Raspiggy Rev 1.2

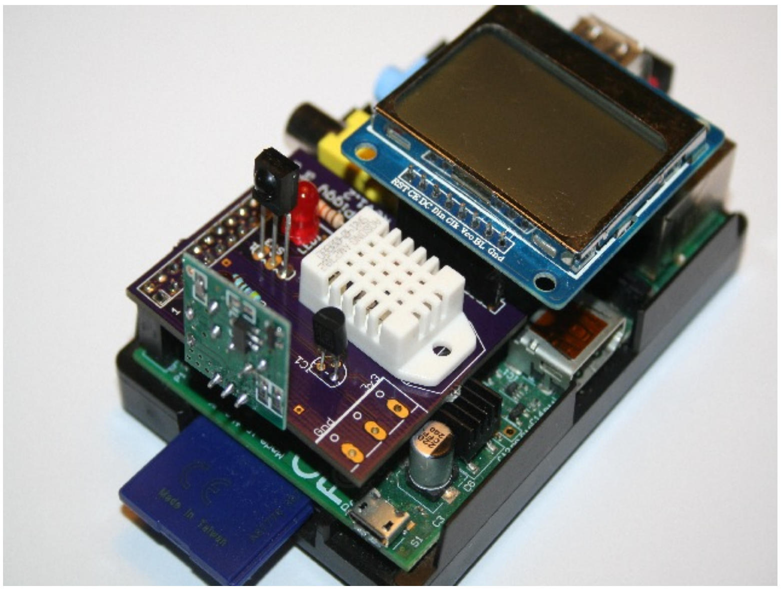
Manual

Version Number: 1.2
Date: June 2, 2014

This page
was intentionally
left blank

Introduction

Overview



RasPiggy is a Raspberry Pi extension board with:

- 433Mhz transmitter
- DS18x20 temperature sensor
- TSOP38x IR receiver
- DHT-22 humidity and temperature sensor
- Nokia 5110 LCD interface
- a user controllable LED / infra-red LED

In this manual you will find all the software and information to deploy RasPiggy quick and easy.

The IO of your Raspberry Pi

GPIO stands for General Purpose Input/Output and a GPIO pin can be set high (taking the value 1) by connecting it to a voltage supply, or set low (taking the value 0) by connecting it to ground. The Raspberry Pi can set the pin to take either value and treat it as an output, or it can detect the value of the pin and treat it as an input.

The Raspberry Pi has a 26-pin General Purpose Input/Output (GPIO) connector and this carries a set of signals and buses. There are 8 general-purpose digital I/O pins – these can be programmed as either digital outputs or inputs. One of these pins can be designated for PWM output too. Additionally there is a 2-wire I2C interface and a 4-wire SPI interface (with a 2nd select line, making it 5 pins in total) and the serial UART with a further 2 pins.

The Revision 2 Raspberry Pi has an additional 4 GPIO lines on a separate connector, which you have to solder onto the board.

IO can be confusing in Raspberry PI. To clear things up the following table may be helpful:

```
+----------+-Rev2-+------+--------+------+-------+
| wiringPi | GPIO | Phys | Name   | Mode | Value |
+----------+------+------+--------+------+-------+
|      0   |  17  |  11  | GPIO 0 | IN   | High  |
|      1   |  18  |  12  | GPIO 1 | IN   | High  |
|      2   |  27  |  13  | GPIO 2 | OUT  | Low   |
|      3   |  22  |  15  | GPIO 3 | IN   | High  |
|      4   |  23  |  16  | GPIO 4 | OUT  | Low   |
|      5   |  24  |  18  | GPIO 5 | OUT  | High  |
|      6   |  25  |  22  | GPIO 6 | OUT  | Low   |
|      7   |   4  |   7  | GPIO 7 | IN   | High  |
|      8   |   2  |   3  | SDA    | IN   | High  |
|      9   |   3  |   5  | SCL    | IN   | High  |
|     10   |   8  |  24  | CE0    | OUT  | Low   |
|     11   |   7  |  26  | CE1    | ALT0 | High  |
|     12   |  10  |  19  | MOSI   | OUT  | Low   |
|     13   |   9  |  21  | MISO   | ALT0 | Low   |
|     14   |  11  |  23  | SCLK   | OUT  | Low   |
|     15   |  14  |   8  | TxD    | ALT0 | High  |
|     16   |  15  |  10  | RxD    | ALT0 | High  |
|     17   |  28  |   3  | GPIO 8 | ALT2 | Low   |
|     18   |  29  |   4  | GPIO 9 | ALT2 | Low   |
|     19   |  30  |   5  | GPIO10 | ALT2 | Low   |
|     20   |  31  |   6  | GPIO11 | ALT2 | Low   |
+----------+------+------+--------+------+-------+
```

Addition information

Extra information on wiringPi can be found in these location(s):

https://projects.drogon.net/raspberry-pi/wiringpi/pins/

Getting your PI ready

The most recent software

The first step is to change where our Pi updates from, by editing a text file. We need to open up the file /etc/apt/sources.list.d/raspi.list as root, so type:

```
sudo leafpad /etc/apt/sources.list.d/raspi.list
```

Now change the line in this file so that it reads "`deb http://archive.raspberrypi.org/debian/ wheezy main untested`", then save and close the file.

Next, do the following commands:

```
apt-get update

apt-get upgrade
```

Installing wiringPi

WiringPi is a GPIO access library written in C for the BCM2835 used in the Raspberry Pi. It's released under the GNU LGPLv3 license and is usable from C and C++ and many other languages with suitable wrappers.

To install it use the following procedure

If you do not have GIT installed, then under any of the Debian releases (e.g. Raspbian), you can install it with:

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

To obtain WiringPi using GIT:

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

If you have already used the clone operation for the first time, then

```
cd wiringPi

git pull origin
```

Build/install wiringPi

To build/install there is a new simplified script:

```
cd wiringPi

./build
```

The new build script will compile and install it all for you – it does use the sudo command at one point, so you may wish to inspect the script before running it.

Test wiringPi’s installation

run the gpio command to check the installation:

```
gpio —v
```

```
gpio readall
```

That should give you some confidence that it’s working OK

Addition information

Extra information on wiringPi can be found in these location(s):

http://wiringpi.com

The LCD display

Connecting your LCD

The LCD display on Raspiggy is a Nokia 5110 compatible LCD display.

Note	Please check the pin assignment of your LCD before you connect it - Nokia 5110 pin assignments can vary, also different breakout-boards can have different pin assignments!

Raspiggy uses the following pin layout:

```
1. RST - Reset – connected to GPIO 5
2. CE - Chip Select
3. DC - data/instruction selection
4. DIN - serial data line
5. CLK - Serial Clock Line
6. VCC - power input (3.3v or 5v)
7. BL - backlight control – connected to GPIO 1
8. GND – Ground
```

Testing the display

You can test the LCD by running the nokia executable that can be found in the Raspiggy GitHub repository.

```
Wget
https://github.com/floresboy/RasPiggy/blob/master/nokia
```

Be sure to set the backlight pin to output before running the program.

```
gpio mode 1 out
```

Addition information

Extra information on this subject can be found in these location(s):

https://github.com/rm-hull/pcd8544

http://www.bartbania.com/raspberry_pi/raspberry-pi-lcd-monitor-raspberry-pi/

The LED

Prerequisites

Be sure to have WireingPi installed. See page 5

Testing method 1

The LED is connected to GPIO3. Toggling this output can control it. In wiringPi terminology GPIO3 is referred as number 2 (see page 4).

```
#include <wiringPi.h>

#define LED 2

int main (void)
{
  char a;
  wiringPiSetup () ;
  pinMode (LED, OUTPUT) ;
  for (a=1;a<10;a++)
  {
    digitalWrite (LED, HIGH) ; delay (500) ;
    digitalWrite (LED,  LOW) ; delay (500) ;
  }
  return 0 ;
}
```

A precompiled version of this test program called blink can be can be found in the Raspiggy GitHub repository:

```
Wget https://github.com/floresboy/RasPiggy/blob/master/blink
```

Testing method 2

As an alternative you could use this:

Open a root terminal from the menu, or by running `sudo su` in a normal terminal.

```
cd /sys/class/gpio/
echo "27" > export
cd gpio27
echo "out" > direction
echo "1" > value
```

The LED should now be turned on. To turn the LED off, type:

```
echo "0" > value
cd ..
echo "27" > unexport
```

The DS18B20 temp-sensor

To test the correct working of the 1-wire temperature sensor type these commands into a terminal:

```
sudo modprobe w1-gpio
sudo modprobe w1-therm
cd /sys/bus/w1/devices
ls
cd 10-xxxx (change this to match what serial number pops up)
cat w1_slave
```

You should see an output of two lines of text. The first line will say YES or NO at the end. If YES, then the measured temperature will be at the end of the second line, in 1/000 degrees C. If a NO appears on the end of the first line, the sensor is still working, it just failed to communicate with the Pi correctly. Reenter

```
cat w1_slave
```

and it should give a YES.

Two lines of text will be printed. On the second line, the section starting "t=" is the temperature in degrees Celsius. A decimal point goes after the first two digits, so the example value of "t=22250" is in fact "t=22.250" degrees Celsius

Loading W1 drivers

To load the 1-wire kernel modules that come pre-installed but not loaded only once:

```
sudo modprobe w1-gpio
sudo modprobe w1_therm
```

To load the drivers automatically every time you boot add the following lines

```
w1-gpio
w1_therm
```

into `/etc/modules` using `sudo nano /etc/modules` so they get loaded automatically next time you restart your Raspberry.

The IR receiver / LiRC

Setup your IR receiver

First, we'll need to install and configure LIRC to run on the RaspberryPi:

```
sudo apt-get install lirc
```

You have to modify two files before you can start testing the receiver and IR LED.

1) Add this to your `/etc/modules` file:

```
lirc_dev
lirc_rpi gpio_in_pin=22 gpio_out_pin=25
```

2) Change your /`etc/lirc/hardware.conf` file to:

```
########################################################
# /etc/lirc/hardware.conf
#
# Arguments which will be used when launching lircd
LIRCD_ARGS="--uinput"

# Don't start lircmd even if there seems to be a good config file
# START_LIRCMD=false

# Don't start irexec, even if a good config file seems to exist.
# START_IREXEC=false

# Try to load appropriate kernel modules
LOAD_MODULES=true

# Run "lircd --driver=help" for a list of supported drivers.
DRIVER="default"
# usually /dev/lirc0 is the correct setting for systems using udev
DEVICE="/dev/lirc0"
MODULES="lirc_rpi"

# Default configuration files for your hardware if any
LIRCD_CONF=""
LIRCMD_CONF=""
########################################################
```

Now restart lircd so it picks up these changes:

```
sudo /etc/init.d/lirc stop
sudo /etc/init.d/lirc start
```

Test your IR receiver

First, we'll need to stop processes that might lock the IR device

```
sudo /etc/init.d/lirc stop

Than point your remote control towards the receiver and run the
following command:

mode2 -d /dev/lirc0
```

You should see something like this:

```
space 16300
pulse 95
space 28794
pulse 135
space 7085
pulse 85
space 2903
```

Testing the IR LED

You're going to need to either find an existing LIRC config file for your remote control or use your IR receiver to generate a new LIRC config file. To do this, read the documentation on the irrecord application that comes with LIRC.

When using irrecord it will ask you to name the buttons you're programming as you program them. Be sure to run irrecord --list-namespace to see the valid names before you begin.

Use these commands to generate a remote configuration file:

```
# Stop lirc to free up /dev/lirc0
sudo /etc/init.d/lirc stop

# Create a new remote control configuration file (using /dev/lirc0)
and save the output to ~/lircd.conf
irrecord -d /dev/lirc0 ~/lircd.conf

# Make a backup of the original lircd.conf file
sudo mv /etc/lirc/lircd.conf /etc/lirc/lircd_original.conf

# Copy over your new configuration file
sudo cp ~/lircd.conf /etc/lirc/lircd.conf


# Start up lirc again
sudo /etc/init.d/lirc start
```

Addition information

Extra information on this subject can be found in these location(s):

http://aron.ws/projects/lirc_rpi/

The 433Mhz send module

Note	The data input to the 433Mhz RF transmitter is connected to SDA=GPIO3=physical pen #3
Note	First install wiringPi, most software needs it.

The 433 send module is generic and needs software for specific devices. To setup the modules do the following:

```
mkdir ~/433test
cd ~/433test
wget https://github.com/floresboy/RasPiggy/blob/master/lights.zip
unzip lights.zip
cd lights
```

Note	Change the int pin_out = 15; into int pin_out = 8;

Compile the version of your hardware:

KlikAanKlikUit

g++ -o kaku kaku.cpp -I/usr/local/include -L/usr/local/lib -lwiringPi

Action

g++ -o action action.cpp -I/usr/local/include -L/usr/local/lib -lwiringPi

Blokker

g++ -o blokker blokker.cpp -I/usr/local/include -L/usr/local/lib -lwiringPi

Elro

g++ -o elro elro.cpp -I/usr/local/include -L/usr/local/lib -lwiringPi

Test it by:

```
usage: ./elro systemcode socket state
example: ./elro 5 D on
```

ELRO controllable outlets

Download 433send-WiPi8.cpp and put in (for example) the WiringPi example directory (~/wiringPi/examples)

```
Wget
https://raw.githubusercontent.com/floresboy/RasPiggy/master/433send
-WiPi8.cpp
```

Note Change the key[ ] variable according to the dipswitches on your Elro switches.

Compile it:

```
g++ -o 433send 433send-WiPi8.cpp —I/usr/local/include  -
L/usr/local/lib -lwiringPi
```

Usage: (Execute as root!)

```
sudo ./433send -d <device number> -s 1|0
```

where -d: device number and -s: 1 = on, 0 = off.

Devices:

A = 1

B = 2

C = 4

D = 8

E = 16

Example:

`sudo ./433send -d 4 -s 1` (turn on device C)

Addition information

Extra information on this subject can be found in these location(s):

http://www.raspberrypi.org/forums/viewtopic.php?f=37&t=66946

http://weejewel.tweakblogs.net/blog/8665/lampen-schakelen-met-een-raspberry-pi.html (in Dutch)

DHT-22

Start by grabbing the Adafruit code from Github

```
git clone git://github.com/adafruit/Adafruit-Raspberry-Pi-Python-
Code.git
cd Adafruit-Raspberry-Pi-Python-Code
cd Adafruit_DHT_Driver
```

After this you can already try to get a reading from the sensor, by running the following command:

```
sudo ./Adafruit_DHT 22 17
```

where 22 is the type of sensor (DHT22 of DHT11) and 17 is GPIO pin

example results look like:

```
Adafruit_DHT 22 17
Using pin #17
Data (40): 0x1 0x58 0x1 0x1e 0x78
Temp =  28.6 *C, Hum = 34.4 %
```

Addition information

Extra information on this subject can be found in these location(s):

http://sharedmemorydump.net/post/2013-07-20-adding-a-dht-sensor-to-the-raspberry-pi

Trademarks en copyrights