BÀI THÍ NGHIỆM 4
GIAO TIẾP ROBOT VÀ MÁY TÍNH

1. MỤC ĐÍCH THÍ NGHIỆM

Trong bài thí nghiệm này, sinh viên sẽ học cách lập trình chương trình máy tính giao tiếp với robot thông qua Ethernet.

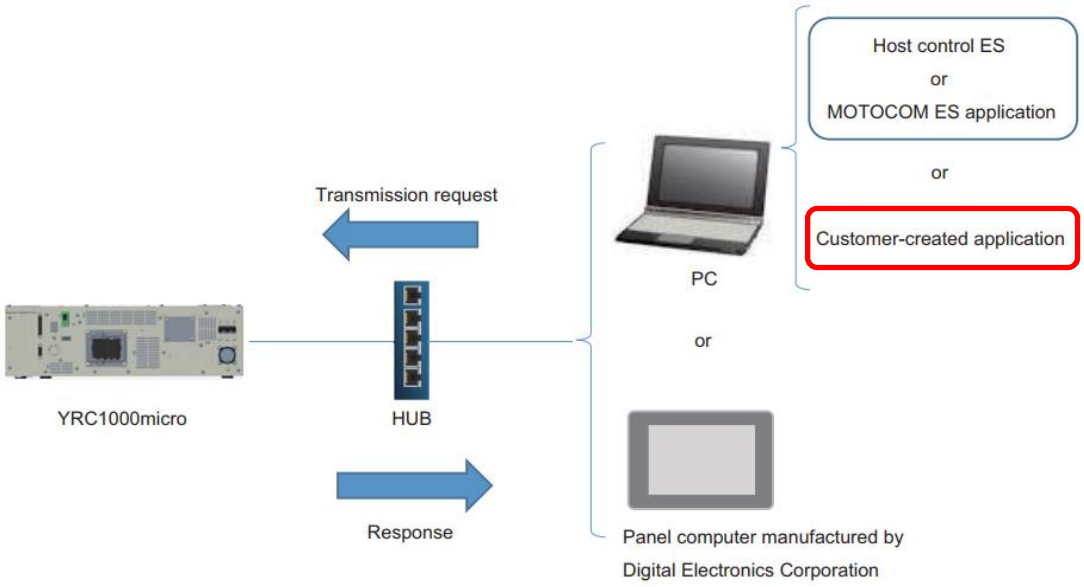
Sau khi thực hiện bài thí nghiệm, sinh viên có thể:

- Hiểu giao thức High-Speed Ethernet Server của bộ điều khiển robot YRC1000micro.
- Biết lập trình giao diện máy tính dùng nền tảng Qt C++ để điều khiển robot và tương tác với người dùng.

2. MÔ TẢ THIẾT BỊ

2.1. Cấu hình phần cứng

High-Speed Ethernet Server là một giao thức lớp ứng dụng trên nền UDP/IP cho phép thực hiện kết nối, điều khiển và truyền nhận dự liệu giữa một thiết bị có kết nối Ethernet với bộ điều khiển YRC1000micro. Trong đó, bộ điều khiển đóng vai trò là server, các thiết bị ngoài (PC) đóng vai trò client. Với mỗi yêu cầu gửi đến từ client, server sẽ xử lý và gửi trả phản hồi tương ứng.



Có hai cách để lập trình ứng dụng máy tính sử dụng giao thức High-speed Ethernet Server. Một là sử dụng thư viện MOTOCOM của nhà sản xuất, trong đó cung cấp một API để người dùng xây dựng lớp ứng dụng bên ngoài. Hai là tự xây dựng thư viện giao tiếp dựa trên giao thức được nhà sản xuất công bố, từ đó viết lớp ứng dụng. Ta sẽ chọn cách thứ hai để hiểu rõ và làm chủ việc lập trình giao tiếp.

2.2. Thông tin gói dữ liệu

Việc trao đổi thông tin được thực hiện bằng các gói dữ liệu (data frame). Gói dữ liệu truyền và nhận gồm header (32 byte) và data part (có giá trị thay đổi tùy vào câu lệnh, tối đa 479 byte)

4 Byte

Type	Byte 0	Byte 1	Byte 2	Byte 3
Identifier	Fixed character strings for identification (YERC)			
Data size	Header part size (fixed to 0x20)		Data part size (variable value)	
Reserve 1 / processing division	Reserve 1 (fixed to "3")	Processing division	ACK	Request ID
Block No.				
Reserve 2	Reserve2 (fixed to "99999999")			
Sub-header	Command No.		Instance	
	Attribute	Service (when requested)	Padding	
Data division	Data division (variable:479Byte at maximum)			

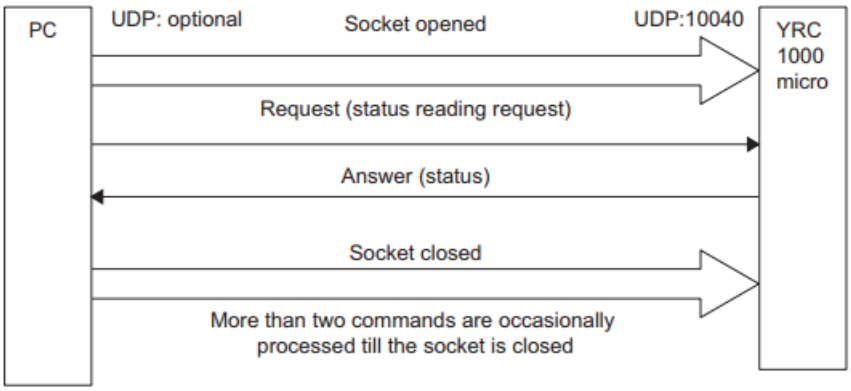
Header part (fixed to 32Byte)

Mô tả chi tiết cho các thành phần trong gói dữ liệu truyền nhận:

Item		Data size	Settings
Identifier		4Byte	Fixed to "YERC"
Header part size		2Byte	Size of header part (fixed to 0x20)
Data part size		2Byte	Size of data part (variable)
Reserve 1		1Byte	Fixed to "3"
Processing division		1Byte	1: robot control 2: file control
ACK		1Byte	0: Request 1: Other than request
Request ID		1Byte	Identifying ID for command session (increment this ID every time the client side outputs a new command. In reply to this, server side answers the received value.)
Block No.		4Byte	Request: 0 Answer: add 0x8000_0000 to the last packet. Data transmission other than above: add 1 (max: 0x7fff_ffff)
Reserve 2		8Byte	Fixed to "99999999"
Sub-header (request)	Command No.	2Byte	Execute processing by this command. (conforms to "Class" of CIP communication protocol)
	Instance	2Byte	Define SECTION to execute a command. (conforms to "Instance" of CIP communication protocol)
	Attribute	1Byte	Define SUB SECTION for executing a command. Attribute: (conforms to "Attribute" of CIP communication protocol)
	Service (request)	1Byte	Define data accessing method.
Sub-header (answer)	Service (answer)	1Byte	Add 0x80 to service (request).
	Status	1Byte	0x00: normal reply 0x1f: abnormal reply (size of added status: 1 or 2) Other than 0x1f: abnormal reply (size of added status: 0) Refer to *chapter 3.4.1 "Status Code"*
	Added status size	1Byte	Size of added status (0: not specified / 1: 1 WORD data / 2: 2 WORD data)
	Added status	2Byte	Error code specified by added status size For details, refer to *chapter 3.4.2 "Added Status Code"*
Padding		Variable	Reserve area

2.3. Một số ví dụ data frame truyền nhận

Đọc thông tin từ Robot:



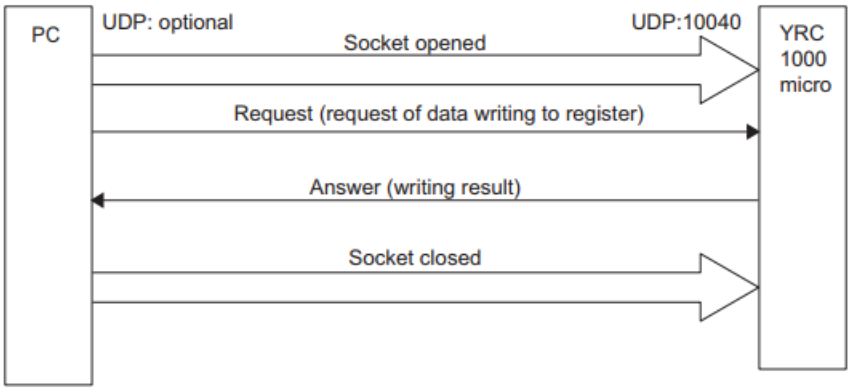
Request <Format>

"YERC"				Identifier			
0x0020		0x0000		Header part size		Data part size	
0x03	0x01	0x00	0x00	Reserve 1	Processing division	ACK	Request ID
0x0000_0000				Block No.			
"99999999"				Reserve 2			
0x0072		0x0001		Command No.		Instance	
0x00	0x01	0x0000		Attribute	Service	Padding	

Answer <Format>

"YERC"				Identifier			
0x0020		0x0000		Header part size		Data part size	
0x03	0x01	0x01	0x00	Reserve 1	Processing division	ACK	Request ID
0x8000_0000				Block No.			
"99999999"				Reserve 2			
0x81	0x00	0x00	0x00	Service	Status	Added status size	Padding
0x0000		0x0000		Added status		Padding	
Status data 1				Reading value 1			
Status data 2				Reading value 2			

Ghi thông tin xuống robot:



Request <Format>

"YERC"				Identifier			
0x0020		0x0002		Header part size		Data part size	
0x03	0x01	0x00	0x01	Reserve 1	Processing division	ACK	Request ID
0x0000_0000				Block No.			
'99999999'				Reserve 2			
0x0079		Register No.		Command No.		Instance	
0x00	0x02	0x0000		Attribute	Service	Padding	
Register data				Writing value			

Answer <Format>

'YERC'				Identifier			
0x0020		0x0000		Header part size		Data part size	
0x03	0x01	0x01	0x01	Reserve 1	Processing division	ACK	Request ID
0x8000_0000				Block No.			
'99999999'				Reserve 2			
0x82	0x00	0x00	0x00	Service	Status	Added status size	Padding
0x0000		0x0000		Added status		Padding	

Trong phần header các câu lệnh được phân biệt với nhau bằng các trường chính Command No, Instance, Attribute, Service. Việc thay đổi các trường này sẽ thay đổi câu lệnh và phản hồi từ robot.

Các trường quan trọng trong sending data frame:

Tên trường	**Ý nghĩa**
Command No	Quy định mã lệnh điều khiển
Instance và Attribute	Quy định giá trị cần thực thi
Service	Quy định phương thức truy cập thông tin
Data size	Độ lớn của trường data
Data division	Data gửi lên cho bộ robot controller, tối đa 479 Byte

Sau khi gửi thành công một câu lệnh, ta sẽ nhận được một phản hồi từ robot, dựa trên phản hồi này ta có thể biết việc gửi lệnh có gặp lỗi phát sinh và mã lỗi.

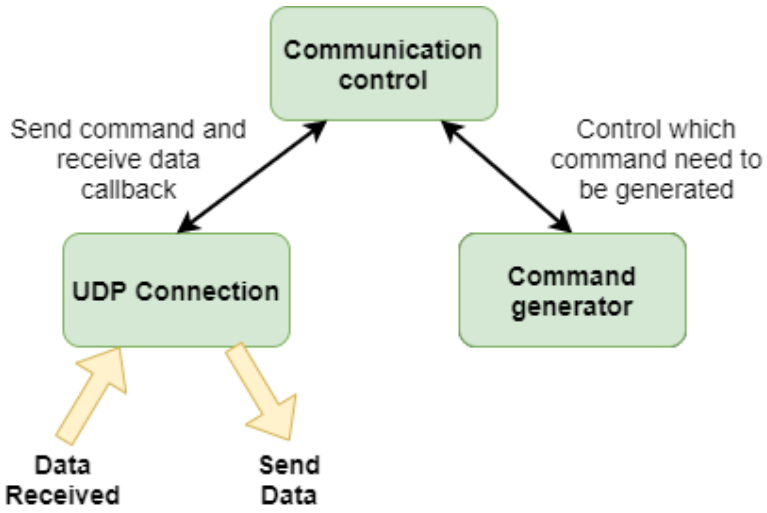
Các trường quan trọng trong receive data frame:

Tên trường	**Ý nghĩa**
Status	Trường báo lỗi, nếu không lỗi sẽ trả về 0x00
Request ID	Mã ID của câu lệnh phản hồi
Added status	Mã lỗi, dựa vào bảng tra cung cấp bởi nhà sản xuất có thể tìm ra lỗi đang gặp phải
Data size	Độ lớn của trường data
Data division	Dữ liệu trả về từ robot controller, tối đa 479 Byte

3. CHUẨN BỊ THÍ NGHIỆM

3.1. Lập trình thư viện YRC Ethernet Server giao tiếp máy tính với robot

Từ frame truyền giao tiếp robot ta tiến hành lập trình thư viện để giao tiếp giữa máy tính và robot. Thư viện gồm có 3 phần: class giao tiếp thông tin UDP, class tạo câu lệnh và class quản lý kết nối với robot.



Chi tiết về các khối:

UDP Connection: khởi tạo liên kết, quản lý việc gửi nhận thông tin thông qua UDP, nhận lệnh từ khối Communication control và gửi data qua các port đã được khởi tạo. Khối này sử dụng thư viện System.Net.Sockets để tạo kết nối UDP từ máy tính và robot.

Command Generator: Khởi tạo câu lệnh theo yêu cầu của khối Communication control. Khối này sẽ tạo một mã lệnh mặc định. Từ mã lệnh đó, ứng với mỗi câu lệnh khác nhau các trường mã lệnh sẽ được thay đổi để tạo ra các câu lệnh. Ngoài ra khối này sẽ chứa những dữ liệu được định nghĩa trước như ID câu lệnh hay các Service hoặc Attribute của từng câu lệnh cụ thể và các khối trên khối này có thể sử dụng các định nghĩa này mà không cần phải khai báo lại.

Communication control: quản lý việc giao tiếp bao gồm thông tin các port, gửi câu lệnh điều khiển, quản lý lỗi và các phản hồi từ robot. Đây là khối trung tâm trong thư viện giao tiếp, giúp kết nối hai khối còn lại. Nơi đây cũng là nơi cung cấp các API để sử dụng các hàm của thư viện. Khi có một yêu cầu gửi lệnh từ các hàm hoặc class bên ngoài, khối Communication control sẽ gọi khối Command Generator để tạo ra một bảng chứa các câu lệnh, bảng này sẽ được đưa vào khối UDP Connection để gửi cho robot. Dữ liệu đọc về và phản hồi từ robot sẽ hàm OnYRCDataCallback() quản lý và thực hiện các yêu cầu tùy vào người sử dụng.

3.2. Lập trình các khối

3.2.1. Khối UDP Connection:

```
public bool ConnectMotoman(){
    ipEndPointReceiving = new IPEndPoint(IPAddress.Any, 0);
    ipEndPointSending = new IPEndPoint(_IP_Address, _port);
    sendingUdpClient = new UdpClient();
    revThread.Start();
```

```
    return true;
}
public bool CloseMotoman(){
    sendingUdpClient.Close();
    return true;
}

public int SendCommand(Byte[] command){
    int ret = sendingUdpClient.Send(command, command.Length,ipEndPointSending);
    return ret;
}
public void ReceiveDataThread(){
    while (true){
        if (sendingUdpClient.Available > 0){
            ReceiveBytes = sendingUdpClient.Receive(ref ipEndPointReceiving);
            OnYRCDataCallback();
        }
    }
}
```

3.2.2. Khối Command generator:

```
public byte[] setServoOn(byte request_id_index){
    byte[] cmd = StringToByteArray(Constant.ON_SERVO_HEADER                    +
Constant.ON_SERVO_DATA);
    cmd[Constant.CMD_REQUEST_ID] = Convert.ToByte(request_id_index);
    cmd[Constant.CMD_ID_ADDRESS] = Constant.CMD_ID_SERVO_ON;
    cmd[Constant.CMD_INSTANCE] = Constant.CMD_HEADER_SERVO_INSTANCE_ON;
       cmd[Constant.CMD_ATTRIBUTE] = Constant.CMD_HEADER_SERVO_ATTRIBUTE;
    cmd[Constant.CMD_SERVICE] = Constant.CMD_HEADER_SERVO_SERVICE;
    cmd[Constant.DATA_BYTE0] = Constant.CMD_DATA_SERVO_ON;
    return cmd;
}
```

3.2.3. Khối Communication control:

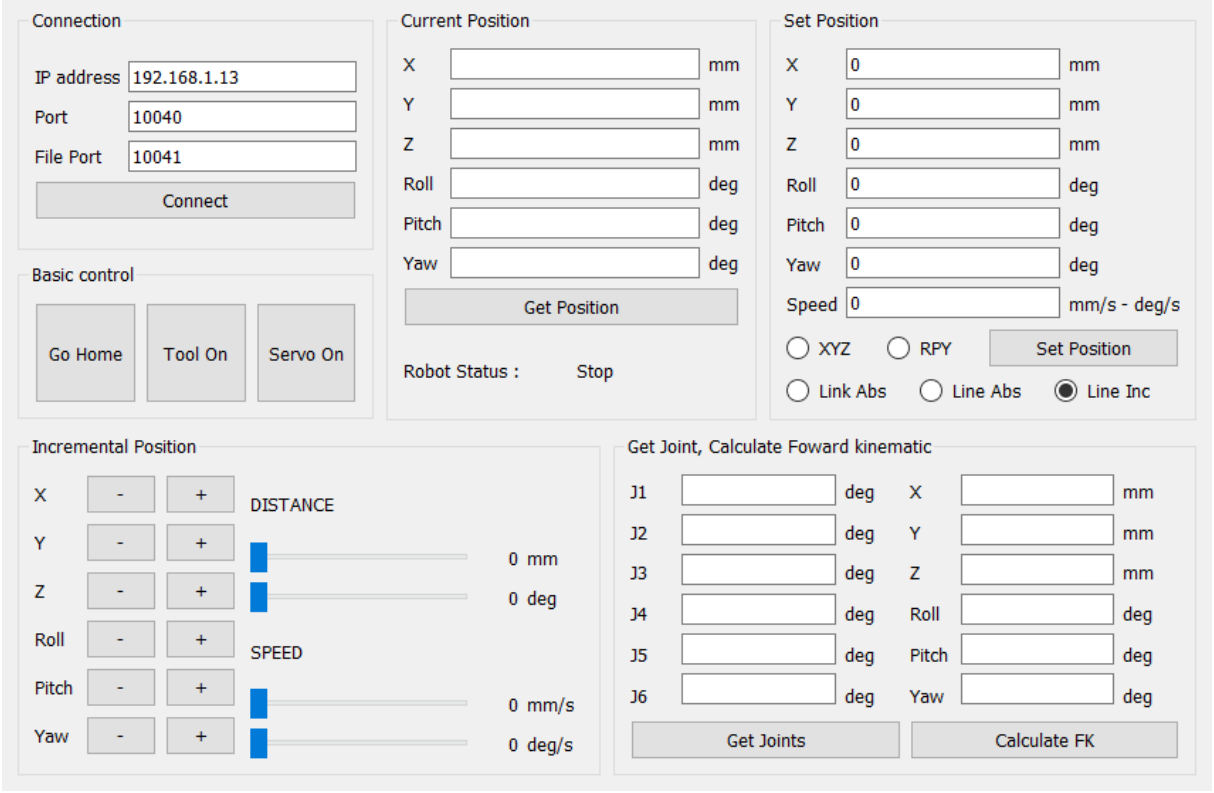
```
// Sending command data
public bool YRCOnServo(){
    byte[] data = YRCCommand.setServoOn(request_id_index);
    UDPSocket.SendCommand(data);
    request_code[request_id_index] = Constant.CMD_ID_SERVO_ON;
    request_id_index++;
    return true;
}
// Data Callback Handler
public void OnYRCDataCallback(object source, EventArgs e){
    byte[] data = UDPSocket.ReceiveBytes;
    string str_data = ByteArrayToHex(data);
    byte res_id_index = data[Constant.CMD_REQUEST_ID];
    byte response_id = request_code[res_id_index];
    if (response_id == Constant.CMD_ID_READ_ROBOT_POS){
        bool response_read_type = read_pos_type[res_id_index];
```

```
            if(response_read_type == Constant.READ_POSITION_CARTESIAN){
                YRCReadPositionResponse(data);
            }
            else if(response_read_type == Constant.READ_POSITION_PULSE){
                YRCReadPositionPulseResponse(data);
            }
        }
        else if (response_id == Constant.CMD_ID_STATUS_READING)
        {
            YRCReadStatusResponse(data);
        }
    }
```

4. THỰC HIỆN THÍ NGHIỆM

4.1. Lập trình phần hiển thị

Yêu cầu 1: Tạo giao diện theo mẫu dưới đây sử dụng phần mềm Qt Creator.



4.2. Lập trình phần xử lý

Yêu cầu 2: Lập trình các tác vụ chức năng như dưới đây:

4.2.1. Lập trình tác vụ Servo on/off:

Ta thực hiện kết nối với Robot bằng việc tạo ra đối tượng YRCCommunication và truyền cho đối tượng đó các tham số Ip address và port. Ta gọi hàm YRCCommunication.YRCConnect() để thực hiện kết nối với Robot, đồng thời bật các timer để quản lý việc cập nhật UI. Ta thực hiện việc bật tắt Servo thông qua lệnh YRCOnServo()/YRCOffServo().

```
private void btn_Connect_Click(object sender, RoutedEventArgs e){
    if (Convert.ToString(btn_Connect.Content) == "Connect"){
      YRCCommunication = new YRCCommunication(IPAddress.Parse(tbx_IpAddress.Te
xt), int.Parse(tbx_Port.Text), int.Parse(tbx_LocalPort.Text));
       if (YRCCommunication.YRCConnect()){
           btn_Connect.Content = "Disconnect";
          timer.Start();
          posUpdateTimer.Start();
       }
    }
    else{
        timer.Stop();
        posUpdateTimer.Stop();
        YRCCommunication.YRCDisconnect();
        btn_Connect.Content = "Connect";
        var bc = new BrushConverter();
    }
}

private void btn_OnOff_Servo_Click(object sender, RoutedEventArgs e){
    if (Convert.ToString(btn_OnOff_Servo.Content) == "ON Servo"){
        if (YRCCommunication.YRCOnServo()){
                    btn_OnOff_Servo.Content = "OFF Servo";
        }
        else{
            MessageBox.Show("Can not turn on servo!");
        }
    }
    else{
        if (YRCCommunication.YRCOffServo()){
            btn_OnOff_Servo.Content = "ON Servo";
        }
        else{
            MessageBox.Show("Can not turn off servo!");
        }
    }
}
```

4.2.2. Lập trình tác vụ điều khiển robot đến một vị trí cho trước.

Ta thực hiện bằng cách gọi hàm YRCMoveCartesian() và truyền vào các thông số hệ tọa độ, các thức di chuyển, đơn vị vận tốc, vận tốc và vị trí, góc hướng cần đi đến. Lưu ý các cách thức di chuyển của robot có thể thay đổi thông qua các tham số truyền vào hàm YRCMoveCartesian (đọc thêm manual để biết thêm chi tiết).

```
private void btn_xP_Click(object sender, RoutedEventArgs e){
    double[] pos = { Convert.ToDouble(txt_speed.Text) / 10, 0, 0, 0, 0, 0 };
    YRCCommunication.YRCMoveCartesian(Constant.CMD_DATA_MOVE_COORDINATE_ROBO
T, Constant.CMD_HEADER_MOVE_INSTANCE_STRAIGHT_INCREMENT,
                                      Constant.CMD_DATA_MOVE_SPEED_TYPE_V_SPEE
D, Convert.ToDouble(txt_speed.Text), pos);
}
```

4.2.3. Lập trình tác vụ điều khiển robot tăng tiến theo từng trục

Thực hiện tương tự như mục 4.2.2, thay tag dữ liệu truyền xuống từ Absolute thành Incremental.

4.2.4. Thực hiện đọc tọa độ không gian hiện tại của robot

Ta thực hiện bằng cách gọi hàm YRCReadPosition(). Hàm này sẽ đọc vị trí hiện tại của robot và lưu và biến nội của thư viện. Để đọc các giá trị vừa được cập nhật ta gọi hàm updateRobotPosition(). Lưu ý việc nhận dữ liệu được chạy trên một thread riêng biệt, để chỉnh sửa, thêm các hàm mới xử lý các thông tin khác từ robot, ta cần chỉnh sửa hàm OnYRCDataCallback() trong file YRCCommunication.cpp.

```
YRCCommunication.YRCReadPosition();
 double[] curr_pos = YRCCommunication.updateRobotPosition();
 txt_xAxis.Text = Convert.ToString(curr_pos[0]);
 txt_yAxis.Text = Convert.ToString(curr_pos[1]);
 txt_zAxis.Text = Convert.ToString(curr_pos[2]);
 txt_roll.Text = Convert.ToString(curr_pos[3]);
 txt_pitch.Text = Convert.ToString(curr_pos[4]);
 txt_yaw.Text = Convert.ToString(curr_pos[5]);
```

4.2.5. Thực hiện đọc tọa độ khớp hiện tại của robot

Tương tự như mục 4.2.4, lập trình tác vụ đọc tọa độ các góc khớp robot theo đơn vị xung, sau đó quy đổi sang đơn vị độ theo tỷ lệ trong bảng dưới đây:

Khớp robot	**Tỷ lệ xung/độ**
S	34816/30
L	102400/90
U	51200/90
R, B, T	10204/30

4.2.6. Lập trình tính động học thuận của robot

Lập trình tác vụ tính động học thuận của robot, sử dụng đầu vào là giá trị góc khớp từ mục 4.2.5, xuất ra giá trị tọa độ không gian của điểm cuối robot. Kiểm chứng với tọa độ không gian đọc được từ mục 4.2.4.

5. BÁO CÁO

Trình bày việc thực hiện các yêu cầu trong bài: chương trình giao diện máy tính, hình ảnh/video kết quả, trả lời câu hỏi, nhận xét, kết luận.

TÀI LIỆU THAM KHẢO

[1] YRC1000micro Instruction for Ethernet Function, Yaskawa Electric Corp, 2018.

[2] Qt Creator Manual, https://doc.qt.io/qtcreator/.