**A Vision Module for Visually Impaired People by Using Raspberry Pi Platform – Raspberry Pi**

**OBJECTIVE**

* The objective of our project to create vision module for visually impaired people.By using deep learning and Convolutional Neural Network with pre-trained model MobileNet to recognize the object with more accuracy at real-time.

**ABSTRACT**

This project investigates methods and procedures to construct an efficient system to assist blinds in their everyday life. In particular, various technologies that can be utilized to build a wearable system are examined. The machine vision and the communication component of the blind navigation and guidance is designed not only to map the surroundings environment but also to determine a safe path to a desired destination. This work highlights the importance and also provides the instructions to blinds for efficient navigation and safe guidance by incorporating object/pedestrian in real-time.

In this project, Object recognition is done by the Pre-trained model MobileNet for recognizing the object with more than 95% accuracy. The model is trained with more than lakhs of images to recognize the object. An object such as Person, chairs, TV Monitor, etc. USB Camera is interfaced with the Raspberry Pi for this application. It can also be done using IP Webcam or Pi camera.

**INTRODUCTION**

Safe navigation and detailed perception in unfamiliar environments are a challenging activity for the blind people. This paper proposes a cloud and vision-based navigation system for the blind. The goal of the system is not only to provide navigation, but also to make the blind people perceive the world in as much detail as possible and live like a normal person.

**EXISTING**

In the existing system, object recognition can be done using any of the Image processing Algorithm like SIFT, SURF. But that kind of techniques has lots of limitations, make more difficult to recognize the object.

**DISADVANTAGE**

* Sift, surf algorithms are very low in accuracy
* Can not recognize plane objects without edges
* Cannot properly classify between similar objects
* Uses only single image as dataset

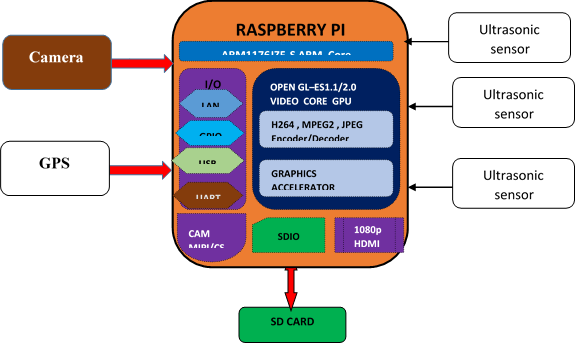
**PROPOSED SYSTEM**

In the proposed system, deep learning is used. Among that Convolutional Neural Network is used with pre-trained model MobileNet to recognize the object with more accuracy at real-time

**ADVANTAGE**

* Recognition is real-time
* Convolutional Neural Network is very advanced deep learning algorithm
* Number of data set is high hence accuracy is also high
* Uses simple USB camera for video capture
* No 3D camera is used
* cost effective

**BLOCK DIAGRAM**

****

**CONNECTION DESCRIPTION**

Raspberry Pi is booted with the SDCard, with libraries installed like Keras, Tensorflow backend, numpy, etc. USB camera is interfaced with the Raspberry pi to make it as the real-time object recognition application.

**HARDWARE REQUIREMENT**

* Raspberry Pi
* USB Camera
* Ultrasonic sensor
* GPS module

**SOFTWARE REQUIREMENT**

* Raspberry pi OS: Raspbian stretch
* Programming platform: python 3 IDLE
* Programing language: python 3
* Library: OpenCV

**MODULES**

**RASSPBERRY PI**

* In this project we use Raspberry Pi is booted with the SDCard, with libraries installed like Keras, Tensorflow backend, numpy.
* raspberry pi is connected with sensor and camera processing for visually impaired people.

**CAMERA**

In this project camera is used for Object recognition is done by the Pre-trained model MobileNet for recognizing the object with more than 95% accuracy. The model is trained with more than lakhs of images to recognize the object.

An object such as Person, chairs, TV Monitor, etc.

USB Camera is interfaced with the Raspberry Pi for this application. It can also be done using IP Webcam or Pi camera**.**

**ULTRASONIC SENSOR**

* In this project ultrasonic sensor is used to find the obstacles for the visually impaired people.

**Hardware explanation**

**Raspberry Pi**

Raspberry Pi is a small single-board Computer developed in UK by Raspberry Pi foundation to promote the teaching of computer science in schools and in developing countries.

Original model become far more popular than anticipated sealing outside of its target market, for uses such as robots.

**3.2.1 History**

Raspberry Pi has mainly three generations Raspberry Pi 1, Raspberry Pi 2, Raspberry Pi 3 and also a reduce simple inexpensive Raspberry Pi zero.

* The first model of Raspberry Pi was launched in February 2012 i.e. Raspberry Pi 1 Model B followed by a simple inexpensive Model A.
* In April 2014 “Compute Model” for embedded application Raspberry Pi 1 model B+ improved versions of A and B was launched.
* In November 2015 with reduced I/O and GPIO Raspberry Pi zero came into market.
* In February 20-15 advance model with 40 GPIO pins, Ethernet, 4 USB slots Raspberry Pi 2 was launched.
* In February 2016 an upgraded model with inbuilt Bluetooth and Wi-Fi Raspberry Pi 3 Model B was launched.
* Recently in February 2017, “Raspberry Pi – Zero W” with in built Wi-Fi and Bluetooth come into the market.

**Features**

The heart of the Raspberry Pi is a Broadcom System on Chip (SOC) which includes ARM compatible CPU and on-chip graphic processing unit and Vediocore IV.

The key feature from First generation to the Third generation includes:

* CPU speed ranges from 700 MHz to 1.2 GHz.
* On board Memory (RAM) ranges from 256 MB to 1 GB.
* USB slot differs from 1 slot to USB slots.
* HDMI, composite video output and 3.5mm phone jack.
* Low level output is provided by GPIO pins which support common. protocols like I2C (inter-integrated circuit).
* Ethernet 8 Position 8 Contact (8P8C).

**3.2.3 Processor**

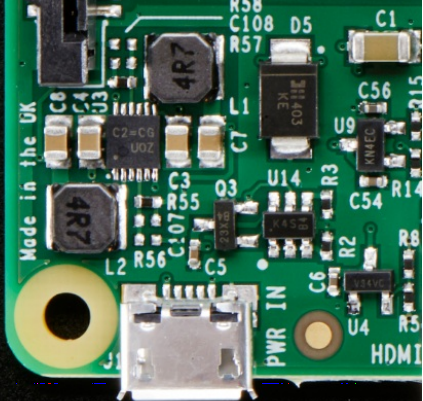
The processor at the heart of the Raspberry Pi is a Broadcom BCM28XX.

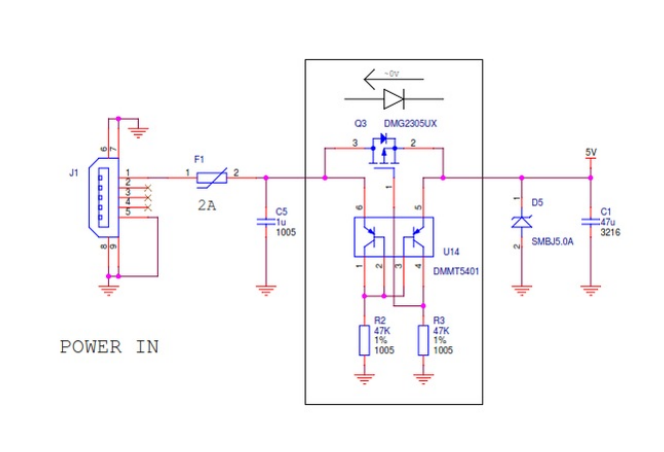
This is the Broadcom System on Chip (SOC) chip use in the Raspberry Pi.The processor from first to third generations include:

* Raspberry Pi 1: Broadcom BCM2835 SOC with 700MHz CPU speed, L2 cache of 128kb with ARM compatibility AR1176JZF-S (ARMv6) 32-bit RISC ARM.
* Raspberry Pi 2: Broadcom BCM 2836 SOC with 900MHz CPU speed, L2 cache of 256kb with 32-bit quad-core ARM cortex-A7 (ARMv7).
* Raspberry Pi 3: Broadcom BCM2837 SOC with 1.2GHz 64-bit quad-core –A53 with 512 kb shared L2 cache (64-bit instruction set ARMv8).

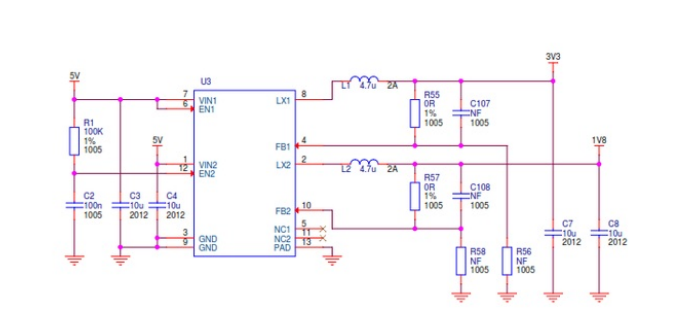
**RASPBERRY PI POWER SUPPLY**

**Model B+ Power Supply** To make the B+ more reliable and actually reduce the current draw, the power supply is completely redesigned.



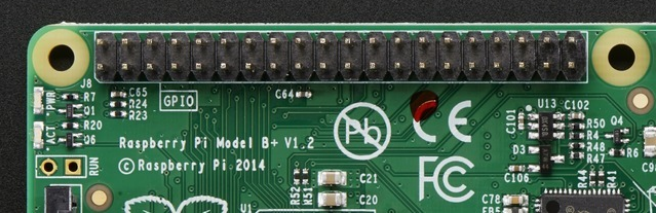


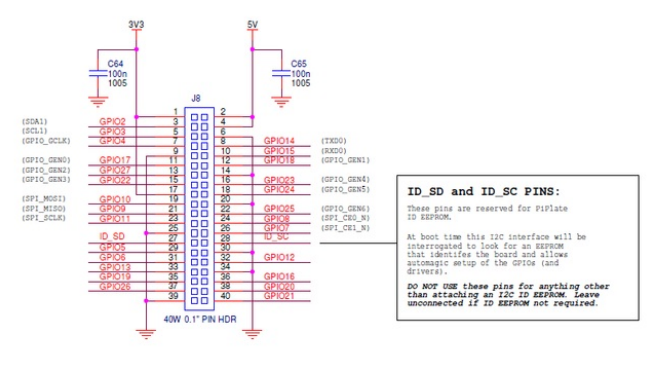
There's still the microUSB jack on the left, and the 1A fuse has been upgraded to a 2A fuse. There's also a DMG2305UX (http://adafru.it/dGU) P-Channel MOSFET. This acts as a polarity protection switch but is much lower 'drop-out' than a diode. It has only 52mW resistance so @ 2A its about 0.1V voltage drop. Most diodes would be at least 0.5V. Watch this great video about this technique here: To the right is a protection TVS diode (D5 part #SMBJ5) which protects from over-voltages. So not a lot has changed here (other than putting in a protection FET) There is a PNP-matched-pair action going on around the polarity FET, but its 3AM and I'm not 100% sure what it's for so I'll wait till I get some rest before doing any analysis. Let's look at the 3.3V & 1.8V supplies:



Instead of heat-spewing LDO (low dropout) regulators, we now have a dual buck converters. These are high efficiency converters that can take 5V down to 3.3V or 1.8V without as much heat loss. They're more expensive than LDO's but not terribly so! The input to the dual buck is 5V (VIN1 and VIN2) - there's no part number marked here for some reason but it has 12 pins, is a DFN-shaped part (I deal with DFN's all day so I can spot them), and has the marking code C2=CGU0G. with some searching around for a 12-DFN dual buck with 1.8V and 3.3V fixed outputs...

**Raspberry Pi Model B+ GPIO Port:**

****

****

First thing to notice, the top 26 pins of the 40-pin connector are the same as the original That means that most/many Pi Plates that plug into the Model B will plug into the B+ just fine. They wont sit in the same location - they'll be slid down just a bit but electrically-wise its the same.

New GPIOs

**RASPBERRY PI CONFIGURATION**

**Setting up Raspberry Pi**

As said earlier Raspberry Pi comes without any peripheral devices. The first thing to do is to unpack RasPi and protect it with an enclosure (Figure 3). Raspberry Pi can be installed to the protective enclosure without using any tools. The enclosure has plastic clips which are holding the Raspberry Pi in its place.

After Raspberry Pi has been installed to enclosure and well protected, all the nec-essary peripherals can be attached to it. Just like any other computer, Raspberry Pi needs some basic devices such as display which is connected via the HDMI cable, the mouse and the keyboard, and the internet connection cable.

Before plugging the power cable, MicroSD-card should be checked if it is flashed and prepared with an operating system. Also it is recommendable to create a backup folder of the MicroSD-card just in case of complications.

The MicroSD-card can be checked with a card-reader. The card-reader can be found from most of the laptops and desktop computers. Insert the MicroSD-card into the card-reader and check that there is something stored in the MicroSD-card. If everything looks good, take the MicroSD-card and plug it into the Raspberry Pi. Now the power cable can be connected.

Raspberry Pi does not have any kind of power switch so it will start up immediately when the power cable is connected to it. At the start up text starts to flow on the monitor and shortly after that there appears a configuration menu. The configura-tion menu is called Raspi-config (Figure 4). In Raspi-config it is possible to change some of the settings on Raspberry.

****

The most important settings that should be checked in Raspi-config are:

 Expand Filesystem, where it is necessary to check that RasPi can use the whole memory capacity of the MicroSD-card. Otherwise the memory can run out fast.

 Internationalisation Options, where it is possible to choose between differ-ent languages and the time zones.

 Advanced Options, if the internet cable has been plugged in, it is possible to update RasPi to the latest version available. (McManus, S. & Cook, M. 2013, 38.)

It is recommendable that users who do not have so much experience with Linux operating systems should choose the English language because then help and advice can be found more easily from the internet.

It is possible to get back to the Raspi-config and change the settings also after the first setup by typing the following command into the terminal:

sudo raspi-config

After making the changes on the Raspberry Pi's settings, the settings can be ac-cepted by choosing the Finish option. Now the terminal view should appear and it might be asking for the username and the password. The username in Raspian Wheezy is by default **pi** and the password should be **raspberry**. Notice that these are written in small letters. The Linux is letter case sensitive and it will recognize the difference between small and capital letters.

The next step is logging in to Raspberry and instead of the graphical environment there will be a command console flashing. However, the graphical environment, or so called desktop view, can be started by entering the command:

Startx

Now Raspberry will be loading for a while and a few seconds later there will ap-pear a more user friendly desktop view. It is recommended to learn how to use the command console as it makes some of the actions faster than doing them in the desktop view.

So far the basic configurations are made for the Raspberry. There might still be some things that are not working correctly. For instance, the keyboard layout might be defined to be in UK style which is the default keyboard layout setting on Rasp-berry Pi. This can be frustrating and annoying. The layout can be changed easily by opening the LXTerminal which opens the command console. Open the key-board file in the command console with the nano text editor by typing the following command:

sudo nano /etc/default/keyboard

The keyboard configuration file (Figure 5) will appear and it can be modified. The keyboard layout can be changed by replacing the XKBLAYOUT value as shown in Figure 5. After the file is edited it can be saved by pressing CTRL + O key combi-nation.

****

The keyboard configuration, setting the layout to the Finnish “fi”

**Controlling the GPIO pins with Python**

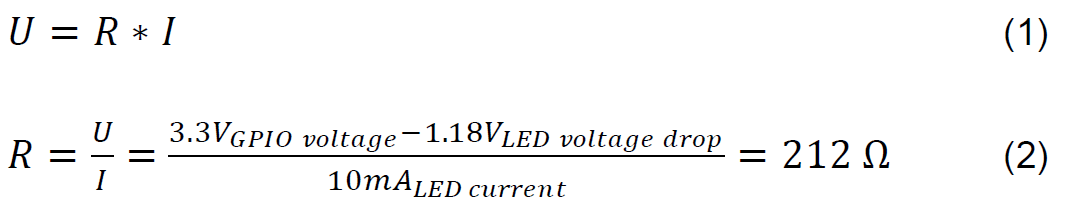
This chapter discovers the GPIO connector and how it can be used in controlling. The first experiments with the GPIO were to light up a LED (light-emitting diode) through the Python Shell.

The second experiment is little bit more complex and it demonstrates the control-ling loop of an heating element. The heating element will start to heat the room when room's temperature is getting below the pre-defined lower limit and stops heating when the temperature in the room reaches the second pre-defined upper limit.

**3.5.1 Controlling the LED with the GPIO**

This is the first experiment with the GPIO connector and it demonstrates how to use it in controlling. This experiment requires a LED and a resistor. The resistor's resistance can be calculated from the Ohm's law which is shown in Formula 1.

Defining the resistance from the Ohm's law:

****

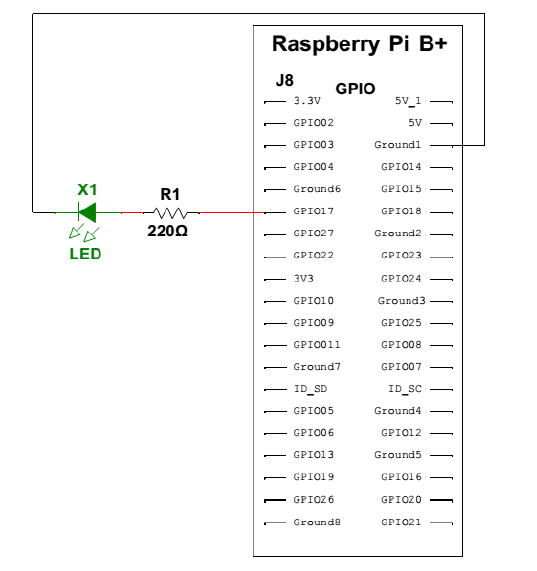
Where

 U is voltage

 R is resistance

 I is current

The resistors above 212 Ω are suitable and can be used for lightning the LED di-rectly from the GPIO. The wiring for the LED and resistor is shown in Figure 11.

****

After the wirings are done the Python library called python-rpi.gpio needs to be installed. This library allows controlling the GPIO pins. It can be installed with the following command:

sudo apt-get install python-rpi.gpio

When the installation is finished, open up a Python Shell from the terminal as root user and import the RPi.GPIO library.

import RPi.GPIO as GPIO

Next thing to do is to set the mode to use the pin numbers from the ribbon cable board and define one of the GPIO pins to be an output. For instance the GPIO 17:

GPIO.setmode(GPIO.BCM) # Ribbon cable board

GPIO.setup(17, GPIO.OUT) # Defines the GPIO17 to be output

Now it is possible to control the GPIO17 pin to high and low. The LED will light up when the pin 17 is set to high and when it is set to low the LED will turn off.

GPIO.output(17, GPIO.HIGH) # Turns the GPIO17 to high

GPIO.output(17, GPIO.LOW) # Turns the GPIO17 to low

**Taking advantage of Raspberry Pi's camera module**

This chapter is about the Pi NoIR camera module's installation to the Raspberry Pi, and observing the built in functions which are made for it. At the end of this chapter a Python script is created to take resized pictures. The pictures are named with current timestamp and saved to an own directory.

**3.6.1 Installing the Pi NoIR camera module**

The Raspberry Pi's NoIR camera module board comes in anti-static plastic bag. It is fast and easy to install. The camera module can be mounted to the protective case's cover, where is reserved slot for the camera. (Figure 12) It is screwed with two small screws, and the ribbon cable is connected to the Raspberry Pi's camera connection port. The connection port is located between the 3.5mm audio jack and the HDMI socket. The connection port's clip has to be pulled up before plugging the camera module's ribbon cable on its place After mounting the camera module, it is required to enable the camera module from the Raspi-config configuration tool and then Raspberry Pi has to be rebooted so that the changes will take effect.

**3.6.2 Taking the first pictures and videos with the Pi NoIR camera**

In Raspbian there are built in functions for the camera module. With these built in functions it is possible to take pictures and record videos, just to try out proper function of the camera module. One of these built in functions or commands is called “raspistill”.

raspistill -v -o first\_image.jpg

After typing the command above into terminal a preview window is started up. The preview window is running for 5 seconds, and then Raspberry takes the picture, and saves it to the file called first\_image.jpg. Parameters -v stands for verbose information during the run and with the -o parameter it is possible to give filename for the output file.

Other simple and useful parameters which can be added into raspistill command are:

– image width **-w**

– image height **-h**

– image quality **-q**

– flip the image vertically **-vf**

– flip the image horizontally **-hf**

– image rotation **-rot**

A complete parameter list can be found from the RaspiCam documentation. (RaspiCam Documentation. 2013. 5-18.)

**3.6.3 Creating a Python script for taking pictures**

First things to consider before creating the script which takes the picture and stores it automatically are: where the picture is stored, finding the right parameters for the picture so that the image quality and size does not suffer too much.

After a while, some limitations for the pictures are found. The size and quality are reduced to minimize the picture size on the hard drive. The Quality of 75% and the resolution of 1280x720 pixels are sufficient. With these parameters the picture size on the hard drive is around 500KB. That is good starting point, and trade-off be-tween picture quality and available space for picture saving.

All the pictures which are taken by the Python script will be saved to the own folder with current timestamp filename. The folder is located at /var/www/camera/. Apache2 is hosting the folder so that the pictures are available on the website.

Creating the script starts with placing the shebang information and importing the necessary libraries. These libraries are datetime, picamera and time.

#!/usr/bin/env/ python

import datetime

import picamera

import time

On the second step a function called takePicture should be defined. It does not take any input variables. The function consists of three parts. The first part is the general settings, where the location to the saved pictures and the filename are defined.

def takePicture():

location=*"/var/www/camera/"* #Location to the files

date=datetime.datetime.now() #Get current date

file\_name=date.strftime(*"%Y-%m-%d %H%M"*) #Format the string

The second part of the function is defining the settings for the picture size and it starts also the preview mode.

#configuration for the pictures

camera = picamera.PiCamera()

camera.resolution = (1280,720)

camera.start\_preview()

In the last part of the function, the preview mode is kept on for a certain time to warm up the camera. After the warm up time, the function captures the picture and saves it to the predefined location. The picture is named with current timestamp. At the end of the script the preview mode is stopped and the camera is closed.

time.sleep(2) #Camera warm up time

# Capture the picture and saved it with the current date

camera.capture(*"%s%s.jpg"* % (location,file\_name), quality=75)

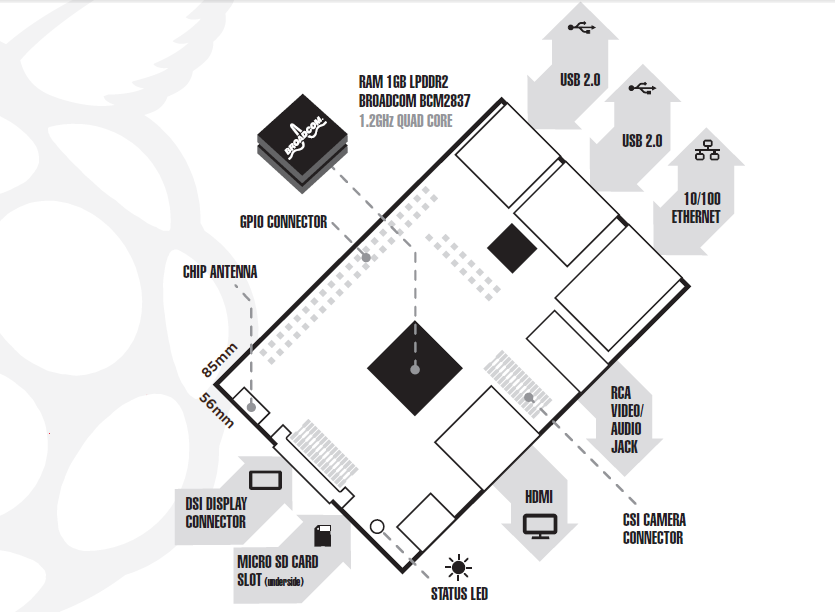
camera.stop\_preview()

camera.close()

**ARCHITECHTURE**

**ARM vs. x86**

The processor at the heart of the Raspberry Pi system is a Broadcom BCM2837 system-on-chip (SoC) multimedia processor. This means that the vast majority of the system’s components, including its central and graphics processing units along with the audio and communications hardware, are built onto that single component hidden beneath the 256 MB memory chip at the centre of the board.



It’s not just this SoC design that makes the BCM2837 different to the processor found in your desktop or laptop, however. It also uses a different instruction set architecture (ISA), known as ARM. The BCM2837 SoC, located beneath a Hynix memory chip Developed by Acorn Computers back in the late 1980s, the ARM architecture is a relatively uncommon sight in the desktop world. Where it excels, however, is in mobile devices: the phone in your pocket almost certainly has at least one ARM-based processing core hidden away inside. Its combination of a simple reduced instruction set (RISC) architecture and low power draw make it the perfect choice over desktop chips with high power demands and complex instruction set (CISC) architectures. The ARM-based BCM2837 is the secret of how the Raspberry Pi is able to operate on just the 5V 1A power supply provided by the onboard micro-USB port. It’s also the reason why you won’t find any heat-sinks on the device: the chip’s low power draw directly translates into very little waste heat, even during complicated processing tasks. It does, however, mean that the Raspberry Pi isn’t compatible with traditional PC software. The majority of software for desktops and laptops is built with the x86 instruction set architecture in mind, as found in processors from the likes of AMD, Intel and VIA. As a result, it won’t run on the ARM-based Raspberry Pi. The BCM2837 uses a generation of ARM’s processor design known as ARM11, which in turn is designed around a version of the instruction set architecture known as ARMv6. This is worth remembering: ARMv6 is a lightweight and powerful architecture, but has a rival in the more advanced ARMv7 architecture used by the ARM Cortex family of processors. Software developed for ARMv7, like software developed for x86, is sadly not compatible with the Raspberry Pi’s BCM2837—although developers can usually convert the software to make it suitable. That’s not to say you’re going to be restricted in your choices. As you’ll discover later in the book, there is plenty of software available for the ARMv6 instruction set, and as the Raspberry Pi’s popularity continues to grow, that will only increase. In this book, you’ll also learn how to create your own software for the Pi even if you have no experience with programming.

**Windows vs. Linux**

Another important difference between the Raspberry Pi and your desktop or laptop, other than the size and price, is the operating system—the software that allows you to control the computer. The majority of desktop and laptop computers available today run one of two operating systems: Microsoft Windows or Apple OS X. Both platforms are closed source, created in a secretive environment using proprietary techniques. These operating systems are known as closed source for the nature of their source code, the computer-language recipe that tells the system what to do. In closed-source software, this recipe is kept a closely-guarded secret. Users are able to obtain the finished software, but never to see how it’s made. The Raspberry Pi, by contrast, is designed to run an operating system called GNU/Linux—hereafter referred to simply as Linux. Unlike Windows or OS X, Linux is open source: it’s possible to download the source code for the entire operating system and make whatever changes you desire. Nothing is hidden, and all changes are made in full view of the public. This open source development ethos has allowed Linux to be quickly altered to run on the Raspberry Pi, a process known as porting. At the time of this writing, several versions of Linux—known as distributions—have been ported to the Raspberry Pi’s BCM2837 chip, including Debian, Fedora Remix and Arch Linux. The different distributions cater to different needs, but they all have something in common: they’re all open source. They’re also all, by and large, compatible with each other: software written on a Debian system will operate perfectly well on Arch Linux and vice versa.

Linux isn’t exclusive to the Raspberry Pi. Hundreds of different distributions are available for desktops, laptops and even mobile devices; and Google’s popular Android platform is developed on top of a Linux core. If you find that you enjoy the experience of using Linux on the Raspberry Pi, you could consider adding it to other computing devices you use as well. It will happily coexist with your current operating system, allowing you to enjoy the benefits of both while giving you a familiar environment when your Pi is unavailable. As with the difference between ARM and x86, there’s a key point to make about the practical difference between Windows, OS X and Linux: software written for Windows or OS X won’t run on Linux. Thankfully, there are plenty of compatible alternatives for the overwhelming majority of common software products—better still, the majority are free to use and as open source as the operating system itself.

**Getting Started with the Raspberry Pi**

Now that you have a basic understanding of how the Pi differs from other computing devices, it’s time to get started. If you’ve just received your Pi, take it out of its protective anti-static bag and place it on a flat, non-conductive surface before continuing with this chapter.

**Connecting a Display**

Before you can start using your Raspberry Pi, you’re going to need to connect a display. The Pi supports three different video outputs: composite video, HDMI video and DSI video. Composite video and HDMI video are readily accessible to the end user, as described in this section, while DSI video requires some specialised hardware.

**Composite Video**

Composite video, available via the yellow-and-silver port at the top of the Pi known as an RCA phono connector is designed for connecting the Raspberry Pi to older display devices. As the name suggests, the connector creates a composite of the colours found within an image—red, green and blue—and sends it down a single wire to the display device, typically an old cathode-ray tube (CRT) TV. The yellow RCA phono connector, for composite video output When no other display device is available, a composite video connection will get you started with the Pi. The quality, however, isn’t great. Composite video connections are significantly more prone to interference, lack clarity and run at a limited resolution, meaning that you can fit fewer icons and lines of text on the screen at once.

**HDMI Video**

A better-quality picture can be obtained using the HDMI (High Definition Multimedia Interface) connector, the only port found on the bottom of the Pi. Unlike the analogue composite connection, the HDMI port provides a high-speed digital connection for pixel-perfect pictures on both computer monitors and high-definition TV sets. Using the HDMI port, a Pi can display images at the Full HD 1920x1080 resolution of most modern HDTV sets. At this resolution, significantly more detail is available on the screen. If you’re hoping to use the Pi with an existing computer monitor, you may find that your display doesn’t have an HDMI input. That’s not a disaster: the digital signals present on the HDMI cable map to a common computer monitor standard called DVI (Digital Video Interconnect). By purchasing an HDMI-to-DVI cable, you’ll be able to connect the Pi’s HDMI port to a monitor with DVI-D connectivity.

Figure 1-3: The silver HDMI connector, for high-definition video output

If your monitor has a VGA input—a D-shaped connector with 15 pins, typically coloured silver and blue—the Raspberry Pi can’t connect to it. Adapters are available that will take in a digital DVI signal and convert it to an analogue VGA signal, but these are expensive and bulky. The best option here is simply to buy a more-modern monitor with a DVI or HDMI input.

**DSI Video**

The final video output on the Pi can be found above the SD card slot on the top of the printed circuit board—it’s a small ribbon connector protected by a layer of plastic. This is for a video standard known as Display Serial Interface (DSI), which is used in the flat-panel displays of tablets and smartphones. Displays with a DSI connector are rarely available for retail purchase, and are typically reserved for engineers looking to create a compact, self-contained system. A DSI display can be connected by inserting a ribbon cable into the matched connector on the Pi, but for beginners, the use of a composite or HDMI display is recommended.

**Connecting Audio**

If you’re using the Raspberry Pi’s HDMI port, audio is simple: when properly configured, the HDMI port carries both the video signal and a digital audio signal. This means that you can connect a single cable to your display device to enjoy both sound and pictures. Assuming you’re connecting the Pi to a standard HDMI display, there’s very little to do at this point. For now, it’s enough to simply connect the cable.

If you’re using the Pi with a DVI-D monitor via an adapter or cable, audio will not be included. This highlights the main difference between HDMI and DVI: while HDMI can carry audio signals, DVI cannot. For those with DVI-D monitors, or those using the composite video output, a black 3.5 mm audio jack located on the top edge of the Pi next to the yellow phonon connector provides analogue audio. This is the same connector used for headphones and microphones on consumer audio equipment and it’s wired in exactly the same way. If you want, you can simply connect a pair of headphones to this port for quick access to audio. **While headphones can be connected directly to the Raspberry Pi, you may find the volume a little lacking. If possible, connect a pair of powered speakers instead. The amplifier inside will help boost the signal to a more audible level.**

If you’re looking for something more permanent, you can either use standard PC speakers that have a 3.5 mm connector or you can buy some adapter cables. For composite video users, a 3.5 mm to RCA phono cable is useful. This provides the two whiteand- red RCA phono connections that sit alongside the video connection, each carrying a channel of the stereo audio signal to the TV.

For those connecting the Pi to an amplifier or stereo system, you’ll either need a 3.5 mm to RCA phono cable or a 3.5 mm to 3.5 mm cable, depending on what spare connections you have on your system. Both cable types are readily and cheaply available at consumer electronics shops, or can be purchased even cheaper at online retailers such as Amazon.

**Connecting a Keyboard and Mouse**

Now that you’ve got your Raspberry Pi’s output devices sorted, it’s time to think about input. As a bare minimum, you’re going to need a keyboard, and for the majority of users, a mouse or trackball is a necessity too. First, some bad news: if you’ve got a keyboard and mouse with a PS/2 connector—a round plug with a horseshoe-shaped array of pins—then you’re going to have to go out and buy a replacement. The old PS/2 connection has been superseded, and the Pi expects your peripherals to be connected over the Universal Serial Bus (USB) port. Depending on whether you purchased the Model A or Model B, you’ll have either one or two USB ports available on the right side of the Pi (see Figure 1-4). If you’re using Model B, you can connect the keyboard and mouse directly to these ports. If you’re using Model A, you’ll need to purchase a USB hub in order to connect two USB devices simultaneously. Figure 1-4: Model B’s two USB ports A USB hub is a good investment for any Pi user: even if you’ve got a Model B, you’ll use up both your available ports just connecting your keyboard and mouse, leaving nothing free for additional devices such as an external optical drive, storage device or joystick. Make sure you buy a powered USB hub: passive models are cheaper and smaller, but lack the ability to run currenthungry devices like CD drives and external hard drives.

**If you want to reduce the number of power sockets in use, connect the Raspberry Pi’s USB power lead to your powered USB hub. This way, the Pi can draw its power directly from the hub, rather than needing its own dedicated power socket and mains adapter. This will only work on hubs with a power supply capable of providing 700mA to the Pi’s USB port, along with whatever power is required by other peripherals.**

Connecting the keyboard and mouse is as simple as plugging them in to the USB ports, either directly in the case of a Model B or via a USB hub in the case of a Model A.

A Note on Storage

As you’ve probably noticed, the Raspberry Pi doesn’t have a traditional hard drive. Instead it uses a Secure Digital (SD) memory card, a solid-state storage system typically used in digital cameras. Almost any SD card will work with the Raspberry Pi, but because it holds the entire operating system, it isnecessary for the card to be at least 2 GB in capacity to store all the required files.

SD cards with the operating system preloaded are available from the official Raspberry Pi Store along with numerous other sites on the Internet. If you’ve

purchased one of these, or received it in a bundle with your Pi, you can simply plug it in to the SD card slot on the bottom side of the left-hand edge. If not,

you’ll need to install an operating system—known as flashing—onto the card before it’s ready to go.

Some SD cards work better than others, with some models refusing to work at all with the Raspberry Pi. For an up-to-date list of SD card models known to

work with the Pi, visit the eLinux

**Flashing the SD Card**

To prepare a blank SD card for use with the Raspberry Pi, you’ll need to flash an operating system onto the card. While this is slightly more complicated than simply dragging and dropping files onto the card, it shouldn’t take more than a few minutes to complete.

Firstly, you’ll need to decide which Linux distribution you would like to use with your Raspberry Pi. Each has its advantages and disadvantages. Don’t worry if you change your mind later and want to try a different version of Linux: an SD card can be flashed again with a new operating system at any point. The most up-to-date list of Linux releases compatible with the Pi is available from the Raspberry Pi website at The Foundation provides BitTorrent links for each distribution. These are small files that can be used with BitTorrent software to download the files from other users. Using these links is an efficient and fast way to distribute large files, and keeps the Foundation’s download servers from becoming overloaded. To use a BitTorrent link, you’ll need to have a compatible client installed. If you don’t already have a BitTorrent client installed, download one and install it before trying to download the Raspberry Pi Linux distribution. One client for Windows, OS X and Linux is μTorrent, Which distribution you choose to download is up to you. Instructions in the rest of the book will be based on the Debian Raspberry Pi distribution, a good choice for beginners. Where possible, we’ll give you instructions for other distributions as well. Linux distributions for the Raspberry Pi are provided as a single image file, compressed to make it faster to download. Once you’ve downloaded the Zip archive (a compressed file, which takes less time to download than the uncompressed files would) for your chosen distribution, you’ll need to decompress it somewhere on your system. In most operating systems, you can simply double-click the file to open it, and then choose Extract or Unzip to retrieve the contents. After you’ve decompressed the archive, you’ll end up with two separate files. The file ending in sha1 is a hash, which can be used to verify that the download hasn’t been corrupted in transit. The file ending in img contains an exact copy of an SD card set up by the distribution’s creators in a way that the Raspberry Pi understands. This is the file that needs to be flashed to the SD card.

**During the following, you’ll be using a software utility called dd. Used incorrectly dd will happily write the image to your main hard drive, erasing**

**your operating system and all your stored data. Make sure you read the instructions in each section thoroughly and note the device address of your**

**SD card carefully. Read twice, write once!**

**Flashing from Linux**

If your current PC is running a variant of Linux already, you can use the dd command to write the contents of the image file out to the SD card. This is a text-interface program operated from the command prompt, known as a terminal in Linux parlance.

Follow these steps to flash the SD card:

**1.** Open a terminal from your distribution’s applications menu.

**2.** Plug your blank SD card into a card reader connected to the PC.

**3.** Type sudo fdisk -l to see a list of disks. Find the SD card by its size, and note the device address (/dev/sdX, whereX is a letter identifying the storage device. Some systems with integrated SD card readers may use the alternative format/dev/mmcblkX—if this is the case, remember to change the target in the following instructions accordingly).

**4.** Use cd to change to the directory with the .img file you extracted from the Zip archive.

**5.** Type sudo dd if=imagefilename.img of=/dev/sdX bs=2M to write the file imagefilename.img to the SD cardconnected to the device address from step 3. Replace imagefilename.img with the actual name of the file extracted from theZip archive. This step takes a while, so be patient! During flashing, nothing will be shown on the screen until the process is

fully complete.

Figure 1-5: Flashing the SD card using the dd command in Linux

**Flashing from OS X**

If your current PC is a Mac running Apple OS X, you’ll be pleased to hear that things are as simple as with Linux. Thanks to a similar ancestry, OS X and Linux both contain the dd utility, which you can use to flash the system image to your blank SD card as follows:

**1.** Select Utilities from the Application menu, and then click on the Terminal application.

**2.** Plug your blank SD card into a card reader connected to the Mac.

**3.** Type diskutil list to see a list of disks. Find the SD card by its size, and note the device address (/dev/diskX,where X is a letter identifying the storage device).

**4.** If the SD card has been automatically mounted and is displayed on the desktop, type diskutil unmountdisk/dev/diskX to unmount it before proceeding.

**5.** Use cd to change to the directory with the .img file you extracted from the Zip archive.

**6.** Type dd if=imagefilename.img of=/dev/diskX bs=2M to write the file imagefilename.img to the SD card connectedto the device address from step 3. Replace imagefilename.img with the actual name of the file extracted from the Zip archive.This step takes a while, so be patient!

**Connecting External Storage**

While the Raspberry Pi uses an SD card for its main storage device—known as a boot device—you may find that you run into space limitations quite quickly. Although large SD cards holding 32 GB, 64 GB or more are available, they are often prohibitively expensive. Thankfully, there are devices that provide an additional hard drive to any computer when connected via a USB cable. Known as USB Mass Storage (UMS) devices, these can be physical hard drives, solid-state drives (SSDs) or even portable pocket-sized flash drives.Two USB Mass Storage devices: a pen drive and an external hard drive The majority of USB Mass Storage devices can be read by the Pi, whether or not they have existing content. In order for the Pi to be able to access these devices, their drives must be mounted—a process you will learn in Chapter 2, “Linux System Administration”. For now, it’s enough to connect the drives to the Pi in readiness.

**Connecting the Network**

While the majority of these setup instructions are equally applicable to both the Raspberry Pi Model A and the Model B,networking is a special exception. To keep the component count—and therefore the cost—as low as possible, the Model Adoesn’t feature any onboard networking. Thankfully, that doesn’t mean you can’t network the Model A; only that you’ll needsome additional equipment to do so.

**Wired Networking**

To get your Raspberry Pi on the network, you’ll need to connect an RJ45 Ethernet patch cable between the Pi and a switch,router or hub. If you don’t have a router or hub, you can get your desktop or laptop talking to the Pi by connecting the twodirectly together with a patch cable.

Usually, connecting two network clients together in this way requires a special cable, known as a crossover cable. In a crossovercable, the receive and transmit pairs are swapped so that the two devices are prevented from talking over each other—a taskusually handled by a network switch or hub.The Raspberry Pi is cleverer than that, however. The RJ45 port on the side of the Pi includes a feature knownas auto-MDI, which allows it to reconfigure itself automatically. As a result, you can use any RJ45 cable—crossover or not—toconnect the Pi to the network, and it will adjust its configuration accordingly.The Raspberry Pi Model B’s Ethernet portIf you do connect the Pi directly to a PC or laptop, you won’t be able to connect out onto the Internet by default. To do so,you’ll need to configure your PC to bridge the wired Ethernet port and another (typically wireless) connection. Doing so isoutside the scope of this book, but if you are completely unable to connect the Pi to the Internet in any other way, you can trysearching your operating system’s help file for “bridge network” to find more guidance.With a cable connected, the Pi will automatically receive the details it needs to access the Internet when it loads its operatingsystem through the Dynamic Host Configuration Protocol (DHCP). This assigns the Pi an Internet Protocol (IP) address on yournetwork, and tells it the gateway it needs to use to access the Internet (typically the IP address of your router or modem).For some networks, there is no DHCP server to provide the Pi with an IP address. When connected to such a network, the Piwill need manual configuration. You’ll learn more about this in Chapter 4, “Network Configuration”.

**Wireless Networking**

Current Raspberry Pi models don’t feature any form of wireless network capability onboard, but—as with adding wired Ethernetto the Model A—it’s possible to add Wi-Fi support to any Pi using a USB wireless adapter.

Two USB wireless adapters, suitable for use with the Raspberry PiUsing such a device, the Pi can connect to a wide range of wireless networks, including those running on the latest 802.11n highspeedstandard. Before purchasing a USB wireless adapter, check the following:

• Ensure that Linux is listed as a supported operating system. Some wireless adapters are provided with drivers for Windowsand OS X only, making them incompatible with the Raspberry Pi. A list of Wi-Fi adapters known to work with theRaspberry Pi.

• Ensure that your Wi-Fi network type is supported by the USB wireless adapter. The network type will be listed in thespecifications as a number followed by a letter. If your network type is 802.11a, for example, an 802.11g wireless adapterwon’t work.

• Check the frequencies supported by the card. Some wireless network standards, like 802.11a, support more than onefrequency. If a USB wireless adapter is designed to work on a 2.4GHz network, it won’t connect to a 5GHz network.

• Check the encryption type used by your wireless network. Most modern USB wireless adapters support all forms ofencryption, but if you’re buying a second-hand or older model, you may find it won’t connect to your network. Commonencryption types include the outdated WEP and more modern WPA and WPA2.Configuration of the wireless connection is done within Linux, so for now it’s enough to simply connect the adapter to the Pi(ideally through a powered USB hub.) You’ll learn how to configure the connection in Chapter 4, “Network Configuration”.

**Connecting Power**

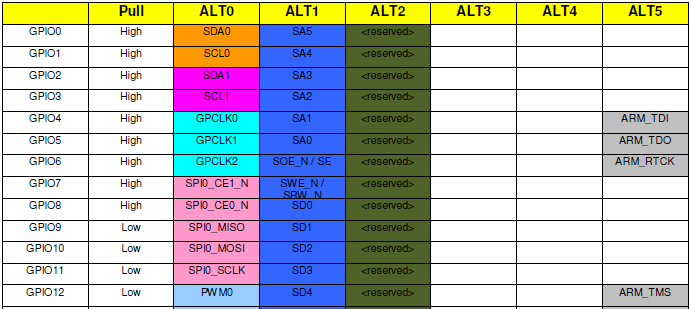
The Raspberry Pi is powered by the small micro-USB connector found on the lower left side of the circuit board. This connectoris the same as found on the majority of smartphones and some tablet devices.Many chargers designed for smartphones will work with the Raspberry Pi, but not all. The Pi is more power-hungry than mostmicro-USB devices, and requires up to 700mA in order to operate. Some chargers can only supply up to 500mA, causingintermittent problems in the Pi’s operation. Connecting the Pi to the USB port on a desktop or laptop computer is possible, but not recommended. As with smallerchargers, the USB ports on a computer can’t provide the power required for the Pi to work properly.Only connect the micro-USB power supply when you are ready to start using the Pi. With no power button on the device, it willstart working the instant power is connected and can only be turned off again by physically removing the power cable.

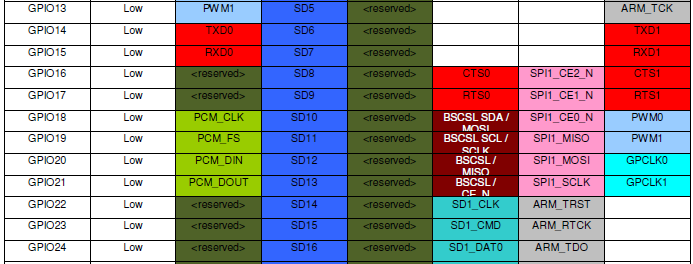
**The GPIO Port**

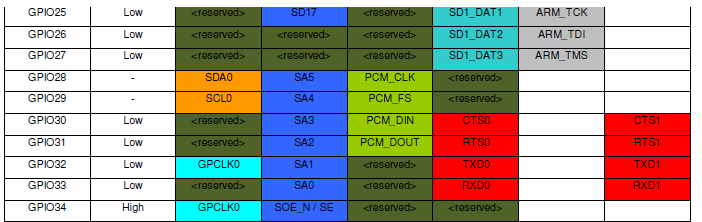
The Raspberry Pi’s GPIO port is located on the top-left of the printed circuit board, labelled P1. It’s a 54-pin port, fitted withtwo rows of 13 male 2.54 mm headers at the factory. The spacing of these headers is particularly important: 2.54 mm pinspacing (0.1 inches in imperial measurements) is a very common sight in electronics, and is the standard spacing for prototypingplatforms that include stripboards and breadboards.Each pin of the GPIO port has its own purpose, with several pins working together to form particular circuits. The layout of the

GPIO port.The Raspberry Pi’s GPIO port and its pin definitionsPin numbers for the GPIO port are split into two rows, with the bottom row taking the odd numbers and the top row the even

numbers. It’s important to keep this in mind when working with the Pi’s GPIO port: most other devices use a different system fornumbering pins, and because there are no markings on the Pi itself, it’s easy to get confused as to which pin is which.



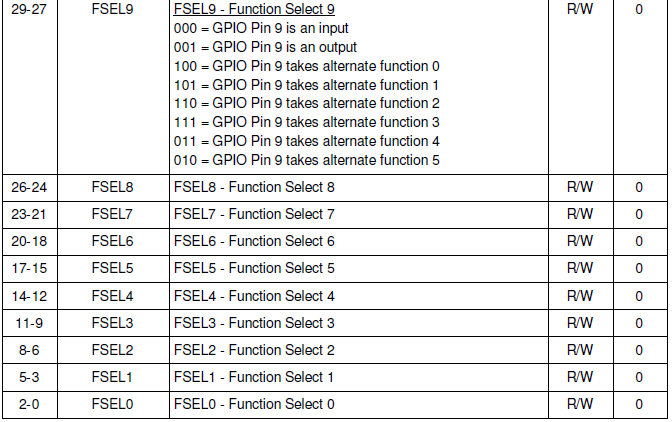




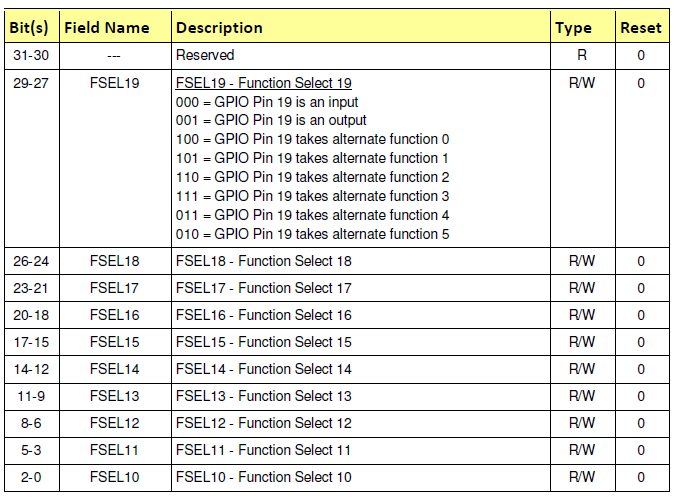


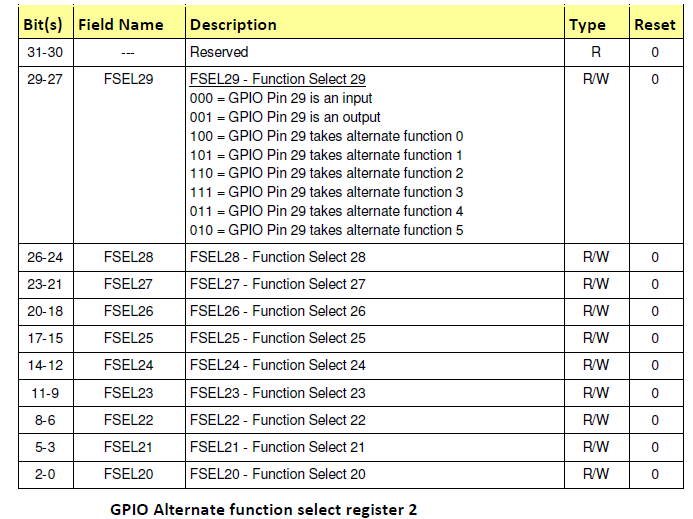
**Never connect anything to the pins marked Do Not Connect; these are reserved for internal functions of the Pi’s BCM2837 system-on-chip (SoC)hardware. Connecting anything to these will result in damage to the Pi.**

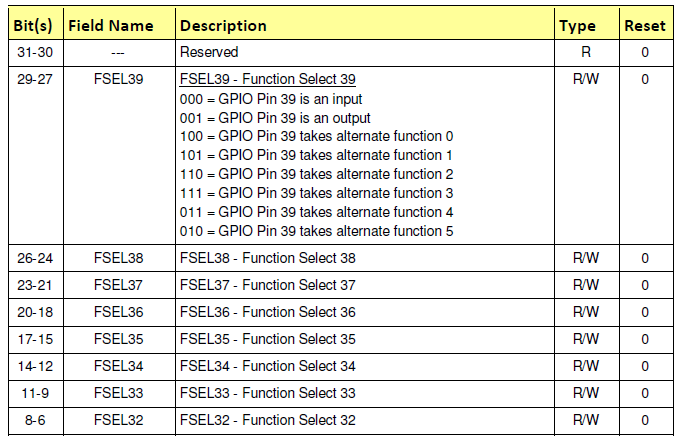
**GPFSEL0:**

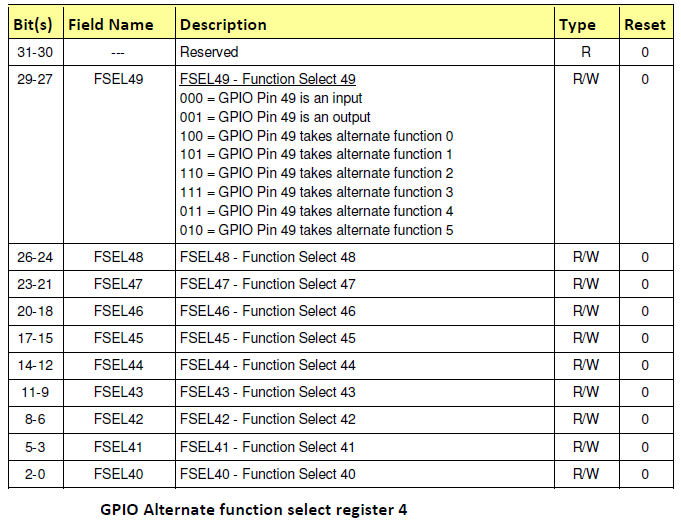
****

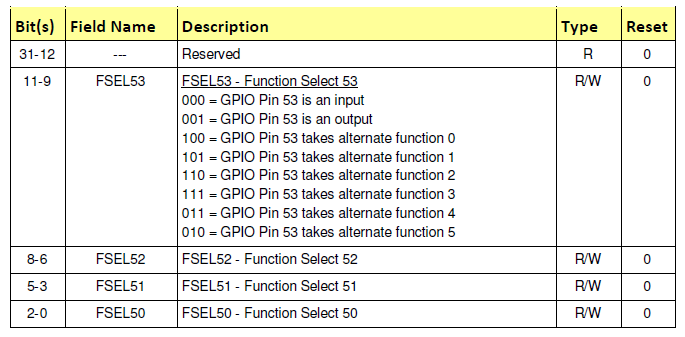
**GPFSEL1:**

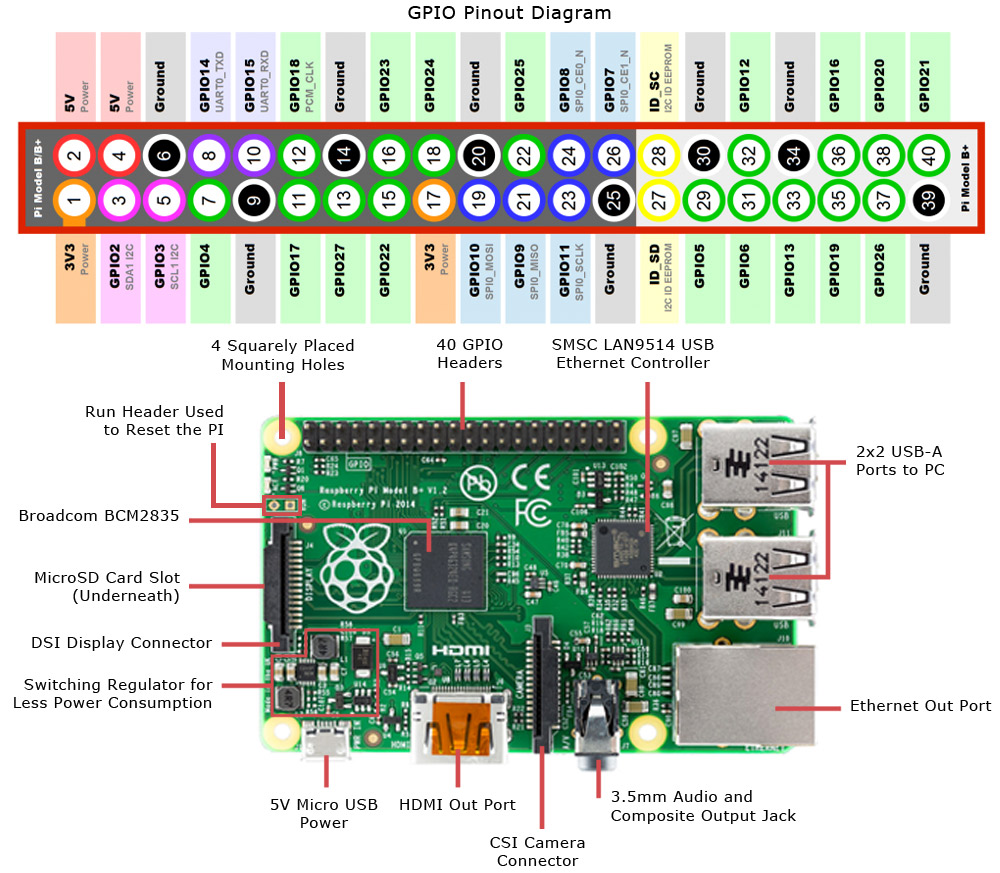
****

****

****

****

****

****

Although the Pi’s GPIO port provides a 5 V power supply, tapped from the incoming power on the micro-USB hub, on Pin 2,the Pi’s internal workings are based on 3.3 V logic. This means that the components on the Pi work from a 3.3 V power supply.If you’re planning on creating a circuit that will interface with the Pi through its GPIO port, make sure you are using componentscompatible with 3.3 V logic or are passing the circuit through a voltage regulator before it reaches the Pi.

**Connecting a 5 V supply to any pin on the Raspberry Pi’s GPIO port, or directly shorting either of the power supply pins (Pin 1 and Pin 2) to anyother pin will result in damage to the Pi. Because the port is wired directly to pins on the Broadcom BCM2837 SoC processor, you will not be able torepair any damage you do to it. Always be extra careful when working around the GPIO port.**

The GPIO port provides seven pins for general-purpose use by default: Pin 11, Pin 12, Pin 13, Pin 15, Pin 16, Pin 18 and Pin22. Additionally, Pin 7—while defaulting to providing a clock signal for general purpose use—can also be used as a generalpurposepin, giving eight pins in total. These pins can be toggled between two states: high, where they are providing a positive

voltage of 3.3 V; and low, where they are equal to ground or 0 V. This equates to the 1 and 0 of binary logic, and can be usedto turn other components on or off. You’ll learn more about this later in the chapter.

**The Pi’s internal logic operates at 3.3 V. This is in contrast to many common microcontroller devices, such as the popular Arduino and its variants,which typically operate at 5 V. Devices designed for the Arduino may not work with the Pi unless a level translator or optical isolator is usedbetween the two. Likewise, connecting pins on a 5 V microcontroller directly to the Raspberry Pi’s GPIO port will not work and may permanentlydamage the Pi.**

In addition to these general-purpose pins, the GPIO port has pins dedicated to particular buses. These buses are described inthe following subsections.

**UART Serial Bus**

The Universal Asynchronous Receiver/Transmitter (UART) serial bus provides a simple two-wire serial interface. When a serialport is configured in the cmdline.txt file (as described in Chapter 6, “Configuring the Raspberry Pi”), it’s this serial bus that isused as the port for the messages. Connecting the Pi’s UART serial bus to a device capable of displaying the data will revealmessages from the Linux kernel. If you’re having trouble getting the Pi to boot, this can be a handy diagnostic tool—especially ifnothing is showing on the display.The UART serial bus can be accessed on Pins 8 and 10, with Pin 8 carrying the transmit signal and Pin 10 carrying the receivesignal. The speed can be set in the cmdline.txt file, and is usually 115,200 bits per second (bps).

**I²C Bus**

As the name suggests, the Inter-Integrated Circuit (I²C) bus is designed to provide communications between multiple integratedcircuits (ICs). In the case of the Pi, one of those integrated circuits is the Broadcom BCM2837 SoC processor at the heart ofthe system. These pins include access to pull-up resistors located on the Pi, meaning no external resistors are required to accessthe I²C functionality.The I²C bus can be accessed on Pins 3 and 5, with Pin 3 providing the Serial Data Line (SDA) signal and Pin 5 providing theSerial Clock (SCL) signal. The I²C bus available on these pins is actually only one of two provided by the BCM2837 chip itself,and is known as I²C0. The second, I²C1, is terminated at resistors on the Raspberry Pi circuit board itself and is not available forgeneral-purpose use.

**SPI Bus**

The Serial Peripheral Interface (SPI) bus is a synchronous serial bus designed primarily for in-system programming (ISP) ofmicrocontrollers and other devices. Unlike the UART and I²C buses, it’s a four-wire bus with multiple Chip Select lines whichallow it to communicate with more than one target device.The Pi’s SPI bus is available on Pins 19, 21 and 23, with a pair of Chip Select lines on Pin 24 and Pin 26. Pin 19 provides theSPI Master Output, Slave Input (MOSI) signal; Pin 21 provides the SPI Master Input, Slave Output (MISO) signal; Pin 23provides the Serial Clock (SLCK) used to synchronise communication; and Pins 24 and 26 provide the Chip Select signals forup to two independent slave devices.Although additional buses are present in the Raspberry Pi’s BCM2837 SoC processor, they are not brought out to the GPIOport and are thus unavailable for use.

**Using the GPIO Port in Python**

With the theory out of the way, it’s time to get practical. In this section, you’ll learn how to install a library to allow easy accessto the general-purpose pins on the Raspberry Pi’s GPIO port in Python. You’ll also be shown two simple electronic circuitswhich demonstrate how to use the GPIO port for input and output.As you saw in Chapter 11, “Python Basics”, Python is a friendly yet powerful programming language. It’s not, however, theperfect choice for every scenario. Although it works fine for the simple circuits you’ll be creating in this chapter, it does not offer

what is known as deterministic real-time operation. For the majority of users, this doesn’t matter; if you’re planning on using thePi at the heart of a nuclear reactor or a complex robotics platform, however, you may want to investigate a lower-level languagesuch as C++ or even assembler running on a dedicated real-time microcontroller.If true real-time operation is required for your project, the Pi may be a bad choice. Instead, consider using a microcontroller

platform such as the popular open-source Arduino, or one of the MSP430 family of microcontrollers from Texas Instruments.Both of these devices can interface with the Pi either through the GPIO header or over USB, and provide a specialised real-timeenvironment for control and sensing.

**Installing the GPIO Python Library**

Since the launch of the Pi, numerous developers have created software modules known as libraries for making full use of itsvarious functions. In particular, programmers have addressed the Pi users’ need to access the GPIO port without having to knowlow-level programming.

These libraries are designed to extend the functionality of the base Python language, much like the pygame software described, “Python Basics”. Installing one of these libraries gives Python the ability to easily address the Pi’s GPIO port,although it means that anyone planning to use the software you create will also have to download and install the library before itwill work.There are several GPIO Python libraries available, but for the purpose of this section, we recommend that you use theraspberry-gpio-python library, which was at version 0.2.0 at the time of writing.

Although it’s possible to download the Python library through a web browser, it’s significantly quicker to do so through theterminal as part of the installation process. Just follow these steps:

**1.** Open a terminal window on your Raspberry Pi from the Accessories menu, or use the console if you haven’t loaded adesktop environment.

**2.** Type wget http://raspberry-gpio-python.googlecode.com/files/RPi.GPIO-0.2.0.tar.gz to downloadthe library to your home directory. If a newer version has been released, replace the version number—0.2.0—with thecurrent version.

**3.** Type tar xvzf RPi.GPIO-0.2.0.tar.gz to extract the contents of the file. This command is case-sensitive, so makesure to type the capital letters.

**4.** Type cd RPi.GPIO-0.2.0 to change to the newly created directory. Again, if you downloaded a newer version of thelibrary, replace the version number with that of the downloaded version.

**5.** Type sudo python setup.py install to install the library into Python.Although the GPIO library is now installed in Python, it won’t be loaded by default. Like pygame, the library needs to be

explicitly imported into your programm. To use the library, start your program with import RPi.GPIO as GPIO at the top.You’ll learn more about this in the following examples.

**The Raspberry Pi’s GPIO port does not provide any protection against voltage spikes or electrical shorts. Always make sure you’ve checked thatyour circuit is sound before connecting it to the Pi. If possible, use an isolation board such as the Gertboard to provide protection.**

Calculating Limiting Resistor ValuesAn LED needs a current limiting resistor to protect it from burning out. Without a resistor, an LED will likely only work for a short time before failing and

An LED needs a current limiting resistor to protect it from burning out. Without a resistor, an LED will likely only work for a short time before failing andneeding to be replaced. Knowing a resistor is required is one thing, but it’s also important to pick the right resistor for the job. Too high a value and the LEDwill be extremely dim or fail to light at all; too low a value and it will burn out.To calculate the resistor value required, you will need to know the forward current of your LED. This is the maximum current the LED can draw before beingdamaged, and is measured in milliamps (mA). You’ll also need to know the forward voltage of the LED. This latter value, measured in volts, should be 3.3 Vor lower—any higher, and the LED will require an external power supply and a switching device known as a transistor before it will work with the Pi.The easiest way to work out how large a resistor is required is with the formula R=(V-F)/I, where R is resistance in ohms, V is the voltage applied to theLED, F is the forward voltage of the LED and I is the maximum forward current of the LED in amps (with a thousand mA to the amp).Taking a typical red LED with a forward current of 25 mA and a forward voltage of 1.7 V, and powering it using the 3.3 V supplied by the Pi’s GPIO port,you can calculate the resistor needed as (3.3 – 1.7) / 0.025 = 64. Thus, a resistor of 64 Ω or higher will protect the LED. These figures rarely come outto match the common resistor values as sold, so when you’re choosing a resistor, always round up to ensure the LED is protected. The nearest commonly

available value is 68 Ω, which will adequately protect the LED.If you don’t know the forward voltage and forward current of your LEDs (for example, if the LEDs did not come with documentation or were salvaged fromscrap electronics), err on the side of caution and fit a reasonably large resistor. If the LED is too dim, you can revise downwards—but it’s impossible to repairan LED that has been blown.

**GPIO Output: Flashing an LED**

For the first example, you’ll need to build a simple circuit consisting of an LED and a resistor. The LED will provide visualconfirmation that the Pi’s GPIO port is doing what your Python program tells it to do, and the resistor will limit the current drawnby the LED to protect it from burning out.To assemble the circuit, you’ll need a breadboard, two jumper wires, an LED and an appropriate current-limiting resistor (asdescribed in the “Calculating Limiting Resistor Values” sidebar). Although it’s possible to assemble the circuit without abreadboard by twisting wires together, a breadboard is a sound investment and makes assembling and disassembling prototype

circuits straightforward.Assuming the use of a breadboard, assemble the circuit in the following manner to match

**1.** Insert the LED into the breadboard so that the long leg (the anode) is in one row and the shorter leg (the cathode) is inanother. If you put the LED’s legs into the same row, it won’t work.

**2.** Insert one leg of the resistor into the same row as the LED’s shorter leg, and the other resistor leg into an empty row. Thedirection in which the resistor’s legs are placed doesn’t matter, as a resistor is a non-polarised (direction-insensitive) device.

**3.** Using a jumper wire, connect Pin 11 of the Raspberry Pi’s GPIO port (or the corresponding pin on an interface boardconnected to the GPIO port) to the same row as the long leg of the LED.

**4.** Using another jumper wire, connect Pin 6 of the Raspberry Pi’s GPIO port (or the corresponding pin on an interfaceboard connected to the GPIO port) to the row that contains only one leg of the resistor and none of the LED’s legs.

**Be very careful when connecting wires to the Raspberry Pi’s GPIO port. As discussed earlier in the chapter, you may do serious damage to the Pi if**

**you connect the wrong pins.**

A breadboard circuit for a simple LED outputAt this point, nothing will happen. That’s perfectly normal: by default, the Raspberry Pi’s GPIO pins are switched off. If youwant to check your circuit immediately, move the wire from Pin 11 to Pin 1 to make the LED light up. Be careful not to connectit to Pin 2, though: a current-limiting resistor suitable for a 3.3 V power supply will be inadequate to protect the LED whenconnected to 5 V. Remember to move the wire back to Pin 11 before continuing.To make the LED do something useful, start a new Python project. “An Introduction toPython”, you can use a plain text editor or the IDLE software included in the recommended Debian distribution for this projectas well.Before you can use the GPIO library you installed earlier in this chapter, you’ll need to import it into your Python project.Accordingly, start the file with the following line:

import RPi.GPIO as GPIO

Remember that Python is case-sensitive, so be sure to type RPi.GPIO exactly as it appears. To allow Python to understand theconcept of time (in other words, to make the LED blink, rather than just turning it on and off), you’ll also need to import the timemodule. Add the following line to the project:

import time

With the libraries imported, it’s time to address the GPIO ports. The GPIO library makes it easy to address the general-purposeports through the instructions GPIO.output and GPIO.input, but before you can use them, you’ll need to initialise the pins aseither inputs or outputs. In this example, Pin 11 is an output, so add the following line to the project:

GPIO.setup(11, GPIO.OUT)

This tells the GPIO library that Pin 11 on the Raspberry Pi’s GPIO port should be set up as an output. If you were controllingadditional devices, you could add more

GPIO.setup

lines into the project. For now, however, one will suffice.

With the pin configured as an output, you can switch its 3.3 V supply on and off in a simple demonstration of binary logic. Theinstruction

GPIO.output(11, True)

will turn the pin on, while

GPIO.output(11, False)

switches it off again. The pin willremember its last state, so if you only give the command to turn the pin on and then exit your Python program, the pin will remainon until told otherwise.

Although you could just add GPIO.output(11, True) to the Python project to switch the pin on, it’s more interesting to makeit blink. First, add the following line to create an infinite loop in the program:

while True:

Next, add the following lines to switch the pin on, wait 2 seconds, and then switch it off again before waiting another 2 seconds.Make sure each line starts with four spaces, to signify that it is part of the infinite while loop:

GPIO.output(11, True)

time.sleep(2)

GPIO.output(11, False)

time.sleep(2)

The finished program should look like this (see Figure 12-4):

import RPi.GPIO as GPIO

import time

GPIO.setup(11, GPIO.OUT)

while True:

GPIO.output(11, True)

time.sleep(2)

GPIO.output(11, False)

time.sleep(2)

The gpiooutput.py program, being edited in nano, and waiting for its final lineSave the file as gpiooutput.py. If you’re using a Python development environment such as SPE, don’t try to run the programfrom within the editor. Most Raspberry Pi Linux distributions restrict the use of the GPIO port to the root user, so the programwill need to be run using the command sudo python gpiooutput.py at the terminal to get it started. If all has gone well, youshould see the LED begin to blink on and off at regular intervals—and you’ve created your first home-made output device forthe Pi.

If things don’t work, don’t panic. First, check all your connections. The holes in a breadboard are quite small, and it’s easy tothink you’ve inserted a component into one row only to find it’s actually in another. Next, check that you’ve connected the circuitto the right pins on the GPIO port—with no labelling on the Pi itself, mistakes are unfortunately easy to make. Finally, doublecheckyour components—if the forward voltage of your LED is higher than 3.3 V or if your current limiting resistor is too large,the LED won’t light up.Although this example is basic, it’s a good demonstration of some fundamental concepts. To extend its functionality, the LEDcould be replaced with a buzzer to make an audible alert, or a servo or motor as part of a robotics platform. The code used toactivate and deactivate the GPIO pin can be integrated into other programs, causing an LED to come on when new email arrivesor a flag to be raised when a friend has joined an IRC channel.

**ULTRASONIC SENSOR**

Ultrasonic sensors measure distance by using ultrasonic waves.  
The sensor head emits an ultrasonic wave and receives the wave reflected back from the target. Ultrasonic Sensors measure the distance to the target by measuring the time between the emission and reception



An optical sensor has a transmitter and receiver, whereas an ultrasonic sensor uses a single ultrasonic element for both emission and reception. In a reflective model ultrasonic sensor, a single oscillator emits and receives ultrasonic waves alternately. This enables miniaturization of the sensor head.

**HC-SR04 Ultrasonic Sensor - Working**

As shown above the HC-SR04 Ultrasonic (US) sensor is a 4 pin module, whose pin names are Vcc, Trigger, Echo and Ground respectively. This sensor is a very popular sensor used in many applications where measuring distance or sensing objects are required. The module has two eyes like projects in the front which forms the Ultrasonic transmitter and Receiver. The sensor works with the simple high school formula that

**Distance = Speed × Time**

The Ultrasonic transmitter transmits an ultrasonic wave, this wave travels in air and when it gets objected by any material it gets reflected back toward the sensor this reflected wave is observed by the Ultrasonic receiver module as shown in the picture below



Now, to calculate the distance using the above formulae, we should know the Speed and time. Since we are using the Ultrasonic wave we know the universal speed of US wave at room conditions which is 330m/s. The circuitry inbuilt on the module will calculate the time taken for the US wave to come back and turns on the echo pin high for that same particular amount of time, this way we can also know the time taken. Now simply calculate the distance using a microcontroller or microprocessor.

### How to use the HC-SR04 Ultrasonic Sensor

**HC-SR04 distance sensor** is commonly used with both microcontroller and microprocessor platforms like Arduino, ARM, PIC, Raspberry Pie etc. The following guide is universally since it has to be followed irrespective of the type of computational device used.



Power the Sensor using a regulated +5V through the Vcc ad Ground pins of the sensor. The current consumed by the sensor is less than 15mA and hence can be directly powered by the on board 5V pins (If available). The Trigger and the Echo pins are both I/O pins and hence they can be connected to I/O pins of the microcontroller. To start the measurement, the trigger pin has to be made high for 10uS and then turned off. This action will trigger an ultrasonic wave at frequency of 40Hz from the transmitter and the receiver will wait for the wave to return. Once the wave is returned after it getting reflected by any object the Echo pin goes high for a particular amount of time which will be equal to the time taken for the wave to return back to the sensor.

The amount of time during which the Echo pin stays high is measured by the MCU/MPU as it gives the information about the time taken for the wave to return back to the Sensor. Using this information the distance is measured as explained in the above heading.

### Ultrasonic Sensor Pin Configuration

|  |  |  |
| --- | --- | --- |
| **Pin Number** | **Pin Name** | **Description** |
| 1 | Vcc | The Vcc pin powers the sensor, typically with +5V |
| 2 | Trigger | Trigger pin is an Input pin. This pin has to be kept high for 10us to initialize measurement by sending US wave. |
| 3 | Echo | Echo pin is an Output pin. This pin goes high for a period of time which will be equal to the time taken for the US wave to return back to the sensor. |
| 4 | Ground | This pin is connected to the Ground of the system. |

**HC-SR04 Sensor Features**

* Operating voltage: +5V
* Theoretical  Measuring Distance: 2cm to 450cm
* Practical Measuring Distance: 2cm to 80cm
* Accuracy: 3mm
* Measuring angle covered: <15°
* Operating Current: <15mA
* Operating Frequency: 40Hz

### Applications

* Used to avoid and detect obstacles with robots like biped robot, obstacle avoider robot, path finding robot etc.
* Used to measure the distance within a wide range of 2cm to 400cm
* Can be used to map the objects surrounding the sensor by rotating it
* Depth of certain places like wells, pits etc can be measured since the waves can penetrate through water

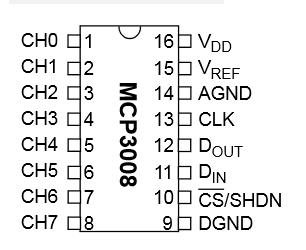
**MCP3008 (ADC)**

**MCP3008 8 channel ADC**

The MCP3008 is a 10bit 8-channel Analogue-to-digital converter (ADC). It is cheap, easy to connect and doesn’t require any additional components. It uses the SPI bus protocol which is supported by the Pi’s GPIO header.

The explanation of how to use an MCP3008 device to provide 8 analogue inputs which you can use with a range of sensors is described. The circuit below shows the MCP3008 to read a temperature and light sensor.

The first step is enabling the SPI interface on the Pi which is usually disabled by default. Please follow **SITE** article to setup SPI and install the SPI Python wrapper. The following list shows how the MCP3008 can be connected. It requires 4 GPIO pins on the Pi P1 Header.



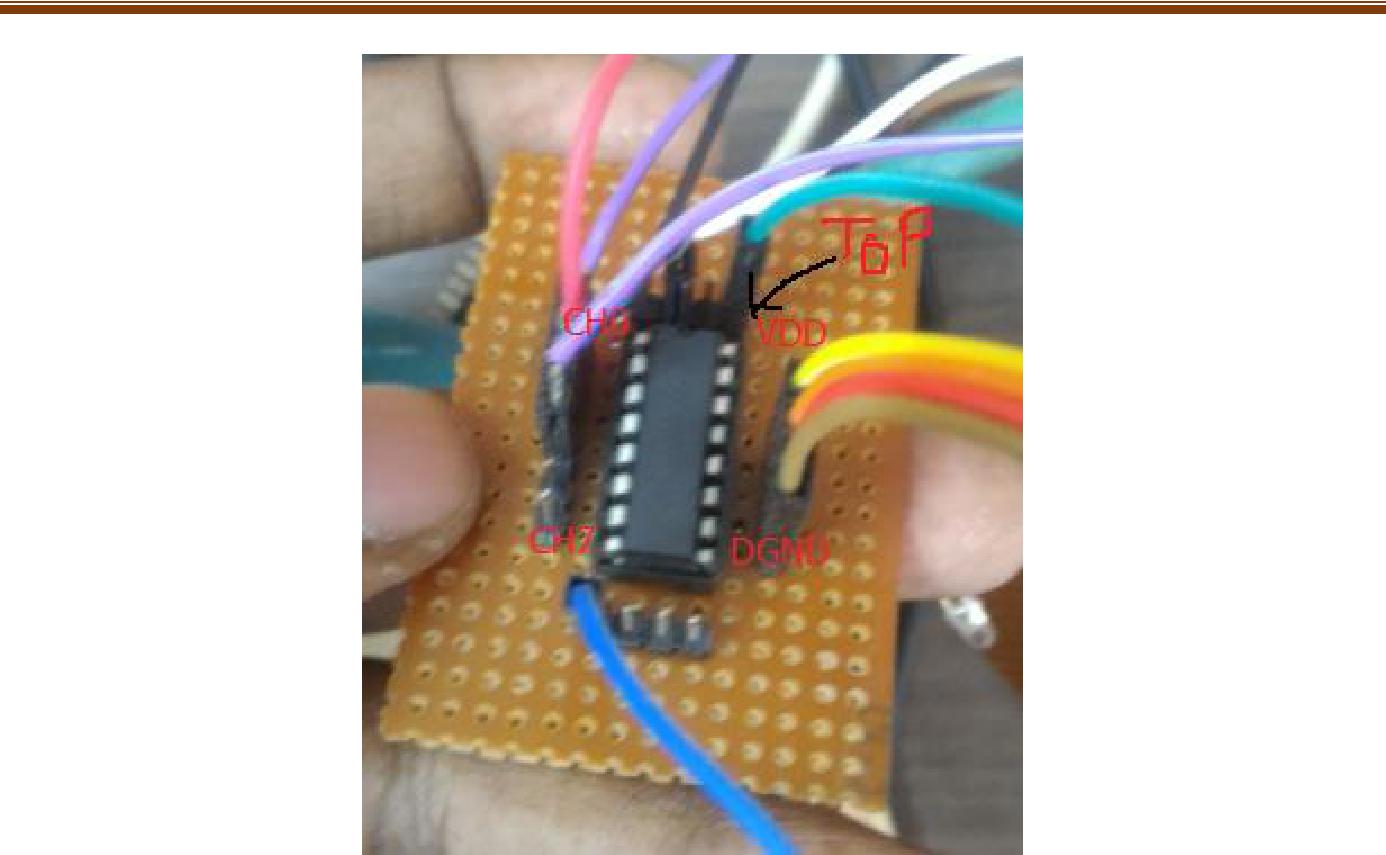
**Fig. 3.8 MCP3008 Pin Description**

**Tabular**

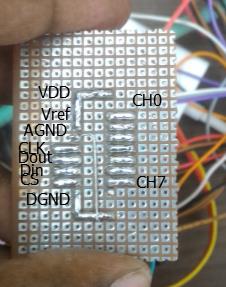
|  |  |  |  |
| --- | --- | --- | --- |
|  | **MCP3008 Pin’s** | **Description** |  |
|  | VDD | 3.3v |  |
|  | VREF | 3.3V |  |
|  | AGND | GROUND |  |
|  | CLK | GPIO18 (Pin no. 12) |  |
|  | DOUT | GPIO23 (Pin no. 16) |  |
|  | DIN | GPIO24 (Pin no. 18) |  |
|  | CS | GPIO25 (Pin no. 22) |  |
|  | DGND | GROUND |  |

**3.5.2 Hardware Making**

We have to make a MCP3008 IC base so that the connection become easier. We need some hardware i.e. PCB board and a 16 pi IC base and few male header pins. Now place your IC base on the PCB and mark that a semi-circle will be the top and numbering should start mention in Fig. 3.8. After that place 8 male header pins beside the CH0 – CH7 pins of the IC base an solder them as show in the below figures, Now take four male header pins on the top of the IC base and solder them and short VDD and Vref as shown in figure Fig 3.9 & Fig 3.10. Same way take four male header pins and place them at the bottom of the IC base short the AGND and DGND as show in the Fig. 3.10 and connect with the four male header pins. Now take four more male header pins and place beside the pin no. 13, 12, 11, 10 and short them with CLK, Dout, Din and CS and connect it to the raspberry pi according to the pins mentioned in the above section.



**Fig 3.9 MCP3008 Front View**



**GPS**

**What is GPS**

The Global Positioning System (GPS) is a satellite-based navigation system made up of at least 24 satellites. GPS works in any weather conditions, anywhere in the world, 24 hours a day, with no subscription fees or setup charges.



**How GPS works**

GPS satellites circle the Earth twice a day in a precise orbit. Each satellite transmits a unique signal and orbital parameters that allow GPS devices to decode and compute the precise location of the satellite. GPS receivers use this information and trilateration to calculate a user's exact location. Essentially, the GPS receiver measures the distance to each satellite by the amount of time it takes to receive a transmitted signal. With distance measurements from a few more satellites, the receiver can determine a user's position and display it.

To calculate your 2-D position (latitude and longitude) and track movement, a GPS receiver must be locked on to the signal of at least 3 satellites. With 4 or more satellites in view, the receiver can determine your 3-D position (latitude, longitude and altitude). Generally, a GPS receiver will track 8 or more satellites, but that depends on the time of day and where you are on the earth.

Once your position has been determined, the GPS unit can calculate other information, such as:

* Speed
* Bearing
* Track
* Trip dist
* Distance to destination

**What's the signal?**

GPS satellites transmit at least 2 low-power radio signals. The signals travel by line of sight, meaning they will pass through clouds, glass and plastic but will not go through most solid objects, such as buildings and mountains. However, modern receivers are more sensitive and can usually track through houses.

A GPS signal contains 3 different types of information:

* Pseudorandom code is an I.D. code that identifies which satellite is transmitting information. You can see which satellites you are getting signals from on your device's satellite page.
* Ephemeris data is needed to determine a satellite's position and gives important information about the health of a satellite, current date and time.
* Almanac data tells the GPS receiver where each GPS satellite should be at any time throughout the day and shows the orbital information for that satellite and every other satellite in the system.

**Overview of NEO-6M GPS Module**

1. **NEO-6M GPS Chip**

The heart of the module is a NEO-6M GPS chip from u-blox. It can track up to 22 satellites on 50 channels and achieves the industry’s highest level of sensitivity i.e. -161 dB tracking, while consuming only 45mA supply current. The u-blox 6 positioning engine also boasts a Time-To-First-Fix (TTFF) of under 1 second. One of the best features the chip provides is Power Save Mode(PSM). It allows a reduction in system power consumption by selectively switching parts of the receiver ON and OFF. This dramatically reduces power consumption of the module to just 11mA making it suitable for power sensitive applications like GPS wristwatch. The necessary data pins of NEO-6M GPS chip are broken out to a "0.1″ pitch headers. This includes pins required for communication with a microcontroller over UART.

**Note**: - The module supports baud rate from 4800bps to 230400bps with default baud of 9600.

[](javascript:openLightBox('cc4f3a8fc7',%200);)

1. **Position Fix LED Indicator**

There is an LED on the NEO-6M GPS Module which indicates the status of Position Fix. It’ll blink at various rates depending on what state it’s in

* No Blinking ==> means It is searching for satellites
* Blink every 1s – means Position Fix is found

[](javascript:openLightBox('af40ddae40',%200);)

1. **3.3V LDO Regulator**

The operating voltage of the NEO-6M chip is from 2.7 to 3.6V. But, the module comes with MIC5205 ultra-low dropout 3V3 regulator from MICREL. The logic pins are also 5-volt tolerant, so we can easily connect it to an Arduino or any 5V logic microcontroller without using any logic level converter.

[](javascript:openLightBox('7586244481',%200);)

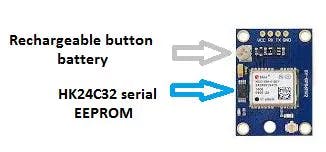
1. **Battery & EEPROM**

The module is equipped with an HK24C32 two wire serial EEPROM. It is 4KB in size and connected to the NEO-6M chip via I2C.The module also contains a rechargeable button battery which acts as a super-capacitor.

An EEPROM together with battery helps retain the battery backed RAM (BBR). The BBR contains clock data, latest position data (GNSS or bit data) and module configuration. But it is not meant for permanent data storage.

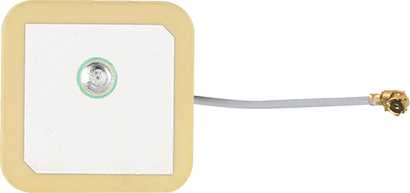
As the battery retains clock and last position, time to first fix (TTFF) significantly reduces to 1s. This allows much faster position locks.

Without the battery the GPS always cold-start so the initial GPS lock takes more time. The battery is automatically charged when power is applied and maintains data for up to two weeks without power.

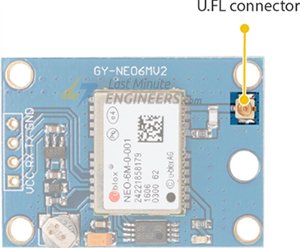
[](javascript:openLightBox('3e7082b631',%200);)

### Antenna

An antenna is required to use the module for any kind of communication. So, the module comes with a patch antenna having -161 dBm sensitivity.

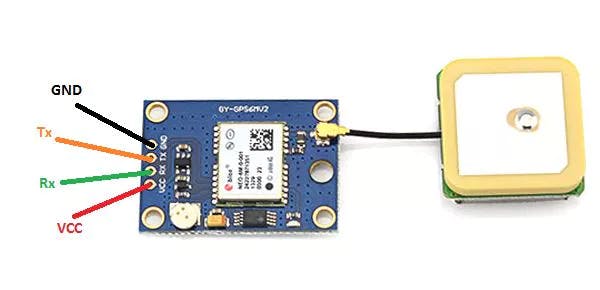


You can snap-fit this antenna to small U.FL connector located on the module.



Patch antenna is great for most projects. But if you want to achieve more sensitivity or put your module inside a metal case, you can also snap on any 3V active GPS antenna via the U.FL connector.

**Pinout**

[](javascript:openLightBox('e5acdffd4b',%200);)

* GND is the Ground Pin and needs to be connected to GND pin on the Arduino.
* TxD (Transmitter) pin is used for serial communication.
* RxD (Receiver) pin is used for serial communication.
* VCC supplies power for the module. You can directly connect it to the 5V pin on the Arduino.

**Specifications:**

|  |  |
| --- | --- |
| Receiver Type | 50 channels, GPS L1(1575.42Mhz) |
| Horizontal Position Accuracy | 2.5m |
| Navigation Update Rate | 1HZ (5Hz maximum) |
| Capture Time | Cool start: 27sHot start: 1s |
| Navigation Sensitivity | -161dBm |
| Communication Protocol | NMEA, UBX Binary, RTCM |
| Serial Baud Rate | 4800-230400 (default 9600) |
| Operating Temperature | -40°C ~ 85°C |
| Operating Voltage | 2.7V ~ 3.6V |
| Operating Current | 45mA |
| TXD/RXD Impedance | 510Ω |

1. Monitor with camera

To monitor the crop field from a remote area we have interfaced the camera to the system. The monitoring process is completely done through wireless transmission

**Camera**

A webcam is a video camera that feeds or streams an image or video in real time to or through a computer to a computer network, such as the Internet. Webcams are typically small cameras that sit on a desk, attach to a user's monitor, or are built into the hardware. Webcams can be used during a video chat session involving two or more people, with conversations that include live audio and video. For example, Apple's iSight camera, which is built into Apple laptops, iMacs and a number of iPhones, can be used for video chat sessions, using the iChat instant messaging program (now called Messages). Webcam software enables users to record a video or stream the video on the Internet. As video streaming over the Internet requires a lot of bandwidth, such streams usually use compressed formats. The maximum resolution of a webcam is also lower than most handheld video cameras, as higher resolutions would be reduced during transmission. The lower resolution enables webcams to be relatively inexpensive compared to most video cameras, but the effect is adequate for video chat sessions



### Optics

Various lenses are available, the most common in consumer-grade webcams being a plastic lens that can be manually moved in and out to focus the camera. Fixed-focus lenses, which have no provision for adjustment, are also available. As a camera system's depth of field is greater for small image formats and is greater for lenses with a large f-number (small aperture), the systems used in webcams have a sufficiently large depth of field that the use of a fixed-focus lens does not impact image sharpness to a great extent.

Most models use simple, focal-free optics (fixed focus, factory-set for the usual distance from the monitor to which it is fastened to the user) or manual focus.

### Compression

Digital video streams are represented by huge amounts of data, burdening its transmission (from the image sensor, where the data is continuously created) and storage alike.

Most if not all cheap webcams come with built-it ASIC to do video compression in real-time.

Support electronics read the image from the sensor and transmit it to the host computer. The camera pictured to the right, for example, uses a Sonix SN9C101 to transmit its image over USB. Typically, each frame is transmitted uncompressed in RGB or YUV or compressed as JPEG. Some cameras, such as mobile-phone cameras, use a CMOS sensor with supporting electronics "on die", i.e. the sensor and the support electronics are built on a single silicon chip to save space and manufacturing costs. Most webcams feature built-in microphones to make video calling and videoconferencing more convenient

### Interface

Typical interfaces used by articles marketed as a "webcam" are USB, Ethernet and IEEE (denominated as IP camera). Further interfaces such as e.g. Composite video or S-Video are also available

The USB video device class (UVC) specification allows inter-connectivity of webcams to computers without the need for proprietary device drivers.

### Software

Various proprietary as well as free and open-source software is available to handle the UVC stream. One could use Guvcview or GStreamer and GStreamer-based software to handle the UVC stream.

**Characteristic:**

Webcams are known for their low manufacturing cost and their high flexibility,making them the lowest-cost form of video telephony. As webcams evolved simultaneously with display technologies, USB interface speeds and broadband internet speeds, the resolution went up from gradually 320×240, to 640×480, and some even offering 1280×720 (aka 720p) or 1920×1080 (aka 1080p) resolution.

Despite the low cost, the resolution offered as of 2019 is impressive, with now the low-end webcams offering resolutions of 720p, mid-range webcams offering 1080p resolution, and high-end webcams offering 4K resolution at 60 fps.

Webcams have become a source of security and privacy issues, as some built-in webcams can be remotely activated by spyware. To address this concern, many webcams come with a physical lens cover.

**Uses:**

The most popular use of webcams is the establishment of video links, permitting computers to act as videophones or videoconference stations. Other popular uses include security surveillance, computer vision, video broadcasting, and for recording social videos.

The video streams provided by webcams can be used for a number of purposes

### Health care

Most modern webcams are capable of capturing arterial pulse rate by the use of a simple algorithmic trick. Researchers claim that this method is accurate to ±5 bpm.

### Video monitoring

Webcams may be installed at places such as childcare centres, offices, shops and private areas to monitor security and general activity.

### Commerce

Webcams have been used for augmented reality experiences online. One such function has the webcam act as a "magic mirror" to allow an online shopper to view a virtual item on themselves. The Webcam Social Shopper is one example of software that utilizes the webcam in this manner.

### Video calling and video conferencing

Further information: Videophone, Videoconferencing, and Video telephony

Webcam can be added to instant messaging, text chat services such as AOL Instant Messenger, and VoIP services such as Skype, one-to-one live video communication over the Internet has now reached millions of mainstream PC users worldwide. Improved video quality has helped webcams encroach on traditional video conferencing systems. New features such as automatic lighting controls, real-time enhancements (retouching, wrinkle smoothing and vertical stretch), automatic face tracking and autofocus, assist users by providing substantial ease-of-use, further increasing the popularity of webcams.

Webcam features and performance can vary by program, computer operating system, and also by the computer's processor capabilities. Video calling support has also been added to several popular instant messaging programs.

### Video security

Webcams can be used as security cameras. Software is available to allow PC-connected cameras to watch for movement and sound, recording both when they are detected. These recordings can then be saved to the computer, e-mailed, or uploaded to the Internet. In one well-publicised case, a computer e-mailed images of the burglar during the theft of the computer, enabling the owner to give police a clear picture of the burglar's face even after the computer had been stolen.

Unauthorized access of webcams can present significant privacy issues (see "Privacy" section below).

### Video clips and stills

Webcams can be used to take video clips and still pictures. Various software tools in wide use can be employed for this, such as PicMaster (for use with Windows operating systems), Photo Booth (Mac), or Cheese (with Unix systems). For a more complete list see Comparison of webcam software.

### Input control devices

Special software can use the video stream from a webcam to assist or enhance a user's control of applications and games. Video features, including faces, shapes, models and colors can be observed and tracked to produce a corresponding form of control. For example, the position of a single light source can be tracked and used to emulate a mouse pointer, a head-mounted light would enable hands-free computing and would greatly improve computer accessibility. This can be applied to games, providing additional control, improved interactivity and immersiveness.

**FreeTrack** is a free webcam motion-tracking application for Microsoft Windows that can track a special head-mounted model in up to six degrees of freedom and output data to mouse, keyboard, joystick and FreeTrack-supported games. By removing the IR filter of the webcam, IR LEDs can be used, which has the advantage of being invisible to the naked eye, removing a distraction from the user. TrackIR is a commercial version of this technology.

The EyeToy for the PlayStation 2, PlayStation Eye for the PlayStation 3, and the Xbox Live Vision camera and Kinect motion sensor for the Xbox 360 and are color digital cameras that have been used as control input devices by some games.

Small webcam-based PC games are available as either standalone executables or inside web browser windows using Adobe Flash

**Raspberry pi software**

This chapter introduces the devices and software which are used in this bachelor's thesis. The chapter also contains short introduction to the Linux operating system which is used in this thesis.

**2.1 Linux**

Linux is a free open source operating system and it belongs to the Unix operating systems. Actually Linux means the kernel itself which is the heart of the operating system and handles the communication between the user and hardware. Normally Linux is used to refer to the whole Linux distribution. (Upton, E. & Halfacree, G. 2012, 28.)

Linux distribution is a collection of software based on the Linux Kernel. It consists of the GNU-project's components and applications. Because Linux is an open source project, anyone can modify and distribute it. That is the reason why there are many variations of Linux distributions. Most popular distributions are Ubuntu, Red Hat Linux, Debian GNU/Linux and SuSe Linux. (Kuutti, W. & Rantala, A. 2007, 2.)

**2.1.1 History of Linux**

Linux is a Unix compatible operating system where the operating system's kernel has been reprogrammed. Because of the compatibility most of the free applica-tions programmed for Unix are also available for Linux. In 1973 Unix was repro-grammed in C programming language instead of the assembly code. At that point Unix reached its current outfit. Unix supported multiple users and it was also easy to transfer to new digital machines. Later Unix was given to Universities for further development. (Kuutti, W. & Rantala, A. 2007, 5-6.)

Linux got started in the early 1990s when Linus Torvalds got tired of MS-DOS op-erating system and decided to create a new operating system for the Intel's cheapx86 processors. At the time there was already available Minix operating system for microcomputers. However, mainly for teaching purposes created Minix was not good enough for Torvalds. (Kuutti, W. & Rantala, A. 2007, 5-6.)

In October 1991 Linus Torvalds released the first unofficial Linux and the first offi-cial Linux version was released in March 1994. Nowadays Linus Torvalds is still partly developing and supporting the kernel's further development. (Kuutti, W. & Rantala, A. 2007, 5-6.)

**2.1.2 Linux compared to Windows**

When comparing Linux and Windows as operating systems, one of the major dif-ferences are that Linux is an open-source project and Windows is a closed-source project. In the closed-source project the users sees only the finished product but do not know how it has been done. In open-source projects everything is made fully visible to the public. (Upton, E. & Halfacree, G. 2012, 13-14.)

In practice this can be seen in Linux's easy customization for different platforms. This process is called porting. There are several distributions ported to the Rasp-berry Pi's BCM2835 chip. One of the distributions is called Raspbian Wheezy. (Up-ton, E. & Halfacree, G. 2012, 14.)

**2.1.3 Raspbian Wheezy**

Raspbian Wheezy is a free operating system based on Debian distribution. It is created by a small team of developers who are fans of Raspberry Pi. Raspbian is optimized for the Raspberry Pi's hardware and it comes with over 35 000 packag-es and pre-compiled software. Raspbian is still under active development and it aims to improve the stability and performance of the Debian packages. (Raspbian [Ref. 15.2.2015])

Raspbian is officially recommended for beginners and it includes the graphical desktop environment called LXDE. Raspbian Wheezy is one of the fastest ways to setup and get the RasPi running. (McManus, S. & Cook, M. 2013, 20.)

**Programming languages**

There are considerable numbers of programming languages which have been adapted for Raspberry Pi. Python programming language is recommended by The Raspberry Pi foundation especially for the beginners. Basically any programming language which can be compiled for ARMv6 can run on the Raspberry Pi. There-fore the users are not restricted to use only the Python. On the Raspberry Pi there are preinstalled several languages for example C, C++, Java, Scratch and Ruby. (Raspberry Pi Foundation [Ref. 5.2.2015].)

**2.4 Python programming language**

Python programming language is developed in the late 1980s at the National Re-search Institute by Guido van Rossum. Python has grown in popularity, and it is widely used commercially. (Upton, E. & Halfacree, G. 2012, 152.)

Python is a flexible and powerful programming language but still it is easy to learn and follow. The clear syntax of Python makes it a valuable tool for users who wants to learn programming. This is one of the reasons why it is recommended by the Raspberry Pi Foundation. Python is published under an open-source license and it is available for different operating systems. Python runs on Linux, OS X and Windows computer systems. (Upton, E. & Halfacree, G. 2012, 152.)

Cross-platform support guarantees that the programs which are written in Python are also compatible in other platforms. There are few exceptions where the pro-grams are not compatible. For instance, when the Python is addressed to use the specific hardware such like Raspberry Pi's GPIO. (Upton, E. & Halfacree, G. 2012, 152.)

**Installing OS to Raspberry pi B+:**

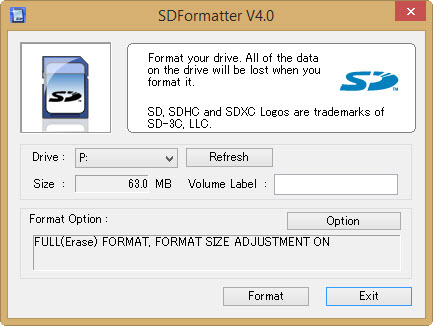
Raspberry Pi doesn’t come with an operating system. This is not a weakness, however, rather a feature that means you can choose from a wide of OSs, each of which can be flashed to an SD card (or microSD card for the [Raspberry Pi B+](http://www.makeuseof.com/tag/the-raspberry-pi-b-is-here-whats-changed/)) in a few simple steps. Here’s how to get a new OS installed and running on your Pi – and how to clone your perfect setup for quick disaster recovery.

Operating systems [such as the recommended Raspbian](http://www.makeuseof.com/tag/optimize-the-power-of-your-raspberry-pi-with-raspbian/), ArchLinux, Risc OS and even Android come ready to run on your Raspberry Pi. I’ll show you the two main ways add an operating system – and once you’ve got your Pi setup how you want it, we’ll look at how to clone the card so that it can be restored following errors (or for temporary reuse of your SD card).

The following tutorials assume that you have a [basic Raspberry Pi package](http://www.makeuseof.com/tag/what-is-the-true-cost-of-running-a-raspberry-pi/) and Windows to manage your SD card writing and cloning.

Flash An OS To SD And Boot Your[Raspberry Pi](http://www.amazon.com/CanaKit-Raspberry-Complete-Original-Preloaded/dp/B008XVAVAW/ref=as_at?tag=mak041-20&linkCode=as2&)

Whichever operating system you download for your Raspberry Pi, the process of writing it to an SD card is the same. However, there are some differences in SD card writing between desktop operating systems. You’ll also need to ensure that your card is blank and formatted, and at least 2 GB.



For Windows users (and also those of you with Mac OS X) the tool of choice for writing a [Raspberry Pi](http://www.amazon.com/CanaKit-Raspberry-Complete-Original-Preloaded/dp/B008XVAVAW/ref=as_at?tag=mak041-20&linkCode=as2&) OS image to SD card is [SD Formatter](https://www.sdcard.org/downloads/formatter_4/), from the SD Association. With the card inserted into your computer’s card reader, and ensuring you have the correct **Drive** letter selected in the drop down menu, open the**Option** menu and select **Full (Erase)** and **On.** This ensures that the full capacity of the storage card will be available.

Click **OK,** then **Format** to begin.

To write the disk image, use [Win32DiskImager](http://sourceforge.net/projects/win32diskimager/), available from Sourceforge. You may need to run with administrator privileges. Select the correct drive letter for your SD card, browse to the image file and click **Write** to commence the process. Win32DiskImager will inform you when the data has been written.

If writing the disk image seems too much hassle or is beyond your abilities, it is possible to [purchase SD cards with Raspbian pre-installed](http://thepihut.com/products/raspbian-preinstalled-sd-card).

Booting Raspbian For The First Time

With Raspbian installed, you’ll need to login with the following credentials:

Username: **pi**

Password: **raspberry**

For other operating systems, check the documentation to find the default login credentials.

Remember that the password will not be displayed as you type it; there are no Windows-style \* symbols representing the letters. Instead, it will appear that you haven’t entered a password. This is a security feature in Linux to prevent people guessing the length of your passphrase. Just type the password regardless.

In Raspbian, you can change your password by running

sudo raspi-config

And selecting the Change Password option.

Install Raspbian, XBMC, Risc OS, OpenElec & ArchLinux Easily With NOOBS

For a simpler, more streamlined install of the more popular [Raspberry Pi](http://www.amazon.com/CanaKit-Raspberry-Complete-Original-Preloaded/dp/B008XVAVAW/ref=as_at?tag=mak041-20&linkCode=as2&) distros such as Raspbian, ArchLinux, XBMC and OpenElec media centres, as well as Risc OS and Pidora – you can employ NOOBS, the New Out Of Box Software installation system.

## Perfecting Your Raspberry Pi OS

Different Raspberry Pi projects require a varying amount of software; it really all depends on what you’re planning to do.

The SD card contains the Raspberry Pi’s operating system (the OS is the software that makes it work, like Windows on a PC or OSX on a Mac). This is very different from most computers and it is what many people find the most daunting part of setting up their Raspberry Pi. It is actually very straightforward— just different!

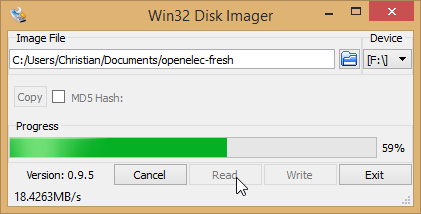
The following instructions are for Windows users. Linux and Mac users can find instructions at [www.raspberrypi.org/downloads](http://www.raspberrypi.org/downloads)

1. Download the Raspberry Pi operating system: The recommended OS is called Raspbian. Download it here: http://downloads.raspberrypi.org/images/raspbian/2012-12-16-wheezy-raspbian/2012-12-16-wheezy-raspbian.zip

2. Unzip the file that you just downloaded: a) Right click on the file and choose “Extract all”. b) Follow the instructions—you will end up with a file ending in .img This .img file can only be written to your SD card by special disk imaging software, so…

3. Download the Win32DiskImager software: a) Download win32diskimager-binary.zip (currently version 0.6) from: https://launchpad.net/win32-image-writer/+download b) Unzip it in the same way you did the Raspbian .zip file c) You now have a new folder called win32diskimager-binary You are now ready to write the Raspbian image to your SD card.

4. Writing Raspbian to the SD card a) Plug your SD card into your PC b) In the folder you made in step 3(b), run the file named Win32DiskImager.exe (in Windows Vista, 7 and 8 we recommend that you right-click this file and choose “Run as administrator”). You will see something like this:



c) If the SD card (Device) you are using isn’t found automatically then click on the drop down box and select it

d) In the Image File box, choose the Raspbian .img file that you downloaded

e) Click Write

f) After a few minutes you will have an SD card that you can use in your Raspberry Pi 5. Booting your Raspberry Pi for the first time a) Follow the Quick start guide on page 1 b) On first boot you will come to the Raspi-config window c) Change settings such as timezone and locale if you want d) Finally, select the second choice: expand\_rootfs and say ‘yes’ to a reboot e) The Raspberry Pi will reboot and you will see raspberrypi login: f) Type: pi g) You will be asked for your Password h) Type: raspberry i) You will then see the prompt: pi@raspberry ~ $ j) Start the desktop by typing:



With this done, you will then need to install various updates from within the media centre software (much as you would if you had installed Raspbian). There may be other applications that you need, add-ins for the media centre such as Vimeo or YouTube, perhaps the TED Talk channel, etc.

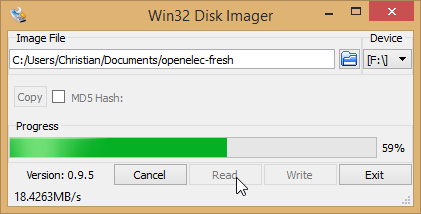
As with any computer system, installing the operating system is only the first step. Updates, optional software and other tools will also need to be added before you’re happy that the Raspberry Pi is ready for its intended use. The same process would be followed if you were configuring your Pi as a wireless print server, perhaps, or even webserver.

Once this is done, you should have your Raspberry Pi installation perfected.

## Disaster Management: Cloning The OS

The point at which you’re happy with the installed OS is precisely when you should make a clone of it. This is a full backup image of the SD card saved to your hard disk drive (or second SD card) that can be easily flashed should you run into trouble.

Windows users can again employ Win32DiskImager here, ensuring that the perfected Raspberry Pi SD card is inserted in your Windows card reader before launching the utility.



Find the **Image File** field, and browse to the folder where you will be saving your SD card image; in the **Device** box, select the drive letter of the SD card, and after confirming both are correct, click **Read** to begin the cloning, following any on-screen instructions. Give the process time to complete before proceeding.

Should you need to revert the SD card back to its “perfect” state later on, you’ll need to remove all partitions (WIN+R to launch Disk Management, right-click the SD card and select **Delete)** before restoring the image. You might also reformat using SD Formatter, as explained above.

Again using Win32DiskImager, browse for the image file, set the **Device** drive letter and then click **Write.** Once the process is complete, your SD card will be reverted to the “perfect” configuration you worked on earlier, and your Raspberry Pi should be ready to put to use once more.

**Python IDE**

**An Introduction to Python**

**The Raspberry Pi** gets the first half of its name from a long-standing tradition of using fruit to name new computing systems—from classic microcomputers like the Acorn, Apricot and Tangerine to more recognizably modern brands including Apple andBlackBerry—but the second half comes courtesy of the Python programming language.

**4.1 Introducing Python**

Flexible and powerful, Python was originally developed in the late 1980s at the National Research Institute for Mathematics andComputer Science by Guido van Rossum as a successor to the ABC language. Since its introduction, Python has grown inpopularity thanks to what is seen as a clear and expressive syntax developed with a focus on ensuring that code is readable.Python is a high-level language. This means that Python code is written in largely recognizable English, providing the Pi withcommands in a manner that is quick to learn and easy to follow. This is in marked contrast to low-level languages, like assembler,which are closer to how the computer “thinks” but almost impossible for a human to follow without experience. The high-levelnature and clear syntax of Python make it a valuable tool for anyone who wants to learn to program. It is also the language that isrecommended by the Raspberry Pi Foundation for those looking to progress from the simple Scratch .

**Example : Hello World**

As you learned in Chapter 10, “An Introduction to Scratch”, the easiest way to learn a new programming language is to create aproject that prints “Hello World!” on the screen. In Scratch, you just had to drag and drop bricks of prewritten code, but inPython, you need to write this program entirely by hand.A Python project is, at heart, nothing more than a text file containing written instructions for the computer to follow. This file canbe created using any text editor. For example, if you enjoy working at the console or in a terminal window, you can use nano; orif you prefer a graphical user interface (GUI), you can use Leafpad. Another alternative is to use an integrated developmentenvironment (IDE) such as IDLE, which provides Python-specific functionality that’s missing from a standard text editor,including syntax checking, debugging facilities and the ability to run your program without having to leave the editor. This chaptergives you instructions on how to create Python files using IDLE, but of course, the IDE program that you choose to use forprogramming is up to you. The chapter also includes instructions for running your created files directly from the terminal, whichcan be used in conjunction with any text editor or other IDE.To begin the Hello World project, open IDLE from the Programming menu in the Debian distribution’s desktop environment. Ifyou’re not using IDLE, create a blank document in your favourite text editor and skip the rest of this paragraph. By default,IDLE opens up in Python shell mode (see Figure 11-1), so anything you type in the initial window will be immediately executed.To open a new Python project which can be executed later, click on the File menu and choose New Window to open a blank

file.The IDLE Python Shell window.

It’s good practice to start all Python programs with a line known as a shebang, which gets its name from the # and !charactersat the beginning of the line. This line tells the operating system where it should look for the Python files. Although this is notentirely necessary for programs that will be run from within IDLE or will call Python explicitly at the terminal, it is required forprograms that are run directly by calling the program’s filename.To ensure the program runs regardless of where the Python executable is installed, the first line of your program should read asfollows:#!/usr/bin/env python.

This line tells the operating system to look at the $PATH environment variable—which is where Linux stores the location of filesthat can be executed as programs—for the location of Python, which should work on any Linux distribution used on the Pi. The$PATH variable contains a list of directories where executable files are stored, and is used to find programs when you type theirname at the console or in a terminal window.To achieve the goal of printing out a message, you should use Python’s print command. As its name suggests, this commandprints text to an output device—by default, to the console or terminal window from which the program is being executed. Itsusage is simple: any text following the word print and placed between quotation marks will be printed to the standard outputdevice. Enter the following line in your new project:

print “Hello, World!”

The final program should look like this:

#!/usr/bin/env python

print “Hello, World!”

If you’re creating the example program in IDLE rather than a plain text editor, you’ll notice that the text is multicolored, (where colours are represented as differing shades of grey in the print edition). This is a feature known as syntaxhighlighting, and is a feature of IDEs and the more-advanced text editing tools. Syntax highlighting changes the colour of sectionsof the text according to their function, in order to make the program easier to understand at a glance. It also makes it easy tospot so-called syntax errors caused by forgetting to put an end-quote in a print command or forgetting to comment out aremark. For this short example, syntax highlighting isn’t necessary—but in larger programs, it can be an invaluable tool for findingerrors.Syntax highlighting in IDLE

Before you run your program, save it as helloworld.py using the File menu. If you’re using IDLE, the file will be given theextension .py automatically. If you’re using a text editor, be sure to type .py at the end of the filename (not .txt) when yousave it. This extension indicates that the file contains Python code—although Python is clever enough to run the program even if

it’s saved with a different file extension.How you run the file will depend on whether you’re using IDLE or a text editor. In IDLE, simply choose Run Module from theRun menu, or press the F5 key on the keyboard. This will switch IDLE back to the Python shell window and run the program.You should then see the message Hello, World! appear onscreen in blue. If not, check your syntax—inparticular, check that you have quotation marks at both the beginning and end of the message on the print line.

Running helloworld.py in IDLE

If you created the helloworld.py program in a text editor, you’ll need to open a terminal window from the Accessories menuon the desktop. If you saved the file anywhere except your home directory, you’ll also have to use the cd command to change tothat directory (see Chapter 2, “Linux System Administration”). Once you’re in the right directory, you can run your program bytyping the following:python helloworld.pyThis tells the operating system to run Python and then load the helloworld.py file for execution. Unlike the Python shell inIDLE, Python will quit when it reaches the end of the file and return you to the terminal. The result, however, is the same: themessage Hello, World! is printed to the standard output .

Running helloworld.py at the terminal

Making Python Programs ExecutableNormally, the only way to run a Python program is to tell the Python software to open the file. With the shebang line at the top of the file, however, it’spossible to execute the file directly without having to call Python first. This can be a useful way of making your own tools that can be executed at theterminal: once copied into a location in the system’s $PATH environment variable, the Python program can be called simply by typing its name.First, you need to tell Linux that the Python file should be marked as executable—an attribute that means the file is a program. To protect the system frommalware being downloaded from the Internet this attribute isn’t automatically set, since only files that are marked as executable will run. To make thehelloworld.py file executable, use the chmod command (described in detail in Chapter 2, “Linux System Administration”) by typing the following:chmod +x helloworld.pyNow try running the program directly by typing the following:

./helloworld.py

Despite the fact that you didn’t call the Python program, the helloworld.py program should run just the same as if you’d typed python helloworld.py.The program can only be run by calling it with its full location—/home/pi/helloworld.py—or from the current directory by using ./ as the location. Tomake the file accessible in the same way as any other terminal command, it needs to be copied to /usr/local/bin with the following command:sudocp helloworld.py /usr/local/bin/

The sudo prefix is required because, for security reasons, non-privileged users cannot write to the /usr/local/bin directory. With the helloworld.py filelocated in /usr/local/bin, which is included in the $PATH variable, it can be executed from any directory by simply typing its name. Try changing to adifferent directory, and then run the program by typing the following:helloworld.pyTo make your custom-made programs seem more like native utilities, you can rename them to remove the .py file extension. To change the helloworld.pyprogram in this way, just type the following line at the terminal as a single line:sudo mv /usr/local/bin/helloworld.py

/usr/local/bin/helloworldOnce renamed, the program can be run simply by typing helloworld at the terminal or console.

**4.2 Python and Networking**

So far, you have learned how Python can be used to create standalone programs, but the language can also be used to createprograms that communicate with the outside world over a computer’s network connection. This next example, written by TomHudson, offers a brief glimpse of these possibilities with a tool for monitoring the users connected to an Internet Relay Chat(IRC) channel.As usual, create a new project in IDLE or a text editor and enter the shebang line along with a comment describing the purposeof the program:

#!/usr/bin/env python

# IRC Channel Checker, written for theRaspberry Pi User Guide by Tom HudsonNext, import the modules required by the program—sys, socket and time—with the following line:import sys, socket, timeYou used the sys and time modules previously in the Raspberry Snake program, but you have not yet used socket. Thesocket module provides Python with the ability to open, close, read from and write to network sockets—giving Pythonprograms rudimentary networking capabilities. It’s the socket module that provides this example with its ability to connect to a

remote IRC server.

There are some constants needed for this program to operate. Constants are like variables in that they can have values assignedto them—but unlike variables, the value in a constant shouldn’t change. To help differentiate a constant from a variable, it’s goodpractice to use all-capital letters for their names—that way it’s easy to see at glance whether a particular section of the code isusing a constant or a variable. Type the following two lines into the program:

RPL\_NAMREPLY = ‘353’

RPL\_ENDOFNAMES = ‘366’

These are IRC status codes, provided by the server to indicate when particular operations have completed. These are used bythe program to know when it has received the required list of names from the IRC server. Next, set up the variables for theserver connection by entering the

following lines:

irc = {

‘host’ : ‘chat.freenode.net’,

‘port’ : 6667,

‘channel’ : ‘#raspiuserguide’,

‘namesinterval’ : 5

}

The first line tells Python to create a dict data type. Short for dictionary, this allows multiple variables to be stored in a singlemaster variable—in this case, irc. These individual variables can then be recalled later in the program. Although you could writethis program without using dicts to store variables, it would make the program significantly more difficult to read. The dict beginswith the opening curly brace, and ends with the closing curly brace on the final line.The host variable should be set to the fully-qualified domain name (FQDN) of the IRC server to which the program willconnect. In this example, chat.freenode.net is used, but if you want to customise the program to use a different server,change the domain name here. The port variable tells the program which network port IRC is running on, which will usually be6667. The channel variable tells Python which channel to join in order to monitor the users, while namesinterval controlshow long the program waits to refresh the list of users, measured in seconds.Set up a second dict to store the user-specific variables by typing in the following lines:

user = {

‘nick’ : ‘botnick’,

‘username’ : ‘botuser’,

‘hostname’ : ‘localhost’,

‘servername’ : ‘localhost’,

‘realname’ : ‘Raspberry Pi Names Bot’

}

As with irc, all these variables are stored within a dict called user to make it clear which variables pertain to which section. Thenick variable should be set to the IRC nickname the program will use. Don’t use your usual nickname if you’re planning toconnect to the IRC server at the same time; instead, try appending -bot to the end of your name to make it clear that the user isa program rather than a real person. Do the same with username, and fill in the realname variable with a descriptive messageabout whom the bot belongs to. The hostname and servername variables can be left set to localhost, or altered to matchyour Internet address.

The socket module requires the user to create a socket object. This object provides network connectivity to the rest of theprogram. Create the socket object by typing in the following line:

s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)Next, you need to tell the program to try connecting to the IRC server specified in the variables at the start of the program. Type

the following lines:

print ‘Connecting to %(host)s:%(port)s...’ % irc

try:s.connect((irc[‘host’], irc[‘port’]))

exceptsocket.error:

print ‘Error connecting to IRC server

%(host)s:%(port)s’ % irc

sys.exit(1)

The try and except commands are included in this code for error handling. If the system fails to connect to the server—because the Pi isn’t connected to the Internet, for example, or because the server is down for maintenance—the program willprint an error message and gracefully exit. The s.connect line tells the socket module to try connecting to the IRC server, usingthe host and port variables held in the irc dict.If the program doesn’t quit from the exception, it has successfully connected to the IRC server. Before you can get a list ofnames in a channel, however, you need to identify yourself to the server and issue some commands using the send function of thesocket module. Type the following lines into the program:

s.send(‘NICK %(nick)s\r\n’ % user)

s.send(‘USER %(username)s %(hostname)s

%(servername)s :%(realname)s\r\n’ % user)

s.send(‘JOIN %(channel)s\r\n’ % irc)

s.send(‘NAMES %(channel)s\r\n’ % irc)

The send function works in almost exactly the same way as the print function, except that instead of printing to the standardoutput—usually the terminal window or console—it sends the output through the network connection. In this case, the programis sending strings of text to the IRC server and telling it to register the program using the nickname held in the nick variable andthe user details held in the username, hostname, servername and realname variables. Next, the program sends thecommand to join the channel specified in the channel variable, and finally, it sends the command to receive the list of users inthat channel. Although this example is tailored to IRC, the same basic principle can be used to issue commands to any networkservice—with modifications, this program could be used to list the files on an FTP server, or unread emails on a POP3 server.Receiving data from the socket is a little more complicated. First, you’ll need to create an empty string variable that will act as thereceive buffer, holding data from the server as it’s received until it can be processed. Initialise the buffer by typing in the followingline:

read\_buffer = ‘‘Note that there are two single quotes after the equals sign, not one double quote.

Next, create an empty list ,which will be used to store the names of users, by typing the following line:names = []

The list data type is the same as you used to store the locations in the Raspberry Snake game. Unlike a normal variable, it canstore multiple values—in this case, the names of users present in the IRC channel.The next step is to create an infinite loop, during which the program will continuously query the server for user names and printthem to the screen. Start the loop by typing:

while True:

read\_buffer += s.recv(1024)

The first line of the loop, following while True:, tells the socket module to receive 1,024 bytes (1 KB) of data from the IRCserver and place it into the read\_buffer variable. Because the += operator is used, rather than just =, the received data will beappended to anything already in the buffer. The value of 1024 bytes is more or less arbitrary.The next step is to split the buffer into individual lines of text, using the following program lines:

lines = read\_buffer.split(‘\r\n’)

read\_buffer = lines.pop();

The first line sets the lines variable to a full line of text from the receive buffer, by using the split function to find end of linecharacters—signified by \r\n. These characters only occur at the end of a line, so when the buffer has been split in this way youknow that lines contains only full-line responses from the server. The pop instruction in the second line makes sure that only fulllines are removed from the read\_buffer: because responses from the server are read in 1 KB chunks, it’s likely that at anygiven time the buffer will contain only fractions of a line. When that’s the case, the fraction is left in the buffer ready to receive theremainder of the line the next time the loop runs and the next 1 KB chunk is received from the server.At this point, the lines variable contains a list of full responses—full lines—received from the server. Type the following toprocess these lines and find the names of channel participants:for line in lines:

response = line.rstrip().split(‘ ‘, 3)

response\_code = response[1]

ifresponse\_code == RPL\_NAMREPLY:

names\_list = response[3].split(‘:’)[1]

names += names\_list.split(‘ ‘)

This runs through every line found in the lines variable, and looks for the numerical IRC response code provided by the server.Although there are plenty of different response codes, this program is only interested in the two defined as constants at the startof the program: 353, which means a list of names follows, and 366, which means the list has ended. The if statement looks forthe first of these responses, and then uses the split function to retrieve these names and add them to the names list.Now, the names list contains all the names received from the server in response to the program’s query. This may not be all thenames, however: until the 366 response, which signals the end of the member names, is received, the list is incomplete. That iswhy the last line—names += names\_list.split(‘ ‘)—is appending the newly-received names to the existing list, ratherthan blanking it out entirely: each time that section of the code runs, the program is only likely to have received a sub-section ofthe entire member list. To tell Python what to do when the full list has been received, enter the following lines:

ifresponse\_code == RPL\_ENDOFNAMES:

# Display the names

print ‘\r\nUsers in %(channel)s:’ % irc

for name in names:

print name

names = []

This tells Python that when the 366 response has been received, it should print out the now-complete list of names to thestandard output before blanking the names list again. This last line—names = []—is important: without it, each time the loopruns it will add users’ names to the list even though they already exist from an earlier run.Finally, finish the program by entering the following lines:

time.sleep(irc[‘namesinterval’])

s.send(‘NAMES %(channel)s\r\n’ % irc)

This tells Python to wait the namesinterval number of seconds before sending another request for user names and beginningthe loop again. Be careful to set namesinterval to a reasonable value—if the IRC server receives too many requests in tooshort a space of time, it may forcibly disconnect you for flooding.Save the program as ircuserlist.py, and run it either by using IDLE’s Run Module option in the Run menu or from theterminal by typing python ircuserlist.py. When the program first runs, it may take a while to connect to the server; onceconnected, however, the list of names (see Figure 11-7) should refresh quickly. To quit the program, press CRTL + C.

Using Python to list users in an IRC channel

A full copy of the program listing for the IRC user list is included in Appendix A, “Python Recipes”, and on the Raspberry PiUser Guide website at http://www.wiley.com/go/raspberrypiuserguide. Downloading the source code from thewebsite will save you some typing, but entering the code by hand is a good way of ensuring that you understand what eachsection does.

**CONCLUSION**

According to this system, the blind people interact with the system in voice. The cloud platform communicates with the smartphone through Wi-Fi or 4G mobile communication technology. For testing the system performance, two groups of tests have been conducted. One is perception and the other is navigation. Test results show that the proposed system can provide more abundant surrounding information and more accurate navigation, and verify the practicability

**REFERENCES**

# [1] “Blind navigation system for visually impaired using windowing-based mean on Microsoft Kinect camera” in [2017 Fourth International Conference on Advances in Biomedical Engineering (ICABME)](https://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=8119420).

# [2] “Mobile phone based indoor navigation system for blind and visually impaired people: VUK — Visionless supporting framework” in [2018 19th International Carpathian Control Conference (ICCC)](https://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=8387527).

# [3] “Virtual-Blind-Road Following-Based Wearable Navigation Device for Blind People” in [IEEE Transactions on Consumer Electronics](https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=30) (Volume: 64 , [Issue: 1](https://ieeexplore.ieee.org/xpl/tocresult.jsp?isnumber=8328045) , Feb. 2018 ).

# [4] P.-H. Tseng, Z. Ding, and K.-T. Feng, “Cooperative self-navigation in a mixed los and nlos environment,” IEEE Transactions on Mobile Computing, vol. 13, no. 2, pp. 350–363, 2014.

# [5] L.Wang,Y.He,W.Liu,N.Jing,J.Wang,andY.Liu,“Onoscillation-free emergency navigation via wireless sensor networks,” IEEE Transactions on Mobile Computing, vol. 14, no. 10, pp. 2086–2100, 2015.

# [6] C. Wang, H. Lin, and H. Jiang, “Cans: Towards congestion-adaptive and smallstretchemergencynavigationwithwirelesssensornetworks,”IEEE Transactions on Mobile Computing, vol. 15, no. 5, pp. 1077–1089, 2016.