

HƯỚNG DẪN SỬ DỤNG PLCPi



ATSCADA Lab

ATPro Corp

Đ/c: 2A – Đường số 1 – P.Tân Thành – Q.Tân Phú - Tp.HCM

Xưởng SX: Kdc.số 3, p.Trương Quang Trọng, Tp.Quảng Ngãi (gần VSIP Q.Ngãi)

Email: soft@atpro.com.vn

Website: <http://atscada.com>

MỤC LỤC

1. PLCPi LÀ GÌ:	5
1.1. Định nghĩa:	5
1.2. Những điểm nổi bậc đặc sắc của PLCPi:	5
2. CẤU TRÚC PHẦN CỨNG:	5
2.1. CPU Rasp01-CH:	6
2.2. Module 8DI/8DO RLY:	8
2.3. Module 8DI/16DO BJT:	8
2.4. Module 16DI/16DO BJT:	9
2.5. Module AI:	9
2.6. Module Display:	10
3. LÀM QUEN VỚI PLCPi:	10
3.1. Cách Boot vào hệ điều hành:	10
3.2. Các công cụ phụ trợ làm việc với CPU Rasp01-CH:	11
3.2.1. Phần mềm WinSCP:	11
3.2.2. Phần mềm Putty:	13
3.2.3. Ứng dụng Remote Desktop Connection:	14
3.2.4. Visual studio community – IDE lập trình PLCPi từ xa, và cách chạy ứng dụng trên PLCPi...	15
3.3. Cách thay đổi địa chỉ IP cho PLCPi:	20
3.4. Kết nối USB 3G:	24
3.5. Cài đặt để chương trình tự khởi động khi boot	26
3.5.1. Chương trình Console:	26
3.5.2. Chương trình HMI Winform:	28
3.5.3. Chương trình HMI Webform:	30
3.5.4. Kết nối 3G để dung internet:	33
4. LẬP TRÌNH PHẦN MỀM CHO PLCPi:	34
4.1. Cài đặt môi trường lập trình cho PLCPi: VS2015 Community + PLCPi.exe	34
4.2. Giới thiệu thư viện PLCPi.dll:	36
4.3. Hướng dẫn sử dụng các tính năng trong thư viện PLCPi.dll	36
4.3.1. NgoVao:	36

4.3.2.	NgoRa:	37
4.3.3.	AI:	38
4.3.4.	DS18B20:.....	40
4.3.5.	DHT21:	41
4.3.6.	ThoiGian:	41
4.3.7.	HienThiLed:.....	42
4.3.8.	mySQLLogger:	42
4.3.9.	SQLLogger:	44
4.3.10.	SMS, Gọi điện thoại:.....	47
4.3.11.	Email:	49
4.3.12.	Đọc ghi dữ liệu từ mảng byte:.....	51
4.3.13.	S7Ethernet:	58
4.3.14.	ModbusTCPClient:	64
4.3.15.	ModbusRTUMaster:.....	70
5.	CÁC VÍ DỤ:	77
5.1.	Hướng dẫn tạo dự án Console_PLCPi:	77
5.1.1.	Ví dụ 1: DI_DO	77
5.1.2.	Ví dụ 2: DS18B20:.....	80
5.1.3.	Ví dụ 3: DHT21:	82
5.1.4.	Ví dụ 4: ThoiGian:	83
5.1.5.	Ví dụ 5: truyền thông S7Ethernet_Server:.....	84
5.1.6.	Ví dụ 6: truyền thông S7Ethernet_Client:	85
5.1.7.	Ví dụ 7: GiamSat_DieuKien_NhietDo	87
5.2.	Hướng dẫn tạo dự án Winform_HMI_PLCPi:.....	90
5.2.1.	Ví dụ 1: Winform_HMI_PLCPi_App3	90
5.2.2.	Ví dụ 2: DI_DO_Winform.....	96
5.2.3.	Ví dụ 3: ModbusRTUMaster	98
5.2.4.	Ví dụ 4: ModbusTCPClient	101
5.2.5.	Ví dụ 5: Gửi SMS	104
5.2.6.	Ví dụ 6: Dự án DKGSNhaXuong:	106
5.2.7.	Ví dụ 7: TrangTrai	108
5.3.	Hướng dẫn tạo dự án Webform_HMI_PLCPi:	111
5.3.1.	Ví dụ1: Webform_HMI_PLCPi_App1	111

1. PLCPI LÀ GÌ:

1.1. Định nghĩa:

Là Programmable Automation Controller được sử dụng làm PLC hoặc HMI trong các ứng dụng dân dụng và công nghiệp.

PLCPI được trang bị bộ xử lý trung tâm là Raspberry Pi 2- mini computer nhúng xuất xứ từ Anh Quốc

1.2. Những điểm nổi bật đặc sắc của PLCPI:

Ngôn ngữ lập trình cho PLCPI là C Sharp (C#). Thư viện để lập trình các ứng dụng cho PLCPI được ATProCorp phát triển từ ngôn ngữ C#. Chúng tôi đã xây dựng bộ thư viện cho PLCPI với phương châm đơn giản và tiện dụng nhất cho người dùng. Vì vậy, PLCPI rất dễ sử dụng.

PLCPI có thể làm PLC hoặc HMI để điều khiển giám sát và thu thập dữ liệu.

PLCPI có web server phục vụ điều khiển giám sát tại chỗ và điều khiển giám sát từ xa.

PLCPI có tính năng datalogger để log dữ liệu vào cơ sở dữ liệu mySQL (mặc định), SQL, Oracle, ...

PLCPI có truyền thông s7 ethernet client và server giúp PLCPI kết nối được với các hệ thống SCADA giống như các PLC S7 của Siemens, và kết nối được với các PLC S7 của Siemens như các PG/PC HMI của Siemens thông qua ethernet.

PLCPI sẽ là các IoT Devices trong mạng phân tán khi triển khai hệ thống Cloud – based ATSCADA.

2. CẤU TRÚC PHẦN CỨNG:

- PLCPI gồm 6 modules: CPU Rasp01-CH, 3 modules mở rộng DI/DO, AI, Display.
- Địa chỉ ngõ vào: có 5 byte ngõ vào DI
 - + CPU Rasp01-CH: I0
 - + Module mở rộng 8DI/8DO RLY: I1
 - + Module mở rộng 8DI/16DO BJT: I2
 - + Module mở rộng 16DI/16DO BJT: I3, I4
- Địa chỉ ngõ ra: có 6 byte ngõ ra DO
 - + CPU Rasp01-CH: Q0

- + Các modules mở rộng có địa chỉ ngõ ra tương đối. Nghĩa là module mở rộng nào được kết nối gần với CPU Rasp01-CH nhất thì ngõ ra của module đó sẽ có địa chỉ là Q1, tương tự như vậy cho đến Q5 nếu ta kết nối hết 3 module mở rộng DI/DO vào CPU Rasp01-CH

2.1. CPU Rasp01-CH:





CPU Rasp01- CH

SIZE: 11.3cm x 12.6cm x5.3cm

CÁU HÌNH

- +CHIP “Broadcom BCM2836 ARMv7 Quad Core”
- +1GB RAM
- +4 cổng USB
- +Cổng Ethernet
- +Cổng HDMI
- +Cổng Audio
- +Cổng camera

Tính năng:

- +8 ngõ vào DI cách ly quang
- +8 ngõ ra DO (DC) cách ly quang
- +Kết nối cảm biến DS18B20
- +Kết nối cảm biến DHT21
- +Thời gian thực RTC DS1307
- +Truyền thông S7 Ethernet Client và Server.

- + Kích thước: 11.3cm x 12.6cm x5.3cm (Dài x Rộng x Cao)
- + Cấu hình của Chip:
 - Sử dụng chip xử lý “Broadcom BCM2836 ARMv7 Quad Core”, 900MHz
 - 1GB RAM
 - 4 cổng USB
 - Cổng Ethernet
 - Cổng HDMI
 - Cổng Audio
- + Tính năng:
 - 8 ngõ vào DI, được cách ly quang
 - 8 ngõ ra DO BJT, được cách ly quang
 - Hỗ trợ kết nối cảm biến DS18B20 để đo nhiệt độ, hoặc kết nối cảm biến DHT21 để đo nhiệt độ độ ẩm
 - Thời gian thực(RTC DS1307)
 - Hỗ trợ truyền thông S7 Ethernet Client và Server.

- + PLCPi-S, phiên bản thu gọn của PLCPi:



- Cấu hình của Chip:
 - * Sử dụng chíp xử lý “Broadcom BCM2836 ARMv7 Quad Core”, 900MHz
 - * 1GB RAM
 - * 4 cổng USB
 - * Cổng Ethernet
 - * Cổng Audio
- Tính năng:
 - * 8 ngõ vào DI, được cách ly quang
 - * 8 ngõ ra DO BJT, trong đó có 6 ngõ BJT, 2 ngõ relay
 - * Hỗ trợ kết nối cảm biến DS18B20 để đo nhiệt độ, hoặc kết nối cảm biến DHT21 để đo nhiệt độ độ ẩm
 - * Thời gian thực(RTC DS1307)
 - * Hỗ trợ truyền thông Profinet Client và Server.
 - * Hỗ trợ truyền thông modbus TCP/IP Client
 - * Hỗ trợ truyền thông modbus RTU Master
 - * Hỗ trợ 2 ngõ vào AI 0-10vdc, và 2 ngõ vào AI 4-20mA

2.2. Module 8DI/8DO RLY:



Module 8DI/8DO RLY



-Size: 13.6cm x 12.6cm x5.3cm
-Tính năng vào ra:
+8 ngõ vào DI, cách ly quang
+8 ngõ ra DO Relay, cách ly relay

- + Kích thước: 13.6cm x 12.6cm x5.3cm (Dài x Rộng x Cao)
- + Tính năng vào ra:
 - 8 ngõ vào DI, cách ly quang
 - 8 ngõ ra DO Relay, cách ly relay

2.3. Module 8DI/16DO BJT:



Module 8DI/16DO (DC)



-Size: 13.6cm x 12.6cm x 5.3cm
-Tính năng vào ra:
+8 ngõ vào DI, cách ly quang
+16 ngõ ra DO (DC), cách ly quang

- + Kích thước: 13.6cm x 12.6cm x5.3cm (Dài x Rộng x Cao)
- + Tính năng vào ra:
 - 8 ngõ vào DI, cách ly quang

- 16 ngõ ra DO BJT, cách ly quang

2.4. Module 16DI/16DO BJT:



-SIZE: 13.6cm x 12.6cm x5.3cm

-Tính năng vào ra:

16 ngõ vào DI, cách ly quang
16 ngõ ra DO (DC), cách ly quang

Module 16DI/16DO (DC)

- + Kích thước: 13.6cm x 12.6cm x5.3cm (Dài x Rộng x Cao)
- + Tính năng vào ra:
 - 16 ngõ vào DI, cách ly quang
 - 16 ngõ ra DO BJT, cách ly quang

2.5. Module AI:



-Size: 8.8cm x 12.6cm x5.3cm

-Tính năng: dùng để đọc các tín hiệu Analog vào, có 13 kênh AI trong đó có 5 kênh đọc tín hiệu Analog từ 0-10VDC, 8 kênh đọc tín hiệu Analog từ 4-20mA.

+Kênh AI0-AI4: tín hiệu 0-10VDC

+Kênh AI5-AI12: tín hiệu 4-20mA

Module AI

- + Kích thước: 8.8cm x 12.6cm x5.3cm (Dài x Rộng x Cao)

- + Tính năng: dùng để đọc các tín hiệu Analog vào, có 13 kênh AI trong đó có 5 kênh đọc tín hiệu Analog từ 0-10VDC, 8 kênh đọc tín hiệu Analog từ 4-20mA. Bộ chuyển ADC 10 bit.
- + Kênh AI0-AI4: tín hiệu 0-10VDC
- + Kênh AI5-AI12: tín hiệu 4-20mA

2.6. Module Display:





-Kích thước: 11.2cm x 12.6cm x5.3cm
-Tính năng:

- +Có 2 kênh led 7 đoạn hiển thị, mỗi kênh 4 số
- +Hiển thị các giá trị cần hiển thị ra led7 đoạn(Ví dụ: nhiệt độ, độ ẩm, ...)
- +Hỗ trợ kết nối cảm biến DS18B20 để đo nhiệt độ, hoặc kết nối cảm biến DHT21 để đo nhiệt độ độ ẩm

Module Display

- + Kích thước: 11.2cm x 12.6cm x5.3cm (Dài x Rộng x Cao)
- + Tính năng:
 - Có 2 kênh led 7 đoạn hiển thị, mỗi kênh 4 số
 - Hiển thị các giá trị cần hiển thị ra led7 đoạn(Ví dụ: nhiệt độ, độ ẩm, ...)
 - Hỗ trợ kết nối cảm biến DS18B20 để đo nhiệt độ, hoặc kết nối cảm biến DHT21 để đo nhiệt độ độ ẩm

3. LÀM QUEN VỚI PLCPI:

CPU Rasp01-CH sử dụng bộ xử lý là Raspi 2 sử dụng hệ điều hành Raspbian Linux, nên giao diện của nó rất dễ sử dụng

3.1. Cách Boot vào hệ điều hành:

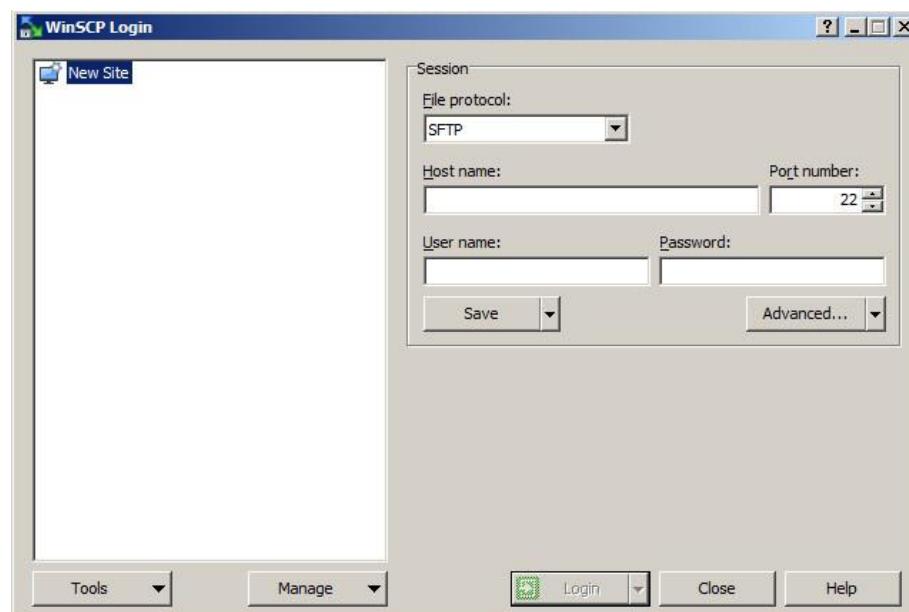
- Để làm việc với CPU Rasp01-CH ta có thể thao tác trực tiếp trên chính nó, hoặc gián tiếp thông qua máy tính.
 - + Trực tiếp:

- Kết nối chuột, bàn phím vào các cổng USB và màn hình vào cổng HDMI rồi cấp nguồn cho CPU Rasp01-CH.
- + Gián tiếp:
 - Kết nối CPU Rasp01-CH với máy tính thông qua mạng LAN hoặc WLAN. Rồi dùng ứng dụng Remote Desktop Connection để kết nối

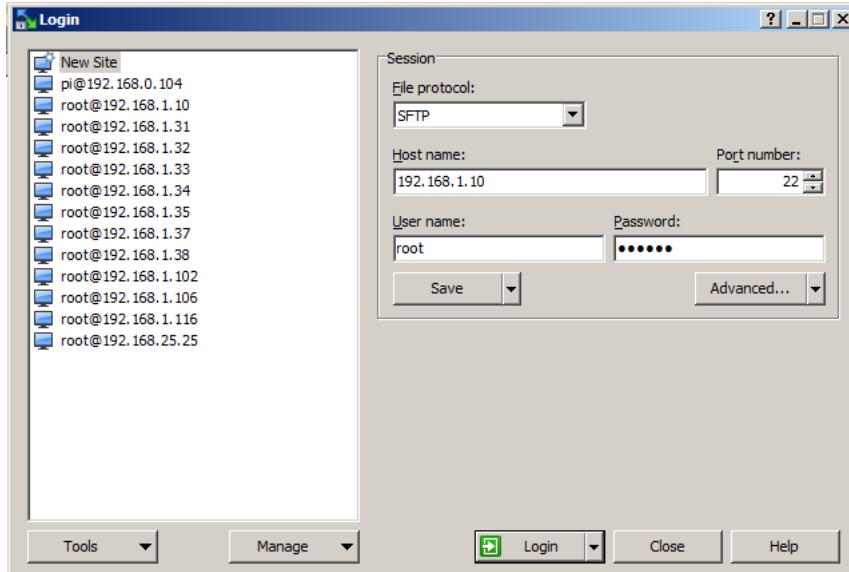
3.2. Các công cụ phụ trợ làm việc với CPU Rasp01-CH:

3.2.1. Phần mềm WinSCP:

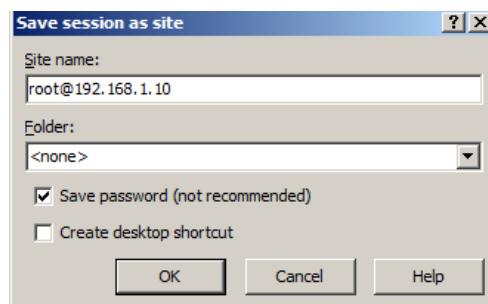
- Tải về tại đây: <http://atscada.com/plcpi/>
- Dùng để sao chép dữ liệu qua lại giữa máy tính và CPU Rasp01-CH
- Sau khi cài đặt phần xong, ta mở phần mềm lên sẽ có giao diện như hình:



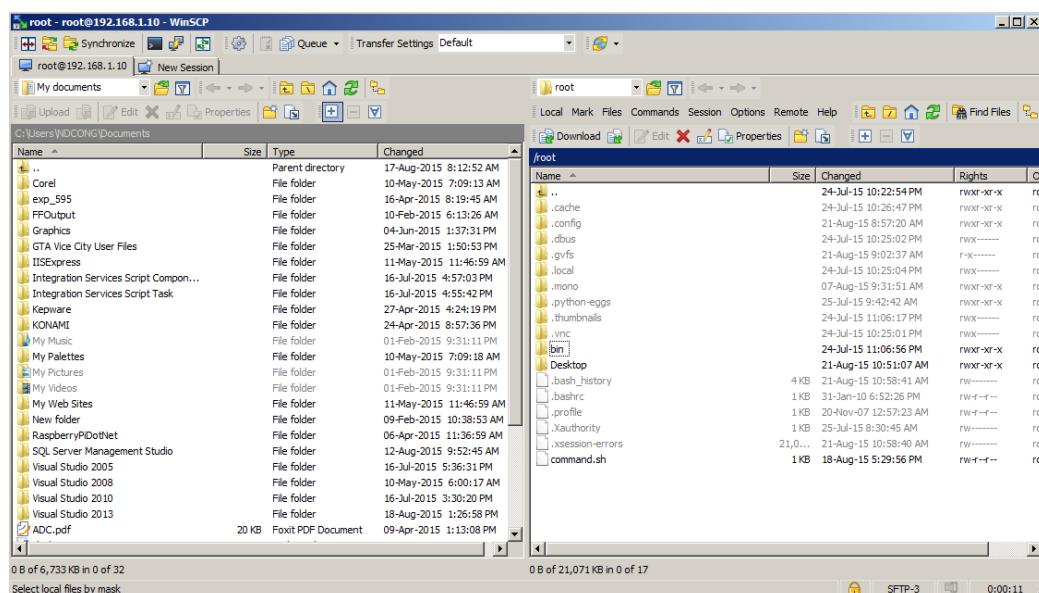
- + Host name: nhập vào địa chỉ IP tĩnh của PLCPi (ví dụ: 192.168.1.33)
- + Port number: giữ nguyên mặc định là 22
- + User name: nhập vào “root”
- + Password: nhập vào “100100”
- Sau khi nhập xong sẽ có giao diện như sau:



- + Chúng ta có thể lưu những thông tin này để những lần sau kết nối không phải nhập lại
 - Chọn nút Save, xuất hiện giao diện:



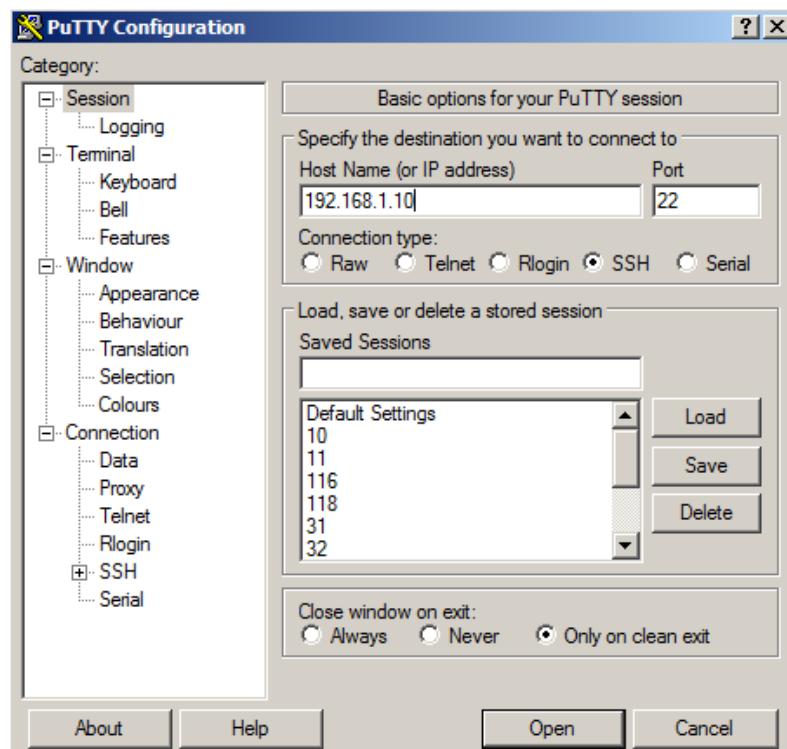
- * Tích vào ô Save password (not recommended), rồi chọn OK
- Chọn nút Login để kết nối với PLCPi. Sau khi kết nối thành công ta sẽ có giao diện như sau:



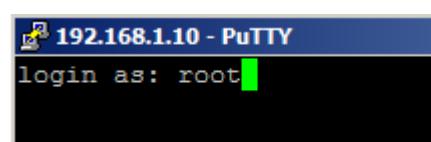
- + Ô bên tay trái là phần quản lý file của máy tính chúng ta, ô bên tay phải là phần quản lý file của PLCPi
- + Để sao chép dữ liệu từ máy tính sang PLCPi thì ta chỉ việc kích giữ file bên máy tính rồi kéo thả qua bên PLCPi. Ngược lại, muốn sao chép dữ liệu từ PLCPi sang máy tính thì ta kích giữ file bên PLCPi rồi kéo thả qua bên máy tính

3.2.2. Phần mềm Putty:

- Tải về tại đây: <http://atscada.com/plcpi/>
- Dùng để chạy các lệnh Linux Terminal trên Windows
- Giao diện của phần mềm:



- + Host Name: nhập địa chỉ IP của PLCPi vào
- + Connection type: chọn SSH
- + Close window on exit: chọn Only on clean exit
- Chọn Open để kết nối. Sẽ hiện ra giao diện như sau:



- Nhập vào “root” rồi enter, sẽ yêu cầu nhập password, ta nhập vào “100100” rồi enter. Ta được giao diện kết nối như sau:

```

login as: root
root@192.168.1.10's password:
Linux raspberrypi 3.18.11-v7+ #781 SMP PREEMPT Tue Apr 21 18:07:59 BST 2015 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Aug 19 08:26:38 2015 from 192.168.1.112
root@raspberrypi:~#

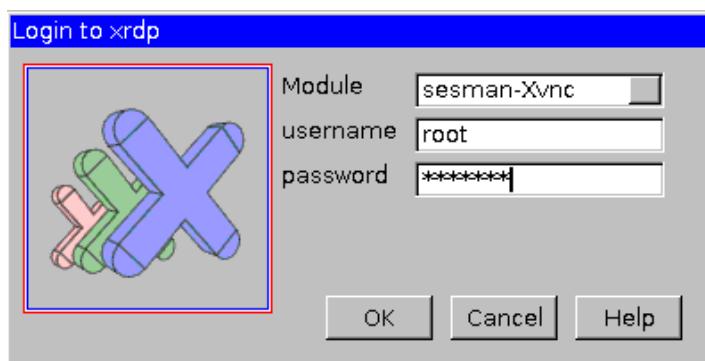
```

3.2.3. Ứng dụng Remote Desktop Connection:

- Dùng để điều khiển Desktop CPU Rasp01-CH trực tiếp từ máy tính

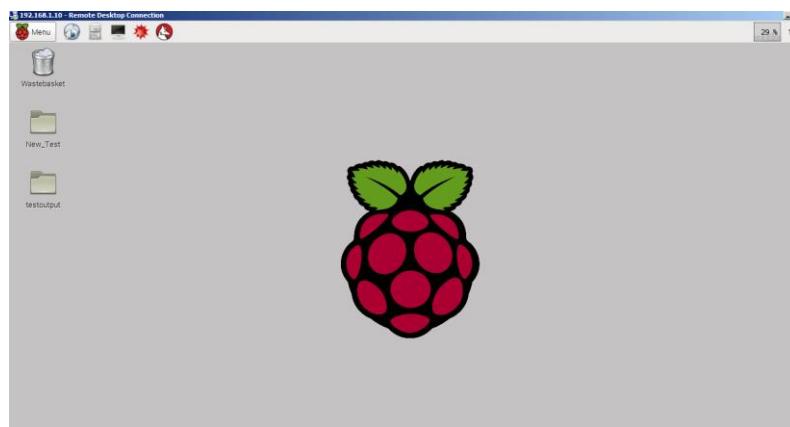


- Nhập địa chỉ IP của CPU Rasp01-CH vào rồi bấm chọn Connect sẽ xuất hiện màn hình



- Username: ta nhập root
- password: ta nhập 100100,

- Chọn OK, thì sẽ vào màn hình Desktop của CPU Rasp01-CH.
- Kết nối thành công ta sẽ có giao diện Desktop của CPU Rasp01-CH trên máy tính như sau:

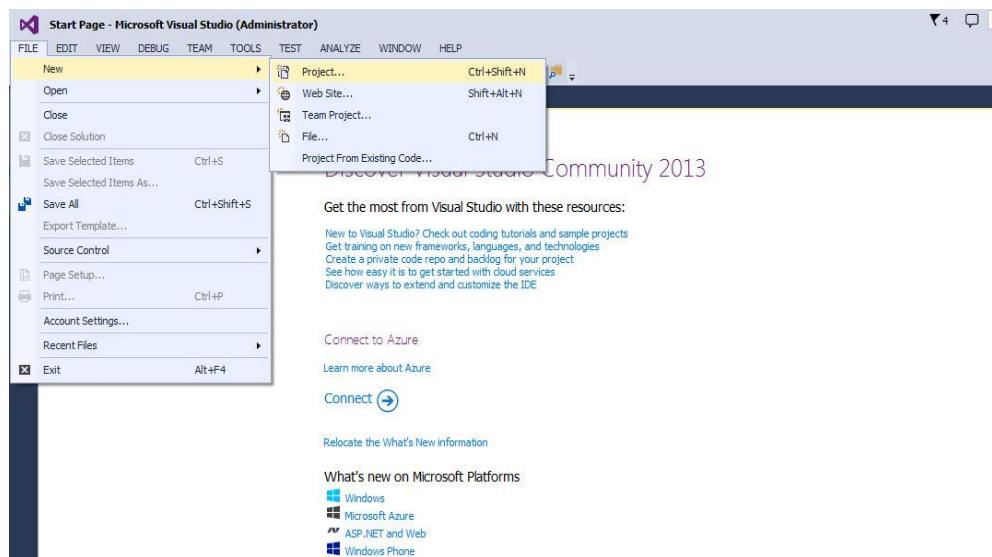


3.2.4. Visual studio community – IDE lập trình PLCPi từ xa, và cách chạy ứng dụng trên PLCPi

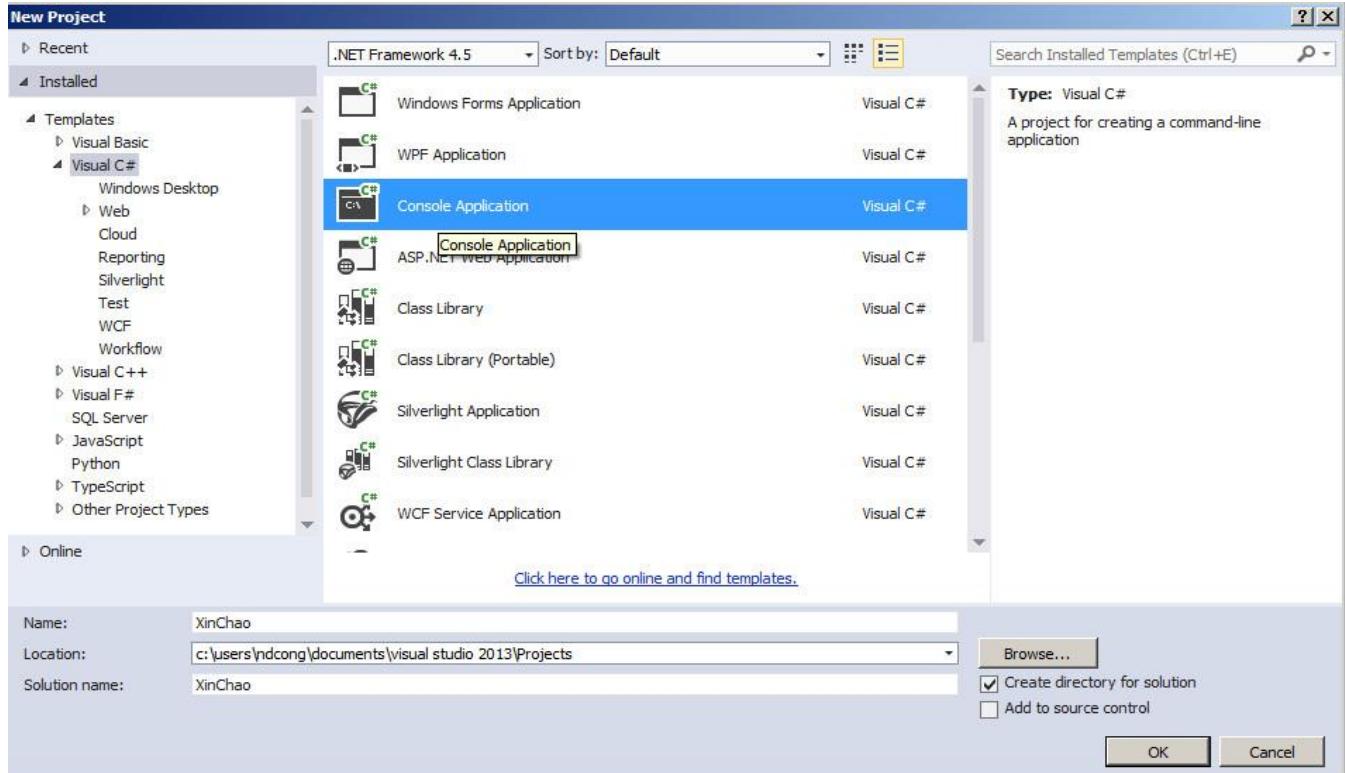
- Kỹ sư lập trình có thể sử dụng công cụ trực tiếp có sẵn trong Raspi đó là công cụ monodevelop để lập trình.



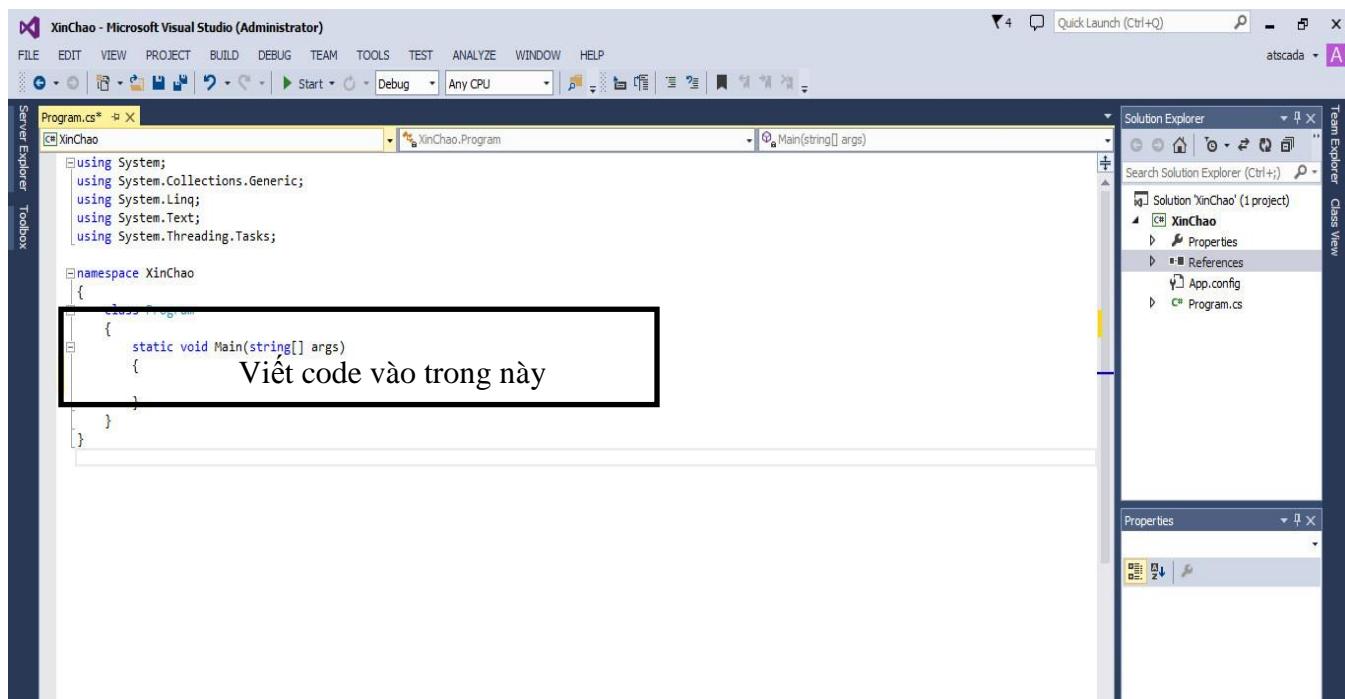
- Hoặc kết nối CPU Rasp01-CH với máy tính và lập trình bằng Visual Studio 2013 community, sau đó dùng phần mềm WinSCP để đưa file ứng dụng sang CPU Rasp01-CH để chạy (lập trình từ xa)
- Tải về Visual Studio 2013 community tại đây <http://atscada.com/plcpi/> để cài đặt.
- Sau khi cài đặt phần mềm xong, ta mở phần mềm lên sẽ có giao diện như hình



- Vào File/New/Project, xuất hiện giao diện



- + Ở cột bên trái ta chọn Visual C #, ô ở giữa chọn Console Application, các mục: Name, Localtion, Solution name là tên và vị trí lưu dự án
- Sau khi chọn xong ta nhấn OK, xuất hiện giao diện làm việc



The screenshot shows the Microsoft Visual Studio interface with the following details:

- Title Bar:** XinChao - Microsoft Visual Studio (Administrator)
- Menu Bar:** FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST ANALYZE WINDOW HELP
- Toolbars:** Standard, Debug, Start, Task List, Status Bar.
- Code Editor:** Shows Program.cs with the following code:

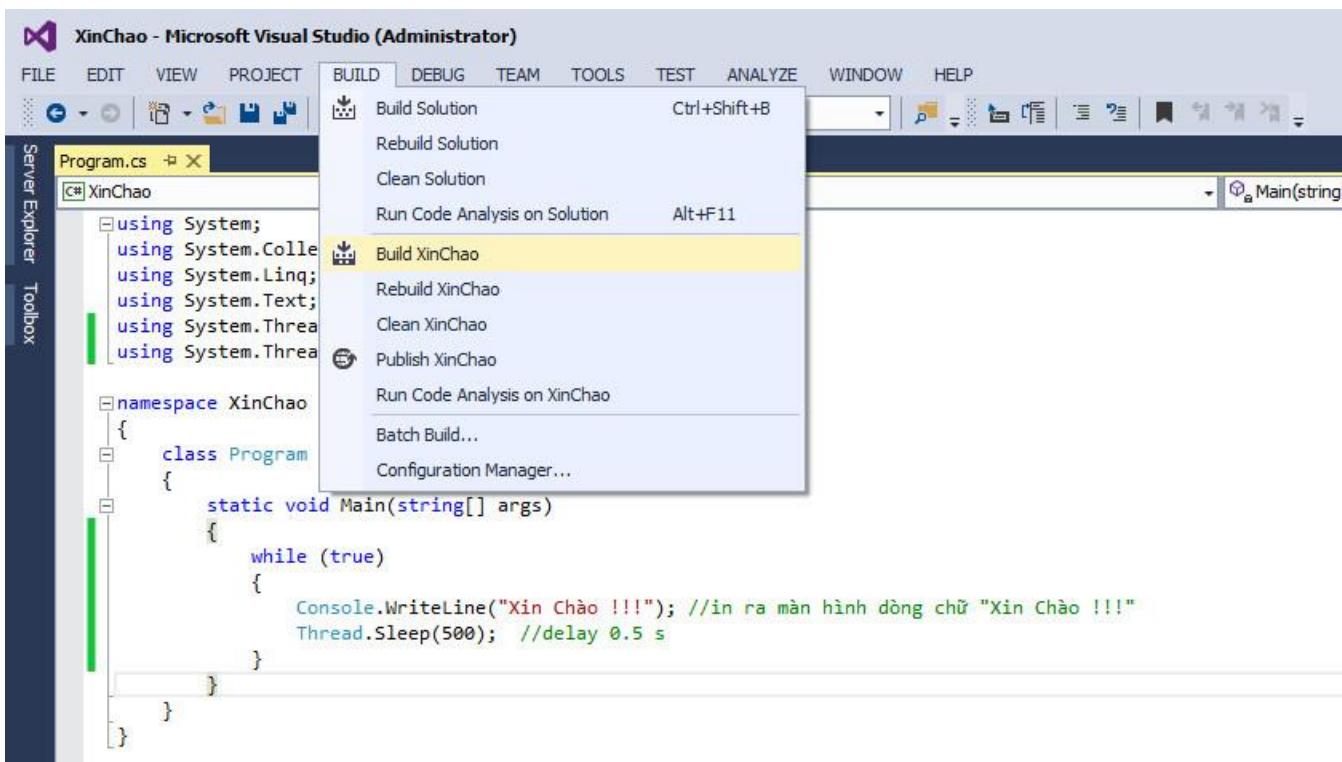
```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Threading;

namespace XinChao
{
    class Program
    {
        static void Main(string[] args)
        {
            while (true)
            {
                Console.WriteLine("Xin Chào !!!"); //in ra màn hình dòng chữ "Xin Chào !!!"
                Thread.Sleep(500); //delay 0.5 s
            }
        }
    }
}

```
- Solution Explorer:** Shows the solution 'XinChao' with one project 'XinChao' containing 'Properties', 'References', 'App.config', and 'Program.cs'.
- Properties Window:** Empty.
- Status Bar:** Error List, Output, 100 %.

- Sau khi lập trình xong ta tiến hành build. Ta làm như sau: BUILD/Build XinChao



- Sau khi Build thành công sẽ có giao diện như hình:

The screenshot shows the Microsoft Visual Studio interface. The title bar reads "XinChao - Microsoft Visual Studio (Administrator)". The menu bar includes FILE, EDIT, VIEW, PROJECT, BUILD, DEBUG, TEAM, TOOLS, TEST, ANALYZE, WINDOW, and HELP. The toolbar has various icons for file operations. The code editor window displays the following C# code:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Threading;

namespace XinChao
{
    class Program
    {
        static void Main(string[] args)
        {
            while (true)
            {
                Console.WriteLine("Xin chào !!!"); // in ra màn hình dòng chữ "Xin chào !!!"
            }
        }
    }
}

```

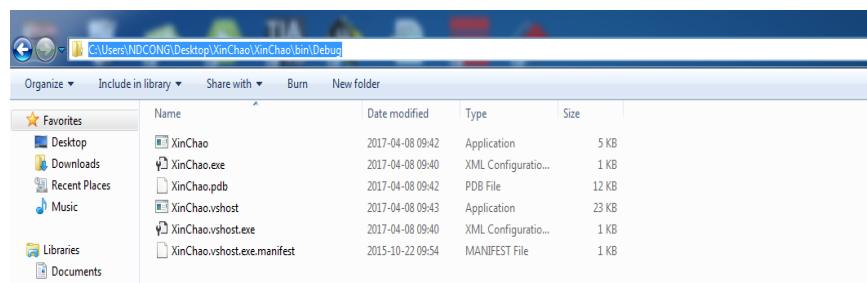
The output window at the bottom shows the build log:

```

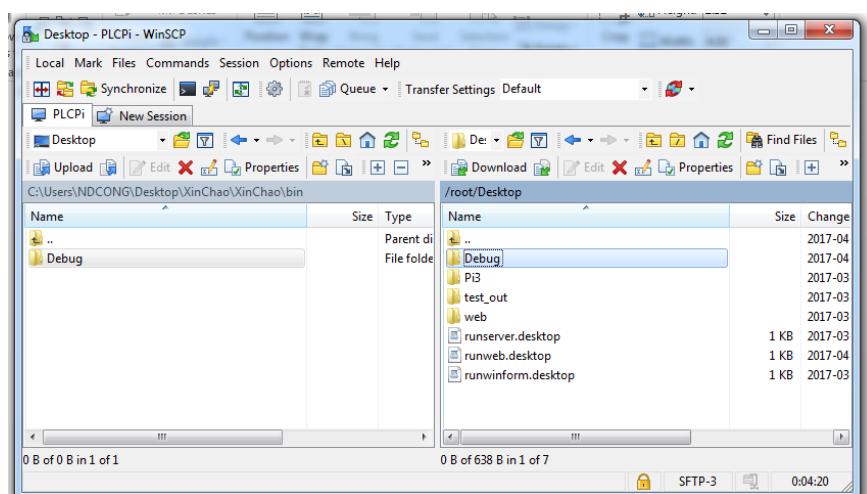
1>----- Build started: Project: XinChao, Configuration: Debug Any CPU -----
1> XinChao > c:\users\ndcong\documents\visual studio 2013\Projects\XinChao\XinChao\bin\Debug\XinChao.exe
===== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped =====

```

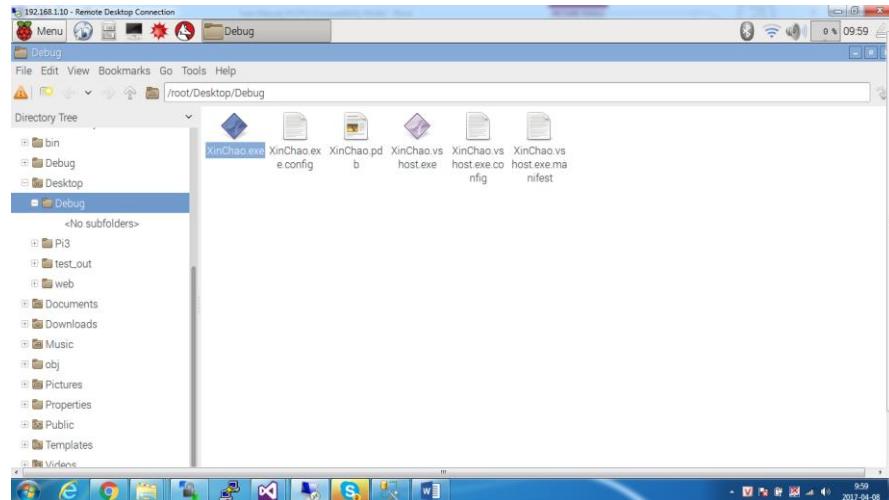
- + Sau khi build thì nó sẽ tạo ra file ứng dụng lưu trong thư mục Debug trong dự án



- + Sau khi build xong, giờ sẽ copy file Debug này sang PLCPi để chạy, dùng phần mềm WinSCP



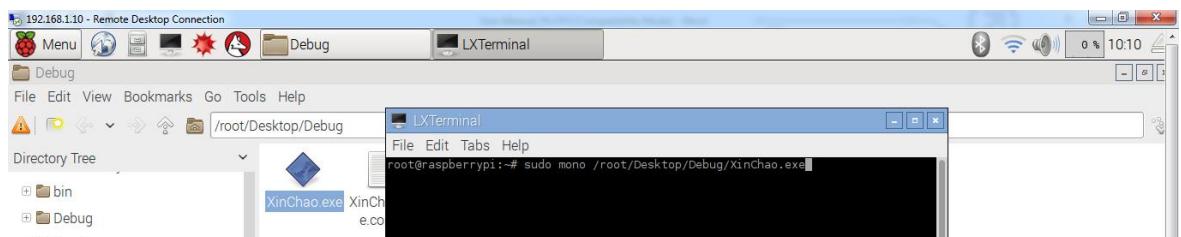
- + Sau khi copy file qua ta dùng Remote Desktop Connection để remote vào PLCPi để chạy demo ứng dụng



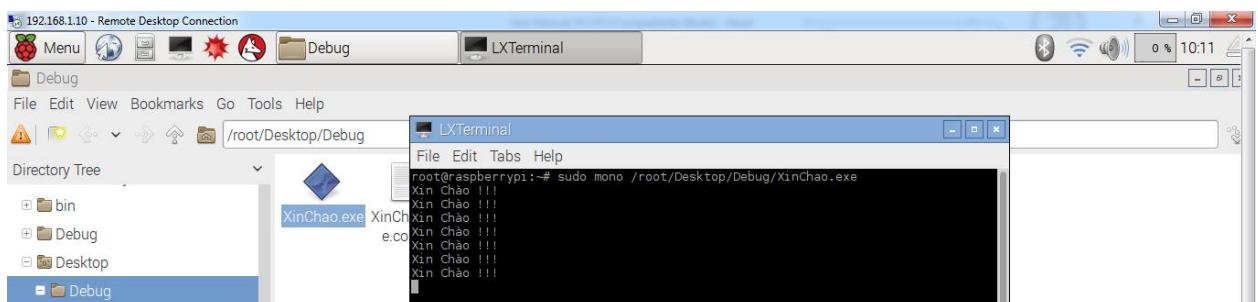
- + Để chạy ứng dụng, ta mở LXTerminal rồi nhập vào lệnh sau:



- Sudo mono path_to_file/filename
- Ví dụ ở đây: sudo mono /root/Desktop/Debug/XinChao.exe



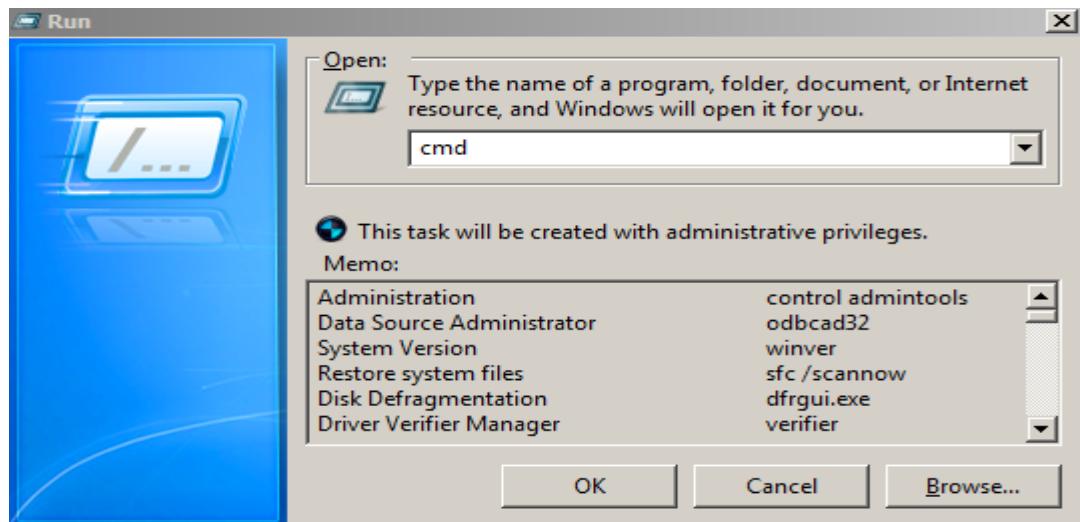
- + Kết quả:



- + Để dừng chạy ứng dụng, bấm Ctrl+C

3.3. Cách thay đổi địa chỉ IP cho PLCPi:

- Thông số mạng mặc định của PLCPi:
 - + IP address: 192.168.1.10
 - + Gateway: 192.168.1.1
- Nếu muốn thay đổi các thông số mạng mới cho PLCPi, ta thực hiện theo các bước bên dưới
 - B1. Kết nối PLCPi với mạng LAN qua dây cáp mạng, hoặc cắm trực tiếp vào máy tính rồi cài đặt các thông số của cổng LAN máy tính giống các thông số mạng của PLCPi
 - B2. Mở máy tính có kết nối cùng mạng LAN với PLCPi.
 - + Mở phần mềm **cdm** bằng cách nhấn tổ hợp phím **Start+R** sau đó gõ vào cdm.



- + Trong giao diện cmd gõ **ping + địa chỉ ip cũ** của PLCPi. Nếu tìm thấy ip PLCPi tức là kết nối tốt. Nếu không tìm thấy thì kiểm tra lại dây cáp, đường truyền, địa chỉ ip cũ của PLCPi đã nhập đúng chưa.

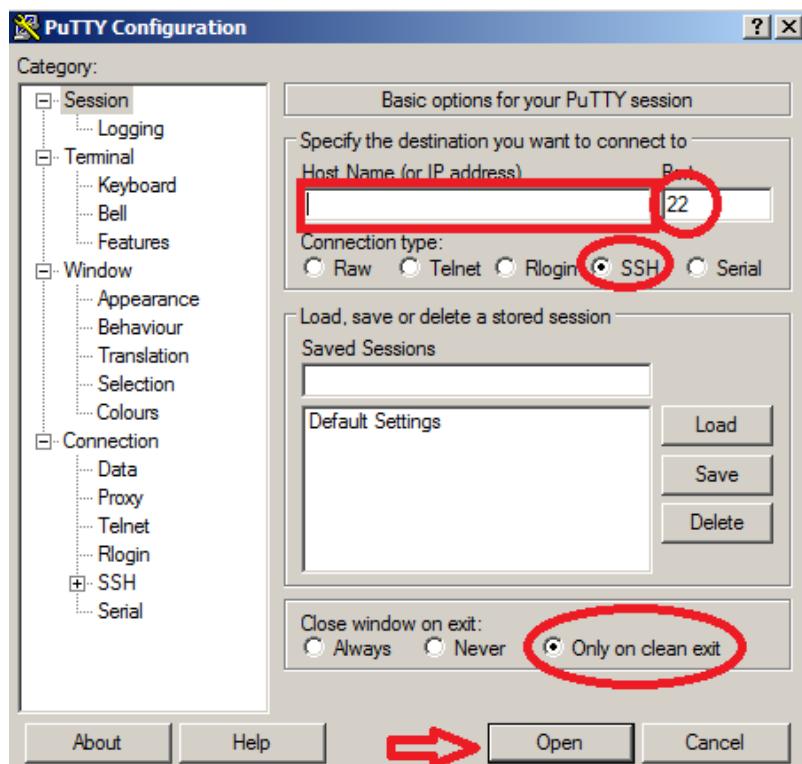
```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright <c> 2009 Microsoft Corporation. All rights reserved.

C:\Users\NDCONG>ping 192.168.1.10

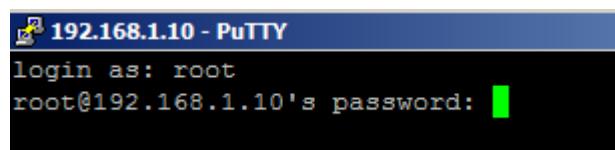
Pinging 192.168.1.10 with 32 bytes of data:
Reply from 192.168.1.10: bytes=32 time=44ms TTL=64
Reply from 192.168.1.10: bytes=32 time=1ms TTL=64
Reply from 192.168.1.10: bytes=32 time=1ms TTL=64
Reply from 192.168.1.10: bytes=32 time=1ms TTL=64

Ping statistics for 192.168.1.10:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 1ms, Maximum = 44ms, Average = 11ms
```

- B3. Tìm các thông số trong mạng LAN.
 - + Để cài ip mới cho đồng hồ cần có các thông số địa chỉ sau của mạng LAN:
 - Ip address
 - Gateway
- B4. Kết nối với PLCPi qua phần mềm PUTTY.
 - + Mở phần mềm PUTTY bằng cách click vào biểu tượng  , sau đó ta gõ địa chỉ ip cũ của PLCPi vào ô **Host Name** (chú ý các thông số khác được chọn như hình). Sau đó nhấn nút **Open**.



- + Tiếp theo sẽ xuất hiện màn hình đăng nhập vào PLCPi. Ta gõ vào **root** rồi nhấn **Enter** sau đó gõ tiếp **100100** rồi nhấn **Enter**.



- + Sau đó xuất hiện giao diện làm việc như sau:

```

192.168.1.10 - PuTTY
login as: root
root@192.168.1.10's password:

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Apr  4 13:28:43 2017 from 192.168.1.110
root@raspberrypi:~#

```

- B5. Thay đổi địa chỉ ip mới cho PLCPi.
 - + Sau khi đăng nhập vào PLCPi như bước 4(B4) ta được giao diện như ở trên. Ta thực hiện tiếp các bước sau để thay đổi ip mới.
 - + Tại giao diện làm việc ta gõ vào dòng lệnh
 - `sudo nano /etc/dhcpcd.conf`
 - + Sau đó nhấn phím **Enter** sẽ xuất hiện giao diện như bên dưới:

```

GNU nano 2.2.6           File: /etc/dhcpcd.conf
# A sample configuration for dhcpcd.
# See dhcpcd.conf(5) for details.

# Allow users of this group to interact with dhcpcd via the control socket.
#controlgroup wheel

# Inform the DHCP server of our hostname for DDNS.
hostname

# Use the hardware address of the interface for the Client ID.
clientid
# or
# Use the same DUID + IAID as set in DHCPv6 for DHCPv4 ClientID as per RFC4361.
#duid

# Persist interface configuration when dhcpcd exits.
persistent

# Rapid commit support.
[ Read 47 lines ]
^G Get Help  ^O WriteOut  ^R Read File  ^Y Prev Page  ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is   ^V Next Page  ^U Uncut Text ^T To Spell

```

- + Kéo xuống dưới như hình:

```

192.168.1.10 - PuTTY
GNU nano 2.2.6          File: /etc/dhcpd.conf

#option interface_mtu

# A ServerID is required by RFC2131.
require dhcp_server_identifier

# Generate Stable Private IPv6 Addresses instead of hardware based ones
slaac private

# A hook script is provided to lookup the hostname if not set by the DHCP
# server, but it should not be run by default.
nohook lookup-hostname
interface eth0

static ip_address=192.168.1.10/24
static routers=192.168.1.1
static domain_name_servers=8.8.8.8 8.8.4.4

^G Get Help  ^O WriteOut  ^R Read File  ^Y Prev Page  ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is   ^V Next Page  ^U UnCut Text^T To Spell

```

- + Static ip_address: là ip tĩnh của PLCPi
- + Static router: là default gateway
- + Để thay đổi thì ta thay đổi lại 2 thông số này rồi sau đó lưu lại

```

192.168.1.10 - PuTTY
GNU nano 2.2.6          File: /etc/dhcpd.conf          Modified

#option interface_mtu

# A ServerID is required by RFC2131.
require dhcp_server_identifier

# Generate Stable Private IPv6 Addresses instead of hardware based ones
slaac private

# A hook script is provided to lookup the hostname if not set by the DHCP
# server, but it should not be run by default.
nohook lookup-hostname
interface eth0

static ip_address=192.168.0.100/24
static routers=192.168.0.1
static domain_name_servers=8.8.8.8 8.8.4.4

^G Get Help  ^O WriteOut  ^R Read File  ^Y Prev Page  ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is   ^V Next Page  ^U UnCut Text^T To Spell

```

- + Lưu ý: IP mới không được trùng với bất kỳ địa chỉ IP nào đã được sử dụng ở trong mạng LAN.
- + Sau khi thay đổi các thông số xong ta nhấn tổ hợp phím **Ctrl +X**. Sau đó nhấn phím **Y**, rồi phím **Enter**, để lưu lại các thay đổi. Sau đó sẽ trở về giao diện làm việc.

```

login as: root
root@192.168.1.10's password:

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Apr  4 13:28:43 2017 from 192.168.1.110
root@raspberrypi:~# sudo nano /etc/dhcpd.conf
Use "fg" to return to nano.

[1]+  Stopped                  sudo nano /etc/dhcpd.conf
root@raspberrypi:~# 
```

- + Ở giao diện làm việc ta gõ dòng lệnh `sudo reboot` để Restart lại PLCPi.
- + Kiểm tra lại xem PLCPi đã được thay đổi ip mới chưa bằng cách thực hiện lại **B2** với ip là **ip mới đã đổi** của PLCPi.

3.4. Kết nối USB 3G:

- Để kết nối usb 3g, thì ta phải thao tác trực tiếp trên PLCPi, tức là ta phải cầm màn hình bàn phím và chuột vào PLCPi để thao tác.
- Cắm usb 3g vào cổng usb của PLCPi, đợi khoảng 5s rồi mở LXTerminal lên nhập vào “`ls /dev/ttyUSB*`”, rồi nhấn enter ta được danh sách các cổng serial như hình dưới. USB 3g khi cắm vào PLCPi sẽ chiếm 3 cổng liên tiếp nhau. Ở đây, nó chiếm từ cổng ttyUSB0 đến ttyUSB2

```

LXTerminal
File Edit Help
root@raspberrypi:~# ls /dev/ttyUSB*
/dev/ttyUSB0  /dev/ttyUSB1  /dev/ttyUSB2
root@raspberrypi:~# 
```

- Nếu bắt đầu bằng ttyUSB0 thì ta không cần phải chỉnh sửa file “`wvdial.conf`”, còn nếu bắt đầu bằng cổng khác ttyUSB0 thì ta phải vào chỉnh sửa file “`wvdial.conf`” trước khi kết nối. Cách làm như sau:
 - + Giả sử ở đây cắm 2 cái usb 3g vào PLCPi, chạy lệnh “`ls /dev/ttyUSB*`”

```

LXTerminal
File Edit Tabs Help
root@raspberrypi:~# ls /dev/ttyUSB*
/dev/ttyUSB0  /dev/ttyUSB2  /dev/ttyUSB4
/dev/ttyUSB1  /dev/ttyUSB3  /dev/ttyUSB5
root@raspberrypi:~#

```

- 1 cái chiết từ cổng ttyUSB0 → ttyUSB2
- 1 cái chiết từ cổng ttyUSB3 → ttyUSB5
- + Nếu muốn kết nối 3g từ cổng ttyUSB3 thì ta phải sửa lại file *wvdial.conf*
- + Tại cửa sổ LXTerminal nhập vào “*sudo nano /etc/wvdial.conf*” rồi bấm enter, ta được như hình dưới

```

LXTerminal
File Edit Tabs Help
GNU nano 2.2.6          File: /etc/wvdial.conf           Modified
[Dialer Defaults]
Init1 = ATZ
Init2 = AT&T00 V1 E1 S0=0 &C1 &D2 +FCLASS=0
Init3 = AT+CGDCONT=1,"IP","v-internet"
Stupid Mode = 1
Modem Type = Analog Modem
ISDN = 0
Phone = *99#
Modem_F=/dev/ttyUSB0
Username = { }
Password = { }
Baud = 460800

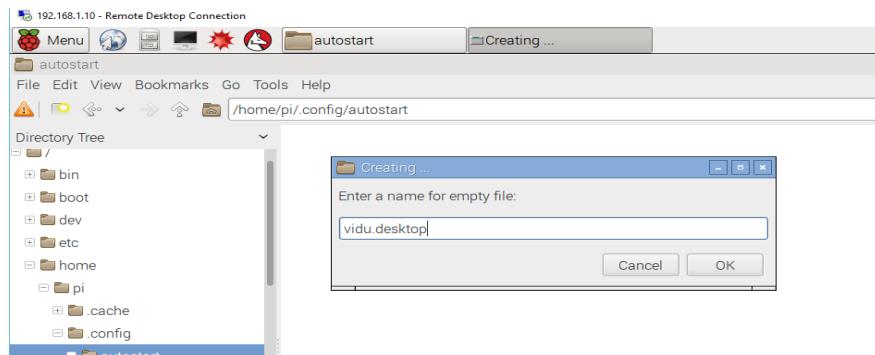
```

- Thay *ttyUSB0* bằng cổng *ttyUSB3*. Sau đó bấm *Ctrl+x* → *y* → *Enter* để lưu lại
- Tiến hành kết nối 3g
 - + Tại cửa sổ LXTerminal nhập vào “*wvdial*” rồi bấm Enter, đợi nó kết nối, sau ít giây nếu kết nối 3g thành công sẽ có giao diện như hình, gồm các thông số mạng như: IP Address, ...

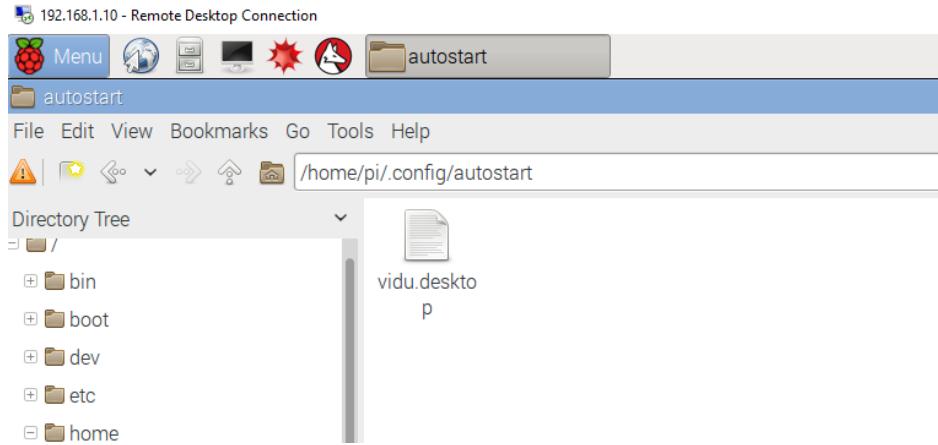
3.5. Cài đặt để chương trình tự khởi động khi boot

3.5.1. Chương trình Console:

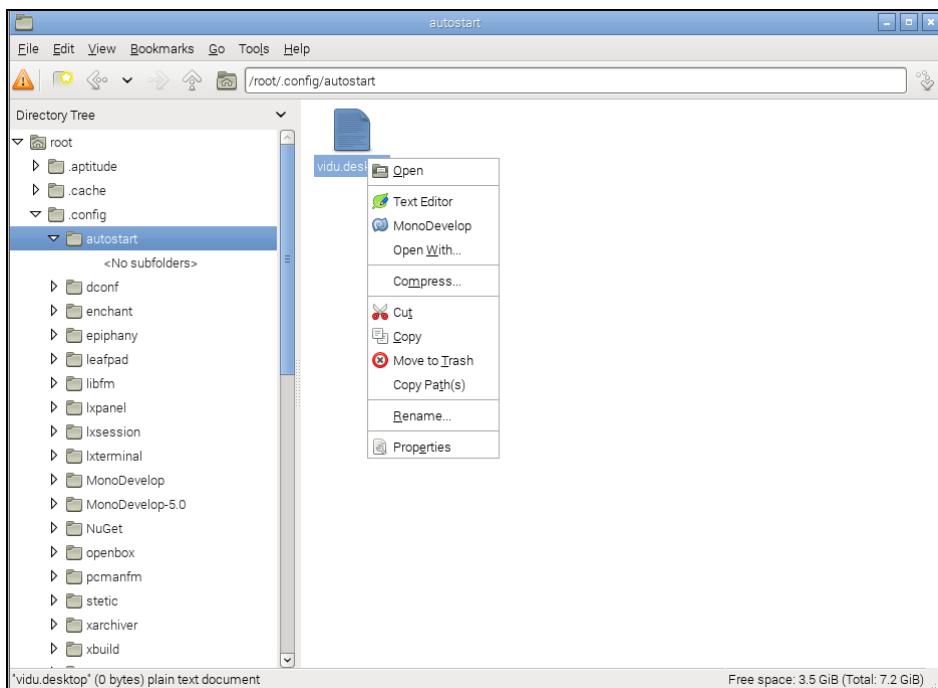
- Để tự động chạy một chương trình Console Application khi Boot ta làm theo các bước sau:
 - Truy nhập vào PLCPi qua phần mềm Remote Desktop Connection.
 - Mở File Manager vào thư mục autostart theo đường dẫn `/home/pi/.config/autostart`
 - Tại thư mục autostart, click chuột phải chọn Create New →Empty File



- Đặt tên file có dạng *filename.desktop*. Bấm Ok ta được một file có đuôi .desktop.



- Click chuột phải vào file vừa tạo chọn mở với trình Text Editor.



- Nhập vào các thông số như sau:

[Desktop Entry]

Name= vidu

Comment= Vi du chay autorun Console Application

Exec=lxterminal --command “sudo mono path_to_file/filename”

Icon=/usr/share/pixmaps/dillo.xpm

Type=Application

Encoding=UTF-8

Terminal=false

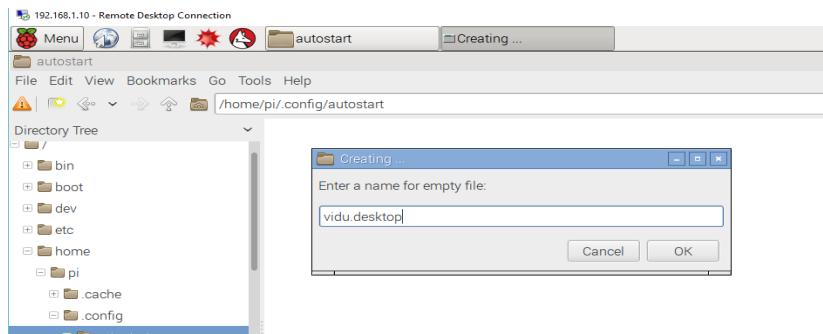
Categories=None;



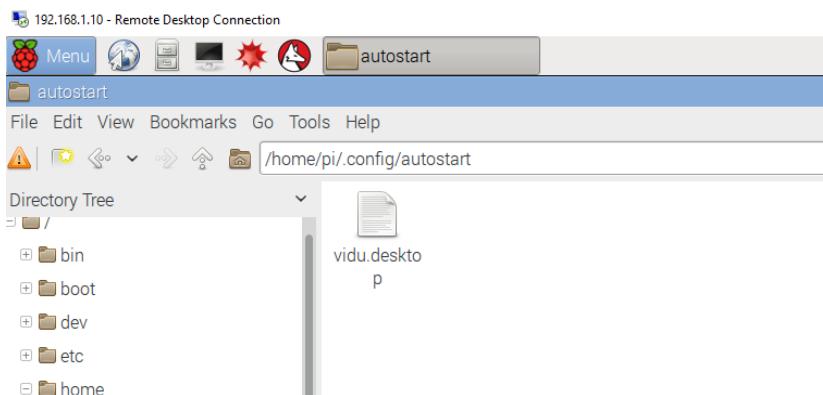
- Lưu file lại bằng Ctrl+S sau đó reboot.

3.5.2. Chương trình HMI Winform:

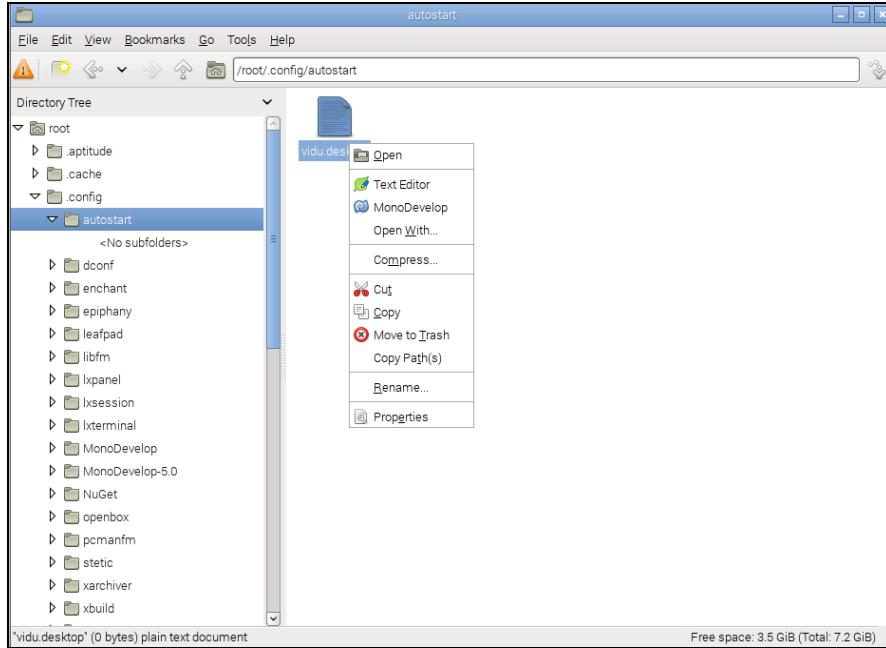
- Để tự động chạy một chương trình Winform Application khi Boot ta làm theo các bước sau:
- Truy nhập vào PLCPi qua phần mềm Remote Desktop Connection.
- Mở File Manager vào thư mục autostart theo đường dẫn `/home/pi/.config/autostart`
- Tại thư mục autostart, click chuột phải chọn Create New →Empty File



- Đặt tên file có dạng `filename.desktop`. Bấm Ok ta được một file có đuôi .desktop.



- Click chuột phải vào file vừa tạo chọn mở với trình Text Editor.



- Nhập vào các thông số như sau:

[Desktop Entry]

Name= vidu

Comment= Vi du chay autorun Winform_HMI_Application

Exec= sudo mono path_to_file/filename

Icon=/usr/share/pixmaps/dillo.xpm

Type=Application

Encoding=UTF-8

Terminal=false

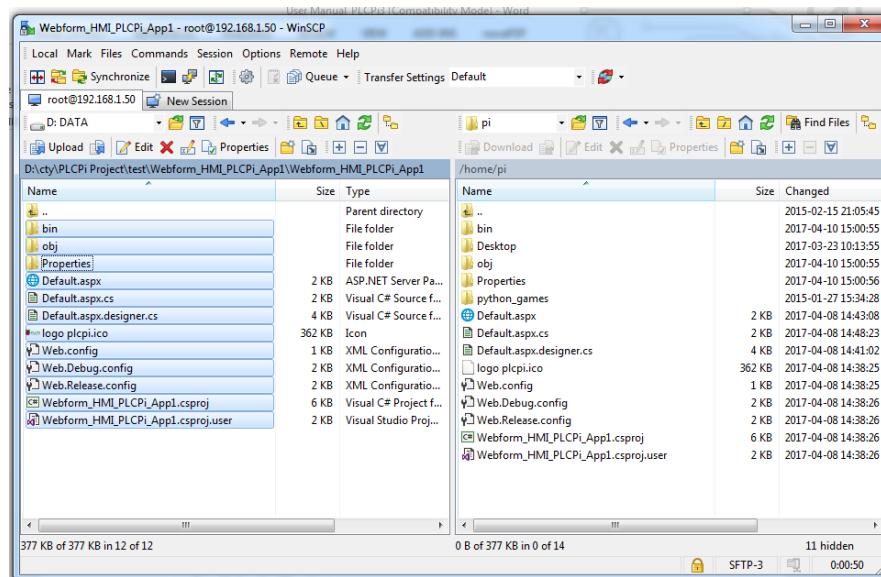
Categories=None;

- Lưu file lại bằng Ctrl+S sau đó reboot.

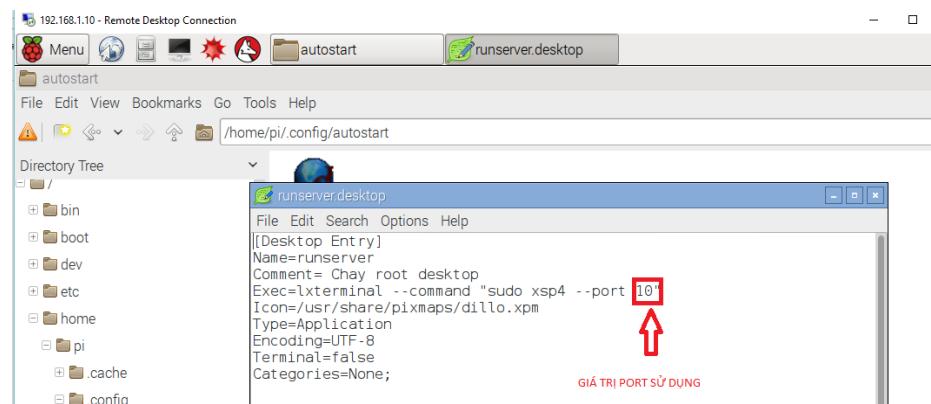
3.5.3. Chương trình HMI Webform:

Để tự động chạy một chương trình HMI Webform Application khi Boot ta làm theo các bước sau:

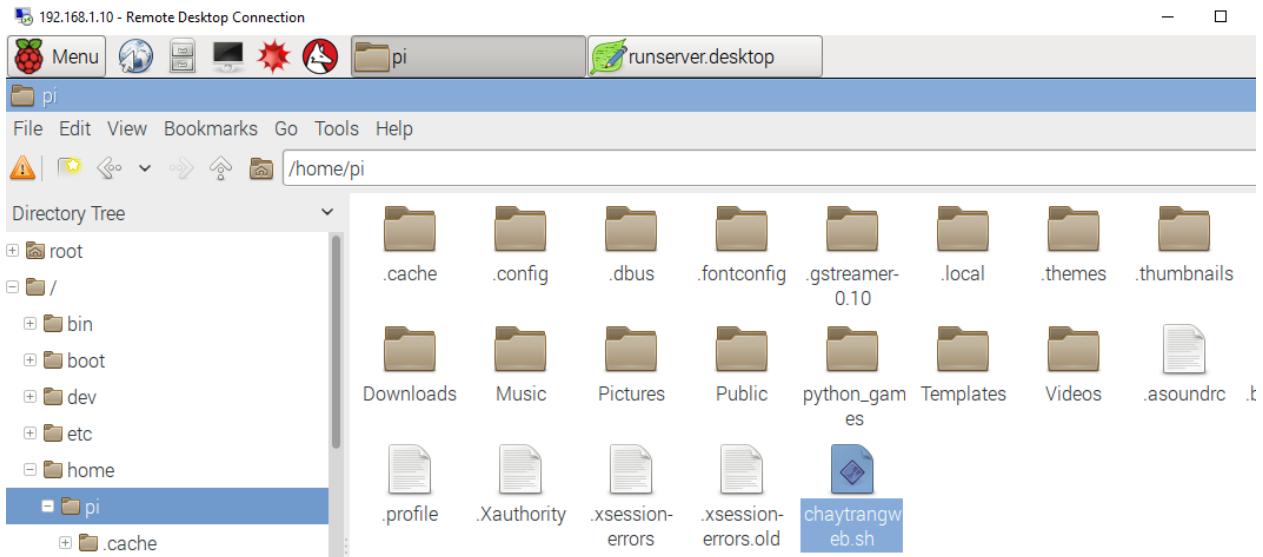
- Copy toàn bộ các file và folder của project web mình vừa tạo vào đường dẫn “/home/pi” trên PLCPi như hình



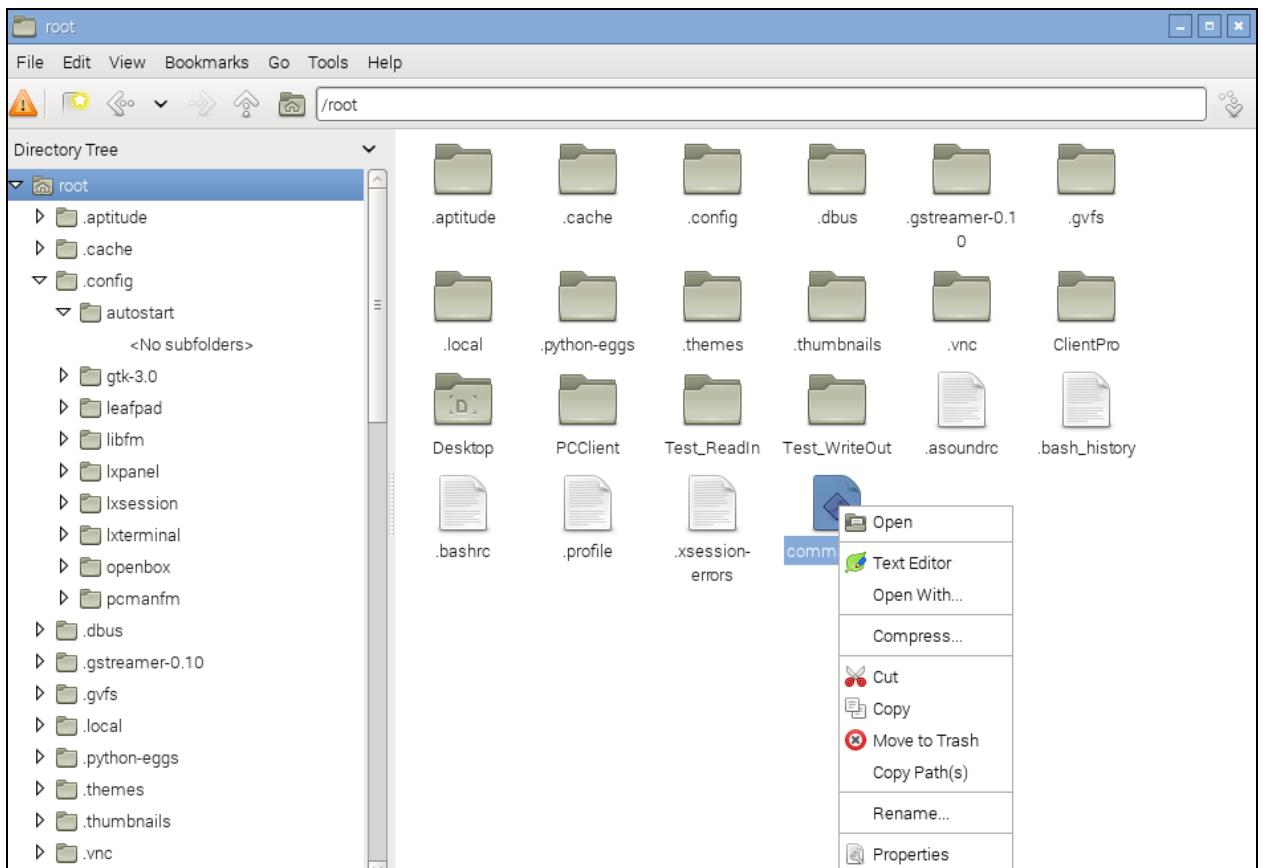
- Tạo file tự động chạy server.
 - + Vào Folder autostart như các bước trên.
 - + Tạo một file **runserver.desktop** để chạy server như các bước trên. Với các thông số như sau:



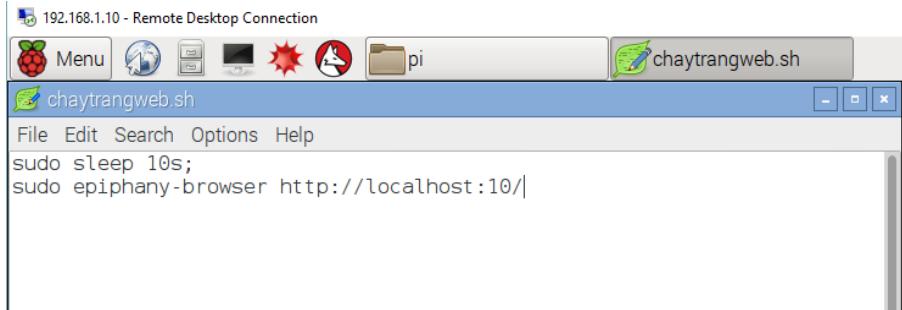
- + Sau đó lưu lại file vừa tạo.
- Tạo file tự động chạy trang web.
 - + Tại thư mục /home/pi tạo một file **chaytrangweb.sh**.



- + Click chuột phải vào file đã tạo chọn Test Editor.

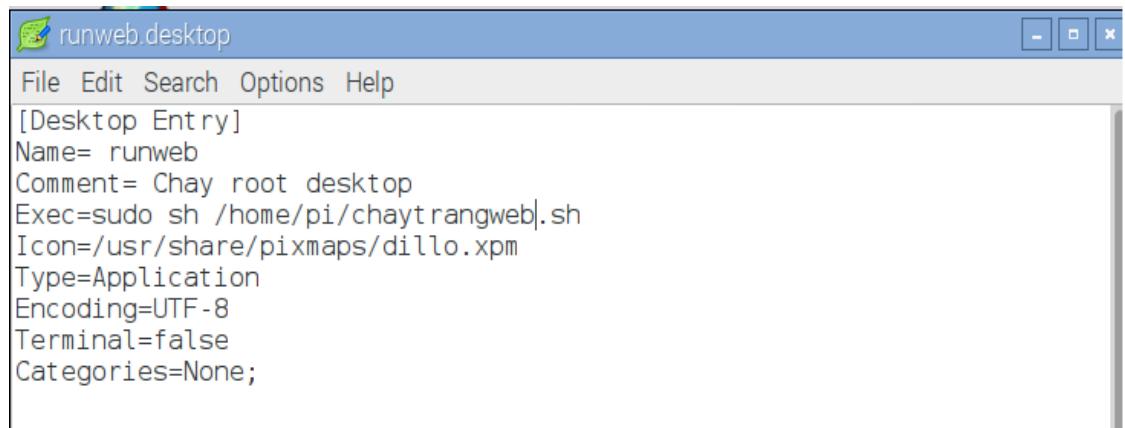


- + Ta nhập vào các thông số như sau và lưu lại.
 - *Sleep 10s;*
 - *Epiphany-browser http://localhost:Port/*

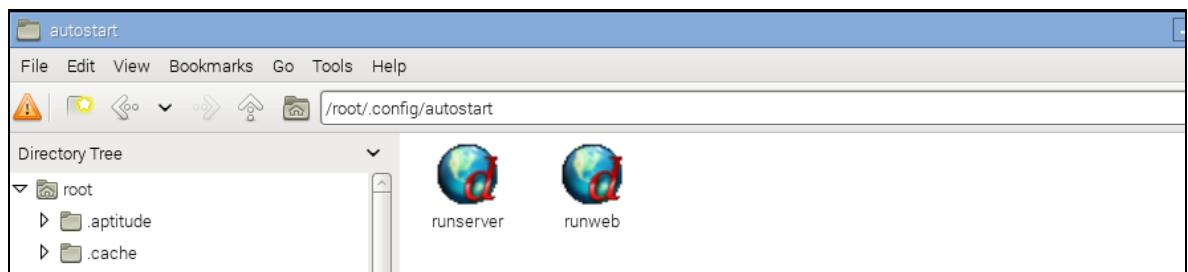


- Tiếp theo ta tạo một file **runweb.desktop** tại thư mục **/home/pi/.config/autostart** để tự động mở một trang web chỉ định khi boot.

+ Nhập vào các thông số như sau, sau đó lưu lại:



- Hoàn thành, ta được kết quả sau.



- Ta tiến hành reboot PLCPi.

3.5.4. Kết nối 3G để dung internet:

- Tạo file “auto3g.desktop” trong thư mục “/home/pi/.config/autostart”
- Nhập nội dung sau vào rồi lưu lại:

```
[Desktop Entry]
Name=auto3g
Comment= Chay root desktop
Exec= lxterminal --command "sudo wvdial 3gconnect"
Icon=/usr/share/pixmaps/dillo.xpm
Type=Application
Encoding=UTF-8
Terminal=false
Categories=None;
```

- Sau khi làm xong, mỗi lần ta reboot PLCPi thì nó sẽ tự kết nối 3G, nếu ta cắm dcom 3G vào PLCPi

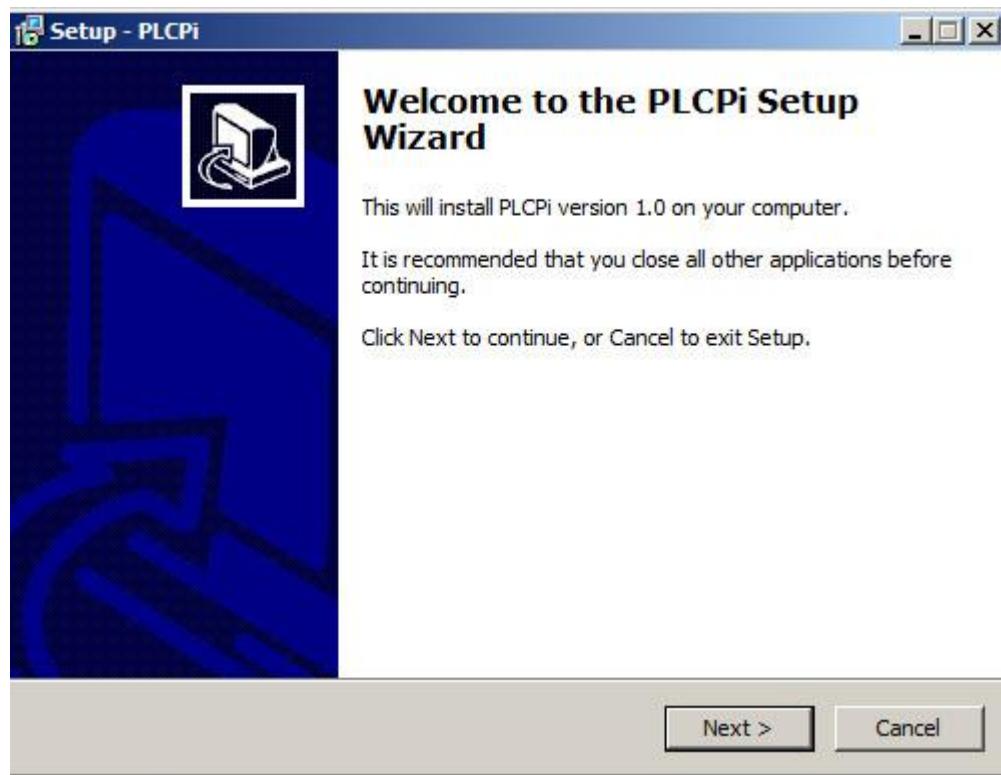
4. LẬP TRÌNH PHẦN MỀM CHO PLCPi:

4.1. Cài đặt môi trường lập trình cho PLCPi: VS2015 Community + PLCPi.exe

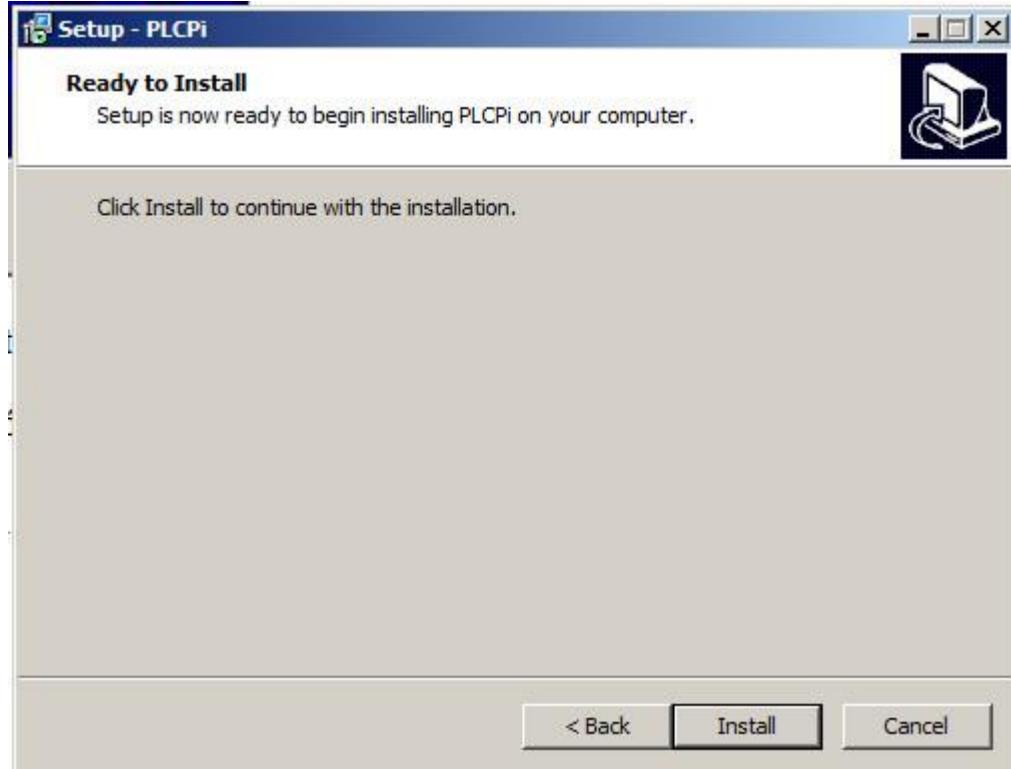
- Trước khi cài đặt thư viện lập trình *PLCPi.exe* thì máy tính cần phải cài đặt phần mềm Visual Studio 2013 community, tải về tại <http://atscada.com/plcpi/> (mục Visual Studio 2013 Community)
- Tải thư viện lập trình PLCPi.exe về tại đây: <http://atscada.com/plcpi/> (mục PLCPi Programming software)
- Cài đặt thư viện lập trình PLCPi:
 - + Kích đúp vào file PLCPi.exe



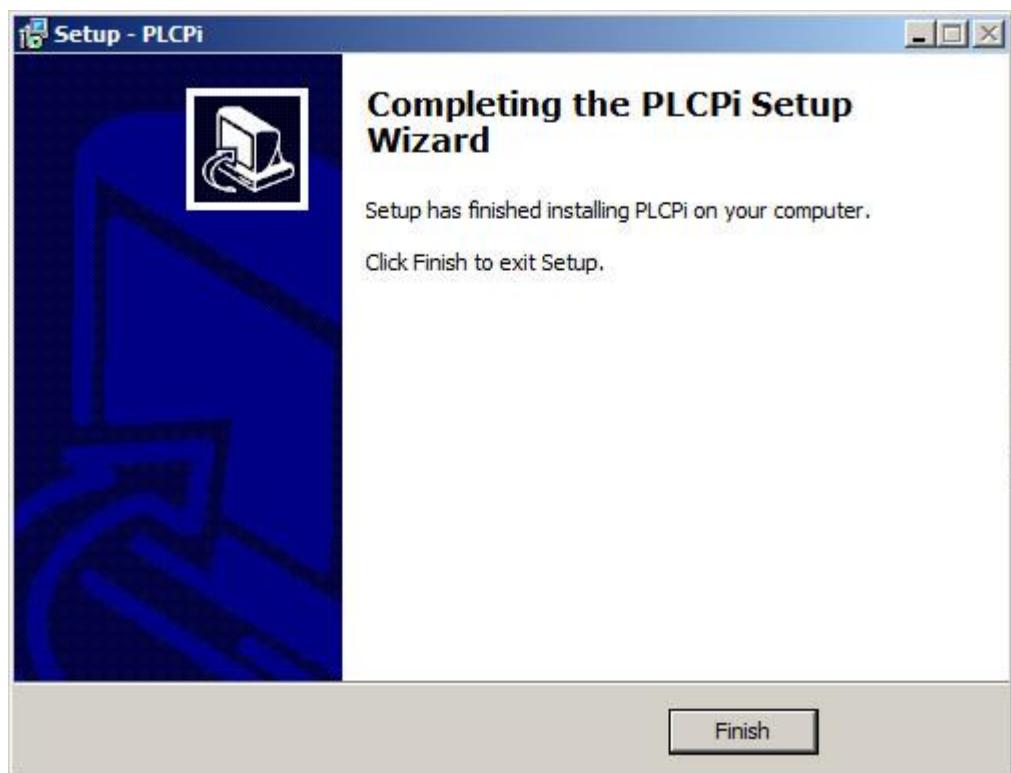
- + Xuất hiện giao diện



- + Bấm Next



- + Bấm Install, đợi chương trình cài đặt một lúc, hiện lên thông báo dưới



- + Bấm Finish.Ta cài đặt xong

4.2. Giới thiệu thư viện PLCPi.dll:

- Đây là thư viện dùng để lập trình ứng dụng PLC và HMI cho PLCPi bằng phương pháp lập trình từ xa trên máy tính sử dụng môi trường Visual Studio 2013 community.
- Thư viện gồm các tính năng sau:
 - + NgoVao: đọc trạng thái của các ngõ vào
 - + NgoRa: xuất dữ liệu cho các ngõ ra, đọc trạng thái hiện tại của các ngõ ra
 - + AI: đọc tín hiệu Analog từ các ngõ vào AI
 - + DS18B20: đọc giá trị nhiệt độ từ cảm biến DS18B20
 - + DHT21: đọc giá trị nhiệt độ độ ẩm từ cảm biến DHT21
 - + ThoiGian: đọc thời gian từ PLCPi
 - + mySQLLogger: thao tác với cơ sở dữ liệu MySQL
 - + SQLLogger: thao tác với cơ sở dữ liệu SQLServer
 - + HienThiLed7: hiển thị các giá trị cần hiển thị ra module Display
 - + SMS: gửi tin nhắn để cảnh báo. Sử dụng GSM modem trên USB 3G
 - + Gọi điện thoại: gọi điện thoại không âm đến 1 số điện thoại nào đó. Sử dụng GSM modem trên USB 3G
 - + Email: gửi email
 - + Đọc ghi dữ liệu từ mảng byte: đọc và ghi dữ liệu có kiểu dữ liệu khác kiểu byte vào mảng kiểu byte.
 - + S7 Ethernet: truyền thông S7Ethernet Client và Server
 - + ModbusTCPClient: dùng để truyền thông modbus TCP
 - + ModbusRTUMaster: dùng để truyền thông modbus RTU

4.3. Hướng dẫn sử dụng các tính năng trong thư viện PLCPi.dll

4.3.1. NgoVao:

- Đọc trạng thái các ngõ vào của PLCPi. Có 2 cách đọc, theo Bit hoặc theo Byte
- Đọc theo Bit: đọc 1 bit của ngõ vào
 - + NgoVao.DocNgoVao(string Channel)
 - + Đôi số:
 - Channel là ngõ vào muốn đọc, có kiểu dữ liệu là kiểu chuỗi
 - Đôi số Channel có dạng “Ix.y”. Trong đó, x là ngõ vào ta muốn đọc(0 đến 4), y là bit của ngõ vào đó ta muốn đọc(0 đến 7)

- + Kết quả trả về kiểu byte có giá trị là 0 hoặc 1
- + Ví dụ: đọc ngõ vào I0.5

```
PLCPi myPLC = new PLCPi(); //tạo đối tượng myPLC
byte data = myPLC.NgoVao.DocNgoVao("I0.5");
```

- Đọc theo Byte: đọc 8 bit của 1 ngõ vào 1 lúc

- + NgoVao.DocNgoVao(string Channel)

- + Đôi số:
 - Channel là ngõ vào muốn đọc, có kiểu dữ liệu là kiểu chuỗi
 - Đôi số Channel có dạng “Ix”. Trong đó, x là ngõ vào ta muốn đọc(0 đến 4)

- + Kết quả trả về kiểu byte

- + Ví dụ: đọc ngõ vào IO

```
PLCPi myPLC = new PLCPi(); //tạo đối tượng myPLC
byte data = myPLC.NgoVao.DocNgoVao("I0");
```

4.3.2. NgoRa:

Có 2 chức năng. XuatNgoRa và DocNgoRa

- **XuatNgoRa:** Xuất giá trị cho các ngõ ra của PLCPi. Có 2 cách xuất: xuất từng bit hoặc xuất nguyên

1 byte

- + Xuất bit: xuất ngõ ra theo từng bit

- NgoRa.XuatNgoRa(string Channel, byte Value)

- Đôi số:

- * Channel là ngõ ra muốn xuất dữ liệu, có kiểu dữ liệu là kiểu chuỗi

- * Đôi số Channel có dạng “Qx.y”. Trong đó, x là ngõ ra ta muốn xuất (0 đến 5), y là bit của ngõ ra đó ta muốn xuất(0 đến 7)

- * Value là trạng thái ta muốn xuất, có kiểu dữ liệu là kiểu byte. Ở đây đang xuất theo bit nên giá trị của đôi số Value chỉ là 0 hoặc 1(tắt hoặc bật)

- Ví dụ:

```
PLCPi myPLC = new PLCPi(); //tạo đối tượng myPLC
myPLC.NgoRa.XuatNgoRa("Q0.0", 1); //Bật ngõ ra Q0.0
myPLC.NgoRa.XuatNgoRa("Q0.0", 0); //Tắt ngõ ra Q0.0
```

- + Xuất byte: xuất ngõ ra theo từng byte

- NgoRa.XuatNgoRa(string Channel, byte Value)

- Đổi số:
 - * Channel là ngõ ra muốn xuất dữ liệu, có kiểu dữ liệu là kiểu chuỗi
 - * Đổi số Channel có dạng “Qx”. Trong đó, x là ngõ ra ta muốn xuất (0 đến 5)
 - * Value là dữ liệu ta muốn xuất, có kiểu dữ liệu là kiểu byte
- Ví dụ:

```
PLCPi myPLC = new PLCPi(); //tạo đối tượng myPLC
myPLC.NgoRa.XuatNgoRa("Q2", 255); // Xuất 8 bit của ngõ ra Q2 lên 1
```

- **DocNgoRa:** Đọc trạng thái hiện tại của các ngõ ra. Có 2 cách đọc: theo bit hoặc theo byte
 - + Đọc bit: đọc trạng thái của các ngõ vào theo từng bit
 - NgoRa.DocNgoRa(string Channel)
 - Đổi số:
 - * Channel là ngõ ra muốn đọc trạng thái về, có kiểu dữ liệu là kiểu chuỗi
 - * Đổi số Channel có dạng “Qx.y”. Trong đó, x là ngõ ra muốn đọc về (0 đến 5), y là bit của ngõ ra đó (0 đến 7)
 - Dữ liệu trả về kiểu Byte, có giá trị 0 hoặc 1
 - Ví dụ: đọc trạng thái ngõ ra Q0.0


```
PLCPi myPLC = new PLCPi(); //tạo đối tượng myPLC
byte data = myPLC.NgoRa.DocNgoRa("Q0.0");
```
 - + Đọc byte: đọc trạng thái của các ngõ vào theo byte
 - NgoRa.DocNgoRa(string Channel)
 - Đổi số:
 - * Channel là ngõ ra muốn đọc trạng thái về, có kiểu dữ liệu là kiểu chuỗi
 - * Đổi số Channel có dạng “Qx”. Trong đó, x là ngõ ra muốn đọc về (0 đến 5)
 - Dữ liệu trả về kiểu Byte
 - Ví dụ: đọc trạng thái ngõ ra Q0


```
PLCPi myPLC = new PLCPi(); //tạo đối tượng myPLC
byte data = myPLC.NgoRa.DocNgoRa("Q0");
```

4.3.3. AI:

- Đọc các ngõ vào Analog từ module AI, có 3 chức năng. DocAI1Khenh, DocAI, Scale
- ADC có độ phân giải 10bit, nên giá trị ADC chạy từ 0 đến 1024

- **DocAI1Kenh:** đọc 1 kênh ngõ vào AI
 - + AI.DocAI1Kenh(Int16 Channel, double X0, double Y0, double X1, double Y1)
 - + Đổi số:
 - Channel: kênh AI muốn đọc, kiểu dữ liệu Int16, có giá trị từ 0 đến 12
 - X0: giá trị ADC nhỏ nhất, kiểu dữ liệu double
 - * Nếu đọc các kênh AI 0-10VDC thì X0 = 0
 - * Nếu đọc các kênh AI 4-20mA thì X0 = 204
 - X1: giá trị ADC lớn nhất, kiểu dữ liệu double, X1 = 1024
 - Y0: giá trị Analog nhỏ nhất, kiểu dữ liệu double
 - Y1: giá trị Analog lớn nhất, kiểu dữ liệu double
 - * 2 giá trị Y0 và Y1 truyền vào tùy thuộc vào giá trị Analog của thiết bị.
 - * Ví dụ: *thiết bị đo tốc độ động cơ có ngõ ra ADC là từ 0-10VDC tương ứng với tốc độ là từ 0 đến 200 vòng . thì* $\begin{cases} Y0 = 0 \\ Y1 = 200 \end{cases}$
 - + Trả về giá trị Analog của kênh đó, dữ liệu kiểu string. Nếu dữ liệu trả về là “BAD” nghĩa là việc đọc ngõ vào AI lỗi.
 - + Ví dụ:
 - *Đọc tín hiệu ADC từ 1 thiết bị đo tốc độ động cơ có ngõ ra ADC 0-10VDC tương ứng với tốc độ từ 0 đến 200 vòng/s. Với tín hiệu ADC này thì ta đưa vào các ngõ vào AI 0-10VDC(AI0 đến 4)*

```
PLCPi myPLC = new PLCPi(); //tạo đối tượng myPLC
string data = myPLC.AI.DocAI1Kenh(0, 0, 0, 1024, 200);
//khi đó data sẽ chạy từ 0 đến 200 tùy thuộc vào tốc độ động cơ
```
 - *Đọc tín hiệu ADC từ 1 thiết bị đo nhiệt độ với ngõ ra ADC 4-20mA tương ứng với nhiệt độ từ 0 đến 400 °C. Với tín hiệu ADC này thì ta đưa vào các ngõ vào AI 0-10VDC(AI5 đến AI12)*

```
string data = myPLC.AI.DocAI1Kenh(5, 204, 0, 1024, 400);
//khi đó data sẽ chạy từ 0 đến 400 tùy thuộc vào nhiệt độ
```
- **DocAI:** Đọc 1 lần 13 kênh AI
 - + AI.DocAI()
 - + Trả về mảng 13 phần tử kiểu string chứa giá trị ADC của 13 kênh AI, theo thứ tự phần tử [0] chứa giá trị AI0, [1] chứa giá trị AI1,... Nếu phần tử nào trong mảng trả về là “BAD” nghĩa là việc đọc ngõ vào AI tương ứng lỗi.

- + Sau khi đọc xong, muốn lấy giá trị ADC của kênh nào thì truy xuất đến phần tử của mảng tương ứng
- + Ví dụ:


```
string[] data = myPLC.AI.DocAI();
```
- **Scale:** dùng để chuyển đổi giá trị ADC đọc lên từ ngõ vào AI về lại giá trị Analog
 - + AI.Scale(double X0, double Y0, double X1, double Y1, double x)
 - + Đôi số:
 - X0: giá trị ADC nhỏ nhất, kiểu dữ liệu double
 - * Nếu đọc các kênh AI 0-10VDC thì X0 = 0
 - * Nếu đọc các kênh AI 4-20mA thì X0 = 204
 - X1: giá trị ADC lớn nhất, kiểu dữ liệu double, X1 = 1024
 - Y0: giá trị Analog nhỏ nhất, kiểu dữ liệu double
 - Y1: giá trị Analog lớn nhất, kiểu dữ liệu double
 - * 2 giá trị Y0 và Y1 truyền vào tùy thuộc vào giá trị Analog của thiết bị.
 - * Ví dụ: *thiết bị đo tốc độ động cơ có ngõ ra ADC là từ 0-10VDC tương ứng với tốc độ là từ 0 đến 200 vòng . thì* $\begin{cases} Y0 = 0 \\ Y1 = 200 \end{cases}$
 - x: giá trị ADC truyền vào, kiểu dữ liệu double
 - + Trả về giá trị Analog, kiểu string. Nếu dữ liệu trả về là “BAD” nghĩa là việc chuyển đổi giá trị ADC về giá trị Analog lỗi.
 - + Ví dụ: ở ví dụ trong tính năng DocAI. Mảng data chứa giá trị ADC của 13 kênh ngõ vào
 - *Chuyển đổi giá trị ADC của kênh AI0 về giá trị Analog (0-150)*

```
string DuLieu = myPLC.AI.Scale(0, 0, 1024, 150, data[0]);
```
 - *Chuyển đổi giá trị ADC của kênh AI12 về giá trị Analog (0-250)*

```
string DuLieu = myPLC.AI.Scale(204, 0, 1024, 250, data[12]);
```

4.3.4. DS18B20:

- Đọc giá trị nhiệt độ từ cảm biến nhiệt độ DS18B20
- DS18B20.DocNhiетDo(string Id_DS18B20)
- Đôi số:
 - + Id_DS18B20: mã số ID của cảm biến DS18B20, kiểu dữ liệu là kiểu chuỗi
 - + ID của cảm biến sẽ hướng dẫn lấy ở ví dụ mẫu
- Giá trị nhiệt độ trả về kiểu string. Nếu giá trị trả về là “BAD” nghĩa là mất kết nối với cảm biến

- Ví dụ: đọc nhiệt độ từ DS18B20 có ID = 28-00042b536dff

```
PLCPi myPLC = new PLCPi(); //tạo đối tượng myPLC
string NhietDo = myPLC.DS18B20.DocNhietDo("28-00042b536dff");
```

4.3.5. DHT21:

- Đọc giá trị nhiệt độ độ ẩm từ cảm biến DHT21. Có 3 chức năng, DocNhietDo, DocDoAm, DocNhietDoDoAm

- DocNhietDo:

- + DHT21.DocNhietDo()

- + Giá trị nhiệt độ trả về kiểu string. Nếu giá trị trả về là “BAD” nghĩa là mất kết nối với cảm biến
- + Ví dụ:

```
PLCPi myPLC = new PLCPi(); //tạo đối tượng myPLC
string NhietDo = myPLC.DHT21.DocNhietDo();
```

- DocDoAm:

- + DHT21.DocDoAm()

- + Giá trị độ ẩm trả về kiểu string. Nếu giá trị trả về là “BAD” nghĩa là mất kết nối với cảm biến

- + Ví dụ:

```
string DoAm = myPLC.DHT21.DocDoAm();
```

- DocNhietDoDoAm:

- + DHT21.DocNhietDoDoAm()

- + Trả về mảng dữ liệu 2 phần tử kiểu string {nhiệt độ, độ ẩm}. Nếu mảng trả về có phần tử “BAD” nghĩa là mất kết nối với cảm biến

- + Ví dụ:

```
string[] T_H = myPLC.DHT21.DocNhietDoDoAm();
```

4.3.6. ThoiGian:

- Có 3 chức năng. DocThoiGian, CaiDat, CaiDatRTC_DS1307

- DocThoiGian: đọc thời gian hiện tại của PLCPi

- + ThoiGian.DocThoiGian()

- + Trả về mảng 6 phần tử kiểu string {ngày, tháng, năm, giờ, phút, giây}. Nếu mảng trả về có phần tử “BAD” nghĩa là việc đọc thời gian lỗi.

- + Ví dụ:

```
PLCPi myPLC = new PLCPi(); //tạo đối tượng myPLC
```

```

string[] Time = new string[6];
Time = myPLC.ThoiGian.DocThoiGian();
//Khi đó mảng Time sẽ chứa các giá trị ngày tháng năm giờ phút giây

```

- CaiDat: cài đặt thời gian cho PLCPi
 - + ThoiGian.CaiDat(string DateTime)
 - + Đôi số:
 - DateTime: giá trị ngày giờ muốn cài đặt, dữ liệu kiểu string, có định dạng “dd-MM-yyyy HH:mm:ss”
 - + Ví dụ:


```
myPLC.ThoiGian.CaiDat("25-02-2015 20:06:30");
```
- CaiDatRTC_DS1307: cài đặt thời gian cho module thời gian thực RTC, module này sẽ giữ cho PLCPi luôn có thời gian thực chính xác khi PLCPi bị mất điện.
 - + ThoiGian.CaiDatRTC()
 - + Ví dụ: `myPLC.ThoiGian.CaiDatRTC_DS1307();`

4.3.7. HienThiLed:

- HienThiLed:
 - + HienThiLed.HienThi(string SoCanHienThi, Int16 KenhSo)
 - + Đôi số:
 - SoCanHienThi: giá trị cần hiển thị, kiểu chuỗi. nếu SoCanHienThi = “Tat” thì sẽ tắt hiển thị
 - KenhSo: kênh muốn hiển thị, kiểu Int16, có 2 kênh, KenhSo = 1 hoặc 2
 - + Ví dụ: hiển thị giá trị độ ẩm đọc từ cảm biến DHT21 ra kênh 1, tắt kênh 2

```
PLCPi myPLC = new PLCPi(); //tạo đối tượng myPLC
```

```
string DoAm = myPLC.DHT21.DocDoAm();
```

```
myPLC.HienThiLed.HienThi(DoAm, 1);
```

```
myPLC.HienThiLed.HienThi("Tat", 2);
```

4.3.8. mySQLLogger:

Hàm	Mục đích
TaoKetNoi	Truyền chuỗi kết nối cho mySQLLogger
TaoBang	Hàm dùng để tạo bảng table trong mysql
GhiDuLieuVaoBang	Dùng để ghi dữ liệu mới vào bảng

- **TaoKetNoi:** Truyền chuỗi kết nối cho mySQLLogger. Trước khi log dữ liệu cần truyền giá trị tạo kết nối trước
 - + mySQLLogger.TaoKetNoi (string TenChuoiKetNoi, string TenDatabase)
 - + Đối số:
 - TenChuoiKetNoi: có dạng "Server = servername;Uid = username;Pwd = password"
 - TenDatabase: tên Database trong cơ sở dữ liệu
 - + Ví dụ:


```
PLCPi myPLC = new PLCPi(); // tạo đối tượng myPLC
//truyền chuỗi kết nối đến cơ sở dữ liệu MySQL muốn kết nối tới cho PLCPi
myPLC.mySQLLogger.TaoKetNoi("Server =127.0.0.1;Uid=root;Pwd=100100", "phatdb");
```
- **TaoBang:** Hàm dùng để tạo bảng table trong mysql. Nếu trùng tên bảng cũ thì bảng cũ sẽ bị xóa, thay bằng bảng mới. Nếu Database chưa tạo thì sẽ tạo mới database theo tên đã truyền vào khi tạo kết nối
 - + mySQLLogger.TaoBang(string TenBang, string[] MangTenCot)
 - + Đối số:
 - TenBang: Tên bảng muốn tạo
 - MangTenCot: Mảng chứa tên các cột muốn tạo trong bảng
 - + Trả về kết quả thực hiện kiểu string. Nếu dữ liệu trả về là “GOOD” nghĩa là đã tạo bảng thành công, “BAD” thất bại.
 - + Ví dụ:


```
String[] MangCot = {"DateTime", "Id", "Value"};
myPLC.mySQLLogger.TaoBang("DataLog", MangCot);
```
- **GhiDuLieuVaoBang:** Dùng để ghi dữ liệu mới vào bảng
 - + mySQLLogger.GhiDuLieuVaoBang(string TenBang, string[] MangGiaTRi)
 - + Đối số:
 - TenBang: Tên bảng cần truyền dữ liệu
 - MangGiaTRi: Mảng giá trị tương ứng với các cột trong bảng bắt đầu từ cột đầu tiên
 - + Trả về kết quả thực hiện kiểu string. Nếu dữ liệu trả về là “GOOD” nghĩa là ghi dữ liệu vào bảng thành công, “BAD” thất bại.
 - + Ví dụ:

```

string[] manggiatri = { myPLC.Datalogger.LayThoiGian(), "100", "200 "};
myPLC.mySQLLogger.GhiDuLieuVaoBang("DataLog", manggiatri);

```

- **TruyvanBaocao:** Dùng để truy vấn dữ liệu từ bảng. Dữ liệu trả về dạng DataTable
 - + mySQLLogger.TruyvanBaocao(string TenBang, string[] MangTenCot, string TenCotDinhVi, string GiaTriDau, string GiaTriCuoi)
 - + Đôi số:
 - TenBang: Tên bảng cần trích xuất
 - MangTenCot: Mảng tên các cột cần lấy dữ liệu. Nếu bảng {"*"} thì lấy dữ liệu ở tất cả các cột
 - TenCotDinhVi: Tên cột định vị để trích xuất
 - GiaTriDau: Giá trị đầu của cột định vị
 - GiaTriCuoi: Giá trị cuối của cột định vị
 - Khi GiaTriDau = GiaTriCuoi thì lấy giá trị ở một dòng.
 - + Trả về kết quả kiểu DataTable. Nếu dữ liệu trả về bằng null nghĩa là truy vấn thất bại.
 - + Ví dụ:

```
string[] mangtencot1 = { "*" };
```

```
DataTable bangnguong = myPLC.mySQLLogger.TruyvanBaocao("DataLog", mangtencot1,
"Id", "1","1");
```

4.3.9. SQLLogger:

Hàm	Mục đích
TaoKetNoi	Truyền chuỗi kết nối cho SQLLogger
TaoBang	Hàm dùng để tạo bảng table trong mysql
GhiDuLieuVaoBang	Dùng để ghi dữ liệu mới vào bảng
TruyvanBaocao	Truy vấn dữ liệu từ bảng
CapnhatDulieu	Cập nhật dữ liệu vào bảng

- **TaoKetNoi:** Truyền chuỗi kết nối cho Datalogger. Trước khi log dữ liệu cần truyền giá trị tạo kết nối trước
 - + SQLLogger. TaoKetNoi (string TenChuoiKetNoi, string TenDatabase)
 - + Đôi số:
 - TenChuoiKetNoi: có dạng "Server = servername;Uid = username;Pwd = password"

- TenDatabase: tên Database trong cơ sở dữ liệu
- + Ví dụ:
- ```
PLCPi myPLC = new PLCPi(); // tạo đối tượng myPLC
//truyền chuỗi kết nối đến cơ sở dữ liệu SQLServer muốn kết nối tới cho PLCPi
myPLC.SQLLogger.TaoKetNoi("Server =127.0.0.1;Uid=sa;Pwd=100100", "phatdb");
```
- **TaoBang:** Hàm dùng để tạo bảng table trong mysql. Nếu trùng tên bảng cũ thì bảng cũ sẽ bị xóa, thay bằng bảng mới. Nếu Database chưa tạo thì sẽ tạo mới database theo tên đã truyền vào khi tạo kết nối
    - + SQLLogger.TaoBang(string TenBang, string[] MangTenCot)
    - + Đôi số:
      - TenBang: Tên bảng muốn tạo
      - MangTenCot: Mảng chứa tên các cột muốn tạo trong bảng
    - + Trả về kết quả thực hiện kiểu string. Nếu dữ liệu trả về là “GOOD” nghĩa là đã tạo bảng thành công, “BAD” thất bại.
    - + Ví dụ:
 

```
String[] MangCot = {"DateTime", "Id", "Value"};
myPLC.SQLLogger.TaoBang("DataLog", MangCot);
```
  - **GhiDuLieuVaoBang:** Dùng để ghi dữ liệu mới vào bảng
    - + SQLLogger.GhiDuLieuVaoBang(string TenBang, string[] MangGiaTRi)
    - + Đôi số:
      - TenBang: Tên bảng cần truyền dữ liệu
      - MangGiaTRi: Mảng giá trị tương ứng với các cột trong bảng bắt đầu từ cột đầu tiên
    - + Trả về kết quả thực hiện kiểu string. Nếu dữ liệu trả về là “GOOD” nghĩa là ghi dữ liệu vào bảng thành công, “BAD” thất bại.
    - + Ví dụ:
 

```
string[] manggiatri = { myPLC.Datalogger.LayThoiGian(), "100", "200 "};
myPLC.SQLLogger.GhiDuLieuVaoBang("DataLog", manggiatri);
```
  - **TruyvanBaocao:** Dùng để truy vấn dữ liệu từ bảng. Dữ liệu trả về dạng DataTable
    - + SQLLogger.TruyvanBaocao(string TenBang, string[] MangTenCot, string TenCotDinhVi, string GiaTriDau, string GiaTriCuoi)
    - + Đôi số:
      - TenBang: Tên bảng cần trích xuất

- MangTenCot: Mảng tên các cột cần lấy dữ liệu. Nếu bằng {"\*"} thì lấy dữ liệu ở tất cả các cột
  - TenCotDinhVi: Tên cột định vị để trích xuất
  - GiaTriDau: Giá trị đầu của cột định vị
  - GiaTriCuoi: Giá trị cuối của cột định vị
  - Khi GiaTriDau = GiaTriCuoi thì lấy giá trị ở một dòng.
- + Trả về kết quả kiểu DataTable. Nếu dữ liệu trả về bằng null nghĩa là truy vấn thất bại.
- + Ví dụ:

```
string[] mangtencot1 = { "*" };
```

```
DataTable bangnguong = myPLC.SQLLogger.TruyvanBaocao("DataLog", mangtencot1,
"Id", "I", "I");
```

- **CapnhatDulieu:** cập nhật dữ liệu vào bảng

- + SQLLogger .CapnhatDulieu(string TenBang, string[] MangTenCotDinhVi, string[] MangGiaTriCotDinhVi, string[] MangTenCot, string[] MangGiaTri)
- + Đối số:
  - TenBang: Tên bảng cần trích xuất
  - MangTenCotDinhVi: mảng tên các cột dùng làm điều kiện
  - MangGiaTriCotDinhVi: mảng giá trị của các cột dùng làm điều kiện
  - MangTenCot: Mảng tên các cột cần lấy dữ liệu. Nếu bằng {"\*"} thì lấy dữ liệu ở tất cả các cột
  - MangGiaTri: mảng chứa giá trị của các cột cần update.
- + Trả về kết quả kiểu string. Nếu dữ liệu trả về bằng “GOOD” nghĩa là cập nhật dữ liệu thành công, bằng “BAD” thất bại.
- + Ví dụ:

*Cho bảng sau: tên bảng là “DataLog”*

| DateTime            | ID | Ten |
|---------------------|----|-----|
| 20-08-2015 07:08:00 | 1  | A   |
| 20-08-2015 07:08:00 | 2  | A   |
| 20-08-2015 07:08:00 | 3  | A   |

```

Cập nhật lại Ten = A thành Ten =B, tại DateTime =20-08-2015 07:08:00 và ID = 1
String[] mangcotdinhvi = { “DateTime”, “ID”};
String[] manggiatridinhvi = {“20-08-2015 07:08:00”, “I”};
string[] mangtencot = { “Ten” };
string[] manggiatri = {“B”};
myPLC.SQLLogger.CapnhatDulieu("DataLog",mangcotdinhvi,manggiatridinhvi,
mangtencot, manggiatri);

```

#### 4.3.10. SMS, Gọi điện thoại:

- Dùng để gửi tin nhắn SMS. Ứng dụng để gửi tin nhắn cảnh báo trong các dự án
- Port\_USB3G: thuộc tính lưu công Serial của USB 3G ta dùng để gửi SMS. Mặc định Port\_USB3G = “ttyUSB0”, nếu khi ta cắm usb 3G vào PLCPi mà nó nhận công Serial khác “ttyUSB0” thì ta phải thay đổi lại thuộc tính Port\_USB3G = “ Port mới”(ví dụ: “ttyUSB3”)
  - + Cách tìm cổng Serial của usb 3G:
    - Cắm usb 3G vào PLCPi, đợi 1 lúc để PLCPi nhận usb 3G
    - Ta mở phần mềm putty lên kết nối tới PLCPi, kết nối xong có giao diện như hình

```

192.168.1.10 - PuTTY
login as: root
root@192.168.1.10's password:
Linux raspberrypi 3.18.11-v7+ #781 SMP PREEMPT Tue Apr 21 18:07:59 BST 2015 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Aug 19 08:26:38 2015 from 192.168.1.112
root@raspberrypi:~#

```

- Ta nhập `sudo /dev/ttyUSB*` vào rồi enter sẽ xuất hiện giao diện như hình

```

root@raspberrypi:~# ls /dev/ttyUSB*
/dev/ttyUSB0 /dev/ttyUSB1 /dev/ttyUSB2
root@raspberrypi:~#

```

- ttyUSB0 chính là cổng serial của usb 3G. Cổng serial của usb 3G nó sẽ chiếm 3 port liền nhau. Trong trường hợp này nó từ ttyUSB0 đến ttyUSB2

- Trường hợp có 2 hoặc nhiều cổng serial usb 3G

```

LXTerminal
File Edit Tabs Help
root@raspberrypi: ~# ls /dev/ttyUSB*
/dev/ttyUSB0 /dev/ttyUSB2 /dev/ttyUSB4
/dev/ttyUSB1 /dev/ttyUSB3 /dev/ttyUSB5
root@raspberrypi: ~#

```

- \* Ở đây có 2 usb 3G cắm vào PLCPi. nên có 2 cổng serial. Bắt đầu từ ttyUSB0 và ttyUSB3. Khi ta gửi sms thông qua cổng ttyUSB0 thì ta ko cần phải thay đổi thuộc tính Port\_USB3G vì mặc định nó là “ttyUSB0”, nhưng khi ta gửi sms thông qua cổng ttyUSB3 thì ta phải thay đổi thuộc tính Port\_USB3G = “ttyUSB3” trước khi gọi method GuiSMS() để gửi tin nhắn
- GuiSMS:
  - + SMS.GuiSMS(string SDT, string Noidung):
  - + Đổi số:
    - SDT: số điện thoại cần gửi, kiểu chuỗi.Có thể là 1 số hoặc nhiều số, nếu nhiều số thì cách nhau bởi dấu “,”. Ví dụ: “0909123456”(1 số); “0909123456,0983456789,0919345678,...” (nhiều số)
    - Noidung: nội dung cần gửi
  - + Trả về trạng thái gửi kiểu string. Nếu chuỗi trả về là “GOOD” gửi thành công, khác “GOOD” gửi thất bại
  - + Ví dụ:
    - Gửi SMS đến 1 số điện thoại
      - PLCPi myPLC = new PLCPi(); //tạo đối tượng từ thư viện PLCPi*
      - myPLC.SMS.Khoitao();*
      - string Status = myPLCPi.SMS.GuiSMS("0909123456", "Test"); //gọi method gửi SMS*
      - if (Status == "GOOD")*
      - { Console.WriteLine("Gui thanh cong");}*
      - else*
      - {Console.WriteLine("Loi");}*

- Gửi SMS đến 3 số điện thoại
 

```

String SoDienThoai = "0909123456,0983456789,0913456789"; //số điện thoại

```

```

String NoiDung = "Sent SMS From PLCPi";//nội dung cần gửi
string Status = myPLCPi.SMS.GuiSMS(SoDienThoai, NoiDung);//gọi method gửi SMS
if (Status == "GOOD")
{
 Console.WriteLine("Gửi thành công");
}
else
{
 Console.WriteLine("Lỗi");
}

```

- Gọi điện:
  - + PhoneCall.GoiDienThoai(string SDT);
  - + Đổi số:
    - SDT: số điện thoại cần gọi, chỉ gọi được 1 số điện thoại
  - + Trả về trạng thái gửi kiểu string. Nếu chuỗi trả về là “GOOD” thành công, khác “GOOD” thất bại
  - + Ví dụ:
    - Gọi đến số điện thoại 0909123456
 

```

PLCPi myPLC = new PLCPi(); //tạo đối tượng từ thư viện PLCPi
myPLC.SMS.Khoitao();
string Status = myPLC.Pi.PhoneCall.GoiDienThoai ("0909123456");//gọi method gọi
điện thoại
if (Status == "GOOD")
{
 Console.WriteLine("Thanh cong");
}
else
{
 Console.WriteLine("Loi");
}

```

#### 4.3.11. Email:

- Dùng để gửi email. Để gửi email chúng ta cần:
  - + From: địa chỉ email gửi
  - + To: địa chỉ email nhận
  - + Subject: chủ đề
  - + Body: nội dung email
- **From:**
  - + Khai báo địa chỉ email dùng để gửi

- + Ví dụ:

```
//khai báo địa chỉ email để gửi là "EmailAddress@gmail.com"
myPLC.Email.CredentialEmail = "EmailAddress@gmail.com";
//mật khẩu email là "12345"
myPLC.Email.CredentialPass = "12345";
//tạo 1 đối tượng gửi email với địa chỉ là "EmailAddress@gmail.com"
myPLC.Email.Message.From = new
System.Net.Mail.MailAddress("EmailAddress@gmail.com")
```

- **To:**

- + Khai báo địa chỉ email nhận. Có thể gửi cho nhiều địa chỉ email khác nhau cùng 1 lúc, mỗi email cách nhau bởi dấu “,”
- + Ví dụ:

```
//khai báo địa chỉ email nhận là "EmailAddTo@gmail.com"
myPLC.Email.Message.To.Clear();
//1 địa chỉ email nhận
myPLC.Email.Message.To.Add("EmailAddTo@gmail.com");
//nhiều địa chỉ email nhận
myPLC.Email.Message.To.Add("EmailAddTo@gmail.com,EmailAddTo1@gmail.com,...");
```

- **Subject:**

- + Khai báo chủ đề của email
- + Ví dụ:

```
//gửi email với chủ đề là "test"
myPLC.Email.Message.Subject = "test";
```

- **Body:**

- + Nhập nội dung của email
- + Ví dụ:

```
//gửi email với nội dung là "gửi email từ PLCPi"
```

```
myPLC.Email.Message.Body = “gửi email từ PLCPi”;
```

- Xong phần soạn email để gửi, bây giờ tiến hành gửi
  - + myPLC.Email.SendEmail(): method gửi email
  - + khi gửi email, có trả về trạng thái gửi kiểu boolean. Nếu = true nghĩa là gửi không thành công
  - + ví dụ:

```
myPLC.Email.TimeOut = 2000;//cài đặt timeout
```

```
myPLC.Email.SendEmail();//gửi email
```

```
//kiểm tra xem gửi đã thành công hay chưa
```

```
if(myPLC.Email.Error == false)
```

```
Console.WriteLine("Gửi thành công");
```

```
else
```

```
Console.WriteLine("Gửi thất bại");
```

#### 4.3.12. Đọc ghi dữ liệu từ mảng byte:

- Gồm 2 chức năng:
  - + Đọc dữ liệu từ mảng byte: đọc giá trị theo kiểu dữ liệu ta cần từ mảng chứa các phần tử kiểu byte, dùng để chuyển đổi các byte trong mảng đó về kiểu dữ liệu ta mong muốn
  - + Ghi giá trị vào mảng byte: ghi giá trị theo kiểu dữ liệu ta cần, xuống các phần tử trong mảng kiểu byte, dùng để ghi xuống mảng dữ liệu kiểu byte các giá trị có kiểu dữ liệu khác kiểu byte.
- Các hàm chức năng

| Hàm            | Mục đích                                                                       |
|----------------|--------------------------------------------------------------------------------|
| GetBoolAt      | Đọc về dữ liệu kiểu Boolean. 1 bit                                             |
| GetUshortAt    | Đọc về dữ liệu kiểu Word . Số nguyên 2 byte không dấu                          |
| GetShortAt     | Đọc về dữ liệu kiểu short. Số nguyên 2 byte có dấu                             |
| GetUIntAt      | Đọc về dữ liệu kiểu DWord. Số nguyên 4 byte không dấu, dạng “first word hight” |
| GetUInt_LSB_At | Đọc về dữ liệu kiểu DWord. Số nguyên 4 byte không dấu, dạng “first word low”   |
| GetIntAt       | Đọc về dữ liệu kiểu int. Số nguyên 4 byte có dấu, dạng “first word             |

|                   |                                                                               |
|-------------------|-------------------------------------------------------------------------------|
|                   | hight”                                                                        |
| GetIntAt_LSB_At   | Đọc về dữ liệu kiểu int. Số nguyên 4 byte có dấu, dạng “first word low”       |
| GetFloatAt        | Đọc về dữ liệu kiểu float. Kiểu dấu chấm động 4 byte, dạng “first word hight” |
| GetFloatAt_LSB_At | Đọc về dữ liệu kiểu float. Kiểu dấu chấm động 4 byte, dạng “first word low”   |
| SetBit            | Ghi giá trị kiểu bit xuống mảng byte                                          |
| SetWord           | Ghi giá trị ushort vào mảng Byte                                              |
| .SetInt           | Ghi giá trị short vào mảng Byte                                               |
| SetDWord          | Ghi giá trị uint vào mảng Byte, dạng “first word hight”                       |
| SetDWord_LSB      | Ghi giá trị uint vào mảng Byte, dạng “first word low”                         |
| SetDint           | Ghi giá trị int vào mảng Byte, dạng “first word hight”                        |
| SetDint_LSB       | Ghi giá trị int vào mảng Byte, dạng “first word low”                          |
| SetFloat          | Ghi giá trị float vào mảng Byte, dạng “first word hight”                      |
| SetFloat_LSB      | Ghi giá trị float vào mảng Byte, dạng “first word low”                        |

- **GetBoolAt:**

+ PLCPi.GetBoolAt(byte[] buffer, int pos, int bit)

+ Đôi số:

- buffer: mảng dữ liệu kiểu byte, truyền vào để chuyển đổi
- pos: vị trí của byte muốn đọc
- bit: bit muốn đọc

+ Trả về 1 bit tại pos.bit

+ Ví dụ: đọc về bít thứ 7 của byte 0 trong mảng sau

```
PLCPi myPLCPi = new PLCPi(); // tạo 1 đối tượng từ lớp PLCPi
byte[] a = {128,1,2,3,4,0}
bool bit7 = myPLC.GetBoolAt(a, 0, 7);
//method GetBoolAt sẽ lấy byte a[0] rồi tách bít thứ 7 chuyển đổi về kiểu bool rồi trả về
```

- **GetUshortAt: số nguyên 16 bit không dấu**

+ PLCPi.GetUshortAt (byte[] buffer, int pos)

+ Đôi số:

- buffer: mảng dữ liệu kiểu byte, truyền vào để chuyển đổi

- pos: vị trí của byte bắt đầu
- + Trả về dữ liệu 16 bit không dấu (S7 Word) 0x0000..0xFFFF
- + Ví dụ: đọc về 1 Word trong mảng sau byte[] a = {128,1,2,3,4,0}, bắt đầu từ a[0]
 

```
ushort Data = myPLC.GetUshortAt(a, 0);
//method GetUshortAt sẽ lấy 2 byte là a[0] và a[1] chuyển đổi về giá trị ushort rồi trả về.
khi đó a[0] là byte cao, a[1] là byte thấp.
```
- **GetShortAt: số nguyên 16 bit có dấu**
  - + PLCPi.GetShortAt(byte[] buffer, int pos)
  - + Đôi số:
    - buffer: mảng dữ liệu kiểu byte, truyền vào để chuyển đổi
    - pos: vị trí của byte bắt đầu
  - + Trả về dữ liệu 16 bit có dấu (S7 int) -32768..32767
  - + Ví dụ: đọc về 1 int trong mảng sau byte[] a = {128,1,2,3,4,0}, bắt đầu từ a[0]
 

```
short Data = myPLC.GetShortAt(a, 0);
//method GetShortAt sẽ lấy 2 byte là a[0] và a[1] chuyển đổi về giá trị short rồi trả về.
khi đó a[0] là byte cao, a[1] là byte thấp.
```
- **GetUIntAt: số nguyên 32 bit không dấu, first word hight**
  - + PLCPi.Get UIntAt(byte[] buffer, int pos)
  - + Đôi số:
    - buffer: mảng dữ liệu kiểu byte, truyền vào để chuyển đổi
    - pos: vị trí của byte bắt đầu
  - + Trả về dữ liệu 32 bit không dấu (S7 DWord) 0x00000000..0xFFFFFFFF
  - + Ví dụ: đọc về 1 DWord trong mảng sau byte[] a = {128,1,2,3,4,0}, bắt đầu từ a[0]
 

```
uint Data = myPLC.GetUIntAt(a, 0);
//method GetUIntAt sẽ lấy 4 byte là a[0], a[1], a[2], a[3] chuyển đổi về giá trị uint rồi trả về.
khi đó a[0]a[1] là word cao.
```
- **GetUInt\_LSB\_At: số nguyên 32 bit không dấu, first word low**
  - + PLCPi.Get UIntAt\_LSB\_At(byte[] buffer, int pos)
  - + Đôi số:
    - buffer: mảng dữ liệu kiểu byte, truyền vào để chuyển đổi
    - pos: vị trí của byte bắt đầu
  - + Trả về dữ liệu 32 bit không dấu (S7 DWord) 0x00000000..0xFFFFFFFF

- + Ví dụ: đọc về 1 DWord trong mảng sau byte[] a = {128,1,2,3,4,0}, bắt đầu từ a[0]

```
uint Data = myPLC.GetUInt_LSB_At(a, 0);
```

//method GetUInt\_LSB\_At sẽ lấy 4 byte là a[0], a[1], a[2], a[3] chuyển đổi về giá trị uint rồi trả về. khi đó a[0]a[1] là word thấp.

- **GetIntAt: số nguyên 32 bit có dấu, first word hight**

- PLCPi.GetIntAt(byte[] buffer, int pos)
- Đối số:
  - \* buffer: mảng dữ liệu kiểu byte, truyền vào để chuyển đổi
  - \* pos: vị trí của byte bắt đầu
- Trả về dữ liệu 32 bit có dấu (S7 Dint) -2147483648..2147483647
- Ví dụ: đọc về 1 Dint trong mảng sau byte[] a = {128,1,2,3,4,0}, bắt đầu từ a[0]

```
int Data = myPLC.GetIntAt(a, 0);
```

//method GetIntAt sẽ lấy 4 byte là a[0], a[1], a[2], a[3] chuyển đổi về giá trị int rồi trả về. khi đó a[0]a[1] là word cao.

- **GetInt\_LSB\_At: số nguyên 32 bit có dấu, first word low**

- + PLCPi.GetInt\_LSB\_At(byte[] buffer, int pos)
- + Đối số:
  - buffer: mảng dữ liệu kiểu byte, truyền vào để chuyển đổi
  - pos: vị trí của byte bắt đầu
- + Trả về dữ liệu 32 bit có dấu (S7 Dint) -2147483648..2147483647
- + Ví dụ: đọc về 1 Dint trong mảng sau byte[] a = {128,1,2,3,4,0}, bắt đầu từ a[0]

```
uint Data = myPLC.GetInt_LSB_At(a, 0);
```

//method GetInt\_LSB\_At sẽ lấy 4 byte là a[0], a[1], a[2], a[3] chuyển đổi về giá trị uint rồi trả về. khi đó a[0]a[1] là word thấp.

- **GetFloatAt: dấu chấm động 32 bit, first word hight**

- PLCPi.GetFloatAt(byte[] buffer, int pos)
- Đối số:
  - \* buffer: mảng dữ liệu kiểu byte, truyền vào để chuyển đổi
  - \* pos: vị trí của byte bắt đầu
- Trả về dữ liệu 32 bit dấu chấm động (S7 float)
- Ví dụ: đọc về 1 float trong mảng sau byte[] a = {128,1,2,3,4,0}, bắt đầu từ a[0]

```
float Data = myPLC.GetFloatAt(a, 0);
```

//method GetFloatAt sẽ lấy 4 byte là a[0], a[1], a[2], a[3] chuyển đổi về giá trị float rồi trả về. khi đó a[0]a[1] là word cao.

- **GetFloat\_LSB\_At: dấu chấm động 32 bit, first word low**

+ PLCPi.GetFloat\_LSB\_At(byte[] buffer, int pos)

+ Đôi số:

- buffer: mảng dữ liệu kiểu byte, truyền vào để chuyển đổi
- pos: vị trí của byte bắt đầu

+ Trả về dữ liệu 32 bit dấu chấm động (S7 float)

+ Ví dụ: đọc về 1 float trong mảng sau byte[] a = {128,1,2,3,4,0}, bắt đầu từ a[0]

*float Data = myPLC.GetFloat\_LSB\_At(a, 0);*

*// method GetFloat\_LSB\_At sẽ lấy 4 byte là a[0], a[1], a[2], a[3] chuyển đổi về giá trị float rồi trả về. khi đó a[0]a[1] là word thấp.*

- **SetBit: ghi dữ liệu kiểu boolean vào mảng kiểu byte**

+ PLCPi.SetBit(byte[] Buffer, int Pos, int Bit, int Value)

+ Đôi số:

- buffer: mảng dữ liệu kiểu byte, truyền vào để ghi dữ liệu vào các byte trong mảng này
- pos: vị trí của byte muốn ghi
- bit: bit muốn ghi
- Value : giá trị ghi xuống bit, 0 hoặc 1

+ ví dụ: set bit 0 của byte đầu tiên trong mảng sau lên 1, byte[] a = {0,0,0,0,0,0}

*myPLC.SetBit(a, 0, 0, 1);*

*//ghi giá trị 1 vào bit 0 của byte a[0]*

- **SetWord: ghi dữ liệu kiểu số nguyên 16bit không dấu vào mảng kiểu byte**

+ PLCPi. SetWord(byte[] Buffer, int Pos, ushort Value)

+ Đôi số:

- \* buffer: mảng dữ liệu kiểu byte, truyền vào để ghi dữ liệu vào các byte trong mảng này
- \* pos: vị trí của byte muốn ghi
- \* Value : giá trị ghi xuống, dữ liệu kiểu ushort

+ Ví dụ: ghi giá trị kiểu ushort xuống mảng a[], bắt đầu từ byte 0

*byte[] a = {0,0,0,0,0,0}*

*ushort Value = 65535;*

*myPLC.SetWord(a, 0, Value);*

//Khi đó SetWord sẽ chuyển đổi Value thành 2 byte, byte cao lưu vào a[0], byte thấp lưu vào a[1]

- **SetInt: ghi dữ liệu kiểu số nguyên 16bit có dấu vào mảng kiểu byte**

+ PLCPi.SetInt(byte[] Buffer, int Pos, short Value)

+ Đôi số:

- buffer: mảng dữ liệu kiểu byte, truyền vào để ghi dữ liệu vào các byte trong mảng này
- pos: vị trí của byte muốn ghi
- Value : giá trị ghi xuống, dữ liệu kiểu short

+ Ví dụ: ghi giá trị kiểu short xuống mảng a[], bắt đầu từ byte 0

byte[] a = {0,0,0,0,0,0}

short Value = -1000;

myPLC.SetInt(a, 0, Value);

//Khi đó SetInt sẽ chuyển đổi Value thành 2 byte, byte cao lưu vào a[0], byte thấp lưu vào a[1]

- **SetDWord: ghi dữ liệu kiểu số nguyên 32bit không dấu vào mảng kiểu byte, first word hight**

+ PLCPi.SetDWord(byte[] Buffer, int Pos, uint Value)

+ Đôi số:

- buffer: mảng dữ liệu kiểu byte, truyền vào để ghi dữ liệu vào các byte trong mảng này
- pos: vị trí của byte muốn ghi
- Value : giá trị ghi xuống, dữ liệu kiểu uint

+ Ví dụ: ghi giá trị kiểu uint xuống mảng a[], bắt đầu từ byte 0

byte[] a = {0,0,0,0,0,0}

uint Value = 4294967296;

myPLC.SetDWord(a, 0, Value);

//Khi đó SetDWord sẽ chuyển đổi Value thành 4 byte, byte cao nhất lưu vào a[0], byte thấp nhất lưu vào a[3]

- **SetDWord\_LSB: ghi dữ liệu kiểu số nguyên 32bit không dấu vào mảng kiểu byte, first word low**

+ PLCPi.SetDWord\_LSB(byte[] Buffer, int Pos, uint Value)

+ Đôi số:

- buffer: mảng dữ liệu kiểu byte, truyền vào để ghi dữ liệu vào các byte trong mảng này
- pos: vị trí của byte muốn ghi

- Value : giá trị ghi xuống, dữ liệu kiểu uint
- + Ví dụ: ghi giá trị kiểu uint xuống mảng a[], bắt đầu từ byte 0
 

```
byte[] a = {0,0,0,0,0,0}
uint Value = 4294967296;
myPLC.SetDWord_LSB(a, 0, Value);
//Khi đó SetDWord_LSB sẽ chuyển đổi Value thành 4 byte, byte cao nhất lưu vào a[2], byte thấp nhất lưu vào a[1]
```
- **SetDint: ghi dữ liệu kiểu số nguyên 32bit có dấu vào mảng kiểu byte, first word hight**
  - + PLCPi.SetDint(byte[] Buffer, int Pos, int Value)
  - + Đôi số:
    - buffer: mảng dữ liệu kiểu byte, truyền vào để ghi dữ liệu vào các byte trong mảng này
    - pos: vị trí của byte muốn ghi
    - Value : giá trị ghi xuống, dữ liệu kiểu int
  - + Ví dụ: ghi giá trị kiểu int xuống mảng a[], bắt đầu từ byte 0
 

```
byte[] a = {0,0,0,0,0,0}
int Value = -1000000;
myPLC.SetDint(a, 0, Value);
//Khi đó SetDint sẽ chuyển đổi Value thành 4 byte, byte cao nhất lưu vào a[0], byte thấp nhất lưu vào a[3]
```
- **SetDint\_LSB: ghi dữ liệu kiểu số nguyên 32bit có dấu vào mảng kiểu byte, first word low**
  - + PLCPi.SetDint\_LSB(byte[] Buffer, int Pos, int Value)
  - + Đôi số:
    - buffer: mảng dữ liệu kiểu byte, truyền vào để ghi dữ liệu vào các byte trong mảng này
    - pos: vị trí của byte muốn ghi
    - Value : giá trị ghi xuống, dữ liệu kiểu int
  - + Ví dụ: ghi giá trị kiểu int xuống mảng a[], bắt đầu từ byte 0
 

```
byte[] a = {0,0,0,0,0,0}
int Value = -1000000;
myPLC.SetDint_LSB(a, 0, Value);
//Khi đó SetDint_LSB sẽ chuyển đổi Value thành 4 byte, byte cao nhất lưu vào a[2], byte thấp nhất lưu vào a[1]
```
- **SetFloat: ghi dữ liệu kiểu dấu chấm động 32bit vào mảng kiểu byte, first word hight**
  - + S7Ethernet.SetFloat(byte[] Buffer, int Pos, int Value)

- + Đôi số:
  - buffer: mảng dữ liệu kiểu byte, truyền vào để ghi dữ liệu vào các byte trong mảng này
  - pos: vị trí của byte muốn ghi
  - Value : giá trị ghi xuống, dữ liệu kiểu float
- + Ví dụ: ghi giá trị kiểu float xuống mảng a[], bắt đầu từ byte 0
 

```
byte[] a = {0,0,0,0,0,0}
float Value = 100.2;
myPLC.SetFloat(a, 0, Value);
//Khi đó SetFloat sẽ chuyển đổi Value thành 4 byte, byte cao nhất lưu vào a[0], byte thấp nhất lưu vào a[3]
```
- **SetFloat\_LSB:** ghi dữ liệu kiểu dấu chấm động 32bit vào mảng kiểu byte, first word low
  - + PLCPi.SetFloat\_LSB(byte[] Buffer, int Pos, int Value)
  - + Đôi số:
    - buffer: mảng dữ liệu kiểu byte, truyền vào để ghi dữ liệu vào các byte trong mảng này
    - pos: vị trí của byte muốn ghi
    - Value : giá trị ghi xuống, dữ liệu kiểu float
  - + Ví dụ: ghi giá trị kiểu float xuống mảng a[], bắt đầu từ byte 0
 

```
byte[] a = {0,0,0,0,0,0}
float Value = 100.2;
myPLC.SetFloat_LSB(a, 0, Value);
//Khi đó SetFloat_LSB sẽ chuyển đổi Value thành 4 byte, byte cao nhất lưu vào a[2], byte thấp nhất lưu vào a[1]
```

#### 4.3.13. S7Ethernet:

- Có 2 chức năng:
  - + Truyền thông S7 Ethernet Server
  - + Truyền thông S7 Ethernet Client
- Các hàm chức năng

| Hàm    | Mục đích                                                       |
|--------|----------------------------------------------------------------|
| Server | Tạo PLCPi thành 1 Server để các S7 Ethernet Client kết nối đến |
| Client | Tạo PLCPi thành 1 Client để kết nối đến các S7 Ethernet Server |

- **Server:**

- + S7 Ethernet Server trên PLCPi có 3 vùng nhớ
  - MangNgoVao: gồm 5 byte, chứa giá trị của 5 byte ngõ vào, từ I0 đến I4, vùng nhớ chỉ đọc
  - MangNgoRa: gồm 6 byte, chứa giá trị của 6 byte ngõ ra, từ Q0 đến Q5, vùng nhớ đọc ghi
  - DataBlock: vùng nhớ data block, gồm 1024 byte để lưu trữ dữ liệu, vùng nhớ đọc ghi
- + Khoitao : Khởi tạo server
  - **S7Ethernet.Server.Khoitao ()**
  - Trả về trạng thái kết nối kiểu chuỗi. Nếu dữ liệu trả về là “GOOD” khởi tạo thành công, khác “GOOD” khởi tạo lỗi.
  - Ví dụ:

```

PLCPi myPLC = new PLCPi(); //tạo đối tượng myPLC
string TrangThai = myPLC.S7Ethernet.Server.Khoitao();
if(TrangThai == "GOOD") //khởi tạo thành công
{
 while(true){....}
}
else //khởi tạo thất bại
{
 Console.WriteLine("loi");
}

```

- + Truy cập đến các vùng nhớ của server:
  - Vùng nhớ ngõ vào: S7Ethernet.Server.NgoVao.MangNgoVao[]
  - Vùng nhớ ngõ ra: S7Ethernet.Server.NgoVao.MangNgoRa[]
  - Vùng nhớ DataBlock: S7Ethernet.Server. DataBlock[]
  - Ví dụ: ghi giá trị 255 vào vùng nhớ DataBlock[0]
 

```
myPLC.S7Ethernet.Server. DataBlock[0] = 255;
```

- **Client:**

- + Các hàm chức năng

| Hàm        | Mục đích                                                                 |
|------------|--------------------------------------------------------------------------|
| KetNoi     | Khởi tạo PLCPi thành S7Ethernet Client                                   |
| NgatKetNoi | Ngừng S7 ethernet client trên PLCPi, ngắt kết nối đến S7 ethernet server |
| DocIB      | Đọc vùng nhớ ngõ vào                                                     |

|       |                                    |
|-------|------------------------------------|
| DocQB | Đọc vùng nhớ ngắn ra               |
| DocDB | Đọc vùng nhớ data block            |
| DocMB | Đọc vùng nhớ memory block          |
| GhiQB | Ghi dữ liệu xuống vùng nhớ ngắn ra |
| GhiDB | Ghi dữ liệu xuống vùng nhớ DB      |
| GhiMB | Ghi dữ liệu xuống vùng nhớ MB      |

- Đối với các method DocIB, DocQB, DocDB, DocMB, khi ta gọi tới các method này thì nó sẽ trả về 1 đối tượng DocS7 gồm 2 thuộc tính
  - \* DocS7.MangGiaTri: là mảng kiểu byte, chứa các giá trị đọc được từ các vùng nhớ của S7Ethernet Server. Có số phần tử bằng với số byte ta đọc lên khi gọi các method đọc
  - \* DocS7.TrangThai: là 1 biến kiểu string, cho ta biết việc đọc thành công hay không, nếu TrangThai = “GOOD” thành công, TrangThai != “GOOD” thất bại
- + KetNoi:
  - S7Ethernet.Client.KetNoi(string ip)
  - Đối số:
    - \* ip: là địa chỉ ip của thiết bị S7 ethernet server cần kết nối tới, có kiểu dữ liệu là kiểu chuỗi
  - Trả về trạng thái kết nối kiểu chuỗi. Nếu dữ liệu trả về là = “GOOD” kết nối thành công, khác “GOOD” kết nối lỗi
  - Ví dụ:
 

```
PLCPi myPLC = new PLCPi(); //tạo đối tượng myPLC
string TrangThai = myPLC.S7Ethernet.Client.KetNoi("192.168.1.100");
if(TrangThai == "GOOD") //kết nối thành công
{
 while(true){....}
}
else //kết nối thất bại
{
 Console.WriteLine("loi");
}
```
- + NgatKetNoi:

- S7Ethernet.Client.NgatKetNoi()
- Trả về trạng thái ngắt kết nối kiểu chuỗi, Nếu dữ liệu trả về là = “GOOD” ngắt kết nối thành công, khác “GOOD” ngắt kết nối lỗi.
- Ví dụ:

```

string TrangThai = myPLC.S7Ethernet.Client.NgatKetNoi()
if(TrangThai == "GOOD") //ngắt kết nối thành công
{
 while(true){....}
}
else //ngắt kết nối thất bại
{
 Console.WriteLine("loi");
}

```

+ DocIB: đọc vùng nhớ ngõ vào

- S7Ethernet.Client.DocIB(int BatDau, int SoByte)
- Đối số:
  - \* BatDau: vị trí bắt đầu đọc
  - \* SoByte: số byte muốn đọc về tính từ vị trí BatDau
- Trả về đối tượng DocS7
- Ví dụ: đọc về 2 byte ngõ vào tính từ byte 0

```

//Ta phải khai báo 1đối tượng để nhận đối tượng trả về
DocS7 DocNgoVao = myPLC.S7Ethernet.Client.DocIB(0, 2);
//Khi đó, để lấy các giá trị đọc về hay trạng thái thì ta làm như sau:
String NgoVao_0 = DocNgoVao.MangGiaTri[0];
String NgoVao_1 = DocNgoVao.MangGiaTri[1];
String TrangThaiDoc = DocNgoVao.TrangThai;

```

+ DocQB: đọc vùng nhớ ngõ ra

- S7Ethernet.Client.DocQB(int BatDau, int SoByte)
- Đối số:
  - \* BatDau: vị trí bắt đầu đọc
  - \* SoByte: số byte muốn đọc về tính từ vị trí BatDau
- Trả về đối tượng DocS7

- Ví dụ: đọc về 5 byte ngõ ra tính từ byte 0

//Ta phải khai báo 1đối tượng để nhận đối tượng trả về  
`DocS7 DocNgoRa = myPLC.S7Ethernet.Client.DocQB(0,5);`  
//Khi đó, để lấy các giá trị đọc về hay trạng thái thì ta làm như sau:  
`String NgoRa_0 = DocNgoRa.MangGiaTri[0];`  
`String NgoRa_4 = DocNgoRa.MangGiaTri[4];`  
`String TrangThaiDoc = DocNgoRa.TrangThai;`

+ DocDB: đọc vùng nhớ data block

- `S7Ethernet.Client.DocDB(int DB_Num, int BatDau, int SoByte)`
- Đối số:
  - \* `DB_Num`: vị trí của vùng nhớ DB, bắt đầu từ 1( DB1, DB2, DB3, ...)
  - \* `BatDau`: vị trí bắt đầu đọc
  - \* `SoByte`: số byte muốn đọc về tính từ vị trí BatDau
- Trả về đối tượng DocS7
- Ví dụ: đọc về 5 byte từ vùng nhớ DB1 tính từ byte 0

//Ta phải khai báo 1đối tượng để nhận đối tượng trả về  
`DocS7 DocDB = myPLC.S7Ethernet.Client.DocDB(1,0, 5);`  
//Khi đó, để lấy các giá trị đọc về hay trạng thái thì ta làm như sau:  
`String DocDB _0 = DocDB.MangGiaTri[0];`  
`String DocDB _4 = DocDB.MangGiaTri[4];`  
`String TrangThaiDoc = DocDB.TrangThai;`

+ DocMB: đọc vùng nhớ MB

- `S7Ethernet.Client.DocMB(int BatDau, int SoByte)`
- Đối số:
  - \* `BatDau`: vị trí bắt đầu đọc
  - \* `SoByte`: số byte muốn đọc về tính từ vị trí BatDau
- Trả về đối tượng DocS7
- Ví dụ: đọc về 10 byte từ vùng nhớ MB tính từ byte 0

//Ta phải khai báo 1đối tượng để nhận đối tượng trả về  
`DocS7 DocMB = myPLC.S7Ethernet.Client.DocMB (0, 10);`  
//Khi đó, để lấy các giá trị đọc về hay trạng thái thì ta làm như sau:  
`String DocMB _0 = DocMB.MangGiaTri[0];`

- ```

String DocMB _9 = DocMB.MangGiaTri[9];
String TrangThaiDoc = DocMB.TrangThai;

```
- + GhiQB:
- S7Ethernet.Client.GhiQB(int BatDau, int SoByte, byte[] Value)
 - Đối số:
 - * BatDau: vị trí bắt đầu ghi
 - * SoByte: số byte muốn ghi xuống tính từ vị trí BatDau
 - * Value: mảng chứa dữ liệu để ghi xuống vùng nhớ, số phần tử của mảng Value >= SoByte
 - Trả về kết quả thực hiện kiểu chuỗi, nếu dữ liệu trả về = “GOOD” là ghi thành công, ngược lại là ghi không thành công
 - Ví dụ: ghi mảng {1,2,3} xuống vùng nhớ ngõ ra, bắt đầu từ byte 0, ghi xuống 3 byte


```

byte[] data = {1,2,3};
string TrangThaiGhi = myPLC.S7Ethernet.Client.GhiQB(0, 3, data);
if(TrangThaiGhi == "GOOD")
{
    Console.WriteLine("Ghi hoan thanh");
}
else {Console.WriteLine("loi");}
      
```
- + GhiDB:
- S7Ethernet.Client.GhiDB(int DB_Num, int BatDau, int SoByte, byte[] Value)
 - Đối số:
 - * DB_Num: vị trí của vùng nhớ DB, bắt đầu từ 1(DB1, DB2, DB3, ...)
 - * BatDau: vị trí bắt đầu ghi
 - * SoByte: số byte muốn ghi xuống tính từ vị trí BatDau
 - * Value: mảng chứa dữ liệu để ghi xuống vùng nhớ, số phần tử của mảng Value >= SoByte
 - Trả về kết quả thực hiện kiểu chuỗi, nếu dữ liệu trả về = “GOOD” là ghi thành công, ngược lại là ghi không thành công
 - Ví dụ: ghi mảng {1,2,3} xuống vùng nhớ DB1, bắt đầu từ byte 0, ghi xuống 3 byte


```

byte[] data = {1,2,3};
myPLC.S7Ethernet.Client.GhiDB(1, 0, 3, data);
      
```
- + GhiMB:
- S7Ethernet.Client.GhiMB(int BatDau, int SoByte, byte[] Value)
 - Đối số:

- * BatDau: vị trí bắt đầu ghi
- * SoByte: số byte muốn ghi xuống tính từ vị trí BatDau
- * Value: mảng chứa dữ liệu để ghi xuống vùng nhớ, số phần tử của mảng Value >= SoByte
- Trả về kết quả thực hiện kiểu chuỗi, nếu dữ liệu trả về = “GOOD” là ghi thành công, ngược lại là ghi không thành công
- Ví dụ: ghi mảng {1,2,3} xuống vùng nhớ MB, bắt đầu từ byte 0, ghi xuống 3 byte
`byte[] data = {1,2,3};`
`myPLC.S7Ethernet.Client.GhiMB(0, 3, data);`

4.3.14. ModbusTCPClient:

- Dùng để truyền thông theo chuẩn ModbusTCP IP
- Các hàm chức năng :

Hàm	Mục đích
KetNoi	Tạo kết nối tới thiết bị đang chạy modbus TCP server
connected	Trạng thái kết nối đến server
NgatKetNoi	ngắt kết nối đến modbus TCP server
ReadCoils	Đọc vùng nhớ coil
ReadDiscreteInputs	Đọc vùng nhớ discrete inputs
ReadHoldingRegister	Đọc thanh ghi Holding
ReadInputRegister	Đọc thanh ghi ngõ vào
WriteSingleCoil	Ghi dữ liệu xuống 1 coil
WriteMultipleCoils	Ghi dữ liệu xuống nhiều coil một lúc
WriteSingleRegister	Ghi dữ liệu xuống 1 thanh ghi Holding
WriteMultipleRegister	Ghi dữ liệu xuống nhiều thanh ghi Holding một lúc

- **KetNoi:**
 - + ModbusTCPClient.KetNoi(string ip, ushort port)
 - + Đối số:
 - ip: địa chỉ IP của server
 - port: cổng modbus TCP của server
 - + Ví dụ:

```

PLCPi myPLCPi = new PLCPi(); // tạo đối tượng myPLCPi
myPLCPi.ModbusTCPClient.KetNoi("192.168.1.100", 502);
// kết nối đến server có địa chỉ IP là 192.168.1.100, và port 502

```

- **connected:** trả về trạng thái kết nối đến server

- + ModbusTCPClient. connected ()

- + Trả về dữ liệu kiểu bool, nếu bằng true kết nối đến server thành công, bằng false thất bại.

- + Ví dụ:

```

myPLCPi.ModbusTCPClient.KetNoi("192.168.1.100", 502);
if(myPLCPi.ModbusTCPClient. connected () == true)
{
    Console.WriteLine("thanh cong");
}
Else
{
    Console.WriteLine("that bai");
}

```

- **NgatKetNoi:** ngắt kết nối đến server

- + ModbusTCPClient.NgatKetNoi ()

- + Ví dụ:

```
myPLCPi.ModbusTCPClient. NgatKetNoi();
```

- **ReadCoils:** đọc coil (địa chỉ 1-9999)

- + ModbusTCPClient.ReadCoils (ushort id, ushort startAddress, ushort numInputs, ref byte[] values)

- + Đối số:

- id: id của server, mặc định là 0
- startAddress: địa chỉ của coil bắt đầu đọc
- numInputs: số coil muốn đọc, tính từ startAddress
- values: mảng để lưu giá trị của các coil đọc về, nếu mảng này bằng null nghĩa là đọc không thành công

- + Ví dụ: đọc về 10 coil từ coil0 đến coil9

```

Byte[] Data = new byte[2];// khai báo mảng để chứa các giá trị đọc về
myPLCPi.ModbusTCPClient.ReadCoils(0,0,10,ref Data);//đọc 10 coil
//vùng nhớ coil chỉ có 2 giá trị là true hoặc false. Khi đọc về thì sẽ ghi giá trị 0 hoặc 1 vào
//các bit của byte. Ở đây, đọc 10 coil thì mình dùng 2 byte là đủ để chứa hết trạng thái của
//10 coil. Khi đó, trạng thái của 8 coil đầu sẽ được ghi vào Data[0], trạng thái 2 coil còn lại
//được ghi vào Data[1]. Bit 0 của byte sẽ lưu trạng thái của coil có địa chỉ nhỏ nhất. Bit 0
//của Data[0] lưu trạng thái coil0, bit 1 lưu coil1,... bit 7 lưu coil7. Bit 0 Data[1] lưu coil8,
//Bit 1 lưu coil9
//ghi trạng thái coil 7 ra màn hình
//dùng method ở mục 4.3.12 để lấy ra kiểu dữ liệu mong muốn từ mảng byte
bool coil7 = myPLC.GetBoolAt(Data, 0, 7); //lấy bit 7 của byte Data[0]
Console.WriteLine(coil7);//in ra màn hình

```

- **ReadDiscreteInputs:** địa chỉ (10001-19999)

- + ModbusTCPClient. ReadDiscreteInputs (ushort id, ushort startAddress, ushort numInputs, ref byte[] values)
- + Đôi số:
 - id: id của server, mặc định là 0
 - startAddress: địa chỉ của DiscreteInputs bắt đầu đọc
 - numInputs: số DiscreteInputs muốn đọc, tính từ startAddress
 - values: mảng để lưu giá trị của các DiscreteInputs đọc về, nếu mảng này bằng null nghĩa là đọc không thành công
- + Ví dụ: đọc 10 ngõ vào rác, từ 0 đến 9

```

Byte[] Data = new byte[2];// khai báo mảng để chứa các giá trị đọc về
myPLCPi.ModbusTCPClient.ReadDiscreteInputs(0,0,10,ref Data);//đọc 10 DiscreteInputs
//vùng nhớ ngõ vào rác chỉ có 2 giá trị là true hoặc false. Nên cách lấy giá trị như đọc
//coil
//in ra màn hình giá trị DiscreteInputs0
Console.WriteLine("DiscreteInputs0 = {0}", myPLC.GetBoolAt(Data, 0, 0));

```

- **ReadHoldingRegister:** đọc thanh ghi Holding (địa chỉ 40001-49999)

- + ModbusTCPClient. ReadHoldingRegister (ushort id, ushort startAddress, ushort numInputs, ref byte[] values)

- + Đôi số:
 - id: id của server, mặc định là 0
 - startAddress: địa chỉ HoldingRegister bắt đầu đọc
 - numInputs: số HoldingRegister muốn đọc, tính từ startAddress
 - values: mảng để lưu giá trị của các HoldingRegister đọc về, nếu mảng này bằng null nghĩa là đọc không thành công
- + Ví dụ: đọc 1 thanh ghi Holding có địa chỉ là 40001

```

Byte[] Data = new byte[2];// khai báo mảng để chứa các giá trị đọc về
myPLCPi.ModbusTCPClient.ReadHoldingRegister(0,0,1,ref Data);//đọc 1 HoldingRegister
//vùng HoldingRegister có kiểu dữ liệu là word, nên cần 2 byte để lưu giá trị của 1 thanh ghi
//Holding. Mảng Data gồm 2 phần tử để lưu giá trị của HoldingRegister 40001. Khi gọi
//method đọc giá trị của HoldingRegister, thì nó sẽ chuyển đổi giá trị kiểu word sang kiểu
//byte, byte cao lưu vào Data[0], byte thấp lưu vào Data[1].
//in ra màn hình giá trị Holding 0
//để lấy giá trị kiểu word từ mảng Data[], ta dùng method ở mục 4.3.12
Console.WriteLine("HoldingRegister 40001 = {0}", myPLC.GetUshortAt(Data, 0));

```

- **ReadInputRegister:** đọc thanh ghi ngõ vào (địa chỉ 30001-39999)
- + ModbusTCPClient.ReadInputRegister (ushort id, ushort startAddress, ushort numInputs, ref byte[] values)
- + Đôi số:
 - id: id của server, mặc định là 0
 - startAddress: địa chỉ ReadInputRegister bắt đầu đọc
 - numInputs: số ReadInputRegister muốn đọc, tính từ startAddress
 - values: mảng để lưu giá trị của các ReadInputRegister đọc về, nếu mảng này bằng null nghĩa là đọc không thành công
- + Ví dụ: đọc 1 thanh ghi ReadInputRegister có địa chỉ là 30001

```

Byte[] Data = new byte[2];// khai báo mảng để chứa các giá trị đọc về
myPLCPi.ModbusTCPClient.ReadInputRegister(0,0,1,ref Data);//đọc 1 ReadInputRegister
//vùng ReadInputRegister có kiểu dữ liệu là word, Nên cách lấy giá trị như đọc
//HoldingRegister
//in ra màn hình giá trị ReadInputRegister 30001
Console.WriteLine("ReadInputRegister 30001 = {0}", myPLC.GetUshortAt(Data, 0));

```

- **WriteSingleCoil:** ghi giá trị vào coil
 - + ModbusTCPClient.WriteSingleCoil(ushort id, ushort startAddress, bool OnOff, ref byte[] result)
 - + Đôi số:
 - id: id của server, mặc định là 0
 - startAddress: địa chỉ coil cần ghi
 - OnOff: giá trị muốn ghi vào coil, kiểu boolean(true hoặc false)
 - result: mảng chứa kết quả của việc ghi dữ liệu vào coil, nếu mảng này bằng null nghĩa là ghi không thành công
 - + Ví dụ: ghi giá trị true vào coil 1


```
Byte[] Data = new byte[1];// khai báo mảng để chứa kết quả ghi
myPLCPi.ModbusTCPClient.WriteSingleCoil(0,0,1,ref Data);//ghi giá trị vào coil
if(Data == null)
{
    Console.WriteLine("Ghi khong thanh cong");
}
```
- **WriteMultipleCoils:** ghi giá trị vào nhiều coil một lúc
 - + ModbusTCPClient. WriteMultipleCoils (ushort id, ushort startAddress, ushort numBits, byte[] values, ref byte[] result)
 - + Đôi số:
 - id: id của server, mặc định là 0
 - startAddress: địa chỉ coil cần ghi
 - numBits: số coil muốn ghi
 - values: giá trị muốn ghi vào coil, mỗi coil là 1 bit của byte
 - result: mảng chứa kết quả của việc ghi dữ liệu vào coil, nếu mảng này bằng null nghĩa là ghi không thành công
 - + Ví dụ: ghi giá trị true vào coil 1 tới 10


```
Byte[] Data = {255,255}// khai báo mảng để chứa trạng thái ghi
Byte[] Mang = new byte[1];//mảng để chứa kết quả ghi
//ghi giá trị vào coil
myPLC.ModbusTCPClient.WriteMultipleCoils(0, 0, 9, Data, ref Mang);
if(Mang == null)
```

```

{
    Console.WriteLine("Ghi khong thanh cong");
}

```

- **WriteSingleRegister:** ghi giá trị vào 1 thanh ghi Holding
 - + ModbusTCPClient.WriteSingleRegister(ushort id, ushort startAddress, byte[] values, ref byte[] result)
 - + Đôi số:
 - id: id của server, mặc định là 0
 - startAddress: địa chỉ coil cần ghi
 - values: mảng chứa giá trị muốn ghi vào thanh ghi, vì thanh ghi Holding có kiểu dữ liệu là word, nên mỗi thanh ghi sẽ chiếm 2byte
 - result: mảng chứa kết quả của việc ghi dữ liệu vào thanh ghi, nếu mảng này bằng null nghĩa là ghi không thành công
 - + Ví dụ: ghi giá trị vào Holding 40001

```

Byte[] value = {1,255}; // khai báo mảng để chứa trạng thái ghi
// lúc này giá trị ghi vào Holding 40001 là  $1 * 256 + 255 = 511$ . Ngoài ra, ta có thể dùng
// method ở mục 4.3.12 để set giá trị kiểu word cho mảng value, như sau
// myPLCPi.SetWord(value, 0, 511); khi đó, value[0] = 1, value[1] = 255
Byte[] Mang = new byte[1]; // mảng để chứa kết quả ghi
// ghi giá trị vào thanh ghi
myPLC.ModbusTCPClient.WriteSingleRegister(0, 0, value, ref Mang);
if(Mang == null)
{
    Console.WriteLine("Ghi khong thanh cong");
}

```

- **WriteMultipleRegister:** ghi giá trị vào nhiều thanh ghi Holding một lúc
 - + ModbusTCPClient.WriteMultipleRegister(ushort id, ushort startAddress, byte[] values, ref byte[] result)
 - + Đôi số:
 - id: id của server, mặc định là 0
 - startAddress: địa chỉ coil cần ghi

- values: mảng chứa giá trị muốn ghi vào thanh ghi, vì thanh ghi Holding có kiểu dữ liệu là word, nên mỗi thanh ghi sẽ chiếm 2byte
 - result: mảng chứa kết quả của việc ghi dữ liệu vào thanh ghi, nếu mảng này bằng null nghĩa là ghi không thành công
- + Ví dụ: ghi giá trị 10000 vào Holding 40001 và 40002

```

Byte[] value = {39,16,39,16}; // khai báo mảng để chứa trạng thái ghi
//lúc này giá trị ghi vào Holding là 39*256 + 16 = 10000. Ngoài ra, ta có thể dùng //method
ở mục 4.3.12 để set giá trị kiểu word cho mảng value, như sau
//myPLCPi.SetWord(value,0,10000); myPLCPi.SetWord(value,2,10000);
Byte[] Mang = new byte[1]; //mảng để chứa kết quả ghi
//ghi giá trị vào thanh ghi
myPLC.ModbusTCPClient.WriteMultipleRegister(0, 0, value, ref Mang);
if(Mang == null)
{
    Console.WriteLine("Ghi khong thanh cong");
}

```

4.3.15. ModbusRTUMaster:

- Dùng để truyền thông theo chuẩn ModbusRTU
- Các hàm chức năng :

Hàm	Mục đích
KetNoi	Tạo kết nối tới thiết bị đang chạy modbus RTU Slave
NgatKetNoi	ngắt kết nối đến modbus TCP Slave
ReadCoils	Đọc vùng nhớ coil
ReadDiscreteInputContact	Đọc vùng nhớ discrete inputs
ReadHoldingRegister	Đọc thanh ghi Holding
ReadInputRegisters	Đọc thanh ghi ngõ vào
WriteSingleCoil	Ghi dữ liệu xuống 1 coil
WriteMultipleCoils	Ghi dữ liệu xuống nhiều coil một lúc
WriteHoldingRegisters	Ghi dữ liệu xuống thanh ghi Holding

- KetNoi:** kết nối đến slave

- + ModbusRTUMaster.KetNoi(string portName, int baudRate, int databits, Parity parity, StopBits stopBits)
- + Đôi số:
 - portName: cổng serial của slave khi kết nối với PLCPi
 - baudRate: tốc độ baud
 - databits: số bit
 - parity: chẵn lẻ
 - stopBits: bit dừng
- + Trả về trạng thái ghi kiểu boolean. Nếu = true thành công, = false thất bại
- + Sau khi ta kết nối slave vào PLCPi, để biết portName ta làm như sau:
 - Ta dùng phần mềm putty kết nối vào PLCPi, sau đó nhập vào dòng lệnh “ls /dev/ttyUSB*” như hình

```

192.168.1.31 - PuTTY
login as: root
root@192.168.1.31's password:
Linux raspberrypi 3.18.11-v7+ #781 SMP PREEMPT Tue Apr 21 18:07:59 BST 2015 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Aug 15 15:25:30 2015 from 192.168.1.115
root@raspberrypi:~# ls /dev/ttyUSB*

```

- Sau đó bấm Enter ta có các thông tin cổng serial kết đang kết nối tới PLCPi

```

192.168.1.31 - PuTTY
login as: root
root@192.168.1.31's password:
Linux raspberrypi 3.18.11-v7+ #781 SMP PREEMPT Tue Apr 21 18:07:59 BST 2015 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Aug 15 15:25:30 2015 from 192.168.1.115
root@raspberrypi:~# ls /dev/ttysUSB*
/dev/ttysUSB0  /dev/ttysUSB1
root@raspberrypi:~# 
```

- Ở đây đang có 2 cổng serial kết nối là "/dev/ttysUSB0" và "/dev/ttysUSB1". Ta xác định xem cổng nào là cổng dùng làm truyền thông thì ta chọn cổng đó
 - + Ví dụ:

```

PLCPi myPLCPi = new PLCPi(); // tạo đối tượng myPLCPi
if(myPLC.ModbusRTUMaster.KetNoi("/dev/ttysUSB0", 9600, 8, System.IO.Ports.Parity.None, System.IO.Ports.StopBits.One) == true)
{
    Console.WriteLine("ket noi thanh cong")
} 
```

- **NgatKetNoi:** ngắt kết nối đến slave
 - + ModbusRTUMaster.NgatKetNoi ()
 - + Trả về trạng thái ghi kiểu boolean. Nếu = true thành công, = false thất bại
 - + Ví dụ:

```
myPLCPi.ModbusRTUMaster.NgatKetNoi(); 
```

- **ReadCoils:** đọc coil (địa chỉ 1-9999)
 - + ModbusRTUMaster.ReadCoils(byte address, ushort start, ushort coils, ref bool[] values)
 - + Đôi số:
 - address: địa chỉ của slave
 - start: địa chỉ coil bắt đầu đọc
 - coils: số lượng coil muốn đọc, tính từ start

- values: mảng kiểu boolean, truyền vào để lưu các giá trị đọc về từ coil
- + Trả về trạng thái đọc kiểu boolean. Nếu = true thành công, = false thất bại
- + Ví dụ: đọc về 10 coil từ coil0 đến coil9

```
bool[] Data = new bool[10];// khai báo mảng để chứa các giá trị đọc về
//đọc 10 coil
If(myPLCPi.ModbusRTUMaster.ReadCoils(0,0,10,ref Data) == true)
{
    //ghi trạng thái coil 7 ra màn hình
    Console.WriteLine(Data[6]);//in ra màn hình
}
```

- **ReadDiscreteInputContact:** địa chỉ (10001-19999)

- + ModbusRTUMaster.ReadDiscreteInputContact(byte address, ushort start, ushort inputs, ref bool[] values)
- + Đôi số:
 - address: địa chỉ của slave
 - start: địa chỉ ngõ vào rời rạc bắt đầu đọc
 - inputs: số lượng ngõ vào rời rạc muốn đọc, tính từ start
 - values: mảng kiểu boolean, truyền vào để lưu các giá trị đọc về từ ngõ vào rời rạc
- + Trả về trạng thái đọc kiểu boolean. Nếu = true thành công, = false thất bại
- + Ví dụ: đọc 10 ngõ vào rời rạc, từ 0 đến 9

```
bool[] Data = new bool[2];// khai báo mảng để chứa các giá trị đọc về
//đọc 10 DiscreteInputs
if(myPLCPi.ModbusRTUMaster.ReadDiscreteInputContact(0,0,10,ref Data) == true)
{
    Console.WriteLine("DiscreteInputs0 = {0}",Data[0]);
}
```

- **ReadHoldingRegisters:** đọc thanh ghi Holding (địa chỉ 40001-49999)

- + ModbusRTUMaster.ReadHoldingRegisters(byte address, ushort start, ushort registers, ref byte[] values)
- + Đôi số:
 - address: địa chỉ của slave

- start: địa chỉ HoldingRegisters bắt đầu đọc
- registers: số lượng HoldingRegisters cần ghi, tính từ start
- values: mảng kiểu byte, truyền vào để lưu các giá trị đọc về từ HoldingRegisters
- + Trả về trạng thái đọc kiểu boolean. Nếu = true thành công, = false thất bại
- + Ví dụ: đọc 1 thanh ghi Holding có địa chỉ là 40001

```

Byte[] Data = new byte[2];// khai báo mảng để chứa các giá trị đọc về
//đọc 1 HoldingRegister
//vùng HoldingRegister có kiểu dữ liệu là word, nên cần 2 byte để lưu giá trị của 1 thanh ghi
//Holding. Mảng Data gồm 2 phần tử để lưu giá trị của HoldingRegister 40001. Khi gọi
//method đọc giá trị của HoldingRegister, thì nó sẽ chuyển đổi giá trị kiểu word sang kiểu
//byte, byte cao lưu vào Data[0], byte thấp lưu vào Data[1].
//để lấy giá trị kiểu word từ mảng Data[], ta dùng method ở mục 4.3.12
If(myPLCPi.ModbusRTUMaster.ReadHoldingRegisters(0,0,1,ref Data) == true)
{
    //in ra màn hình giá trị của thanh ghi
    Console.WriteLine("HoldingRegister 40001 = {0}", myPLC.GetUshortAt(Data, 0));
}

```

- **ReadInputRegister:** đọc thanh ghi ngõ vào (địa chỉ 30001-39999)
- + ModbusRTUMaster.ReadInputRegisters(byte address, ushort start, ushort registers, ref byte[] values)
- + Đối số:
 - address: địa chỉ của slave
 - start: địa chỉ InputRegisters bắt đầu đọc
 - registers: số lượng InputRegisters cần ghi, tính từ start
 - values: mảng kiểu byte, truyền vào để lưu các giá trị đọc về từ InputRegisters
- + Trả về trạng thái đọc kiểu boolean. Nếu = true thành công, = false thất bại
- + Ví dụ: đọc 1 thanh ghi InputRegister có địa chỉ là 30001

```

Byte[] Data = new byte[2];// khai báo mảng để chứa các giá trị đọc về
//đọc 1 InputRegister
If (myPLCPi.ModbusRTUMaster.ReadInputRegisters(0,0,1,ref Data))
{

```

```

    //vùng ReadInputRegister có kiểu dữ liệu là word, Nên cách lấy giá trị như đọc
    //HoldingRegister
    //in ra màn hình giá trị ReadInputRegister 30001
    Console.WriteLine("ReadInputRegister 30001 = {0}", myPLC. GetUshortAt(Data,
        0));
}

```

- **WriteSingleCoil:** ghi giá trị vào coil

- + ModbusRTUMaster.WriteSingleCoil(byte address, ushort start, bool value)
- + Đôi số:
 - address: địa chỉ của slave
 - start: địa chỉ coil cần ghi
 - values: giá trị ghi vào coil, kiểu boolean(true hoặc false)
- + Trả về trạng thái ghi kiểu boolean. Nếu = true ghi thành công, = false ghi thất bại
- + Ví dụ: ghi giá trị true vào coil 1

```

//ghi giá trị vào coil
if(myPLCPi.ModbusRTUMaster.WriteSingleCoil(0,0,true)==false)
{
    Console.WriteLine("Ghi khong thanh cong");
}

```

- **WriteMultipleCoils:** ghi giá trị vào nhiều coil một lúc

- + ModbusRTUMaster.WriteMultipleCoils(byte address, ushort start, ushort coils, bool[] values)
- + Đôi số:
 - address: địa chỉ của slave
 - start: địa chỉ coil bắt đầu ghi
 - coils: số lượng coil cần ghi, tính từ start
 - values: mảng chứa giá trị muốn ghi vào coil, mảng kiểu boolean
- + Trả về trạng thái ghi kiểu boolean. Nếu = true ghi thành công, = false ghi thất bại
- + Ví dụ: ghi giá trị true vào coil 1 tới 10

```

bool[] Data = new bool[10];// khai báo mảng để chứa trạng thái ghi
//ghi giá trị vào coil
for( int i = 0; i < 10; i++)

```

```

{
    Data[i] = true;
}
if(myPLC.ModbusRTUMaster.WriteMultipleCoils(0, 0, 9, Data) == false)
{
    Console.WriteLine("Ghi khong thanh cong");
}

```

- **WriteHoldingRegisters:** ghi giá trị vào thanh ghi Holding
 - + ModbusRTUMaster.WriteHoldingRegisters(byte address, ushort start, ushort registers, byte[] values)
 - + Đôi số:
 - address: địa chỉ của slave
 - start: địa chỉ HoldingRegisters bắt đầu ghi
 - registers: số lượng thanh ghi cần ghi, tính từ start
 - values: mảng chứa giá trị muốn ghi vào thanh ghi, vì thanh ghi Holding có kiểu dữ liệu là word, nên mỗi thanh ghi sẽ chiếm 2byte
 - + Trả về trạng thái ghi kiểu boolean. Nếu = true ghi thành công, = false ghi thất bại
 - + Ví dụ: ghi giá trị 10000 vào Holding 40001, 40005, 40006

```

Byte[] value = {39,16,39,16}; // khai báo mảng để chứa trạng thái ghi
// lúc này giá trị ghi vào Holding là 39*256 + 16 = 10000. Ngoài ra, ta có thể dùng //method
ở mục 4.3.12 để set giá trị kiểu word cho mảng value, như sau
//myPLCPi.SetWord(value,0,10000); myPLCPi.SetWord(value,2,10000);
//ghi giá trị vào thanh ghi 40001
if(myPLC.ModbusRTUMaster.WriteHoldingRegisters(0, 0, 1, value) == false)
{
    Console.WriteLine("Ghi khong thanh cong");
}

//ghi giá trị vào thanh ghi 40005,40006
if(myPLC.ModbusRTUMaster.WriteHoldingRegisters(0, 4, 2, value) == false)
{
    Console.WriteLine("Ghi khong thanh cong");
}

```

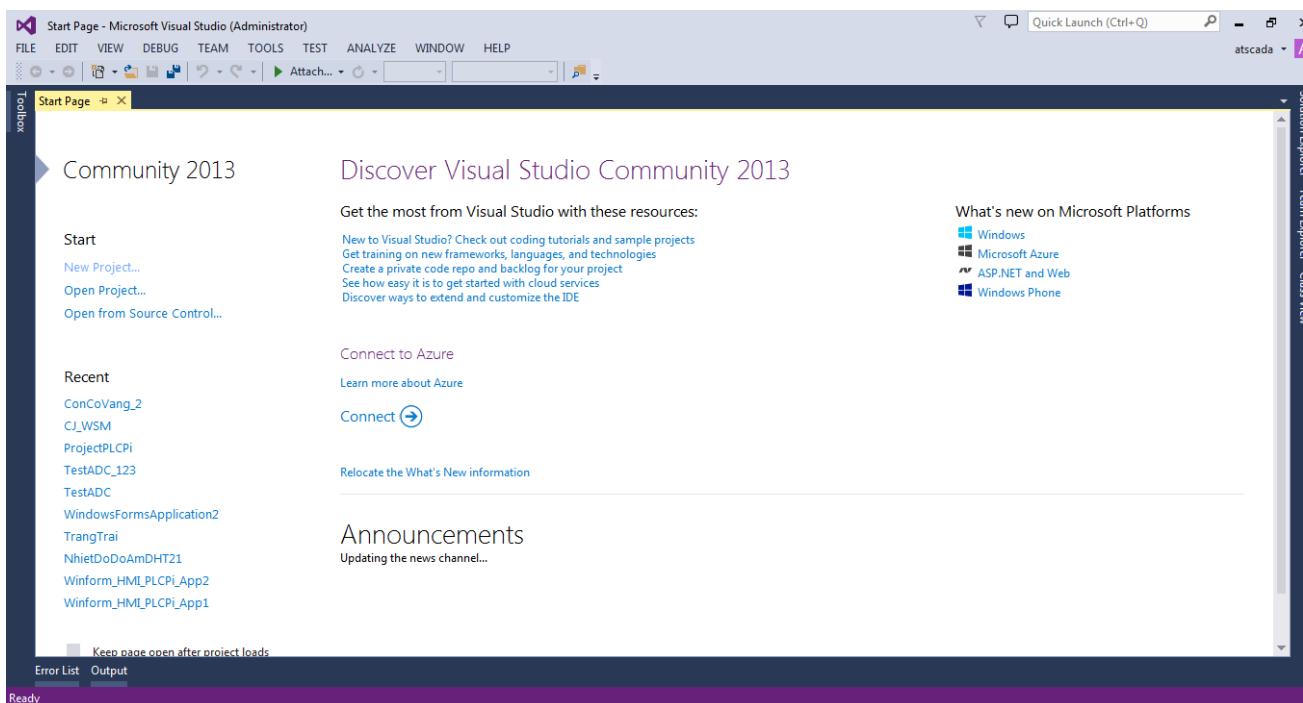
}

5. CÁC VÍ DỤ:

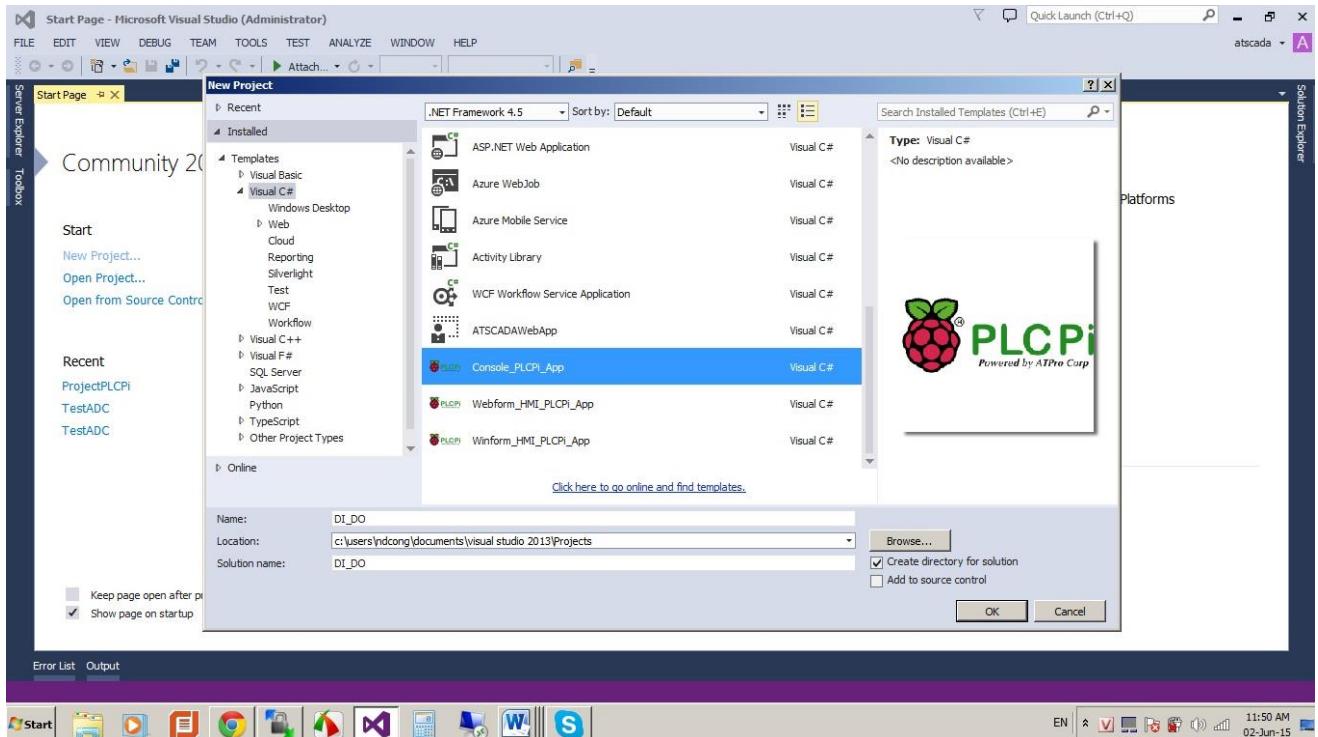
5.1. Hướng dẫn tạo dự án Console_PLCPi:

5.1.1. Ví dụ 1: DI_DO

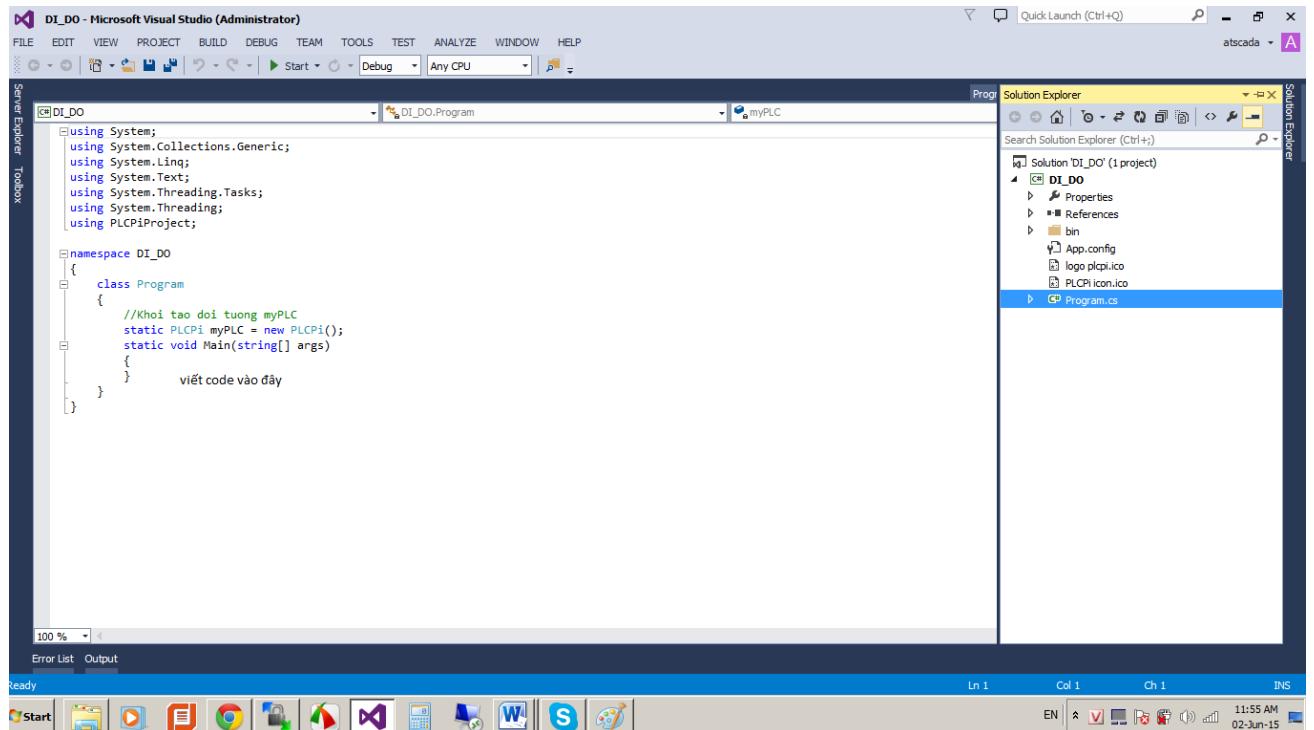
- Giới thiệu:
 - + Đây là ví dụ về sử dụng các method ngõ vào, ngõ ra trong thư viện PLCPi, sử dụng giao diện Console
 - + Tải về tại đây: <http://atscada.com/plcpi/> (mục Examples, Project/Console, dự án tên: DI_DO)
- Yêu cầu:
 - + Đọc ngõ vào I0.0 xuất trạng thái bit đó ra ngõ ra Q0.0
 - + Đọc ngõ vào I0 theo byte rồi xuất byte đó ra ngõ ra Q0
 - + In ra màn hình trạng thái ngõ ra Q0
- Hướng dẫn tạo dự án PLC với thư viện PLCPi.dll:
 - + Bước 1: Mở Visual Studio lên, tạo 1 dự án Console mới



- Chọn New Project, lúc này xuất hiện giao diện như hình.



- Chọn Visual C#. Trong ô ở giữa có 3 templates để lập trình cho PLCPi:
 - * Console_PLCPI_App: Console template
 - * Webform_HMI_PLCPI_App: Webform template
 - * Winform_HMI_PLCPI_App: giao diện Winform template
 - * **Chúng ta có thể chọn 1 trong 3 template này để viết ứng dụng, tùy thuộc vào mục đích của chúng ta. Nếu bạn chỉ lập trình ứng dụng điều khiển, không có giám sát thì dùng Console template, nếu lập trình ứng dụng có kết nối lên LCD thường hoặc TouchScreen LCD để điều khiển giám sát thì bạn chọn Winform template, nếu bạn muốn truy cập từ xa vào PLCPi thì chọn Webform template.**
- Ở đây ta chọn template Console_PLCPI_App
- Đặt tên cho Project xong ta chọn OK
- Ta được giao diện như hình:



+ Bước 2: viết code

- Tạo đối tượng: khởi tạo 1 đối tượng myPLC từ thư viện PLCPi

```
//Khai tao doi tuong myPLC
static PLCPi myPLC = new PLCPi();
```

- Xuất ngõ ra Q0 theo bit, với giá trị đọc từ ngõ vào I0 (đọc theo bit)

```
myPLC.NgoRa.XuatNgoRa("Q0.0", myPLC.NgoVao.DocNgoVao("I0.0"));
```

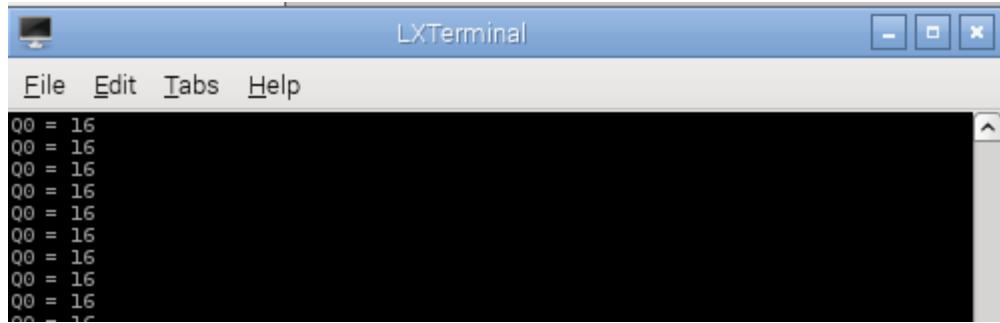
- Xuất ngõ ra Q0 theo byte, với giá trị đọc từ ngõ vào I0 (đọc theo byte)

```
myPLC.NgoRa.XuatNgoRa("Q0", myPLC.NgoVao.DocNgoVao("I0"));
```

- Đọc trạng thái ngõ ra Q0 rồi in ra màn hình

```
Console.WriteLine("Q0 = {0}", myPLC.NgoRa.DocNgoRa("Q0"));
```

- + Bước 3: Sau khi code xong ta tiến hành biên dịch và copy thư mục Debug sang PLCPi để chạy (xem mục 3.2.4)
- + Kết quả chạy chương trình trên PLCPi



5.1.2. Ví dụ 2: DS18B20:

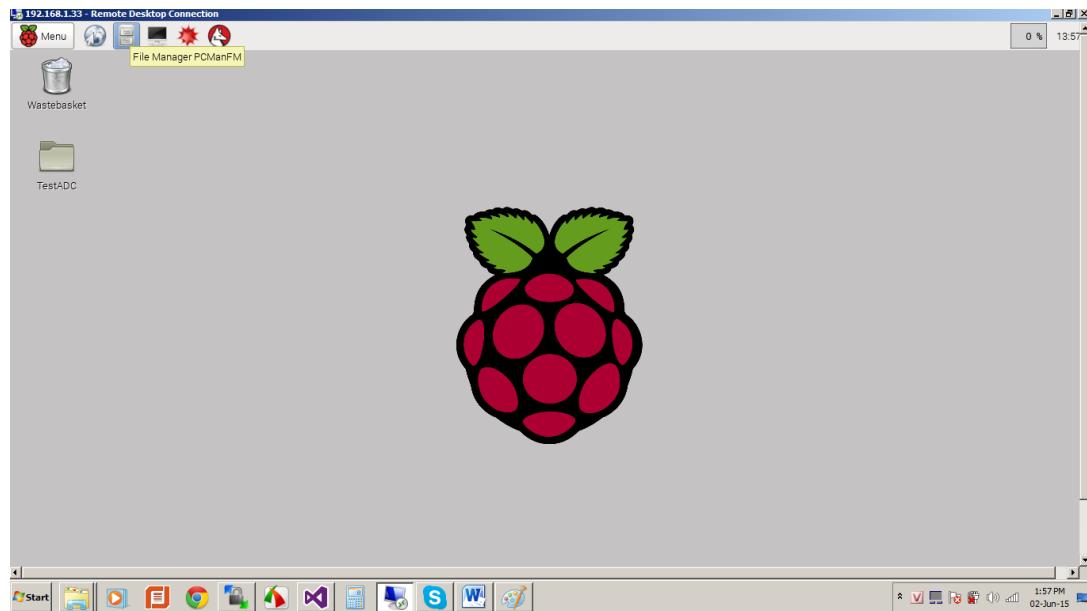
- Giới thiệu:
 - + Đây là ví dụ về sử dụng method DocNhiетDo từ cảm biến DS18B20, sử dụng giao diện Console
 - + Tải về tại đây: <http://atscada.com/plcpi/> (mục Examples, Project/Console, dự án tên: DS18B20)
- Yêu cầu:
 - + Đọc nhiệt độ từ cảm biến DS18B20
 - + Hiển thị giá trị nhiệt độ ra kênh 1 của module led 7. Tắt hiển thị kênh 2
 - + In ra màn hình giá trị nhiệt độ đọc được
- Hướng dẫn tạo dự án PLC với thư viện PLCPi.dll:
 - + Bước 1: Tạo 1 project mới như ở Ví dụ 1
 - + Bước 2: viết code
 - Tạo đối tượng myPLC và biến để lưu giá trị nhiệt độ đọc về

```
static PLCPi myPLC = new PLCPi();
static string NhiệtDo = "BAD";
```

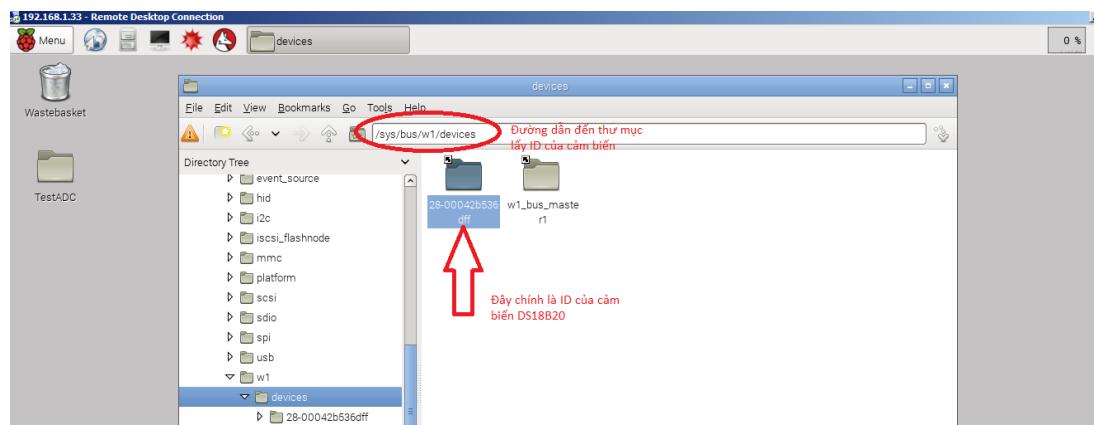
- Gọi method đọc nhiệt độ

```
NhiệtDo = myPLC.DS18B20.DocNhiệtDo("28-00042b536dff");
```

- * Phần gạch chân đó chính là ID của cảm biến DS18B20. Mỗi con DS18B20 sẽ có 1 địa chỉ ID riêng. Khi ta gọi method đọc nhiệt độ từ cảm biến DS18B20 thì ta phải truyền vào địa chỉ ID của nó. Dưới đây là cách để lấy địa chỉ ID của cảm biến
- * Kết nối cảm biến với PLCPi
- * Dùng Remote Desktop Connection kết nối tới PLCPi, ta được giao diện như hình dưới



- * Vào quản lý file và tìm đến đường dẫn: `/sys/bus/wl/devices`.
 - * ID của cảm biến ở đây là: `28-00042b536df`



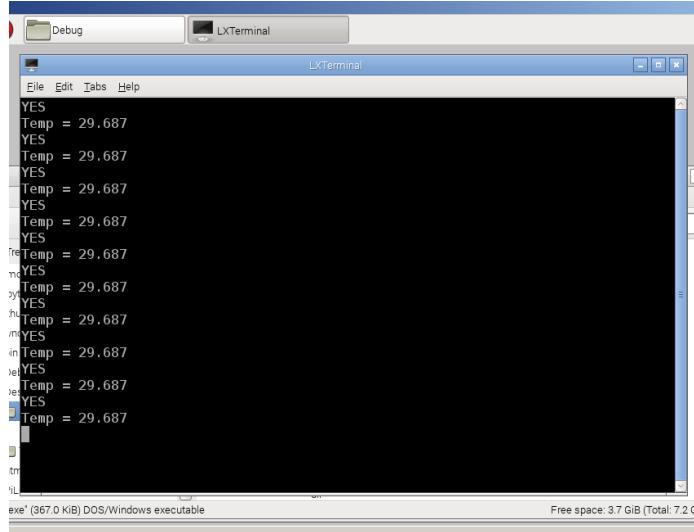
- Hiển thi giá trị nhiệt độ ra module led 7

```
myPLC.HienThiLed.HienThi(NhietDo, 1);
```

- Tắt hiển thị kênh 2 của module led 7

```
myPLC.HienThiLed.HienThi("Tat", 2);
```

- + Bước 3: Sau khi code xong ta tiến hành biên dịch và copy thư mục Debug sang PLCPi để chạy (xem mục 3.2.4)
 - + Kết quả khi chạy ứng dụng trên PLCPi



5.1.3. Ví dụ 3: DHT21:

- Giới thiệu:
 - + Đây là ví dụ về sử dụng method đọc nhiệt độ độ ẩm từ cảm biến DHT21, sử dụng giao diện Console
 - + Tải về tại đây: <http://atscada.com/plcpi/> (mục Examples, Project/Console, dự án tên: DHT21)
- Yêu cầu:
 - + Đọc nhiệt độ từ cảm biến DHT21 hiển thị ra kênh led 1
 - + Đọc độ ẩm từ cảm biến DHT21 hiển thị ra kênh led 2
 - + In ra màn hình giá trị nhiệt độ, độ ẩm đọc được
- Hướng dẫn tạo dự án PLC với thư viện PLCPi.dll:
 - + Bước 1: tạo 1 project mới như ở Ví dụ 1
 - + Bước 2: viết code
 - Tạo đối tượng myPLC và khai báo các biến để lưu các giá trị nhiệt độ, độ ẩm đọc về

```
static PLCPi myPLC = new PLCPi();
static string NhietDo = null, DoAm = null;
```

- Gọi các method đọc nhiệt độ, độ ẩm

```
NhietDo = myPLC.DHT21.DocNhietDo();
DoAm = myPLC.DHT21.DocDoAm();
```

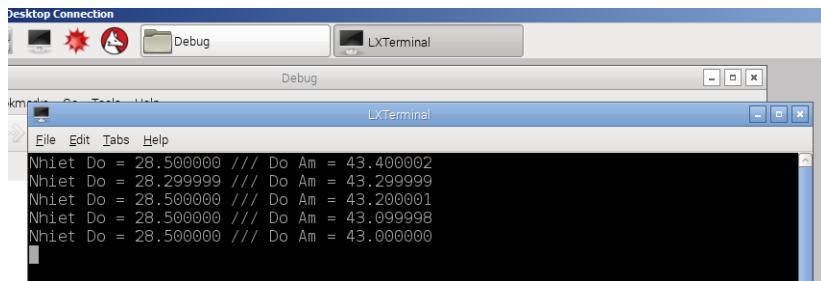
- Hiển thị ra module led 7 và in ra màn hình giá trị nhiệt độ, độ ẩm

```

myPLC.HienThiLed.HienThi(NhietDo, 1); //hiển thị giá trị nhiệt độ ra kênh led 1
myPLC.HienThiLed.HienThi(DoAm, 2); //hiển thị giá trị độ ẩm ra kênh led 2
Console.WriteLine("Nhiet Do = {0} /// Do Am = {1}", NhietDo, DoAm); //in giá trị ra màn hình

```

- + Bước 3: Sau khi code xong ta tiến hành biên dịch và copy thư mục Debug sang PLCPi để chạy (xem mục 3.2.4)
- + Kết quả khi chạy trên PLCPi



5.1.4. Ví dụ 4: ThoiGian:

- Giới thiệu:
 - + Đây là ví dụ về sử dụng method đọc thời gian từ PLCPi, cài đặt thời gian cho PLCPi và module thời gian thực, sử dụng giao diện Console
 - + Tải về tại đây: <http://atscada.com/plcpi/> (mục Examples, Project/Console, dự án tên: ThoiGian)
- Yêu cầu:
 - + Đọc thời gian từ PLCPi rồi in ra màn hình
 - + Cài đặt thời gian cho PLCPi và module thời gian thực
 - + Đọc và in ra màn hình thời gian mới cập nhật
- Hướng dẫn tạo dự án PLC với thư viện PLCPi.dll:
 - + Bước 1: tạo 1 project mới như ở Ví dụ 1
 - + Bước 2: viết code
 - Tạo đối tượng myPLC và khai báo biến để lưu các giá trị thời gian đọc về

```

static PLCPi myPLC = new PLCPi();
static string[] mang = new string[6];

```

- Gọi method đọc thời gian từ PLCPi

```

mang = myPLC.ThoiGian.DocThoiGian();

```

- In ra màn hình

```
Console.WriteLine("ngay {0} thang {1} nam {2} gio {3} phut {4} giay {5}", mang[0], mang[1], mang[2], mang[3], mang[4], mang[5]);
```

- Cài đặt thời gian cho PLCPi

```
myPLC.ThoiGian.CaiDat("02-06-2015 15:38:00");
```

- Cài đặt thời gian cho module thời gian thực(RTC_DS1307)

```
myPLC.ThoiGian.CaiDatRTC_DS1307();
```

- + Bước 3: Sau khi code xong ta tiến hành biên dịch và copy thư mục Debug sang PLCPi để chạy (xem mục 3.2.4)
- + Kết quả:

```
ngay 20 thang 08 nam 2015 gio 17 phut 35 giay 58
Sat Jun 6 15:09:00 ICT 2015
ngay 06 thang 06 nam 2015 gio 15 phut 09 giay 01
```

5.1.5. Ví dụ 5: truyền thông S7Ethernet_Server:

- Giới thiệu:
 - + Đây là ví dụ về cách để khởi tạo PLCPi thành 1 S7Ethernet Server để truyền thông với các S7Ethernet Client như: ATDriverServer; KEPServer
 - + Tải về tại đây: <http://atscada.com/plcpi/> (mục Examples, Project/Console, dự án tên: S7Ethernet_Server)
- Yêu cầu:
 - + Khởi tạo PLCPi thành S7Ethernet Server
- Hướng dẫn tạo dự án PLC với thư viện PLCPi.dll:
 - + Bước 1: tạo 1 project mới như ở Ví dụ 1
 - + Bước 2: viết code
 - Tạo đối tượng myPLC và khai báo biến Error để lấy trạng thái khởi tạo S7Ethernet Server

```
static PLCPi myPLC = new PLCPi();
static string Error = null;
```

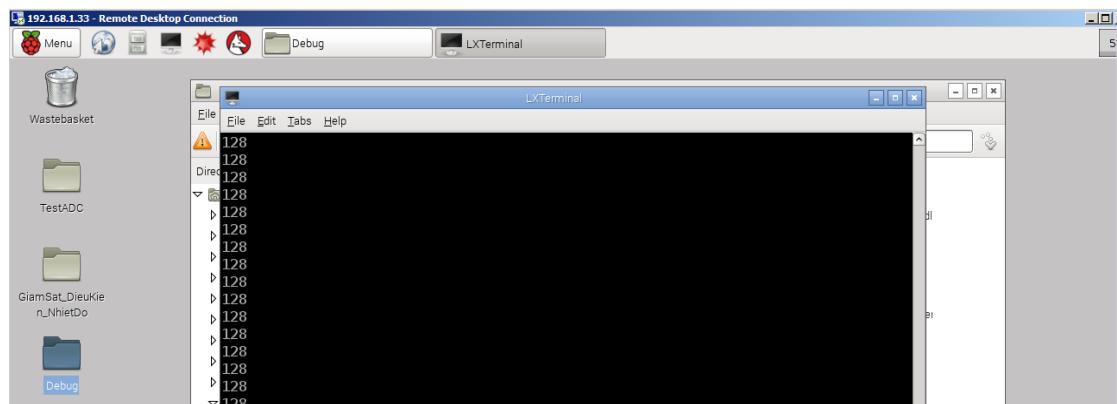
- Gọi method khởi tạo PLCPi thành 1 S7Ethernet Server

```

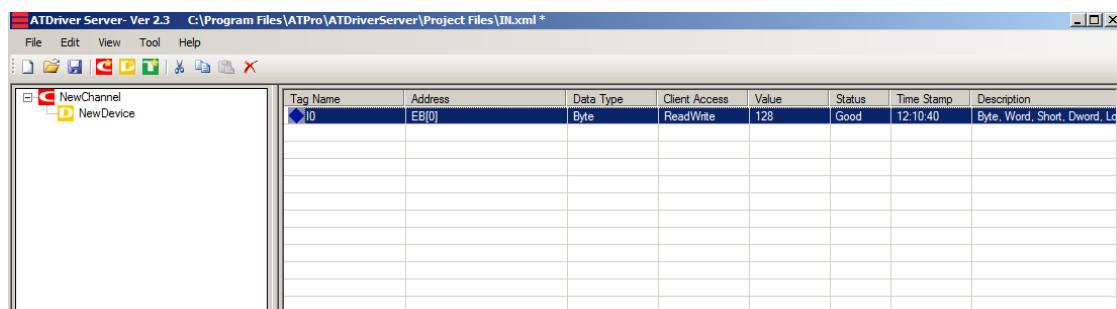
Error = myPLC.S7Ethernet.Server.Khoitao(); //khởi tạo PLCPi thành 1 server.
if (Error == "GOOD")//khởi tạo thành công
{
    while (true)
    {
        Console.WriteLine(myPLC.NgoVao.DocNgoVao("I0")); //đọc ngõ vào I0 và in ra màn hình
    }
}
else //khởi tạo server lỗi
{
    Console.WriteLine(Error); //in ra lỗi
}

```

- * Khi gọi lệnh khởi tạo S7Ethernet Server,nếu chuỗi trả về là “GOOD” nghĩa là khởi tạo thành công, khác “GOOD” thì khởi tạo thất bại
- + Bước 3: Sau khi code xong ta tiến hành biên dịch và copy thư mục Debug sang PLCPi để chạy (xem mục 3.2.4)
- + Kết quả:



- Dùng ATDriver Server làm S7Ethernet Client kết nối đến PLCPi S7Ethernet Server để lấy dữ liệu



5.1.6. Ví dụ 6: truyền thông S7Ethernet_Client:

- Giới thiệu:

- + Đây là ví dụ về cách để khởi tạo PLCPi thành 1 S7Ethernet client để truyền thông với các S7Ethernet Server khác, ví dụ như các PLC S7 1200, S7 1500 của Siemens
- + Tải về tại đây: <http://atscada.com/plcpi/> (mục Examples, Project/Console, dự án tên: S7Ethernet_Client)
- Yêu cầu:
 - + Khởi tạo PLCPi thành S7Ethernet Client
 - + Ghi dữ liệu xuống vùng nhớ Q0
 - + Đọc trạng thái ngõ ra Q0
- Hướng dẫn tạo dự án PLC với thư viện PLCPi.dll:
 - + Bước 1: tạo 1 project mới như ở Ví dụ 1
 - + Bước 2: viết code
 - Tạo đối tượng myPLC và khai báo các biến để sử dụng

```
//Khai tao doi tuong myPLC
static PLCPi myPLC = new PLCPi();
static string Error = "BAD"; //chứa trạng thái kết nối
static byte[] Ghi_DB = { 0, 0 }; //mảng chứa các giá trị để ghi xuống vùng nhớ DB của S7 Ethernet server
static DocS7 DocDB; //khai báo đối tượng để đọc vùng nhớ ngõ ra
```

- Gọi method kết nối đến 1 thiết bị làm S7Ethernet Server có IP “192.168.1.105”

```
Error = myPLC.S7Ethernet.Client.KetNoi("192.168.1.105");
```

- * Khi gọi method kết nối đến S7Ethernet Server, thì ta phải truyền vào địa chỉ IP của server ta cần kết nối đến. Server ở đây là PLC S7 1200 có địa chỉ IP là “192.168.1.105”
- * Nếu kết nối thành công thì sẽ vào thực hiện đọc và ghi vùng nhớ DB của server
- * Nếu kết nối không thành công thì sẽ in ra màn hình lỗi và thoát chương trình
- Gọi method đọc vùng nhớ DB của server

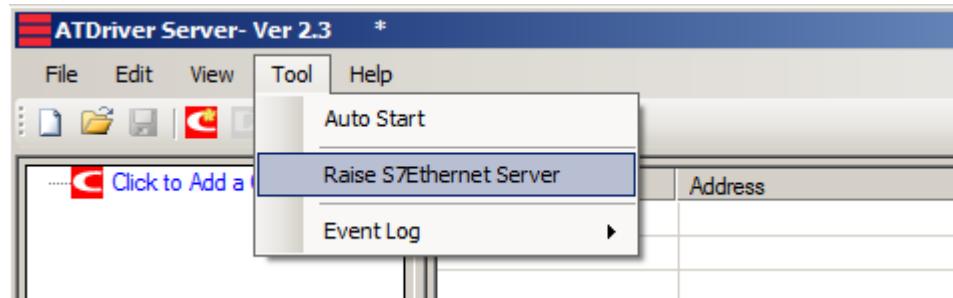
```
DocDB = myPLC.S7Ethernet.Client.DocDB(1, 0, 2);
```

- Gọi method ghi vùng nhớ DB của server

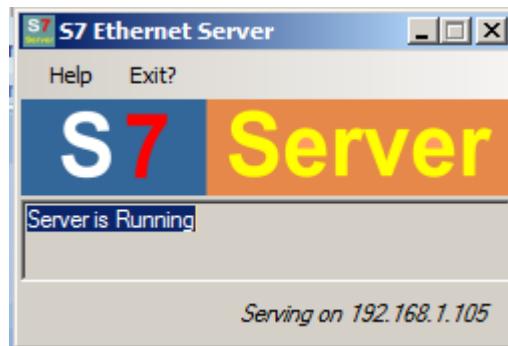
```
myPLC.S7Ethernet.Client.GhiDB(1,0, 2, Ghi_DB);
```

- + Bước 3: Sau khi code xong ta tiến hành biên dịch và copy thư mục Debug sang PLCPi để chạy (xem mục 3.2.4)

- + Kết quả:
 - Dùng ATDriver server làm S7Ethernet Server, cách làm như sau:
 - * Mở phần mềm ATDriver server lên vào Tool chọn Raise S7Ethernet Server để khởi tạo S7Ethernet Server



- * Sau khi khởi tạo xong sẽ có giao diện như hình dưới



- * Kết quả hiển thị trên PLCPi

```
LXTerminal
File Edit Tabs Help
Complete
ket noi den server thanh cong
fDB1[0] = 0 |||DB1[1] = 0
fComplete
DB1[0] = 100 |||DB1[1] = 200
```

The screenshot shows an LXTerminal window with the title 'LXTerminal'. The terminal window displays the output of a PLC program. The text shows the program has connected to the server successfully, set fDB1[0] to 0, set DB1[1] to 0, completed the process, and then set DB1[0] to 100 and DB1[1] to 200.

- Ban đầu, DB1[0] và DB1[1] có giá trị là 0
- Sau đó ghi giá trị 100 xuống DB1[0] và 200 xuống DB1[1], rồi đọc lại. Khi đó, DB1[0] = 100 và DB1[1] = 200.

5.1.7. Ví dụ 7: GiamSat_DieuKien_NhietDo

- Giới thiệu:

- + Tạo dự án Console, điều khiển giám sát nhiệt độ môi trường, giá trị nhiệt độ đọc từ cảm biến DS18B20, có truyền thông S7Ethernet Server
- + Tải về tại đây: <http://atscada.com/plcpi/> (mục Examples, Project/Console, dự án tên: GiamSat_DieuKien_NhietDo)
- Yêu cầu:
 - + Đọc giá trị nhiệt độ môi trường từ cảm biến DS18B20 hiển thị ra module hiển thị led 7, ghi giá trị nhiệt độ vào vùng nhớ DB1 của S7Ethernet Server để phục vụ truyền thông
 - + Cài đặt mức ngưỡng min max, để điều khiển nhiệt độ theo mong muốn, nếu nhiệt độ vượt ra ngoài mức ngưỡng thì sẽ điều khiển ngõ ra để bật tắt quạt và đèn để giảm hoặc tăng nhiệt độ. Giá trị min max được cài đặt thông qua truyền thông S7Ethernet, nghĩa là sẽ dùng 1 thiết bị làm S7Ethernet Client kết nối vào PLCPi và cài đặt
 - Đèn : Q0.6
 - Quạt: Q0.7
- Hướng dẫn tạo dự án PLC với thư viện PLCPi.dll:
 - + Bước 1: tạo 1 project mới như ở Ví dụ 1
 - + Bước 2: viết code
 - Với ứng dụng này, ta cần add thêm 2 thư viện sau vào để lấy đường dẫn lưu file

```
using System.IO;
using System.Reflection;
```

 - Tự động tìm đến thư mục mà mình lưu file minmax.txt, chứa các giá trị ngưỡng

```
string Path_Link = Path.GetDirectoryName(Assembly.GetExecutingAssembly().Location);
Path_Temp = Path.Combine(Path_Link, "minmax.txt");
```

 - Đọc file minmax.txt để lấy các giá trị ngưỡng khi chạy chương trình lần đầu tiên

```
StreamReader file_r = new StreamReader(Path_Temp); // tạo đối tượng để đọc file 'minmax.txt'
Line = file_r.ReadToEnd(); //đọc hết tất cả
mang = Line.Split(' '); // cắt chuỗi
min = Convert.ToDouble(mang[0]); //giá trị min chính là mang[0]
max = Convert.ToDouble(mang[1]); // giá trị max chính là mang[1]

Console.WriteLine("{0}://{1}", mang[1], mang[0]); //in giá trị min mã ra màn hình
file_r.Dispose(); //hủy đối tượng
```

 - Ghi dữ liệu vào file

```

StreamWriter file_w = new StreamWriter(Path_Temp); //tạo đối tượng để ghi dữ liệu vào file 'minmax.txt'
file_w.WriteLine(Convert.ToString(min) + " " + Convert.ToString(max)); //ghi dữ liệu vào file
file_w.Dispose(); //hủy đối tượng

```

- Trong PLCPi có các method dùng để đọc và ghi các giá trị có kiểu dữ liệu khác byte vào mảng kiểu byte. Trong ứng dụng này ta có sử dụng chức năng này
 - Ghi giá trị kiểu 32bit có dấu(4 byte) vào vùng nhớ DataBlock của PLCPi. Đây là vùng nhớ dùng để truyền thông S7Ethernet Server của PLCPi, là mảng kiểu byte

```

myPLC.SetDint(myPLC.S7Ethernet.Server.DataBlock, 4, Convert.ToInt32(max * 100));
myPLC.SetDint(myPLC.S7Ethernet.Server.DataBlock, 8, Convert.ToInt32(min * 100));

```

- Chỗ gạch chân màu đỏ chính là vị trí của byte trong mảng DataBlock bắt đầu ghi. Ở đây ghi giá trị 32bit, nên bắt đầu từ 4 thì sẽ ghi dữ liệu vào từ byte 4 đến byte 7
- Đọc từ vùng nhớ DataBlock của PLCPi, trả về giá trị 32bit(4byte) có dấu

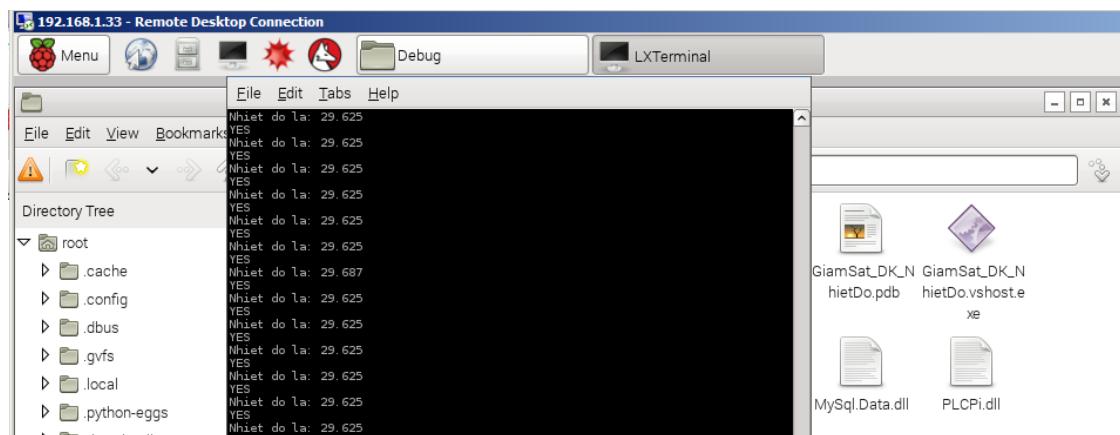
```

max_buffer = Convert.ToDouble(myPLC.GetIntAt(myPLC.S7Ethernet.Server.DataBlock, 4)) / 100;
min_buffer = Convert.ToDouble(myPLC.GetIntAt(myPLC.S7Ethernet.Server.DataBlock, 8)) / 100;

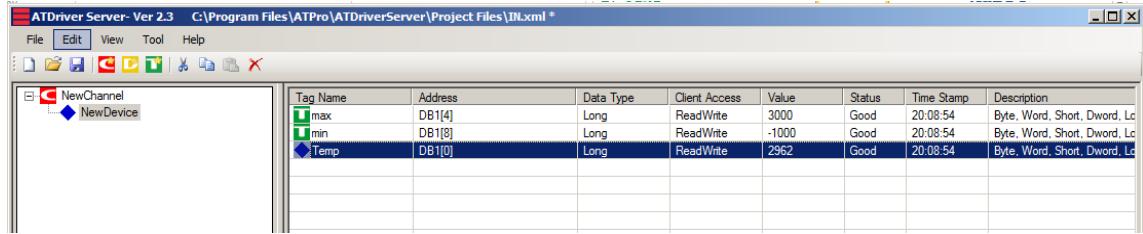
```

- Bắt đầu từ byte 4, thì sẽ lấy các byte 4 tới byte 7 chuyển đổi về Int32 rồi trả về
- Bước 3: Sau khi code xong ta tiến hành biên dịch và copy thư mục Debug sang PLCPi để chạy (xem mục 3.2.4)
- Kết quả:

- Kết quả chạy trên PLCPi



- Dùng phần mềm ATDriver Server làm S7Ethernet Client kết nối đến PLCPi để lấy giá trị nhiệt độ thực và cài đặt các mức ngưỡng nhiệt độ
 - $\text{Nhiệt độ} = 2962/100 = 29.6^{\circ}\text{C}$, $\text{max} = 3000/100 = 30^{\circ}\text{C}$, $\text{min} = -1000/10 = -10^{\circ}\text{C}$



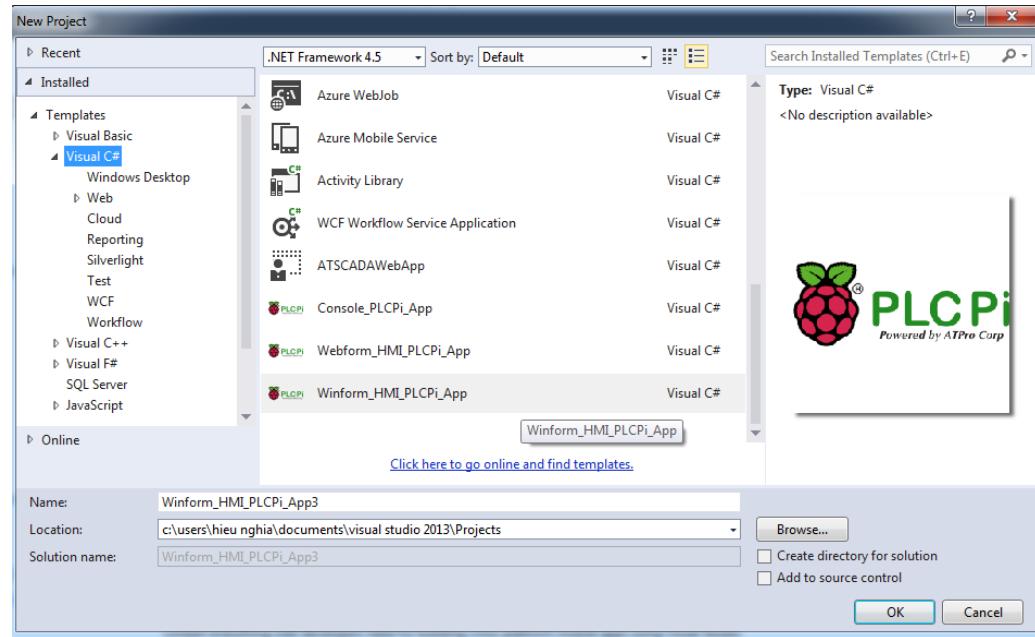
5.2. Hướng dẫn tạo dự án Winform_HMI_PLCPi:

5.2.1. Ví dụ 1: Winform_HMI_PLCPi_App3

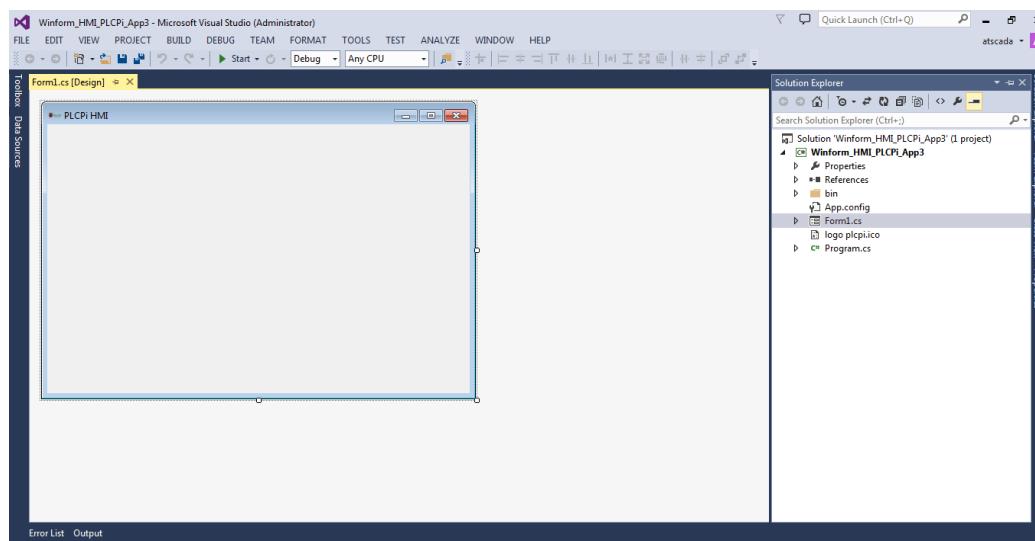
- Giới thiệu:
 - + Đây là ví dụ về cách tạo 1 dự án với giao diện Winform_HMI trên PLCPi
 - + Tải về tại đây: <http://atscada.com/plcpi/> (mục Examples, Project/Winform, dự án tên: Winform_HMI_PLCPi_App3)
- Yêu cầu:
 - + Đọc giá trị nhiệt độ rồi hiển thị lên Winform_HMI
- Hướng dẫn tạo dự án Winform_HMI trên PLCPi với thư viện PLCPi.dll:
 - + Bước 1: tạo 1 dự án winform
 - Khởi động Visual Studio 2013



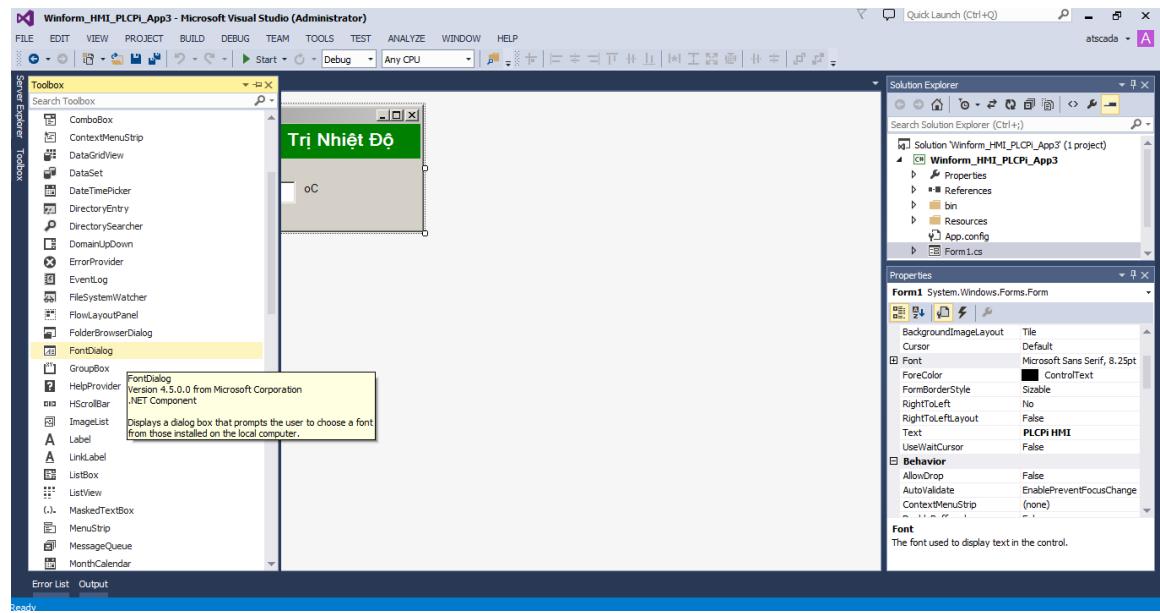
- Chọn New Project, lúc này xuất hiện giao diện như hình



- * Chọn Winform_HMI_PLCPi_App để tạo 1 dự án mới với giao diện Winform, đặt tên cho dự án và chọn vị trí lưu rồi bấm OK. Lúc này sẽ xuất hiện 1 màn hình thiết kế như hình:



- + Bước 2: thiết kế giao diện cho form
 - Từ cửa sổ Toolbox kéo thả các tool vào form thiết kế.

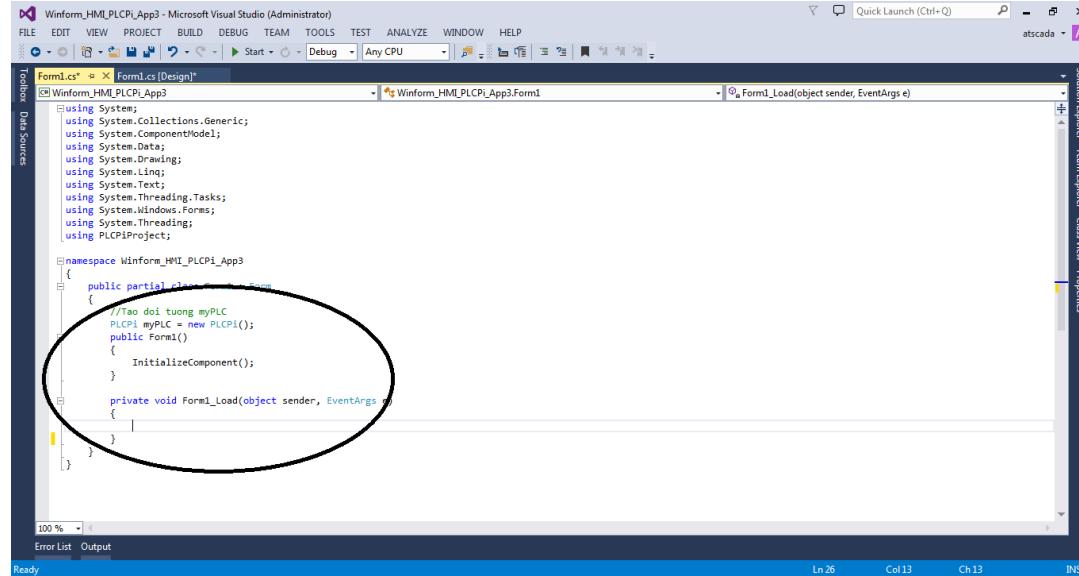


- Giao diện sau khi thiết kế xong



+ Bước 3: viết code.

- Sau khi thiết kế giao diện xong, ta kích chuột phải vào màn hình thiết kế chọn View Code hoặc double click vào các tool để tiến hành lập trình cho ứng dụng. Lúc này sẽ xuất hiện giao diện lập trình như hình:



```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Threading;
using PLCPiProject;

namespace Winform_HMI_PLCPI_App3
{
    public partial class Form1 : Form
    {
        //Tao doi tuong myPLC
        PLCPI myPLC = new PLCPI();
        public Form1()
        {
            InitializeComponent();
        }

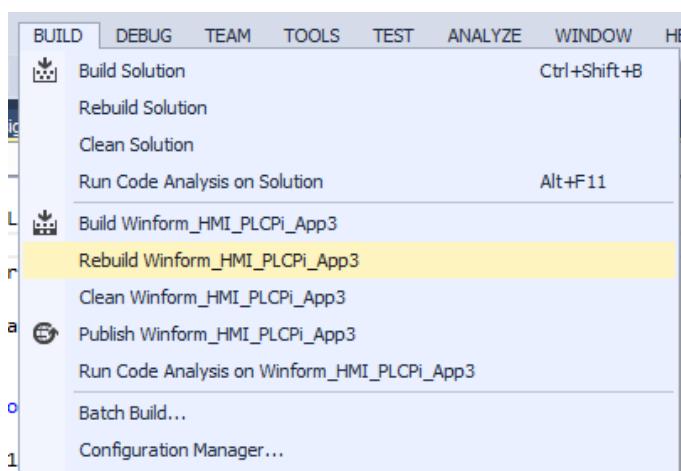
        private void Form1_Load(object sender, EventArgs e)
        {
        }
    }
}

```

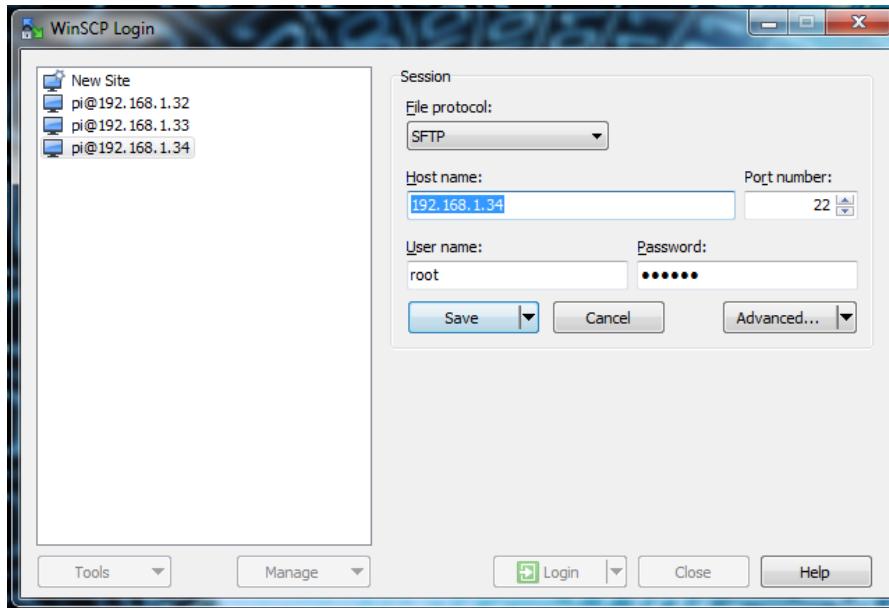
- Tại đây chúng ta sẽ tiến hành lập trình tại khu vực khoanh tròn.
- Code chương trình:
 - * Đọc nhiệt độ và hiển thị lên TextBox

`textBox1.Text = myPLC.DHT21.DocNhiетDo();`

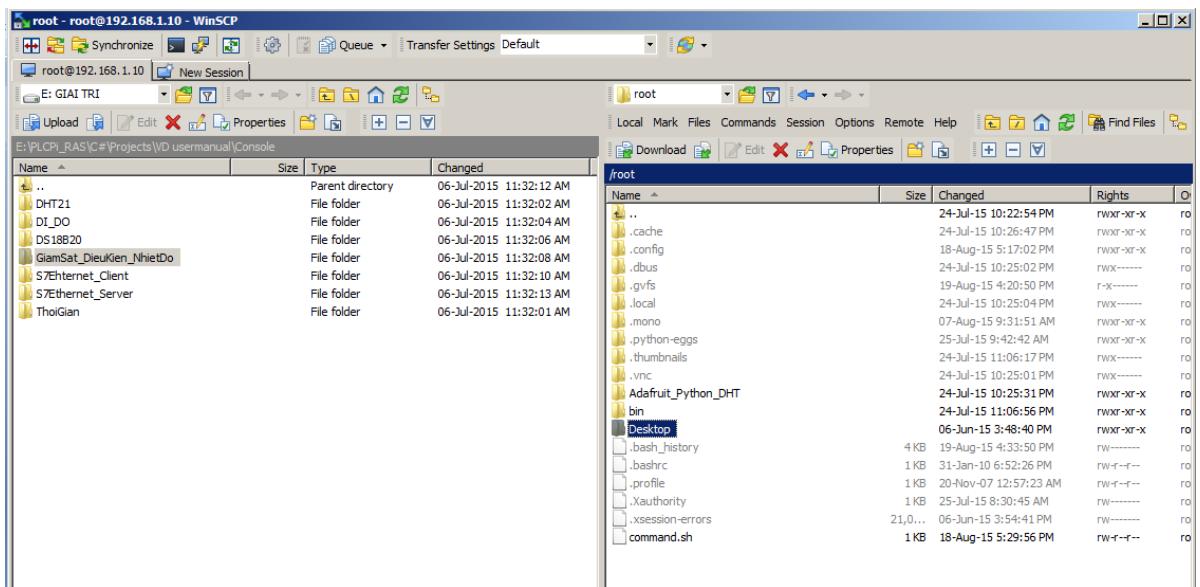
- + Bước 4: sau khi code xong, tiến hành biên dịch



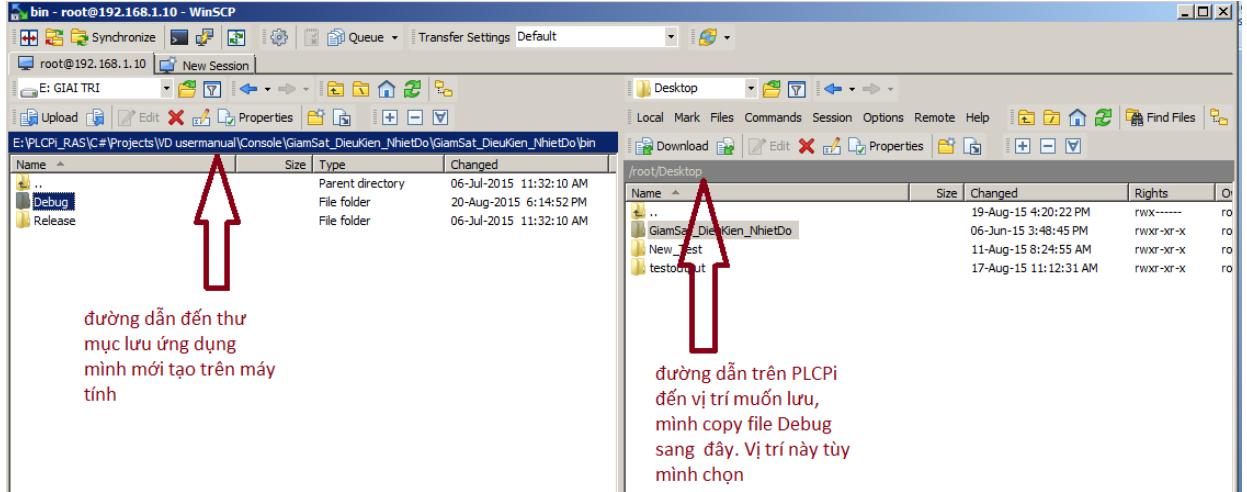
- + Bước 5: copy ứng dụng sang PLCPi và chạy ứng dụng Winform_HMI trên PLCPi:
 - Sau khi lập trình xong, tiến hành Build sau đó sử dụng phần mềm WinSCP Login để copy ứng dụng vào PLCPi để chạy



- Nhập địa chỉ Host name của PLCPi. Username và Password để tiến hành chuyển dữ liệu.
- Khi kết nối thành công sẽ xuất hiện giao diện như sau:



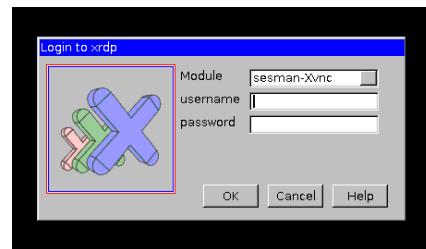
- Chọn đến đường dẫn chứa Project đã thiết kế



- Kéo thả thư mục Debug vào PLCPi(ở đây copy thư mục Debug sang thư mục /root/Desktop của PLCPi).
- Sau khi copy xong, tiến hành chạy chương trình. Trên window sử dụng phần mềm Remote Desktop Connection kết nối vào PLCPi để chạy ứng dụng đã thiết kế. Mở chương trình



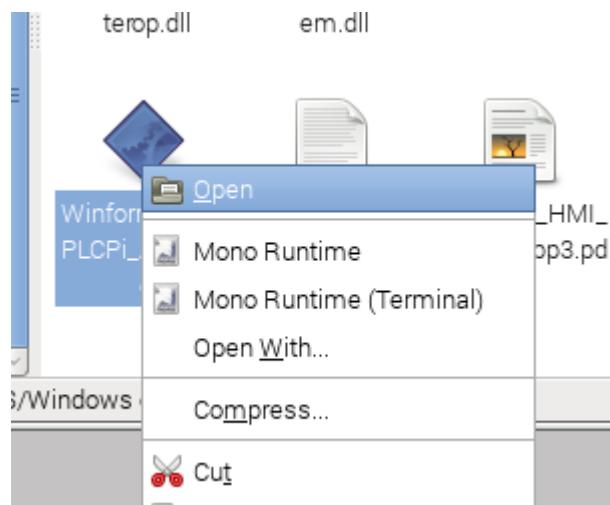
- * Nhập IP của PLCPi vào rồi bấm Connect, xuất hiện giao diện như hình



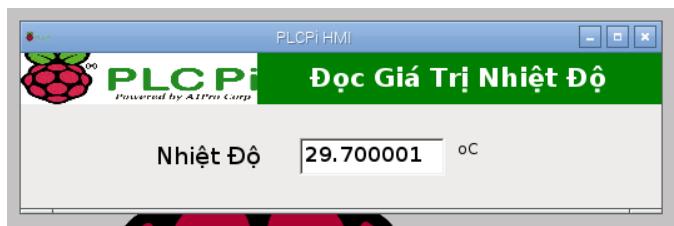
- * Username nhập vào root, password nhập vào 100100 rồi OK
- Sau khi kết nối thành công. Vào giao diện PLCPi chọn đến thư mục Debug đã sao chép sang ở bước trên.



- Có 2 cách để chạy ứng dụng Winform_HMI trên PLCPi đó là:
 - * Hoặc kích đúp chuột trái vào file ứng dụng
 - * Hoặc kích chuột phải vào ứng dụng chọn Open



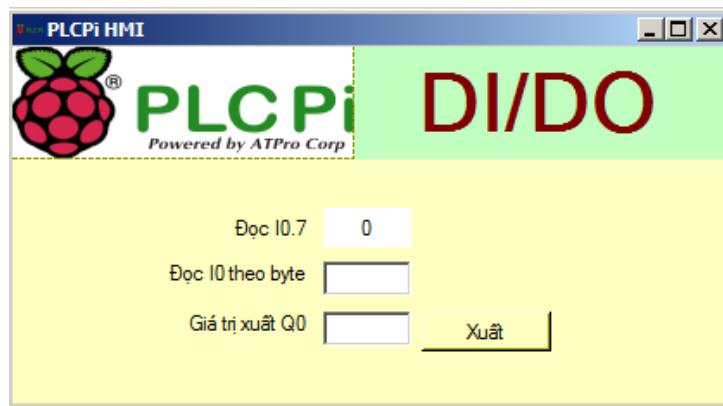
+ Kết quả như hình.



5.2.2. Ví dụ 2: DI_DO_Winform

- Giới thiệu:
 - + Đây là ví dụ về sử dụng các method ngõ vào, ngõ ra trong thư viện PLCPi, sử dụng giao diện winform
 - + Tải về tại đây: <http://atscada.com/plcpi/> (mục Examples, Project/Winform, dự án tên: DI_DO_Winform)

- Tính Năng:
 - + Thay đổi màu của label dựa vào giá trị của I0.7
 - + Đọc ngõ vào I0 theo byte hiển thị lên textbox
 - + Xuất giá trị ra ngõ ra Q0
- Hướng dẫn tạo dự án:
 - + Bước 1: tạo 1 dự án Winform_HMI_PLCPi_App mới như ở Ví dụ 1(trang 89)
 - + Bước 2: thiết kế giao diện thiết kế như hình



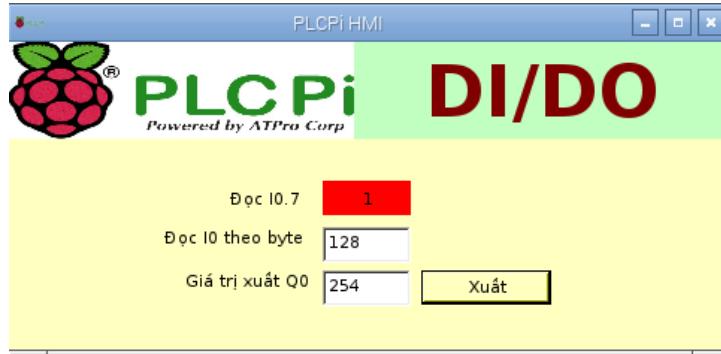
- Đọc ngõ vào I0 dùng timer quét
- Xuất ngõ ra Q0 thì nhập giá trị cần xuất vào textbox rồi bấm xuất
- + Bước 3: viết code
 - Code nút xuất:

```
myPLC.NgoRa.XuatNgoRa("Q0", Convert.ToByte(textBox3.Text));
```

- Code trong timer đọc ngõ vào I0

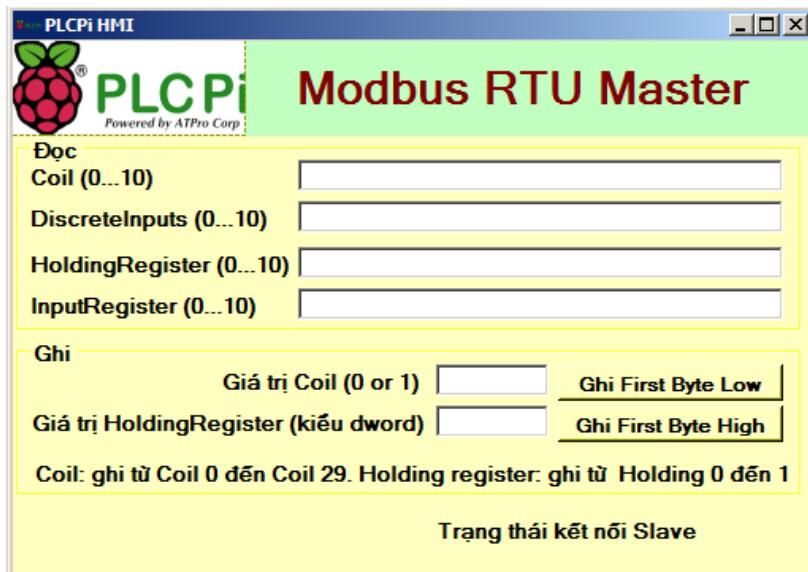
```
if (myPLC.NgoVao.DocNgoVao("I0.7") == 1)
{
    label3.BackColor = Color.Red;
    label3.Text = "1";
}
else
{
    label3.BackColor = Color.White;
    label3.Text = "0";
}
textBox2.Text = Convert.ToString(myPLC.NgoVao.DocNgoVao("I0"));
```

- + Bước 4: sau khi code xong, tiến hành biên dịch và copy sang PLCPi để chạy(xem Ví dụ 1, bước 4 và bước 5, trang92).
- + Kết quả:



5.2.3. Ví dụ 3: ModbusRTUMaster

- Giới thiệu:
 - + Đây là ví dụ về sử dụng tính năng truyền thông modbus RTU master
 - + Tải về tại đây: <http://atscada.com/plcpi/> (mục Examples, Project/Winform, dự án tên: ModbusRTUMaster)
- Tính Năng:
 - + Đọc ghi giá trị của các vùng nhớ modbus RTU Slave
- Hướng dẫn tạo dự án:
 - + Bước 1: tạo 1 dự án Winform_HMI_PLCPi_App mới như ở Ví dụ 1(trang 89)
 - + Bước 2: thiết kế giao diện thiết kế như hình



- Đọc giá trị các vùng nhớ hiển thị lên textbox
- Ghi giá trị xuống vùng nhớ coil và Holding.
 - Số coil ghi xuống: 30 coil
 - Số HoldingRegister ghi xuống: 2 Holding
 - Giá trị Coil: nhập 0 hoặc 1(tương ứng true, false)
 - Giá trị HoldingRegister: nhập vào số dword(4 byte),
 - Bấm nút Ghi First Byte Low thì sẽ lấy số dword chuyển đổi thành 4byte, rồi ghi xuống Holding, Holding0 sẽ lưu word thấp Holding1 lưu word cao
 - Bấm nút Ghi First Byte Hight thì sẽ lấy số dword chuyển đổi thành 4byte, rồi ghi xuống Holding, Holding0 sẽ lưu word cao Holding1 lưu word thấp.
- + Bước 3: viết code
- Kết nối đến thiết bị đang chạy modbus RTU Slave

```
myPLC.ModbusRTUMaster.ResponseTimeout = 1000;
if (myPLC.ModbusRTUMaster.KetNoi("/dev/ttyUSB0", 9600, 8, System.IO.Ports.Parity.None, System.IO.Ports.StopBits.One) == true)
    label3.BackColor = Color.Green;
else
    label3.BackColor = Color.Red;
```

- Các method đọc các vùng nhớ modbus
 - Vùng nhớ Coil

```
if (myPLC.ModbusRTUMaster.ReadCoils(1, 0, 10, ref MangDoc) == true)
```

- Vùng nhớ DiscreteInput

```
if (myPLC.ModbusRTUMaster.ReadDiscreteInputContact(1, 0, 10, ref MangDoc) == true)
```

- Vùng nhớ HoldingRegister

```
if (myPLC.ModbusRTUMaster.ReadHoldingRegisters(1, 0, 10, ref Mang) == true)
```

- Vùng nhớ InputRegister

```
if (myPLC.ModbusRTUMaster.ReadInputRegisters(1, 0, 10, ref Mang) == true)
```

- Các method ghi các vùng nhớ modbus
 - Ghi giá trị kiểu Dword vào mảng byte value, với Word đầu tiên là Word thấp

```
myPLC.SetDWord_LSB(value, 0, Convert.ToInt32(textBox5.Text));
```

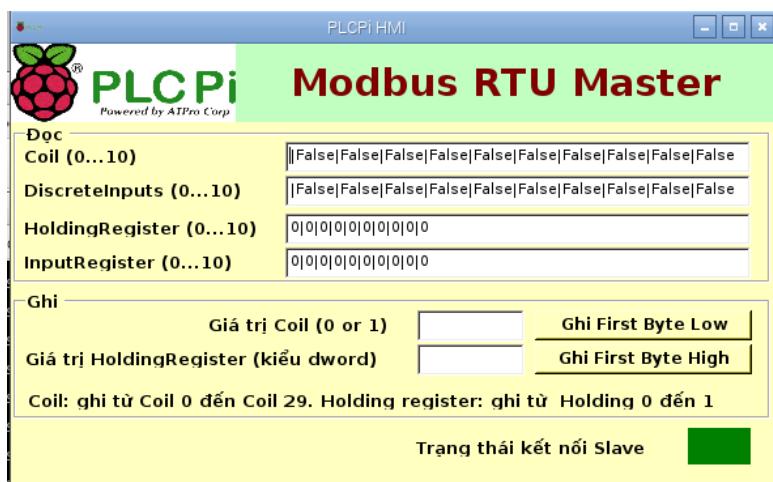
- * Ghi giá trị kiểu Dword vào mảng byte value, với Word đầu tiên là Word cao

```
myPLC.SetDWord(value, 0, Convert.ToInt32(textBox5.Text));
```

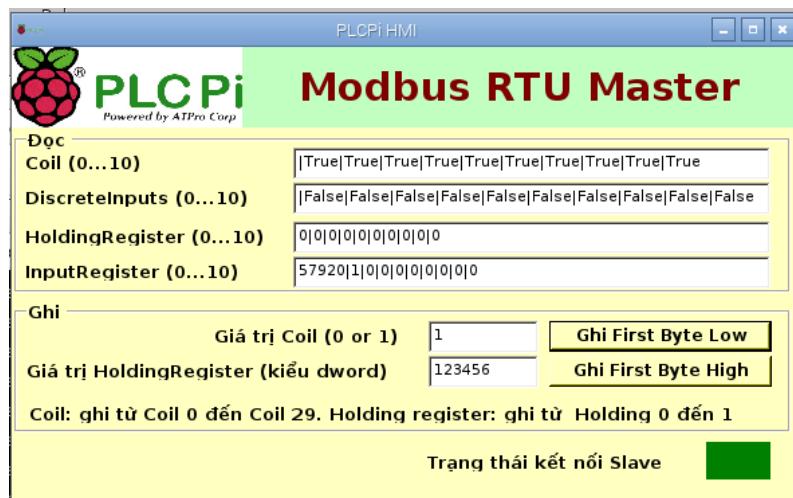
- * Ghi dữ liệu xuống vùng nhớ Coil và HoldingRegister

```
myPLC.ModbusRTUMaster.WriteMultipleCoils(1, 0, 32, Mang1);
Thread.Sleep(1000);
myPLC.ModbusRTUMaster.WriteHoldingRegisters(1, 0, 2, value);
Thread.Sleep(1000);
```

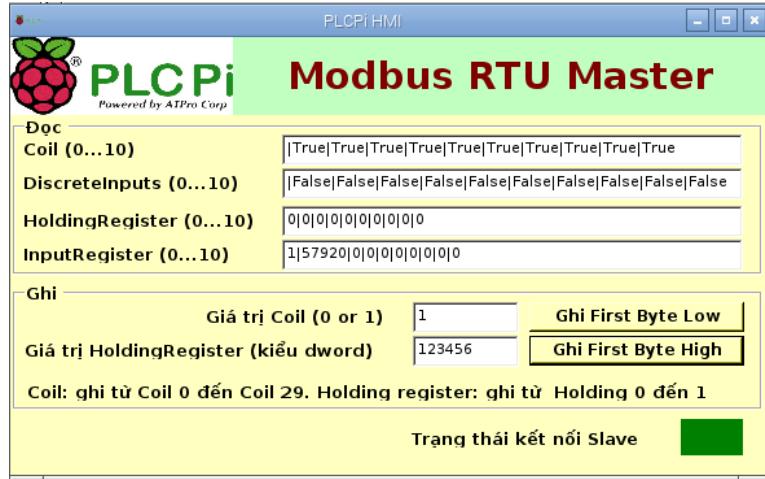
- + Bước 4: sau khi code xong, tiến hành biên dịch và copy sang PLCPi để chạy(xem Ví dụ 1, bước 4 và bước 5, trang92).
- + Kết quả:
 - Giao diện khi mới mở chương trình lên



- Nhập giá trị ghi vào, bấm nút Ghi First Byte Low, được kết quả như hình

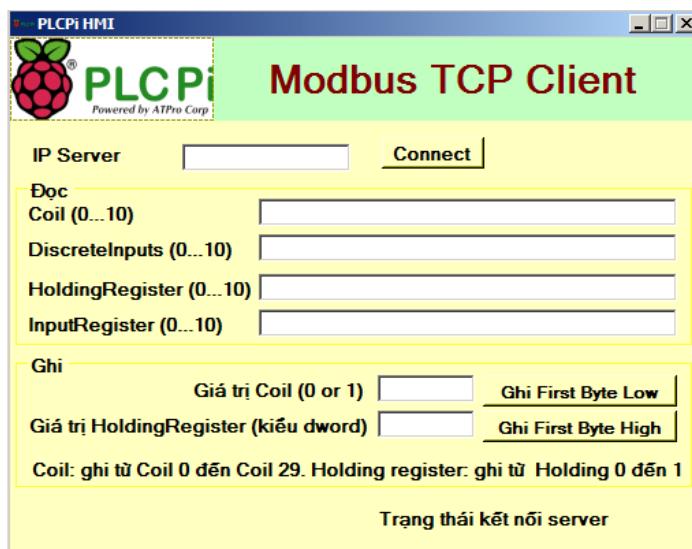


- Bấm nút Ghi First Byte Hight, được kết quả như hình



5.2.4. Ví dụ 4: ModbusTCPClient

- Giới thiệu:
 - + Đây là ví dụ về sử dụng tính năng truyền thông ModbusTCPClient
 - + Tải về tại đây: <http://atscada.com/plcpi/> (mục Examples, Project/Winform, dự án tên: ModbusTCPClient)
- Tính Năng:
 - + Đọc ghi giá trị của các vùng nhớ modbus TCP Server
- Hướng dẫn tạo dự án:
 - + Bước 1: tạo 1 dự án Winform_HMI_PLCPI_App mới như ở Ví dụ 1(trang 89)
 - + Bước 2: thiết kế giao diện thiết kế như hình



- IP Server: nhập địa chỉ IP của server cần kết nối vào
 - Đọc giá trị các vùng nhớ hiển thị lên textbox
 - Ghi giá trị xuống vùng nhớ coil và holding.
 - * Số coil ghi xuống: 30 coil
 - * Số HoldingRegister ghi xuống: 2 holding
 - * Giá trị Coil: nhập 0 hoặc 1(tương ứng true, false)
 - * Giá trị HoldingRegister: nhập vào số dword(4 byte),
 - * Bấm nút Ghi First Byte Low thì sẽ lấy số dword chuyển đổi thành 4byte, rồi ghi xuống Holding, Holding0 sẽ lưu word thấp Holding1 lưu word cao
 - * Bấm nút Ghi First Byte Hight thì sẽ lấy số dword chuyển đổi thành 4byte, rồi ghi xuống Holding, Holding0 sẽ lưu word cao Holding1 lưu word thấp.
- + Bước 3: viết code

- Kết nối đến thiết bị đang chạy modbusTCP Server

```
myPLC.ModbusTCPClient.KetNoi(textBox1.Text, 502);
if (myPLC.ModbusTCPClient.connected == true)
{
    label3.BackColor = Color.Green;
    timer1.Enabled = true;
}
else
    label3.BackColor = Color.Red;
```

- Các method đọc các vùng nhớ modbus

- * Vùng nhớ Coil

```
myPLC.ModbusTCPClient.ReadCoils(0, 0, 10, ref Mang);
```

- * Vùng nhớ DiscreteInput

```
myPLC.ModbusTCPClient.ReadDiscreteInputs(0, 0, 10, ref Mang);
```

- * Vùng nhớ HoldingRegister

```
myPLC.ModbusTCPClient.ReadHoldingRegister(0, 0, 10, ref Mang);
```

- * Vùng nhớ InputRegister

```
myPLC.ModbusTCPClient.ReadInputRegister(0, 0, 10, ref Mang);
```

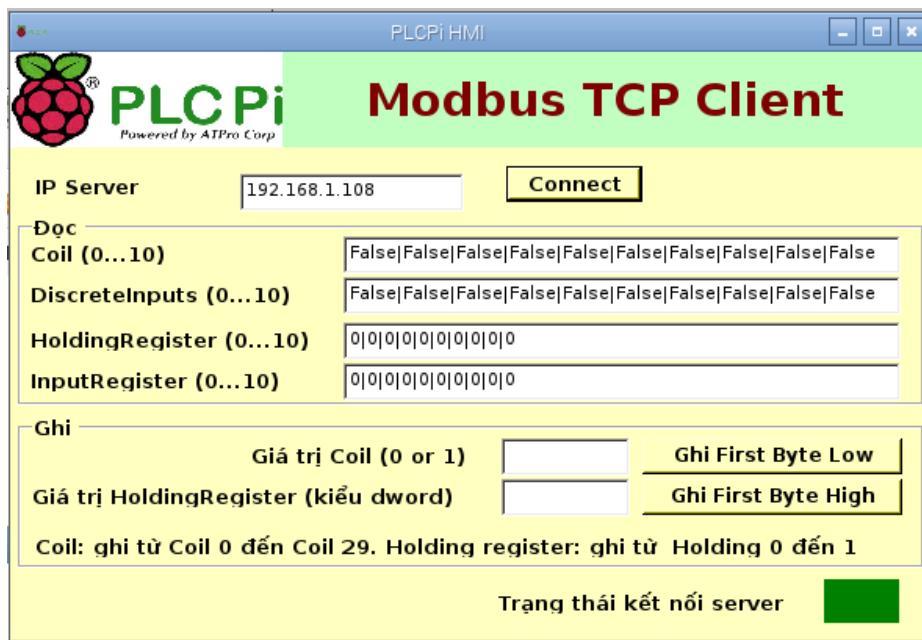
- Các method ghi các vùng nhớ modbus
 - * Ghi giá trị kiểu Dword vào mảng byte value, với Word đầu tiên là Word thấp


```
myPLC.SetDWord_LSB(value3, 0, Convert.ToInt32(textBox5.Text));
```
 - * Ghi giá trị kiểu Dword vào mảng byte value, với Word đầu tiên là Word cao

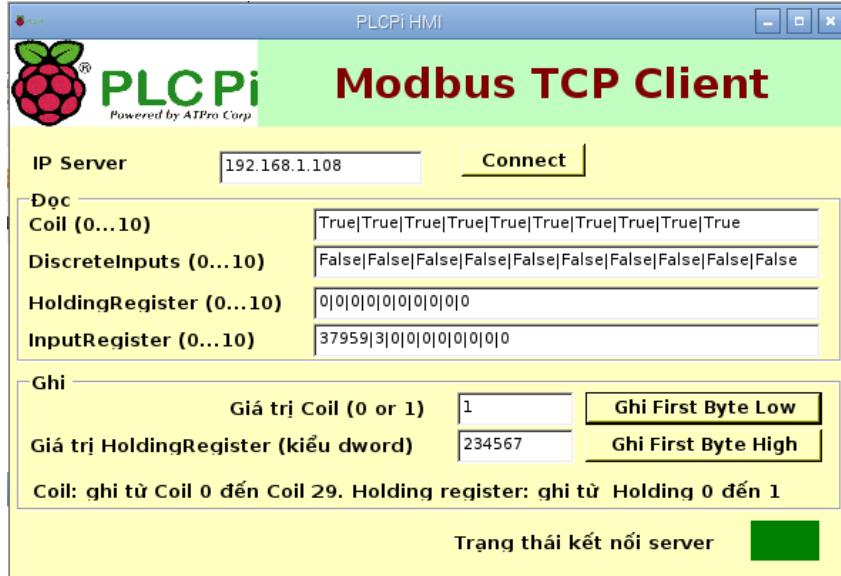

```
myPLC.SetDWord(value3, 0, Convert.ToInt32(textBox5.Text));
```
 - * Ghi dữ liệu xuống vùng nhớ Coil và HoldingRegister


```
myPLC.ModbusTCPClient.WriteMultipleCoils(0, 0, 29, value, ref Mang);
myPLC.ModbusTCPClient.WriteMultipleRegister(0, 0, value3, ref Mang);
```

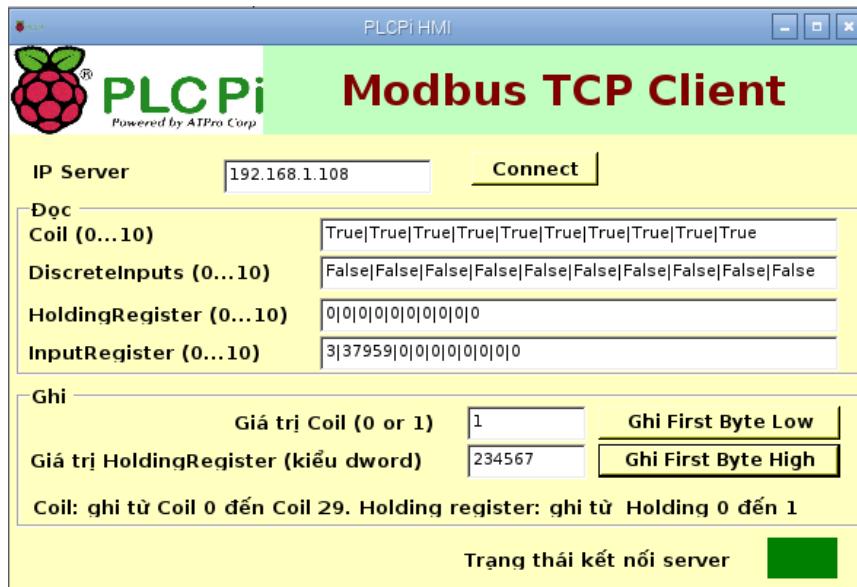
- + Bước 4: sau khi code xong, tiến hành biên dịch và copy sang PLCPi để chạy(xem Ví dụ 1, bước 4 và bước 5, trang92).
- + Kết quả:
 - Giao diện khi mới mở chương trình lên



- Nhập giá trị ghi vào, bấm nút Ghi First Byte Low, được kết quả như hình



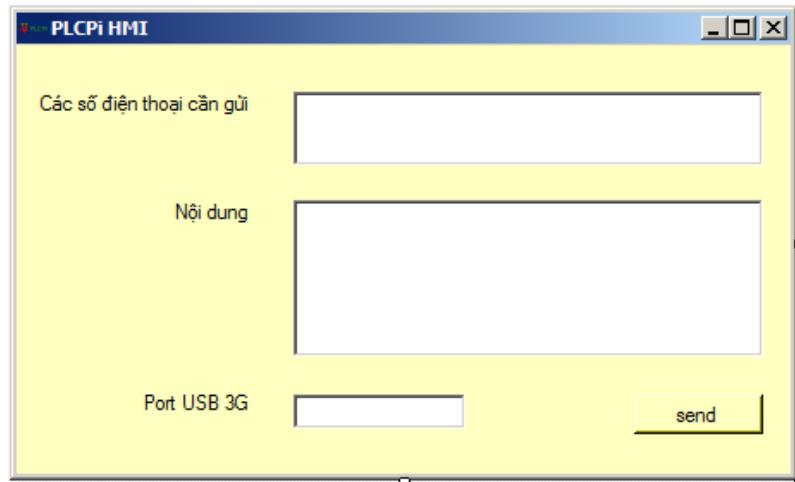
- Bấm nút Ghi First Byte Hight, được kết quả như hình



5.2.5. Ví dụ 5: Gửi SMS

- Giới thiệu:
 - + Sử dụng GSM modem trong USB 3G để gửi sms đến các số điện thoại
 - + Tải về tại đây: <http://atscada.com/plcpi/> (mục Examples, Project/Winform, dự án tên SendSMS)
- Tính Năng:
 - + Gửi sms đến 1 hoặc nhiều số điện thoại
- Hướng dẫn tạo dự án:

- + Bước 1: tạo 1 dự án Winform_HMI_PLCPi_App mới như ở Ví dụ 1(trang 89)
- + Bước 2: thiết kế giao diện thiết kế như hình

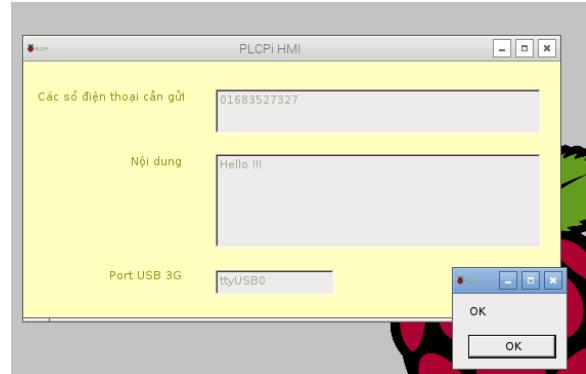


- Nhập các số điện thoại và nội dung cần gửi rồi bấm nút send để gửi
- Port USB 3G để hiển thị port serial của usb 3g, và ta có thể thay đổi giá trị của Port thông qua Textbox này
- + Bước 3: viết code
 - Kiểm tra xem có thay đổi Port_USB3G hay không

```
if (textBox3.Text != myPLC.SMS.Port_USB3G)
{
    myPLC.SMS.Port_USB3G = textBox3.Text;
}
```

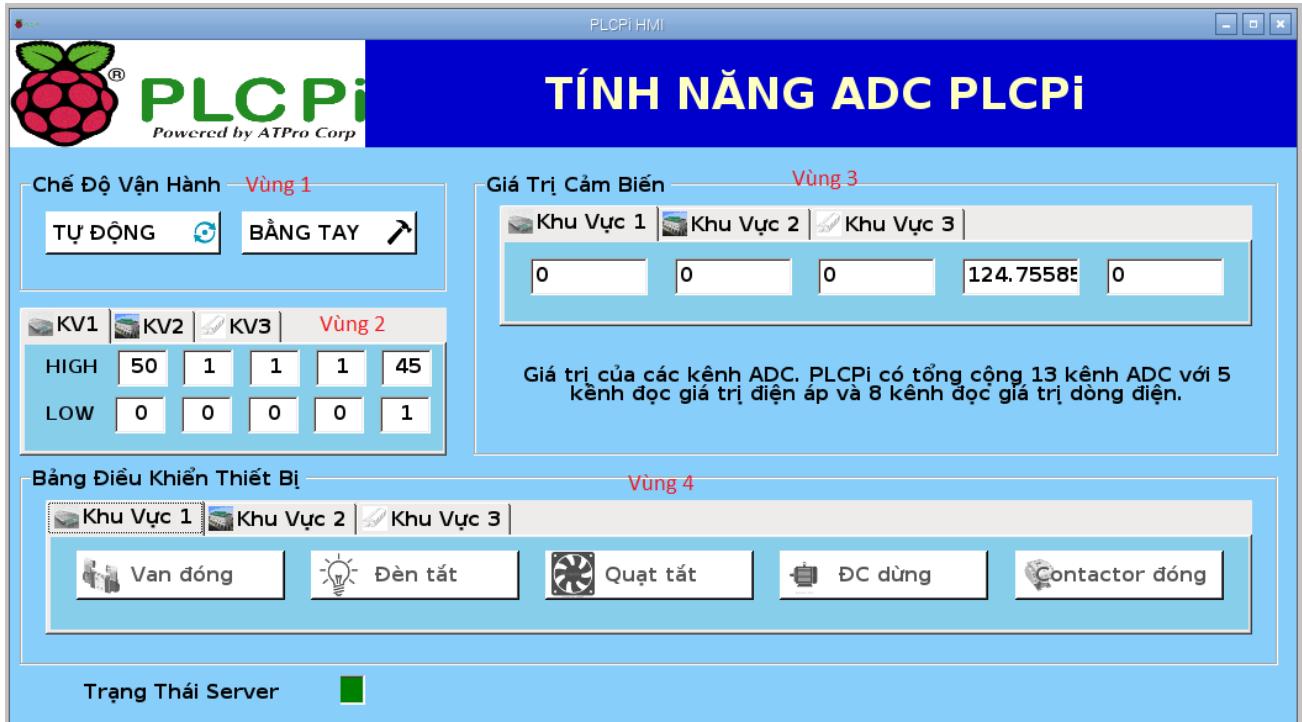
- Gọi method gửi sms


```
MessageBox.Show(myPLC.SMS.GuiSMS(textBox1.Text, textBox2.Text));
```
- + Bước 4: sau khi code xong, tiến hành biên dịch và copy sang PLCPi để chạy(xem Ví dụ 1, bước 4 và bước 5, trang92).
- + Kết quả:khi bấm nút send thì sẽ xuất hiện 1 messagebox thông báo trạng thái gửi, OK là gửi thành công, BAD là thất bại



5.2.6. Ví dụ 6: Dự án DKGSNhaXuong:

- Giới thiệu:
 - + Dự án này sử dụng các ngõ vào AI để giám sát các thiết bị trong nhà xưởng thông qua các cảm biến, và điều khiển các thiết bị trong nhà xưởng theo các mức ngưỡng mà ta cài đặt
 - + Tải về tại đây: <http://atscada.com/plcpi/> (mục Examples, Project/Winform, dự án tên DKGSNhaXuong)
- Tính Năng:
 - + Project này có 2 chế độ vận hành: Tự động và Bằng tay:
 - + Tự động: đọc giá trị cảm biến từ các khu vực xưởng khác nhau. So sánh với giá trị ngưỡng đã được cài đặt trước. Nếu vượt ngưỡng giới hạn thì điều khiển các thiết bị trong xưởng bật hoặc tắt
 - + Bằng tay: người dùng trực tiếp bật hoặc tắt các thiết bị thông qua Winform_HMI
 - + Lưu trữ dữ liệu trên S7Ethernet để truyền thông
- Hướng dẫn tạo dự án:
 - + Bước 1: tạo 1 dự án Winform_HMI_PLCPi_App mới như ở Ví dụ 1(trang 89)
 - + Bước 2: thiết kế giao diện



- Vùng 1: chọn chế độ điều khiển các thiết bị trong xưởng
 - Vùng 2: cài đặt các mức ngưỡng để khi ở chế độ tự động, nó sẽ so sánh các giá trị thực với giá trị ngưỡng để tự động điều khiển các thiết bị trong xưởng
 - Vùng 3: giá trị Analog đọc về từ ngõ vào AI của PLCPI. Ví dụ như: tốc độ động cơ, nhiệt độ, áp suất, ...
 - Vùng 4: bảng điều khiển các thiết bị trong nhà xưởng khi ở chế độ bằng tay
- + Bước 3: viết code
- Đọc ngõ vào AI. Đọc 1 lần 13 kênh của ngõ vào AI, trả về mảng 13 giá trị ADC

```
dulieu = myPLC.AI.DocAI();
```

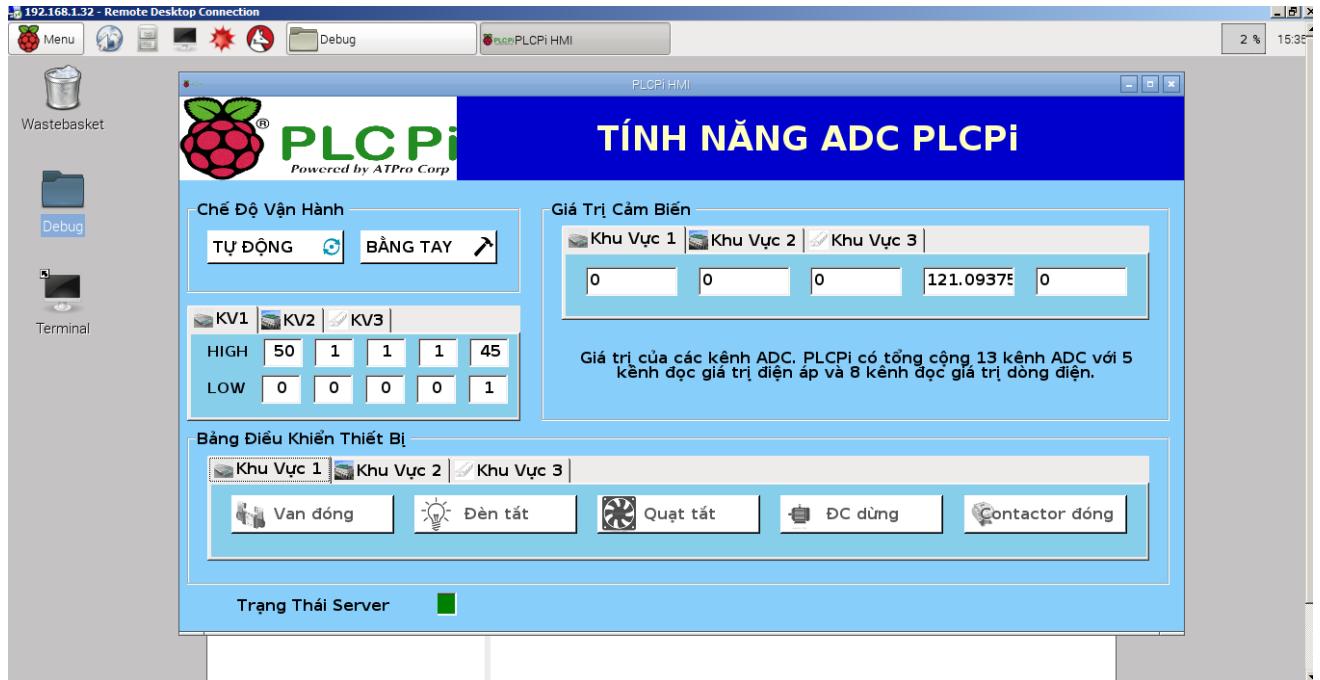
- Cập nhật các giá trị ADC vào vùng nhớ DataBlock của PLCPI để truyền thông

```
if (i == 0)
    myPLC.SetInt(myPLC.S7Ethernet.Server.DataBlock, 0, dulieu1);
else if (i == 1)
    myPLC.SetInt(myPLC.S7Ethernet.Server.DataBlock, 2, dulieu1);
```

- Dùng method Scale để chuyển đổi các giá trị ADC về lại giá trị Analog ban đầu

```
ADC_Scale[0] = Convert.ToDouble(myPLC.AI.Scale(0, 0, 1024, 100, Convert.ToDouble(dulieu[0])));
ADC_Scale[1] = Convert.ToDouble(myPLC.AI.Scale(0, 0, 1024, 200, Convert.ToDouble(dulieu[1])));
```

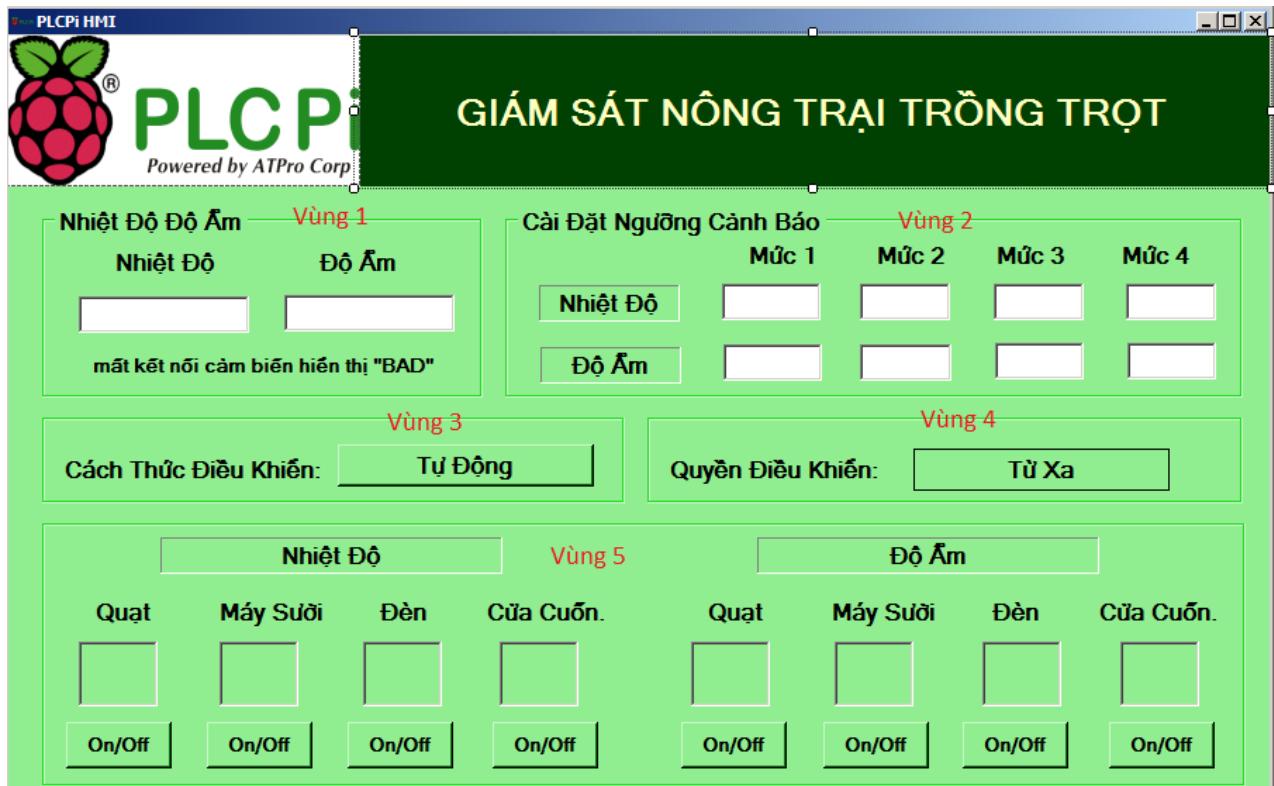
- + Bước 4: sau khi code xong, tiến hành biên dịch và copy sang PLCPi để chạy(xem Ví dụ 1, bước 4 và bước 5, trang92).
- + Kết quả:



5.2.7. Ví dụ 7: TrangTrai

- Giới thiệu:
 - + Điều khiển giám sát nhiệt độ, độ ẩm của trang trại chăn nuôi
 - + Dự án này sử dụng cảm biến nhiệt độ độ ẩm DHT21 để đo nhiệt độ độ ẩm của trang trại và tự động duy trì giá trị nhiệt độ độ ẩm trong trang trại luôn ở trong mức ngưỡng cho phép
 - + Tải về tại đây: <http://atscada.com/plcpi/> (mục Examples, Project/Winform, dự án tên TrangTrai)
- Tính Năng:
 - + Đọc giá trị nhiệt độ độ ẩm từ cảm biến DHT21 hiển thị lên HMI trên PLCPi, và hiển thị ra module hiển thị
 - + Project có 2 chế độ điều khiển: từ xa và tại chỗ
 - + Chế độ từ xa cho phép người dùng điều khiển trực tiếp trên giao diện HMI của PLCPi. Có 2 chế độ điều khiển: tự động và bằng tay.
 - Chế độ tự động: cài đặt các ngưỡng giá trị, khi vượt mức ngưỡng thì các thiết bị sẽ tự động được mở. Người dùng không cần phải thao tác điều khiển trên màn hình HMI.

- Chế độ bằng tay: Cho phép người dùng sử dụng các nút nhấn được thiết kế trên giao diện HMI để bật tắt thiết bị từ xa.
- + Chế độ tại chỗ. Người dùng chỉ được phép thao tác trực tiếp với các thiết bị tại nơi giám sát.
- Hướng dẫn tạo dự án:
 - + Bước 1: tạo 1 dự án Winform_HMI_PLCPi_App mới như ở Ví dụ 1(trang 89)
 - + Bước 2: thiết kế giao diện



- Vùng 1: giá trị nhiệt độ độ ẩm thực
- Vùng 2: cài đặt các mức ngưỡng của nhiệt độ, độ ẩm, dùng trong chế độ tự động
- Vùng 3: chọn chế độ điều khiển tự động hoặc bằng tay
- Vùng 4: chọn quyền điều khiển tại chỗ hoặc từ xa
- Vùng 5: hiển thị trạng thái của các thiết bị, và các button để bật tắt thiết bị khi ở chế độ bằng tay
- + Bước 3: viết code
 - Ghi các giá trị ngưỡng vào vùng nhớ DataBlock của PLCPi, lúc chạy chương trình lần đầu tiên

```

myPLC.SetInt(myPLC.S7Ethernet.Server.DataBlock, 4, Convert.ToInt16(offsetarray[0]));
myPLC.SetInt(myPLC.S7Ethernet.Server.DataBlock, 8, Convert.ToInt16(offsetarray[1]));
myPLC.SetInt(myPLC.S7Ethernet.Server.DataBlock, 12, Convert.ToInt16(offsetarray[2]));

```

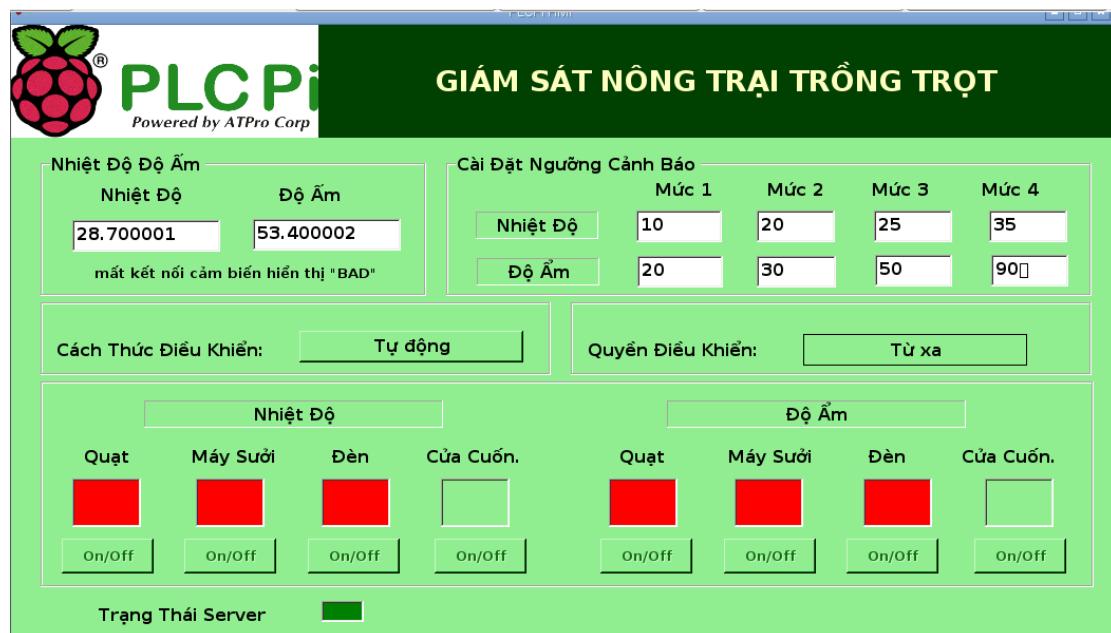
- Xử lý các chế độ làm việc

```

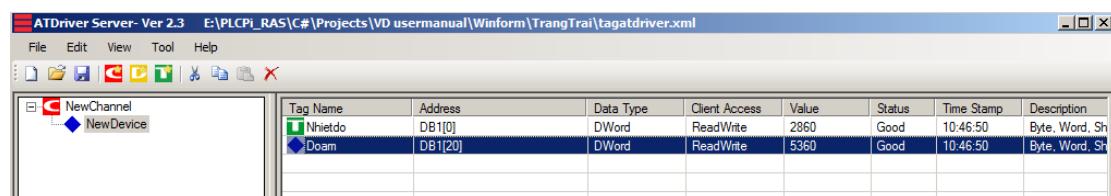
dieukhien = myPLC.NgoVao.DocNgoVao("I0");
chedohatdong = myPLC.GetBoolAt(myPLC.S7Ethernet.Server.DataBlock, 40, 0);

```

- * Chọn quyền điều khiển từ xa hoặc tại chỗ, sẽ đọc từ ngõ vào I0 để chọn. I0.0 = 0 chế độ từ xa, I0.0 = 1 chế độ tại chỗ
- * Chọn chế độ hoạt động là tự động hoặc bằng tay, sẽ đọc vùng nhớ DB1[40] theo bit để chọn. Ở đây sẽ đọc bit 0 của byte 40 trong vùng nhớ DB_Server, dùng method GetBoolAt
- + Bước 4: sau khi code xong, tiến hành biên dịch và copy sang PLCPi để chạy(xem Ví dụ 1, bước 4 và bước 5, trang92).
- + Kết quả:



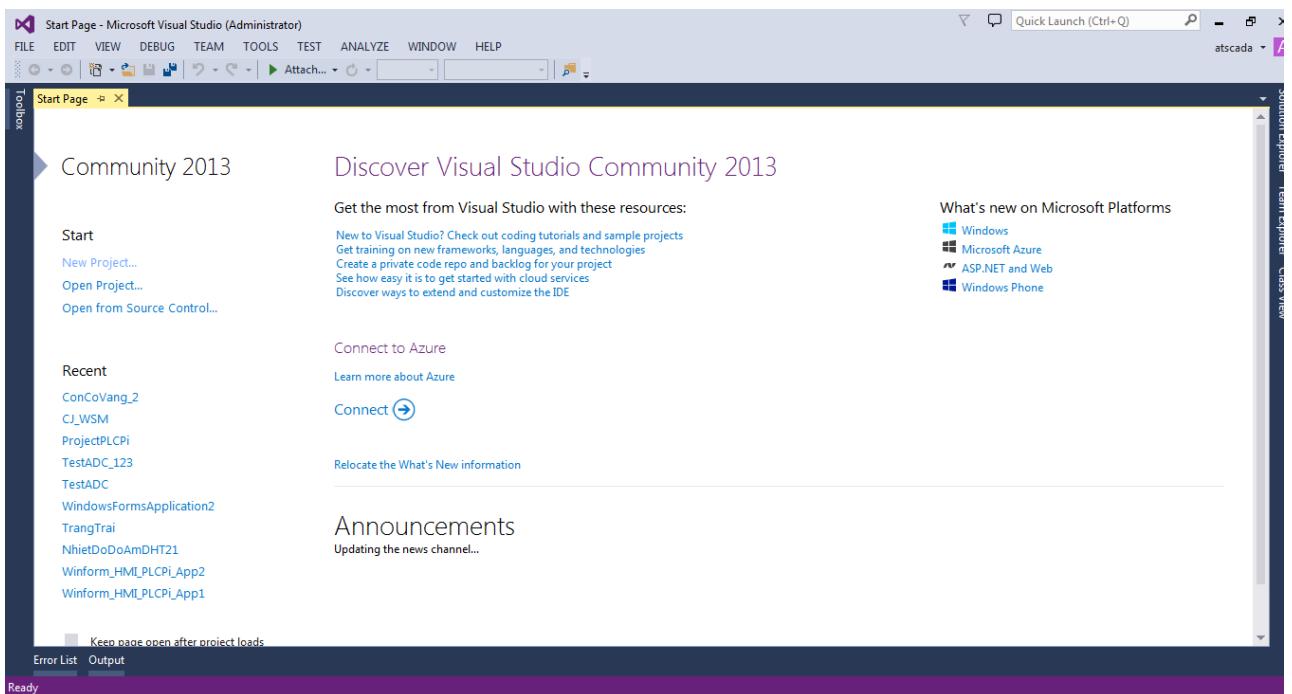
- Dùng ATDriver làm S7Ethernet Client kết nối tới PLCPi để lấy dữ liệu. Ta được như hình



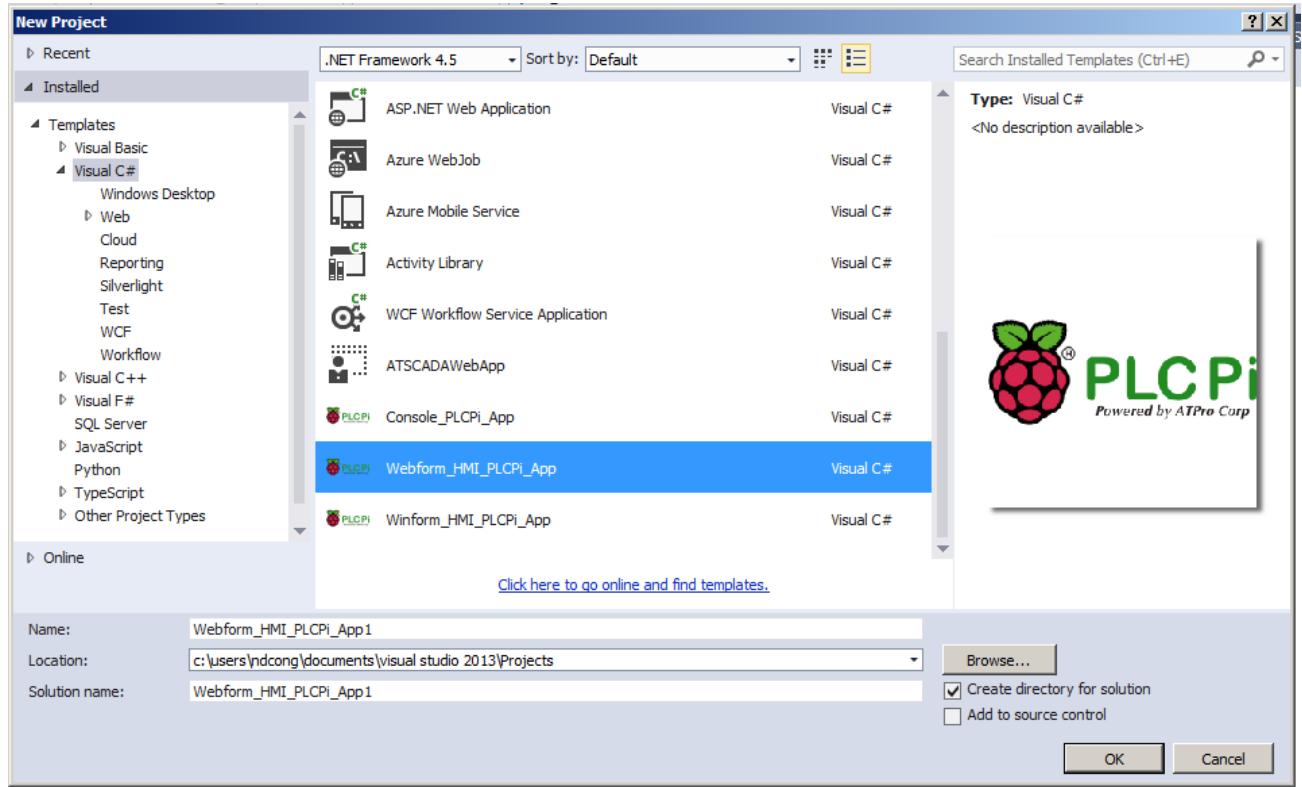
5.3. Hướng dẫn tạo dự án Webform_HMI_PLCPi:

5.3.1. Ví dụ1: Webform_HMI_PLCPi_App1

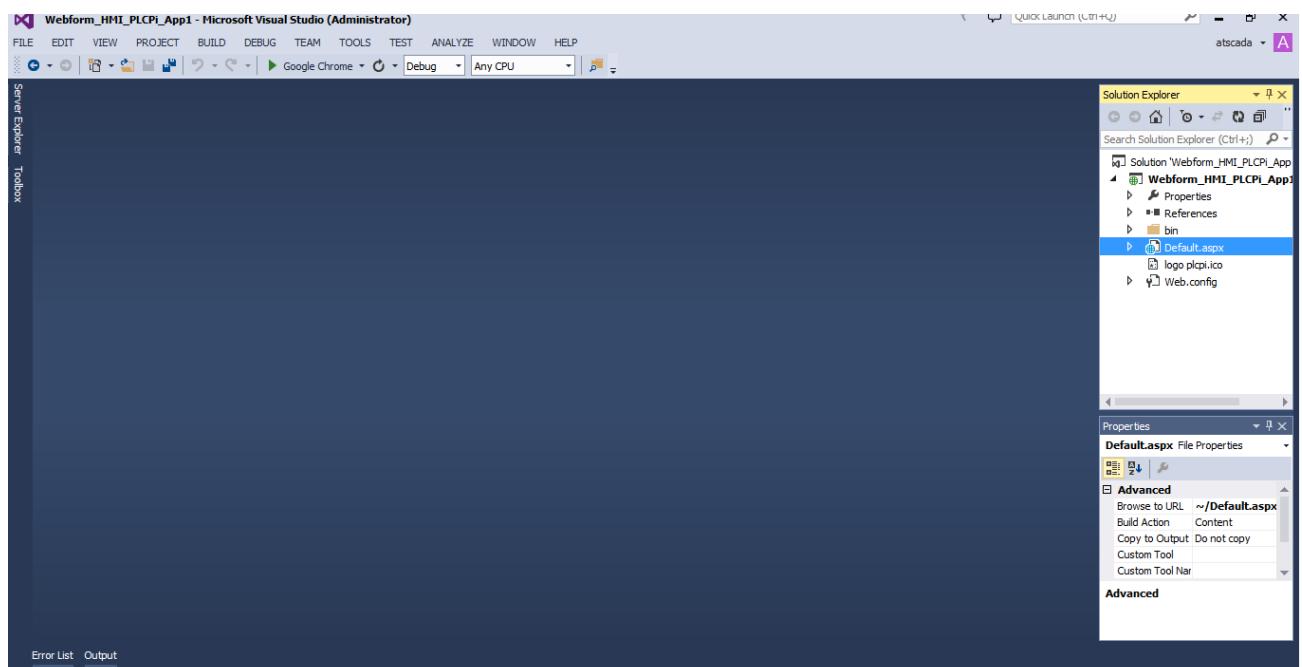
- Giới thiệu:
 - + Tạo dự án với giao diện Webform_HMI trên PLCPi
 - + Tải về tại đây: <http://atscada.com/plcpi/> (mục Examples, Project, dự án tên Webform_HMI_PLCPi_App1)
- Tính Năng:
 - + Đọc giá trị ngõ vào I0 hiển thị lên màn hình
 - + Xuất ngõ ra Q5 = 255 hoặc 0
 - + Đọc giá trị ngõ ra Q5 hiển thị lên màn hình
- Hướng dẫn tạo dự án:
 - + Bước 1: tạo 1 dự án Webform_HMI_PLCPi_App
 - Khởi động Visual Studio 2013



- Chọn New Project, lúc này xuất hiện giao diện như hình

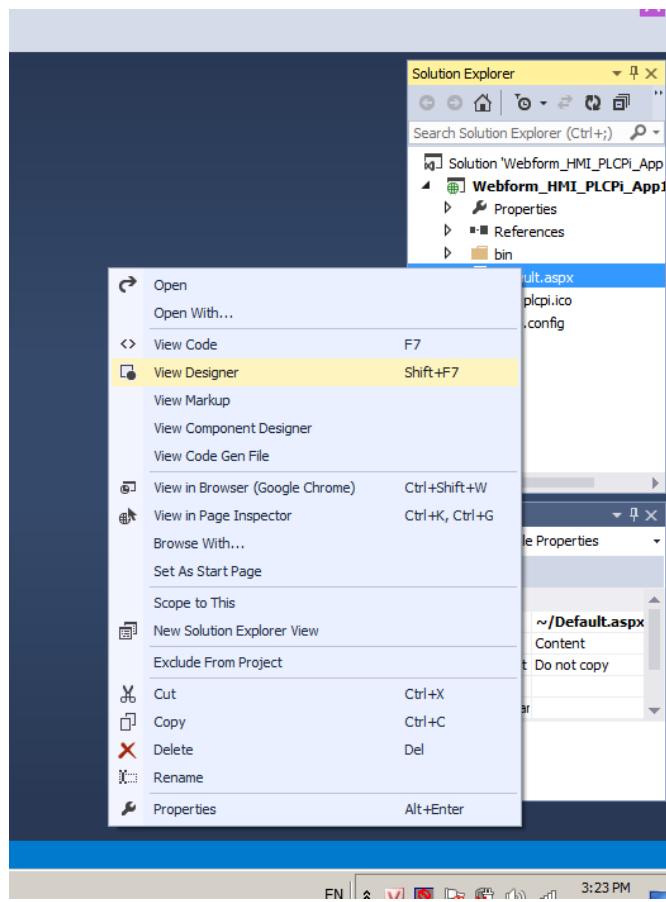


- Chọn Webform_HMI_PLCPi_App để tạo 1 dự án mới với giao diện Webform, đặt tên cho dự án và chọn vị trí lưu, xong rồi bấm OK. Lúc này sẽ xuất hiện giao diện:

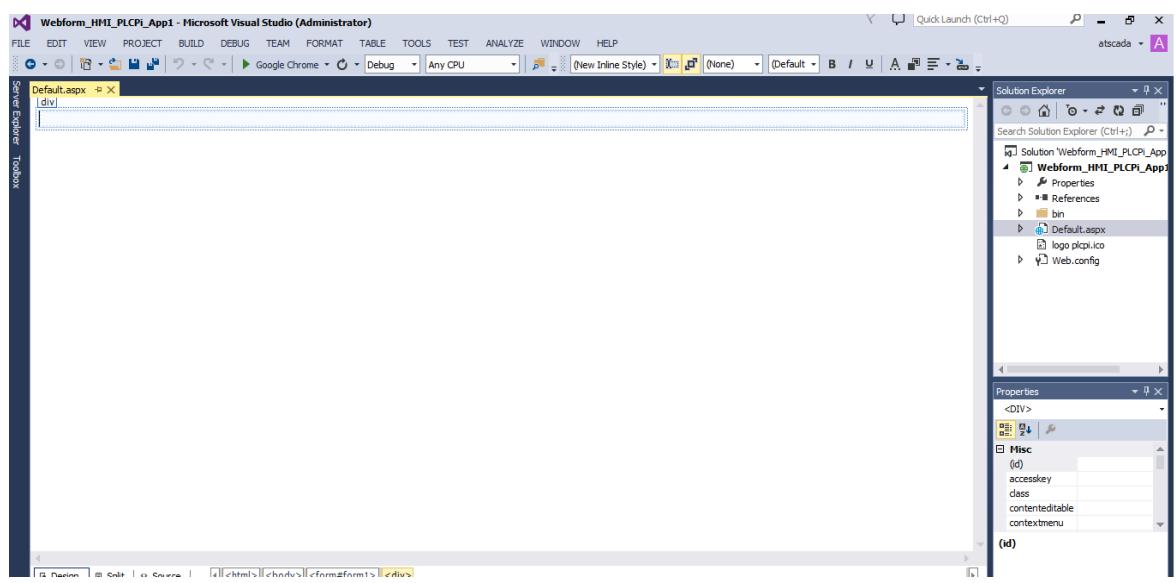


+ Bước 2: thiết kế giao diện

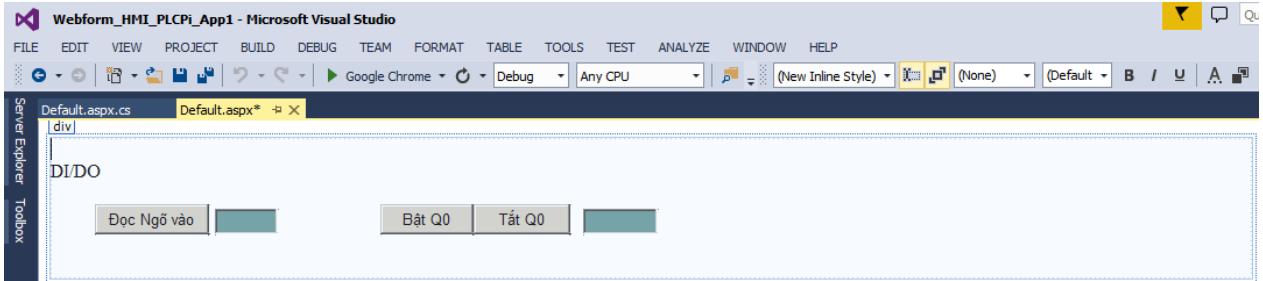
- Kích chuột phải vào Default.aspx chọn View Design



- Xuất hiện màn hình thiết kế:

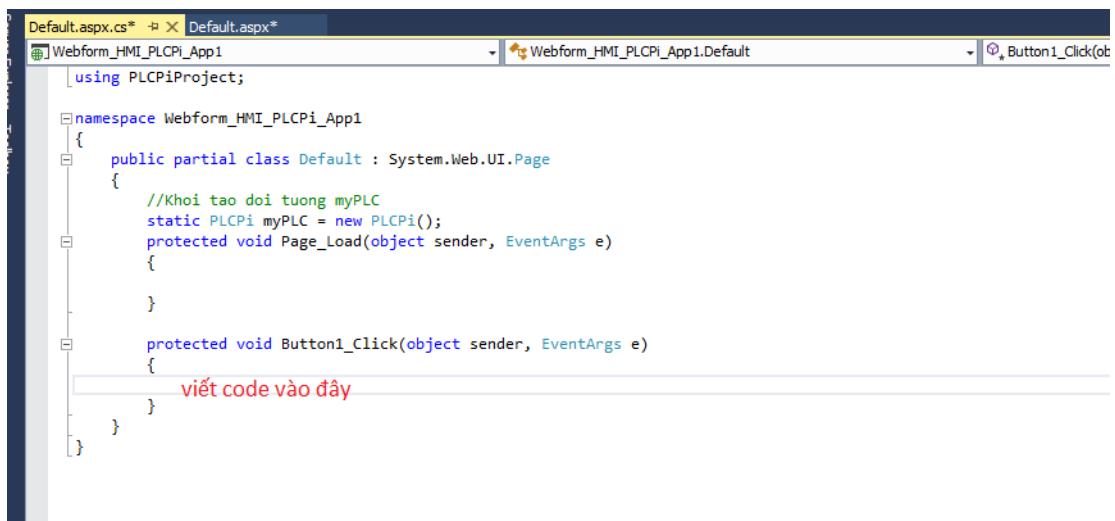


- Ta tiến hành thiết kế giao diện cho ứng dụng. Từ cửa sổ Toolbox kéo thả các tool vào form thiết kế



+ Bước 3: viết code

- Sau khi thiết kế xong, ta kích đúp chuột vào các nút nhấn để viết code



- Đọc ngõ vào I0 hiển thị lên màn hình

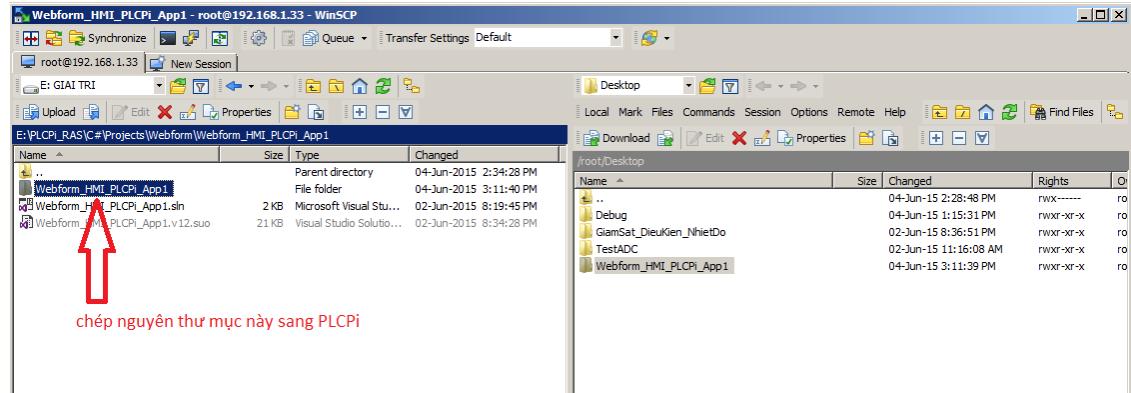
```
TextBox1.Text = myPLC.NgoVao.DocNgoVao("I0").ToString();
```

- Xuất ngõ ra Q0 và đọc trạng thái ngõ ra Q0 hiển thị lên màn hình

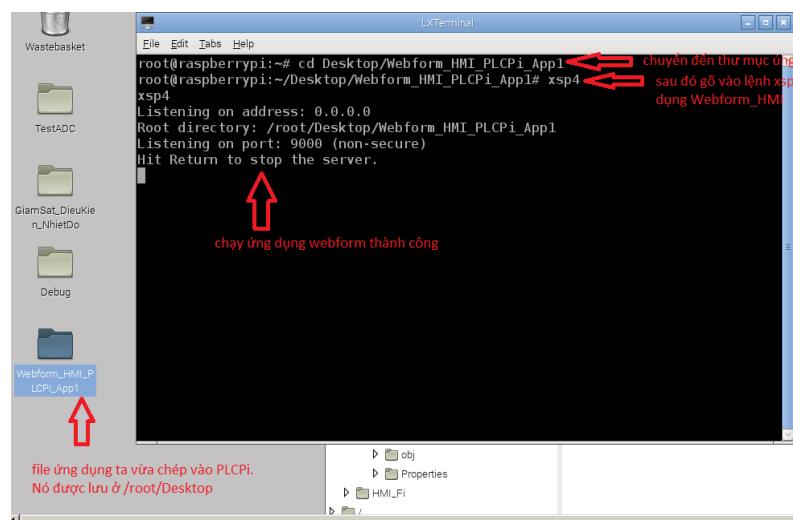
```
//xuất giá trị 255 ra ngõ ra Q5
myPLC.NgoRa.XuatNgoRa("Q0", 255);
//đọc trạng thái ngõ ra Q5 hiển thị lên TextBox3
TextBox3.Text = myPLC.NgoRa.DocNgoRa("Q0").ToString();
```

+ Bước 4:Sau khi code xong ta tiến hành Build. Xong rồi ta dùng phần mềm WinSCP để đưa dữ liệu sang PLCPi và chạy ứng dụng

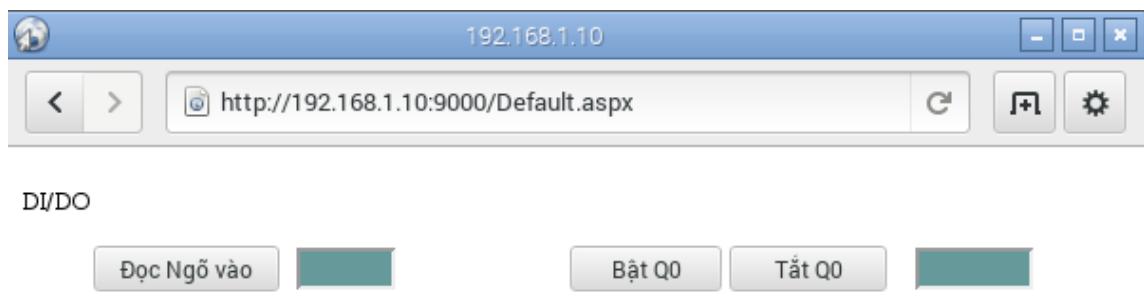
- Đối với ứng dụng Webform_HMI, khi Build xong không có thư mục Debug. Khi ta sao chép dữ liệu sang PLCPi ta phải sao chép nguyên thư mục. Sau khi Build xong ta dùng WinSCP kết nối đến PLCPi, vào thư mục lưu dự án ta vừa làm xong, mở ra và chép nguyên thư mục Webform_HMI_PLCPI_App1 sang PLCPi như hình dưới:



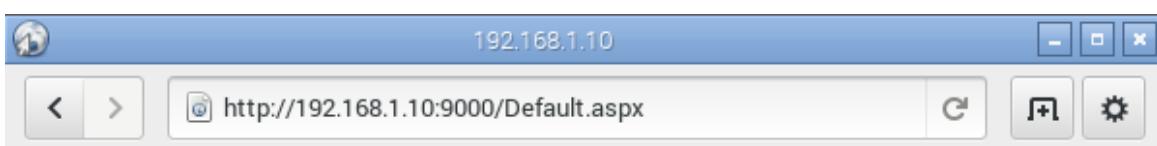
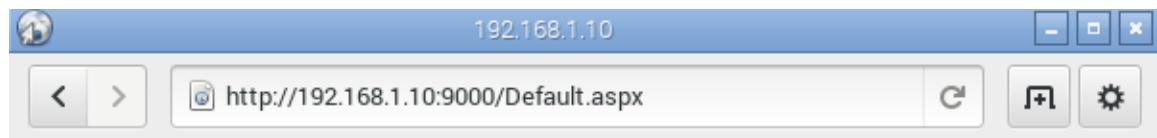
- Bây giờ ta tiến hành chạy ứng dụng
- Dùng Remorte Desktop Connection kết nối đến PLCPi
- Mở ứng dụng LXTerminal và nhập vào các dòng lệnh như hình:



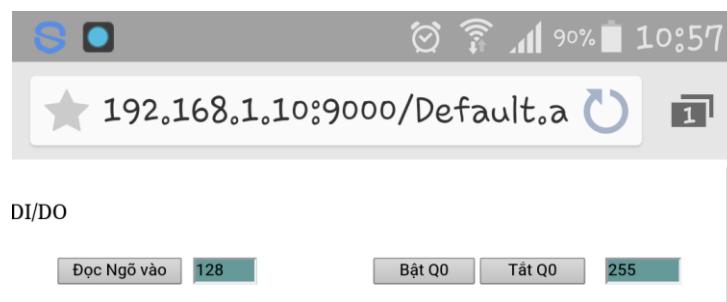
- Sau đó ta mở trình duyệt web lên và nhập vào thanh địa chỉ IP của PLCPi và port 9000. Ví dụ: 192.168.1.33:9000, và enter ta được kết quả



- Bấm button Đọc Ngõ Vào thì sẽ đọc ngõ vào I0 và hiện giá trị lên TextBox. Bấm button Bật Q0, xuất giá trị 255 ra ngõ ra Q0, bấm button Tắt Q0, xuất giá trị 0 ra ngõ ra Q0, đồng thời đọc trạng thái hiện tại của ngõ ra Q0 hiện lên TextBox



- Với ứng dụng Webform_HMI_PLCPi thì ta có thể dùng máy tính, điện thoại thông minh để giám sát điều khiển từ xa. Hình dưới là hình dùng điện thoại để giám sát điều khiển hệ thống từ xa



----- HẾT TẬP 1 -----