nopeusohjearvo on 0-255, jossa 255 on nopein ja 0 hitain. Voima-anturille voidaan antaa tieto, kuinka voimakkaasti tarttujan halutaan puristavan. Voima-anturin arvo 0 vastaa 40 N voimaa ja 255 vastaa 100 N voimaa. Täydellä nopeudella ja suurimmalla voimalla rekisteriin täytyy siis kirjoittaa bittijono 1111 1111 1111 1111, joka vastaa kymmenjärjestelmälukua 65535. Koska samaan aikaan haluttiin lukea ja kirjoittaa dataa, oli ModbusMasterilla käytettävä funktiota ReadWriteRegs.

8 MELFA BASIC IV-OHJELMOINTIKIELI

Mitsubishin CR2B-574-robottikontrolleria ohjelmoidaan Melfa BASIC IV -ohjelmointikielellä. Kieli perustuu BASIC-ohjelmointikieleen, joka kehitettiin Yhdysvalloissa Dartmouthin yliopistossa vuonna 1964. (Bellis, 2019.)

Ongelmia ohjelmoinnissa tuotti se, että kieli on todella vanhakantaista. Uudemmissa ohjelmointikielissä on paljon sellaisia komentoja, joilla pystyy toteuttamaan helpommin monimutkaisia ohjelmia. Esimerkiksi kommunikaatioon käytetyt PRINT- ja INPUT-komennot eivät intuitiivisesti vastaa uudemmista kielistä löytyviä komentoja saman asian saavuttamiseksi.

Suurin osa ohjelmointiin tarvituista asetuksista täytyi asettaa kontrollerin parametreihin. Parametrit ovat globaali muuttujalista kontrollerin muistissa, jota BASIC-ohjelman komennot lukevat. Jos esimerkiksi tiedonsiirron asetuksia täytyy muuttaa, sitä ei voi tehdä suoraan ohjelmassa vaan muuttamalla parametrejä.

9 ESIMERKKI LAITTEISTON KÄYTÖSTÄ JA OHJELMOINNISTA

9.1 Digitaalisten signaalien käyttö

Digitaaliset signaalit on kytketty robotille paikkoihin 8-11. Jos robotin ohjelmassa kytketään General Purpose Output 8 päälle, tulee logiikan EL1004 -kortilla input 1 aktiiviseksi. Toiseen suuntaan kytkentä on sama eli, jos logiikalla kytketään EL2004-kortin output 1 päälle, tulee robotin kontrollerin General Purpose Input 8 aktiiviseksi.

Logiikan output 1 eli kontrollerin input 8 on kytketty antamaan tieto siitä, kun tarttujan liike on valmis. Tällä voidaan varmistaa, että tarttuja on tehnyt liikkeen loppuun asti ennen, kuin ohjelma jatkuu.

9.2 RS232-yhteyden käyttö tarttujan ohjaamiseen

Logiikan kortti EL6021 on kytketty Robotiqin tarttujan speksien mukaisesti RS485-masteriksi ja kaapeli on viety tarttujalle robotin ulkoisesti asennetun kaapelikourun kautta. Robotin käsivarressa olevaan M12-liittimeen voidaan näin siis kytkeä suoraan Robotiqin tarttuja muuttamatta kytkentää. Tarttujan kommunikaatioparametrien tulee olla Baud rate: 115200, Slave Id: 2, Stop bit: 1, Parity: None. Tarttujalle voidaan asettaa nämä parametrit Robotiq User Interface-ohjelmistolla.

Kontrollerin ohjelmaan voidaan kirjoittaa koodi, jolla tarttujaa ohjataan. Ohjaukseen käytetään yhdeksän numeron mittaista komentoa, jolla kerrotaan tarttujan haluttu paikkapiste, tarttujan nopeus ja tarttujan voima-anturin herkkyys. Komennon muodostus on selitetty tarkemmin liitteessä 2.

9.3 Esimerkki tarttujan ohjaamisesta

Tarttujaa voitaisiin ohjata esimerkiksi seuraavanlaisella koodilla:

```
100 OPEN ”COM2:” AS #1
110 DLY 0.05
120 PRINT #1,”255255255”
130 CLOSE
140 WAIT M_IN(8)=1
```

Tällä koodilla tarttuja ajetaan kiinni mahdollisimman nopeasti ja mahdollisimman voimakkaasti. Tarttujan avaamiseen voitaisiin kirjoittaa esimerkiksi komento ”000255255”. Wait-komennolla odotetaan, että tarttujalta on tullut tieto, että tarttujan liike on valmis.

Open-komennon COM-muuttujassa kerrotaan, mitä kontrollerin kommunikaatioväylistä käytetään. Komennossa itsessään ei voida määrittää väylien asetuksia, vaan ne täytyy määrittää erikseen kontrollerin parametreihin. Kontrollerin COMDEV-parametrilla määritetään, mitä väyliä Open-komennolla on mahdollista käyttää (Mitsubishi, 2009, s. 353). COMDEV-parametriin asetettiin ”RS232, OPT11”, jotta kontrollerin etupuolella oleva vapaaksi jäävä vakioväyläportti on määritelty COM1 ja käytetty laajennusväylä on COM2. Koska väylä on liitetty kontrolleriin asennettuun väylälaajennuskorttiin, on väyläksi asetettu COM2.

Logiikalla voidaan myös kirjoittaa muuttujia kontrollerille. Logiikan ohjelmassa RS232Send määritellään, mitä kontrollerille kirjoitetaan. Datatyyppien täytyy vastata sekä logiikan että kontrollerin kanssa muodoltaan toisiaan, jotta kontrolleri hyväksyy datan.

Logiikalta lähtevän datan pitää olla muodossa ”X,Y,Z$0D”, jossa X, Y ja Z ovat eri muistipaikkoihin kirjoitettavat muuttujat ja $0D on RS232-väylän lopetusmerkki. Esimerkiksi logiikalta voidaan kirjoittaa merkkijono ”100,100,SERIAL LINE OK$0D”, jolloin lähetetään kolme merkkijonoa kolmeen eri muistipaikkaan kontrollerille.

Kontrollerilla vastaanotettava datatyyppi määritellään INPUT-komennossa. String-muotoisia muuttujia käytettäessä täytyy käyttää muistialuetta C (character string). Muuttuja muodostuu muistialueesta C, muistipaikasta (numero) ja lopetusmerkistä $. Muuttujien määrän pitää vastata sekä logiikalla että kontrollerilla toisiaan. Robotin kontrolleri saatiin lukemaan dataa RS232-väylästä seuraavanlaiseslla koodilla:

```
100 OPEN ”COM2:” AS #1
110 DLY 0.05
120 INPUT #1,C1$,C2$,C3$
130 CLOSE
140 IF C3$=”SERIAL LINE OK” THEN M_OUT(8)=1 ELSE
M_OUT(8)=0
```

INPUT-komennolla ohjelma odottaa, kunnes logiikalta on tullut vastaus. Jos ohjelmassa halutaan käyttää numerotietoa, voidaan se lähettää kontrollerille string-muodossa ja muuttaa integeriksi ohjelmassa seuraavalla koodilla:

```
150 M1=VAL(C1$)
```

Tällöin muistipaikassa C1 oleva numerotieto tallennetaan muuttujaan M1. Tämän jälkeen muuttujaa M1 voidaan käyttää ohjelmassa matemaattisiin operaatioihin, joihin tyyppiä string oleva muuttuja ei kelpaa. Esimerkiksi kahden numeron vertailu ei onnistu string-tyyppisellä muuttujalla:

```
160 IF M1 > 100 THEN M_OUT(9)=1 ELSE M_OUT(9) = 0
```

Lopuksi voidaan tyhjentää muuttujat, jos halutaan, että niihin ei jää ”vanhaa” dataa kun ohjelmakierto alkaa alusta.

```
170 M1= 0
180 C1$=“blank”
190 C2$=”blank”
200 C3$ =“blank”
```

10 YHTEENVETO JA JOHTOPÄÄTÖKSET

Työssä toteutettiin järjestelmä, jolla Robotiq 2F-85-älytarttuja saadaan toimimaan vanhan Mitsubishi RV-6S-robotin kanssa (kuva 31). Järjestelmässä hyödynnettiin RS-232- ja RS-485-kommunikaatioprotokollia sekä Beckhoff C6015 -ohjelmoitavaa logiikkaa. Robotin kontrolleri on yhdistetty logiikkaan RS-232-protokollaa hyödyntäen ja tarttuja on yhdistetty logiikkaan RS-485-protokollaa hyödyntäen.

Kuva 31. Robotti käyttövalmiina tarttujan kanssa.

Työssä todettiin, että erilaisia kommunikaatioprotokollia käyttämällä on mahdollista kytkeä monenlaisia eri laitteita kommunikoimaan toistensa kanssa laitteen valmistajasta, valmistusvuodesta ja laitteen tyypistä riippumatta. Vaikka kommunikaatioprotokollissa ei ole laitevalmistajien välillä yhtenäistä standardia käytössä, on mahdollista konfiguroida eri valmistajien laitteet siten, että kommunikaatio onnistuu.

Työtä tehdessä huomattiin, että RS-232- ja RS-485-protokollilla on paljon samankaltaisia ominaisuuksia, joista olennaisimmat ovat datan enkoodauksen ja tiedonsiirtonopeuden määrittely. Kun kaksi laitetta kytkettiin toisiinsa määrittelemällä kommunikaatioasetukset samoiksi molemmissa päissä, kommunikaatio toimi moitteettomasti.

Opinnäytetyö onnistui tavoitteiden mukaisesti ja laitteiston dokumentaatiosta tuli riittävä jatkokehitystä varten. Tulevaisuudessa robottisolua voisi vielä parantaa esimerkiksi liittämällä älykamera osaksi laitteistoa, jolloin robotin ohjauksessa voitaisiin hyödyntää konenäköä. Demonstraatioissa voitaisiin hyödyntää laitteistoon kytkettyä kosketusnäyttöä.

LÄHTEET

Advantech. (2018). RS-232 Connections That Work: Connecting Devices or Converters. Haettu 18.5.2025 osoitteesta https://www.advantech.com/en/resources/white-papers/e7e269e7-78b9-4848-96d1-aa7b6acca0d4

Bellis, M. (2019). The history of the BASIC programming language. Haettu 18.5.2025 osoitteesta https://www.thoughtco.com/history-basic-programming-language-1991662

Bies, L. (2021). Mitsubishi PLC cable layouts. Haettu 18.5.2025 osoitteesta https://www.lammertbies.nl/comm/cable/plc-mitsubishi

Dawoud, D & Dawoud, P. (2020). Serial Communication Protocols and Standards RS232/485, UART/USART, SPI, USB, INSTEON, Wi-Fi and WiMAX. River Publishers.

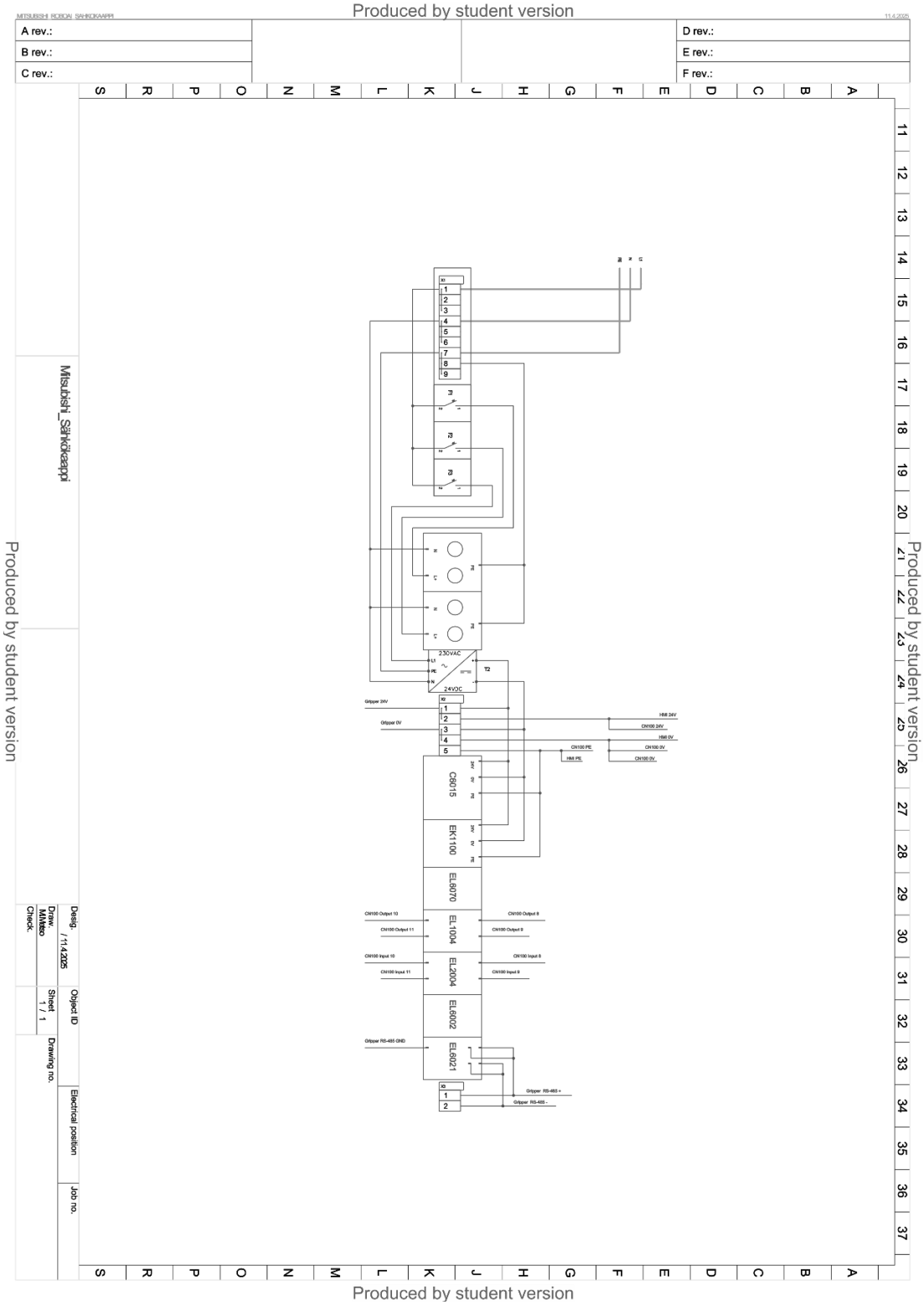
Mitsubishi. (2009). Industrial Robot CR1/CR2/CR3/CR4/CR7/CR8/CR9 Controller Instruction Manual. https://dl.mitsubishielectric.com/dl/fa/document/manual/robot/bfp-a5992/bfp-a5992p.pdf

Reynders, D, Mackay, S & Wright, E. (2005). Practical industrial data communications: best practice techniques. Newnes.

Robotiq. (2021). Robotiq 2F-85 & 2F-140 for Cobots - Instruction Manual. https://assets.robotiq.com/website-assets/support_documents/document/2F-85_2F-140_General_PDF_20210623.pdf

VLSIFacts. (2023). ASCII Code. Haettu 18.5.2025 osoitteesta https://www.vlsifacts.com/ascii-code/

LIITE 1: SÄHKÖPIIRUSTUKSET

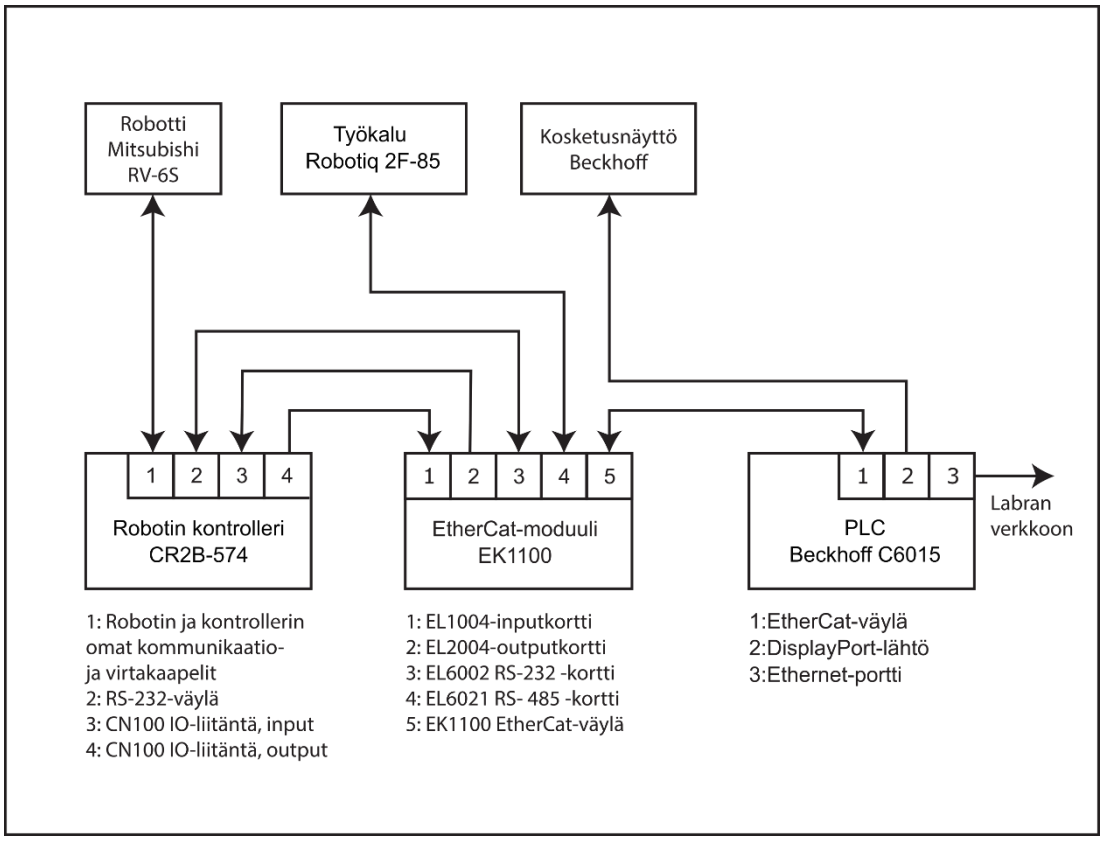


LIITE 2: LAITTEISTON KÄYTTÖOHJE

RoboAI Mitsubishi-robottisolun tarttujan käyttö

Laitteiston kuvaus

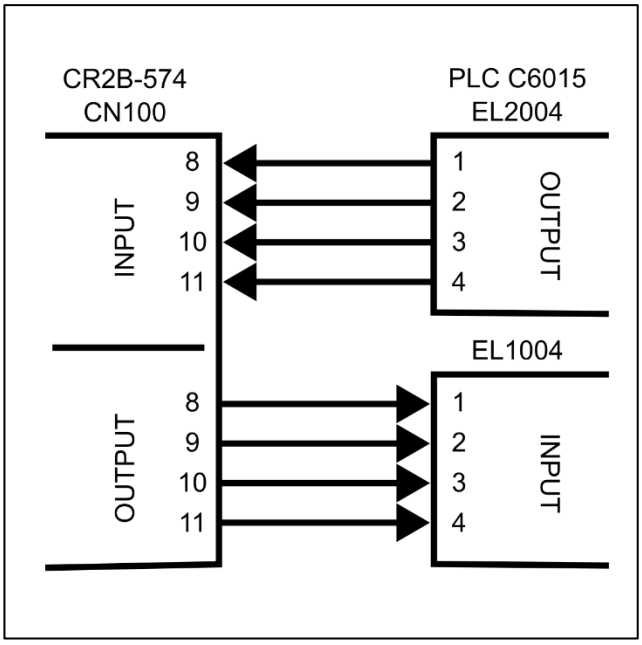
Laitteistoon kuuluu RV-6S-robotin ja sen kontrollerin CR2B-574 lisäksi Beckhoff C6015-ohjelmoitava logiikka, joka ohjaa RS-485-väylään liitettyä Robotiq-tarttujaa. Sen lisäksi kontrollerin ja logiikan välille on liitetty 8 digitaalista IO-signaalia sekä RS-232-väylä niiden keskinäistä kommunikaatiota varten.



Kuva 1. Kaavio laitteiston osista.

Kontrollerin ja Beckhoff PLC:n välinen kommunikaatio

Kontrollerin CN100-liittimen inputit 8-11 ja outputit 8-11 ovat liitetty johtimilla logiikan input- ja output-kortteihin suoraan. Esimerkiksi, kun kontrollerilta kytketään output 8 päälle, tulee logiikan input 1 aktiiviseksi ja käänteisesti, kun PLC:n output 1 kytketään päälle, tulee kontrollerin input 8 aktiiviseksi.



Kuva 2. Kontrollerin ja logiikan väliset digitaaliset signaalit.

RS-232-väylää käytettäessä kontrollerilla käytetään COM2-porttia, johon väylä on liitetty. BASICkomento PRINT lähettää väylän kautta PLC:lle merkkijonon. INPUT-komento lukee merkkijonon, jota lähetetään PLC:ltä.

Tarttujan aktivoiminen BASIC-koodissa

Tarttujan aktivoimiseksi tarvitsee ajaa seuraavanlainen koodi Mitsubishin kontrollerilla. OPENkomennolla avataan RS-232-väylä ja CLOSE- komennolla suljetaan se. Seuraavanlainen koodi aktivoi tarttujan:

```
100 OPEN ”COM2:” AS #1
110 DLY 0.05
120 INPUT #1,C1$,C2$,C3$
130 CLOSE #1
140 IF C3$=”SERIAL LINE OK” THEN M_OUT(8)=1 ELSE
M_OUT(8)=0
```

INPUT-komennolla ohjelma odottaa, kunnes logiikalta on tullut vastaus. PLC:lle on luotu koodi, joka lähettää jatkuvasti kolme merkkijonoa kontrollerin muistipaikkohin C1, C2 ja C3 joista kolmas on ”SERIAL LINE OK”. Jos kolmanteen paikkaan tulee tämä merkkijono, sarjaväylä toimii ja asetetaan kontrollerin output 8 päälle jolloin tarttuja aktivoi itsensä. Aktivoituessaan tarttuja sulkee ja avaa itsensä kertaalleen.

Jos tarttuja ei aktivoidu, tarkasta että logiikalle on kytketty jännite.

Tarttujan ohjaus BASIC-koodilla

PRINT-komennolla voidaan lähettää tarttujalle paikkatieto, nopeusohje ja tarttujan voima-anturin herkkyysasetus. Tämä tieto on koodattu yhdeksi 9-kirjaimiseksi merkkijonoksi, joka koostuu kolmesta kolmen numeron sarjasta välillä 000-255.

Ohjaussanan muodostus

	Paikkatieto	Nopeustieto	Voima-anturi
Tarttuja auki	000	XXX	XXX
Tarttuja kiinni	255	XXX	XXX
Tarttujan nopeus 13 mm/s*	XXX	000	XXX
Tarttujan nopeus 100 mm/s*	XXX	255	XXX
Voima 30N**	XXX	XXX	000
Voima 100N**	XXX	XXX	255

*Tarttujan nopeus kasvaa noin 0,34 mm/s per lukema yli 000 **Voima-anturin herkkyys kasvaa noin 0,27 N per lukema yli 000

Huomio! Ohjaussanan tulee aina olla 9-merkkinen.

Esimerkki tarttujan ohjaamisesta

```
200 OPEN ”COM2:” AS #1
210 DLY 0.05
220 PRINT #1,”255255255”
230 CLOSE #1
240 WAIT M_IN(8)=1
```

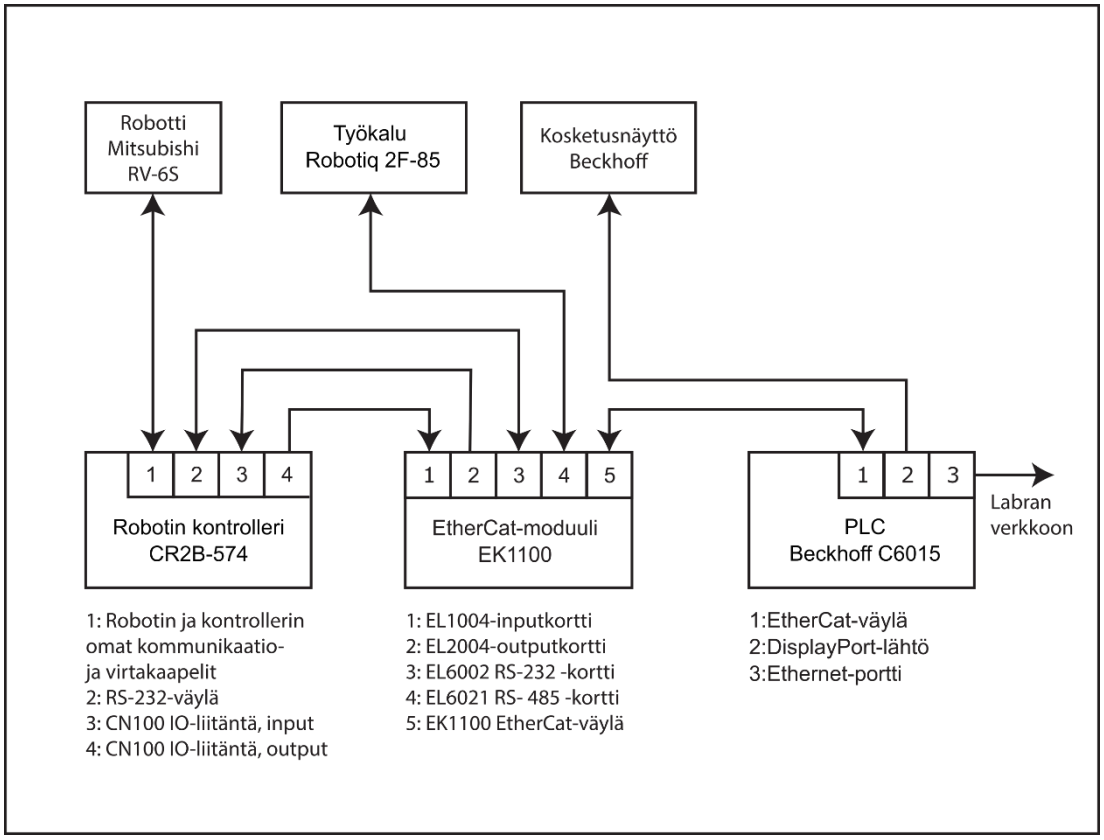
Ohjaussanalla ”255255255” tarttuja ajetaan kiinni mahdollisimman nopeasti ja mahdollisimman voimakkaasti. Tarttujan avaamiseen voitaisiin kirjoittaa esimerkiksi ohjaussana ”000255255”. PLC:lle on luotu toiminto, jossa kontrollerin input 8 tulee aktiiviseksi kun tarttuja on suorittanut liikkeensä. Käyttämällä WAIT-komentoa odottamaan tämän signaalin aktivoitumista robotti ei lähde liikkeelle ennen kuin liike on valmis.

Muista, että tarttuja tulee aktivoida edellä mainitulla tavalla ennen kuin sitä voidaan ohjata.

Usage of a Robotiq gripper with the Mitsubishi RV-6S

Hardware overview

The equipment used in the robot cell consists of the robot RV-6S and its controller CR2B-574, a programmable logic controller (PLC) Beckhoff C6015, the IO module EK1100 for the PLC and a human-machine interface (HMI) panel also connected to the PLC. The robot controller and the PLC are connected by a RS-232 serial bus and 8 digital IO connections. The gripper is connected through a RS-485 serial bus to the PLC.

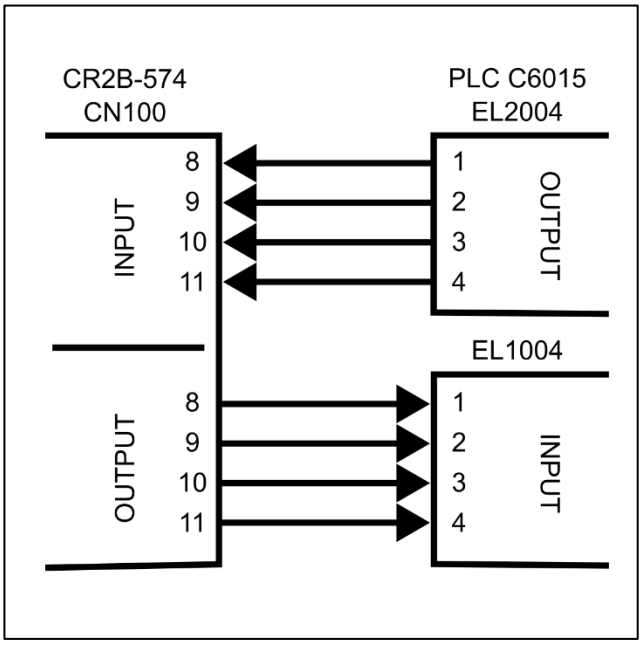


Picture 1. Hardware overview

Communication between the PLC and the controller

The robot controller's CN100-port inputs 8-11 and outputs 8-11 are wired to the PLC's digital IO cards EL1004 and EL2004. For example, when you set

the output 8 on the controller, the PLC's input 1 becomes active. Inversely, when you set the output 1 on the PLC, the controller's input 8 becomes active.



Picture 2. The digital signals connected between the PLC and the controller.

When using the RS232 serial bus to communicate, the serial port COM2 is to be used. When programming the robot, the BASIC command PRINT sends data from the controller to the PLC and the command INPUT receives data from the PLC.

Activating the gripper in BASIC code

To activate the gripper, the digital output 8 must activated on the controller. The OPEN command starts communication in the serial bus and the CLOSE command ends it. The following code is used to activate the gripper:

```
100 OPEN "COM2:" AS #1
110 DLY 0.05
120 INPUT #1,C1$,C2$,C3$
130 CLOSE #1 140 IF C3$="SERIAL LINE OK" THEN M_OUT(8)=1
ELSE M_OUT(8)=0
```

The INPUT command waits until the PLC has sent a response. The PLC is designed to continuously send three string type messages to the controller's memory addresses C1, C2 and C3, with the third one being "SERIAL LINE OK". If this message is received, the output 8 is set and the gripper becomes active. While activating, the gripper will close itself and then open itself once.

If the controller doesn't activate when running this code, check that the PLC is running.

Controlling the gripper in BASIC

With the PRINT command it is possible to send a control word to the gripper that requests the gripper to move into a certain position and the speed and force to be used. The control word should always be 9 characters long and is formed in the following way:

Control word formation

	Position	Speed	Force
Open gripper	000	XXX	XXX
Close gripper	255	XXX	XXX
Gripper speed 13 mm/s*	XXX	000	XXX
Gripper speed 100 mm/s*	XXX	255	XXX
Force 30N**	XXX	XXX	000
Force 100N**	XXX	XXX	255

*The gripper's speed increases by approximately 0,34 mm/s per count over 000 **Gripper force increases by approximately 0,27 N per count over 000

Remember that the control word should always be exactly 9 characters long.

Example of gripper control

```
200 OPEN "COM2:" AS #1
210 DLY 0.05
220 PRINT #1,"255255255"
230 CLOSE #1
240 WAIT M_IN(8)=1
```

In this example, the control word "255255255" is used to close the gripper as fast as possible with full strength. To open the gripper with the same speed and force, the control word "000255255" could be used. The PLC is configured to activate input 8 when the gripper has completed its task so the WAIT command can be used here to stop the program until the motion is finished.

Remember that the gripper needs to be activated before it can be controlled.