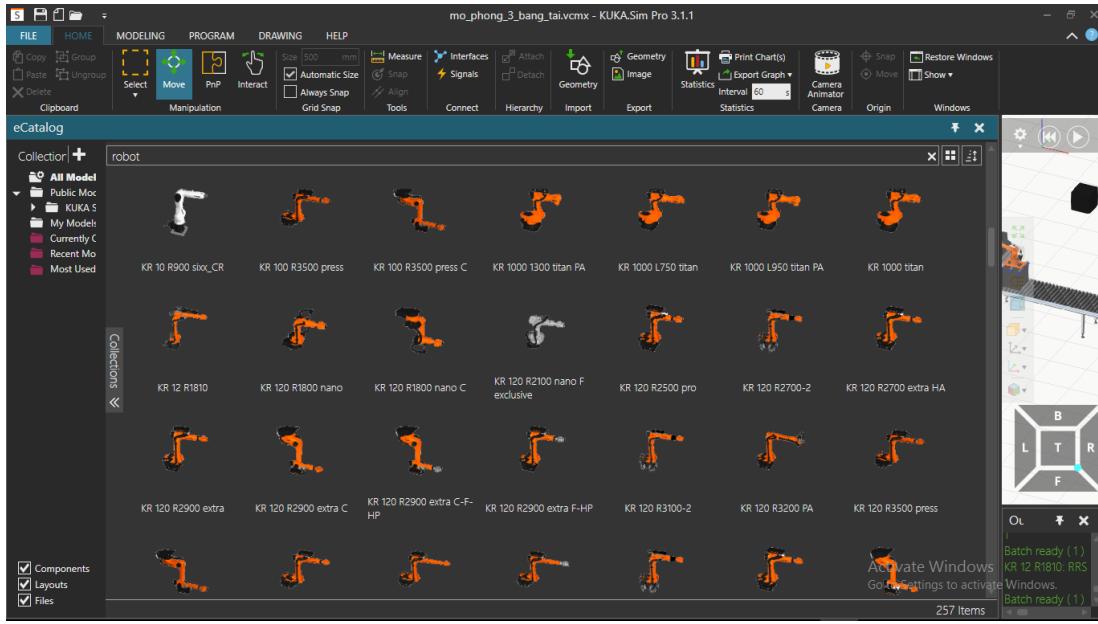


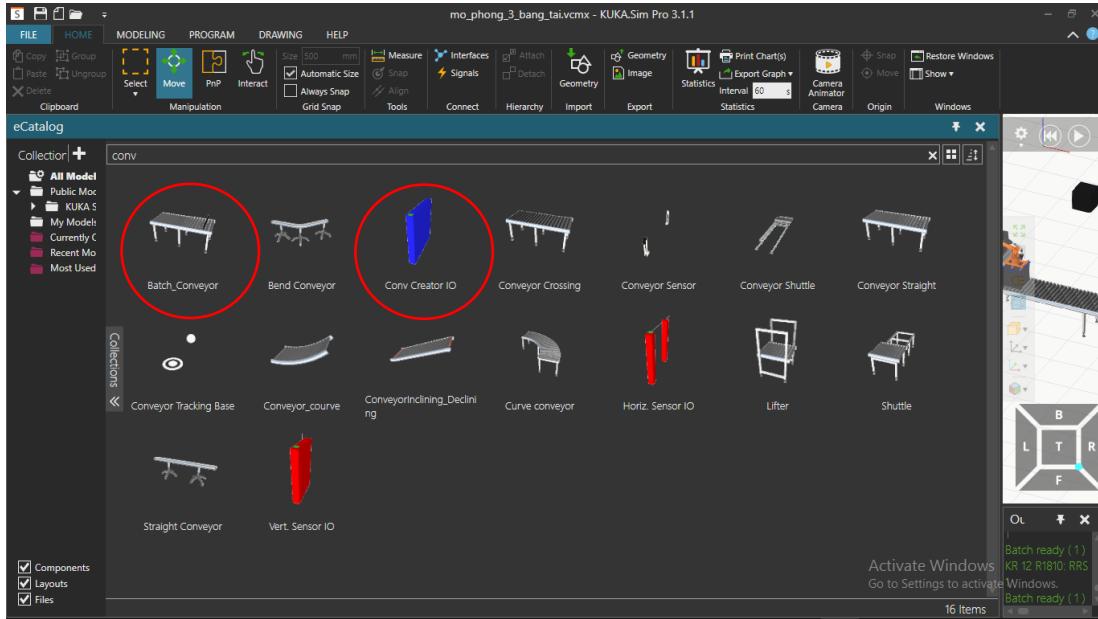
Ngày 8 tháng 5 năm 2023

Buổi sáng

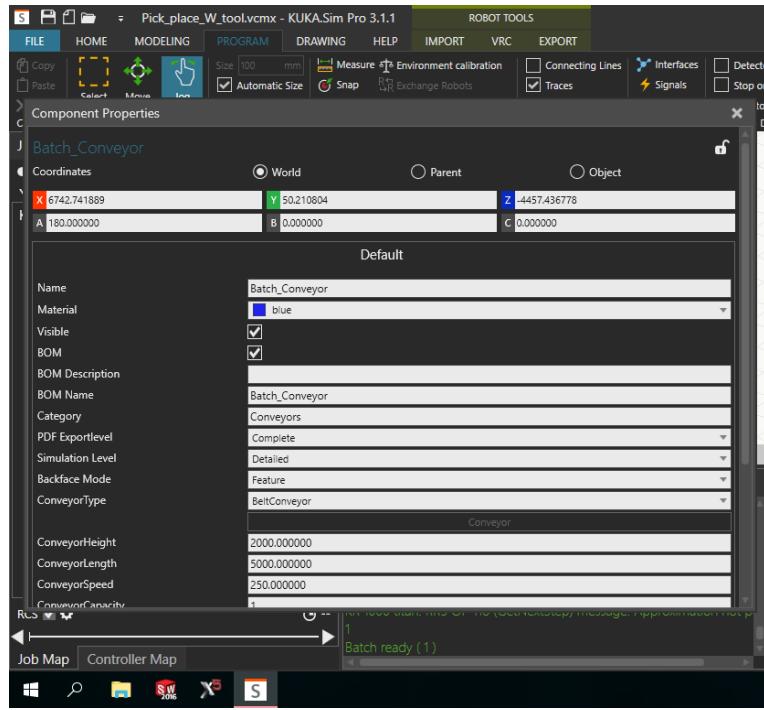
Bước 1 chọn Model 3D robot



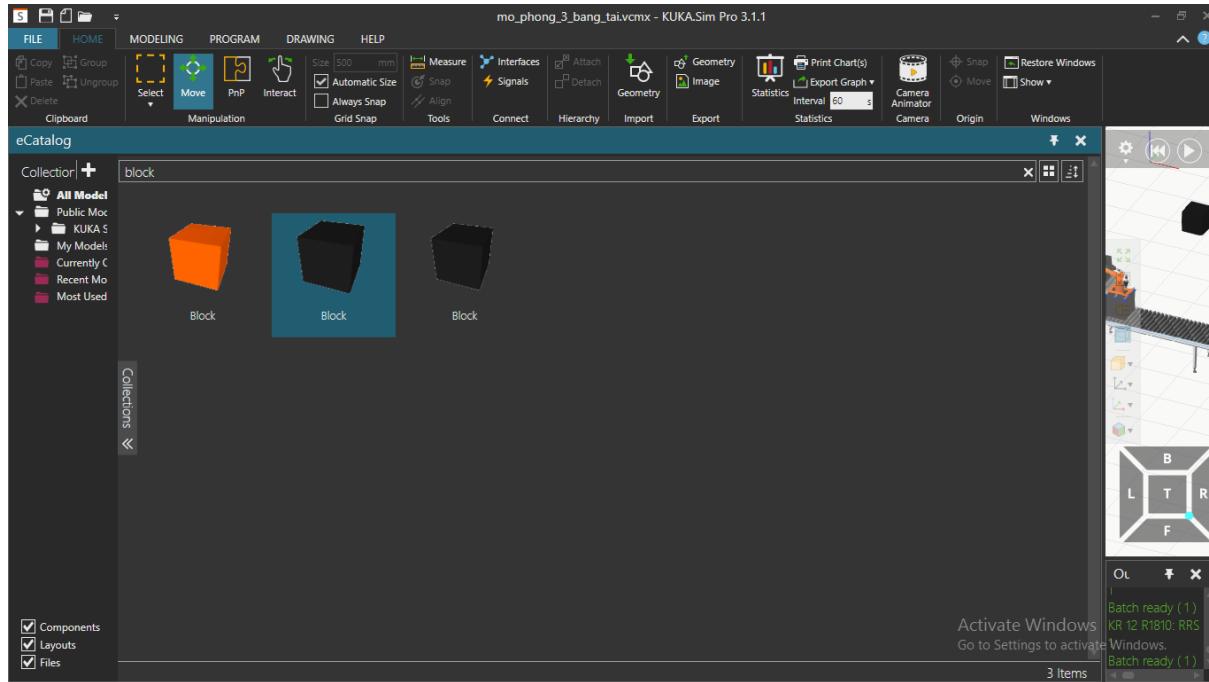
Bước 2 chọn Model 3D băng truyền và khối tạo phôi liên tục



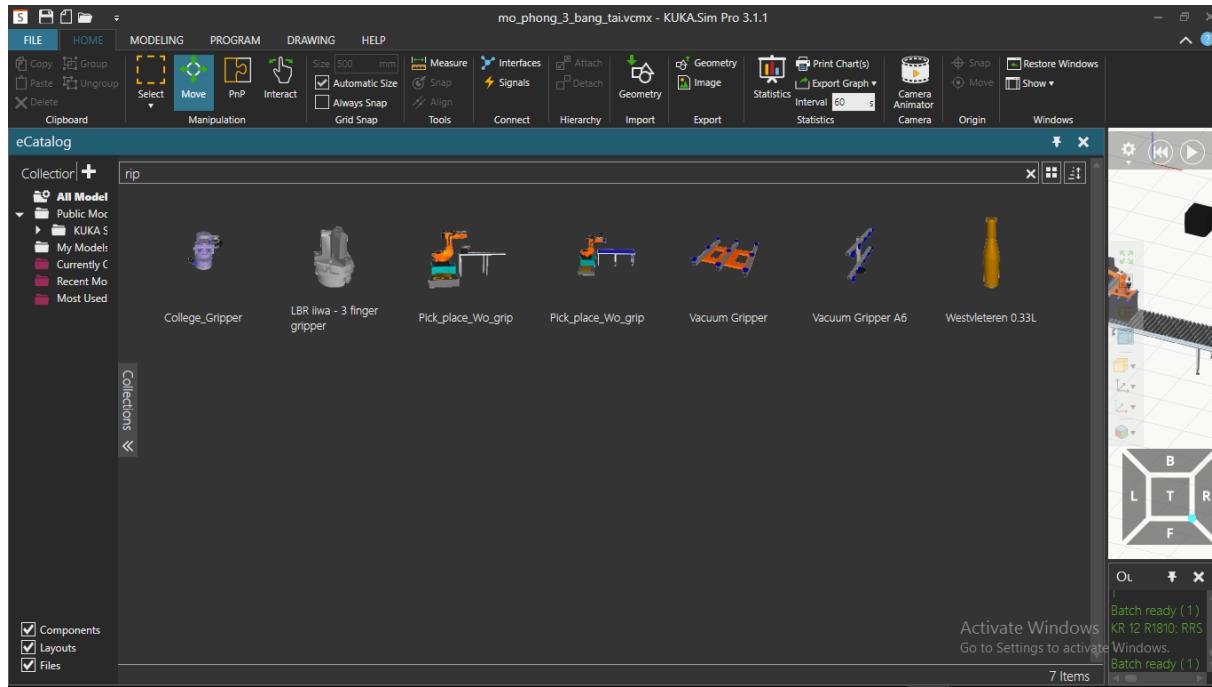
Bước 3 Setup cho băng tải



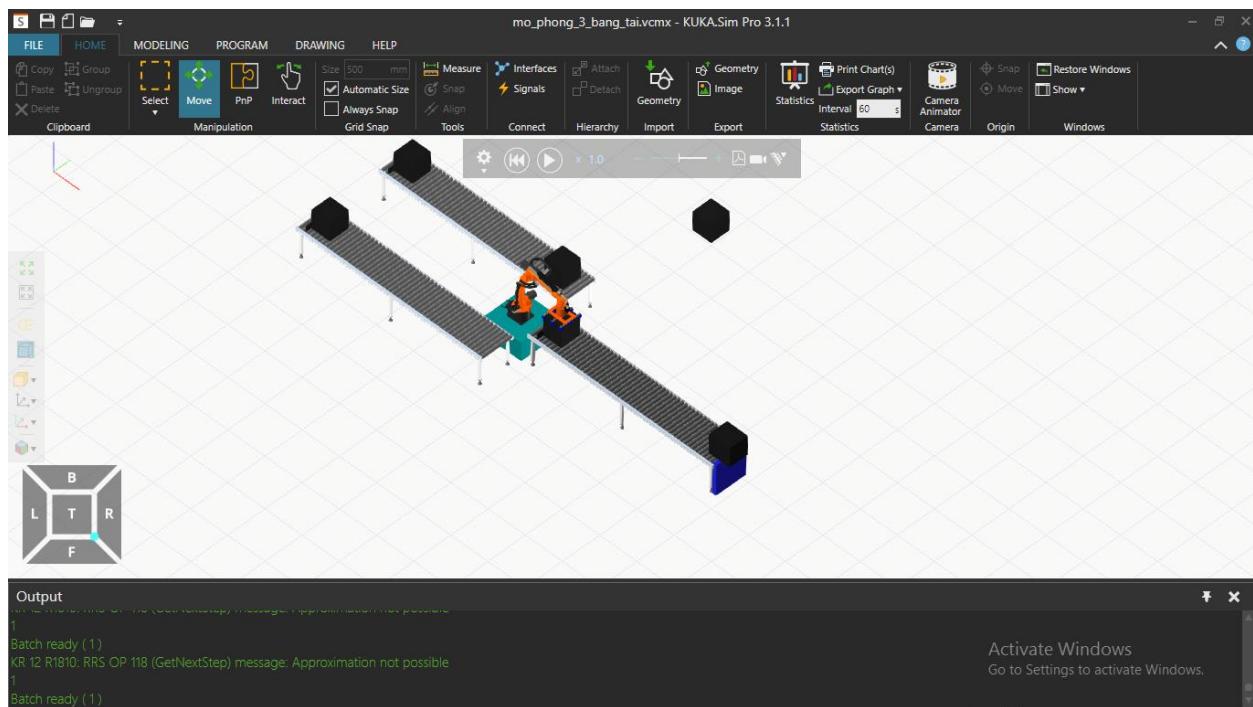
Bước 4 chọn Phôi



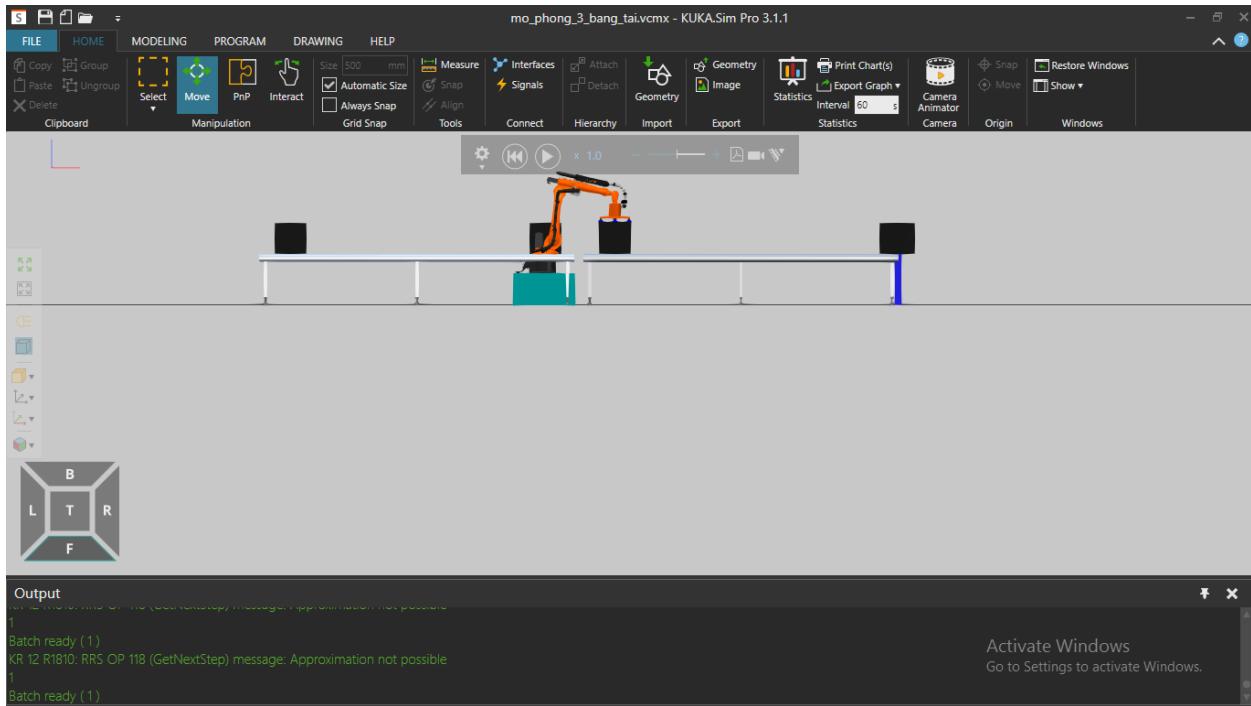
Bước 5 Chọn tay gấp robot



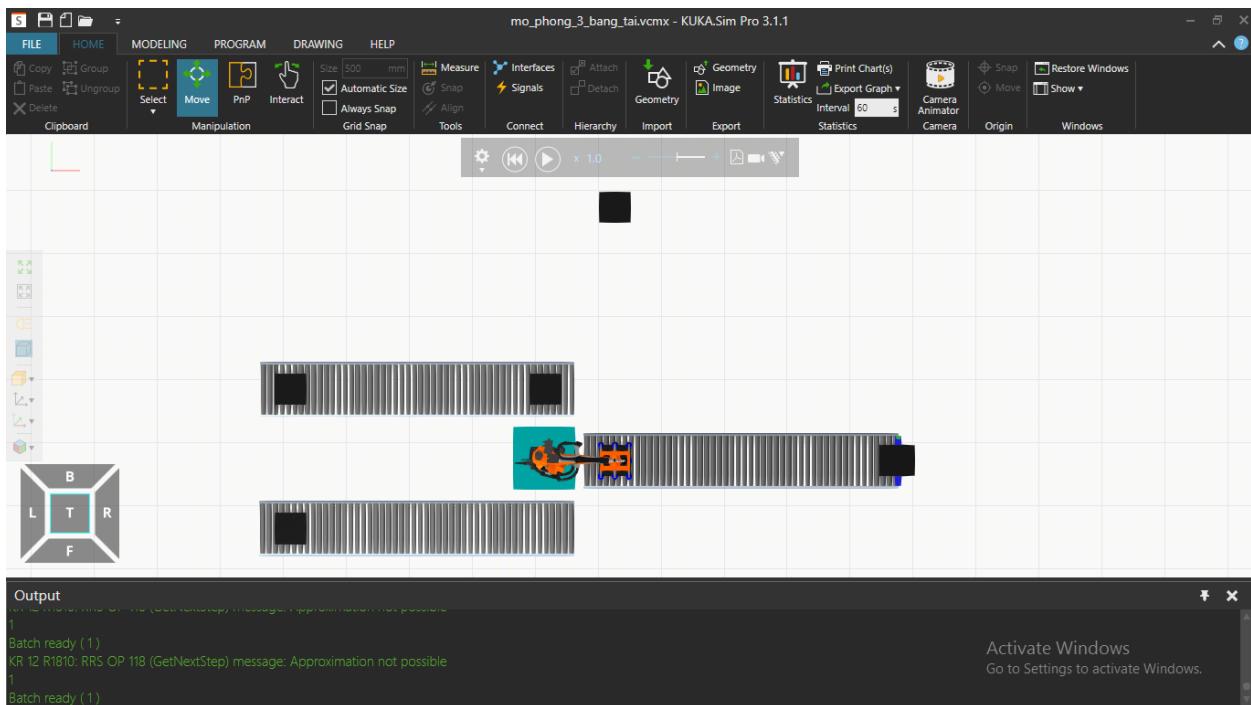
Bước 6 setup model trên hình chiếu trực đo



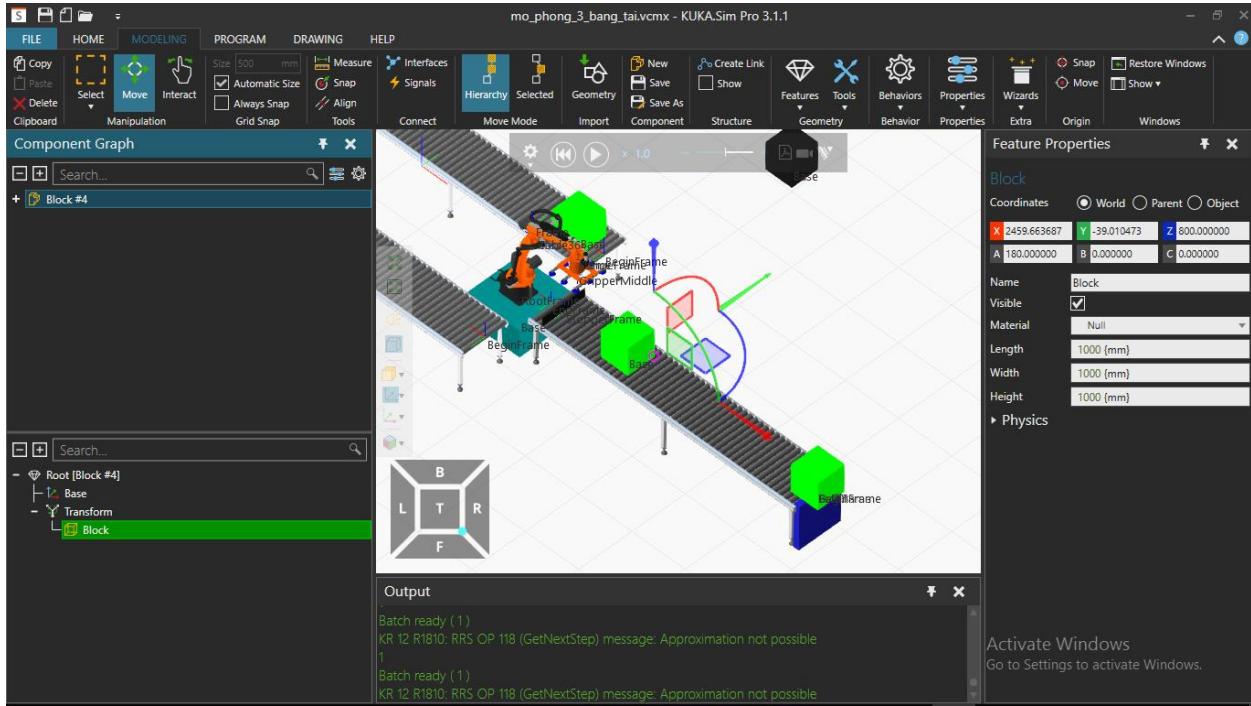
Bước 7 setup model trên hình chiếu đứng



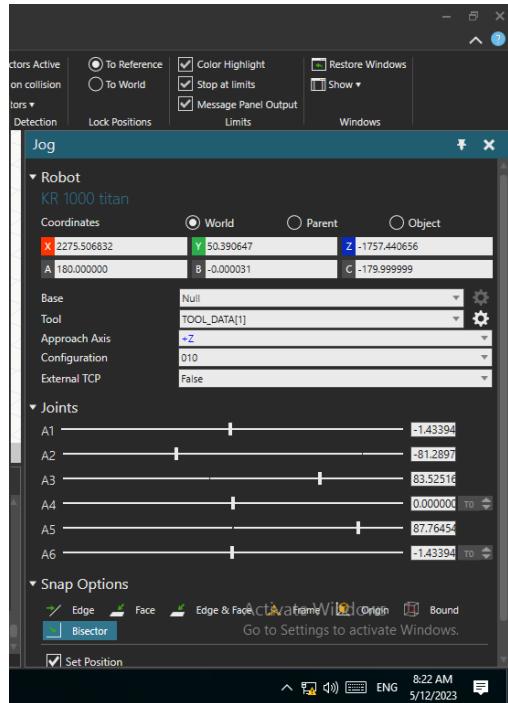
Bước 8 setup model trên hình chiếu bằng



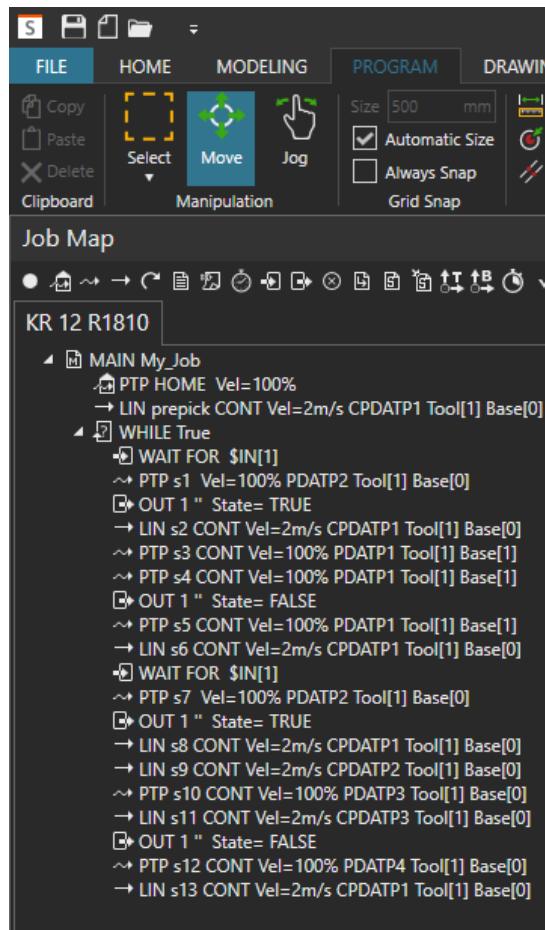
Bước 9 setup kích cỡ vị trí phôi



Bước 10 setup cho vị trí cánh tay robot di chuyển đến



Bước 11 viết code cho chương trình



MAIN My Job: Đây là khai báo chương trình chính của robot, được gọi là "My Job".

PTP HOME Vel=100%: Lệnh này yêu cầu robot di chuyển đến vị trí HOME (vị trí mặc định) với tốc độ 100% (tốc độ tối đa).

LIN prepick CONT Vel=2m/s CPDATP1 Tool[1] Base[0]: Lệnh này yêu cầu robot di chuyển theo đường thẳng (LINEAR) đến một vị trí cụ thể được gọi là "prepick" với tốc độ 2m/s. Các thông số khác như CPDATP1, Tool[1], Base[0] tọa độ cơ sở của robot.

WHILE True: Đây là một vòng lặp vô hạn. Chương trình sẽ tiếp tục thực hiện các lệnh bên trong vòng lặp này.

WAIT FOR \$IN[1]: Lệnh này yêu cầu robot chờ đợi cho đến khi tín hiệu đầu vào \$IN[1] được kích hoạt.

PTP s1 Vel=100% PDATP2 Tool[1] Base[0]: Lệnh này yêu cầu robot di chuyển đến vị trí "s1" với tốc độ 100% và các thông số khác như PDATP2, Tool[1], Base[0].

OUT 1" State= TRUE: Lệnh này gửi một tín hiệu ra (OUTPUT) để đặt trạng thái của một thiết bị nào đó thành TRUE.

→ LIN s2 CONT Vel=2m/s CPDATP1 Tool[1] Base[0]: Lệnh này yêu cầu robot di chuyển theo đường thẳng đến vị trí "s2" với tốc độ 2m/s và các thông số khác.

~ PTP s3 CONT Vel=100% PDATP1 Tool[1] Base[1]: Lệnh này yêu cầu robot di chuyển đến vị trí "s3" với tốc độ 100% và các thông số khác.

~ PTP s4 CONT Vel=100% PDATP1 Tool[1] Base[1]: Lệnh này yêu cầu robot di chuyển đến vị trí "s4" với tốc độ 100% và các thông số khác.

OUT 1" State= FALSE: Lệnh này gửi một tín hiệu ra để đặt trạng thái của thiết bị OUT 1" State= FALSE

~ PTP s5 CONT Vel=100% PDATP1 Tool[1] Base[1]: Lệnh này yêu cầu robot di chuyển đến vị trí "s5" với tốc độ 100% và các thông số khác.

→ LIN s6 CONT Vel=2m/s CPDATP1 Tool[1] Base[0]: Lệnh này yêu cầu robot di chuyển theo đường thẳng đến vị trí "s6" với tốc độ 2m/s và các thông số khác.

-WAIT FOR \$IN[1] Lệnh này yêu cầu robot chờ đợi cho đến khi tín hiệu đầu vào \$IN[1] được kích hoạt.

~ PTP s7 Vel=100% PDATP2 Tool[1] Base[0] : Lệnh này yêu cầu robot di chuyển đến vị trí "s7" với tốc độ 100% và các thông số khác.

OUT 1" State= TRUE: Lệnh này gửi một tín hiệu ra (OUTPUT) để đặt trạng thái của một thiết bị nào đó thành TRUE.

→ LIN s8 CONT Vel=2m/s CPDATP1 Tool[1] Base[0] : Lệnh này yêu cầu robot di chuyển theo đường thẳng đến vị trí "s8" với tốc độ 2m/s và các thông số khác.

→ LIN s9 CONT Vel=2m/s CPDATP2 Tool[1] Base[0] : Lệnh này yêu cầu robot di chuyển theo đường thẳng đến vị trí "s9" với tốc độ 2m/s và các thông số khác.

~ PTP s10 CONT Vel=100% PDATP3 Tool[1] Base[0] : Lệnh này yêu cầu robot di chuyển đến vị trí "s10" với tốc độ 100% và các thông số khác.

→ LIN s11 CONT Vel=2m/s CPDATP3 Tool[1] Base[0] : Lệnh này yêu cầu robot di chuyển theo đường thẳng đến vị trí "s11" với tốc độ 2m/s và các thông số khác.

OUT 1" State= FALSE: Lệnh này gửi một tín hiệu ra để đặt trạng thái của thiết bị OUT 1" State= FALSE

~ PTP s12 CONT Vel=100% PDATP4 Tool[1] Base[0] : Lệnh này yêu cầu robot di chuyển đến vị trí "s12" với tốc độ 100% và các thông số khác.

→ LIN s13 CONT Vel=2m/s CPDATP1 Tool[1] Base[0] : Lệnh này yêu cầu robot di chuyển theo đường thẳng đến vị trí "s13" với tốc độ 2m/s và các thông số khác.

Buổi chiều

Các biến khai báo:

pTerm, iTerm, dTerm: Biến kiểu float để lưu trữ các giá trị của các thành phần P, I, D trong điều khiển PID.

error: Biến kiểu int để lưu trữ giá trị sai số (error) tính toán từ đọc cảm biến.

previousError: Biến kiểu int để lưu trữ giá trị sai số (error) của lần đọc trước.

kp, ki, kd: Các hệ số P, I, D của điều khiển PID.

output: Biến kiểu float để lưu trữ giá trị đầu ra của điều khiển PID.

integral, derivative: Biến kiểu int để lưu trữ giá trị tổng tích phân và đạo hàm của sai số (error).

irSensors: Mảng chứa các chân của cảm biến hồng ngoại.

irReadings: Mảng chứa giá trị đọc từ các cảm biến hồng ngoại.

Các biến motor1Forward, motor1Backward, motor1pwmPin, motor2Forward, motor2Backward, motor2pwmPin: Chứa các chân kết nối đến động cơ và các tín hiệu điều khiển của chúng.

motor1newSpeed, motor2newSpeed: Biến kiểu int để lưu trữ tốc độ mới của các động cơ.

motor2Speed, motor1Speed: Tốc độ mặc định của động cơ.

Phần setup(): Được gọi một lần duy nhất khi chương trình khởi động. Thiết lập các chân kết nối và cảm biến hồng ngoại là đầu vào (INPUT) hoặc đầu ra (OUTPUT).

Phần loop(): Được thực thi liên tục sau khi phần setup() đã hoàn thành. Chứa các hàm chính để điều khiển robot:

readIRSensors(): Đọc giá trị từ các cảm biến hồng ngoại và lưu trữ chúng trong mảng irReadings.

calculateError(): Tính toán giá trị sai số (error) dựa trên giá trị đọc từ các cảm biến hồng ngoại.

pidCalculations(): Tính toán các thành phần P, I, D của điều khiển PID dựa trên sai số (error)

pTerm = kp * error;: Tính toán thành phần P (Proportional) bằng cách nhân hệ số P (kp) với giá trị sai số (error).

integral += error;; Cập nhật giá trị tổng tích phân (integral) bằng cách thêm giá trị sai số (error) vào giá trị hiện tại của tổng tích phân.

iTerm = ki * integral;; Tính toán thành phần I (Integral) bằng cách nhân hệ số I (ki) với giá trị tổng tích phân (integral).

derivative = error - previousError;; Tính toán thành phần D (Derivative) bằng hiệu của giá trị sai số (error) và giá trị sai số trước đó (previousError).

dTerm = kd * derivative;; Tính toán thành phần D (Derivative) bằng cách nhân hệ số D (kd) với giá trị đạo hàm (derivative).

output = pTerm + iTerm + dTerm;; Tính toán giá trị đầu ra (output) bằng tổng của các thành phần P, I, D.

previousError = error;; Cập nhật giá trị sai số trước đó (previousError) bằng giá trị sai số hiện tại (error).

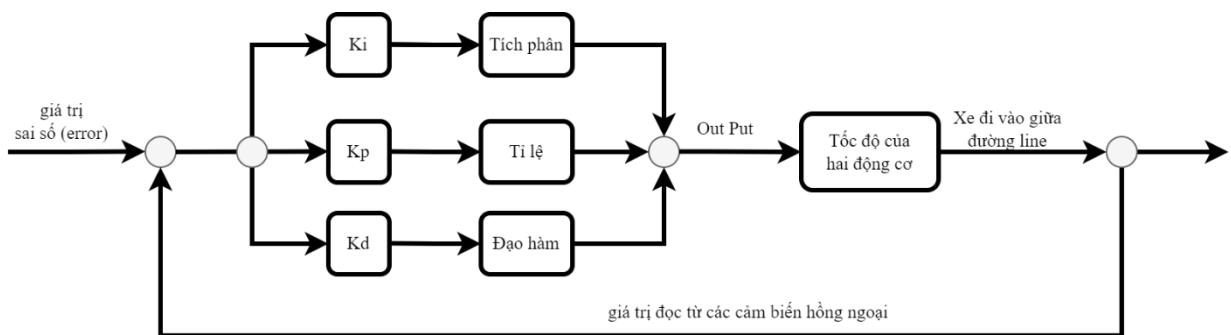
Cuối cùng, trong hàm changeMotorSpeed(), tốc độ mới của các động cơ được tính toán dựa trên giá trị đầu ra (output) của điều khiển PID. Giá trị này được áp dụng vào tốc độ của mỗi động cơ và sử dụng các tín hiệu điều khiển để đảm bảo chuyển động tiến về phía trước.

Ngày 9 tháng 5 năm 2023

Buổi sáng

Robot – hoàn thành

Buổi chiều



Ngày 10 tháng 5 năm 2023

Buổi sáng

Robot – hoàn thành

Buổi chiếu

```
#include <AFMotor.h>

AF_DCMotor motor1(1); // Create motor object for motor 1
AF_DCMotor motor2(2); // Create motor object for motor 2
float pTerm, iTerm, dTerm;
int error;
int previousError;
float kp = 10;
float ki = 0;
float kd = 10;
float output;
int integral, derivative;

int R1 = A1;
int R2 = A2;
int M = A3;
int L2 = A4;
int L1 = A5;

int motor1Forward = 11;
int motor2Forward = 6;
int motor1Backward = 10;
int motor2Backward = 9;
int motor1pwmPin = 5;
int motor2pwmPin = 3;
int motor1newSpeed;
int motor2newSpeed;
int s = 100;
int motor1Speed = s; //Default r
int motor2Speed = s; //Default l

void setup() {
    //Declare all IR sensors as inputs
    pinMode(L1, INPUT); // L1
    pinMode(L2, INPUT); // L2
    pinMode(M, INPUT); // M
    pinMode(R2, INPUT); // R2
    pinMode(R1, INPUT); // R1
    pinMode(motor1Forward, OUTPUT);
    pinMode(motor1Backward, OUTPUT);
    pinMode(motor1pwmPin, OUTPUT);
    pinMode(motor2Forward, OUTPUT);
```

```

pinMode(motor2Backward, OUTPUT);
pinMode(motor2pwmPin, OUTPUT);
motor1.setSpeed(motorSpeed);

Serial.begin(9600);
}

void loop() {
    //Put all of our functions here
    if ((digitalRead(L1) == 1) && (digitalRead(L2) == 1) && (digitalRead(M) == 1)
&& (digitalRead(R2) == 1) && (digitalRead(R1) == 1)) {
        stopMotors();
    }
    calculateError();
    pidCalculations();
    changeMotorSpeed();

    Serial.print("R1 : ");Serial.println(digitalRead(R1));
    Serial.print("R2 : ");Serial.println(digitalRead(R2));
    Serial.print("M : ");Serial.println(digitalRead(M));
    Serial.print("L2 : ");Serial.println(digitalRead(L2));
    Serial.print("L1 : ");Serial.println(digitalRead(L1));

    Serial.print("motor1newSpeed: ");
    Serial.println(motor1newSpeed);
    Serial.print("motor2newSpeed: ");
    Serial.println(motor2newSpeed);

    delay(1000);
}

//đổi digital thành analog
void calculateError() {
    //Determine an error based on the readings
    if      ((digitalRead(L1)== 1) && (digitalRead(L2) == 0) && (digitalRead(M)
== 0) && (digitalRead(R2)== 0) && (digitalRead(R1) == 0)) {
        error = 4; // extreme right sensor
    } else if ((digitalRead(L1)== 1) && (digitalRead(L2) == 1) && (digitalRead(M)
== 0) && (digitalRead(R2)== 0) && (digitalRead(R1) == 0)) {
        error = 3;// right
    } else if  ((digitalRead(L1)== 1) && (digitalRead(L2) == 1) && (digitalRead(M)
== 1) && (digitalRead(R2)== 0) && (digitalRead(R1) == 0)) {
        error = 2;
    } else if  ((digitalRead(L1)== 1) && (digitalRead(L2) == 1) && (digitalRead(M)
== 1) && (digitalRead(R2)== 0) && (digitalRead(R1) == 1)) {

```

```

        error = 1;
    } else if ((digitalRead(L1)== 1) && (digitalRead(L2) == 1) && (digitalRead(M)
== 0) && (digitalRead(R2)== 0) && (digitalRead(R1) == 1)) {
        error = 1;
    } else if ((digitalRead(L1)== 1) && (digitalRead(L2) == 0) && (digitalRead(M)
== 0) && (digitalRead(R2)== 0) && (digitalRead(R1) == 1)) {
        error = 0;
    } else if ((digitalRead(L1)== 1) && (digitalRead(L2) == 1) && (digitalRead(M)
== 0) && (digitalRead(R2)== 1) && (digitalRead(R1) == 1)) {
        error = 0;
    } else if ((digitalRead(L1)== 1) && (digitalRead(L2) == 0) && (digitalRead(M)
== 1) && (digitalRead(R2)== 1) && (digitalRead(R1) == 1)) {
        error = -1;
    } else if ((digitalRead(L1)== 1) && (digitalRead(L2) == 0) && (digitalRead(M)
== 0) && (digitalRead(R2)== 1) && (digitalRead(R1) == 1)) {
        error = -1;
    } else if ((digitalRead(L1)== 0) && (digitalRead(L2) == 0) && (digitalRead(M)
== 1) && (digitalRead(R2)== 1) && (digitalRead(R1) == 1)) {
        error = -4;
    } else if ((digitalRead(L1)== 0) && (digitalRead(L2) == 0) && (digitalRead(M)
== 0) && (digitalRead(R2)== 1) && (digitalRead(R1) == 1)) {
        error = -6;
    } else if ((digitalRead(L1)== 0) && (digitalRead(L2) == 0) && (digitalRead(M)
== 0) && (digitalRead(R2)== 0) && (digitalRead(R1) == 1)) {
        error = -8;//extreme left sensor
    } else if ((digitalRead(L1)== 0) && (digitalRead(L2) == 0) && (digitalRead(M)
== 0) && (digitalRead(R2)== 0) && (digitalRead(R1) == 0)) {
        error = 0;
    }
}

void pidCalculations()  {
    pTerm = kp * error;
    integral += error;
    iTerm = ki * integral;
    derivative = error - previousError;
    dTerm = kd * derivative;
    output  = pTerm + iTerm + dTerm;
    previousError = error;
}

void stopMotors() {
    analogWrite(motor1pwmPin, 0); // set PWM to 0 to stop motor 1
    analogWrite(motor2pwmPin, 0); // set PWM to 0 to stop motor 2
}

```

```

digitalWrite(motor1Forward, LOW);
digitalWrite(motor1Backward, LOW);
digitalWrite(motor2Forward, LOW);
digitalWrite(motor2Backward, LOW);
}

void changeMotorSpeed() {
    // Check if all IR readings are 0
    if ((digitalRead(L1) == 1) && (digitalRead(L2) == 1) && (digitalRead(M) == 1)
&& (digitalRead(R2) == 1) && (digitalRead(R1) == 1)) {
        // Perform specific action when all IR readings are 0
        // For example, stop both motors
        analogWrite(motor1pwmPin, 0); // Tắt động cơ 1
        analogWrite(motor2pwmPin, 0); // Tắt động cơ 2
        digitalWrite(motor1Forward, LOW);
        digitalWrite(motor1Backward, LOW);
        digitalWrite(motor2Forward, LOW);
        digitalWrite(motor2Backward, LOW);
    } else {
        // Change motor speed accordingly
        motor1newSpeed = motor1Speed - output;
        motor2newSpeed = motor2Speed + output;
        // Constrain the new speed of the motors to be between the range 0-255
        motor1newSpeed = constrain(motor1newSpeed, 0, 255);
        motor2newSpeed = constrain(motor2newSpeed, 0, 255);
        // Set new speed and run the motors in the forward direction
        analogWrite(motor1pwmPin, motor1newSpeed);
        analogWrite(motor2pwmPin, motor2newSpeed);
        digitalWrite(motor1Forward, HIGH);
        digitalWrite(motor1Backward, LOW);
        digitalWrite(motor2Forward, HIGH);
        digitalWrite(motor2Backward, LOW);
    }
}

```

Ngày 11 tháng 5 năm 2023

Mục lục

BÀI 1 : THỰC HÀNH LẬP TRÌNH TRUYỀN THÔNG PROFINET S7-300 VÀ S7-1200.....	15
I. Nội dung thực hành	15
II. Bài toán :	15

III. Thực hành :	15
BÀI 2 : THỰC HÀNH LẬP TRÌNH TRUYỀN THÔNG PROFIBUS S7-300 VÀ ET-200	28
I. Nội dung thực hành :	28
II. Bài toán :	28
III. Thực hành :	28
BÀI 3 : THỰC HÀNH LẬP TRÌNH TRUYỀN THÔNG ASI S7-1200 VÀ ASI SLAVE.....	35
I. Nội dung thực hành :	35
II. Bài toán :	35
III. Thực hành :	35
BÀI 4 : THỰC HÀNH LẬP TRÌNH TRUYỀN THÔNG ASI S7-300 VÀ ASI SLAVE THÔNG QUA MODULE CHUYỂN ĐỔI	43
I. Nội dung thực hành :	43
II. Bài toán :	43
III. Thực hành :	43
BÀI 5 : THỰC HÀNH LẬP TRÌNH TRUYỀN THÔNG MODBUS RTU PLC S7-1200 VÀ BIẾN TẦN V20	52
I. Nội dung thực hành :	52
II. Bài toán :	52
III. Thực hành :	52

BÀI 1 : THỰC HÀNH LẬP TRÌNH TRUYỀN THÔNG PROFINET S7-300 VÀ S7-1200

I. Nội dung thực hành

Trong bài thí nghiệm này sinh viên cần thực hành và nắm được 2 nội dung chính :

Nội dung 1 : Kết nối và cấp nguồn thiết bị

Nội dung 2 : Lập trình điều khiển truyền thông Profinet giữa PLC S7-300 và S7-1200

II. Bài toán :

Đặt PLC S7-300 làm master và PLC S7-1200 làm Slave.

+ Tác động I0.0 trên S7-1200 bằng công tắc lên 1, đèn sáng bằng việc S7-1200 sét Q0.0 lên 1 (S7-300 truyền tín hiệu số cho S7-1200)

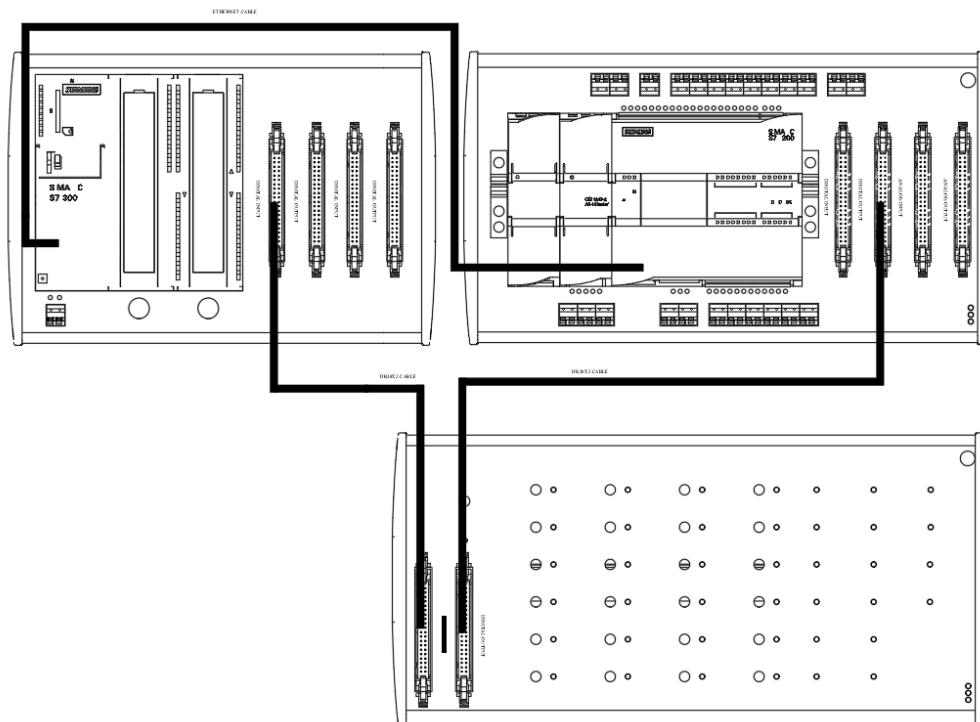
+ Gửi tín hiệu analog đến kênh 1 của S7-300 bằng module mô phỏng vào ra analog, S7-1200 sẽ xuất tín hiệu analog tương ứng trên kênh 1 đến module mô phỏng vào ra analog.

III. Thực hành :

- Các bước thực hành :

Bước 1 : Kết nối Digital input của module mô phỏng vào ra số với digital input của S7-300 bằng cáp DB20x2

Bước 2 : Kết nối Digital Output của S7-1200 với digital output của module mô phỏng vào ra số

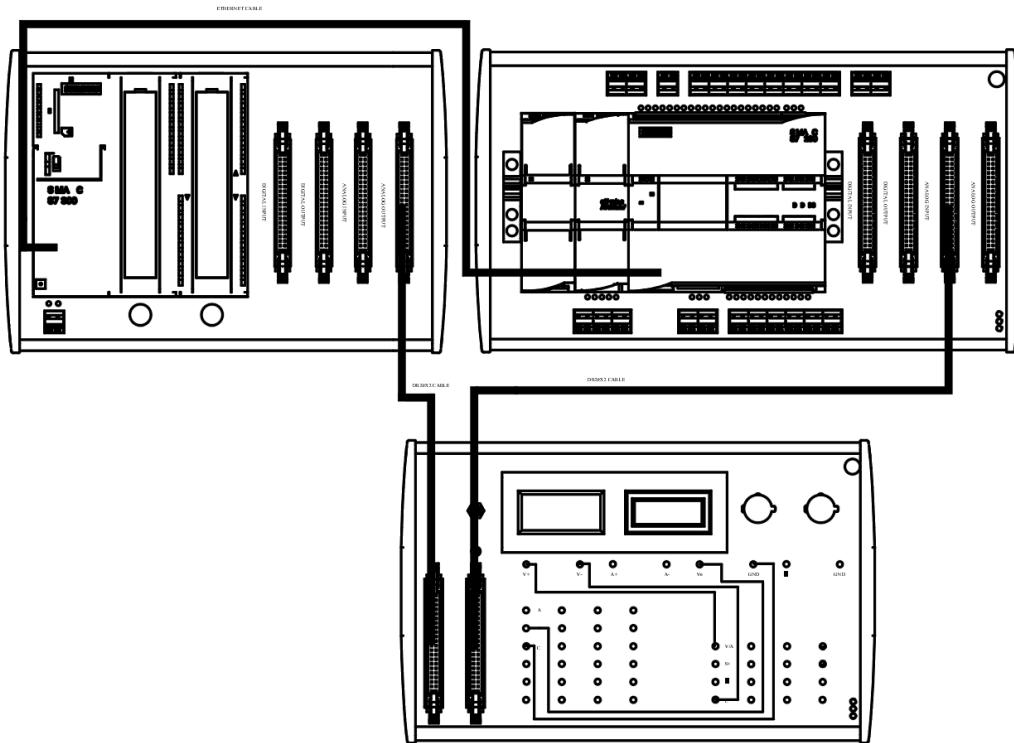


Bước 3 : Kết nối analog input của module mô phỏng vào ra analog với analog input S7-300

Bước 4 : Kết nối analog output S7-1200 với analog output của module analog output của module mô phỏng vào ra analog

Bước 5 : Dùng jack M2 nối các chân để xuất tín hiệu analog điện áp 0-10V từ module vào ra analog đến S7-300 trên kênh 1

Bước 6 : Dùng jack M2 nối các chân để xuất tín hiệu analog điện áp 0-10V từ S7-1200 đến module vào ra analog

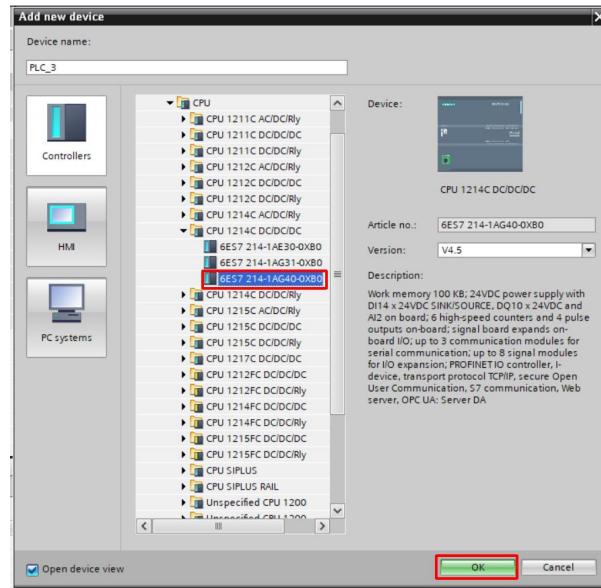


Bước 7 : Cấp nguồn cho thiết bị, cáp Ethernet giữa máy tính với PLC và giữa PLC S7-300, S7-1200

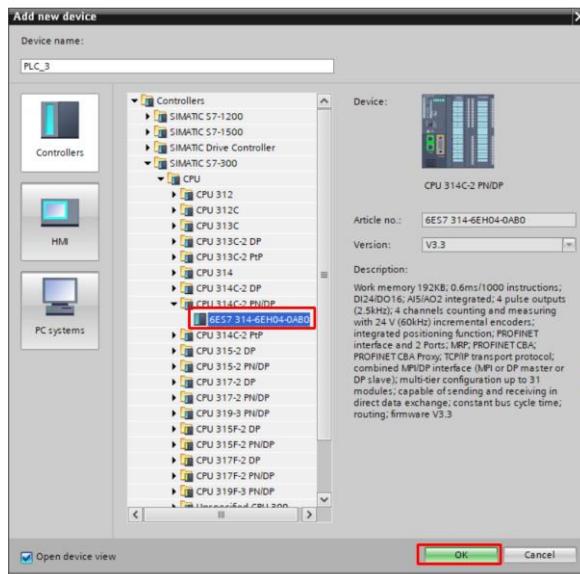
Bước 8 : Khai báo và kết nối phần cứng

* Thêm phần cứng :

- Thêm PLC S7 1200. Chọn PLC 6ES7 214-1AG40-0XB0 ver 4.5



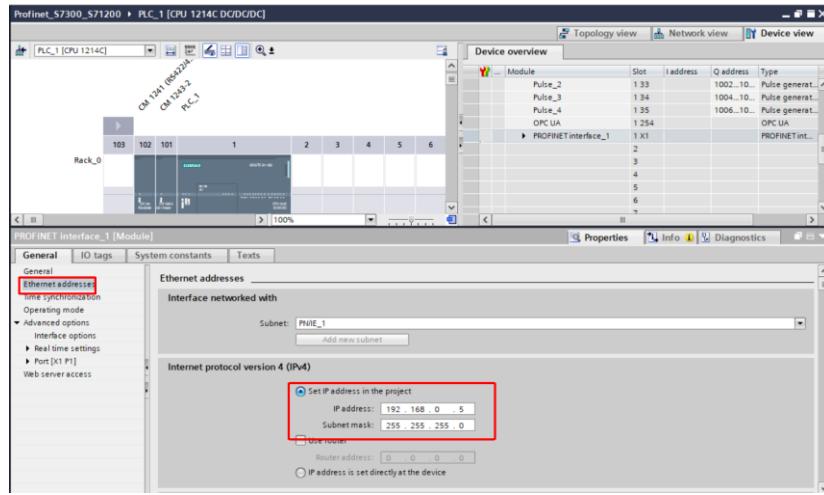
- Thêm PLC S7 300. Chọn PLC 6ES7 314-6EH04-0AB0 ver3.3



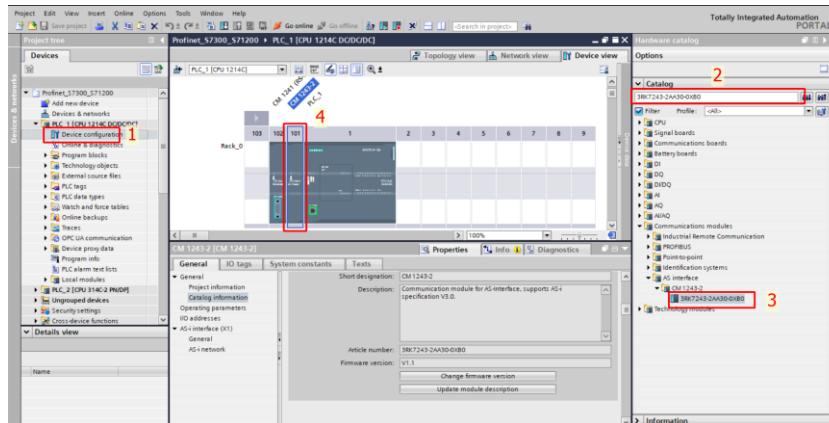
* Cài đặt địa chỉ cho PLC

Đặt địa chỉ cho Slave S7-1200 :

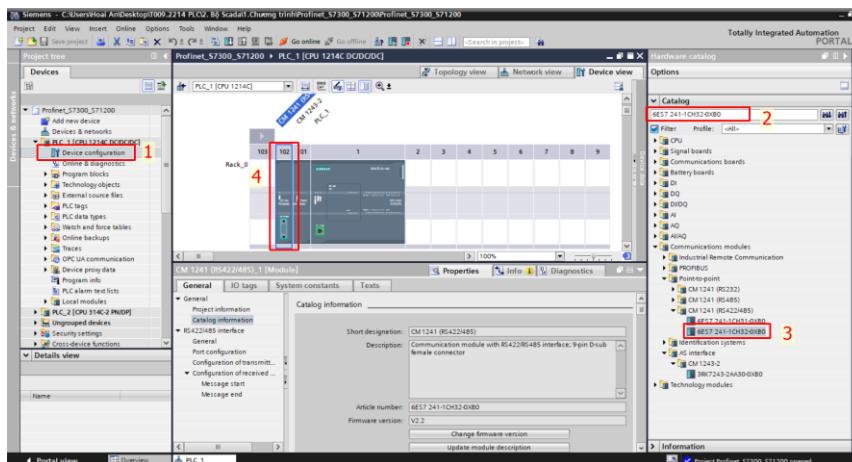
Ví dụ : đặt địa chỉ IP S7-1200 là 192.168.0.5



Khai báo module truyền thông ASI. Tại mục Hardware Catalog ta nhập mã module 3RK7243-2AA30-0XB0 để tạo module.

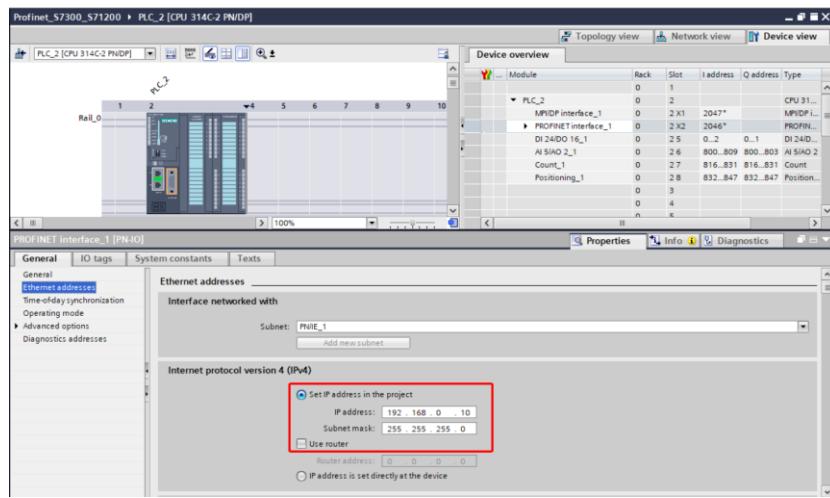


Khai báo module truyền thông Modbus. Tại mục Hardware Catalog ta nhập mã module 6ES7 241-1CH32-0XB0 để tạo module.

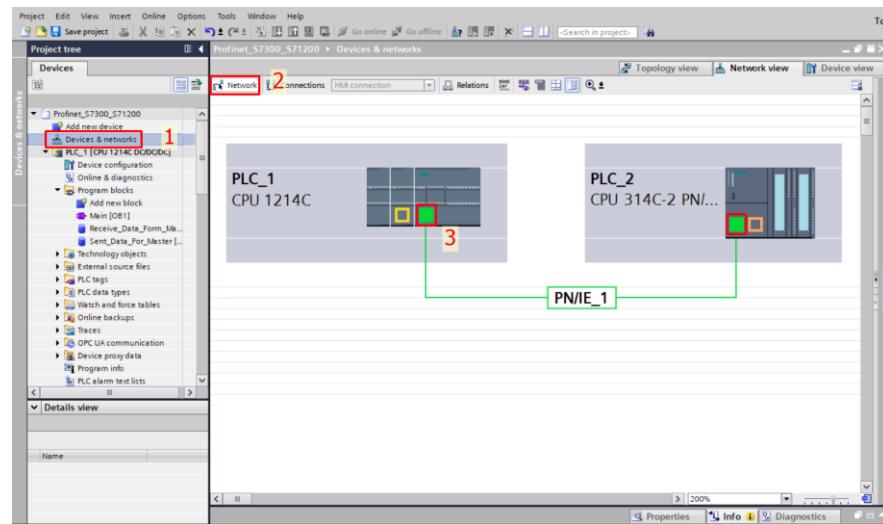


Đặt địa chỉ cho Master S7-300

Ví dụ đặt PLC S7-300 địa chỉ 192.168.0.10



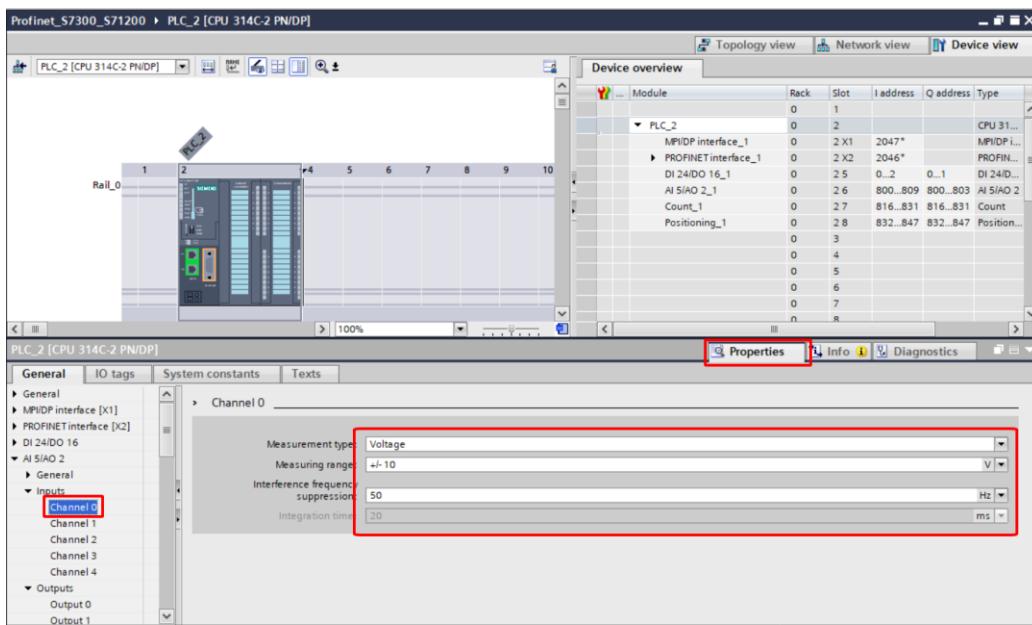
Kết nối phần cứng thiết bị



Bước 9 : Cấu hình analog cho PLC

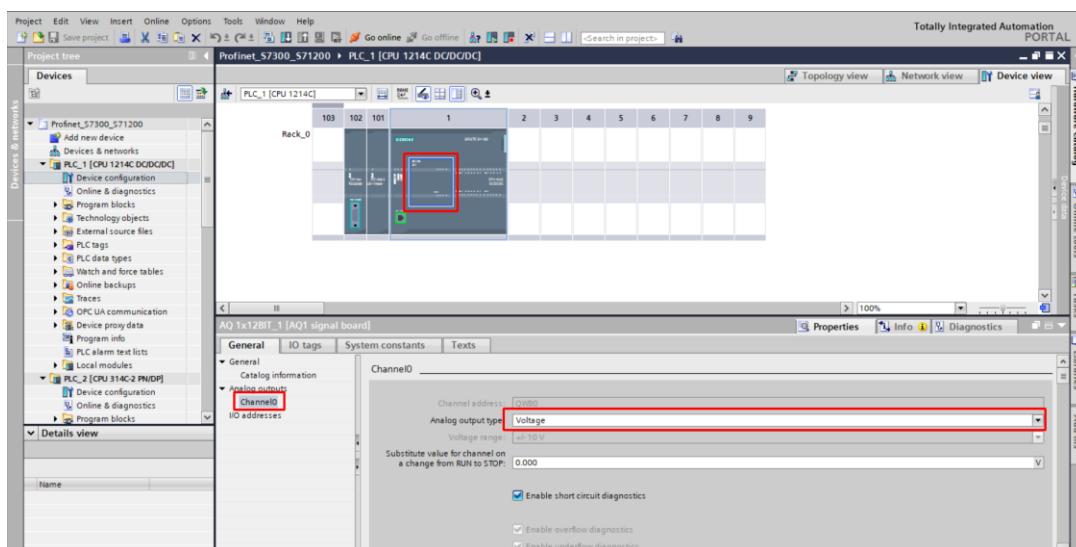
* PLC S7-300 :

Cài đặt kênh AI0 đọc chế độ điện áp 0-10VDC



* PLC S7-1200 :

Đặt analog output kênh 0 xuất giá trị áp 0-10VDC



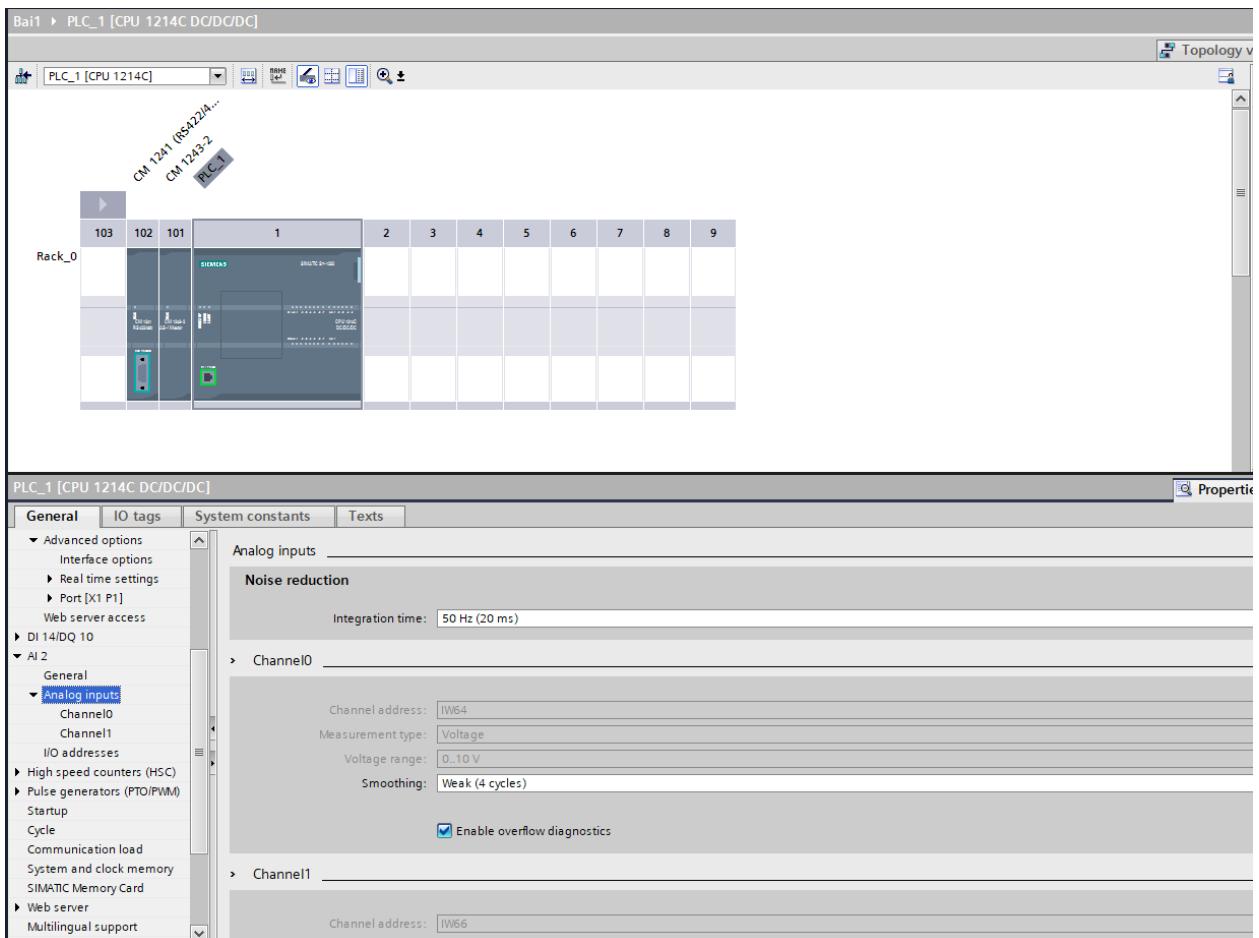
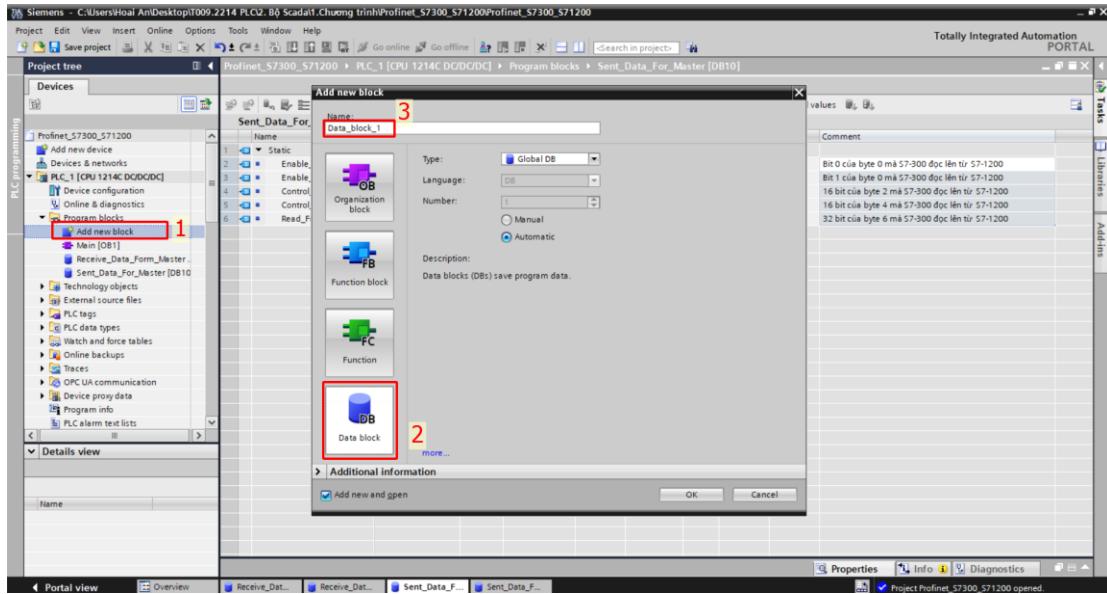


Figure 1 Không có analog output (đã cài có device)

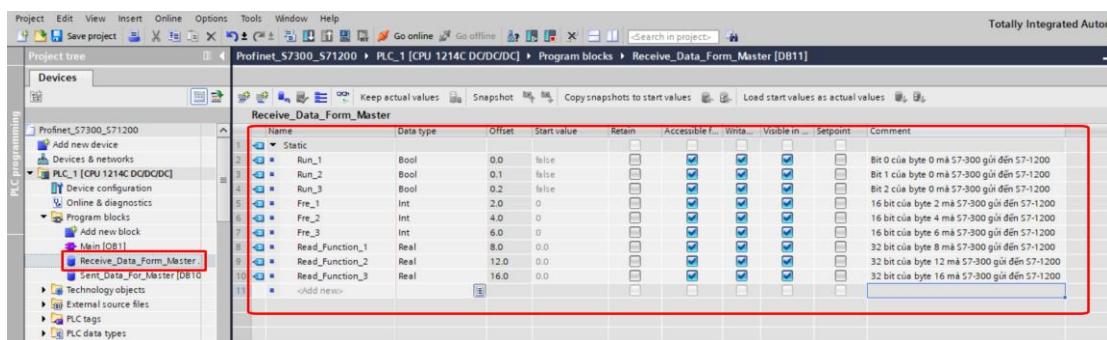
Bước 10 : Khai báo biến :

Tạo datablock và khai báo các biến sử dụng

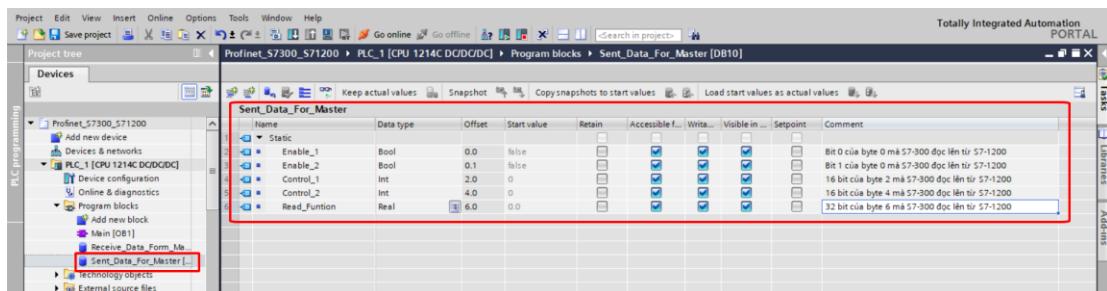


Khai báo biến trên S7-1200

Tạo data block Receive_Data_Form_Master để nhận giá trị từ S7-300 gửi đến S7-1200

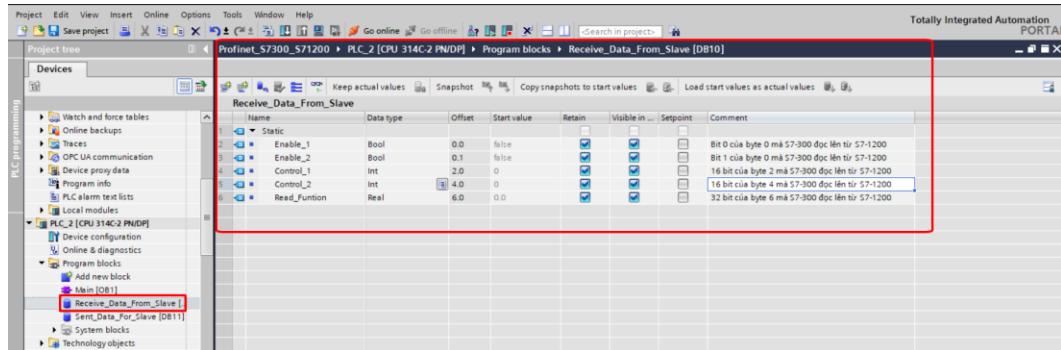


Tạo data block Sent_Data_For_Master để gửi giá trị từ S7-1200 gửi đến S7-300

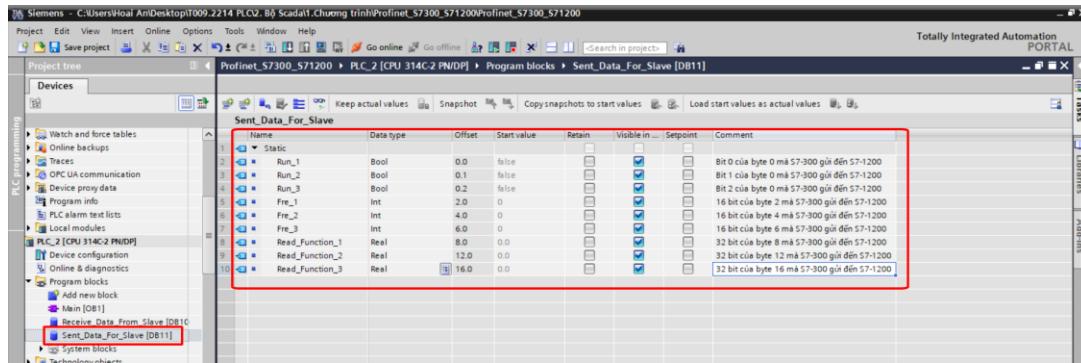


Khai báo biến trên S7-300

Tạo data block Receive_Data_From_Slave để đọc giá trị từ S7-1200 gửi đến S7-300



Tạo data block Sent_Data_For_Slave để gửi giá trị từ S7-300 gửi đến S7-1200

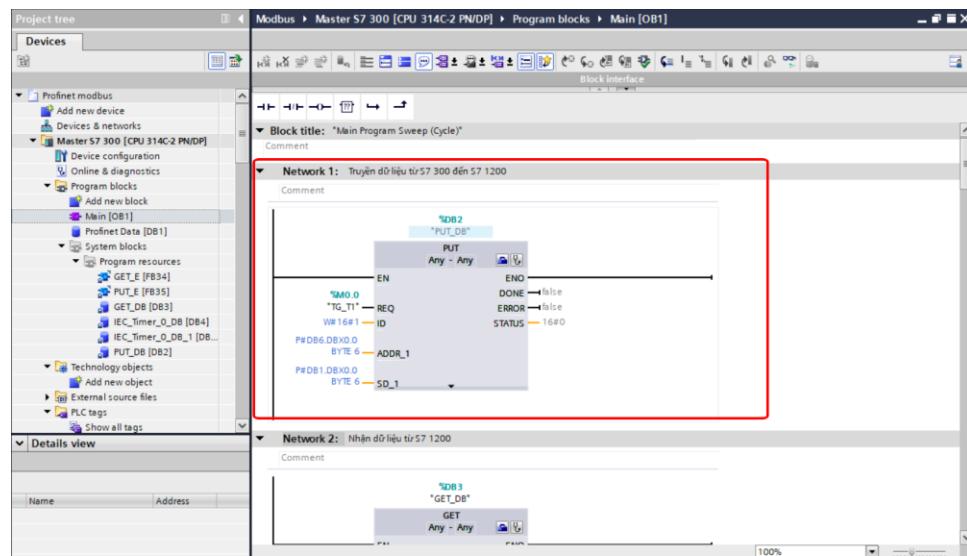


Bước 11 : Lập chương trình trên máy tính, tải chương trình cho PLC

Bước 12 : Chạy chương trình :

Sử dụng lệnh PUT để truyền giữ liệu từ S7 300 đến S7 1200

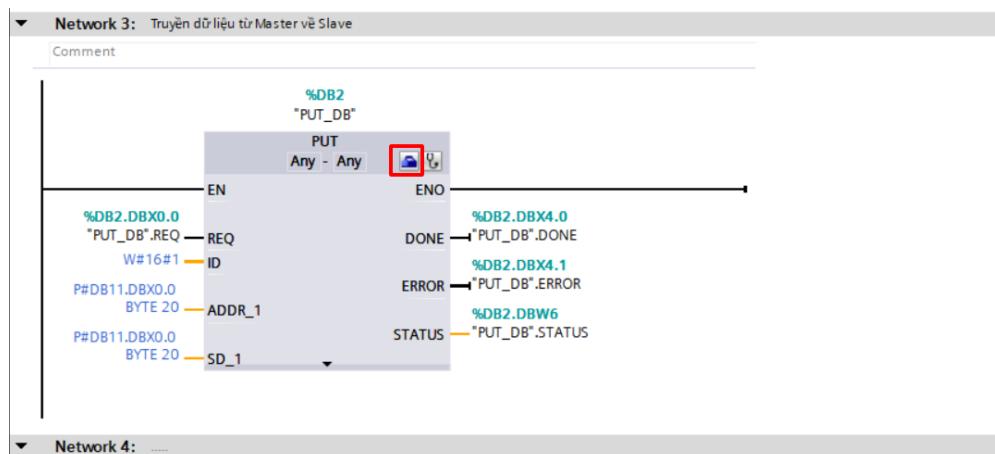
Với PLC S7-300 hay S7-1200 đều có thể làm master hay slave. Với bài toàn này ta để S7-300 làm Master con S7-1200 làm slave



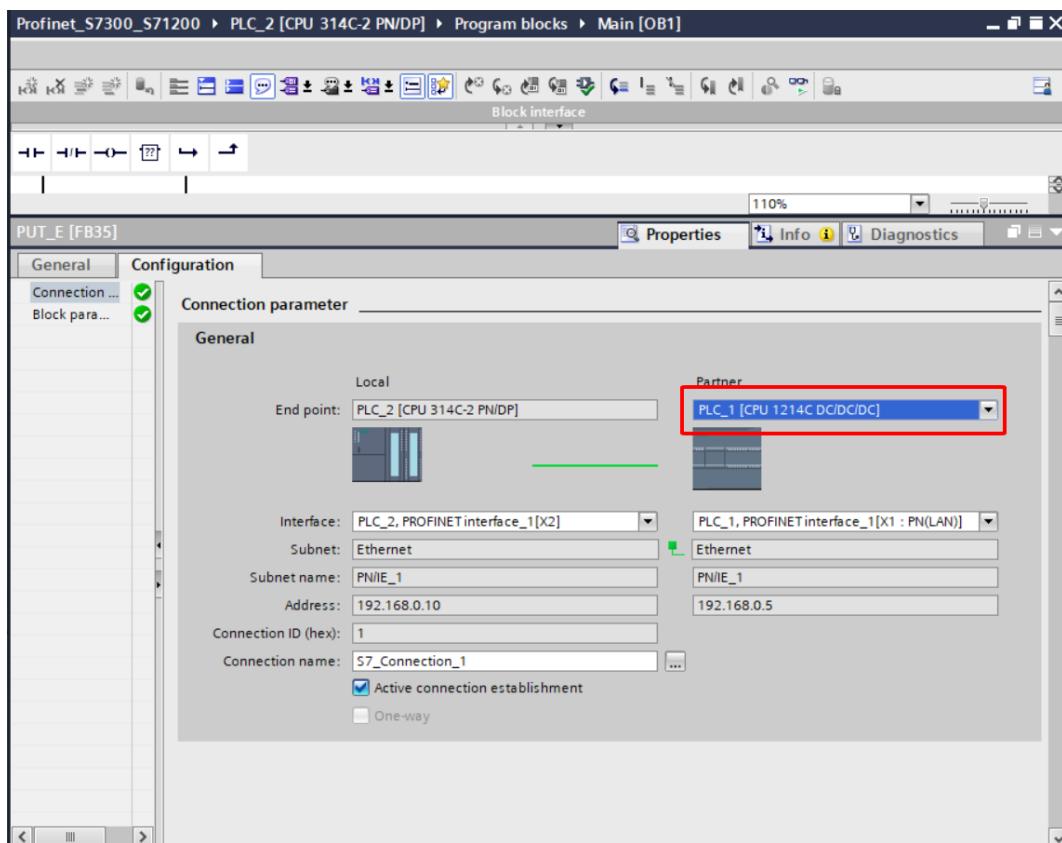
+ REQ : Tần số quét (Chu kì sau bao lâu lệnh này sẽ được thực thi)

+ ID : Địa chỉ. ID sẽ được tự động định danh bằng cách :

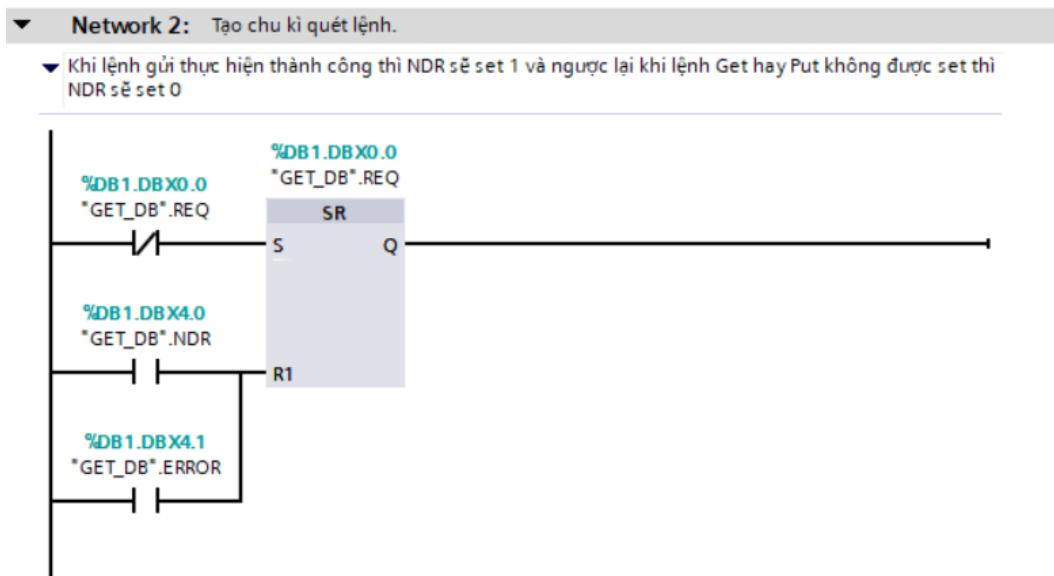
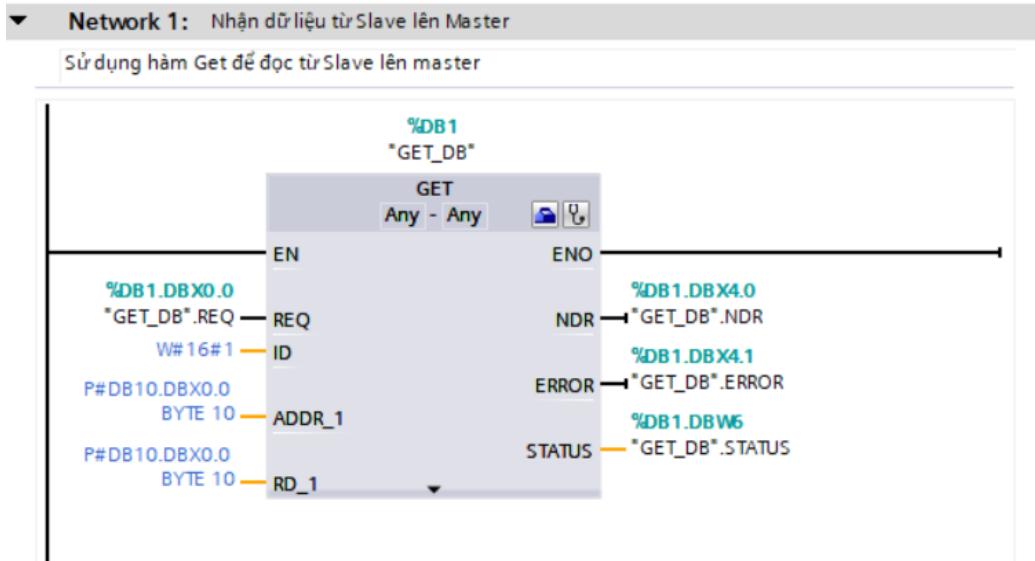
Tích vào biểu tượng



Chọn PLC để truyền thông

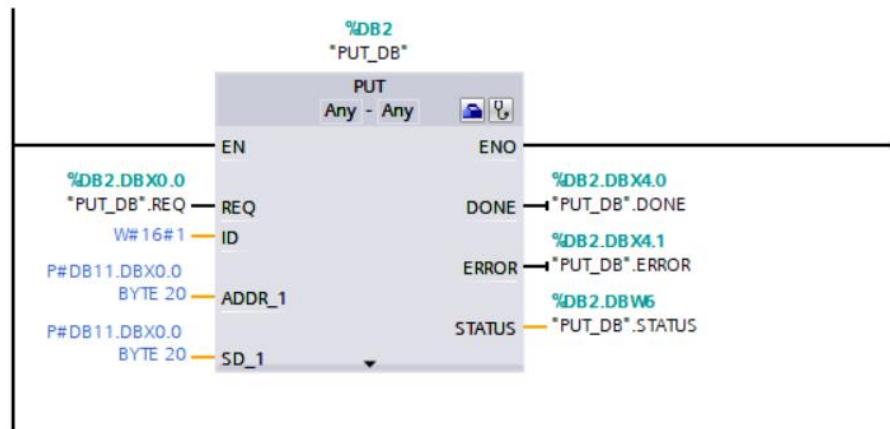


- + ADR_1 : Địa chỉ dữ liệu của Slave
 - + SD_1 : Địa chỉ dữ liệu của Master
 - + Done : Trạng thái lệnh đã thực thi thành công
 - + Error : Báo lỗi
 - + Status : Mã Lỗi
- Sử dụng lệnh GET để thu thập giữ liệu từ PLC S7 1200 gửi đến PLC S7 300



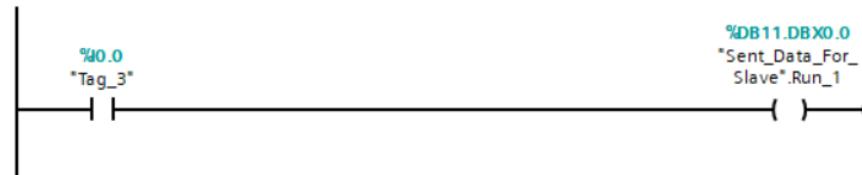
Network 3: Truyền dữ liệu từ Master về Slave

Sử dụng hàm Put để gửi dữ liệu từ Master đến Slave



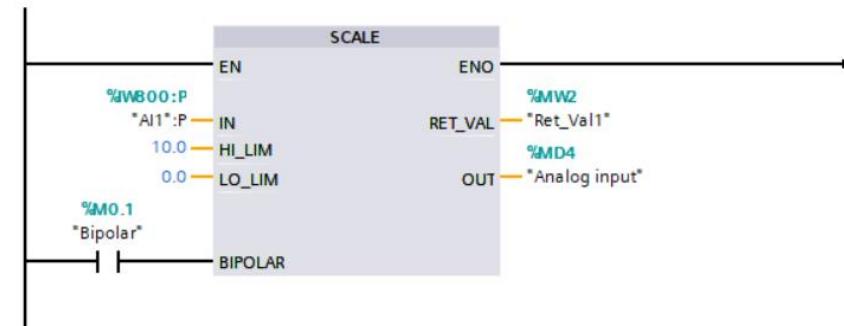
Network 5: Truyền tín hiệu bool

Khi I0.0 trên S7-300 được set thì set 1 tín hiệu bool đến S7-1200. Khi bit DB11.DBX0.0 của S7-300 On tương ứng DB11.DBX0.0 của S7-1200 On



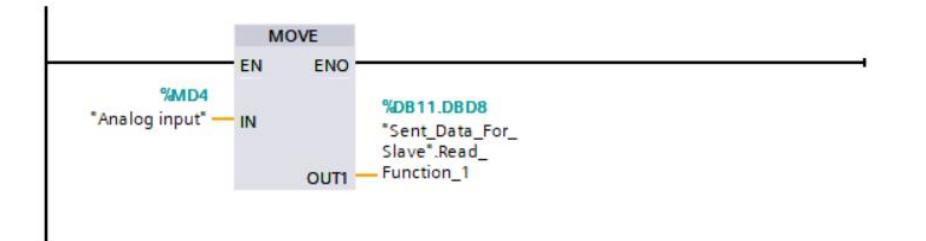
Network 7: Đọc analog 0-10V trên kênh 1

Comment

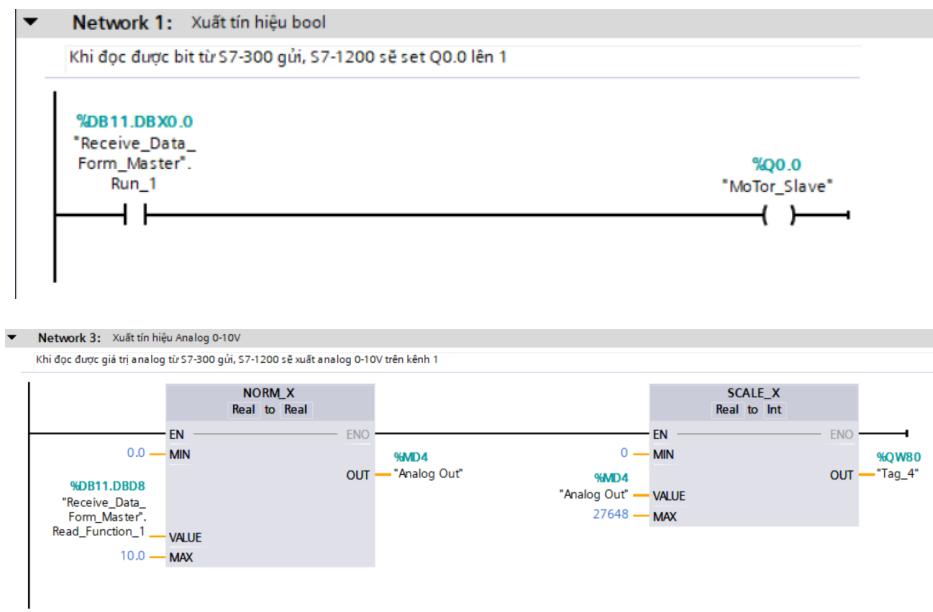


Network 8: Gửi giá trị analog đọc được từ Master đến Slave

Gửi giá trị analog đọc được trên kênh 1 của S7-300 đến S7-1200 bằng việc gửi giá trị đến thanh ghi DB11.D8D8 của S7-300 sẽ tương ứng thanh ghi DB11.D8D8 của S7-1200



Chương trình PLC S7-1200



Kết luận

Q0.0 không sáng

BÀI 2 : THỰC HÀNH LẬP TRÌNH TRUYỀN THÔNG PROFIBUS S7-300 VÀ ET-200

I. Nội dung thực hành :

Trong bài thí nghiệm này sinh viên cần thực hành và nắm được 2 nội dung chính :

Nội dung 1 : Kết nối và cấp nguồn thiết bị

Nội dung 2 : Lập trình điều khiển truyền thông Profibus giữa PLC S7-300 và ET-200

II. Bài toán :

Lập trình theo bài toán sau

+ Tác động I0.0 trên S7-300 bằng công tắc lên 1, đèn sáng bằng việc ET-200 set Q0.0 lên 1 (S7-300 truyền tín hiệu số cho ET-200)

+ Tác động I0.0 trên ET-200 bằng công tắc lên 1, đèn sáng bằng việc S7-300 set Q0.0 lên 1 (ET-200 truyền tín hiệu số cho S7-300)

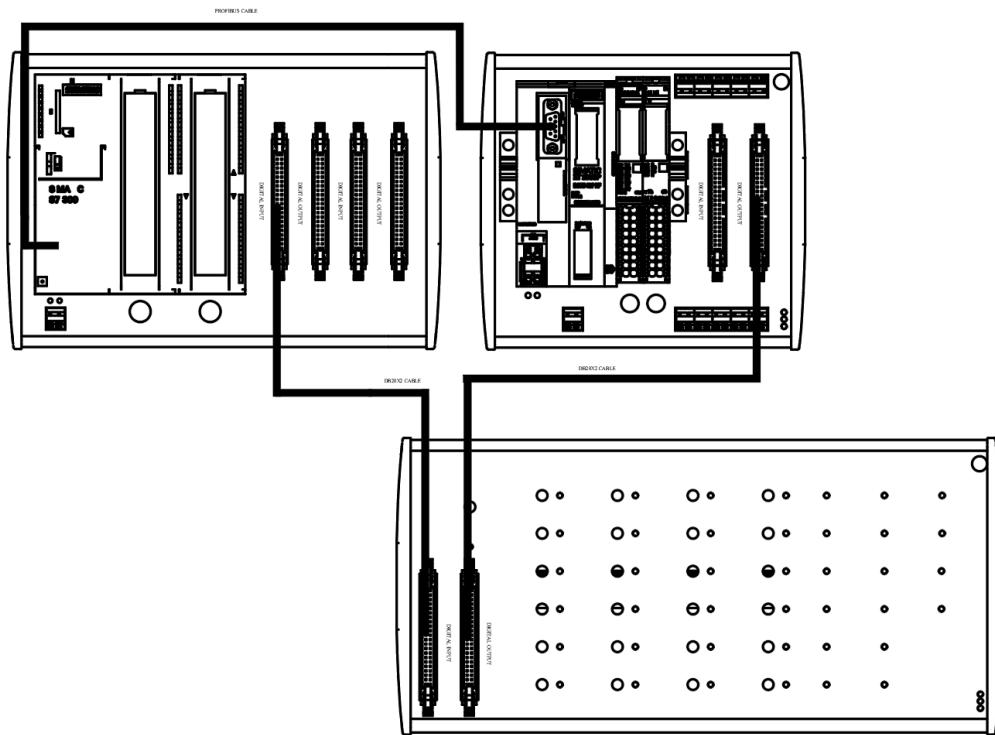
III. Thực hành :

- Các bước thực hành :

Bước 1 : Kết nối Digital input của module mô phỏng vào ra số với digital input của S7-300 bằng cáp DB20x2

Bước 2 : Kết nối Digital Output của ET-200 với digital output của module mô phỏng vào ra số

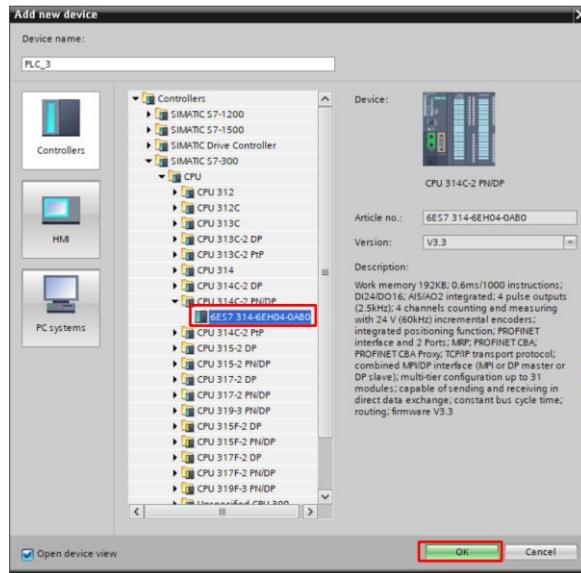
Bước 3 : Cấp nguồn cho thiết bị, cáp Ethernet giữa máy tính với PLC và cáp DB9 Profibus giữa PLC S7-300, ET-200



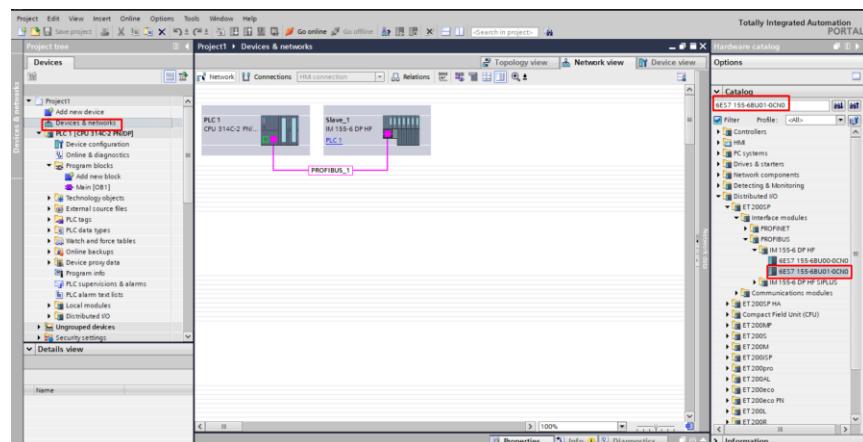
Bước 4 : Khai báo và kết nối phần cứng

* **Thêm phần cứng :**

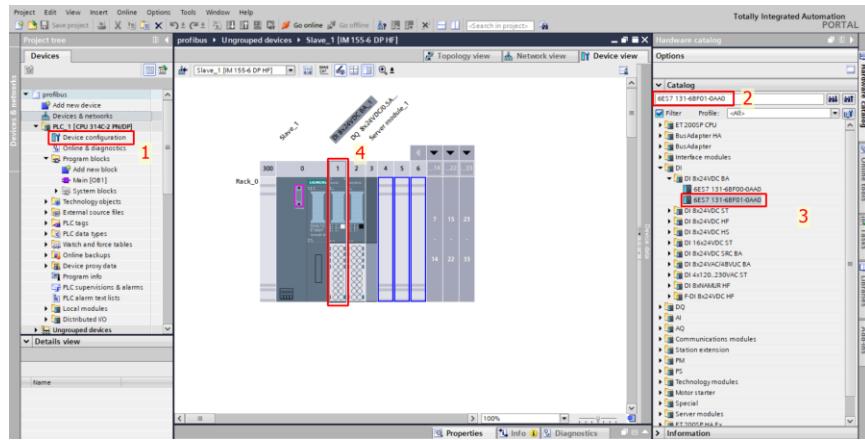
- Thêm PLC S7 300. Chọn PLC 6ES7 314-6EH04-0AB0 ver3.3



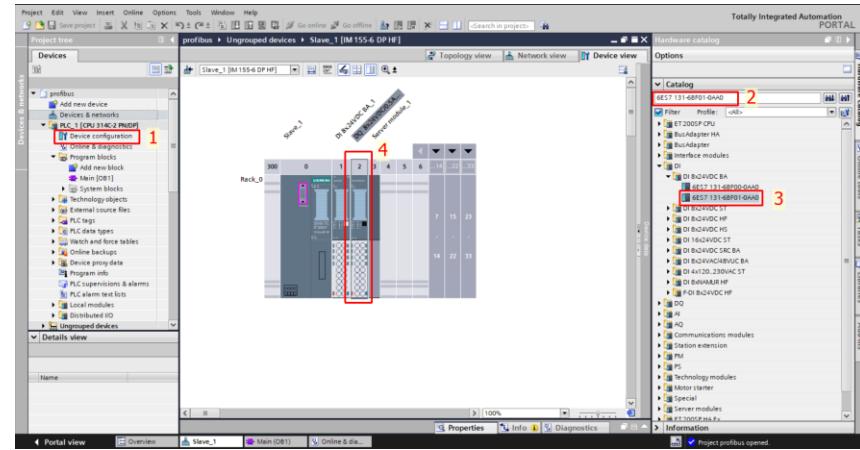
- Thêm ET 200. Vào mục Hardware Catalog điền mã 6ES7 155-6BU01-OCN0 để lấy module



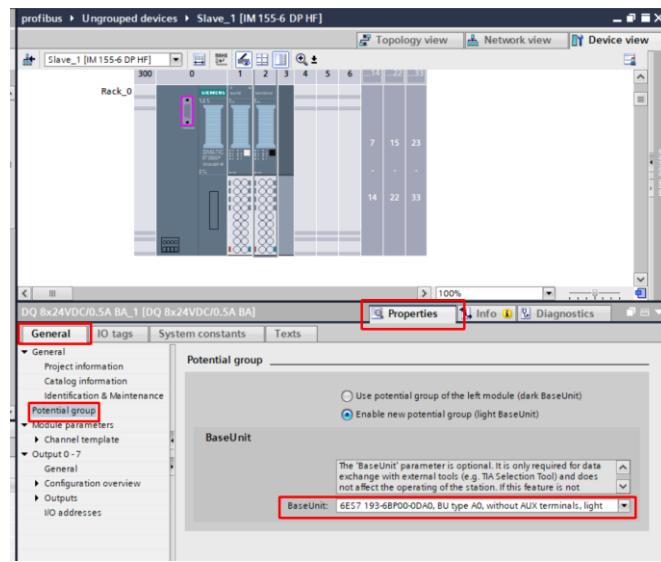
- Thêm module digital input cho ET-200. Mã module 6ES7 131-6BF01-0AA0



- Thêm module digital output cho ET200. Mã module 6ES7 132-6BF01-0AA0.



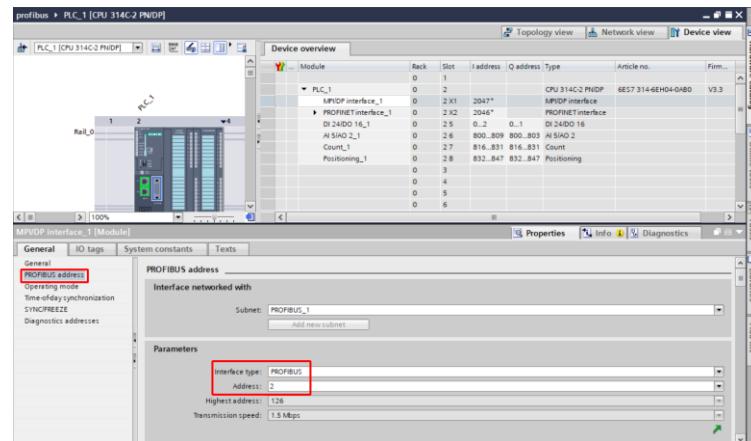
- Cài đặt base cho module I/O của ET 200. Ta chọn base trắng với mã như hình



* Cài đặt địa chỉ profibus

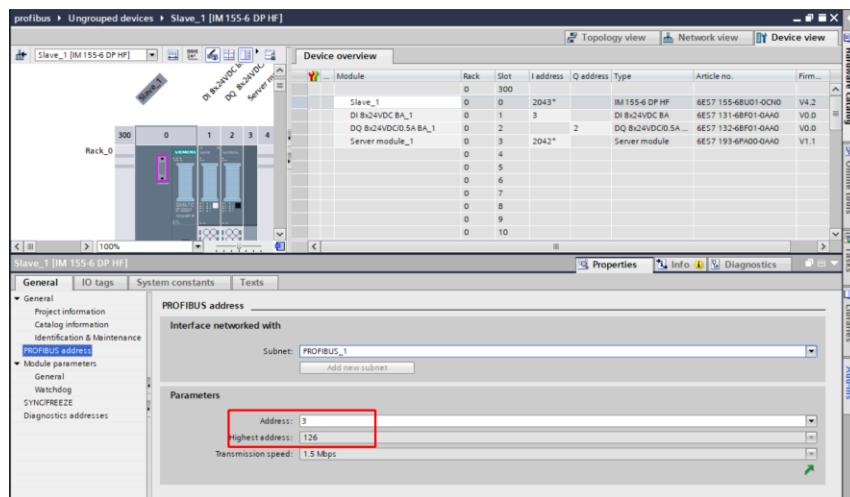
Địa chỉ profibus S7-300.

Ví dụ ta cài địa chỉ là 2

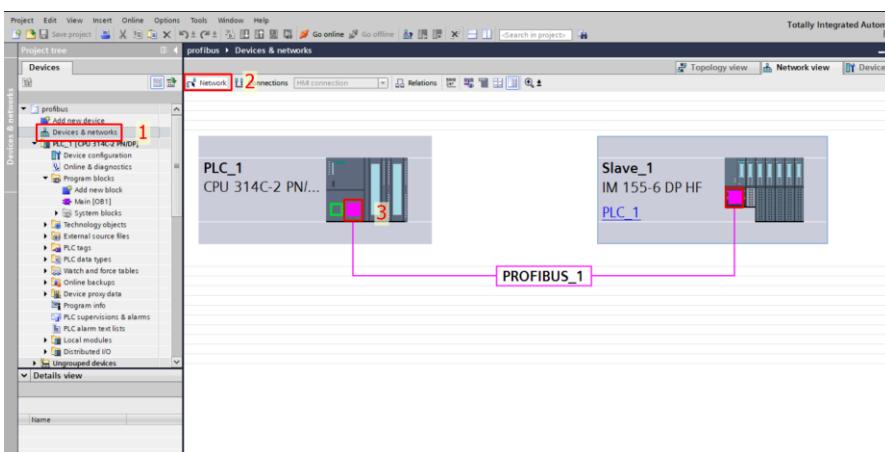


Địa chỉ profibus ET-200 :

Ví dụ đặt địa chỉ là 3. Địa chỉ phải khác với địa chỉ module truyền thông profibus của S7-300

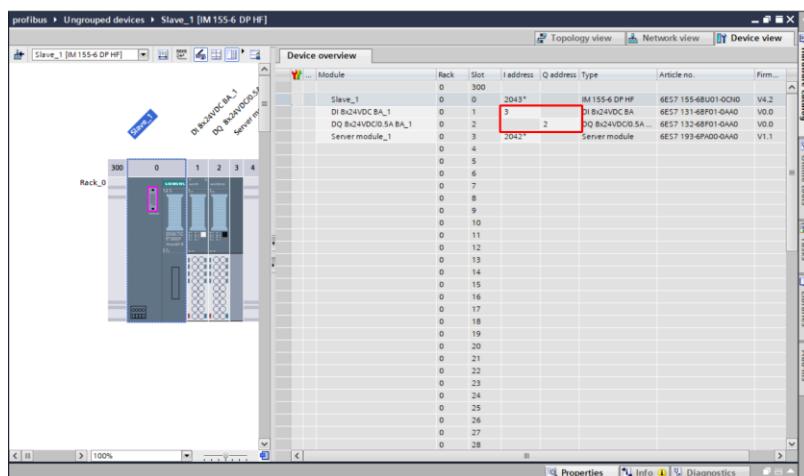


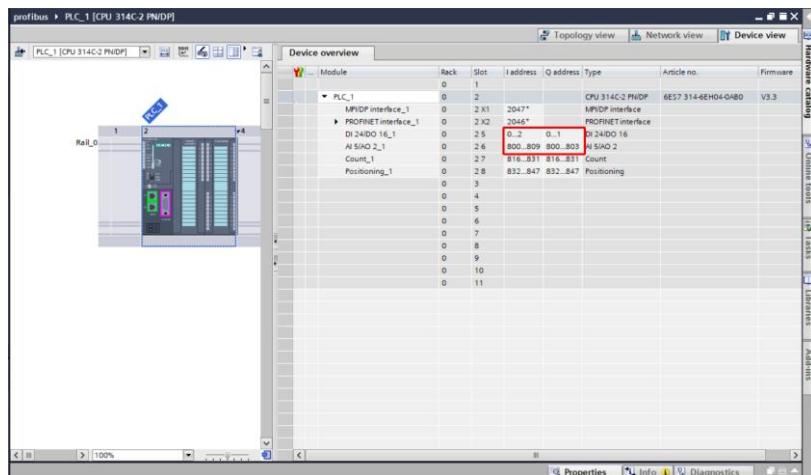
Khai báo kết nối thiết bị



Bước 5 : Thiết lập I/O

Địa chỉ I/O này có thể thay đổi tùy ý bằng cách nhấp vào và gõ giá trị. Lưu ý phải đảm bảo không được trùng giá trị địa chỉ với thanh ghi phần cứng khác





Bước 6 : Khai báo biến

Vào Default tag table khai báo các biến như hình dưới

Name	Data type	Address	Retain	Access	Visible	Comment
1 Input 57 300	Bool	%IO.0				Dầu vào đầu tiên của ET200
2 Output ET200	Bool	%Q2.0				Dầu ra đầu tiên của ET200
3 Input ET200	Bool	%I3.0				Dầu ra đầu tiên của ET200
4 Output 57 300	Bool	%IO.0				Dầu ra đầu tiên của 57-300
5 Analog_input_1	Int	%IW800				Ngô vào analog kênh 1 của 57-300
6 Analog_input_2	Int	%IW802				Ngô vào analog kênh 2 của 57-300
7 Analog_Output_1	Int	%OW800				Ngô ra analog kênh 1 của 57-300
8 Analog_Output_2	Int	%OW802				Ngô ra analog kênh 2 của 57-300
9 Real_Analog_Input_1	Real	%AD0				Thánh ghi trung gian để lưu giá trị analog đọc được kênh 1
10 Test	Bool	%IA.0				
11 Ret_val_L_1	Word	%MW8				
12 Ret_Out_1	Real	%AD8				
13 Ret_Val_L_out_1	Word	%MW12				
14 Ret_val_L_2	Word	%MW18				
15 Real_Analog_Input_2	Real	%AD14				
16 Test_Out_2	Real	%AD20				
17 Ret_Val_L_out_2	Word	%MW24				
18 -vdd netcc						

Bước 7 : Lập chương trình trên máy tính, tải chương trình cho PLC

Bước 8 : Chạy chương trình :

Chương trình PLC S7-300



Kết luận

Q0.0 không sáng

Q1.0 không sáng

BÀI 3 : THỰC HÀNH LẬP TRÌNH TRUYỀN THÔNG ASI S7-1200 VÀ ASI SLAVE

I. Nội dung thực hành :

Trong bài thí nghiệm này sinh viên cần thực hành và nắm được 2 nội dung chính :

Nội dung 1 : Kết nối và cấp nguồn thiết bị

Nội dung 2 : Lập trình điều khiển truyền thông ASI giữa PLC S7-1200 và ASI Slave

II. Bài toán :

Lập trình theo bài toán sau

+ Tác động nút nhấn 1 contactor đóng, tác động nút nhấn 2 contactor trở về trạng thái ban đầu

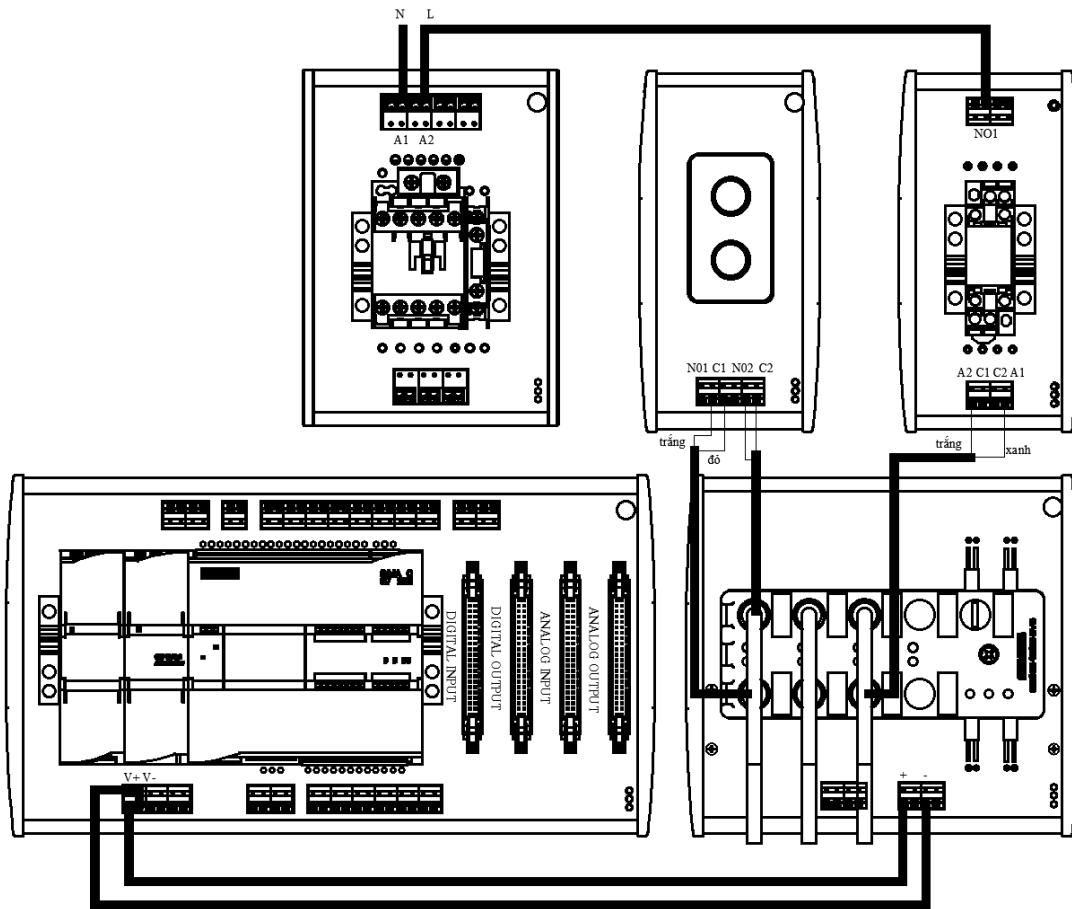
III. Thực hành :

- Các bước thực hành :

Bước 1 : Kết nối cáp từ nút nhấn 1 và 2 đến input 1 và 2 của ASI Slave

Bước 2 : Kết nối cáp từ đầu ra output 1 của ASI Slave đến module relay

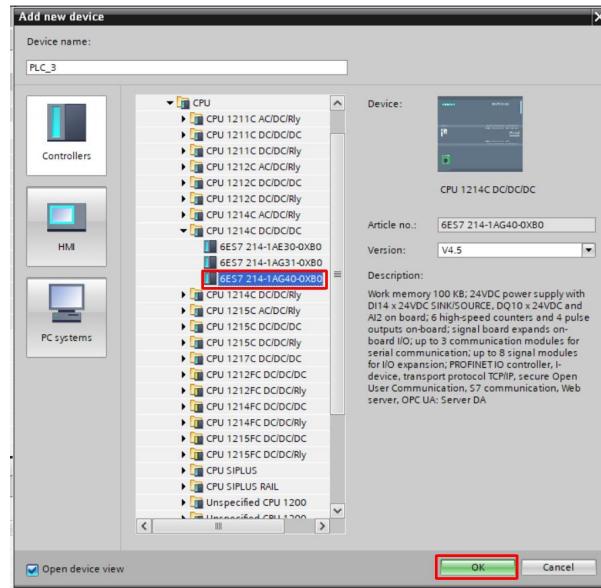
Bước 3 : Cấp nguồn cho thiết bị, cáp Ethernet giữa máy tính với PLC và cáp module mở rộng ASI trên S7-1200 đến ASI Slave



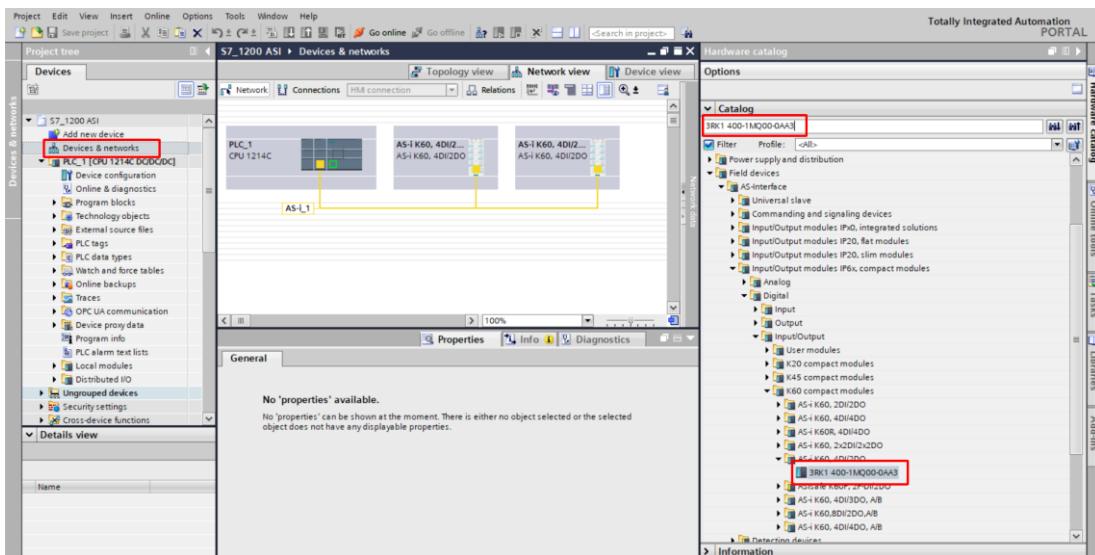
Bước 4 : Khai báo và kết nối phần cứng

* Thêm phần cứng

- Thêm PLC S7 1200. Chọn PLC 6ES7 214-1AG40-0XB0 ver 4.5



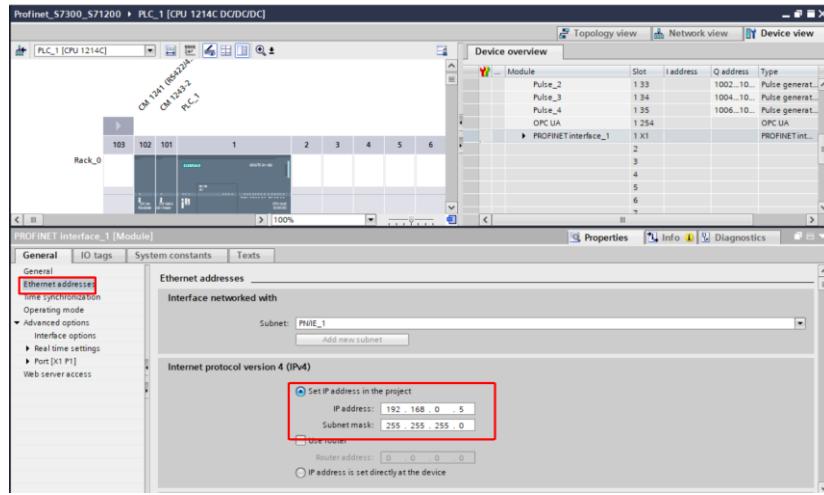
- Thêm module ASI Slave :



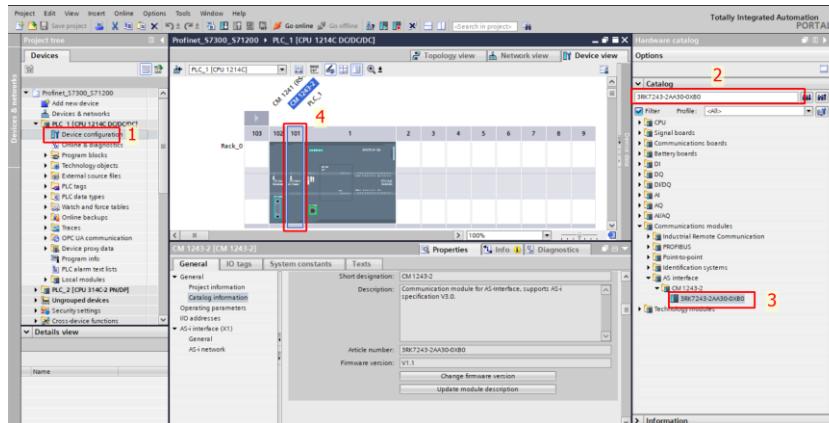
* Cài đặt địa chỉ cho PLC

Đặt địa chỉ cho Slave S7-1200 :

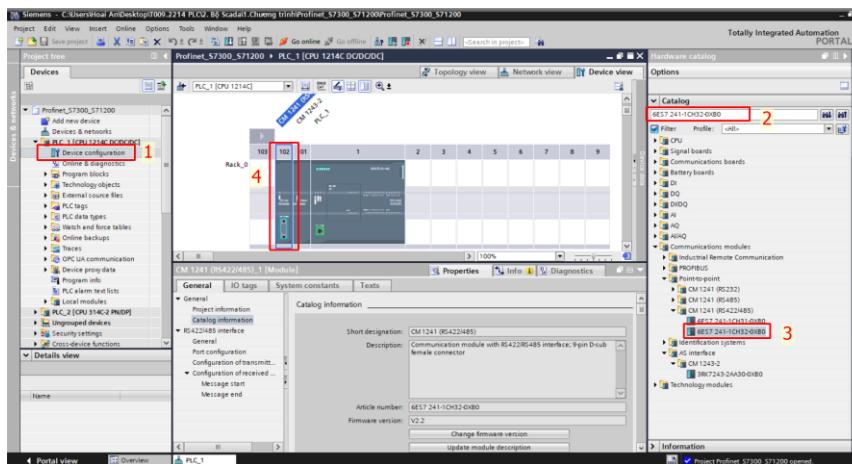
Ví dụ : đặt địa chỉ IP S7-1200 là 192.168.0.5



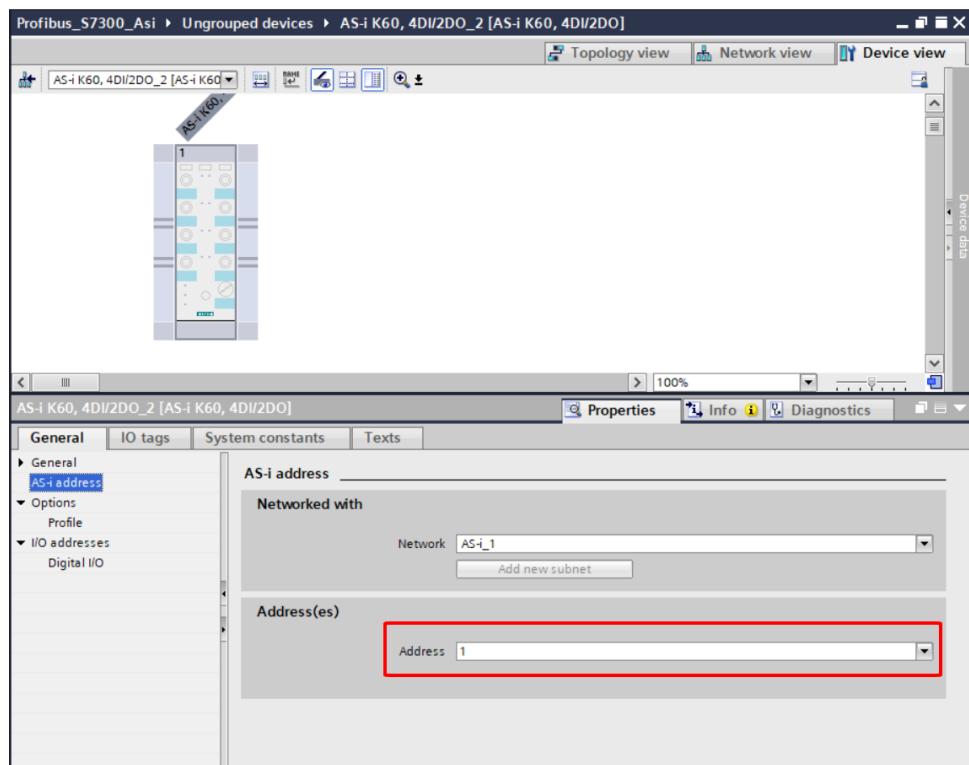
Khai báo module truyền thông ASI. Tại mục Hardware Catalog ta nhập mã module 3RK7243-2AA30-0XB0 để tạo module.



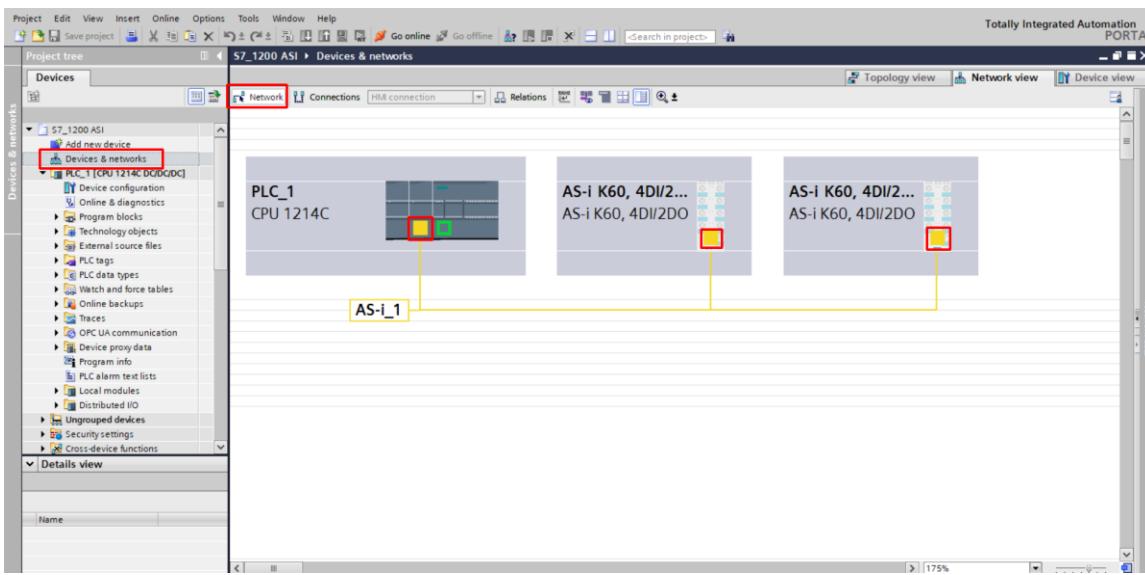
Khai báo module truyền thông Modbus. Tại mục Hardware Catalog ta nhập mã module 6ES7 241-1CH32-0XB0 để tạo module.



Cài địa chỉ ASI Slave

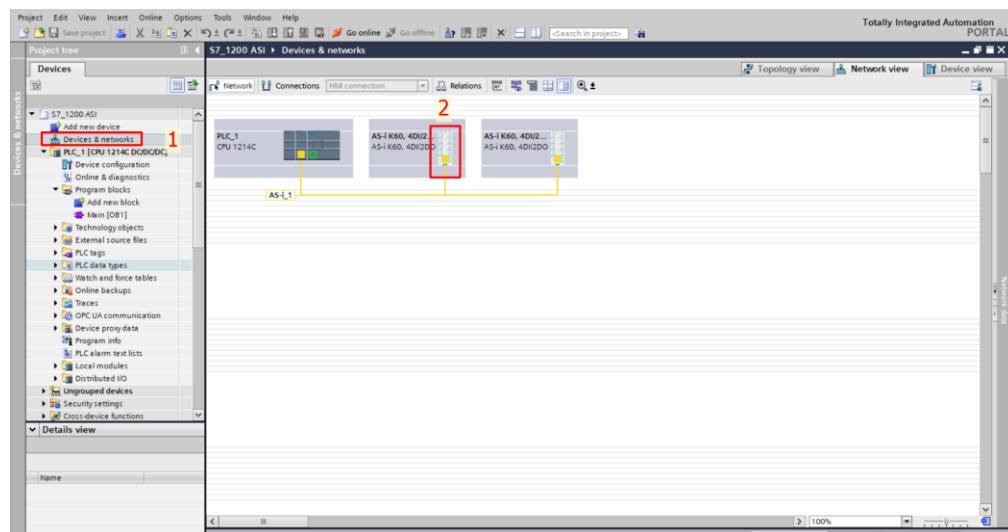


Khai báo kết nối thiết bị

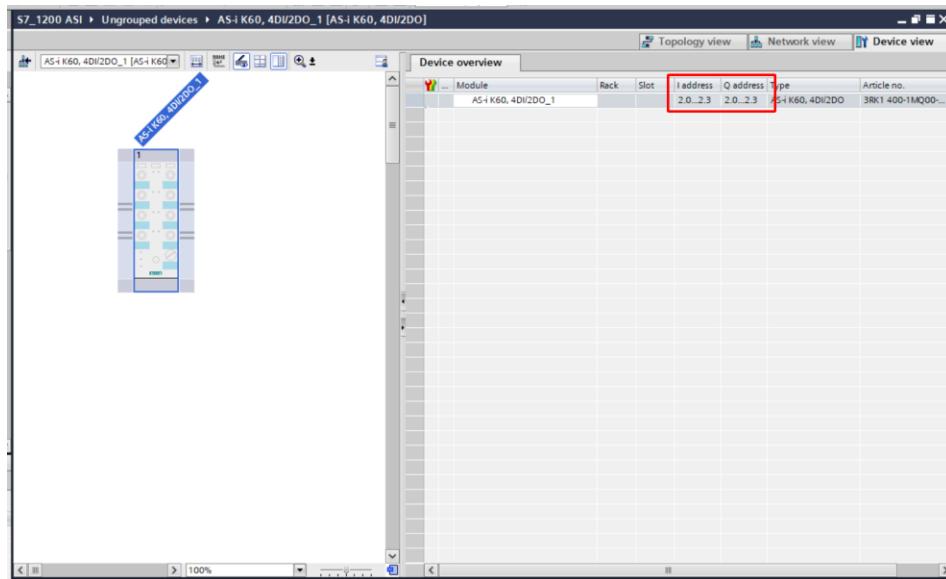


Bước 5 : Thiết lập I/O

Vào giao diện cài đặt ASI Slave



Địa chỉ này có thể thay đổi tùy người dùng. Ví dụ như hình ASI Slave đang có địa chỉ đầu vào từ I2.0 đến I2.3 và đầu ra từ Q2.0 đến Q2.3.



Bước 6 : Lập chương trình trên máy tính, tải chương trình cho PLC

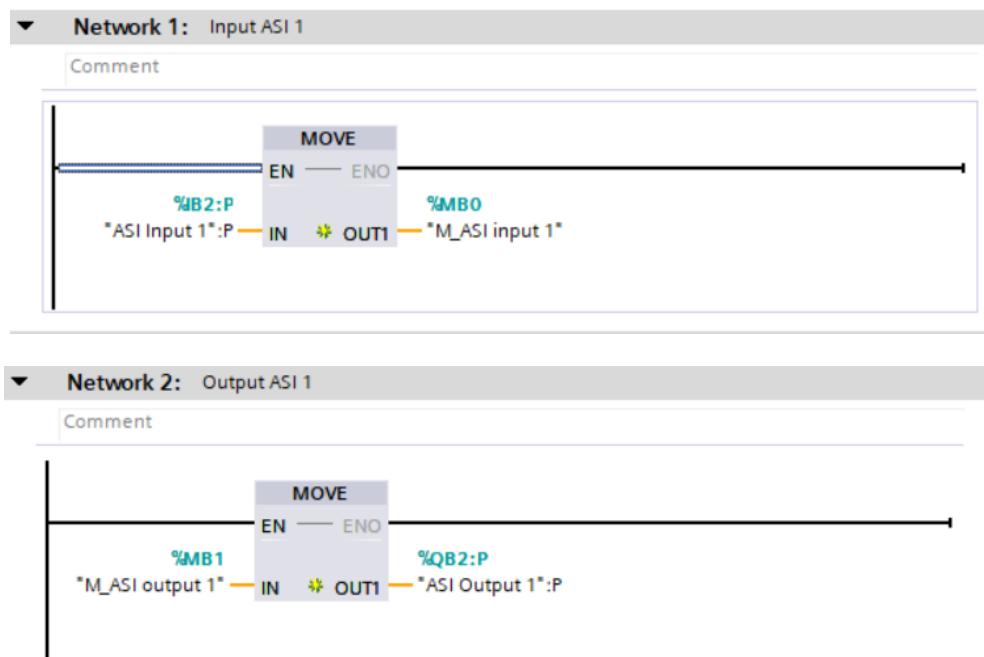
Bước 7 : Chạy chương trình :

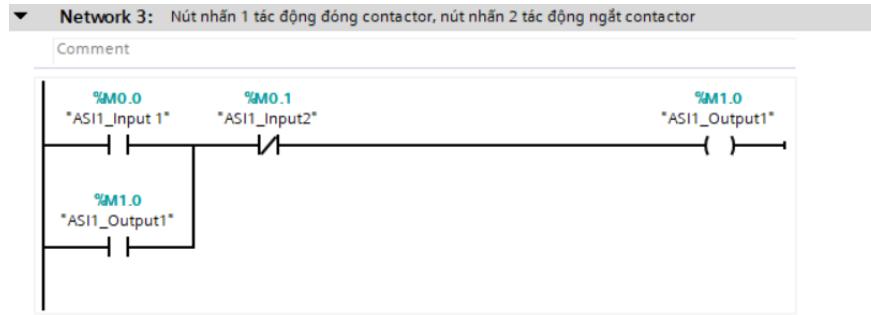
Chương trình PLC S7-1200

Lần đọc trạng thái đầu vào của ASI-Slave. Các đầu vào ASI-Slave bắt đầu từ I2.0 đến I2.3

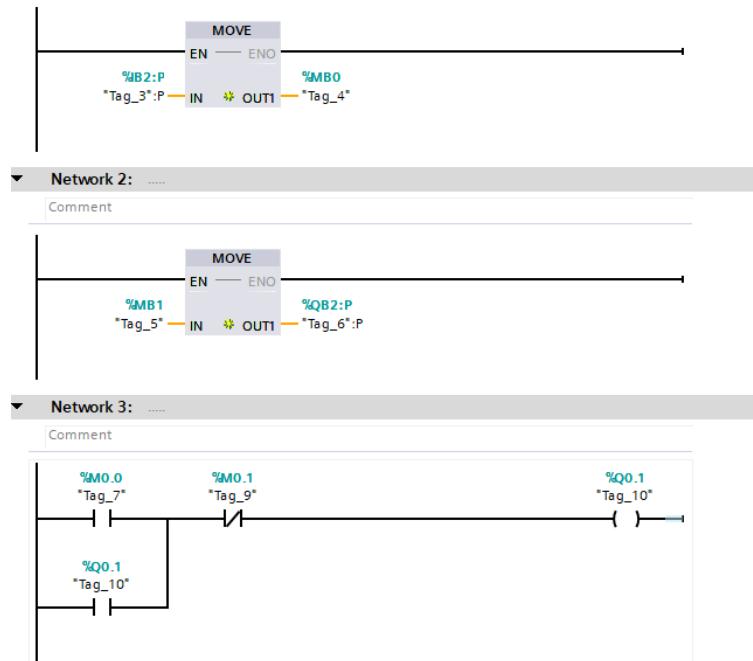
Ta sử dụng hàm %IB2:P để chuyển 1 byte bắt đầu từ I2.0 đến I2.7 sang thanh ghi MBO (M0.0 đến M0.7). Tức là khi giá trị I2.0 đến I2.7 thay đổi giá trị thì giá trị thanh ghi M0.0 đến M0.7 sẽ thay đổi tương ứng.

Cách truyền giá trị output cũng tương tự





Kết luận



Ngày 12 tháng 5 năm 2023

Buổi sáng

BÀI 4 : THỰC HÀNH LẬP TRÌNH TRUYỀN THÔNG ASI S7-300 VÀ ASI SLAVE THÔNG QUA MODULE CHUYỂN ĐỔI

I. Nội dung thực hành :

Trong bài thí nghiệm này sinh viên cần thực hành và nắm được 2 nội dung chính :

Nội dung 1 : Kết nối và cấp nguồn thiết bị

Nội dung 2 : Lập trình điều khiển truyền thông ASI giữa PLC S7-300 và ASI Slave

II. Bài toán :

Lập trình theo bài toán sau

+ Tác động nút nhấn 1 contactor đóng, tác động nút nhấn 2 contactor trở về trạng thái ban đầu

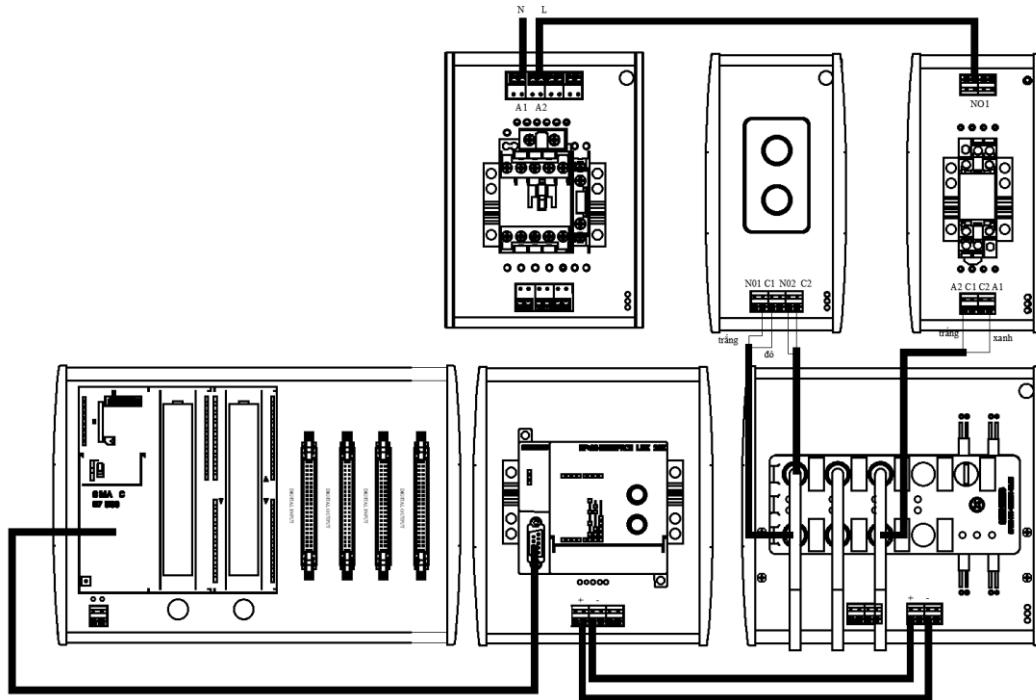
III. Thực hành :

- Các bước thực hành :

Bước 1 : Kết nối cáp từ nút nhấn 1 và 2 đến input 1 và 2 của ASI Slave

Bước 2 : Kết nối cáp từ đầu ra output 1 của ASI Slave đến module relay

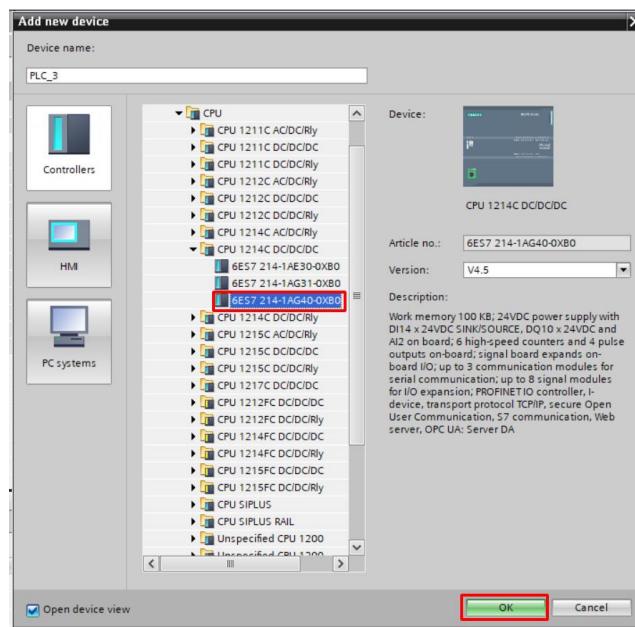
Bước 3 : Cấp nguồn cho thiết bị, cáp Ethernet giữa máy tính với PLC và cáp module S7-300 đến module chuyển đổi profibus sang ASI, cáp từ module chuyển đổi đến ASI Slave



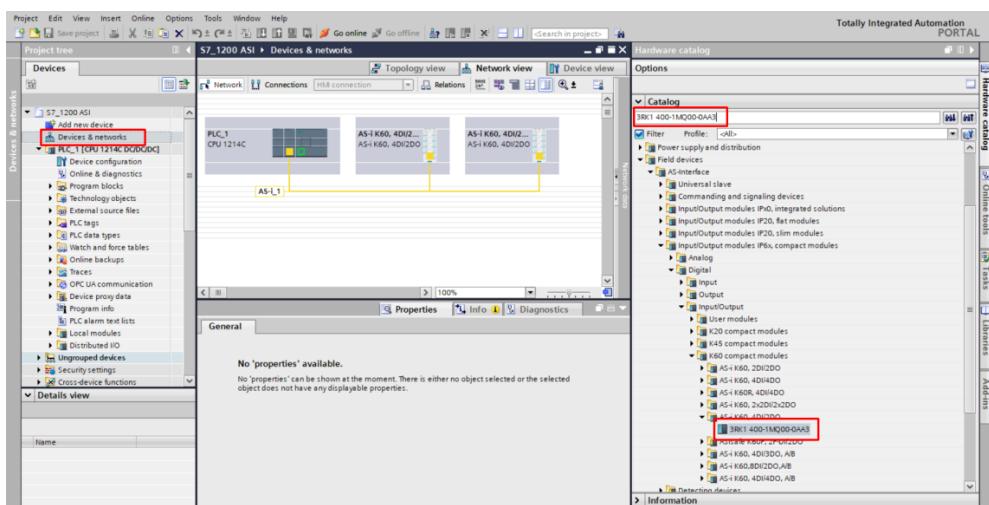
Bước 4 : Khai báo và kết nối phần cứng

* Thêm phần cứng

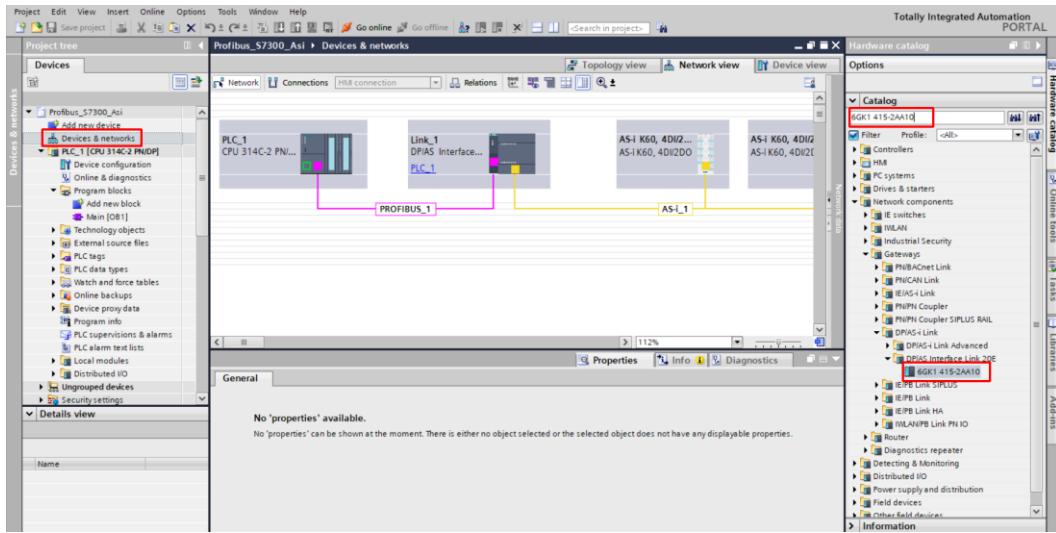
- Thêm PLC S7 1200 -> không cần thiết



- Thêm module ASI Slave :

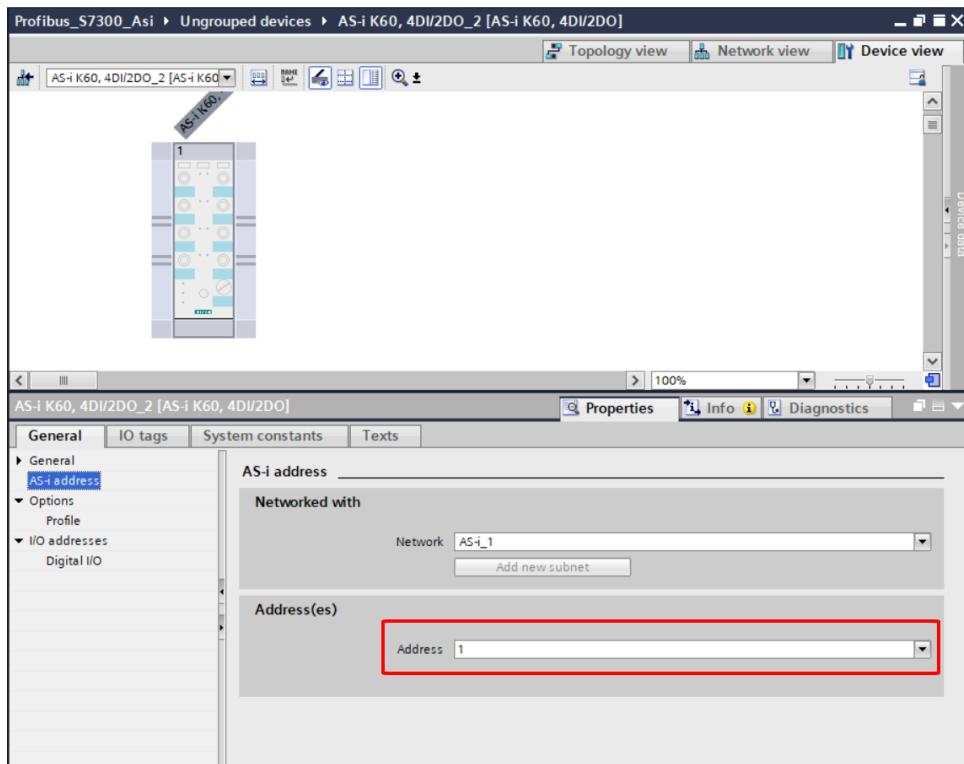


- Thêm module chuyển đổi Profibus sang ASI

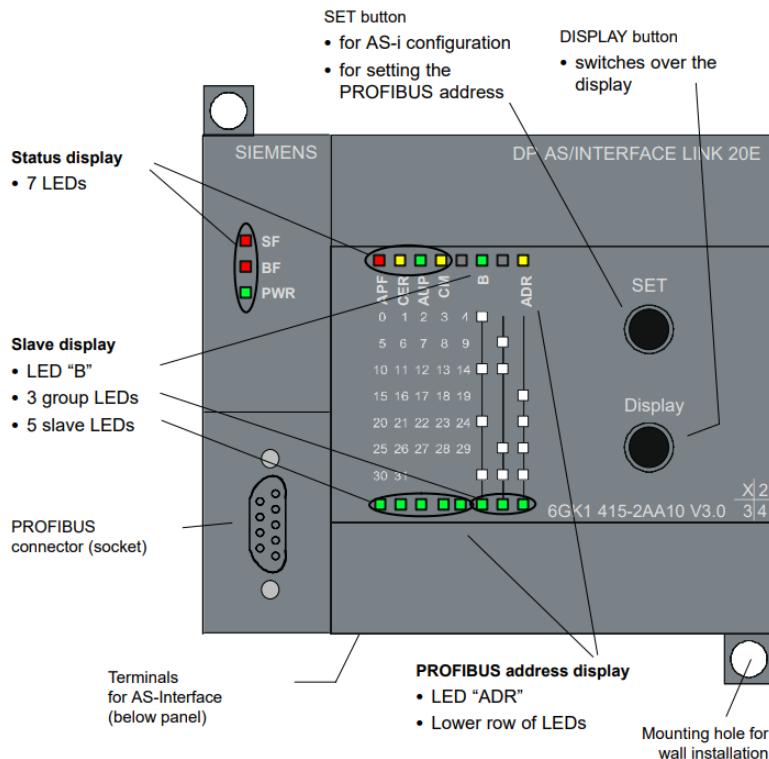
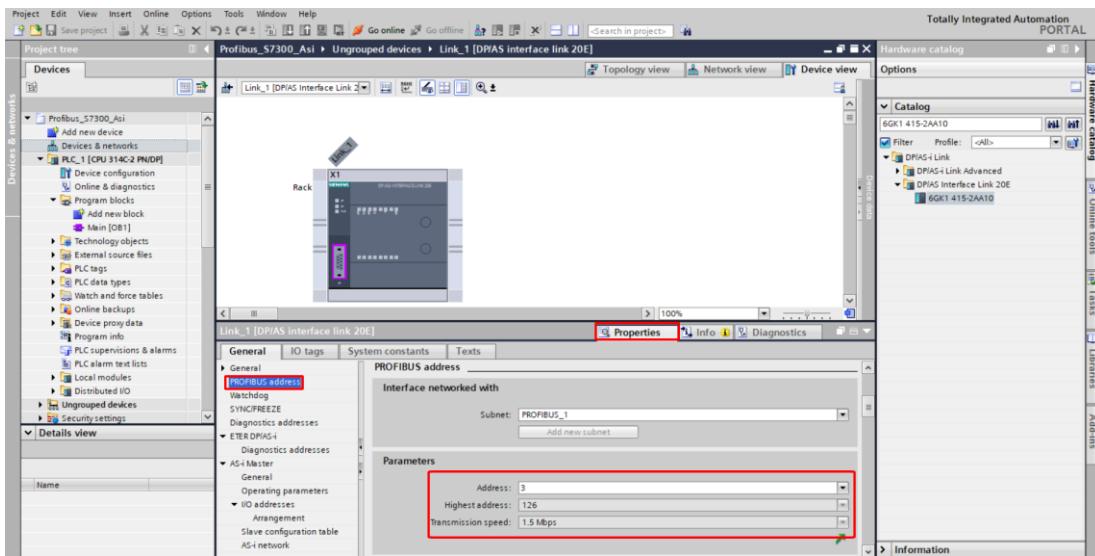


* Cài đặt địa chỉ

Cài địa chỉ ASI Slave



Cài đặt địa chỉ module chuyển đổi profibus sang ASI



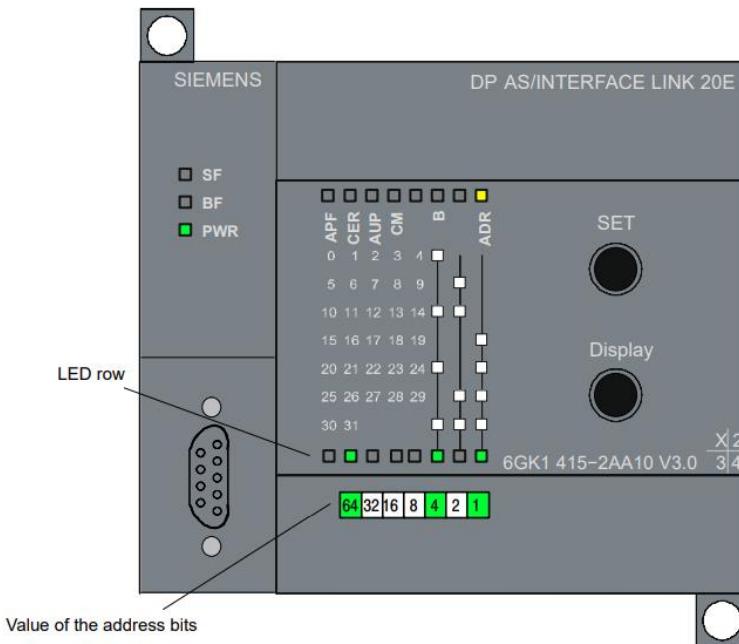
- + Ngắt kết nối với DP master (ví dụ bằng cách rút đầu nối PROFIBUS) hoặc chuyển DP master sang STOP
- + Nhấn liên tục nút “DISPLAY” cho đến khi đèn LED “ADR” sáng lên.
- + Nhấn nút “SET”, bạn có thể đặt giá trị mới cho địa chỉ
- + Đèn LED nhấp nháy (đèn LED thứ hai từ trái sang) hiển thị bit quan trọng nhất của địa chỉ PROFIBUS.

+ Khi bạn nhấn nút “SET”, bit này sẽ on (đèn LED bật).

Ngược lại, nếu bạn nhấn nút “Display”, bit sẽ OFF (đèn LED tắt). Màn hình sau đó chuyển sang đèn LED tiếp theo (đèn LED thứ ba từ bên trái) bit địa chỉ tiếp theo của địa chỉ PROFIBUS

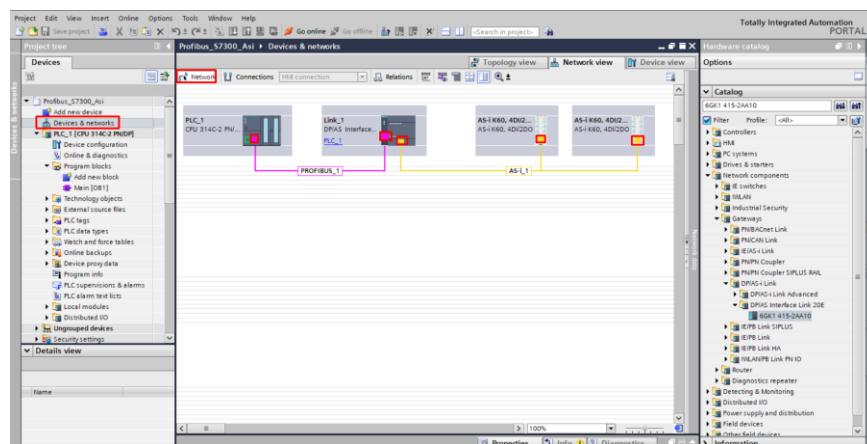
Để cài địa chỉ ta nhấn liên tục nút Display đến khi đến ADR sáng

+ Địa chỉ sẽ được tính như sau :



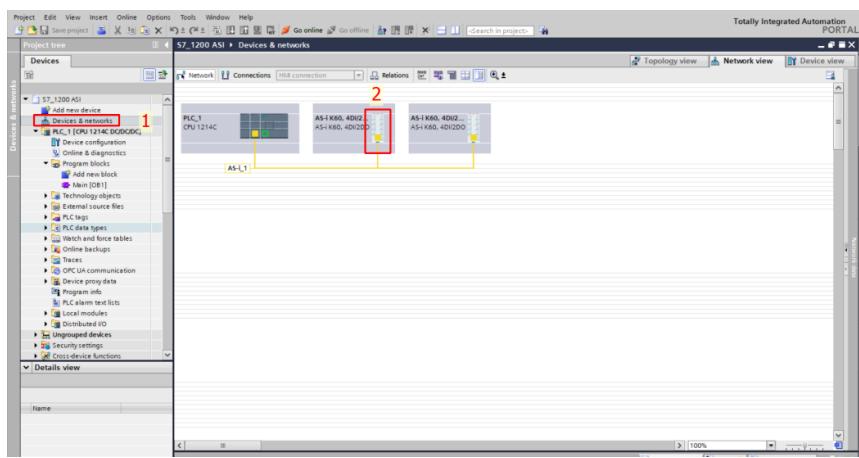
Địa chỉ = Tổng số bit sáng. Ví dụ muốn địa chỉ là 3. Ta sẽ on bit 1 và 2

Khai báo kết nối thiết bị

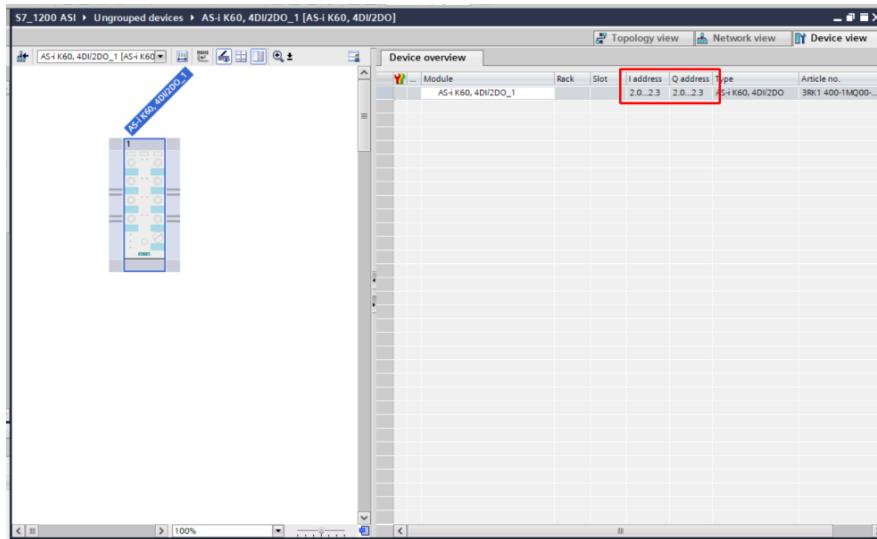


Bước 5 : Thiết lập I/O

Vào giao diện cài đặt ASI Slave



Địa chỉ này có thể thay đổi tùy người dùng. Ví dụ như hình ASI Slave đang có địa chỉ đầu vào từ I2.0 đến I2.3 và đầu ra từ Q2.0 đến Q2.3.



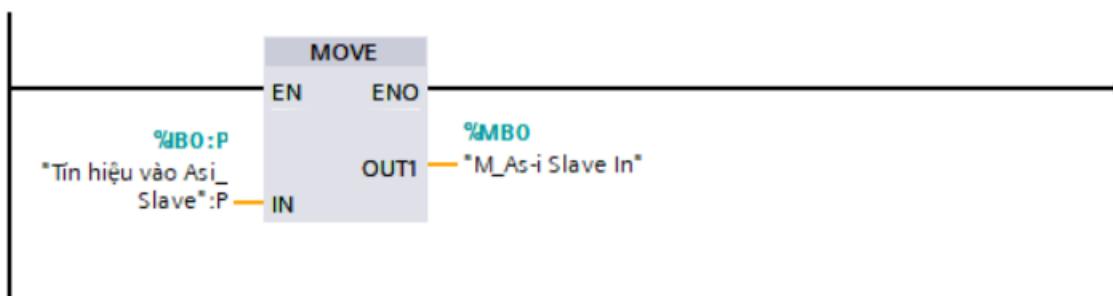
Bước 6 : Lập chương trình trên máy tính, tải chương trình cho PLC

Bước 7 : Chạy chương trình :

Chương trình PLC S7-1200

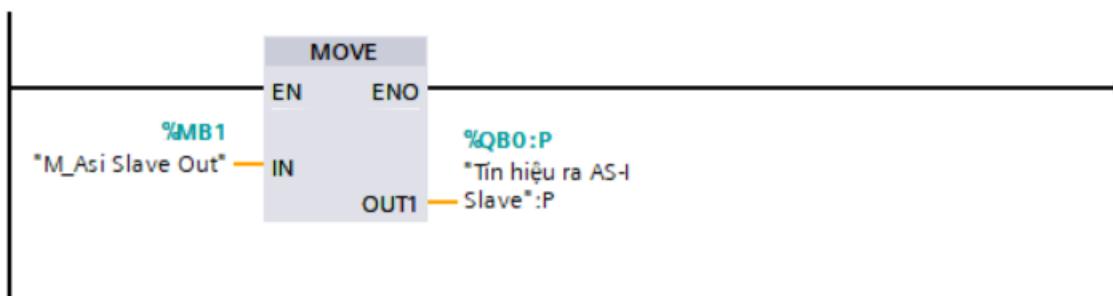
▼ Network 1: Tín hiệu As-i In bộ 1

Comment



▼ Network 2: Tín hiệu As-i Out bộ 1

Comment



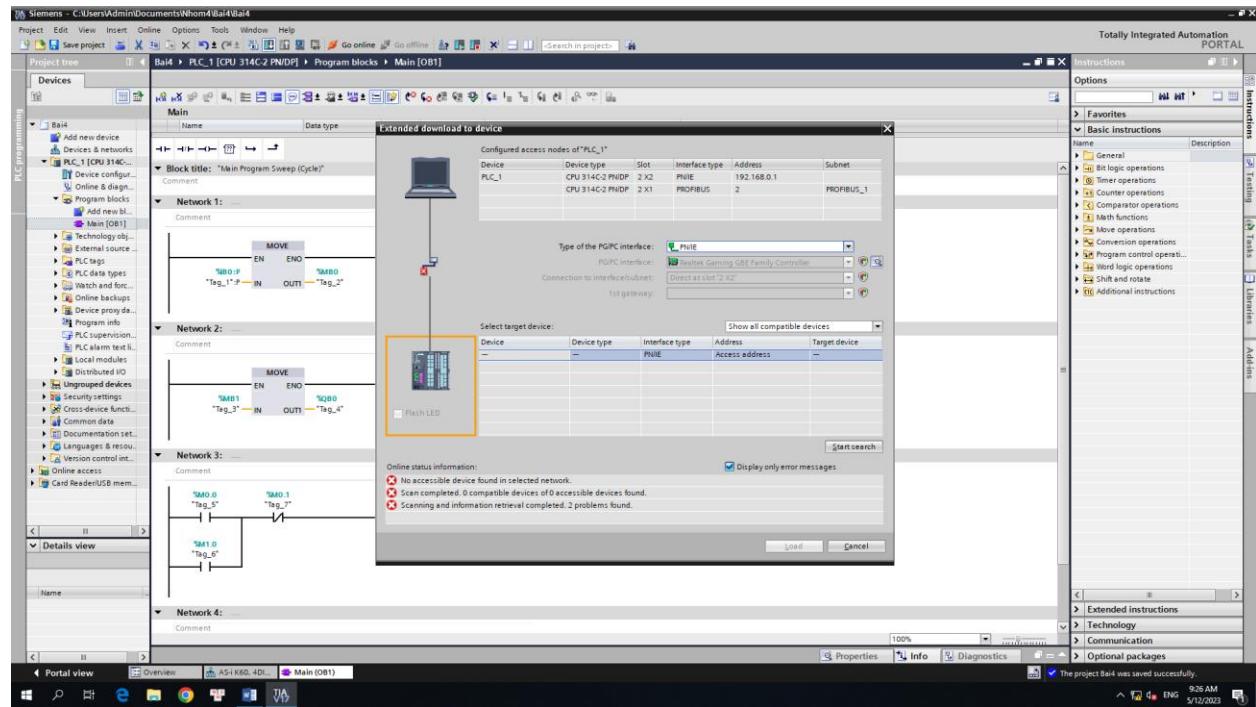
▼ Network 3: Nút nhấn 1 tác động đóng contactor, nút nhấn 2 tác động ngắt contactor

Comment



Kết luận

Không nạp được code PLC



BÀI 5 : THỰC HÀNH LẬP TRÌNH TRUYỀN THÔNG MODBUS RTU PLC S7-1200 VÀ BIẾN TẦN V20

I. Nội dung thực hành :

Trong bài thí nghiệm này sinh viên cần thực hành và nắm được 2 nội dung chính :

Nội dung 1 : Kết nối và cấp nguồn thiết bị

Nội dung 2 : Lập trình điều khiển truyền thông Modbus PLC S7-1200 với biến tần V20 điều khiển động cơ

Nội dung 3 : Lập trình hiển thị và điều khiển động cơ trên màn hình HMI

II. Bài toán :

Lập trình theo bài toán sau

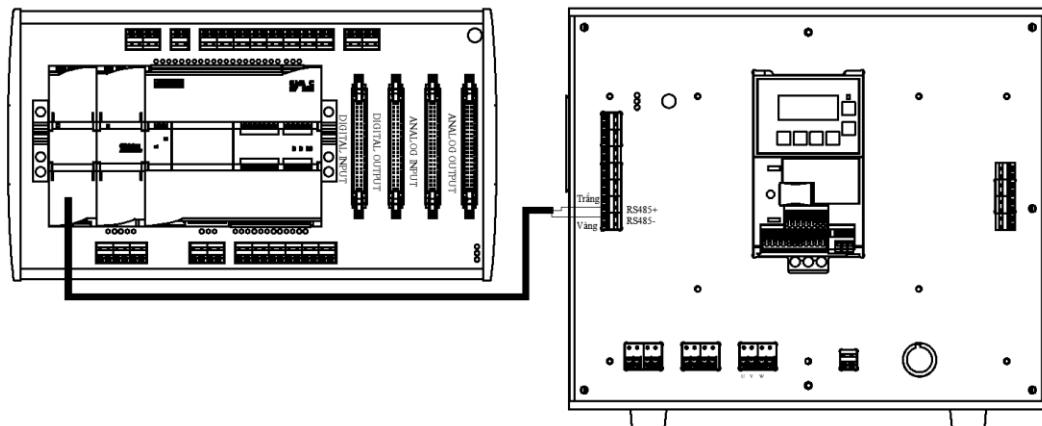
+ Tác động nút nhấn trên HMI để chạy Run, Stop, Reset động cơ

+ Thay đổi tốc độ và chiều quay động cơ trên màn hình HMI

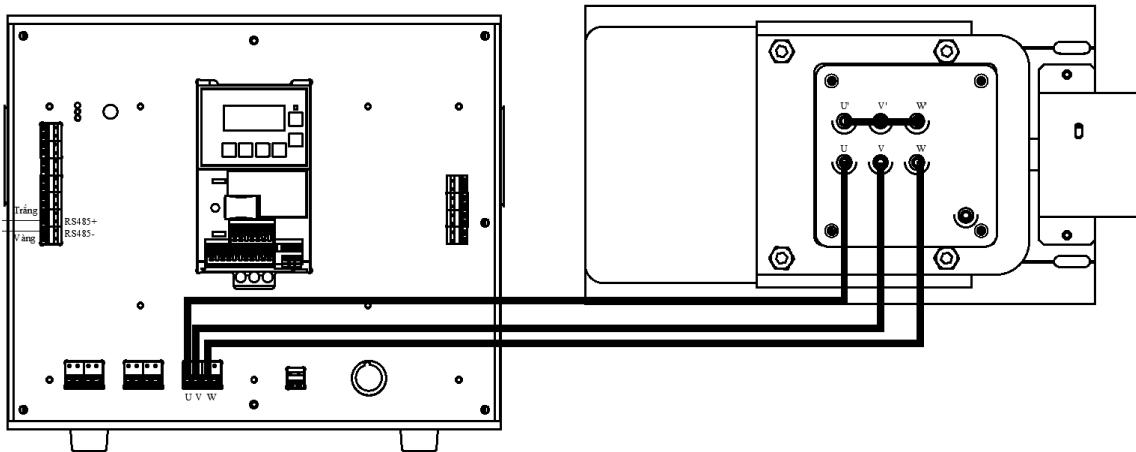
III. Thực hành :

- Các bước thực hành :

Bước 1 : Kết nối cáp từ module mở rộng Modbus từ PLC S7-1200 đến biến tần V20



Bước 2 : Kết nối dây UVW với động cơ và đấu sao cho động cơ



Bước 3 : Cấp nguồn cho thiết bị, cáp Ethernet giữa máy tính với PLC

Bước 4 : Cài đặt thông số biến tần

Tham số trên biến tần	Giá trị	Ghi chú
P0003	1	Reset biến tần về mặc định
P0010	30	
P0970	21	
OK		
P0304	380	Các tham số động cơ
P0301	0.75	
P0308	0.8	
P0310	50	
P0311	1446	
M->Cn011-> OK->M		
P0003	3	Chọn chế độ chuyên gia
P0700	5	Cài đặt chạy bằng rs485
P1000	5	Đặt giá trị tần số xuống biến tần
P2010	6	Cài đặt tần số bằng rs485
P2014	0	Thời gian nhận data
P2021	1	Địa chỉ biến tần
P2022	5000	Thời gian trễ nhận data
P2023	2	Chon chế độ Modbus RTU

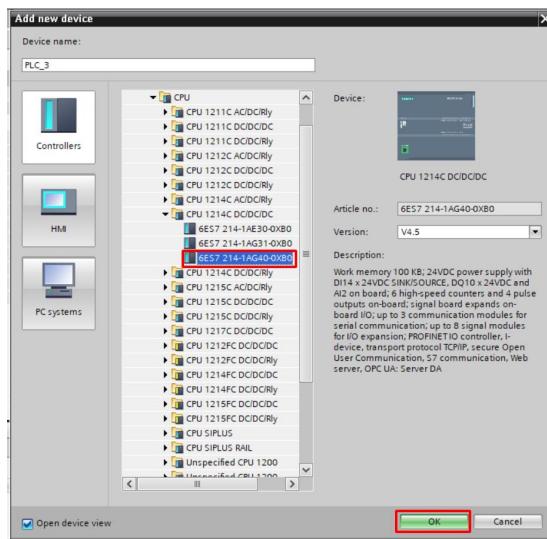
P2034	0	Parity
P2035	1	Stop bit
Tắt nguồn 30s		

Không set được giá trị

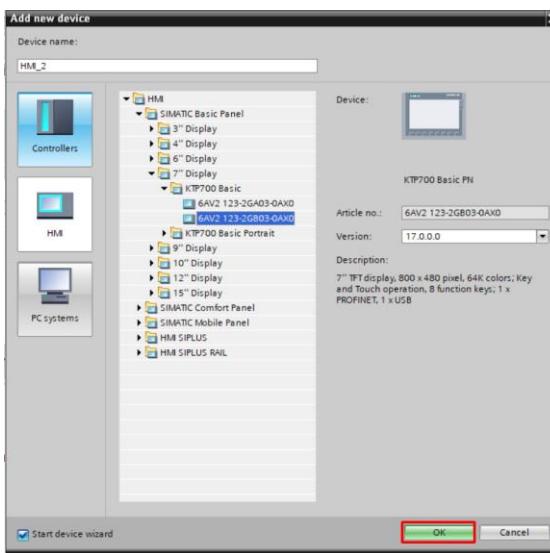
Bước 4 : Khai báo và kết nối phần cứng

*** Thêm phần cứng**

- Thêm PLC S7 1200

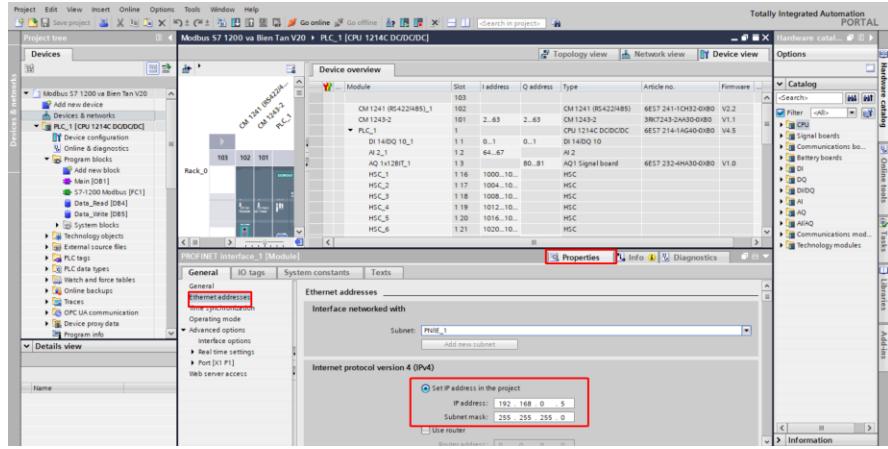


- Thêm HMI

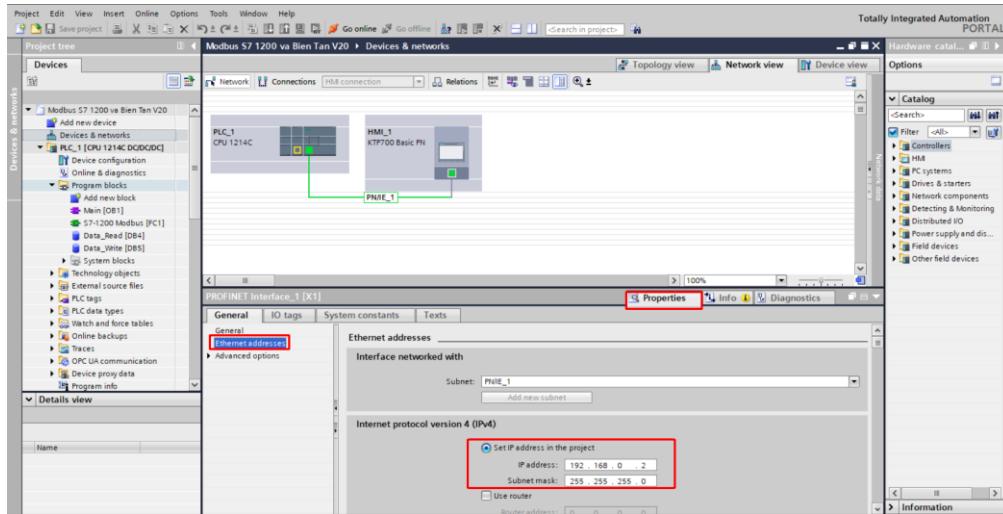


* Cài đặt địa chỉ

Cài địa chỉ PLC S7-1200



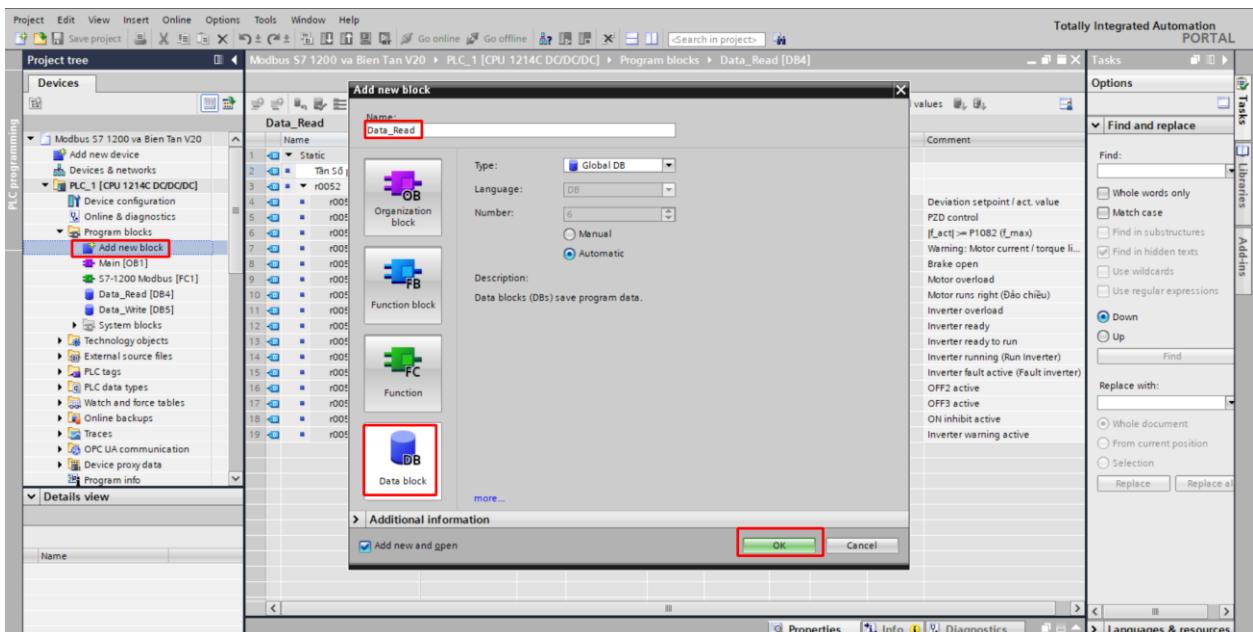
Cài đặt địa chỉ HMI



Bước 5 : Thiết lập I/O

- Khai báo các giá trị đọc từ biến tần lên S7-1200

Tạo khối data block



Khai báo tham số đọc biến tần

- + Tần số phản hồi : Tần số thực tế của biến tần
- + r0052[6] : Phản hồi trạng thái đảo chiều
- + r0052[10] : Phản hồi trạng thái run
- + r0052[11] : Phản hồi trạng thái lỗi

Data_Read										
	Name	Data type	Offset	Start value	Retain	Accessible f...	Write...	Visible in ...	Setpoint	Comment
1	Static									
2	Tần Số phản hồi	Int	0.0	0	<input checked="" type="checkbox"/>					
3	r0052	Array[0..15] of Bool	2.0		<input checked="" type="checkbox"/>					
4	r0052[0]	Bool	2.0	false	<input checked="" type="checkbox"/>	Deviation setpoint / act. value				
5	r0052[1]	Bool	2.1	false	<input checked="" type="checkbox"/>	PZD control				
6	r0052[2]	Bool	2.2	false	<input checked="" type="checkbox"/>	$[f_{act}] \geq P1082 (f_{max})$				
7	r0052[3]	Bool	2.3	false	<input checked="" type="checkbox"/>	Warning: Motor current / torque li...				
8	r0052[4]	Bool	2.4	false	<input checked="" type="checkbox"/>	Brake open				
9	r0052[5]	Bool	2.5	false	<input checked="" type="checkbox"/>	Motor overload				
10	r0052[6]	Bool	2.6	false	<input checked="" type="checkbox"/>	Motor runs right (Đảo chiều)				
11	r0052[7]	Bool	2.7	false	<input checked="" type="checkbox"/>	Inverter overload				
12	r0052[8]	Bool	3.0	false	<input checked="" type="checkbox"/>	Inverter ready				
13	r0052[9]	Bool	3.1	false	<input checked="" type="checkbox"/>	Inverter ready to run				
14	r0052[10]	Bool	3.2	false	<input checked="" type="checkbox"/>	Inverter running (Run Inverter)				
15	r0052[11]	Bool	3.3	false	<input checked="" type="checkbox"/>	Inverter fault active (Fault inverter)				
16	r0052[12]	Bool	3.4	false	<input checked="" type="checkbox"/>	OFF2 active				
17	r0052[13]	Bool	3.5	false	<input checked="" type="checkbox"/>	OFF3 active				
18	r0052[14]	Bool	3.6	false	<input checked="" type="checkbox"/>	ON inhibit active				
19	r0052[15]	Bool	3.7	false	<input checked="" type="checkbox"/>	Inverter warning active				

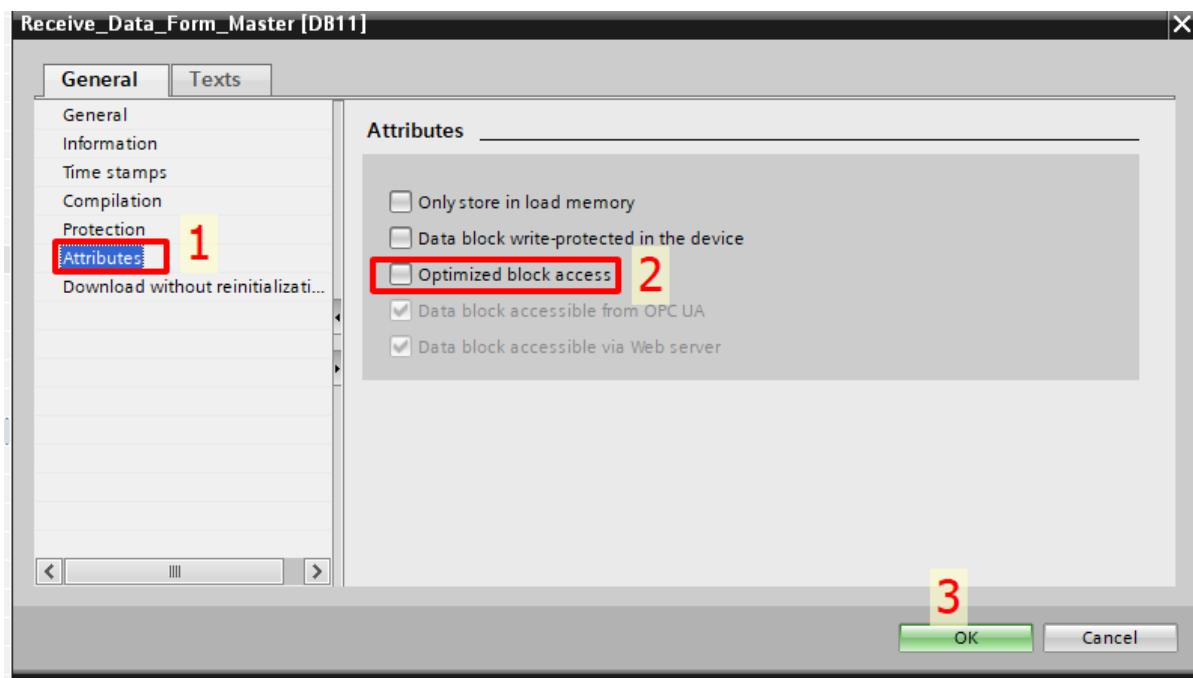
Không offset được giá trị Data Read

- Khai báo tham số truyền từ PLC S7-1200 xuống biến tần

Name	Data type	Offset	Start value	Retain	Accessible f...	Writabl...	Visible in ...	Setpoint	Comment
Control	Struct	0.0	false						
JOG right	Bool	0.0	false						
JOG left	Bool	0.1	false						
Control from PLC	Bool	0.2	false						
Reverse (setpoint i...	Bool	0.3	false						
None	Bool	0.4	false						
Motor potentiome...	Bool	0.5	false						
Motor potentiome...	Bool	0.6	false						
CDS Bit 0 Hand /...	Bool	0.7	false						
ON / OFF1	Bool	1.0	false						
OFF2: Electrical stop	Bool	1.1	false						
OFF3: Faststop	Bool	1.2	false						
Pulse enable	Bool	1.3	false						
RFG enable	Bool	1.4	false						
RFG start	Bool	1.5	false						
Setpoint enable	Bool	1.6	false						
Fault acknowledge	Bool	1.7	false						
Setpoint Inverter	Int	2.0	0						

Không offset được giá trị Data Write

- Click vào block -> Nhấn Alt + Enter bỏ tích Optimized block access (Tất cả các block trên PLC Master và PLC Slave)



- Bảng tín hiệu điều khiển

Bit	Signal name
00	ON / OFF1
01	OFF2: Electrical stop
02	OFF3: Fast stop
03	Pulse enable
04	RFG enable
05	RFG start
06	Setpoint enable
07	Fault acknowledge
08	JOG right
09	JOG left
10	Control from PLC
11	Reverse (setpoint inversion)
12	-
13	Motor potentiometer MOP up
14	Motor potentiometer MOP down
15	CDS Bit 0 (Hand / Auto)

Bước 6 : Lập chương trình trên máy tính, tải chương trình cho PLC

Bước 7 : Chạy chương trình :

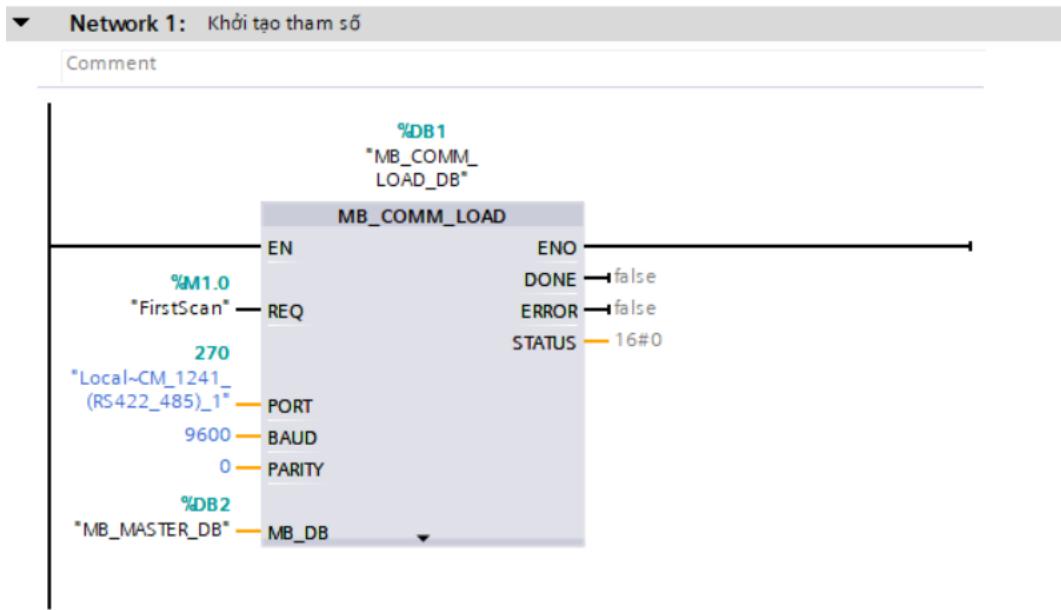
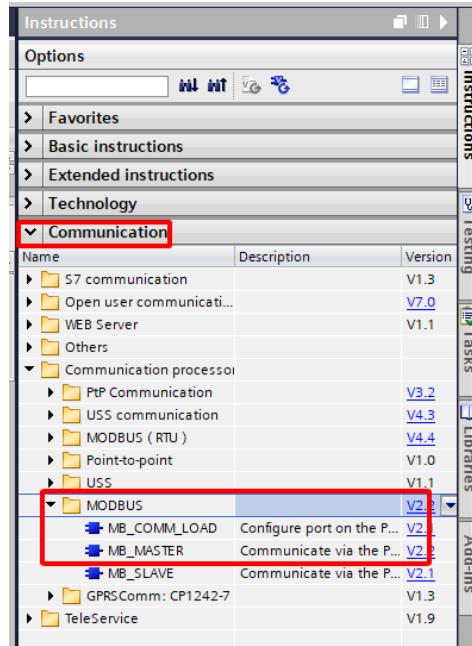
* Lập trình PLC S7-1200

- Địa chỉ biến tần

Register No.		Description	Access	Unit	Scaling factor	Range or On/Off text	Read	Write
Inverter	Modbus						PZD 1	PZD 1
99	40100	STW	R/W	-	1		PZD 1	PZD 1
100	40101	HSW	R/W	-	1		PZD 2	PZD 2
109	40110	ZSW	R	-	1		PZD 1	PZD 1
110	40111	HIW	R	-	1		PZD 2	PZD 2

- HSW (Haupsollwert): Cài đặt tốc độ biến tần
- HIW (Hauptistwert): Tốc độ biến tần hiện tại
- STW (Steuerwort): Tham số điều khiển từ PLC xuống biến tần
- ZSW (Zustandswort): Trạng thái đọc được từ tần
- PZD: Khối dữ liệu
- R/W: Đọc lên/Truyền xuống
- R: Đọc lên
- 40100: Start/Stop/Reset
- 40101: Cài đặt tần số
- 40111: Tần số phản hồi

- Khởi tạo thông số

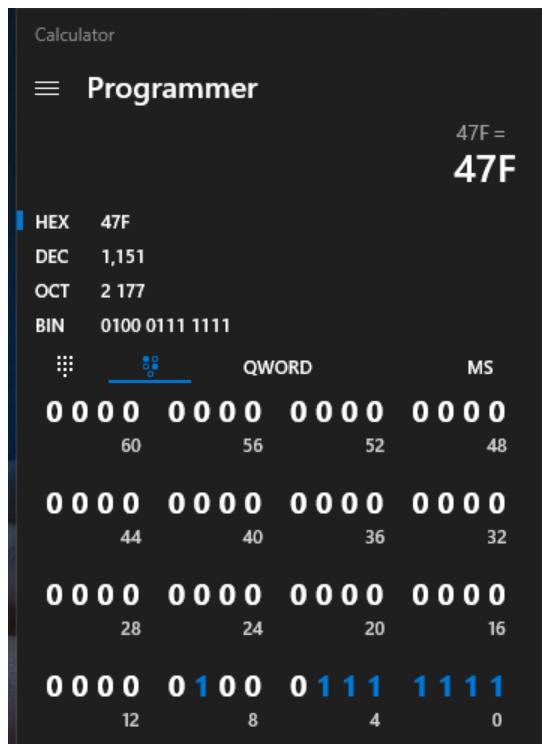


Lệnh "MB_COMM_LOAD" định cấu hình một cổng để liên lạc bằng giao thức Modbus RTU.

- + Req: Tín hiệu cho phép truyền thông
- + Port: Địa chỉ RS485
- + Baud: Tốc độ truyền
- + Parity: Sự kiện
- + MB_DB: Tham chiếu đến khôi dữ liệu

Không tạo code upload giá trị lên HMI để ON, OFF, RESET giá trị động cơ

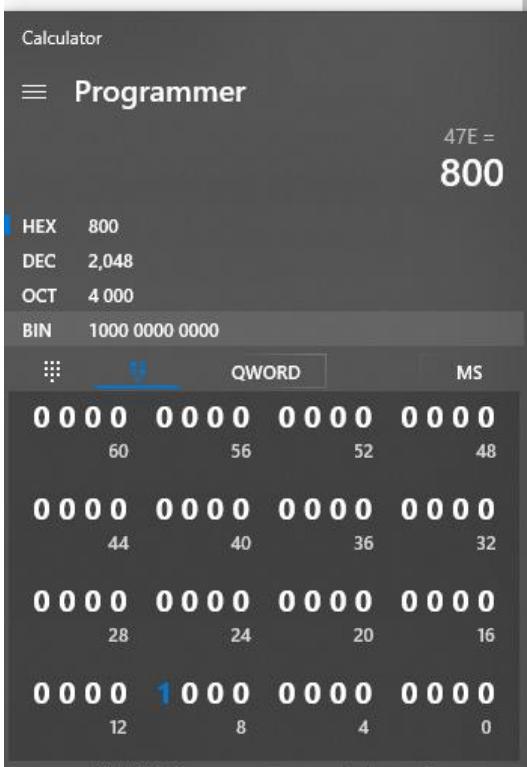
- Cấu hình MB_Master để chạy, dừng và Reset biến tần
 - 47F – Start biến tần



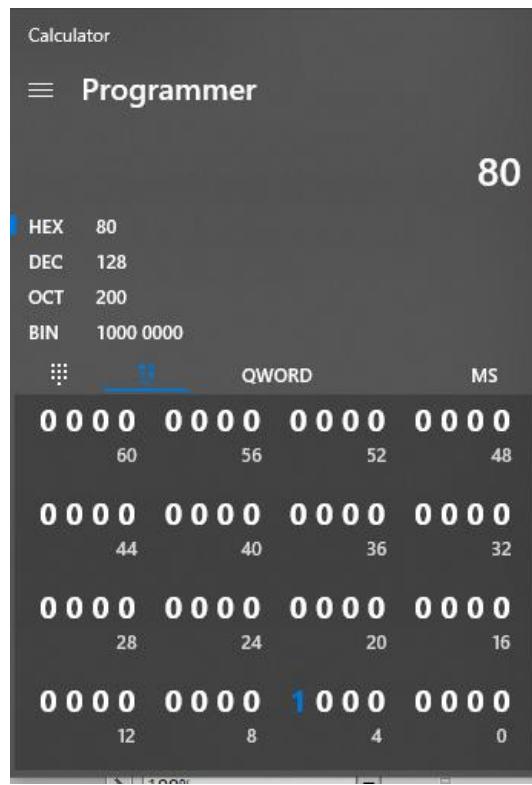
- 47E Stop biển tần



- 0800 – Đảo chiều động cơ



- 0080 Reset biến tần

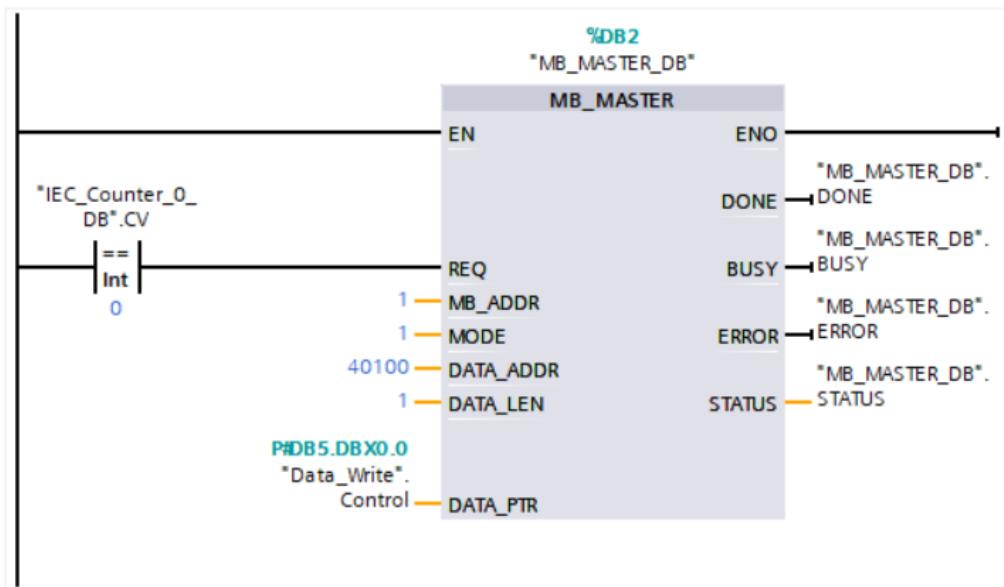


Bit	Signal name	Start	Stop	Reverse	Reset
00	ON / OFF1	1	0	0	0
01	OFF2: Electrical stop	1	1	0	0
02	OFF3: Fast stop	1	1	0	0
03	Pulse enable	1	1	0	0
04	RFG enable	1	1	0	0
05	RFG start	1	1	0	0
06	Setpoint enable	1	1	0	0
07	Fault acknowledge	0	0	0	1
08	JOG right	0	0	0	0

09	JOG left	0	0	0	0
10	Control from PLC	1	1	0	0
11	Reverse (setpoint inversion)	0	0	1	0
12	-	0	0	0	0
13	Motor potentiometer MOP up	0	0	0	0
14	Motor potentiometer MOP down	0	0	0	0
15	CDS Bit 0 (Hand / Auto)	0	0	0	0
Convert Hex		47F	47E	800	80

Network 3: Run/Stop/Reset Biến tần

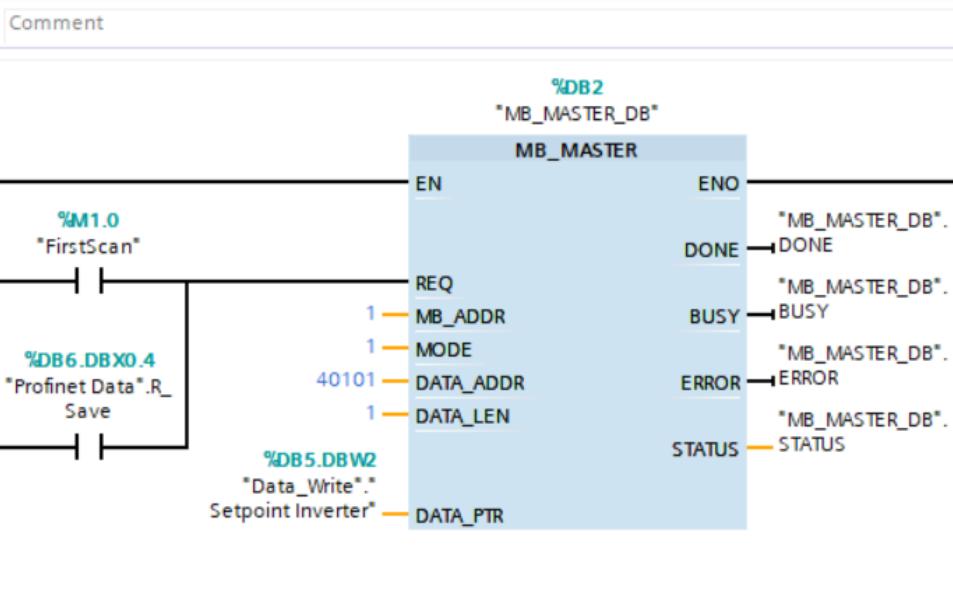
Comment



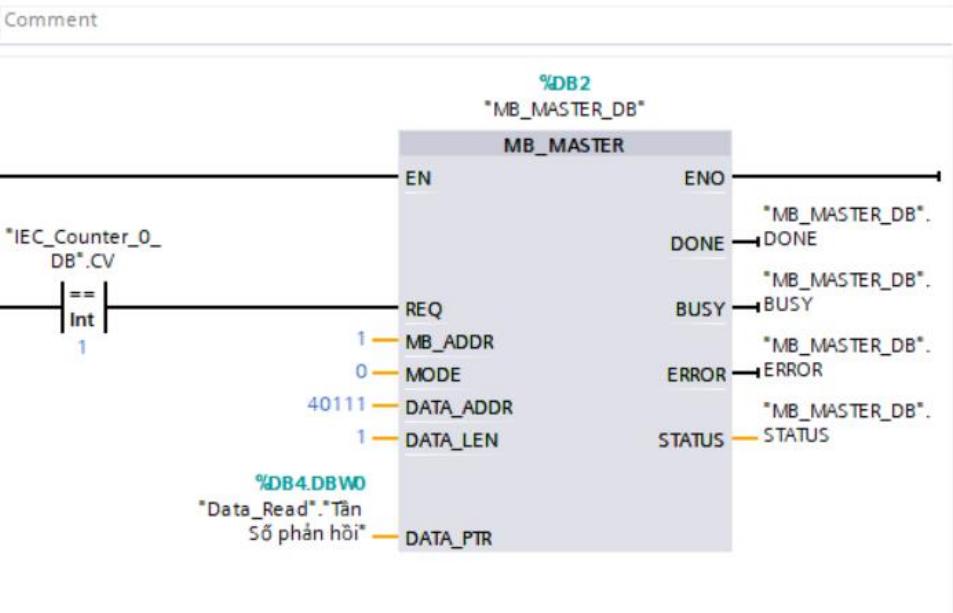
Lệnh "MB_MASTER" cho phép chương trình của bạn giao tiếp với tư cách là chủ Modbus bằng cách sử dụng cổng trên mô-đun điểm-điểm (CM) hoặc bảng giao tiếp (CB)

- + Req: Tín hiệu cho phép truyền thông
- + MB_ADDR: Địa chỉ biến tần
- + Mode: Chế độ đọc/ ghi (1 ghi, 0 đọc)
- + Data_Len: Độ dài
- + Data_PTR: Khối truyền
- Cấu hình MB_Master để đọc ghi giá trị biến tần

Network 5: Gửi giá trị tần số xuống biến tần



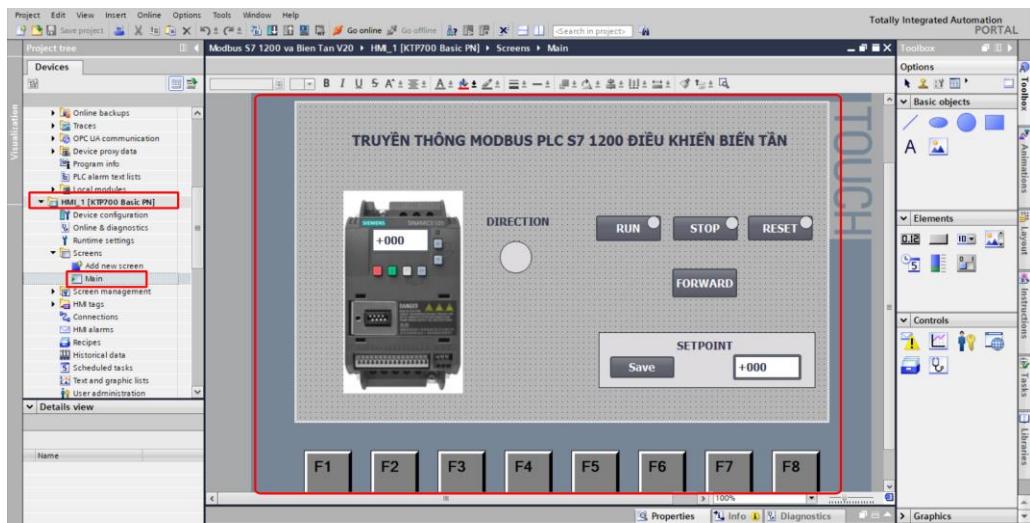
Network 7: Đọc Tân Số đang chạy của biến tần



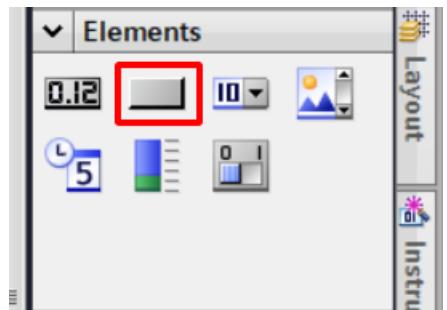
Lệnh "MB_MASTER" cho phép chương trình của bạn giao tiếp với tư cách là chủ Modbus bằng cách sử dụng cổng trên mô-đun điểm-điểm (CM) hoặc bảng giao tiếp (CB)

- + Req: Tín hiệu cho phép truyền thông
- + MB_ADDR: Địa chỉ biến tần
- + Mode: Chế độ đọc/ ghi (1 ghi, 0 đọc)
- + Data_Len: Độ dài
- + Data_PTR: Khối truyền

* Lập trình HMI :

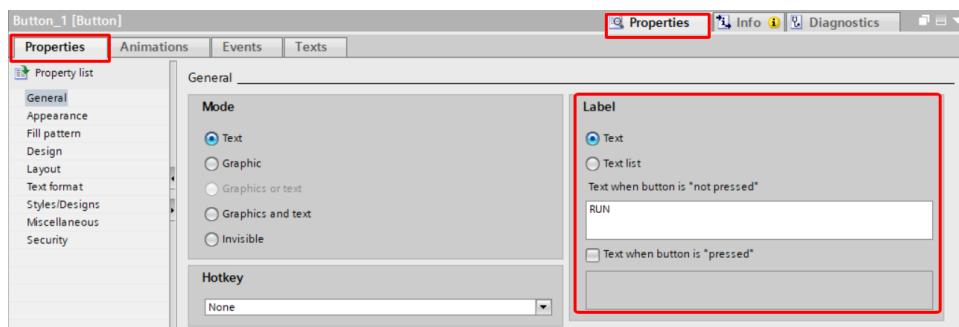


Tạo nút nhấn : Vào mục Elements kéo nút nhấn ra giao diện sử dụng



Cài đặt nút nhấn :

+ Đặt tên nút nhấn



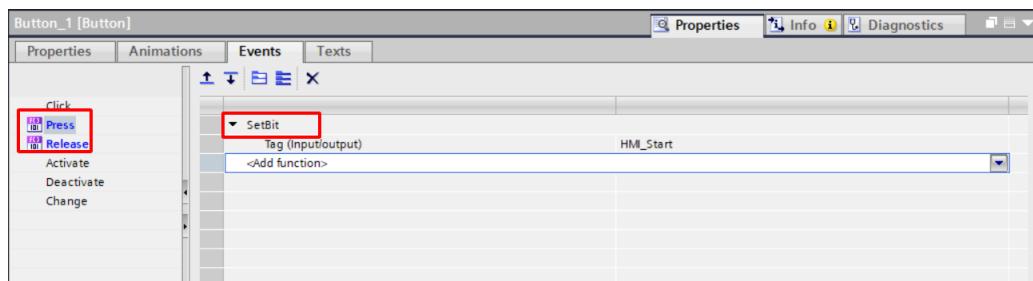
+ Tạo trạng thái nút nhấn :

Press : trạng thái nhấn

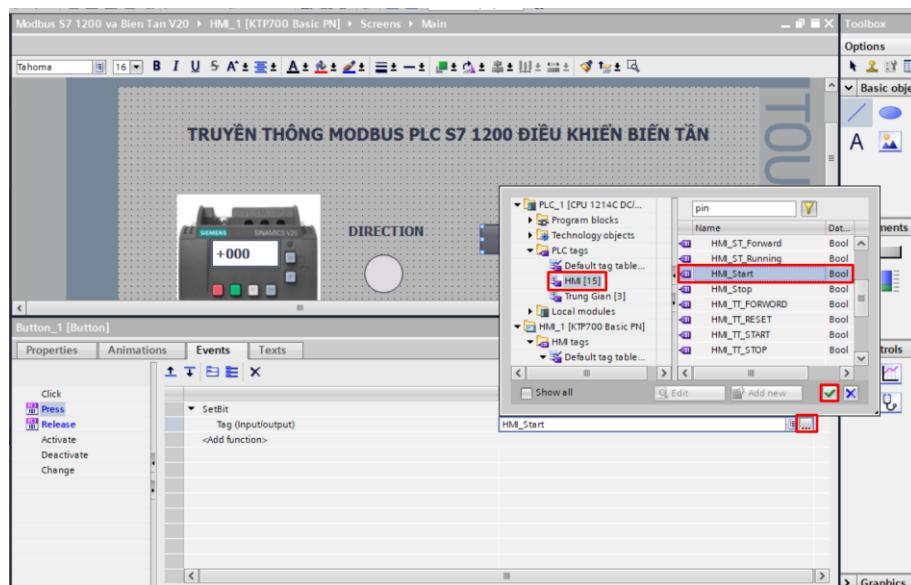
Release : trạng thái nhả

Setbit : Chức năng set lên 1 của tag

Reset bit : Set về 0 của tag

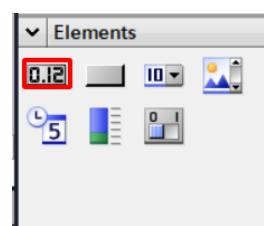


Chọn tag cần tác động



+ Tạo khối chức năng nhập hoặc hiển thị tần số :

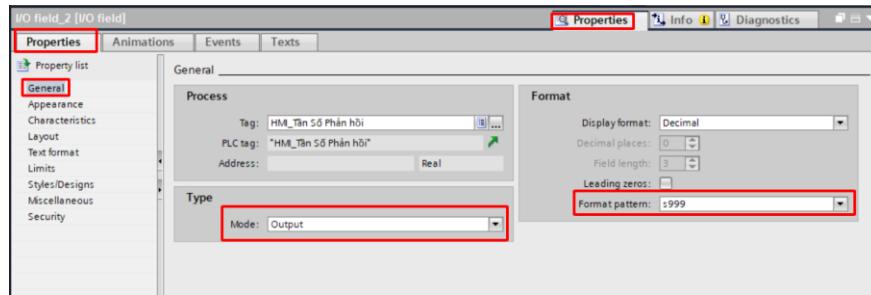
Vào Elements chọn khối chức năng



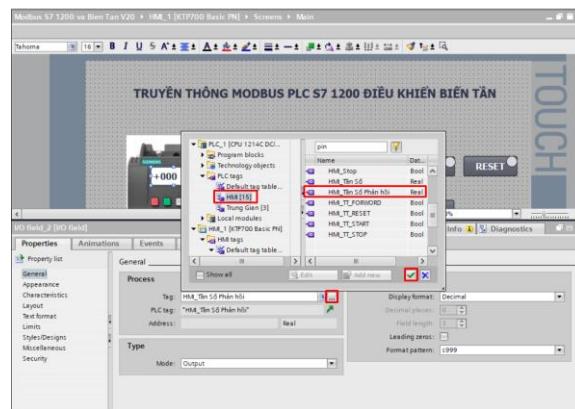
Cài đặt chức năng

Mode : Output để hiển thị tần số / input để nhập tần số

Format pattern : định dạng hiển thị bao nhiêu chữ số

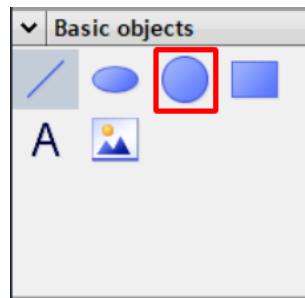


Chọn tag cần tác động



+ Tạo đèn hiển thị trạng thái

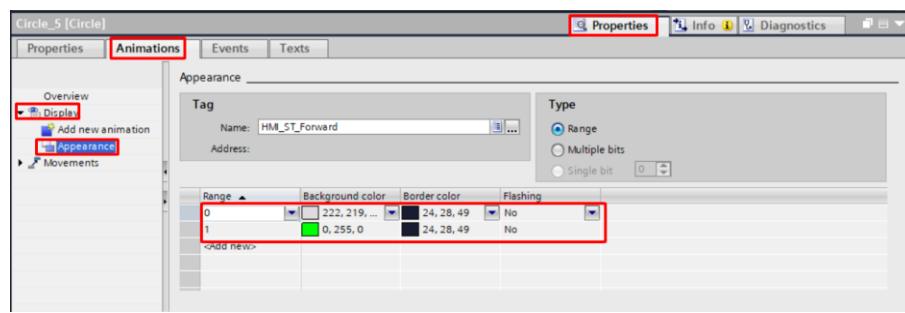
Vào Basic objects chọn chức năng đèn



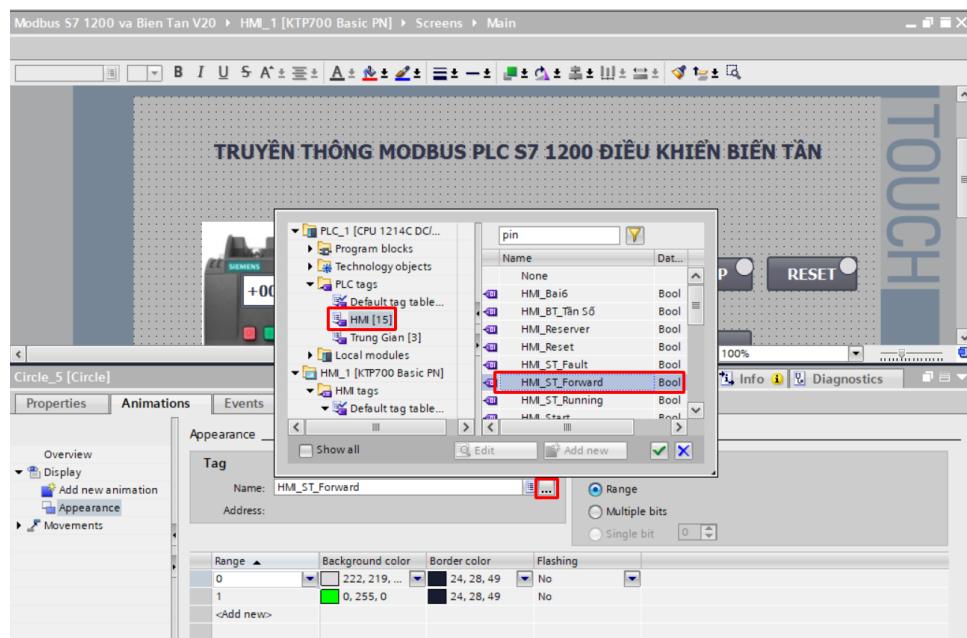
Cài đặt chức năng

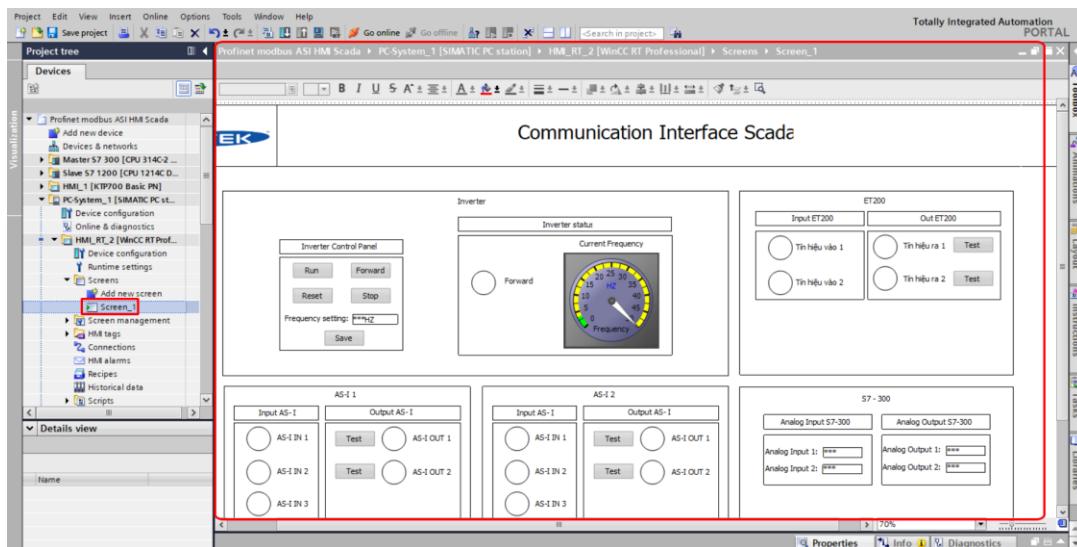
0 : màu hiển thị khi bit trạng thái 0

1 : màu hiển thị khi bit trạng thái 1



Chọn tag cần tác động





Kết luận

Chạy được các giá trị trên

Tham số trên biến tần	Giá trị	Ghi chú
P0003	1	Reset biến tần về mặc định
P0010	30	
P0970	21	
OK		
P0304	380	Các tham số động cơ
P0308	0.8	
P0310	50	
P0311	1446	
OK		

Buổi chiều

Chạy thử xe, xe không chạy do không điều khiển được tốc độ motor