



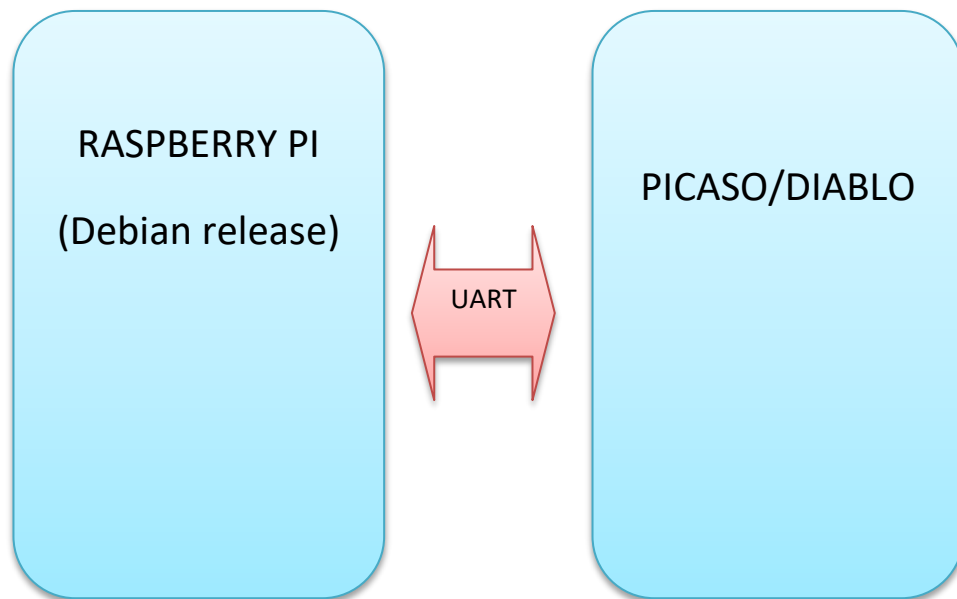
ViSi Setting Up the Raspberry Pi for ViSi Projects

DOCUMENT DATE: **22nd April 2019**
DOCUMENT REVISION: **1.1**



Description

UART/Serial communication with Raspberry Pi is one way to transport information from-and-to a Raspberry Pi and an external device. [WiringPi](#) is a library written to access general purpose inputs-outputs of the Raspberry Pi. This library is released under the GNU LGPLv3 license. In this application note, the procedure on WiringPi library setup is presented. Also, an example code written using the Workshop IDE-ViSI environment using 4D Graphics Language is included for testing the serial communications.



Before getting started, the following are required:

- Any of the following 4D Picaso display modules:

[uLCD-24PTU](#)
[gen4-uLCD-24PT](#)

[uLCD-28PTU](#)
[gen4-uLCD-28PT](#)

[uVGA-III](#)
[gen4-uLCD-32PT](#)

and other superseded modules which support the ViSi environment.

- The target module can also be a Diablo16 display

[gen4-uLCD-24D](#)
[Series](#)
[gen4-uLCD-35D](#)
[Series](#)
[gen4-uLCD-70D](#)
[Series](#)
[uLCD-35DT](#)

[gen4-uLCD-28D](#)
[Series](#)
[gen4-uLCD-43D](#)
[Series](#)
[uLCD-43D Series](#)

[gen4-uLCD-32D](#)
[Series](#)
[gen4-uLCD-50D](#)
[Series](#)
[uLCD-70DT](#)

Visit www.4dsystems.com.au/products to see the latest display module products that use the Diablo16 processor.

- [4D Programming Cable](#) / [µUSB-PA5/µUSB-PA5-II](#) for non-gen4 displays (uLCD-xxx)
- [4D Programming Cable](#) & [gen4-IB](#) / [gen4-PA](#) / [4D-UPA](#), for gen-4 displays (gen4-uLCD-xxx)
- [micro-SD \(µSD\)](#) memory card
- [Workshop 4 IDE](#) (installed according to the installation document)

- When downloading an application note, a list of recommended application notes is shown. It is assumed that the user has read or has a working knowledge of the topics presented in these recommended application notes.

Content

Description 2

Content 3

Application Overview 4

Setup WiringPi on Raspberry Pi 4

Download and Install using GIT 4

Obtaining WiringPi using GIT 5

Raspberry Pi WiringPi Test Program 6

Open Test Serial Project 8

Connect the Raspberry Pi to 4D Systems Intelligent Display 9

Run the ViSi Based Program 9

Proprietary Information 10

Disclaimer of Warranties & Limitation of Liability 10

Application Overview

Raspberry Pi is a credit-card sized single board computer that is capable of running a several Linux distribution images. The Linux images running on the Raspberry Pi is a stand-alone operating system that is capable of allowing internal processing and communication with external devices.

This application note demonstrates the setup of WiringPi library and a simple uart/serial loopback test. The uart/serial loopback test is done together with a 4D Systems intelligent display module. The basic program for the display module is developed using the ViSi environment of the Workshop IDE.

Setup WiringPi on Raspberry Pi

The procedure included in this section is referenced from the official [WiringPi Download and Install](#) procedure. The content of this section intends to consolidates setup procedures necessary to establish serial communication between Raspberry Pi and 4D Systems intelligent displays using the aforementioned library.

Download and Install using GIT

Before proceeding to the library download and installation process, ensure that the Raspberry Pi is up to date. Issuing the commands below will update and upgrade your Raspberry Pi.

```
sudo apt-get update
```

```
sudo apt-get upgrade
```

Git-core based downloads requires that the GIT application is installed on the Raspberry Pi. To install use the line below.

```
sudo apt-get install git-core
```

Image of successful GIT installation

```
pi@raspberrypi ~ $ sudo apt-get install git-core
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  git-core
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 1,342 B of archives.
After this operation, 21.5 kB of additional disk space will be used.
Get:1 http://mirrordirector.raspbian.org/raspbian/ wheezy/main git-core all 1:1.7.10.4-1+wheezy1+rp12 [1,342 B]
Fetched 1,342 B in 1s (794 B/s)
Selecting previously unselected package git-core.
(Reading database ... 76937 files and directories currently installed.)
Unpacking git-core (from .../git-core_1%3a1.7.10.4-1+wheezy1+rp12_all.deb) ...
Setting up git-core (1:1.7.10.4-1+wheezy1+rp12) ...
pi@raspberrypi ~ $
```

Obtaining WiringPi using GIT

Having been able to successfully install GIT on your Raspberry Pi, use the line below to download the WiringPi library files.

```
git clone git://git.drogon.net/wiringPi
```

Image for successful clone of WiringPi library files.

```
pi@raspberrypi ~ $ git clone git://git.drogon.net/wiringPi
Cloning into 'wiringPi'...
remote: Counting objects: 742, done.
remote: Compressing objects: 100% (676/676), done.
remote: Total 742 (delta 537), reused 95 (delta 58)
Receiving objects: 100% (742/742), 264.40 KiB | 27 KiB/s, done.
Resolving deltas: 100% (537/537), done.
pi@raspberrypi ~ $ ls -al
total 36
drwxr-xr-x 4 pi pi 4096 Apr 14 02:51 .
drwxr-xr-x 3 root root 4096 Feb 15 14:05 ..
-rw-r--r-- 1 pi pi 148 Feb 16 14:07 .bash_history
-rw-r--r-- 1 pi pi 220 Feb 15 14:05 .bash_logout
-rw-r--r-- 1 pi pi 3243 Feb 15 14:05 .bashrc
-rw-r--r-- 1 pi pi 675 Feb 15 14:05 .profile
drwxr-xr-x 2 pi pi 4096 Jan 27 08:34 python_games
drwxr-xr-x 9 pi pi 4096 Apr 14 02:52 wiringPi
-rw-r--r-- 1 pi pi 57 Apr 14 02:47 xauthrity
pi@raspberrypi ~ $
```

After successfully obtaining the files, navigate to the download directory using the line below. The build script contained in the downloaded files will compile and install the library.

```
cd wiringPi
```

```
./build
```

Image for successful compile and installation

```
pi@raspberrypi ~/wiringPi $ ./build
wiringPi Build script
=====

WiringPi Library
[UnInstall]
[Compile] wiringPi.c
[Compile] wiringSerial.c
[Compile] piHiPri.c
[Compile] wiringShift.c
[Compile] piThread.c
[Compile] wiringPiSPI.c
[Compile] wiringPiI2C.c
[Compile] softPwm.c
[Compile] softTone.c
[Compile] mcp23008.c
[Compile] mcp23016.c
[Compile] mcp23017.c
[Compile] mcp23s08.c
[Compile] mcp23s17.c
[Compile] sr595.c
[Compile] pcf8574.c
[Compile] pcf8591.c
[Compile] mcp3002.c
[Compile] mcp3004.c
[Compile] mcp4802.c
[Compile] mcp3422.c
[Compile] max31855.c
[Compile] max5322.c
[Compile] sn3218.c
[Compile] drcSerial.c
[Compile] wpiExtensions.c
[Link (Dynamic)]
[Install Headers]
[Install Dynamic Lib]
```

```

WiringPi Devices Library
[UnInstall]
[Compile] piNes.c
[Compile] ds1302.c
[Compile] maxdetect.c
[Compile] gertboard.c
[Compile] piFace.c
[Compile] lcd128x64.c
[Compile] lcd.c
[Compile] piGlow.c
[Link (Dynamic)]
[Install Headers]
[Install Dynamic Lib]

GPIO Utility
[Compile] readall.c
[Compile] gpio.c
[Compile] pins.c
[Link]
[Install]

All Done.

NOTE: To compile programs with wiringPi, you need to add:
    -lwiringPi
to your compile line(s) To use the Gertboard, MaxDetect, etc.
code (the devLib), you need to also add:
    -lwiringPiDev
to your compile line(s).

pi@raspberrypi ~/wiringPi $

```

Raspberry Pi WiringPi Test Program

WiringPi library is bundled with examples. The examples are located inside the wiringPi folder /home/pi/wiringPi/examples. Navigate to the installation directory using the following line.

```
pi@raspberrypi ~ $ cd wiringPi/examples/
```

```

pi@raspberrypi ~ $ cd wiringPi/examples/
pi@raspberrypi ~/wiringPi/examples $ ls -al
total 168
drwxr-xr-x 6 pi pi 4096 Apr 14 02:52 .
drwxr-xr-x 9 pi pi 4096 Apr 14 02:52 ..
-rw-r--r-- 1 pi pi 2622 Apr 14 02:52 blink12.c
-rw-r--r-- 1 pi pi 3134 Apr 14 02:52 blink12drcs.c
-rw-r--r-- 1 pi pi 2922 Apr 14 02:52 blink6drcs.c
-rw-r--r-- 1 pi pi 1673 Apr 14 02:52 blink8.c
-rw-r--r-- 1 pi pi 1459 Apr 14 02:52 blink.c
-rw-r--r-- 1 pi pi 1190 Apr 14 02:52 blink.rtb
-rw-r--r-- 1 pi pi 1242 Apr 14 02:52 blink.sh
-rw-r--r-- 1 pi pi 5049 Apr 14 02:52 clock.c
-rw-r--r-- 1 pi pi 7651 Apr 14 02:52 COPYING.LESSER
-rw-r--r-- 1 pi pi 2630 Apr 14 02:52 delayTest.c
-rw-r--r-- 1 pi pi 5649 Apr 14 02:52 ds1302.c
drwxr-xr-x 2 pi pi 4096 Apr 14 02:52 Gertboard
-rw-r--r-- 1 pi pi 1011 Apr 14 02:52 header.h
-rw-r--r-- 1 pi pi 3322 Apr 14 02:52 isr.c
-rw-r--r-- 1 pi pi 3289 Apr 14 02:52 isr-osc.c
-rw-r--r-- 1 pi pi 7822 Apr 14 02:52 lcd-adafruit.c
-rw-r--r-- 1 pi pi 6215 Apr 14 02:52 lcd.c
-rw-r--r-- 1 pi pi 1981 Apr 14 02:52 lowPower.c
-rw-r--r-- 1 pi pi 3795 Apr 14 02:52 Makefile

```

```
-rw-r--r-- 1 pi pi 2243 Apr 14 02:52 nes.c
-rw-r--r-- 1 pi pi 2181 Apr 14 02:52 okLed.c
drwxr-xr-x 2 pi pi 4096 Apr 14 02:52 PiFace
drwxr-xr-x 2 pi pi 4096 Apr 14 02:52 PiGlow
-rw-r--r-- 1 pi pi 1550 Apr 14 02:52 pwm.c
drwxr-xr-x 2 pi pi 4096 Apr 14 02:52 q2w
-rw-r--r-- 1 pi pi 353 Apr 14 02:52 README.TXT
-rw-r--r-- 1 pi pi 1843 Apr 14 02:52 rht03.c
-rw-r--r-- 1 pi pi 1456 Apr 14 02:52 serialRead.c
-rw-r--r-- 1 pi pi 1995 Apr 14 02:52 serialTest.c
-rw-r--r-- 1 pi pi 1653 Apr 14 02:52 servo.c
-rw-r--r-- 1 pi pi 2229 Apr 14 02:52 softPwm.c
-rw-r--r-- 1 pi pi 1527 Apr 14 02:52 softTone.c
-rw-r--r-- 1 pi pi 2673 Apr 14 02:52 speed.c
-rw-r--r-- 1 pi pi 3401 Apr 14 02:52 spiSpeed.c
-rw-r--r-- 1 pi pi 3762 Apr 14 02:52 wfi.c
pi@raspberrypi ~/wiringPi/examples $
```

Included in the example codes is the 'serialTest.c'. The serialTest.c program file code is as follows.

```
#include <stdio.h>
#include <string.h>
#include <errno.h>

#include <wiringPi.h>
#include <wiringSerial.h>

int main ()
{
    int fd ;
    int count ;
    unsigned int nextTime ;

    if ((fd = serialOpen ("/dev/ttyAMA0", 115200)) < 0)
    {
        fprintf (stderr, "Unable to open serial device: %s\n", strerror (errno)) ;
        return 1 ;
    }
}
```

```
if (wiringPiSetup () == -1)
{
    fprintf (stdout, "Unable to start wiringPi: %s\n", strerror (errno)) ;
    return 1 ;
}

nextTime = millis () + 300 ;

for (count = 0 ; count < 256 ; )
{
    if (millis () > nextTime)
    {
        printf ("\nOut: %3d: ", count) ;
        fflush (stdout) ;
        serialPutch (fd, count) ;
        nextTime += 300 ;
        ++count ;
    }

    delay (3) ;

    while (serialDataAvail (fd))
    {
        printf (" -> %3d", serialGetchar (fd)) ;
        fflush (stdout) ;
    }

    printf ("\n") ;
    return 0 ;
}
```

The program is implementing a loopback sequence using the /dev/ttyAMA0 or on-board UART port at a rate of 115200 baudrate. The following sections will present the other end of the serial communication system, wherein the information sent by the Raspberry Pi is posted on the 4D Systems intelligent module's screen and then sent back.

The Raspberry Pi will then display on the terminal the original information sent.

After navigating to the examples folder, compilation of the project can be initiated. Referring to the README.TXT included in the examples folder this

is the commands to be used when current working path is wiringPi/examples/ .

```
pi@raspberrypi ~ $ cd wiringPi/examples/  
pi@raspberrypi ~/wiringPi/examples $ make serialTest  
[CC] serialTest.c  
[link]  
pi@raspberrypi ~/wiringPi/examples $ chmod +x serialTest
```

Use the 'make' command included in the examples folder to build the serialTest.c project. After a successful compilation files are generated having the same name but with different extension names. To run the project, simply type

```
pi@raspberrypi ~/wiringPi/examples $ sudo ./serialTest
```

The output should print onto the terminal the current value of 'Out'.

**** To ensure access to the /dev/ttyAMA0 or UART port, disable shell and kernel messages on the serial connection.**

Using 'sudo raspi-config' on terminal. Then, navigate to the user interface.

ADVANCE OPTIONS >> SERIAL >> DISABLE

Open Test Serial Project

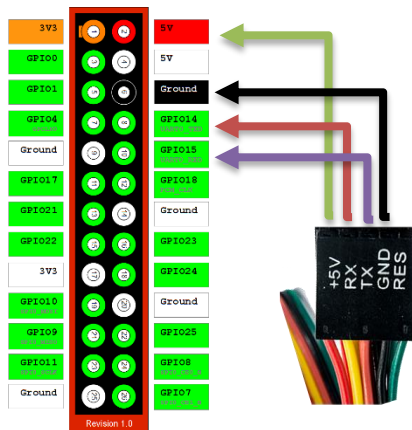
Download the application note ViSi project file – 'testSerial.zip'. Please follow the link below.

[ViSi Setting Up the Raspberry Pi for ViSi Projects](#)

Connect the Raspberry Pi to 4D Systems Intelligent Display



Connect the pins of the Raspberry Pi and 4D Systems Intelligent Display Module following the connection guide below.



Run the ViSi Based Program

For instructions on how to save a **ViSi** project, how to connect the target display to the PC, how to select the program destination, and how to compile and download a program, please refer to the section “**Run the Program**” of the application note

[ViSi Getting Started - First Project for Picaso and Diablo16](#)

Proprietary Information

The information contained in this document is the property of 4D Systems Pty. Ltd. and may be the subject of patents pending or granted, and must not be copied or disclosed without prior written permission.

4D Systems endeavours to ensure that the information in this document is correct and fairly stated but does not accept liability for any error or omission. The development of 4D Systems products and services is continuous and published information may not be up to date. It is important to check the current position with 4D Systems.

All trademarks belong to their respective owners and are recognised and acknowledged.

Disclaimer of Warranties & Limitation of Liability

4D Systems makes no warranty, either expresses or implied with respect to any product, and specifically disclaims all other warranties, including, without limitation, warranties for merchantability, non-infringement and fitness for any particular purpose.

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications.

In no event shall 4D Systems be liable to the buyer or to any third party for any indirect, incidental, special, consequential, punitive or exemplary damages (including without limitation lost profits, lost savings, or loss of business opportunity) arising out of or relating to any product or service provided or to be provided by 4D Systems, or the use or inability to use the same, even if 4D Systems has been advised of the possibility of such damages.

4D Systems products are not fault tolerant nor designed, manufactured or intended for use or resale as on line control equipment in hazardous environments requiring fail – safe performance, such as in the operation of nuclear facilities, aircraft navigation or communication systems, air traffic control, direct life support machines or weapons systems in which the failure of the product could lead directly to death, personal injury or severe physical or environmental damage ('High Risk Activities'). 4D Systems and its suppliers specifically disclaim any expressed or implied warranty of fitness for High Risk Activities.

Use of 4D Systems' products and devices in 'High Risk Activities' and in any other application is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless 4D Systems from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any 4D Systems intellectual property rights.