Serial communication between Raspberry Pi and Arduino
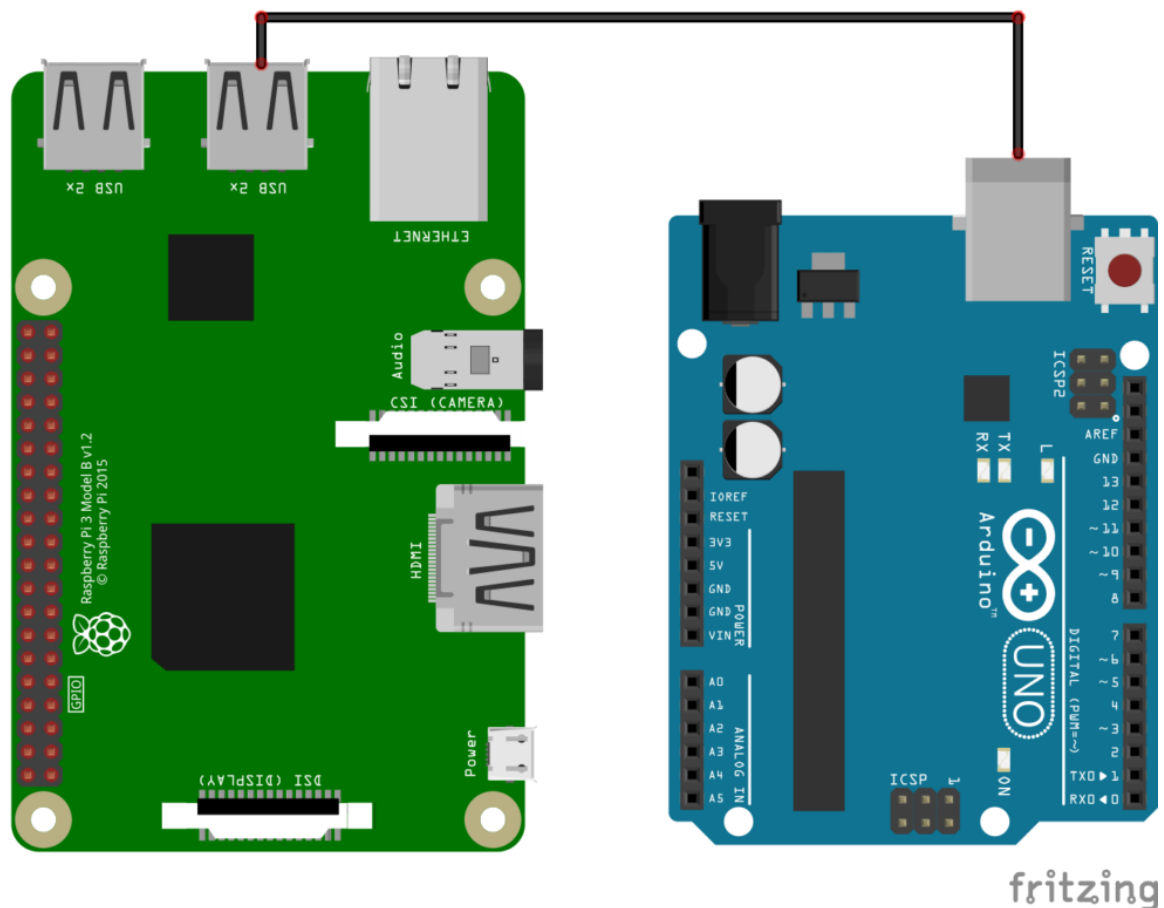
by Xukyo | 27 Sep 2020 | Tutorials | 16 comments


Tags: Arduino, C/C++, Python, Raspberry PI, Serial communication

★★★★☆ 4 (16)

In some projects it may be interesting to establish a serial communication between Raspberry Pi and Arduino. This makes it possible to couple the computing power and wireless interfaces of the Raspberry Pi with the inputs/outputs and the Arduino collection of modules. The first example that comes to mind is the use of this system for home automation in which the Raspberry Pi will host the control interface and intelligence and the Arduino will act as a programmable automaton acting on the components at the end of the chain (light, radiator, fan, sensors, etc.).

We will see in this tutorial how to set up a serial communication between Raspberry Pi and Arduino via the USB port. In this article we use the Arduino UNO card but it can be adapted to other types of cards with a serial connection (Nano, Mega, Feather, EPS32, ESP8266, etc.).

ⓘ

F3L400R10W3S7F

Wind power inverter system Module

CDH IC MODULE
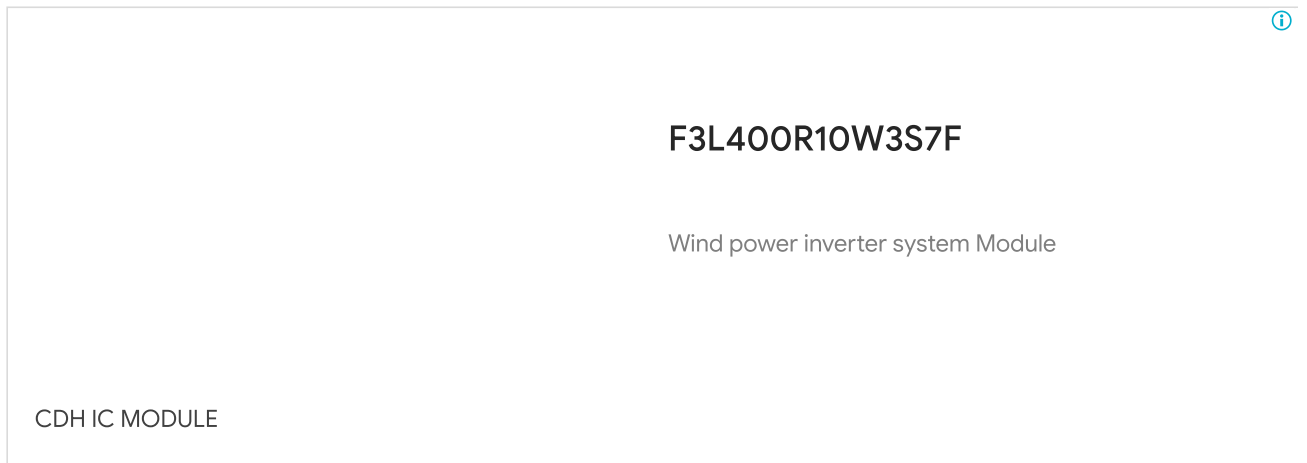
Prerequisite: Serial communication with Arduino,Remote access to Raspberry Pi with VNC

Hardware

- Computer
- Arduino UNO
- Raspberry Pi 3B+
- USB A Male / USB B Male

Wiring diagram

To establish serial communication between Raspberry Pi and Arduino, simply connect them with a suitable USB cable. In our case, we use a Raspberry Pi 3B+ and an Arduino UNO. So we need a USBA Male to USB B Male cable.
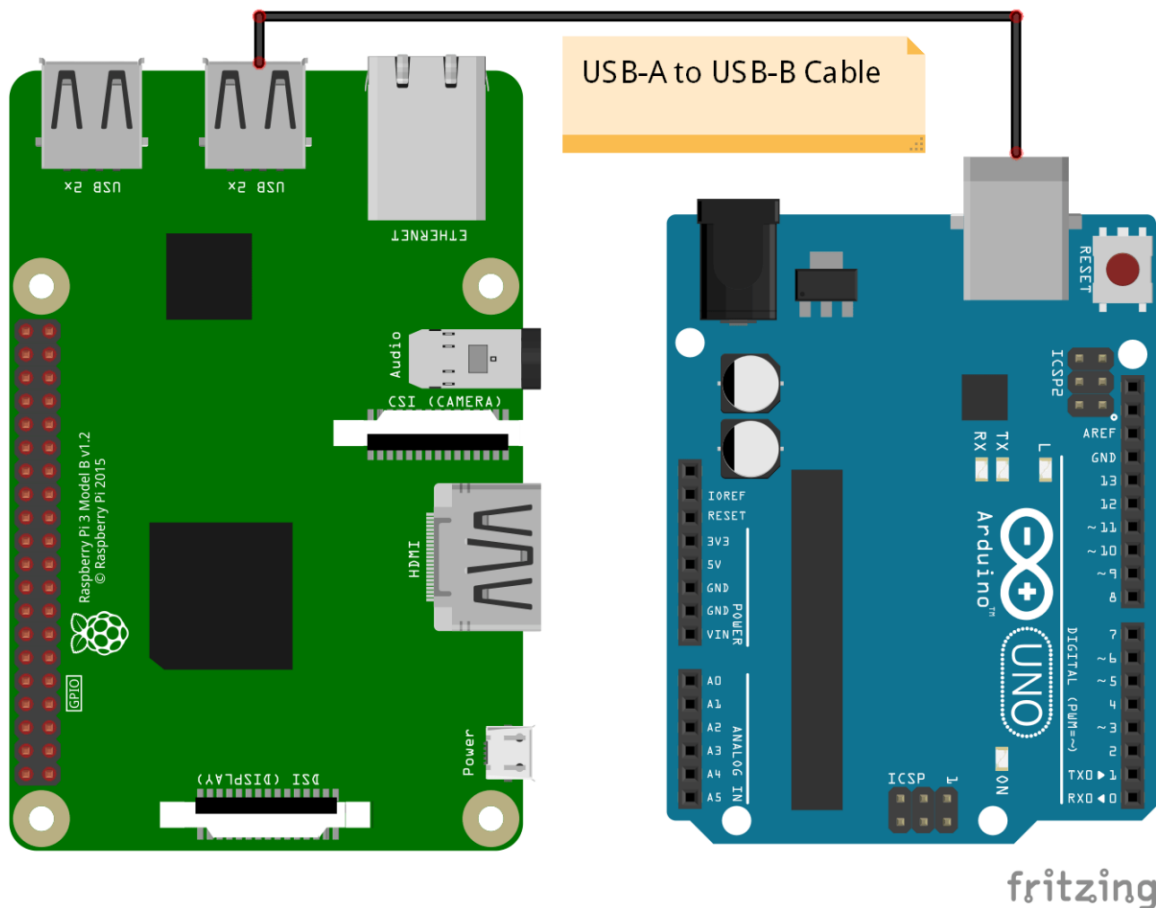
It is also possible to create serial communication by using the Rx/Tx pins of both components. In this tutorial we will focus on the USB connection.

- Tx GPIO14(RPI) <–> Rx 0(Arduino)
- Rx GPIO15(RPI) <–> Tx 1(Arduino)
- GND (RPI) <–> GND(Arduino)

Raspberry Pi configuration

We remind you that in order to be able to use your Raspberry Pi without screen or keyboard, the VNC remote connection must be configured.

To use the serial interface of the Raspberry Pi, it must be enabled in the configuration menu. To do this, enter the following command in a terminal:

```
sudo raspi-config
```
Copy

In the menu, select "5 – Interfacing Options" then "P6 Serial" and validate.

Once the connection has been made, you can check the devices connected to the serial port by typing the command in the terminal:

```
lsusb
```
Copy

The Raspberry Pi returns the list of devices connected to the USB ports.

<div align="right">Copy</div>

```
pi@raspberrypi:~ $ lsusb
Bus 001 Device 002: ID 2341:0043 Arduino SA Uno R3 (CDC ACM)
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

To find the name of the port to which the Arduino is connected, we use the command:

<div align="right">Copy</div>

```
dmesg | grep "tty"
```

This command returns the system messages related to the serial ports. You had to find the name of the port in the last messages. In our case the port name is ttyACM0.

<div align="right">Copy</div>

```
pi@raspberrypi:~ $ dmesg | grep "tty"
[    0.000000] Kernel command line: coherent_pool=1M
8250.nr_uarts=1 bcm2708_fb.fbwidth=1824
bcm2708_fb.fbheight=984 bcm2708_fb.fbswap=1
smsc95xx.macaddr=B8:27:EB:23:CF:07 vc_mem.mem_base=0x1ec00000
vc_mem.mem_size=0x20000000  console=ttyS0,115200 console=tty1
root=/dev/mmcblk0p7 rootfstype=ext4 elevator=deadline
fsck.repair=yes rootwait quiet splash plymouth.ignore-serial-
consoles
[    0.000636] console [tty1] enabled
[    0.951553] 20201000.serial: ttyAMA0 at MMIO 0x20201000
(irq = 81, base_baud = 0) is a PL011 rev2
[    0.954393] console [ttyS0] disabled
[    0.954466] 20215040.serial: ttyS0 at MMIO 0x0 (irq = 53,
base_baud = 31250000) is a 16550
[    0.954570] console [ttyS0] enabled
[    5.378641] systemd[1]: Created slice system-
serial\x2dgetty.slice.
[ 1455.402071] cdc_acm 1-1:1.0: ttyACM0: USB ACM device
[ 1581.980257] cdc_acm 1-1:1.0: ttyACM0: USB ACM device
```

Installation of the Arduino IDE on Raspberry Pi

To install the Arduino IDE on Raspberry Pi, it is best to go through the terminal. Simply enter the following lines of code:

Copy

```
mkdir ~/Applications
cd ~/Applications
wget https://downloads.arduino.cc/arduino-1.8.9-
linuxarm.tar.xz
tar xvJf arduino-1.8.9-linuxarm.tar.xz
cd arduino-1.8.9/
 ./install.sh
rm ../arduino-1.8.9-linuxarm.tar.xz
```

This will then allow you to program the Arduino directly from the Raspberry Pi.

Code

Code Arduino

The library used for serial communication on the Arduino side is the same as for communicating with the serial monitor, the Serial.h library that we know well. Make sure that the communication speed is the same for both devices (baudrate=9600) otherwise the communication will not work.

Copy

```cpp
String nom = "Arduino";
String msg;

void setup() {
  Serial.begin(9600);
}

void loop() {
  readSerialPort();

  if (msg != "") {
    sendData();
  }
  delay(500);
}

void readSerialPort() {
  msg = "";
  if (Serial.available()) {
```

Code Python

In this tutorial we will use the Python language on the Raspberry Pi side. The library used to manage the serial communication is the serial library.
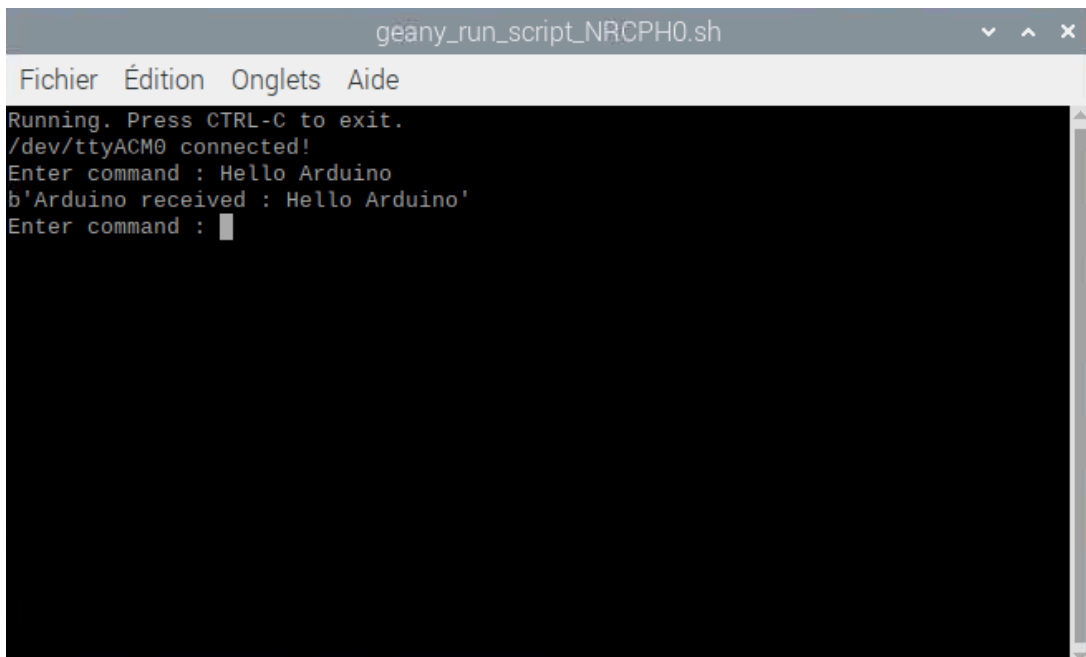
```python
1.    #!/usr/bin/env python
2.    # -*- coding: utf-8 -*-
3.    # lsusb to check device name
4.    #dmesg | grep "tty" to find port name
5.
6.    import serial,time
7.
8.
9.    if __name__ == '__main__':
10.
11.       print('Running. Press CTRL-C to exit.')
12.       with serial.Serial("/dev/ttyACM0", 9600, timeout=1) as arduino:
13.           time.sleep(0.1) #wait for serial to open
14.           if arduino.isOpen():
15.               print("{} connected!".format(arduino.port))
16.               try:
17.                   while True:
18.                       cmd=input("Enter command : ")
19.                       arduino.write(cmd.encode())
20.                       #time.sleep(0.1) #wait for arduino to answer
21.                       while arduino.inWaiting()==0: pass
22.                       if  arduino.inWaiting()>0:
23.                           answer=arduino.readline()
24.                           print(answer)
25.                           arduino.flushInput() #remove data after readi
26.               except KeyboardInterrupt:
27.                   print("KeyboardInterrupt has been caught.")
```

Result

The Raspberry Pi sends the order "Hello Arduino" to the Arduino, and the Arduino replies with its name and the order received.

Practical example

Once communication is established between Raspberry Pi and Arduino, the interesting thing is to control the Arduino's I/O and retrieve the sensor values. In this example, we will define different commands:

- one to retrieve sensor values
- one for switching the LED on and off on pin 13

Code Arduino

The Arduino returns data when it receives the "data" command and switches the LED connected to pin 13 on and off according to the "led0" and "led1" commands.

```arduino
const int ledPin=13;
String nom = "Arduino";
String msg;

void setup() {
  Serial.begin(9600);
  pinMode(ledPin,OUTPUT);
}

void loop() {
  readSerialPort();

  if (msg == "data") {
    sendData();
  }else if(msg=="led0"){
    digitalWrite(ledPin,LOW);
    Serial.println(" Arduino set led to LOW");
  }else if(msg=="led1"){
    digitalWrite(ledPin,HIGH);
```

Copy

Code Python

```python
1.  #!/usr/bin/env python
2.  # -*- coding: utf-8 -*-
3.  # lsusb to check device name
4.  #dmesg | grep "tty" to find port name
5.
6.  import serial,time
7.
8.
9.  if __name__ == '__main__':
10.     print('Running. Press CTRL-C to exit.')
11.     with serial.Serial("/dev/ttyACM0", 9600, timeout=1) as arduino:
12.         time.sleep(0.1) #wait for serial to open
13.         if arduino.isOpen():
14.             print("{} connected!".format(arduino.port))
```

```
15.                     try:
16.                         while True:
17.                             cmd=input("Enter command (data,led0 or led1): ")
18.                             arduino.write(cmd.encode())
19.                             #time.sleep(0.1) #wait for arduino to answer
20.
21.                             while arduino.inWaiting()==0: pass
22.                             if  arduino.inWaiting()>0:
23.                                 answer=str(arduino.readline())
24.                                 print("---> {}".format(answer))
25.                                 if cmd=="data":
26.                                     dataList=answer.split("x")
27.                                     print("led state : {}".format(dataList[(
28.                                     print("Analog input A0 : {}".format(data
29.                                     print("Analog input A1: {}".format(datal
```

Once the two codes have been uploaded and launched, it can be seen that when the "data" command is entered into the terminal, Arduino returns many bytes containing the sensor values. It is possible to separate this response into a list using the split() function and the "x" character and, in this way, recover the sensor values and the status of the led. With this code we can control the status of the LED on pin 13.

```
                    geany_run_script_PIZ4H0.sh          ∨  ∧  ✕

 Fichier  Édition  Onglets  Aide

 Running. Press CTRL-C to exit.
 /dev/ttyACM0 connected!
 Enter command (data,led0 or led1): data
 ---> b'0x517x443'
 led state : b'0
 Analog input A0 : 517
 Analog input A1: 443'
 Enter command (data,led0 or led1): led1
 ---> b' Arduino set led to HIGH'
 Enter command (data,led0 or led1): data
 ---> b'1x304x285'
 led state : b'1
 Analog input A0 : 304
 Analog input A1: 285'
 Enter command (data,led0 or led1): led0
 ---> b' Arduino set led to LOW'
 Enter command (data,led0 or led1): data
 ---> b'1x300x284'
 led state : b'1
 Analog input A0 : 300
 Analog input A1: 284'
 Enter command (data,led0 or led1): █
```

By modifying this code, you will be able to control and observe any entry/exit of the Arduino on the Raspberry Pi.

Application

- Create a graphical user interface (GUI) under Raspberry Pi to drive an Arduino and retrieve sensor values

Sources

- Library pySerial
- Serial communication with Arduino
- Remote access to Raspberry Pi with VNC

Find other examples and tutorials in our Automatic code generator

Code Architect

How useful was this post?

Click on a star to rate it!

★★★★★

Average rating 4 / 5. Vote count: 16

16 Comments

Par Electric on 2020-11-27 at 11:36 AM

great example. This is the first example that I have tried that works for 2 way serial comm. Great job. thanks

Reply

Xukyo on 2020-11-27 at 2:16 PM

Thanks a lot for your comment!

Reply

shan on 2021-02-03 at 1:02 AM

can u share the code please? its urgent as i'm stuck..

Reply

Xukyo on 2021-02-03 at 10:34 AM

What do you mean? The code is available in the tutorial.

Reply

Sam on 2021-03-11 at 10:58 AM

the second python code is the same as the first one, so the second example is not working for me

Reply

Xukyo on 2021-03-11 at 11:41 AM

Thanks for letting me know! It has been corrected

jordi on 2021-05-25 at 8:35 PM

Hello Good Example,

But i have one problem when i execute the code in Raspberry:

if arduino.isOpen():

IdenticationError: unexpected ident

can you help me?

Reply

Xukyo on 2021-05-27 at 12:31 PM

Unexpected indent might be due to a copy/paste error.
In python the code need to respect indentation. I suggest you check indent or space at the beginnig of each line. There is a tool for that in Geany on Raspbian

Reply

John on 2021-06-10 at 3:15 PM

These examples seem very promising but unfortunately I can't make them work. I always get NameError like below:

Running. Press CTRL-C to exit.

/dev/ttyUSB0 connected!

Enter command (data,led0 or led1): data

Traceback (most recent call last):

File "com-test.py", line 17, in

cmd=input("Enter command (data,led0 or led1): ")

File "", line 1, in

NameError: name 'data' is not defined

I tried some 2-way serial examples from another site and they work fine. The code in this post seems much simpler and would be easier to modify and use.

Reply

Xukyo on 2021-06-10 at 4:11 PM

How do you run the python script?

try to add quotation marks when writing in the command prompt : "data"

this error is weird.

File « », line 1, in

NameError: name 'data' is not defined

Did you redefined the function "input"?

Reply

John on 2021-06-10 at 5:18 PM

Thanks, all is well now. User error. I previously tried to run it with "python com-test.py", New try with "python3 com-test.py" and that error is no more! I have some experience with Arduino but I'm a total noob when it comes to Python...

Reply

Xukyo on 2021-06-14 at 6:58 PM

Great news! I should have mentionned that the script is written in Python3

Reply

Yuri Franken on 2022-01-14 at 3:28 PM

Nice article!

I need to set pin13 to output (pinMode(13, OUTPUT); to make the led example work

also I need to run the codes in python3

Reply

Dillan on 2022-04-29 at 8:41 PM

Hi there, I have managed to connect the raspberry pi for writing to the arduinos serial monitor, however when i implement the reading of the serial monitor i get the following error. I cannot get around this, can you help?

self.fd = os.open(self.portstr, os.O_RDWR | os.O_NOCTTY | os.O_NONBLOCK)
FileNotFoundError: [Errno 2] No such file or directory: '/dev/ttyACM0'

During handling of the above exception, another exception occurred:
with serial.Serial("/dev/ttyACM0", 9600, timeout=1) as arduino:
File "/usr/lib/python3/dist-packages/serial/serialutil.py", line 240, in _init_
self.open()
File "/usr/lib/python3/dist-packages/serial/serialposix.py", line 268, in open
raise SerialException(msg.errno, "could not open port {}: {}".format(self._port, msg))
serial.serialutil.SerialException: [Errno 2] could not open port /dev/ttyACM0: [Errno 2] No such file or directory: '/dev/ttyACM0'

Reply

Xukyo on 2022-05-01 at 2:52 PM

Hi,
Did you activate the serial com in raspi-config?
the error message "could not open port /dev/ttyACM0" means seril com not enabled or arduino device not hooked up on this port. You need to find the port on which the arduino is hooked

Reply

Fella on 2022-06-08 at 9:52 PM

I know you have C++ tutorial on raspberry pi but in this tutorial I need the C++ code in place of python code on raspberry pi side.

I have raspberry pi 3B and I want to build open CV in C++. I have tried some tutorials but that didn't work.

Reply

Search

Recent Posts

Get started with TE0720 and Xilinx Vivado

Management of several sensors with a shift register

Multitasking program with Arduino Due and Scheduler

Managing an IP Camera with Python

Creating (more beautiful) components in Fritzing

Newsletter

Prénom

Adresse E-mail

Inscrivez-moi !

Raspberry Pi

Arduino

Processing

Fritzing

- About

- Contact

- CGV

- CGU

- Raspberry Pi

- Arduino

- Processing

- Fritzing

- Projects

- News

- Tutorials

- Politique de cookies (EU)