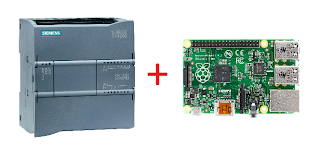
**Raspberry Pi getting data from a S7-1200 PLC**

[](https://4.bp.blogspot.com/-Vl0kCBAc5Dc/VI0h5hnr3sI/AAAAAAAAUYk/y_-miD_SlnE/s1600/S7-1200%2BRaspberry%2BPi.png)

**UPDATE: If you want the raspberry pi to be the s7 server go**[**here**](http://simplyautomationized.blogspot.com/2015/03/raspberry-pi-scada-part-3-communicate.html)  
**UPDATE 2: If you want to see communication with**[**S7-200**](http://amazon.com/s/ref=as_li_bk_tl/?url=search-alias%3Daps&field-keywords=S7-200%20plc&tag=simplyautomat-20&linkId=ffac6de8f8cb400d7393cbf02b4ddd24&linkCode=ktl)**go**[**here**](http://simplyautomationized.blogspot.com/2015/04/raspberry-pi-scada-communitating-with.html)  
**UPDATE 3: Video walkthrough on setup go**[**here**](https://youtu.be/yJNEsI5KJxs)  
  
I recently borrowed a [S7-1200 PLC](http://amazon.com/s/ref=as_li_bk_tl/?url=search-alias%3Daps&field-keywords=S7-1200%20cpu&tag=simplyautomat-20&linkId=cc41829dc993abb47887c98f4e7255e0&linkCode=ktl) from work to see if I could get data from it using a [Raspberry Pi](http://amazon.com/s/ref=as_li_bk_tl/?url=search-alias%3Daps&field-keywords=Raspberry%20Pi&tag=simplyautomat-20&linkId=40d2200989f114b9bf182c172f445b50&linkCode=ktl). In my search for something I found that Snap7 was the best option.

Steps to getting it work

1. Download and compile snap7 (http://sourceforge.net/projects/snap7/files/1.2.1/snap7-full-1.2.1.tar.gz/download)
2. Download and install python library to use snap7 (https://pypi.python.org/pypi/python-snap7)

#download and compile snap7 for rpi

wget http://sourceforge.net/projects/snap7/files/1.2.1/snap7-full-1.2.1.tar.gz/download

tar -zxvf snap7-full-1.2.1.tar.gz

cd snap7-full-1.2.1/build/unix

sudo make –f arm\_v6\_linux.mk all

#copy compiled library to your lib directories

sudo cp ../bin/arm\_v6-linux/libsnap7.so /usr/lib/libsnap7.so

sudo cp ../bin/arm\_v6-linux/libsnap7.so /usr/local/lib/libsnap7.so

#install python pip if you don't have it:

sudo apt-get install python-pip

sudo pip install python-snap7

**You will need to edit the lib\_location on common.py in the /usr/local/lib/python2.7/dist-packages/snap7/ directory**  
**Add a line in the \_\_init\_\_ part of the Snap7Library class:**  
**lib\_location='/usr/local/lib/libsnap7.so'**  
**example below:**

class Snap7Library(object):

"""

Snap7 loader and encapsulator. We make this a singleton to make

sure the library is loaded only once.

"""

\_instance = None

def \_\_new\_\_(cls, \*args, \*\*kwargs):

if not cls.\_instance:

cls.\_instance = object.\_\_new\_\_(cls)

cls.\_instance.lib\_location = None

cls.\_instance.cdll = None

return cls.\_instance

def \_\_init\_\_(self, lib\_location=None):

lib\_location='/usr/local/lib/libsnap7.so' # add this line here

if self.cdll:

return

self.lib\_location = lib\_location or self.lib\_location or find\_library('snap7')

if not self.lib\_location:

msg = "can't find snap7 library. If installed, try running ldconfig"

raise Snap7Exception(msg)

self.cdll = cdll.LoadLibrary(self.lib\_location)

Now you can write your client code :-)  
Here's an example on how to connect and read an output Q0.0:

from time import sleep

import snap7

from snap7.util import \*

import struct

plc = snap7.client.Client()

plc.connect("192.168.12.73",0,1)

area = 0x82 # area for Q memory

start = 0 # location we are going to start the read

length = 1 # length in bytes of the read

bit = 0 # which bit in the Q memory byte we are reading

byte = plc.read\_area(area,0,start,length)

print "Q0.0:",get\_bool(mbyte,0,bit)

plc.disconnect()

I created a helper class on my github here to make the syntax easier for people who are used to DAServer and Ladder:  
<https://github.com/SimplyAutomationized/raspberrypi/raw/master/S7-1200pi/S71200.py>  
Example on how to use it:

import S71200

from time import sleep

import snap7

from snap7.util import \*

import struct

plc = S71200.S71200("192.168.21.65")

plc.writeMem('QX0.0',True) # write Q0.0 to be true, which will only turn on the output if it isn't connected to any rung in your ladder code

print plc.getMem('MX0.1') # read memory bit M0.1

print plc.getMem('IX0.0') # read input bit I0.0

print plc.getMem("FREAL100") # read real from MD100

print plc.getMem("MW20") # read int word from MW20

print plc.getMem("MB24",254) # write to MB24 the value 254

plc.plc.disconnect()

Let me know if there are questions. Hope I can help :-)  
Also let me know if you can help me clean up my S71200.py helper class. I know it looks messy.  
  
**Follow me to get updates on a Raspberry pi Sensor the DA or OPC server can get data using S7 protocol.**  
[+Simply Automationized](https://plus.google.com/117701528311085454325)

**Check out my other SCADA posts**

* [Modbus TCP and Raspberry Pi Temperature Sensor](http://simplyautomationized.blogspot.com/2013/09/raspberry-pi-scada-part-1-modbus.html)
* [Controlling Outputs Using Modbus TCP](http://simplyautomationized.blogspot.com/2014/07/raspberry-pi-scada-part-2-modbus-pwm.html)

at [9:40 PM](https://simplyautomationized.blogspot.com/2014/12/raspberry-pi-getting-data-from-s7-1200.html)

[Email This](https://www.blogger.com/share-post.g?blogID=999578816195622480&postID=7003851588879408301&target=email)[BlogThis!](https://www.blogger.com/share-post.g?blogID=999578816195622480&postID=7003851588879408301&target=blog)[Share to Twitter](https://www.blogger.com/share-post.g?blogID=999578816195622480&postID=7003851588879408301&target=twitter)[Share to Facebook](https://www.blogger.com/share-post.g?blogID=999578816195622480&postID=7003851588879408301&target=facebook)[Share to Pinterest](https://www.blogger.com/share-post.g?blogID=999578816195622480&postID=7003851588879408301&target=pinterest)

Labels: [Pi](https://simplyautomationized.blogspot.com/search/label/Pi), [PLC](https://simplyautomationized.blogspot.com/search/label/PLC), [SCADA](https://simplyautomationized.blogspot.com/search/label/SCADA)

**26 comments:**

1. 

[**Unknown**](https://www.blogger.com/profile/01509490688528680716)[February 21, 2019 at 1:29 AM](https://simplyautomationized.blogspot.com/2014/12/raspberry-pi-getting-data-from-s7-1200.html?showComment=1550741383269#c7786503230262634455)

sudo make –f arm\_v6\_linux.mk all when I am running this line, I got the error like this  
  
pi@raspberrypi:~/Downloads/snap7-full-1.2.1/build/unix $ sudo make –f arm\_v6\_linux.mk all  
make: \*\*\* No rule to make target '–f'. Stop.  
  
Please help me to overcome that error.thanks in advance.

[Reply](javascript:;)

[Replies](javascript:;)

* 1. 

[**A Life**](https://www.blogger.com/profile/14442162522578916111)[July 3, 2019 at 12:55 PM](https://simplyautomationized.blogspot.com/2014/12/raspberry-pi-getting-data-from-s7-1200.html?showComment=1562183700939#c1716976868530577743)

Hi, Please use the following instead:  
  
sudo make -f arm\_v6\_linux.mk

* 1. 

[**aicomings**](https://www.blogger.com/profile/08841903891354880269)[September 23, 2021 at 5:30 AM](https://simplyautomationized.blogspot.com/2014/12/raspberry-pi-getting-data-from-s7-1200.html?showComment=1632400221161#c6993149115112501646)

the "-f" should be written by yourself.

[**Reply**](javascript:;)

1. 

[**Unknown**](https://www.blogger.com/profile/11944244144022882767)[February 27, 2019 at 7:37 AM](https://simplyautomationized.blogspot.com/2014/12/raspberry-pi-getting-data-from-s7-1200.html?showComment=1551281827219#c4220034026547530432)

Don't copy the line, write yourself the line. (Sorry for my english, is very basic.)

[Reply](javascript:;)

1. 

[**Mani Ram**](https://www.blogger.com/profile/07311419449783918801)[February 27, 2019 at 12:55 PM](https://simplyautomationized.blogspot.com/2014/12/raspberry-pi-getting-data-from-s7-1200.html?showComment=1551300911064#c1873283635354107357)

This comment has been removed by the author.

[Reply](javascript:;)

1. 

[**Mani Ram**](https://www.blogger.com/profile/07311419449783918801)[February 27, 2019 at 12:56 PM](https://simplyautomationized.blogspot.com/2014/12/raspberry-pi-getting-data-from-s7-1200.html?showComment=1551300969194#c6318066253681404900)

I too have the same error for the line "sudo make –f arm\_v6\_linux.mk all"  
Help me to overcome this problem.

[Reply](javascript:;)

[Replies](javascript:;)

* 1. 

[**Nate**](https://www.blogger.com/profile/01982148014185893042)[February 27, 2019 at 1:07 PM](https://simplyautomationized.blogspot.com/2014/12/raspberry-pi-getting-data-from-s7-1200.html?showComment=1551301631939#c9185105824258742063)

you probably are using a Raspberry Pi 3.. which would be  
"sudo make -f arm\_v7\_linux.mk all"

[**Reply**](javascript:;)

1. 

[**Mani Ram**](https://www.blogger.com/profile/07311419449783918801)[February 27, 2019 at 1:08 PM](https://simplyautomationized.blogspot.com/2014/12/raspberry-pi-getting-data-from-s7-1200.html?showComment=1551301721021#c8626930656646423906)

i got it work, thanks :)

[Reply](javascript:;)

1. 

[**Sade**](https://www.blogger.com/profile/15132066461916452819)[March 7, 2019 at 9:51 AM](https://simplyautomationized.blogspot.com/2014/12/raspberry-pi-getting-data-from-s7-1200.html?showComment=1551981072822#c5336595965207512139)

This comment has been removed by the author.

[Reply](javascript:;)

1. 

[**Unknown**](https://www.blogger.com/profile/05961518883658521328)[March 19, 2019 at 3:20 AM](https://simplyautomationized.blogspot.com/2014/12/raspberry-pi-getting-data-from-s7-1200.html?showComment=1552990836830#c5373014806087268982)

Hello. I try fix allow above. But dont success. Can you help me!  
snap7.snap7exceptions.Snap7Exception: can't find snap7 library. If installed, try running ldconfig

[Reply](javascript:;)

1. 

[**Nilanj Patel**](https://www.blogger.com/profile/11766256990145326423)[March 25, 2019 at 2:09 AM](https://simplyautomationized.blogspot.com/2014/12/raspberry-pi-getting-data-from-s7-1200.html?showComment=1553504963710#c4316592222716577641)

i am not able to read address starting from "MW" of S71200 plc, but can read address starting from "MD" why is that so please help me.

[Reply](javascript:;)

1. 

[**houssem bahri**](https://www.blogger.com/profile/12124862887117137741)[April 5, 2019 at 7:48 AM](https://simplyautomationized.blogspot.com/2014/12/raspberry-pi-getting-data-from-s7-1200.html?showComment=1554475739785#c6501870285916651176)

How can i solve this problem  
In [19]: plc.connect("192.168.0.1",0,1)  
---------------------------------------------------------------------------  
Snap7Exception Traceback (most recent call last)  
in ()  
----> 1 plc.connect("192.168.0.1",0,1)  
  
/home/pi/.local/lib/python2.7/site-packages/snap7/client.pyc in f(\*args, \*\*kw)  
23 def f(\*args, \*\*kw):  
24 code = func(\*args, \*\*kw)  
---> 25 check\_error(code, context="client")  
26 return f  
27  
  
/home/pi/.local/lib/python2.7/site-packages/snap7/common.pyc in check\_error(code, context)  
63 error = error\_text(code, context)  
64 logger.error(error)  
---> 65 raise Snap7Exception(error)  
66  
67  
  
Snap7Exception: TCP : Connection timed out

[Reply](javascript:;)

[Replies](javascript:;)

* 1. 

[**kdres1989**](https://www.blogger.com/profile/05849664282145619663)[June 12, 2020 at 10:59 PM](https://simplyautomationized.blogspot.com/2014/12/raspberry-pi-getting-data-from-s7-1200.html?showComment=1592027994863#c1834724484685825002)

si lo lograste solucionar y como gRACIAS

[**Reply**](javascript:;)

1. 

[**Unknown**](https://www.blogger.com/profile/16807446536508909431)[May 3, 2019 at 12:43 AM](https://simplyautomationized.blogspot.com/2014/12/raspberry-pi-getting-data-from-s7-1200.html?showComment=1556869385351#c6098807230844977749)

when i try to install the snap7 lib i get this error i this line $ tar -zxvf snap7-full-1.2.1.tar.gz  
tar (child): snap7-full-1.2.1.tar.gz : open impossible: Aucun fichier ou dossier de ce type  
tar (child): Error is not recoverable: exiting now  
tar: Child returned status 2  
tar: Error is not recoverable: exiting now  
what does it mean ?

[Reply](javascript:;)

1. 

[**Unknown**](https://www.blogger.com/profile/16807446536508909431)[May 3, 2019 at 2:14 AM](https://simplyautomationized.blogspot.com/2014/12/raspberry-pi-getting-data-from-s7-1200.html?showComment=1556874860023#c8161660663391753346)

hey , i installed the lib and when i tried to excute the code i got an error : no module named snap7 .  
how can i fix this ?

[Reply](javascript:;)

1. 

[**jml69**](https://www.blogger.com/profile/12852081530965994028)[May 14, 2019 at 11:35 PM](https://simplyautomationized.blogspot.com/2014/12/raspberry-pi-getting-data-from-s7-1200.html?showComment=1557902107186#c7502318212368196966)

in case of [tar -zxvf snap7-full-1.2.1.tar.gz] it was not working for me. Then found out the downloaded filename was [download.tar.gz], i needed to use [tar -zxvf download.tar.gz], by using [cd] / [dir] command first check the name of downloaded file, then execute the [tar -zxvf] command.

[Reply](javascript:;)

1. 

[**Souheil Ghribi**](https://www.blogger.com/profile/15180680909131615633)[July 6, 2019 at 4:20 AM](https://simplyautomationized.blogspot.com/2014/12/raspberry-pi-getting-data-from-s7-1200.html?showComment=1562412037908#c7877906385627851906)

How can i fix tis  
File "plc2.py", line 1, in  
import S71200  
File "/home/pi/s7/S71200.py", line 110, in  
plc = S71200('192.168.1.10') #,debug=True)  
File "/home/pi/s7/S71200.py", line 16, in \_\_init\_\_  
self.plc.connect(ip,0,1)  
File "/usr/local/lib/python2.7/dist-packages/snap7/client.py", line 25, in f  
check\_error(code, context="client")  
File "/usr/local/lib/python2.7/dist-packages/snap7/common.py", line 66, in check\_error  
raise Snap7Exception(error)  
snap7.snap7exceptions.Snap7Exception: TCP : Unreachable peer

[Reply](javascript:;)

1. 

[**Unknown**](https://www.blogger.com/profile/01949725735598446489)[August 14, 2019 at 3:07 AM](https://simplyautomationized.blogspot.com/2014/12/raspberry-pi-getting-data-from-s7-1200.html?showComment=1565777225949#c3083368603536817479)

i have 2 questions :  
first one : how can we find the IP address that you wrote in your code [plc.connect("192.168...")] ?  
and what is that IP address exactly ?  
second one : do we connent raspberry to plc via Ethernet ?  
Thanks

[Reply](javascript:;)

[Replies](javascript:;)

* 1. 

[**Unknown**](https://www.blogger.com/profile/10350859972947510727)[May 30, 2021 at 1:35 AM](https://simplyautomationized.blogspot.com/2014/12/raspberry-pi-getting-data-from-s7-1200.html?showComment=1622363734428#c6434941582775846839)

The ip address is the ip address of plc... You can set it up with tia portal (siemens automation software).... 2. Yes the snap7 uses ethernet to comunicate with the plc. It connects via "pg interface" (siemens administration protocol)... Of course you can setup an access point and access the plc via WiFi...

[**Reply**](javascript:;)

1. 

[**Unknown**](https://www.blogger.com/profile/10939046893046231491)[September 11, 2019 at 1:49 AM](https://simplyautomationized.blogspot.com/2014/12/raspberry-pi-getting-data-from-s7-1200.html?showComment=1568191753423#c6156263381580625984)

superr!! thank you for this .

[Reply](javascript:;)

1. 

[**Unknown**](https://www.blogger.com/profile/15104700101284067582)[September 13, 2019 at 12:24 AM](https://simplyautomationized.blogspot.com/2014/12/raspberry-pi-getting-data-from-s7-1200.html?showComment=1568359470518#c7593033693633574838)

It'able to get a howto with c++

[Reply](javascript:;)

1. 

[**Unknown**](https://www.blogger.com/profile/11736397345116248135)[October 7, 2020 at 12:42 AM](https://simplyautomationized.blogspot.com/2014/12/raspberry-pi-getting-data-from-s7-1200.html?showComment=1602056526929#c3464228950368056921)

h

[Reply](javascript:;)

1. 

[**Unknown**](https://www.blogger.com/profile/11736397345116248135)[October 7, 2020 at 12:43 AM](https://simplyautomationized.blogspot.com/2014/12/raspberry-pi-getting-data-from-s7-1200.html?showComment=1602056594006#c1481303232162300959)

how to get response ...on db\_write or any write funtion.... that we know value write successfully in DBaddress????

[Reply](javascript:;)

1. 

[**Reachrao**](https://www.blogger.com/profile/00226120145537361064)[November 30, 2020 at 5:00 AM](https://simplyautomationized.blogspot.com/2014/12/raspberry-pi-getting-data-from-s7-1200.html?showComment=1606741256316#c4168253574234440304)

Dear All,  
i am trying to communicate S71200 PLC with RaspberryPI using Python program.  
  
Following are the Pthon program and Error:  
import S71200  
from time import sleep  
import snap7  
from snap7.util import \*  
import struct  
  
plc = S71200.S71200("192.168.43.2")  
#plc.writeMem('QX0.0',True) # write Q0.0 to be true, which will only turn on the output if it isn't connected to any rung in your ladder code  
print plc.writeMem('%DB1.QX4.0',False)# writing value  
print plc.getMem("%DB1.DBX4.0") # read  
print plc.getMem("%DB1.DBX8.0") # read  
print plc.getMem("%DB1.DBW20") # read  
plc.plc.disconnect()  
  
Error:  
>>> %Run Snap7\_Example3.py  
Traceback (most recent call last):  
File "/home/pi/Desktop/Snap7\_Example3.py", line 9  
print plc.writeMem('QX4.0',False)# writing value  
^  
SyntaxError: invalid syntax  
>>>  
  
  
  
In SIEMENS TIA:  
Address is: %DB1.DBX4.0  
%DB1.DBX8.0  
%DB1.DBD.16  
  
  
Please advice

[Reply](javascript:;)

[Replies](javascript:;)

* 1. 

[**Nakul**](https://www.blogger.com/profile/02525801278100475085)[February 4, 2021 at 2:49 AM](https://simplyautomationized.blogspot.com/2014/12/raspberry-pi-getting-data-from-s7-1200.html?showComment=1612435742764#c7638432390993413555)

Hi ReachRao,  
  
It seems you are using Python3. The error here is showing invalid syntax as from Python3 print "abcd" is not supported and data needs to be kept in brackets.  
For e.g. please change to print(plc.writeMem('QX4.0',False)) to resolve this error.

[**Reply**](javascript:;)

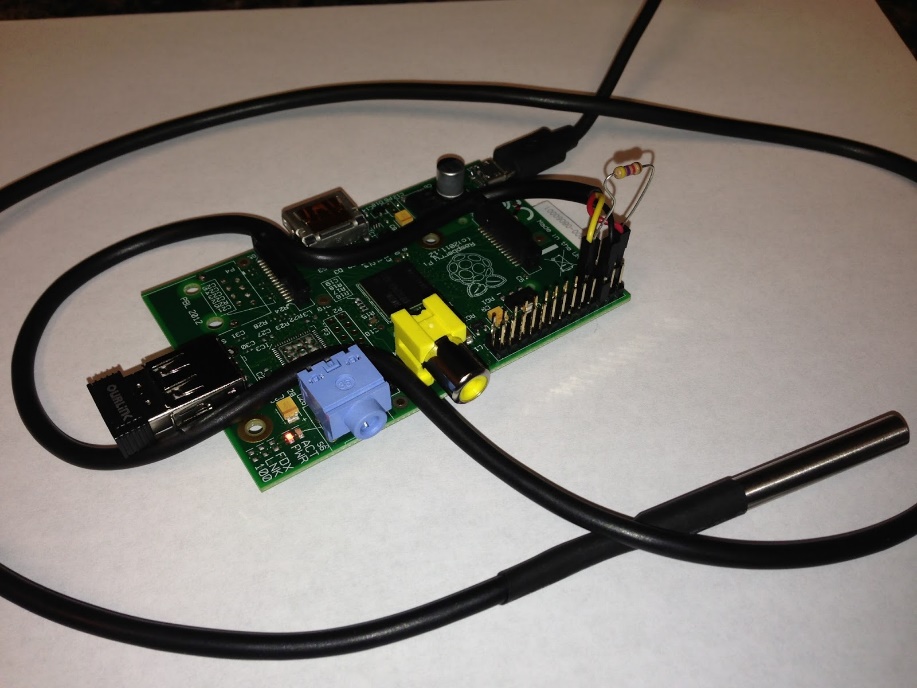
1. 

[**Unknown**](https://www.blogger.com/profile/17430050104570914189)[June 22, 2021 at 12:31 AM](https://simplyautomationized.blogspot.com/2014/12/raspberry-pi-getting-data-from-s7-1200.html?showComment=1624347096981#c8974756160087494057)

Thanks for a very detailed explaination, can you tell me were can i get the area for other plcs? or is it the same for all the S7 seires?

[Reply](javascript:;)

**Raspberry Pi SCADA Part 3: Communicate with the Pi using S7 Protocol.**

[](https://2.bp.blogspot.com/-x2whcNQZay0/UjSJUIXWxbI/AAAAAAAAIo0/fDrfeCa14g4/s1600/IMG_0345%5B1%5D.JPG)

I've been planning on trying this and have been able to get some success. In this example we are going to serve up temperature from the Raspberry Pi using Siemens S7 Protocol. I'm using Wonderware and their DAServer in my tests and will show images of the setup.   
  
For this example in Modbus TCP see [here](http://simplyautomationized.blogspot.com/2013/09/raspberry-pi-scada-part-1-modbus.html)  
To use the Pi to communicate with a PLC see [here](http://simplyautomationized.blogspot.com/2014/12/raspberry-pi-getting-data-from-s7-1200.html)  
  
  
First From a fresh Raspbian Image:  
  
For Raspberry Pi 2 B add the following to the bottom boot/config.txt  
  
#device tree config  
dtoverlay=w1-gpio,gpiopin=4

**Add the following to the /etc/modules**

w1-gpio

w1-therm

**Install python pip:**

sudo apt-get install python-pip

#Download and build the latest snap7 library:

wget http://iweb.dl.sourceforge.net/project/snap7/1.3.0/snap7-full-1.3.0.tar.gz

tar -zxvf snap7-full-1.3.0.tar.gz

cd snap7-full-1.2.1/build/unix

#if you have a Raspberry Pi 2 B use this command to compile:

make –f arm\_v7\_linux.mk

sudo cp ../bin/arm\_v7-linux/libsnap7.so /usr/lib/

sudo ldconfig

#if you have a Raspberry B,B+,A,A+ then use this command to compile:

make –f arm\_v6\_linux.mk all

sudo cp ../bin/arm\_v6-linux/libsnap7.so /usr/lib/

sudo ldconfig

**Download and install python-snap7**

sudo pip install python-snap7

**Now the Pi should be ready!**

We are going to serve up temperature data from a DS18b20 probe connected to the pi. If you are unsure how to connected the probe see [here](https://learn.adafruit.com/adafruits-raspberry-pi-lesson-11-ds18b20-temperature-sensing/hardware)

import time,ctypes

from ctypes import \*

import logging

from threading import Thread

import snap7

import snap7.snap7types

import sys

import os

from time import sleep

logging.basicConfig()

logger = logging.getLogger()

logger.setLevel(logging.INFO)

globalData = (snap7.snap7types.wordlen\_to\_ctypes[snap7.snap7types.S7WLByte]\*128)()

digitalOutputs = (snap7.snap7types.wordlen\_to\_ctypes[snap7.snap7types.S7WLByte]\*16)()

digitalInputs = (snap7.snap7types.wordlen\_to\_ctypes[snap7.snap7types.S7WLByte]\*128)()

server = None

class TempProbe(Thread):

"""

A class for getting the current temp of a DS18B20

"""

def \_\_init\_\_(self, fileName='',tempChangeEvent=None):

Thread.\_\_init\_\_(self)

self.tempDir = '/sys/bus/w1/devices/'

list = os.listdir(self.tempDir)

if(list[0][:2]=="28"):

fileName=list[0]

self.fileName = fileName

self.currentTemp = -999

self.correctionFactor = 1

self.enabled = True

self.oldTemp = 0

self.temperatureChangeEvent=tempChangeEvent

def run(self):

while self.enabled:

try:

f = open(self.tempDir + self.fileName + "/w1\_slave", 'r')

lines = f.readlines()

crcLine = lines[0]

tempLine = lines[1]

result\_list = tempLine.split("=")

temp = float(result\_list[-1])/1000 # temp in Celsius

temp += self.correctionFactor # correction factor

# if you want to convert to Celsius, comment this line

temp = (9.0/5.0)\*temp + 32

if crcLine.find("NO") > -1:

temp = -999

self.currentTemp = temp

if self.currentTemp != self.oldTemp and self.temperatureChangeEvent:

self.oldTemp = self.currentTemp

self.temperatureChangeEvent(self.currentTemp)

except IOError as e:

print "Error: File " + self.tempDir + self.fileName + "/w1\_slave" + " does not exist"

sleep(5)

pass

sleep(1)

# returns the current temp for the probe

def getCurrentTemp(self):

return self.currentTemp

def setEnabled(self, enabled):

self.enabled = enabled

def isEnabled(self):

return self.enabled

temp\_probe=None

def mainloop():

server = snap7.server.Server()

global globaldata

server.register\_area(snap7.snap7types.srvAreaPA, 0, digitalOutputs) # digital outputs

server.register\_area(snap7.snap7types.srvAreaMK, 0, globalData) # internal memory

server.register\_area(snap7.snap7types.srvAreaPE, 0, digitalInputs) # digital inputs

server.start()

global temp\_probe

temp\_probe = TempProbe()

temp\_probe.start() # start getting temperature async

while True:

while True:

event = server.pick\_event()

# write temperature to input memory Input Real 67 (IREAL67)

snap7.util.set\_real(digitalInputs, 67, temp\_probe.getCurrentTemp())

# fires the following when an event happens (connecting clients, read requests...)

if event:

#logger.info(server.event\_text(event))

# print server.event\_text(event)

print temp\_probe.getCurrentTemp()

else:

break

time.sleep(.01)

server.stop()

server.destroy()

temp\_probe.setEnabled(False)

temp\_probe.join()

if \_\_name\_\_ == '\_\_main\_\_':

if len(sys.argv) > 1:

snap7.common.load\_library(sys.argv[1])

mainloop()

temp\_probe.setEnabled(False)

temp\_probe.join()

**Now run it!**

**Setting up your Wonderware and DAserver**

|  |
| --- |
|  |
| Add a S7Cp device |

|  |
| --- |
|  |
| Put in the Ip Address of your Raspberry pi |

|  |
| --- |
|  |
| Create a device group.  I called mine Pi |

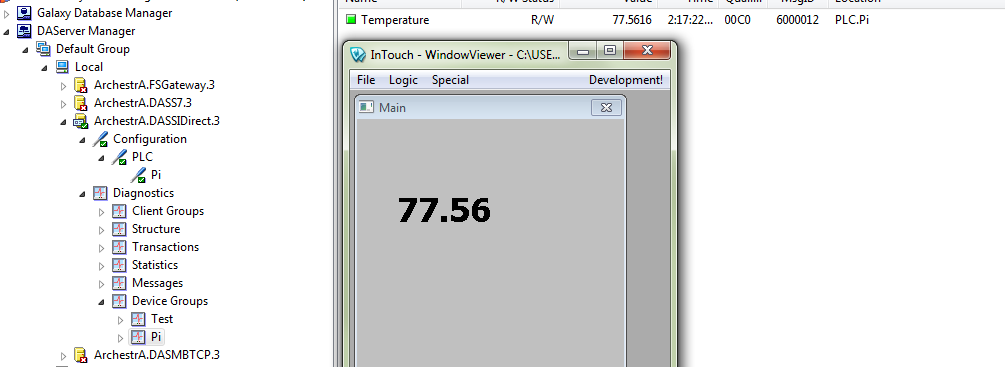
|  |
| --- |
|  |
| Add your device items. In this example we are doing temperature at IREAL67 |

|  |
| --- |
|  |
| Create an Access  Name in your wonderware. I called mine Pi |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
| **Create a tagname and reference it to our Temperature item** |

|  |
| --- |
|  |
| attach your tagname to a text field |

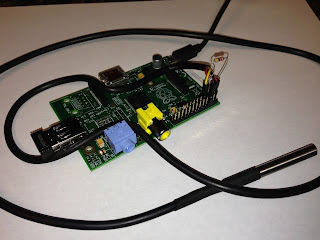
[](https://1.bp.blogspot.com/-Ot3qCaghnTk/VQY98atnKFI/AAAAAAAAWco/gtLB4maZLCE/s1600/result.PNG)

If you wish me to show anymore examples of how to write or read inputs or outputs in the pi using this protocol please let me know.

### Raspberry Pi SCADA Part 1, Modbus Temperature Sensor

# Raspberry Pi SCADA Part 1, Modbus Temperature Sensor.

One great thing about the Pi is that it is so cost effective in some [SCADA](http://en.wikipedia.org/wiki/SCADA) applications. With several different languages to be able to present your data. In the many crazy off the wall things I will do in my series here I will start off with using pymodbus,the DS18b20, and a 4.7k resistor temperature probe to get building temperature. Now there are known security flaws with modbus since it is an open protocol, but with this example you can only read data from the addresses.  This will be in holding the register 0x00 and will need to be scaled by 100 afterwords.

[](https://4.bp.blogspot.com/-x2whcNQZay0/UjSJUIXWxbI/AAAAAAAAIow/OHoulOA1lJ8/s1600/IMG_0345%5B1%5D.JPG)

|  |
| --- |
| My total cost: |
| [Model-A Pi](http://amazon.com/s/ref=as_li_bk_tl/?url=search-alias%3Daps&field-keywords=raspberry%20pi%20model%20a&tag=simplyautomat-20&linkId=ba6b19dedf81bf5470c2da894287ba1a&linkCode=ktl): | $25.00 |
| [Resistors](http://amazon.com/s/ref=as_li_bk_tl/?url=search-alias%3Dindustrial&field-keywords=4.7k%20resistor&tag=simplyautomat-20&linkId=d1242f8b101639b3fae16a01a41cd03b&linkCode=ktl): | .05 |
| [Connectors](http://www.amazon.com/Jumper-Housing-Header-Female-Connector/dp/B00UJ9KXJ0/ref=as_li_bk_tl/?tag=simplyautomat-20&linkId=b3a2f42c0fa21875e214a7ad86cf086c&linkCode=ktl): | .20 |
| [Wifi](http://www.amazon.com/Edimax-EW-7811Un-150Mbps-Raspberry-Supports/dp/B003MTTJOY/ref=as_li_bk_tl/?tag=simplyautomat-20&linkId=4f5aab150c935c6926e6ba0cdfcaf8f5&linkCode=ktl) | $10.00 |
| [Temp Sensor](http://amazon.com/s/ref=as_li_bk_tl/?url=search-alias%3Dindustrial&field-keywords=ds18b20&tag=simplyautomat-20&linkId=159e394a630b788373725e493596bc3f&linkCode=ktl) | $3.00 |
| Total | $38.25 |

### SETUP

get pymodbus and dependencies:

sudo apt-get install python-pymodbus python-twisted-conch

from pymodbus.server.async import StartTcpServer

from pymodbus.device import ModbusDeviceIdentification

from pymodbus.datastore import ModbusSequentialDataBlock

from pymodbus.datastore import ModbusSlaveContext, ModbusServerContext

from pymodbus.transaction import ModbusRtuFramer, ModbusAsciiFramer

from twisted.internet.task import LoopingCall

from threading import Thread

from time import sleep

import os

os.system('modprobe w1-gpio')

os.system('modprobe w1-therm')

temperature =0

class Temp(Thread):

"""

A class for getting the current temp of a DS18B20

"""

def \_\_init\_\_(self, fileName=''):

Thread.\_\_init\_\_(self)

self.tempDir = '/sys/bus/w1/devices/'

list = os.listdir(self.tempDir)

if(list[0][:2]=="28"):

fileName=list[0]

self.fileName = fileName

self.currentTemp = -999

self.correctionFactor = 1;

self.enabled = True

def run(self):

while True:

if self.isEnabled():

try:

f = open(self.tempDir + self.fileName + "/w1\_slave", 'r')

except IOError as e:

print "Error: File " + self.tempDir + self.fileName + "/w1\_slave" + " does$

return;

lines=f.readlines()

crcLine=lines[0]

tempLine=lines[1]

result\_list = tempLine.split("=")

temp = float(result\_list[-1])/1000 # temp in Celcius

temp = temp + self.correctionFactor # correction factor

#if you want to convert to Celcius, comment this line

temp = (9.0/5.0)\*temp + 32

if crcLine.find("NO") > -1:

temp = -999

self.currentTemp = temp

#print "Current: " + str(self.currentTemp) + " " + str(self.fileName)

sleep(1)

#returns the current temp for the probe

def getCurrentTemp(self):

return self.currentTemp

#setter to enable this probe

def setEnabled(self, enabled):

self.enabled = enabled

#getter

def isEnabled(self):

return self.enabled

def updating\_writer(a):

context = a[0]

register = 3

slave\_id = 0x00

address = 0x00

#print pi.getCurrentTemp(),str(int(pi.getCurrentTemp()\*10))

values = [int(pi.getCurrentTemp()\*100)]

context[slave\_id].setValues(register,address,values)

store = ModbusSlaveContext(

di = ModbusSequentialDataBlock(0, [0]\*100),

co = ModbusSequentialDataBlock(0, [0]\*100),

hr = ModbusSequentialDataBlock(0, [0]\*100),

ir = ModbusSequentialDataBlock(0, [0]\*100))

context = ModbusServerContext(slaves=store, single=True)

identity = ModbusDeviceIdentification()

identity.VendorName = 'pymodbus'

identity.ProductCode = 'PM'

identity.VendorUrl = 'http://github.com/bashwork/pymodbus/'

identity.ProductName = 'pymodbus Server'

identity.ModelName = 'pymodbus Server'

identity.MajorMinorRevision = '1.0'

pi = Temp()

pi.start()

time = 5 # 5 seconds delaytime = 5 # 5 seconds delay

loop = LoopingCall(f=updating\_writer, a=(context,))

loop.start(time, now=False) # initially delay by time

StartTcpServer(context, identity=identity, address=("localhost", 502))

#change localhost to your ip address.

Top of Form

Check out how to control Raspberry Pi outputs using this library [here](http://simplyautomationized.blogspot.com/2014/07/raspberry-pi-scada-part-2-modbus-pwm.html)

Bottom of Form

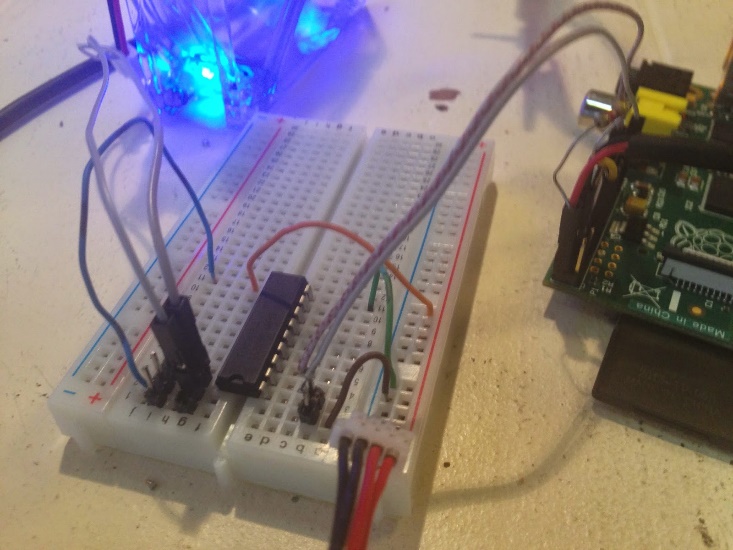
**Raspberry Pi SCADA Part 2, Modbus TCP PWM Controller**

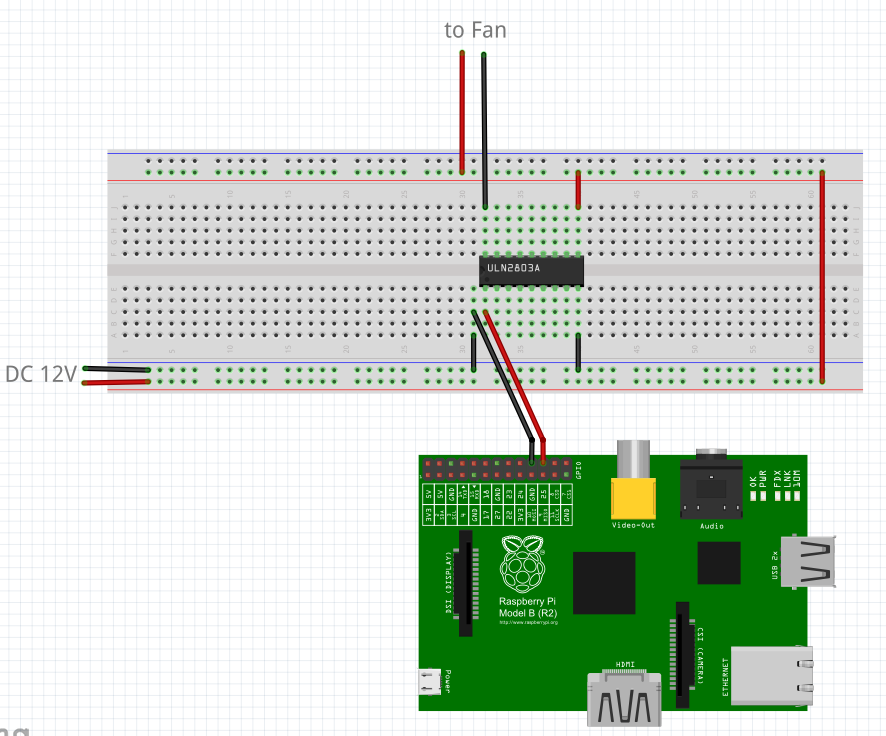
Raspberr Pi SCADA Part 2, Modbus PWM Controller  
  
Since finding a cheap alternative to PLC whilst using an industrial protocol is a popular idea the Raspberry Pi has caught many eyes on doing this.  I [posted](http://simplyautomationized.blogspot.com/2013/09/raspberry-pi-scada-part-1-modbus.html?view=sidebar)once on reading a temperature sensor and serving it up on the Pi using ModbusTCP. This time I expound on it and show you how to control something. In this case it will be a PC 12v fan.  
  
Parts:

* 1 x Raspberry pi
* 1 x Darlington Transistor
* 1 x PC Fan (with about  500ma load)
* 1 x 12v DC Power Supply

Setup:

1. Installing pymodbus and dependencies:
   * sudo apt-get install python-pymodbus python-twisted-conch
2. Wire up your Fan, LED, or motor

[](https://2.bp.blogspot.com/-i9B3xzzvWww/U7oo1XQr5qI/AAAAAAAARNI/wpG0CvX3_-A/s1600/IMG_2095.JPG)

[](https://3.bp.blogspot.com/-XPayS2kehPo/VRoUdJOx6wI/AAAAAAAAWy8/5kgcAGcLFv8/s1600/ModbusTCP_Fan_test.PNG)

1. View my code from [Github](https://github.com/SimplyAutomationized/raspberrypi/blob/master/modbusPWMpi/modbusPWM.py" \t "_blank).

**Video of what I did:**

**Source Code:**

from pymodbus.server.async import StartTcpServer

from pymodbus.device import ModbusDeviceIdentification

from pymodbus.datastore import ModbusSequentialDataBlock, ModbusSlaveContext, ModbusServerContext

from pymodbus.transaction import ModbusRtuFramer, ModbusAsciiFramer

from twisted.internet.task import LoopingCall

from threading import Thread

import pid

import threading

from time import sleep

import RPi.GPIO as GPIO

import os

os.system('modprobe w1-gpio')

os.system('modprobe w1-therm')

#set up Raspberry GPIO

GPIO.setmode(GPIO.BCM)

GPIO.setwarnings(False)

GPIO.setup(25,GPIO.OUT)

pwm = GPIO.PWM(25,60)

pwmDutyCycle=100

pwm.start(pwmDutyCycle)

temperaturePoll = None

class Temp(Thread):

"""

A class for getting the current temp of a DS18B20

"""

def \_\_init\_\_(self, fileName=''):

Thread.\_\_init\_\_(self)

super(Temp, self).\_\_init\_\_()

self.\_stop = threading.Event()

self.tempDir = '/sys/bus/w1/devices/'

list = os.listdir(self.tempDir)

if(list[0][:2]=="28"):

fileName=list[0]

self.fileName = fileName

self.currentTemp = -999

self.correctionFactor = 1;

self.enabled = True

self.Run=True

def run(self):

while self.isEnabled():

try:

f = open(self.tempDir + self.fileName + "/w1\_slave", 'r')

except IOError as e:

print "Error: File " + self.tempDir + self.fileName + "/w1\_slave" + " does not exits"

return;

lines=f.readlines()

crcLine=lines[0]

tempLine=lines[1]

result\_list = tempLine.split("=")

temp = float(result\_list[-1])/1000 # temp in Celcius

temp = temp + self.correctionFactor # correction factor

#if you want to convert to Celcius, comment this line

temp = (9.0/5.0)\*temp + 32

if crcLine.find("NO") > -1:

temp = -999

self.currentTemp = temp

#print "Current: " + str(self.currentTemp) + " " + str(self.fileName)

sleep(.5)

#returns the current temp for the probe

def getCurrentTemp(self):

return self.currentTemp

#setter to enable this probe

def setEnabled(self, enabled):

self.enabled = enabled

#getter

def isEnabled(self):

return self.enabled

def updating\_writer(a):

context = a[0]

register = 3

slave\_id = 0x00

address = 0x00

global pwmDutyCycle,temp

#uncomment to debug temperature

print temp.getCurrentTemp()

values = [int(pwmDutyCycle),temp.getCurrentTemp()\*100]

context[slave\_id].setValues(register,address,values)

def read\_context(a):

context = a[0]

register = 3

slave\_id = 0x00

address = 0x00

value = context[slave\_id].getValues(register,address)[0]

global pwmDutyCycle

if(value!=pwmDutyCycle):

print value

pwmDutyCycle=value

pwm.ChangeDutyCycle(pwmDutyCycle)

def main():

store = ModbusSlaveContext(

di = ModbusSequentialDataBlock(0, [0]\*100),

co = ModbusSequentialDataBlock(0, [0]\*100),

hr = ModbusSequentialDataBlock(0, [0]\*100),

ir = ModbusSequentialDataBlock(0, [0]\*100))

context = ModbusServerContext(slaves=store, single=True)

identity = ModbusDeviceIdentification()

identity.VendorName = 'pymodbus'

identity.ProductCode = 'PM'

identity.VendorUrl = 'http://github.com/simplyautomationized'

identity.ProductName = 'pymodbus Server'

identity.ModelName = 'pymodbus Server'

identity.MajorMinorRevision = '1.0'

time = 5 # 5 seconds delaytime = 5 # 5 seconds delay

writer = LoopingCall(read\_context,a=(context,))

loop = LoopingCall(updating\_writer, a=(context,))

loop.start(.5) # initially delay by time

writer.start(.1)

StartTcpServer(context, identity=identity)#, address=("localhost", 502))

#cleanup async tasks

temp.setEnabled(False)

loop.stop()

writer.stop()

GPIO.cleanup()

if \_\_name\_\_ == "\_\_main\_\_":

temp = Temp()

temp.start()

main()

**Raspberry Pi SCADA: Communitating with S7-200 Using Python**

Video Demo

For how to compiling snap7 on the pi click [here](http://simplyautomationized.blogspot.com/2014/12/raspberry-pi-getting-data-from-s7-1200.html)  
  
Since the S7-200 is a bit different then the newer S7-1500/S7-1200 PLCs it took a bit of work to get it working. The python-snap7 library was missing the connection method for the old TSAP method of connecting. So I forked the library and added the missing function to connect. I did a pull request to get it merged back with the original and it's pending currently. Here is the link to the forked library:  
<https://github.com/SimplyAutomationized/python-snap7>  
  
I've also been working on a [din rail enclosure](http://www.amazon.com/BUD-Industries-DPB-4760-Plastic-Textured/dp/B005T85A6W/ref=as_li_bk_tl/?tag=simplautom-20&linkId=154fddf5c64dfee033484365da6f7502&linkCode=ktl) for the pi that wasn't so large.   I'm going to be adding a 24vdc to 5vdc dc-dc converter to the enclosure it so I don't have to bring in usb to it.

[](https://1.bp.blogspot.com/-Tmze6JzUryo/VTwvsYKl84I/AAAAAAAAX5k/nP1cgvn36H4/s1600/IMG_3898.JPG)

[](https://1.bp.blogspot.com/-J66VgCSL4HE/VTwvsesrTtI/AAAAAAAAX5k/gC_IPhPm0XQ/s1600/IMG_3899.JPG)  
[](https://2.bp.blogspot.com/-se7OSonk6pw/VTwvsSStviI/AAAAAAAAX5k/TYcwoWYC1LI/s1600/IMG_3900.JPG)

[](https://2.bp.blogspot.com/-_CLTadfQNNU/VTwwRbsImUI/AAAAAAAAX5s/m3O4yFfJf88/s1600/IMG_3985.JPG)

[](https://2.bp.blogspot.com/-Sk_mZRPEWGM/VTwwRRcd4-I/AAAAAAAAX5s/s_jNQ_YN_20/s1600/IMG_3984.JPG)

Anyway back to the good stuff. I added the Cli\_Connection in the Client.py library so the code will now be as follows:

import snap7 as p

plc = p.client.Client()

plc.set\_connection\_params('10.10.55.250',11,11)

plc.connect()

print plc.get\_connected()

if(plc.get\_connected()):

plc.disconnect()

If you import my helper S7200.py library from github the code is as follows:

import S7200 as p

plc = p.S7\_200('10.10.55.250',0x1100,0x1100,debug=True)

plc.writeMem('QX0.0',True)

print plc.getMem('freal10')

plc.writeMem('freal10',3.141592653589)

print plc.getMem('freal10')

print plc.getMem('QX0.0')

* This will turn on output 1 (Q0.0, will turn on if there's no ladder associated with it).
* Then read the real number stored in V memory 10.
* Then write pi to it ;-).
* Then read from it again.
* Then read if output 1 (Q0.0) is on still\

Github for helper library found here:  
<https://github.com/SimplyAutomationized/raspberrypi/blob/master/S7-200pi/S7200.py>