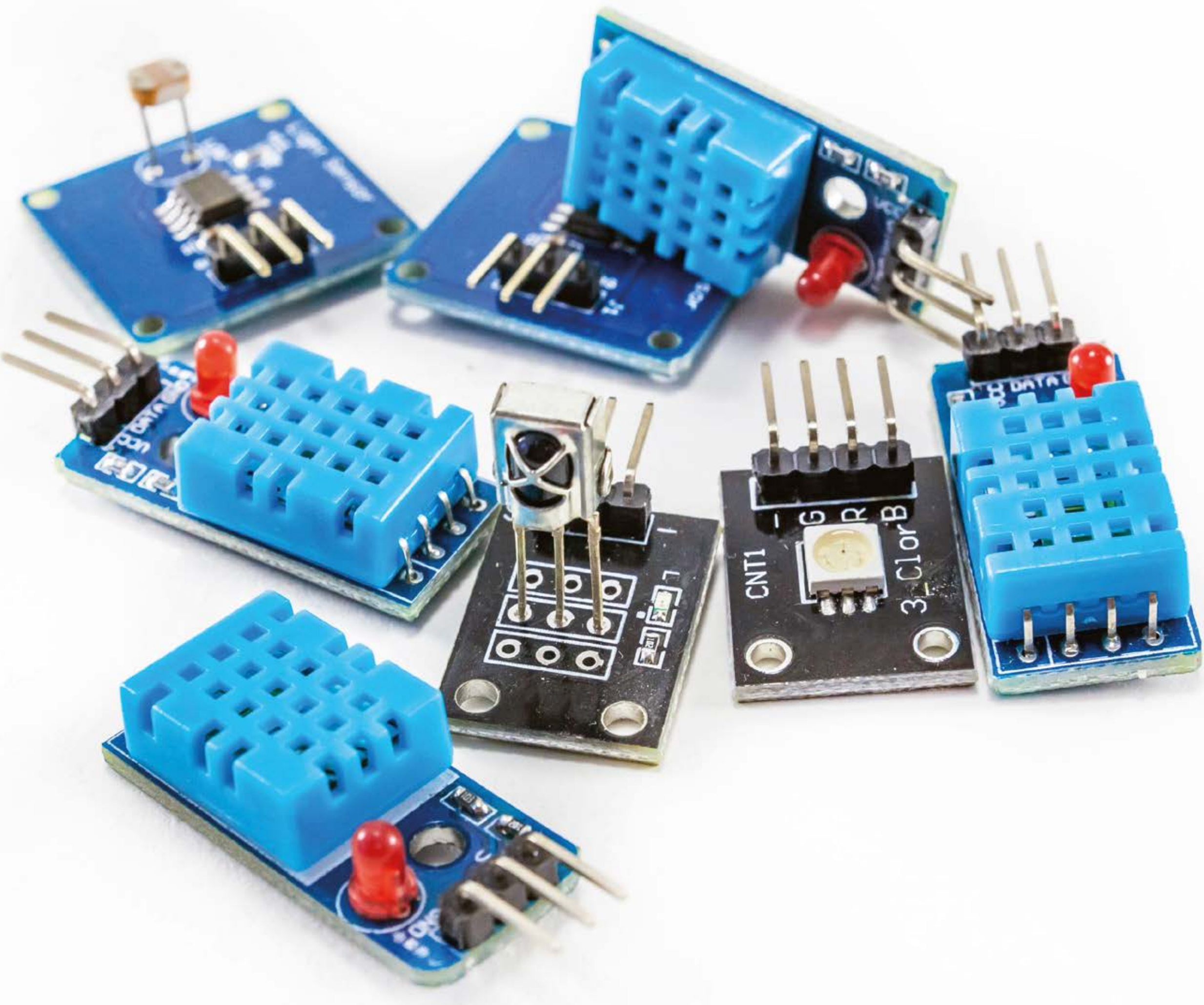
40+ Projects using Arduino, Raspberry Pi and ESP32

The Ultimate Compendium of Sensor Projects



Dogan Ibrahim

elektor

LEARN DESIGN SHARE

The Ultimate Compendium of Sensor Projects

40+ Projects using Arduino, Raspberry Pi and ESP32

Dogan Ibrahim

elektor
LEARN DESIGN SHARE

● This is an Elektor Publication. Elektor is the media brand of
Elektor International Media B.V.
78 York Street, London W1H 1DP, UK
Phone: (+44) (0)20 7692 8344

● Acknowledgements
The author would like to express his thanks to Ferdinand te Walvaart of Elektor for the valuable suggestions he made throughout the duration of the preparation of this book. The author would like to thank also to his wife Nadire for her encouragement, motivation, and for being patient with me while working on this book.

● British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

● **ISBN 978-1-907920-78-3**
EISBN 978-3-89576-354-0
EPUB 978-3-89576-355-7

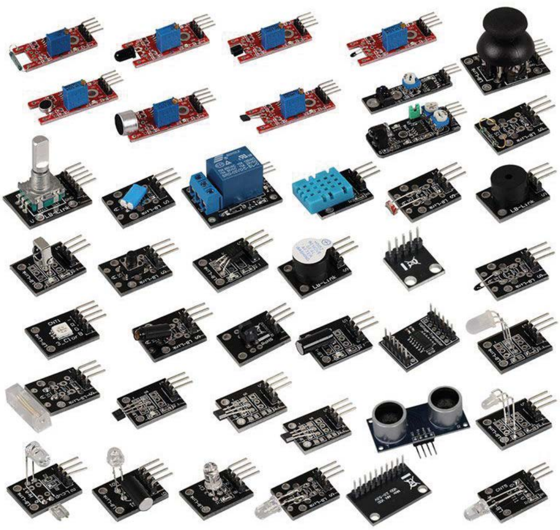
Prepress Production: D-Vision, Julian van den Berg
First published in the United Kingdom 2019
Printed in the Netherlands by Wilco

Preface

Sensors are devices or components that detect events or changes in their environments and send information to other electronics, frequently to microcontroller systems. Sensors are used in everyday life, to measure such things as temperature, humidity, pressure, wind and rain, touch, light levels, liquid levels, altitude, force, and many more.

Although some sensors have digital outputs, most sensors used in everyday life have analog outputs, usually in the form of voltages that are proportional to a measured quantity. This output voltage is normally fed to the input port of a microcontroller for processing. For example, the output of an analog temperature sensor is connected to an analog input port (an analog-to-digital converter) of a microcontroller. The microcontroller reads the temperature as a digital value and converts it into real physical temperature, which is then displayed or used to control the temperature of a machine or room.

This book is about using the sensors found in the sensor kit. Altogether, there are 40 sensors distributed with the kit. Some sensors provide analog outputs while others have digital outputs, and some have both analog and digital outputs. The kit includes sensors to measure temperature, humidity, atmospheric pressure, light intensity, and sound. There are also 2 and 3 colour LEDs, tilt switches, magnetic switches, relay, reed switches, piezo buzzer, button, joystick, obstacle detector, heartbeat sensor, analog-digital converter, voltage translator, vibrations switch, etc.



40 Sensors All-in-1 Kit

The book is intended to teach how to use sensors with the popular microcontroller development systems: Arduino Uno, ESP32 DevKitC, and Raspberry Pi. Example projects are given to show how to use the sensors with these microcontrollers. The programs can be modified for other microcontrollers, such as PIC, STM32, Banana Pi, CubieBoard, Beaglebone, etc. All the projects given in the book were built using a standard size breadboard, and they were fully tested and were all working. The projects are described with the following sub-headings:

- Project description
- Aim of the project
- Block diagram
- Circuit diagram
- Program listing

The operation of each sensor and each program listing are described in detail so that the readers will have no difficulty in either constructing or expanding a given project. Some projects make use of more than one sensor from the kit. It is recommended that readers follow the projects in the given order since some of the software tools used in some projects depend on the installation of these tools in an earlier project.

Full program listings of the projects with many comments are available at the Elektor website of the book and readers should be able to copy and use these programs without having to make modifications.

Arduino Uno and ESP32 DevKitC programs are based on using the Arduino IDE with the C language. Raspberry Pi projects use the Python programming language. Although the Arduino based projects specify Arduino Uno as the development board, these projects should also work with other Arduino development boards, such as Arduino Mega, Arduino Nano, etc.

I hope the readers find the book useful and enjoy experimenting with the various sensors.

Dogan Ibrahim
June 2019

Chapter 1 • Arduino Uno hardware interface and project development

1.1 Overview

In this book, we will use the Arduino Uno board in the Arduino based sensor projects. This Chapter shows the location of the various components on the Arduino Uno board and also describes the hardware interface to the external world. A simple project is given in this Chapter to make the reader familiar with developing programs with the Arduino Uno. The first program flashes a two colour LED alternately every second.

1.2 Arduino Uno board

Figure 1.1 shows the Arduino Uno board in close-up with the major components marked. The pin definitions are as follows (see Figure 1.2).

A0 – A5:	Analog input ports
0 – 13:	Digital input/output ports
~3,~5,~6,~9,~10,~11:	PWM output ports
0,1:	UART RX/TX pins. LEDs labelled TX, RX will flash when data is transmitted or received respectively
GND:	Power supply ground pin
5V:	Regulated +5V output
3.3V:	Regulated +3.3V output
Vin:	Voltage input (instead of using Power In or USB). The voltage must be in the range 7-12V. It is regulated to +5V internally. This pin can also be used as a voltage output (if power is supplied using Power In or USB port). The voltage is a copy of the voltage supplied through Power In or the USB port.
IORef:	Used by external shield boards to know if they should operate as +3.3V or as +5V devices
Power In:	Power supply Barrel Jack pin (6V to 12V)
USB port:	Power and data port (connect to computer)
User LED:	Onboard LED connected to output port 13 (can be used for testpurposes)

Notice that when the Arduino Uno is powered by the USB port (e.g. from a computer) the maximum current capacity is around 400mA for the +5V pin and 150mA for the +3.3V pin. When powered by an external source, the maximum current for the 5V pin is around 900mA and 150mA for the +3.3V pin. Any current drawn from the +3.3V goes through the +5V pin. Therefore, you have to take this into account when powering external devices.

The absolute maximum current for any I/O pin is specified as 40mA (it is however recommended not to exceed 20mA). The absolute total current from all the I/O pins is 200mA.

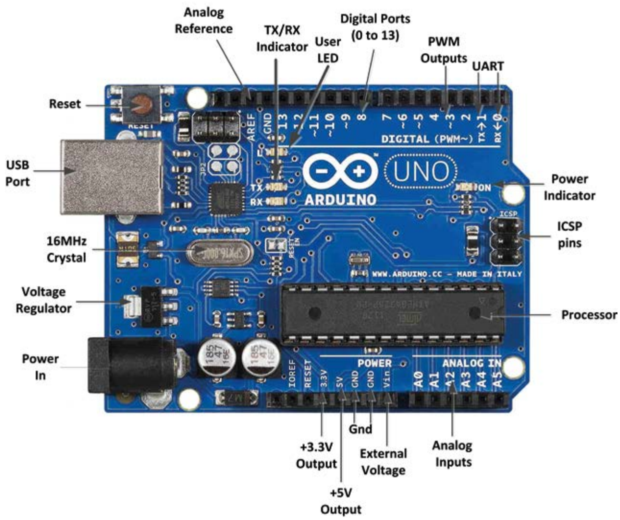


Figure 1.1 Arduino Uno board

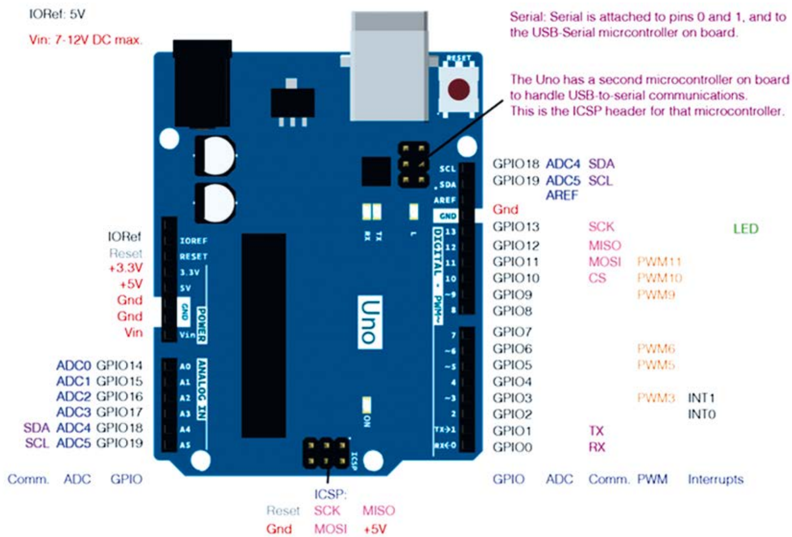


Figure 1.2 Arduino Uno pin layout

1.3 Arduino Uno program development

A nice feature of all Arduino boards is that they can all be programmed using the Arduino IDE. The latest version of Arduino IDE can be downloaded from the following web site. At the time of writing this book the latest version was 1.8.8:

https://www.arduino.cc/en/Main/Software

The steps to writing and upload a program to your Arduino Uno are as follows:

- Connect your Arduino Uno to the USB port of your computer
- Start the Arduino IDE on your computer
- Click **Tools -> Board** and select board type as **Arduino/Gerduino Uno**
- Click **Tools -> Port** and select the serial port that your Arduino Uno is connected to
- Write and then save your program
- Click **Sketch -> Verify/Compile** to compile your program. Make sure there are no compilation errors
- Click **Sketch->Upload** to upload the executable code to the program memory of your Arduino Uno

Two example projects are given in the following sections to make the reader familiar with the project development cycle using the Arduino Uno. It is assumed that the reader has some background in basic electronics and also some working knowledge of writing a high-level language.

1.4 Project – Flashing two colour LEDs
description: This is perhaps the easiest project you can design using your Arduino Uno. In this project, a two colour LED with common cathode is connected to the Arduino Uno. The LED colours are flashed alternately at a rate of one second.

Aim: The aim of this project is to show how to write, compile and upload a program to the Arduino Uno. The project additionally shows how to use some of the I/O and timing functions of the Arduino.

Sensor Used: Sensor KY-011 is used in this project. This sensor consists of a red and green colour LED housed in a package with a common cathode terminal. The sensor and its pins are shown in Figure 1.3. The sensor has 3 pins: red LED pin, green LED pin and GND pin. **The green LED pin is marked with letter S on the board for identification.** An LED is turned ON when logic 1 is applied to its pins. The basic specifications of this sensor are as follows:

Operating voltage:	2.0V to 2.5V
Operating current:	10mA (depends on the required brightness)
Wavelength:	green (571nm), red (644nm)
Luminous intensity (MCD):	green (20-40), red (40-80)

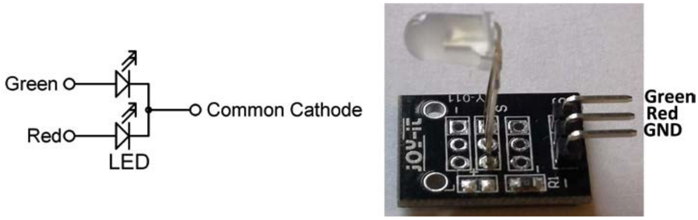


Figure 1.3 Sensor KY-011

Block diagram: The block diagram of the project is shown in Figure 1.4.

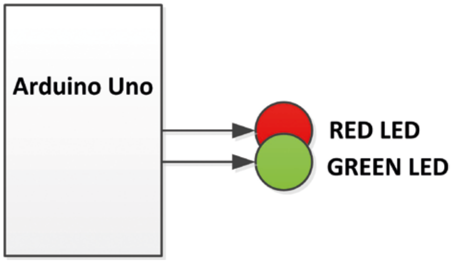


Figure 1.4 Block diagram of the project

Circuit diagram: The circuit diagram of the project is shown in Figure 1.5. The green and red LED pins are connected to port pins 2 and 3 of the Arduino Uno respectively through current limiting resistors. The GND pin is connected to Arduino Uno GND pin. The value of the current limiting resistor is calculated as follows:

The high voltage of an output pin is 5V. The forward voltage across an LED is approximately 2.0V (for the red LED this is 1.8V, and for the green LED this is 2.8V). Assuming that the forward current to the LED will be set to 10mA (reasonable brightness), then, the value of the current limiting resistor is:

R = (5V – 2V) / 10mA = 300 ohm, use 330 ohm resistor (you can use smaller value, e.g. 270 ohms for the green LED)

In Figure 1.5 the LED is operated in current sourcing mode where a high output from the I/O pin drives the LED. The LED can also be operated in current sinking mode where the other end of the LED is connected to +5V supply and not to the ground. In current sinking mode, the LED is turned ON when the I/O pin is at logic low.

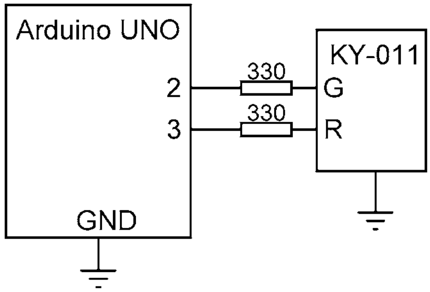


Figure 1.5 Circuit diagram of the project

Construction: The project is constructed on a breadboard as shown in Figure 1.6. Jumper wires are used to connect the KY-011 to digital port pins 2 and 3 and GND of the Arduino Uno.

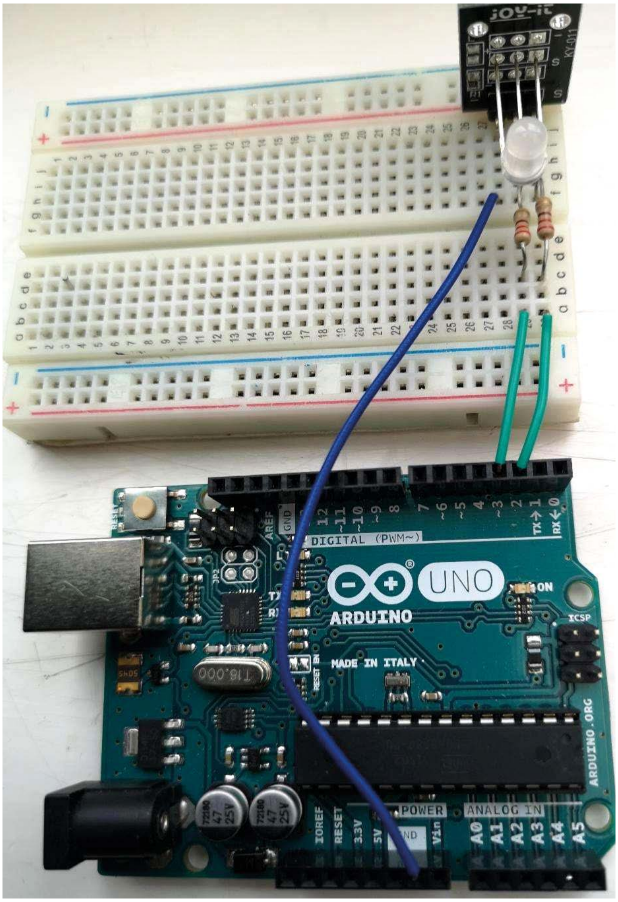


Figure 1.6 The project constructed on a breadboard

Program listing: The program is called **TWOCOLOUR** and the listing is shown in Figure 1.7. At the beginning of the program, port pins 2 and 3 are assigned to the green LED and red LED pins respectively. Inside the setup routine, port pins 2 and 3 where the LED pins are connected to are configured as outputs. Inside the main program loop, the two colours are turned ON and OFF alternately with a one-second delay between each output. **Verify/ Compile** to make sure that there are no errors and then **Upload** the program code to your Arduino Uno. You should see the two colours flashing at a rate of one second.

It is highly recommended by the author to comment the lines in your program and describe the operation being performed as shown in Figure 1.7. You should also include a heading and describe what the program does briefly. This makes it easier to understand and maintain the program at a later date. It also makes it easier for anyone else to understand the logic of the program when they read it.

```
/*******************************************************************
 *               FLASHING GREEN AND RED LEDs
 *               ===========================
 *
 * In this program the KY-011 sensor is used which has a green and
 * a red LED with a common cathode terminal. The green LED and the
 * red LED are connected to I/O pins 2 and 3 of the Arduino Uno
 * respectively. The program flashes the LEDs alternately at a rate
 * of one second.
 *
 * Author: Dogan Ibrahim
 * Date  : April 2019
 * File  : TWOCOLOUR
 ******************************************************************/
int GreenLED = 2;                              // Green LED pin
int RedLED = 3;                                // Red LED pin
#define ON HIGH
#define OFF LOW

void setup()
{
  pinMode(GreenLED, OUTPUT);                   // Set as output
  pinMode(RedLED, OUTPUT);                     // Set as output
}

void loop()
{
  digitalWrite(GreenLED, ON);                  // Turn ON green
  digitalWrite(RedLED, OFF);                   // Turn OFF red
  delay(1000);                                 // Wait 1 sec
  digitalWrite(GreenLED, OFF);                 // Turn OFF green
  digitalWrite(RedLED, ON);                    // Turn ON red
```

```
  delay(1000);                                    // Wait 1 sec
}
```

Figure 1.7 Program listing of the project

What we have learned: In this project, we have learned how to use the following Arduino functions:

int:	declares an integer variable
#define:	assigns value or text to a string
pinMode (port pin, mode):	used to configure an I/O port pin as input or output
digitalWrite(port pin, value):	used to output digital value (logic LOW or HIGH) to a port pin
delay(n):	creates a delay of **n** milliseconds

1.5 Summary

In this Chapter, we have seen the various components and the pin definitions of the Arduino Uno microcontroller. Additionally, a simple project is given to demonstrate how to design simple projects and also how to compile upload the program code to the program memory of the Arduino Uno microcontroller.

In the next Chapter, we will be looking at how to use the Raspberry Pi microcontroller in projects.

Chapter 2 • Raspberry Pi hardware interface and project development

2.1 Overview

In this book, we will be using a Raspberry Pi 3 Model B in our Raspberry Pi projects. This Chapter shows the location of the various components on the Raspberry Pi board and also describes the hardware interface to the external world. Setting up Wi-Fi and remote access to your Raspberry Pi computer, and becoming familiar with the Python programming environment are also described briefly. A simple project is given to familiarize the reader with the steps of designing a project. In this book, when we write Raspberry Pi 3 we actually mean Raspberry Pi 3 Model B.

2.2 The Raspberry Pi 3 board

Figure 2.1 shows the Raspberry Pi 3 board with the major components marked. Some details on each component are given in this section.

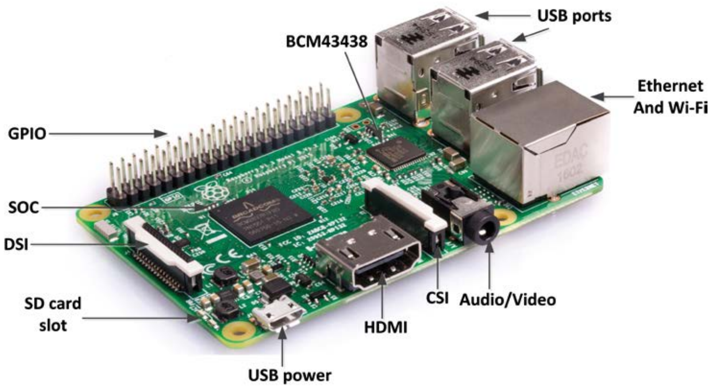


Figure 2.1 Raspberry Pi 3 board

USB ports: Raspberry Pi 3 has 4 USB ports to connect a mouse, keyboard, webcam, etc.

Ethernet and Wi-Fi: Although the Raspberry Pi 3 has built-in Wi-Fi, it can also directly be connected to a router through an Ethernet cable connected to this socket.

Audio/Video Jack: A headphone or a speaker can be connected to this 3.5mm socket. This socket also carries a composite video interface.

CSI: This is the Camera Serial Interface where a compatible Raspberry Pi camera can be attached here.

HDMI: A suitable monitor can be connected to this port. The port carries both audio and video signals.

USB power: A +5V 2A power supply should be connected to this USB socket to provide power to the Raspberry Pi 3.

SD card slot:	A micro SD card carrying the operating system must be attached to this slot.
DSI:	A suitable display can be connected to this Display Interface connector.
SOC:	This is the Broadcom BCM2837 System On Chip (SOC) which contains the 1.2GHz 64-bit quad-core ARM Cortex-A53 processor.
GPIO:	The General Purpose Input-Output port is 40 pins wide.
BCM43438:	This chip provides the Wi-Fi and Bluetooth to the Raspberry Pi 3.

2.3 Raspberry Pi 3 GPIO pin definitions

The Raspberry Pi 3 is connected to external digital electronic circuits and devices using its GPIO (General Purpose Input Output) port connector. This is a 2.54mm, 40-pin expansion header, arranged in a 2x20 strip as shown in Figure 2.2.

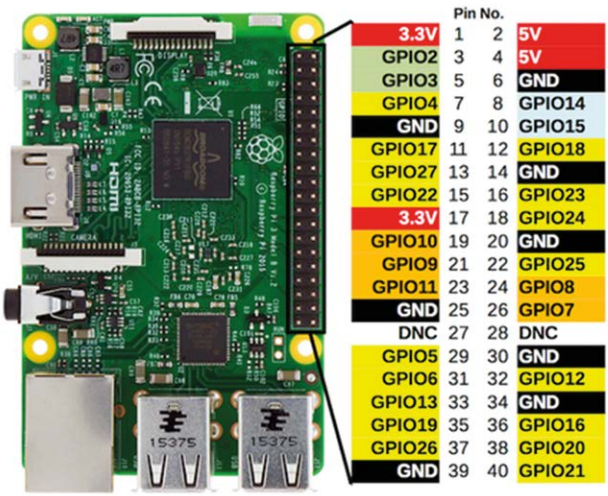


Figure 2.2 Raspberry Pi 3 GPIO pins

When the GPIO connector is at the far side of the board, the pins at the bottom, starting from the left of the connector are numbered as 1, 3, 5, 7, and so on, while the ones at the top are numbered as 2, 4, 6, 8 and so on.

The GPIO provides 26 general purpose bi-directional I/O pins. Some of the pins have multiple functions. For example, pins 3 and 5 are the GPIO2 and GPIO3 input-output pins respectively. These pins can also be used as the I^2C bus I^2C SDA and I^2C SCL pins respectively. Similarly, pins 9,10,11,19 can either be used as general purpose input-output, or as the SPI bus pins. Pins 8 and 10 are reserved for UART serial communication.

Two power outputs are provided: +3.3V and +5.0V. The GPIO pins operate at +3.3V logic levels (not like many other computer circuits that operate with +5V). A pin can either be an input or an output. When configured as an output, the pin voltage is either 0V (logic

0) or +3.3V (logic 1). Raspberry Pi 3 is normally operated using an external power supply (e.g. a mains adapter) with +5V output and minimum 2A current capacity. A 3.3V output pin can supply up to 16mA of current. The total current drawn from all output pins should not exceed the 51mA limit. Care should be taken when connecting external devices to the GPIO pins as drawing excessive currents or short-circuiting a pin can easily damage your Pi. The amount of current that can be supplied by the 5V pin depends on many factors such as the current required by the Pi itself, current taken by the USB peripherals, camera current, HDMI port current, and so on.

When configured as an input, a voltage above +1.7V will be taken as logic 1, and a voltage below +1.7V will be taken as logic 0. Care should be taken not to supply voltages greater than +3.3V to any I/O pin as large voltages can easily damage your Pi. The Raspberry Pi 3, like others in the family, has no over-voltage protection circuitry.

2.4 Setting up the Wi-Fi and remote access

It is very likely that you will want to access your Raspberry Pi 3 remotely from your desktop or laptop computer. The easiest option here is to enable Wi-Fi on your Pi computer and then access it from your computer using the SSH client protocol. This protocol requires a server and a client. The server is your Pi computer and the client is your desktop or laptop computer. In this section, we will see how to enable the Wi-Fi on your Pi computer and how to access it remotely.

Setting up Wi-Fi

To enable the Wi-Fi on your Pi, the steps are as follows:

- Click on the Wi-Fi icon which is a pair of red crosses at the top right-hand side of the screen

- Select your Wi-Fi router from the displayed list (see Figure 2.3)

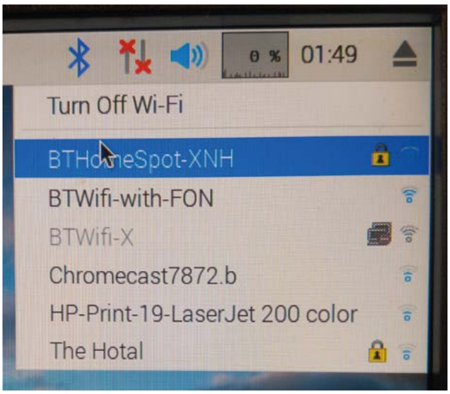


Figure 2.3 Select your Wi-Fi from the list

- Enter the password for your Wi-Fi router

- The WiFi icon should become a typical Wi-Fi image. If you click on the icon now you should see a green tick next to the selected router as shown in Figure 2.4.

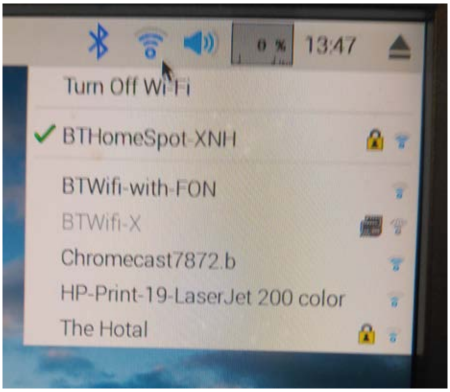


Figure 2.4 Connected to the Wi-Fi successfully

- To see the IP address of your Wi-Fi connection, place the mouse over the Wi-Fi icon as shown in Figure 2.5. In this example, the IP address was 192.168.1.84

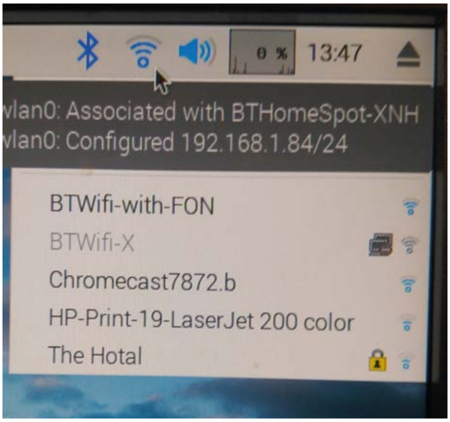


Figure 2.5 IP address of our connection

Remote Access

The program we will be using to access our Raspberry Pi 3 is called **Putty** with the SSH protocol. The steps to download and use Putty are as follows:

- Download Putty from the following link (or search Google for "Download Putty")

  http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

- For security reasons, the SSH protocol is disabled by default on a new operating system. To enable it, click on the **Applications** menu at the top left of the screen, click **Accessories**, and then click **Terminal** (see Figure 2.6)

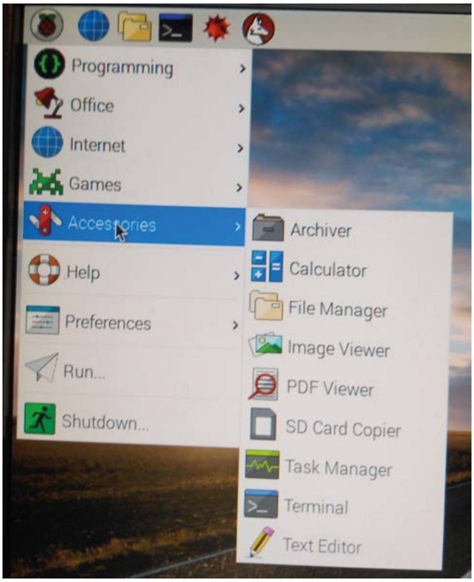


Figure 2.6 Access the Terminal menu

- You should now be in the Raspberry Pi 3 command prompt. Type:

```
sudo raspi-config
```

- to go into the configuration menu and select **Interface Options**. Go down to **P2 SSH** and enable SSH as shown in Figure 2.7

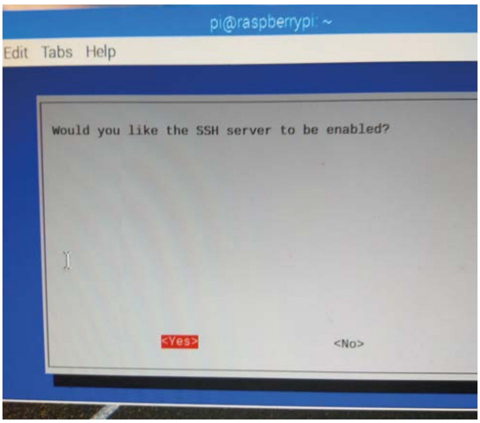


Figure 2.7 Enable the SSH server

- Click **<Finish>** to exit the configuration menu. You should now be back in the command mode, identified by the prompt:

```
pi@raspberrypi:~ $
```

- Putty is a standalone program and there is no need to install it. Simply double click to run it. You should see the Putty startup screen as in Figure 2.8.

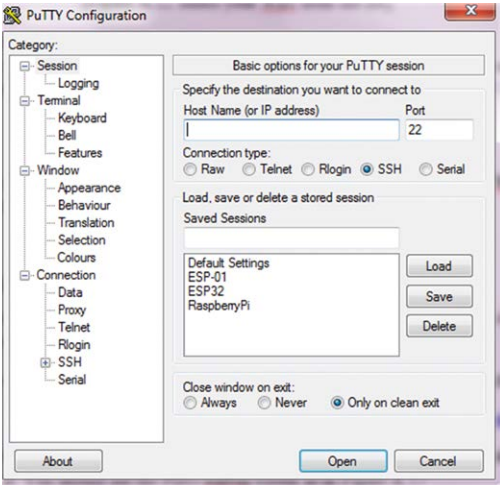


Figure 2.8 Putty startup screen

- Make sure that the Connection type is SSH and enter the IP address of your Raspberry Pi 3. Click Open as shown in Figure 2.9.

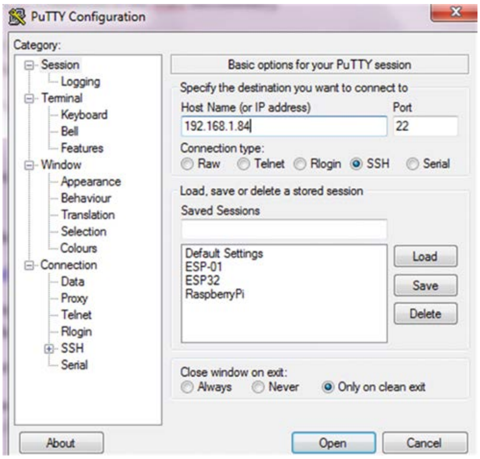


Figure 2.9 Enter the IP address

- The message shown in Figure 2.10 will be displayed on the PC screen the first time you access the Raspberry Pi 3. Click Yes to accept this security alert.

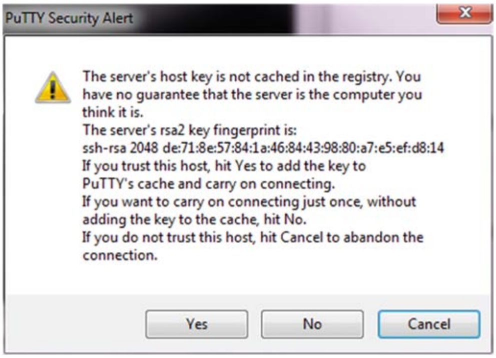


Figure 2.10 Click Yes to accept

- You will then be prompted for the username and password. The default values are:

  Username: **pi**
  Password: **raspberry**

- After a successful login, you should see the Raspberry Pi command prompt as in Figure 2.11.

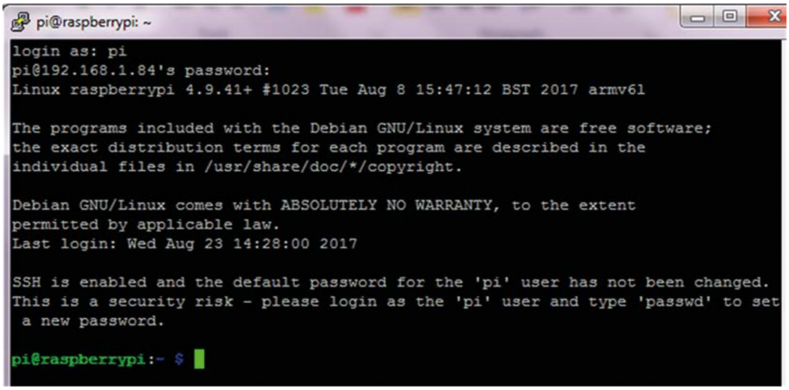


Figure 2.11 Successful login

- To change your password, enter the following command:

```
passwd
```

- To restart the Raspberry Pi ZW enter the following command:

```
sudo reboot
```

- To shut down the Raspberry Pi ZW enter the following command. Never shut down by pulling the power cable as this may result in the corruption or loss of files:

```
sudo shutdown -h now
```

2.5 Shutting down or rebooting in GUI mod

You must always shut down your Pi computer properly. To shut down while in the GUI mode, follow the steps given below:

- Click Applications menu (top left corner)
- Click Shutdown (see Figure 2.12)
- Click Shutdown (or Reboot as required)

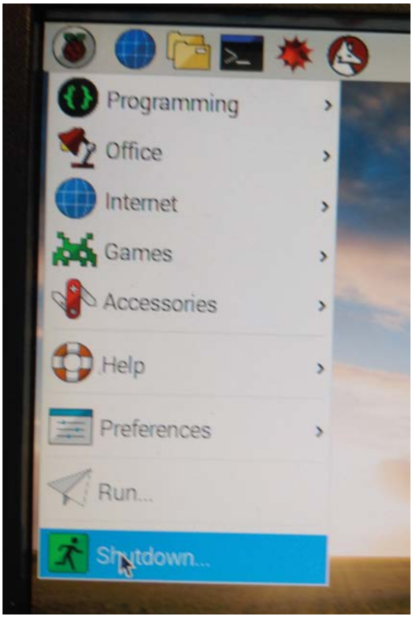


Figure 2.12 Shutdown or reboot in GUI mode

2.6 Remote access of the desktop

If you will be using your Raspberry Pi 3 with local keyboard, mouse, and monitor you can skip this section. If on the other hand, you want to access your Desktop remotely over the network, you will find that SSH services cannot be used. The easiest and simplest way to access your Desktop remotely from a computer is by installing the VNC (Virtual Network Connection) client and server. The VNC server runs on your Pi and the VNC client runs on your computer. The steps to install and use the VNC are given below:

- Connect to your Pi computer using SSH as explained earlier. Then enter the following command to install a program called **TightVNC** server on your Pi computer. You will see many lines of messages. Make sure there are no error messages:

```
sudo apt-get update
sudo apt-get install tightvncserver
```

- Run the VNC server on your Pi computer by entering the following command:

```
vncserver :1
```

- You will be prompted to enter and verify a password. This will be the password you will be using to access the Desktop remotely (see Figure 2.13).

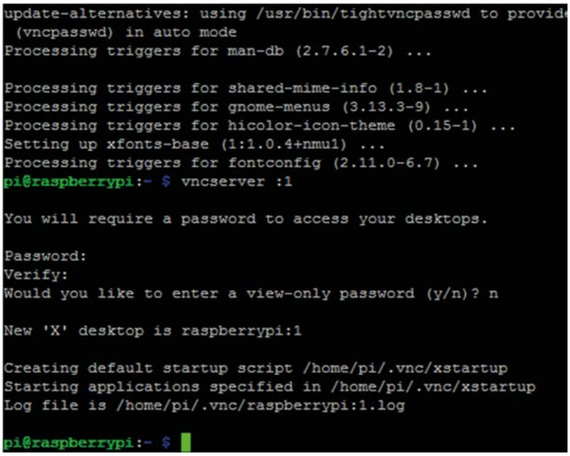


Figure 2.13 Enter a password for the VNC server

- The VNC server is now running on your Pi computer. The only command you need to enter on your Pi computer to start the VNC server is:

```
vncserver :1
```

- We must now set up a VNC client on our laptop (or desktop). There are many VNC clients available, but the recommended one which is compatible with TightVNC is the VNC Viewer, which can be downloaded from the following link. Notice that this program is not free of charge, but a 30-day free trial version is available. You should register to get a trial license and then apply this license to the software to use free of charge for 30 days:

  http://www.realvnc.com

  Download the VNC Viewer program into a suitable directory on your computer.

  Double click to install it and enter the required license. Start the **VNC Viewer** program by double-clicking its icon in your desktop. Enter the IP address of your Raspberry Pi 3 followed by **1** as shown in Figure 2.14 and click **Connect**.

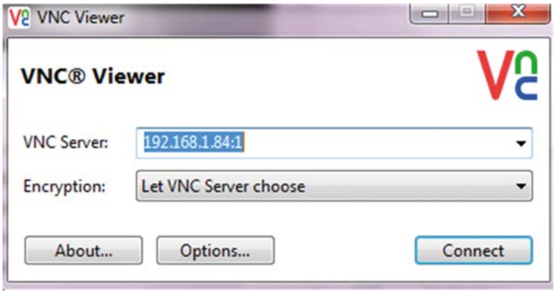


Figure 2.14 Enter the IP address

Enter the password selected previously. You should now see the Raspberry Pi 3 Desktop displayed on your laptop (or desktop) computer as in Figure 2.15 and you can access all of the Desktop applications remotely.

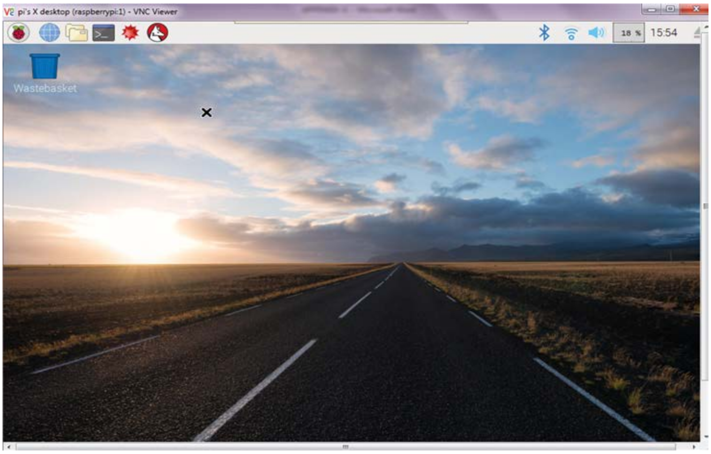


Figure 2.15 Raspberry Pi 3 Desktop displayed on the laptop

2.7 Creating and running a Python program

We will be programming our Raspberry Pi 3 using the Python programming language. It is worthwhile to look at the creation and running of a simple Python program on our Pi computer. In this section, we will display the message **Hello From Raspberry Pi 3** on our PC screen.

As described below, there are 3 methods that we can create and run Python programs on our Raspberry Pi 3:

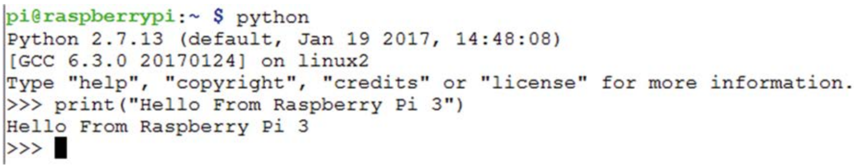
Method 1 – Interactively from Command Prompt

In this method, we will log in to our Raspberry Pi 3 using the SSH and then create and run our program interactively. This method is excellent for small programs. The steps are as follows:

- Login to the Raspberry Pi 3 using SSH
- At the command prompt enter **python**. You should see the Python command mode which is identified by three characters **>>>**
- Type the program:

```
print ("Hello From Raspberry Pi 3")
```

- The required text will be displayed interactively on the screen as shown in Figure 2.16



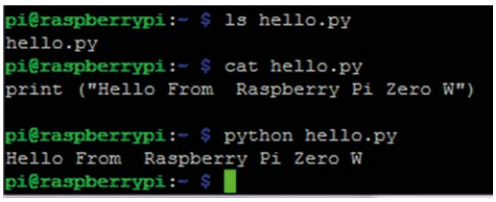
```
pi@raspberrypi:~ $ python
Python 2.7.13 (default, Jan 19 2017, 14:48:08)
[GCC 6.3.0 20170124] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello From Raspberry Pi 3")
Hello From Raspberry Pi 3
>>>
```

Figure 2.16 Running a program interactively

Method 2 – Create a Python File in Command Mode

In this method, we will log in to our Raspberry Pi 3 using the SSH as before and then create a Python file. A Python file is simply a text file with the extension **.py**. We can use a text editor, e.g. the **nano** text editor to create our file. In this example, a file called **hello.py** is created using the **nano** text editor. Figure 2.17 shows the contents of file hello.py. This figure also shows how to run the file under Python. Notice that the program is run by entering the command:

```
>>> python hello.py
```



```
pi@raspberrypi:~ $ ls hello.py
hello.py
pi@raspberrypi:~ $ cat hello.py
print ("Hello From  Raspberry Pi Zero W")

pi@raspberrypi:~ $ python hello.py
Hello From  Raspberry Pi Zero W
pi@raspberrypi:~ $
```

Figure 2.17 Creating and running a Python file

Method 3 – Create a Python File in GUI mode

In this method, we will log in to our Raspberry Pi 3 using the VNC and create and run our program in GUI mode. The steps are given below:

- Click Applications menu
- Click Programming and select Python 2 or Python 3 (see Figure 2.18)

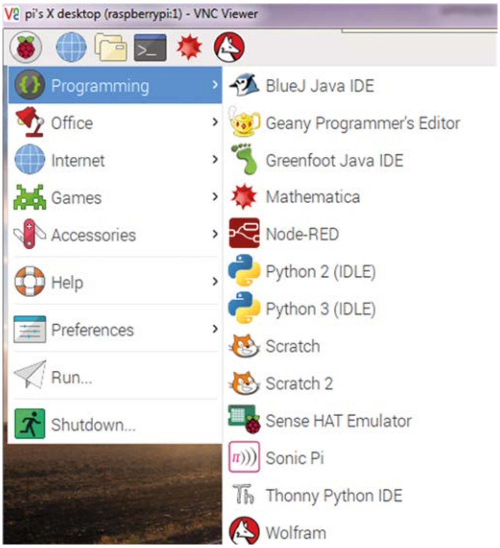


Figure 2.18 Select Python 2 programming

- You should see the Python command mode, identified by characters >>>
- Click **File** and then click **New File** and write your program
- Save the file by giving it a name (e.g. hello2)
- Run the program by clicking **Run** and then **Run Module** as shown in Figure 2.19

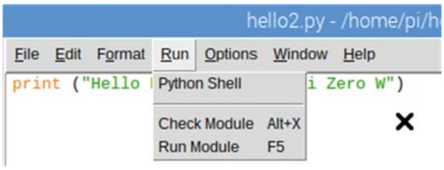


Figure 2.19 Run the program

- A new screen will be shown with the output of the program displayed as in Figure 2.20

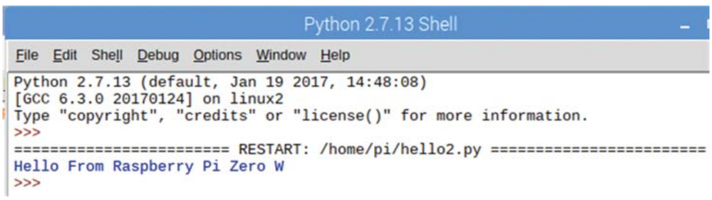


Figure 2.20 Output of the program