TITLE

**DRONE DETECTION IN RESTRICTED AREAS USING OPENCV**

ABSTRACT:

***Main use of this project is to detect the drone at any type. We can detect with in seconds using YOLO with RCNN. using this we can detect drones,. Yolo gives better output and taking less speed than others. Real-time apple detection in orchards is one of the most effective ways of estimating apple yields, which helps in managing apple supplies more effectively. Traditional detection methods used highly computational machine learning algorithms with intensive hardware set up, which are not suitable for infield real-time apple detection due to their weight and power constraints. In this study, a real-time embedded solution inspired from “Edge AI” is proposed for apple detection with the implementation of YOLOv3-tiny algorithm on various embedded platforms such as Raspberry Pi 3 B+ in combination with Intel Movidius Neural Computing Stick (NCS), Nvidia's Jetson Nano and Jetson AGX Xavier. Data set for training were compiled using acquired images during field survey of apple orchard situated in the north region of Italy, and images used for testing were taken from widely used google data set by filtering out the images***

***Index terms: yolo v4, drone, smtp, object detection***

***containing apples in different scenes to ensure the robustness of the algorithm. The proposed study adapts YOLOv3-tiny architecture to detect small objects. It shows the feasibility of deployment of the customized model on cheap and power-efficient embedded hardware without compromising mean average detection accuracy (83.64%) and achieved frame rate up to 30 fps even for the difficult scenarios such as overlapping apples, complex background, less exposure of apple due to leaves and branches. Furthermore, the proposed embedded solution can be deployed on the unmanned ground vehicles to detect, count, and measure the size of the apples in real-time to help the farmers and agronomists in their decision making and management skills.these project doing with the hardware part of raspberry-pi for the real time applications.***

INTRODUCTION:

Drones are known as Unmanned Aerial Vehicles (UAV) is an aircraft without a human pilot aboard. Drones may operate with various degrees of autonomy either under remote control by a human operator or autonomously by onboard computers .Drones are originally used for missions too dirty or dangerous for humans while they originated mostly in military applications. Compared to the manned aircraft, drones are easily man unbearable and can be operated from anywhere. They provide greater visibility for the pilots operating them because they are guided by satellites and have highly advanced cameras. Nowadays, drones are rapidly used expanding to commercial, scientific, recreational, agricultural, and other applications such as policing, peacekeeping, surveillance, product deliveries, aerial photography, agriculture, smuggling, and drone racing.

**DIGITAL IMAGE PROCESSING**

The identification of objects in an image and this process would probably start with image processing techniques such as noise removal, followed by (low-level) feature extraction to locate lines, regions and possibly areas with certain textures.

The clever bit is to interpret collections of these shapes as single objects, e.g. cars on a road, boxes on a conveyor belt or cancerous cells on a microscope slide. One reason this is an AI problem is that an object can appear very different when viewed from different angles or under different lighting. Another problem is deciding what features belong to what object and which are background or shadows etc. The human visual system performs these tasks mostly unconsciously but a computer requires skilful programming and lots of processing power to approach human performance. Manipulation of data in the form of an image through several possible techniques. An image is usually interpreted as a two-dimensional array of brightness values, and is most familiarly represented by such patterns as those of a photographic print, slide, television screen, or movie screen. An image can be processed optically or digitally with a computer.

1. **Basics of Image Processing:-**

**FUNDAMENTALS OF DIGITAL IMAGE**

**1.1 IMAGE:**

An image is a two-dimensional picture, which has a similar appearance to some subject usually a physical object or a person.

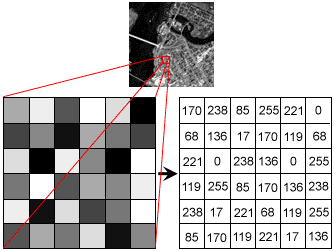
Image is a two-dimensional, such as a photograph, screen display, and as well as a three-dimensional, such as a statue. They may be captured by optical devices—such as cameras, mirrors, lenses, telescopes, microscopes, etc. and natural objects and phenomena, such as the human eye or water surfaces.

The word image is also used in the broader sense of any two-dimensional figure such as a map, a graph, a pie chart, or an abstract painting. In this wider sense, images can also be rendered manually, such as by drawing, painting, carving, rendered automatically by printing or computer graphics technology, or developed by a combination of methods, especially in a pseudo-photograph.

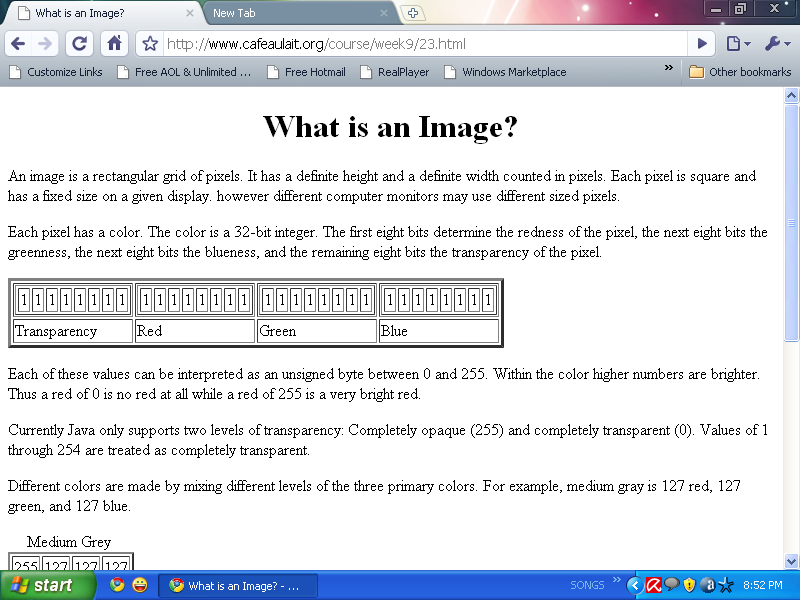
**Fig: Colour image to Gray scale Conversion Process**

An image is a rectangular grid of pixels. It has a definite height and a definite width counted in pixels. Each pixel is square and has a fixed size on a given display. However different computer monitors may use different sized pixels. The pixels that constitute an image are ordered as a grid (columns and rows); each pixel consists of numbers representing magnitudes of brightness and color.



**Fig: Gray Scale Image Pixel Value Analysis**

Each pixel has a color. The color is a 32-bit integer. The first eight bits determine the redness of the pixel, the next eight bits the greenness, the next eight bits the blueness, and the remaining eight bits the transparency of the pixel.



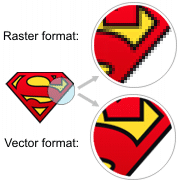
**Fig: BIT Transferred for Red, Green and Blue plane (24bit=8bit red;8-bit green;8bit blue)**

**IMAGE FILE SIZES:**

Image file size is expressed as the number of bytes that increases with the number of pixels composing an image, and the color depth of the pixels. The greater the number of rows and columns, the greater the image resolution, and the larger the file. Also, each pixel of an image increases in size when its color depth increases, an 8-bit pixel (1 byte) stores 256 colors, a 24-bit pixel (3 bytes) stores 16 million colors, the latter known as true color.Image compression uses algorithms to decrease the size of a file. High resolution cameras produce large image files, ranging from hundreds of kilobytes to megabytes, per the camera's resolution and the image-storage format capacity. High resolution digital cameras record 12 megapixel (1MP = 1,000,000 pixels / 1 million) images, or more, in true color. For example, an image recorded by a 12 MP camera; since each pixel uses 3 bytes to record true color, the uncompressed image would occupy 36,000,000 bytes of memory, a great amount of digital storage for one image, given that cameras must record and store many images to be practical. Faced with large file sizes, both within the camera and a storage disc, image file formats were developed to store such large images.

**IMAGE FILE FORMATS:**

Image file formats are standardized means of organizing and storing images. This entry is about digital image formats used to store photographic and other images. Image files are composed of either pixel or vector (geometric) data that are rasterized to pixels when displayed (with few exceptions) in a vector graphic display. Including proprietary types, there are hundreds of image file types. The PNG, JPEG, and GIF formats are most often used to display images on the Internet.



**Fig: Horizontal and Vertical Process**

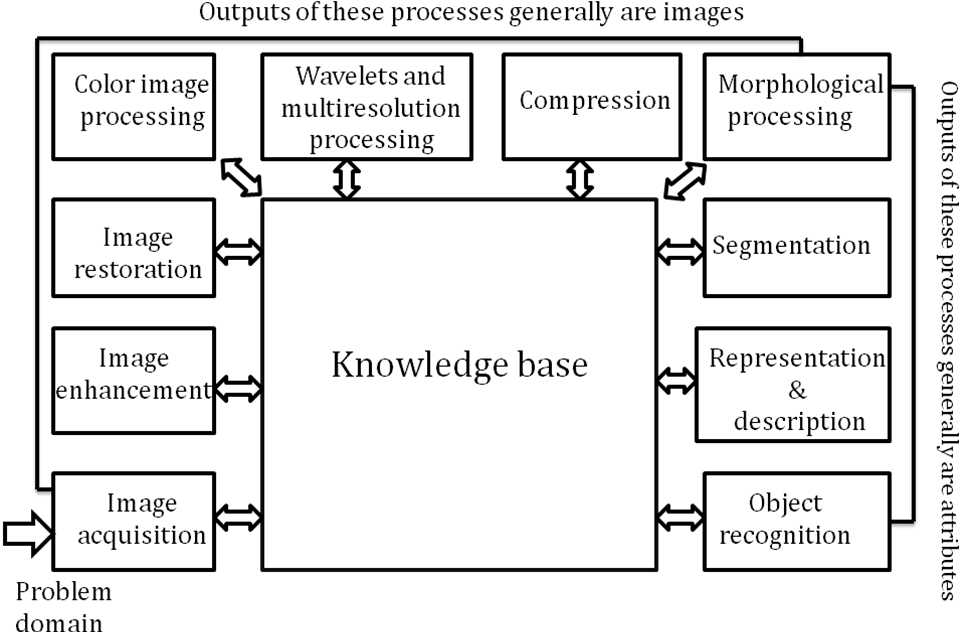
In addition to straight image formats, Metafile formats are portable formats which can include both raster and vector information. The metafile format is an intermediate format. Most Windows applications open metafiles and then save them in their own native format.

**IMAGE PROCESSING:**

Digital image processing, the manipulation of images by computer, is relatively recent development in terms of man’s ancient fascination with visual stimuli. In its short history, it has been applied to practically every type of images with varying degree of success. The inherent subjective appeal of pictorial displays attracts perhaps a disproportionate amount of attention from the scientists and also from the layman. Digital image processing like other glamour fields, suffers from myths, mis-connect ions, mis-understandings and mis-information. It is vast umbrella under which fall diverse aspect of optics, electronics, mathematics, photography graphics and computer technology. It is truly multidisciplinary endeavor ploughed with imprecise jargon.

Several factor combine to indicate a lively future for digital image processing. A major factor is the declining cost of computer equipment. Several new technological trends promise to further promote digital image processing. These include parallel processing mode practical by low cost microprocessors, and the use of charge coupled devices (CCDs) for digitizing, storage during processing and display and large low cost of image storage arrays.

**FUNDAMENTAL STEPS IN DIGITAL IMAGE PROCESSING:**

**Fig: Basics steps of image Processing**

**Image Acquisition:**

**Image Acquisition** is to acquire a digital image. To do so requires an image sensor and the capability to digitize the signal produced by the sensor. The sensor could be monochrome or color TV camera that produces an entire image of the problem domain every 1/30 sec. the image sensor could also be line scan camera that produces a single image line at a time. In this case, the objects motion past the line.



**Fig: Digital camera**

Scanner produces a two-dimensional image. If the output of the camera or other imaging sensor is not in digital form, an analog to digital converter digitizes it. The nature of the sensor and the image it produces are determined by the application.



**Fig: Mobile based Camera**

**Image Enhancement:**

**Image enhancement**is among the simplest and most appealing areas of digital image processing. Basically, the idea behind enhancement techniques is to bring out detail that is obscured, or simply to highlight certain features of interesting an image. A familiar example of enhancement is when we increase the contrast of an image because “it looks better.” It is important to keep in mind that enhancement is a very subjective area of image processing.

  
**Fig: Image enhancement process for Gray Scale Image and Colour Image using Histogram Bits**

**1.5.3 Image restoration:**

**Image restoration**is an area that also deals with improving the appearance of an image. However, unlike enhancement, which is subjective, image restoration is objective, in the sense that restoration techniques tend to be based on mathematical or probabilistic models of image degradation.



Fig: Noise image🡪 Image Enhancement

Enhancement, on the other hand, is based on human subjective preferences regarding what constitutes a “good” enhancement result. For example, contrast stretching is considered an enhancement technique because it is based primarily on the pleasing aspects it might present to the viewer, where as removal of image blur by applying a deblurring function is considered a restoration technique.

**Color image processing:**

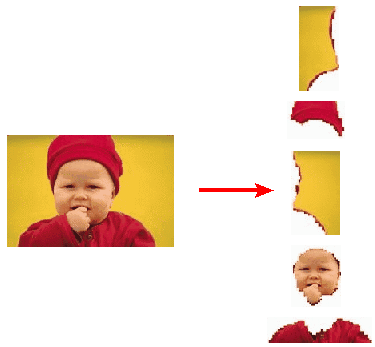
The use of color in image processing is motivated by two principal factors. First, color is a powerful descriptor that often simplifies object identification and extraction from a scene. Second, humans can discern thousands of color shades and intensities, compared to about only two dozen shades of gray. This second factor is particularly important in manual image analysis.

****

**Fig: gray Scale image 🡪 Colour Image**

**Segmentation:**

**Segmentation**procedures partition an image into its constituent parts or objects. In general, autonomous segmentation is one of the most difficult tasks in digital image processing. A rugged segmentation procedure brings the process a long way toward successful solution of imaging problems that require objects to be identified individually.



**Fig: Image Segment Process**

On the other hand, weak or erratic segmentation algorithms almost always guarantee eventual failure. In general, the more accurate the segmentation, the more likely recognition is to succeed.

Digital image is defined as a two dimensional function f(x, y), where x and y are spatial (plane) coordinates, and the amplitude of f at any pair of coordinates (x, y) is called intensity or grey level of the image at that point. The field of digital image processing refers to processing digital images by means of a digital computer. The digital image is composed of a finite number of elements, each of which has a particular location and value. The elements are referred to as picture elements, image elements, pels, and pixels. Pixel is the term most widely used.

**Image Compression**

Digital Image compression addresses the problem of reducing the amount of data required to represent a digital image. The underlying basis of the reduction process is removal of redundant data. From the mathematical viewpoint, this amounts to transforming a 2D pixel array into a statically uncorrelated data set. The data redundancy is not an abstract concept but a mathematically quantifiable entity. If n1 and n2 denote the number of information-carrying units in two data sets that represent the same information, the relative data redundancy  [2] of the first data set (the one characterized by n1) can be defined as,



Where  called as compression ratio [2]. It is defined as

= 

In image compression, three basic data redundancies can be identified and exploited: Coding redundancy, interpixel redundancy, and phychovisal redundancy. Image compression is achieved when one or more of these redundancies are reduced or eliminated. The image compression is mainly used for image transmission and storage. Image transmission applications are in broadcast television; remote sensing via satellite, air-craft, radar, or sonar; teleconferencing; computer communications; and facsimile transmission. Image storage is required most commonly for educational and business documents, medical images that arise in computer tomography (CT), magnetic resonance imaging (MRI) and digital radiology, motion pictures, satellite images, weather maps, geological surveys, and so on.

**Image Compression Model**

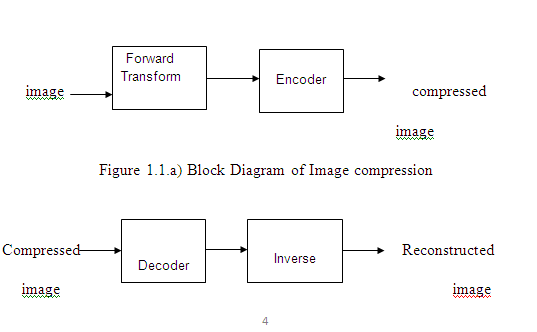


Fig:1.1b) Decompression Process for Image

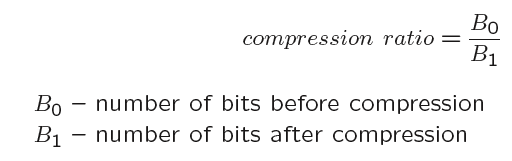
**Image Compression Types**

There are two types’ image compression techniques.

1. Lossy Image compression

2. Lossless Image compression

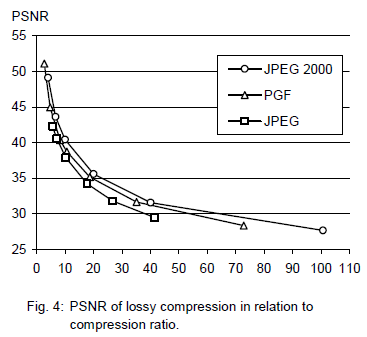
Compression ratio:



**1. Lossy Image compression :**

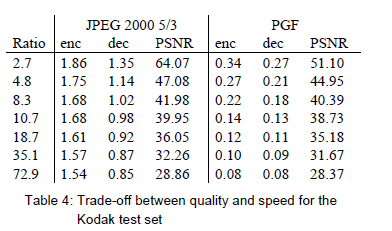
Lossy compression provides higher levels of data reduction but result in a less than perfect reproduction of the original image. It provides high compression ratio. lossy image compression is useful in applications such as broadcast television, videoconferencing, and facsimile transmission, in which a certain amount of error is an acceptable trade-off for increased compression performance. Originally, PGF has been designed to quickly and progressively decode lossy compressed aerial images. A lossy compression mode has been preferred, because in an application like a terrain explorer texture data (e.g., aerial orthophotos) is usually mid-mapped filtered and therefore lossy mapped onto the terrain surface. In addition, decoding lossy compressed images is usually faster than decoding lossless compressed images.

In the next test series we evaluate the lossy compression efficiency of PGF. One of the best competitors in this area is for sure JPEG 2000. Since JPEG 2000 has two different filters, we used the one with the better trade-off between compression efficiency and runtime. On our machine the 5/3 filter set has a better trade-off than the other. However, JPEG 2000 has in both cases a remarkable good compression efficiency for very high compression ratios but also a very poor encoding and decoding speed. The other competitor is JPEG. JPEG is one of the most popular image file formats.

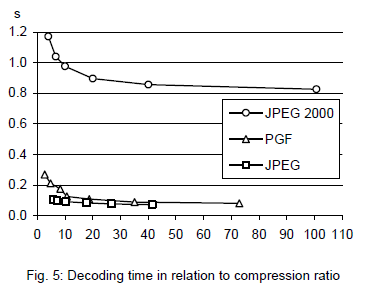


It is very fast and has a reasonably good compression efficiency for a wide range of compression ratios. The drawbacks of JPEG are the missing lossless compression and the often missing progressive decoding. Fig. 4 depicts the average rate-distortion behavior for the images in the Kodak test set when fixed (i.e., nonprogressive) lossy compression is used. The PSNR of PGF is on average 3% smaller than the PSNR of JPEG 2000, but 3% better than JPEG.

These results are also qualitative valid for our PGF test set and they are characteristic for aerial ortho-photos and natural images. Because of the design of PGF we already know that PGF does not reach the compression efficiency of JPEG 2000. However, we are interested in the trade-off between compression efficiency and runtime. To report this trade-off we show in Table 4 a comparison between JPEG 2000 and PGF and in Fig. 5 (on page 8) we show for the same test series as in Fig. 4 the corresponding average decoding times in relation to compression ratios.Table 4 contains for seven different compression ratios (mean values over the compression ratios of the eight images of the Kodak test set) the corresponding average encoding and decoding times in relation to the average PSNR values. In case of PGF the encoding time is always slightly longer than the corresponding decoding time. The reason for that is that the actual encoding phase (cf. Subsection 2.4.2) takes slightly longer than the corresponding decoding phase. For six of seven ratios the PSNR difference between JPEG 2000 and PGF is within 3% of the PSNR of JPEG 2000. Only in the first row is the difference larger (21%), but because a PSNR of 50 corresponds to an almost perfect image quality the large PSNR difference corresponds with an almost undiscoverable visual difference. The price they pay in JPEG 2000 for the 3% more PSNR is very high. The creation of a PGF is five to twenty times faster than the creation of a corresponding JPEG 2000 file, and the decoding of the created PGF is still five to ten times faster than the decoding of the JPEG 2000 file. This gain in speed is remarkable, especially in areas where time is more important than quality, maybe for instance in real-time computation.



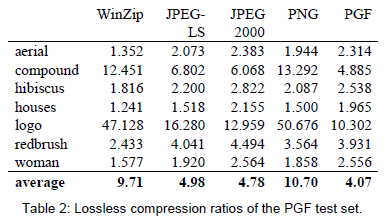
In Fig. 5 we see that the price we pay in PGF for the 3% more PSNR than JPEG is low: for small compression ratios (< 9) decoding in PGF takes two times longer than JPEG and for higher compression ratios (> 30) it takes only ten percent longer than JPEG. These test results are characteristic for both natural images and aerial ortho-photos. Again, in the third test series we only use the ‘Lena’ image. We run our lossy coder with six different quantization parameters and measure the PSNR in relation to the resulting compression ratios. The results (ratio: PSNR) are:



**2.Lossless Image compression :**

Lossless Image compression is the only acceptable amount of data reduction. It provides low compression ratio while compared to lossy. In Lossless Image compression techniques are composed of two relatively independent operations: (1) devising an alternative representation of the image in which its interpixel redundancies are reduced and (2) coding the representation to eliminate coding redundancies.

Lossless Image compression is useful in applications such as medical imaginary, business documents and satellite images.Table 2 summarizes the lossless compression efficiency and Table 3 the coding times of the PGF test set. For WinZip we only provide average runtime values, because of missing source code we have to use an interactive testing procedure with runtimes measured by hand. All other values are measured in batch mode.



In Table 2 it can be seen that in almost all cases the best compression ratio is obtained by JPEG 2000, followed by PGF, JPEG-LS, and PNG. This result is different to the result in [SEA+00], where the best performance for a similar test set has been reported for JPEG-LS. PGF performs between 0.5% (woman) and 21.3% (logo) worse than JPEG 2000. On average it is almost 15% worse. The two exceptions to the general trend are the ‘compound’ and the ‘logo’ images. Both images contain for the most part black text on a white background. For this type of images, JPEG-LS and in particular WinZip and PNG provide much larger compression ratios. However, in average PNG performs the best, which is also reported in [SEA+00].

These results show, that as far as lossless compression is concerned, PGF performs reasonably well on natural and aerial images. In specific types of images such as ‘compound’ and ‘logo’ PGF is outperformed by far in PNG.

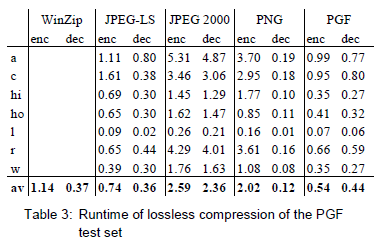


Table 3 shows the encoding (enc) and decoding (dec) times (measured in seconds) for the same algorithms and images as in Table 2. JPEG 2000 and PGF are both symmetric algorithms, while WinZip, JPEG-LS and in particular PNG are asymmetric with a clearly shorter decoding than encoding time. JPEG 2000, the slowest in encoding and decoding, takes more than four times longer than PGF. This speed gain is due to the simpler coding phase of PGF. JPEG-LS is slightly slower than PGF during encoding, but slightly faster in decoding images.

WinZip and PNG decode even more faster than JPEG-LS, but their encoding times are also worse. PGF seems to be the best compromise between encoding and decoding times.

Our PGF test set clearly shows that PGF in lossless mode is best suited for natural images and aerial ortho photos. PGF is the only algorithm that encodes the three Mega Byte large aerial ortho photo in less than second without a real loss of compression efficiency. For this particular image the efficiency loss is less than three percent compared to the best. These results should be underlined with our second test set, the Kodak test set.

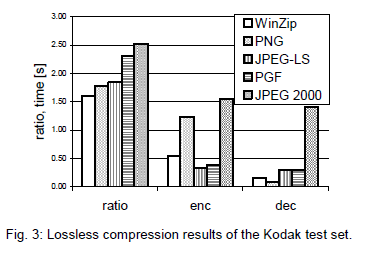


Fig. 3 shows the averages of the compression ratios (ratio), encoding (enc), and decoding (dec) times over all eight images. JPEG 2000 shows in this test set the best compression efficiency followed by PGF, JPEG-LS, PNG, and WinZip. In average PGF is eight percent worse than JPEG 2000. The fact that JPEG 2000 has a better lossless compression ratio than PGF does not surprise,

because JPEG 2000 is more quality driven than PGF.

However, it is remarkable that PGF is clearly better than JPEG-LS (+21%) and PNG (+23%) for natural images. JPEG-LS shows in the Kodak test set also a symmetric encoding and decoding time behaviour. It is encoding and decoding times are almost equal to PGF. Only PNG and WinZip can faster decode than PGF, but they also take longer than PGF to encode.

If both compression efficiency and runtime is important, then PGF is clearly the best of the tested algorithms for lossless compression of natural images and aerial ortho photos. In the third test we perform our lossless coder on the ‘Lena’ image.

To digitally process an image, it is first necessary to reduce the image to a series of numbers that can be manipulated by the computer. Each number representing the brightness value of the image at a particular location is called a picture element, or pixel. A typical digitized image may have 512 × 512 or roughly 250,000 pixels, although much larger images are becoming common. Once the image has been digitized, there are three basic operations that can be performed on it in the computer. For a point operation, a pixel value in the output image depends on a single pixel value in the input image. For local operations, several neighbouring pixels in the input image determine the value of an output image pixel. In a global operation, all of the input image pixels contribute to an output image pixel value.

Correspondingly, these combinations attempt to strike a winning tradeoff: be flexible and hence bring tolerance toward intraclass variation, while also being discriminative enough to be robust to background clutter and interclass similarity. An important feature of our contour-based recognition approach is that it affords us substantial flexibility to incorporate additional image information. Specifically, we extend the contour-based recognition method and propose a new hybrid recognition method which exploits shape tokens and SIFT features as recognition cues. Shape-tokens and SIFT features are largely orthogonal, where the former corresponds to shape boundaries and the latter to sparse salient image patches. Here, each learned combination can comprise features that are either 1) purely shape-tokens, 2) purely SIFT features, or 3) a mixture of shape-tokens and SIFT features. The number and types of features to be combined together are learned automatically from training images, and represent the more discriminative ones based on the training set. Consequently, by imparting these two degrees of variability (in both the number and the types of features) to a combination, we empower it with even greater flexibility and discriminative potential. A shorter version of this paper appeared in [9].

**CLASSIFICATION OF IMAGES:**

There are 3 types of images used in Digital Image Processing. They are

1. Binary Image
2. Gray Scale Image
3. Colour Image

**BINARY IMAGE:**

A binary image is a [digital image](http://en.wikipedia.org/wiki/Digital_image) that has only two possible values for each [pixel](http://en.wikipedia.org/wiki/Pixel).  Typically the two colors used for a binary image are black and white though any two colors can be used.  The color used for the object(s) in the image is the foreground color while the rest of the image is the background color.

Binary images are also called bi-level or two-level. This means that each pixel is stored as a single bit (0 or 1).This name black and white, monochrome or monochromatic are often used for this concept, but may also designate any images that have only one sample per pixel, such as [grayscale images](http://en.wikipedia.org/wiki/Grayscale)

Binary images often arise in [digital image processing](http://en.wikipedia.org/wiki/Digital_image_processing) as [masks](http://en.wikipedia.org/w/index.php?title=Mask_(image_processing)&action=edit&redlink=1) or as the result of certain operations such as [segmentation](http://en.wikipedia.org/wiki/Segmentation_(image_processing)), [thresholding](http://en.wikipedia.org/wiki/Thresholding_(image_processing)), and [dithering](http://en.wikipedia.org/wiki/Dither). Some input/output devices, such as [laser printers](http://en.wikipedia.org/wiki/Laser_printer), [fax machines](http://en.wikipedia.org/wiki/Fax), and bi-level [computer displays](http://en.wikipedia.org/wiki/Visual_display_unit), can only handle bi-level images

**GRAY SCALE IMAGE**

A grayscale Image is [digital image](http://en.wikipedia.org/wiki/Digital_image) is an image in which the value of each [pixel](http://en.wikipedia.org/wiki/Pixel) is a single [sample](http://en.wikipedia.org/wiki/Sample_(signal)), that is, it carries only [intensity](http://en.wikipedia.org/wiki/Luminous_intensity) information. Images of this sort, also known as [black-and-white](http://en.wikipedia.org/wiki/Black-and-white), are composed exclusively of shades of [gray](http://en.wikipedia.org/wiki/Gray)(0-255), varying from black(0) at the weakest intensity to white(255) at the strongest.

Grayscale images are distinct from one-bit [black-and-white](http://en.wikipedia.org/wiki/Black-and-white) images, which in the context of computer imaging are images with only the two [colors](http://en.wikipedia.org/wiki/Color), [black](http://en.wikipedia.org/wiki/Black), and [white](http://en.wikipedia.org/wiki/White) (also called bi-level or [binary images](http://en.wikipedia.org/wiki/Binary_image)). Grayscale images have many shades of gray in between. Grayscale images are also called [monochromatic](http://en.wikipedia.org/wiki/Monochromatic), denoting the absence of any [chromatic](http://en.wikipedia.org/wiki/Chromaticity) variation.

Grayscale images are often the result of measuring the intensity of light at each pixel in a single band of the [electromagnetic spectrum](http://en.wikipedia.org/wiki/Electromagnetic_spectrum) (e.g. [infrared](http://en.wikipedia.org/wiki/Infrared), [visible light](http://en.wikipedia.org/wiki/Visible_spectrum), [ultraviolet](http://en.wikipedia.org/wiki/Ultraviolet), etc.), and in such cases they are monochromatic proper when only a given [frequency](http://en.wikipedia.org/wiki/Frequency) is captured. But also they can be synthesized from a full color image; see the section about converting to grayscale.

**COLOUR IMAGE:**

A (digital) color image is a [digital image](http://en.wikipedia.org/wiki/Digital_image) that includes [color](http://en.wikipedia.org/wiki/Color) information for each [pixel](http://en.wikipedia.org/wiki/Pixel). Each pixel has a particular value which determines its appearing color. This value is qualified by three numbers giving the decomposition of the color in the three primary colors Red, Green and Blue. Any color visible to human eye can be represented this way. The decomposition of a color in the three primary colors is quantified by a number between 0 and 255. For example, white will be coded as R = 255, G = 255, B = 255; black will be known as (R,G,B) = (0,0,0); and say, bright pink will be : (255,0,255).

In other words, an image is an enormous two-dimensional array of color values, pixels, each of them coded on 3 bytes, representing the three primary colors. This allows the image to contain a total of 256x256x256 = 16.8 million different colors. This technique is also known as RGB encoding, and is specifically adapted to human vision

|  |
| --- |
| http://images.gamedev.net/features/programming/imageproc/image004.gif |

**Fig.1 Hue Saturation Process of RGB SCALE Image**

From the above figure, colors are coded on three bytes representing their decomposition on the three primary colors. It sounds obvious to a mathematician to immediately interpret colors as vectors in a three dimension space where each axis stands for one of the primary colors. Therefore we will benefit of most of the geometric mathematical concepts to deal with our colors, such as norms, scalar product, projection, rotation or distance.

EXISITNG METHOD:

**Existing System**

With the upsurge of machine learning, deep learning algorithms have been extensively used in agriculture-related applications. Deep learning can be used for crop mapping, crop image segmentation, crop target detection. Convolutional Neural Networks (CNNs) are used in to extract target regions in the image, object segmentation, and counting number of fruits on a tree using a successive CNN counting algorithm. Dias et al. used CNN in combination with support vector machine (SVM) to extract the features of apple blossoms automatically way to counter complex background, which leads to achieving comparatively accurate apple blossom area segmentation results.

Large data sets training and validation require high performance computing machines such as clusters or servers, which are widely being used in deployment of power extensive deep learning algorithms

**Limitations of Existing System**

* High cost
* More hardware is required
* Not suitable for real-time apple detection
* Computes huge data set

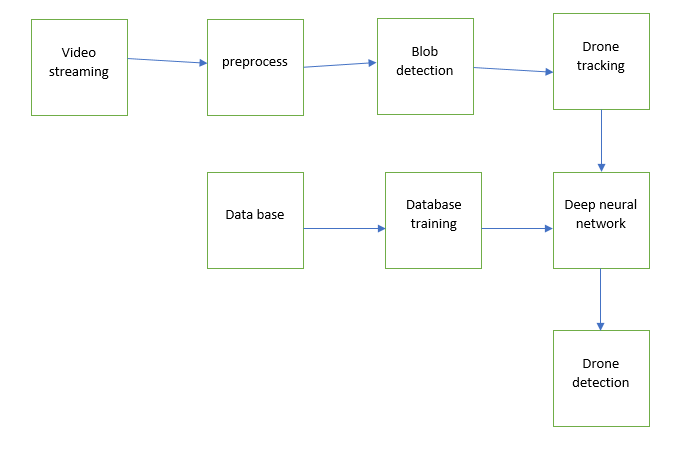
**Proposed System**

The concept of Edge AI consists of performing computations locally on an embedded system in real-time. Since the training process requires a lot more computational power as compared to the inference process, it is not performed on the embedded system, but a dedicated workstation. Then, the model with the obtained weights is deployed on the target hardware in order to be executed. The intrinsic characteristics of the ‘‘tiny’’ version make it suitable for AI applications at the edge with the use of embedded systems, enhanced with hardware accelerators. For this reason, this project has taken the last available ‘‘tiny’’ version of YOLO, YOLOv3-tiny, as a starting point for the realization of an embedded apple detector system.

**Advantage of Proposed model**

* Low cost hardware
* Standalone real-time system
* High accuracy can be obtained
* Training can be done on a local embedded system

BLOCK DIAGRAM:



Video streaming technology is one way to deliver video over the Internet.  Using streaming technologies, the delivery of audio and video over the Internet can reach many millions of customer using their personal computers, PDAs, mobile smartphones or other streaming devices. The reasons for video streaming technology growth are:

* broadband networks are being deployed
* video and audio compression techniques are more efficient
* quality and variety of audio and video services over internet are increasing

There are two major ways for the transmission of video/audio information over the Internet:

*Download mode.* The content file is completely downloaded and then played. This mode requires long downloading time for the whole content file and requires hard disk space.

*Streaming mode.*The content file is not required to be downloaded completely and it is playing while parts of the content are being received and decoded.

**Pre-processing:**

Pre-processing is a common name for operations with images at the lowest level of abstraction -- both input and output are intensity images.ο The aim of pre-processing is an improvement of the image data that suppresses unwanted distortions or enhances some image features important for further processing

**Blob Detection:**

In computer vision, blob detection methods are aimed at detecting regions in a digital image that differ in properties, such as brightness or color, compared to surrounding regions. Informally, a blob is a region of an image in which some properties are constant or approximately constant; all the points in a blob can be considered in some sense to be similar to each other. The most common method for blob detection is convolution.

Given some property of interest expressed as a function of position on the image, there are two main classes of blob detectors: (i) differential methods, which are based on derivatives of the function with respect to position, and (ii) methods based on local extrema, which are based on finding the local maxima and minima of the function. With the more recent terminology used in the field, these detectors can also be referred to as interest point operators, or alternatively interest region operators (see also interest point detectionand corner detection).

There are several motivations for studying and developing blob detectors. One main reason is to provide complementary information about regions, which is not obtained from edge detectors or corner detectors. In early work in the area, blob detection was used to obtain regions of

**Darknet:**

The architecture it works up on is called dark net. a neural network framework created by the first author of the Yolo paper the algorithm works off  by dividing an image into a grid of cells, for each cell bounding boxes and their confidence scores are predicted, alongside class probabilities. That means by using these property we can divide all things what we are seeing in object  and mansion at the outside. This one is the best for the classification of the image.it gives more clarity. we can detect more number of things in the video or images also. this will  be working based on web camera for real time applications.by using these detects maximum number of features on face. this will obtained by the classification of each pixel.

it is written in c++, pyhon. Here using caffe model datasety for this process.it has taken by the four stps on it.they are

Data Preparation:in this step ,clean the image abnd store them in a format that can be used by caffe.we will wrte write python script that will handle both pre-processing and storage.

Model Defination:in this,we choose CNN architecture and define parameters in a configuration file with extension.

**Deep Learning Classification:**

deep learning is a computer software that mimics the network of neurons in a brain. It is a subset of machine learning and is called deep learning because it makes use of deep neural networks.

Deep learning algorithms are constructed with connected layers.

The first layer is called the Input Layer

The last layer is called the Output Layer

All layers in between are called Hidden Layers. The word deep means the network join neurons in more than two layers.

Each Hidden layer is composed of neurons. The neurons are connected to each other. The neuron will process and then propagate the input signal it receives the layer above it. The strength of the signal given the neuron in the next layer depends on the weight, bias and activation function.

The network consumes large amounts of input data and operates them through multiple layers; the network can learn increasingly complex features of the data at each layer.

  Classification of Neural Networks Shallow neural network: The Shallow neural network has  only one hidden layer between the  input and output.Deep neural network: Deep neural networks have more than one layer. For instance, Google LeNet model for image recognition counts 22 layers.

Nowadays, deep learning is used in many ways like a driverless car, mobile phone, Google Search Engine, Fraud detection, TV, and so on.

Feed-forward neural networks

The simplest type of artificial neural network. With this type of architecture, information flows in only one direction, forward. It means, the information's flows starts at the input layer, goes to the "hidden" layers, and end at the output layer. The network does not have a loop. Information stops at the output layers.

Recurrent neural networks (RNNs)

RNN is a multi-layered neural network that can store information in context nodes, allowing it to learn data sequences and output a number or another sequence. In simple words it an Artificial neural networks whose connections between neurons include loops. RNNs are well suited for processing sequences of inputs.Example, if the task is to predict the next word in the sentence "Do you want a…………?The RNN neurons will receive a signal that point to the start of the sentence.The network receives the word "Do" as an input and produces a vector of the number. This vector is fed back to the neuron to provide a memory to the network. This stage helps the network to remember it received "Do" and it received it in the first position.

The network will similarly proceed to the next words. It takes the word "you" and "want." The state of the neurons is updated upon receiving each word.

The final stage occurs after receiving the word "a." The neural network will provide a probability for each English word that can be used to complete the sentence. A well-trained RNN probably assigns a high probability to "café," "drink," "burger," etc.

Common uses of RNN

Help securities traders to generate analytic reports Detect abnormalities in the contract of financial statement Detect fraudulent credit-card transaction, Provide a caption for images, Power chatbots

The standard uses of RNN occur when the practitioners are working with time-series data or sequences (e.g., audio recordings or text).

Convolutional neural networks (CNN)

CNN is a multi-layered neural network with a unique architecture designed to extract increasingly complex features of the data at each layer to determine the output. CNN's are well suited for perceptual tasks.CNN is mostly used when there is an unstructured data set (e.g., images) and the practitioners need to extract information from it

For instance, if the task is to predict an image caption:The CNN receives an image of let's say a cat, this image, in computer term, is a collection of the pixel. Generally, one layer for the greyscale picture and three layers for a color picture.

During the feature learning (i.e., hidden layers), the network will identify unique features, for instance, the tail of the cat, the ear, etc.

When the network thoroughly learned how to recognize a picture, it can provide a probability for each image it knows. The label with the highest probability will become the prediction of the network.

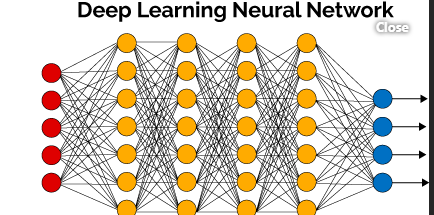
**Deep learning:**

INTRODUCTION:-

Deep learning represents the very cutting edge of [artificial intelligence (AI)](https://bernardmarr.com/default.asp?contentID=963). Instead of teaching computers to process and learn from data (which is how machine learning works), with deep learning, the computer trains itself to process and learn from data.

This is all possible thanks to layers of ANNs. Remember that I said an ANN in its simplest form has only three layers? Well an ANN that is made up of more than three layers – i.e. an input layer, an output layer and multiple hidden layers – is called a ‘deep neural network’, and this is what underpins deep learning. A deep learning system is self-teaching, learning as it goes by filtering information through multiple hidden layers, in a similar way to humans.

As you can see, the two are closely connected in that one relies on the other to function. Without neural networks, there would be no deep learning.





Algorithm used:

RCNN:-

Region convolutional neural network. It is two stage of cnn. using this we can detect object easily with grid lines for the classification.

SOFTWARE PART Python is an object- acquainted, high position language, interpreted, dynamic and multipurpose programming language. Python is easy to learn yet important and protean scripting language which makes it seductive for Application Development. Python's syntax and dynamic codifying with its interpreted nature, make it an ideal language for scripting and rapid-fire operation development in numerous areas. Python supports multiple programming pattern, including object acquainted programming, imperative and functional programming or procedural styles. Python isn't intended to work on special area similar as web programming. That's why it's known as multipurpose because it can be used with web, enterprise, 3D CAD etc. We do not need to use data types to declare variable because it's stoutly compartmented so we can write a = 10 to declare an integer value in a variable. Python makes the development and debugging presto because there's no compendium step included in python development and edit- test-debug cycle is veritably presto. System Conditions Software Python Anaconda tar Hardware Windows (64 bit) RAM 3 GB Jeer pi 3 B Jeer Pi Jeer Pi is a small single- board Computer developed in UK by Raspberry Pi foundation to promote the tutoring of computer wisdom in seminaries and in developing countries. Original model come far more popular than anticipated sealing outside of its target request, for uses similar as robots. Processor The processor at the heart of the Jeer Pi is a Broadcom BCM28XX. This is the Broadcom System on Chip (SOC) chip use in the Jeer Pi. The processor from first to third generations include Raspberry Pi 1 Broadcom BCM2835 SOC with 700 MHz CPU speed, L2 cache of 128kb with ARM comity AR1176JZF-S (ARMv6) 32- bit RISC ARM. Jeer Pi 2 Broadcom BCM 2836 SOC with 900 MHz CPU speed, L2 cache of 256kb with 32- bit quadrangle- core ARM cortex-A7 (ARMv7). Jeer Pi 3 Broadcom BCM2837 SOC with1.2 GHz 64- bit quadrangle- core – A53 with 512 kb participated L2 cache (64- bit instruction set ARMv8). Raspberry Pi 3 PYTHON FEATURES 1) Easy to Use 2) Suggestive Language 3) Interpreted Language 4)Cross-platform language 5) Free and Open Source 6) Object- Acquainted language 7) Extensible 8) Large Standard Library 9) GUI Programming 10) Integrated.

After giving to algorithm with the data base by input it will give output when it will detect the drones.

**Modules used:**

1.Opencv:-

* Used for object based applications.

2.Numpy:-

* It  is used to algebraic, numerical, fourier and matrices etc…
* It is python library.

**Introduction to Computer Vision**

Using software to parse the world’s visual content is as big of a revolution in computing as mobile was 10 years ago, and will provide a major edge for developers and businesses to build amazing products.

Computer Vision is the process of using machines to understand and analyze imagery (both photos and videos). While these types of algorithms have been around in various forms since the 1960’s, recent advances in [Machine Learning](https://blog.algorithmia.com/introduction-to-machine-learning/), as well as leaps forward in data storage, computing capabilities, and cheap high-quality input devices, have driven major improvements in how well our software can explore this kind of content.

**What is Computer Vision?**

Computer Vision is the broad parent name for any computations involving visual content – that means images, videos, icons, and anything else with pixels involved. But within this parent idea, there are a few specific tasks that are core building blocks:

* In **object classification**, you train a model on a dataset of specific objects, and the model classifies new objects as belonging to one or more of your training categories.
* For **object identification**, your model will recognize a specific instance of an object – for example, parsing two faces in an image and tagging one as Tom Cruise and one as Katie Holmes.

A classical application of computer vision is handwriting recognition for digitizing handwritten content (we’ll explore more use cases below). Outside of just recognition, other methods of analysis include:

* Video **motion analysis** uses computer vision to estimate the velocity of objects in a video, or the camera itself.
* In **image segmentation**, algorithms partition images into multiple sets of views.
* **Scene reconstruction** creates a 3D model of a scene inputted through images or video (check out [Selva](https://www.selva3d.com/)).
* In **image restoration**, noise such as blurring is removed from photos using Machine Learning based filters.

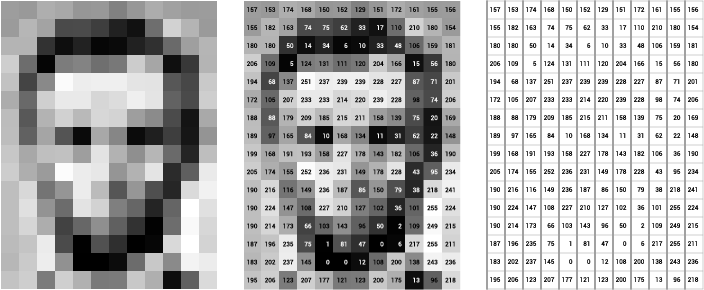
Any other application that involves understanding pixels through software can safely be labeled as computer vision.

**How Computer Vision Works**

One of the major open questions in both Neuroscience and Machine Learning is: how exactly do our brains work, and how can we approximate that with our own algorithms? The reality is that there are very few working and comprehensive theories of brain computation; so despite the fact that Neural Nets are supposed to “mimic the way the brain works,” nobody is quite sure if that’s actually true. Jeff Hawkins has an [entire book on this topic called On Intelligence](https://www.amazon.com/Intelligence-Understanding-Creation-Intelligent-Machines/dp/0805078533).

The same paradox holds true for computer vision – since we’re not decided on how the brain and eyes process images, it’s difficult to say how well the algorithms used in production approximate our own internal mental processes. For example, [studies have shown](https://www.technologyreview.com/s/508376/in-a-frogs-eye/) that some functions that we thought happen in the brain of *frogs* actually take place in the eyes. We’re a far cry from amphibians, but similar uncertainty exists in human cognition.

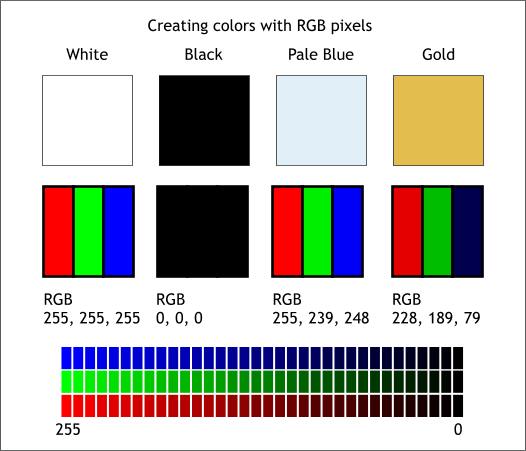
Machines interpret images very simply: as a series of pixels, each with their own set of color values. Consider the simplified image below, and how grayscale values are converted into a simple array of numbers:



Source: [Openframeworks](http://openframeworks.cc/ofBook/chapters/image_processing_computer_vision.html)

Think of an image as a giant grid of different squares, or pixels (this image is a very simplified version of what looks like either Abraham Lincoln or a Dementor). Each pixel in an image can be represented by a number, usually from 0 – 255. The series of numbers on the right is what software sees when you input an image. For our image, there are 12 columns and 16 rows, which means there are 192 input values for this image.

When we start to add in color, things get more complicated. Computers usually read color as a series of 3 values – red, green, and blue (RGB) – on that same 0 – 255 scale. Now, each pixel actually has 3 values for the computer to store in addition to its position. If we were to colorize President Lincoln (or Harry Potter’s worst fear), that would lead to 12 x 16 x 3 values, or 576 numbers.



Source: [Xaraxone](http://archive.xaraxone.com/webxealot/workbook35/page_5.htm)

For some perspective on how computationally expensive this is, consider this tree:

* Each color value is stored in 8 bits.
* 8 bits x 3 colors per pixel = 24 bits per pixel.
* A normal sized 1024 x 768 image x 24 bits per pixel = almost 19M bits, or about 2.36 megabytes.

That’s a lot of memory to require for one image, and a lot of pixels for an algorithm to iterate over. But to train a model with meaningful accuracy – especially when you’re talking about [Deep Learning](https://blog.algorithmia.com/introduction-to-deep-learning/) – you’d usually need tens of thousands of images, and the more the merrier. Even if you were to use [Transfer Learning](https://en.wikipedia.org/wiki/Transfer_learning) to use the insights of an already trained model, you’d still need a few thousand images to train yours on.

With the sheer amount of computing power and storage required just to train deep learning models for computer vision, it’s not hard to understand why advances in those two fields have driven Machine Learning forward to such a degree.

**Business Use Cases for Computer Vision**

Computer vision is one of the areas in Machine Learning where core concepts are already being integrated into major products that we use every day. [Google is using maps](https://research.googleblog.com/2017/05/updating-google-maps-with-deep-learning.html) to leverage their image data and identify street names, businesses, and office buildings. Facebook is using computer vision to identify people in photos, and do a number of things with that information.

But it’s not just tech companies that are leverage Machine Learning for image applications. Ford, the American car manufacturer that has been around [literally since the early 1900’s](https://en.wikipedia.org/wiki/Ford_Motor_Company), is [investing heavily in autonomous vehicles (AVs)](https://media.ford.com/content/fordmedia/fna/us/en/news/2016/08/16/ford-targets-fully-autonomous-vehicle-for-ride-sharing-in-2021.html). Much of the underlying technology in AVs relies on analyzing the multiple video feeds coming into the car and using computer vision to analyze and pick a path of action.

Another major area where computer vision can help is in the medical field. Much of diagnosis is image processing, like reading x-rays, MRI scans, and other types of diagnostics. [Google has been working with medical research teams](https://research.google.com/teams/brain/healthcare/) to explore how deep learning can help medical workflows, and have made significant progress in terms of accuracy. To paraphrase from their research page:

“*Collaborating closely with doctors and international healthcare systems, we developed a state-of-the-art computer vision system for reading retinal fundus images for diabetic retinopathy and determined our algorithm’s performance is on par with U.S. board-certified ophthalmologists. We’ve recently published some of our research in the* [*Journal of the American Medical Association*](https://research.google.com/pubs/archive/45732.pdf) *and summarized the highlights in a* [*blog post*](https://research.googleblog.com/2016/11/deep-learning-for-detection-of-diabetic.html)*.*”

But aside from the groundbreaking stuff, it’s getting much easier to integrate computer vision into your own applications. A number of high-quality third party providers like Clarifai offer [a simple API for tagging and understanding images](https://www.clarifai.com/), while Kairos [provides functionality around facial recognition](https://www.kairos.com/). We’ll dive into the open-source packages available for use below.

**Computer Vision on Algorithmia**

Algorithmia makes it easy to deploy computer vision applications as scalable microservices. Our marketplace has a few algorithms to help get the job done:

* [SalNet](https://algorithmia.com/algorithms/deeplearning/SalNet) automatically identifies the most important parts of an image
* [Nudity Detection](https://algorithmia.com/algorithms/sfw/NudityDetectioni2v) detects nudity in pictures
* [Emotion Recognition](https://algorithmia.com/algorithms/deeplearning/EmotionRecognitionCNNMBP) parses emotions exhibited in images
* [DeepStyle](https://demos.algorithmia.com/deep-style/) transfers next-level filters onto your image
* [Face Recognition](https://algorithmia.com/algorithms/cv/FaceRecognition)…recognizes faces.
* [Image Memorability](https://algorithmia.com/algorithms/deeplearning/LargescaleImageMemorability) judges how memorable an image is.

A typical workflow for your product might involve passing images from a security camera into Emotion Recognition and raising a flag if any aggressive emotions are exhibited, or using Nudity Detection to block inappropriate profile pictures on your web application.

For a more detailed exploration of how you can use the Algorithmia platform to implement complex and useful computer vision tasks,

### Computer Vision Resources

##### **Packages and Frameworks**

[OpenCV](https://opencv.org/) – “OpenCV was designed for computational efficiency and with a strong focus on real-time applications. Adopted all around the world, OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 14 million. Usage ranges from interactive art, to mines inspection, stitching maps on the web or through advanced robotics.”

[SimpleCV](http://simplecv.org/) – “SimpleCV is an open source framework for building computer vision applications. With it, you get access to several high-powered computer vision libraries such as OpenCV – without having to first learn about bit depths, file formats, color spaces, buffer management, eigenvalues, or matrix versus bitmap storage.”

[Mahotas](http://mahotas.readthedocs.io/en/latest/) – “Mahotas is a computer vision and image processing library for Python. It includes many algorithms implemented in C++ for speed while operating in numpy arrays and with a very clean Python interface. Mahotas currently has over 100 functions for image processing and computer vision and it keeps growing.”

Numpy package:

NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed. This tutorial explains the basics of NumPy such as its architecture and environment. It also discusses the various array functions, types of indexing, etc. An introduction to Matplotlib is also provided. All this is explained with the help of examples for better understanding.

Audience

This tutorial has been prepared for those who want to learn about the basics and various functions of NumPy. It is specifically useful for algorithm developers. After completing this tutorial, you will find yourself at a moderate level of expertise from where you can take yourself to higher levels of expertise.

Prerequisites

You should have a basic understanding of computer programming terminologies. A basic understanding of Python and any of the programming languages is a plus.

NumPy is a Python package. It stands for 'Numerical Python'. It is a library consisting of multidimensional array objects and a collection of routines for processing of array.

**Numeric**, the ancestor of NumPy, was developed by Jim Hugunin. Another package Numarray was also developed, having some additional functionalities. In 2005, Travis Oliphant created NumPy package by incorporating the features of Numarray into Numeric package. There are many contributors to this open source project.

## Operations using NumPy

Using NumPy, a developer can perform the following operations −

* Mathematical and logical operations on arrays.
* Fourier transforms and routines for shape manipulation.
* Operations related to linear algebra. NumPy has in-built functions for linear algebra and random number generation.

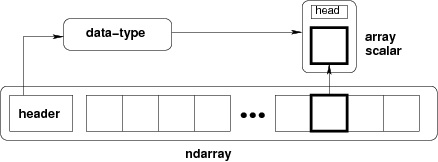
## NumPy – A Replacement for MatLab

NumPy is often used along with packages like **SciPy** (Scientific Python) and **Mat−plotlib** (plotting library). This combination is widely used as a replacement for MatLab, a popular platform for technical computing. However, Python alternative to MatLab is now seen as a more modern and complete programming language.

The most important object defined in NumPy is an N-dimensional array type called **ndarray**. It describes the collection of items of the same type. Items in the collection can be accessed using a zero-based index.

Every item in an ndarray takes the same size of block in the memory. Each element in ndarray is an object of data-type object (called **dtype**).

Any item extracted from ndarray object (by slicing) is represented by a Python object of one of array scalar types. The following diagram shows a relationship between ndarray, data type object (dtype) and array scalar type −



An instance of ndarray class can be constructed by different array creation routines described later in the tutorial. The basic ndarray is created using an array function in NumPy as follows −

numpy.array

It creates an ndarray from any object exposing array interface, or from any method that returns an array.

Software used:

1.What is Python

**Python** is an object-oriented, high level language, interpreted, dynamic and multipurpose programming language.

Python is *easy to learn* yet powerful and versatile scripting language which makes it attractive for Application Development.

Python's syntax and *dynamic typing* with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas.

Python supports *multiple programming pattern*, including object oriented programming, imperative and functional programming or procedural styles.

Python is not intended to work on special area such as web programming. That is why it is known as *multipurpose* because it can be used with web, enterprise, 3D CAD etc.

We don't need to use data types to declare variable because it is *dynamically typed* so we can write a=10 to declare an integer value in a variable.

Python makes the development and debugging *fast* because there is no compilation step included in python development and edit-test-debug cycle is very fast.

2. Python Features

#### 1) Easy to Use:

Python is easy to very easy to use and high level language. Thus it is programmer-friendly language.

#### 2) Expressive Language:

Python language is more expressive. The sense of expressive is the code is easily understandable.

#### 3) Interpreted Language:

Python is an interpreted language i.e. interpreter executes the code line by line at a time. This makes debugging easy and thus suitable for beginners.

#### 4) Cross-platform language:

Python can run equally on different platforms such as Windows, Linux, Unix , Macintosh etc. Thus, Python is a portable language.

#### 5) Free and Open Source:

Python language is freely available(www.python.org).The source-code is also available. Therefore it is open source.

#### 6) Object-Oriented language:

Python supports object oriented language. Concept of classes and objects comes into existence.

#### 7) Extensible:

It implies that other languages such as C/C++ can be used to compile the code and thus it can be used further in your python code.

#### 8) Large Standard Library:

Python has a large and broad library.

#### 9) GUI Programming:

Graphical user interfaces can be developed using Python.

#### 10) Integrated:

It can be easily integrated with languages like C, C++, JAVA etc.

3. Python History

* Python laid its foundation in the late 1980s.
* The implementation of Python was started in the December 1989 by **Guido Van Rossum** at CWI in Netherland.
* *ABC programming language* is said to be the predecessor of Python language which was capable of Exception Handling and interfacing with Amoeba Operating System.
* Python is influenced by programming languages like:
  + ABC language.
  + Modula-3

# 4. Python Version

Python programming language is being updated regularly with new features and support. There are a lot of updation in python versions, started from 1994 to current date.

A list of python versions with its released date is given below.

|  |  |
| --- | --- |
| **Python Version** | **Released Date** |
| Python 1.0 | January 1994 |
| Python 1.5 | December 31, 1997 |
| Python 1.6 | September 5, 2000 |
| Python 2.0 | October 16, 2000 |
| Python 2.1 | April 17, 2001 |
| Python 2.2 | December 21, 2001 |
| Python 2.3 | July 29, 2003 |
| Python 2.4 | November 30, 2004 |
| Python 2.5 | September 19, 2006 |
| Python 2.6 | October 1, 2008 |
| Python 2.7 | July 3, 2010 |
| Python 3.0 | December 3, 2008 |
| Python 3.1 | June 27, 2009 |
| Python 3.2 | February 20, 2011 |
| Python 3.3 | September 29, 2012 |

# 5. Python Applications

Python as a whole can be used in any sphere of development.

Let us see what are the major regions where Python proves to be handy.

#### 1) Console Based Application

Python can be used to develop console based applications. For example: **IPython**.

#### 2) Audio or Video based Applications

Python proves handy in multimedia section. Some of real applications are: TimPlayer, cplay etc.

#### 3) 3D CAD Applications

Fandango is a real application which provides full features of CAD.

#### 4) Web Applications

Python can also be used to develop web based application. Some important developments are: PythonWikiEngines, Pocoo, PythonBlogSoftware etc.

#### 5) Enterprise Applications

Python can be used to create applications which can be used within an Enterprise or an Organization. Some real time applications are: OpenErp, Tryton, Picalo etc.

#### 6) Applications for Images

Using Python several application can be developed for image. Applications developed are: VPython, Gogh, imgSeek etc.

There are several such applications which can be developed using Python

# 6. Python Example

Python code is simple and easy to run. Here is a simple Python code that will print "Welcome to Python".

A simple python example is given below.

1. >>> a="Welcome To Python"
2. >>> print a
3. Welcome To Python
4. >>>

**Explanation:**

* Here we are using IDLE to write the Python code. Detail explanation to run code is given in Execute Python section.
* A variable is defined named "a" which holds "Welcome To Python".
* "print" statement is used to print the content. Therefore "print a" statement will print the content of the variable. Therefore, the output "Welcome To Python" is produced.

## Python 3.4 Example

In python 3.4 version, you need to add parenthesis () in a string code to print it.

1. >>> a=("Welcome To Python Example")
2. >>> print a
3. Welcome To Python Example
4. >>>

# 7. How to execute python

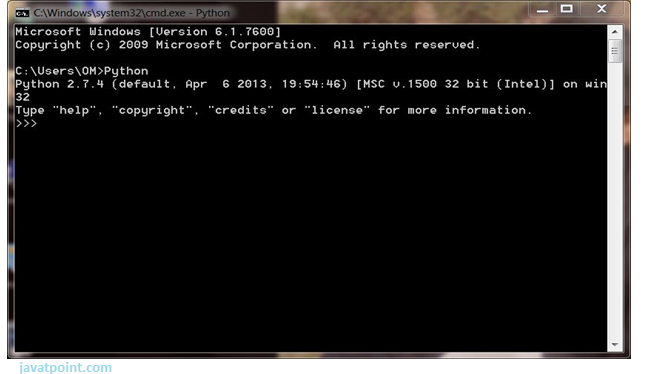
There are three different ways of working in Python:

## 1) Interactive Mode:

You can enter python in the command prompt and start working with Python.

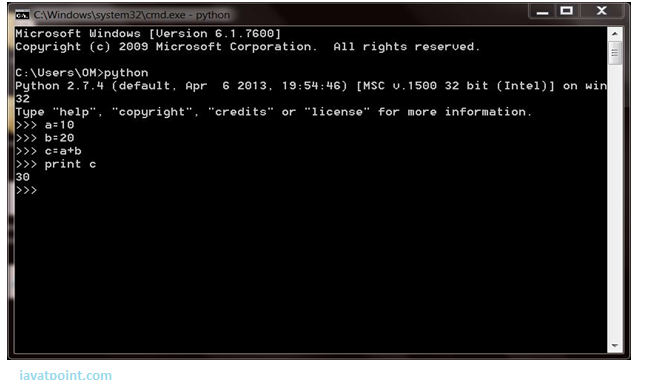


Press Enter key and the Command Prompt will appear like:



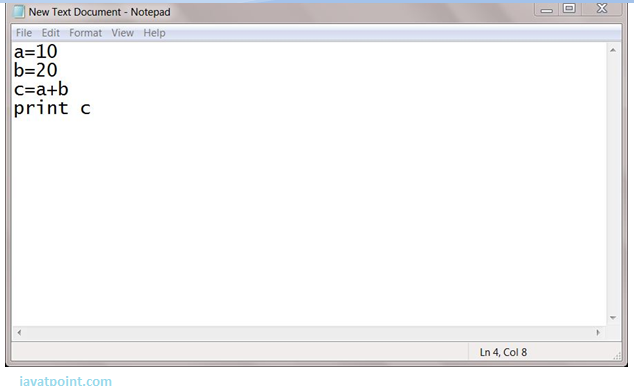
Now you can execute your Python commands.

**Eg:**

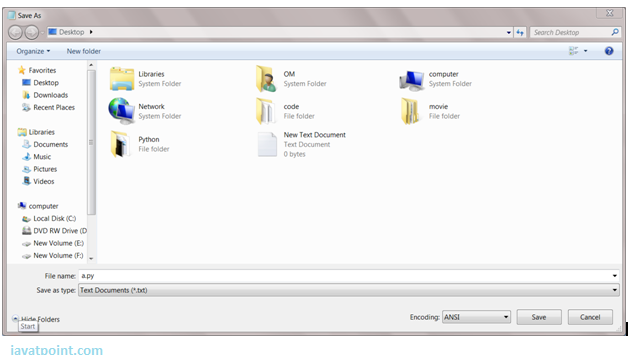


## 2) Script Mode:

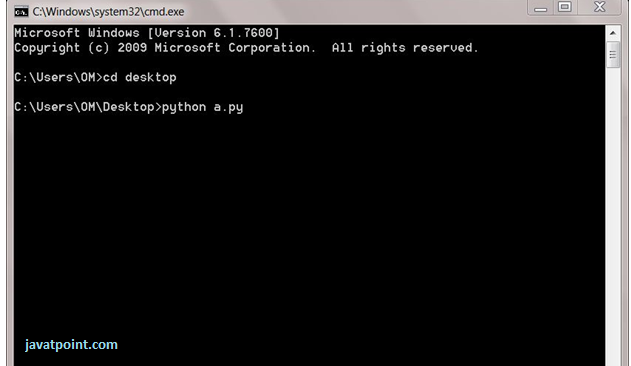
Using Script Mode , you can write your Python code in a separate file using any editor of your Operating System.



Save it by .py extension.



Now open Command prompt and execute it by :



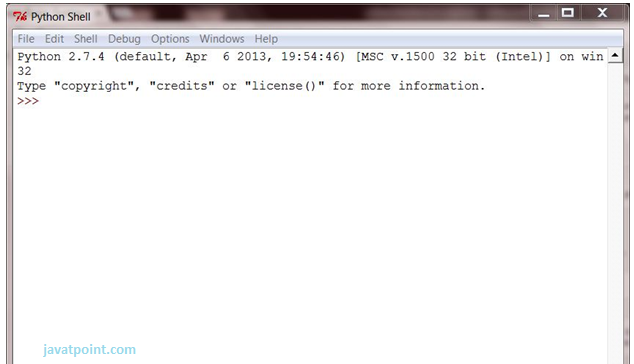
NOTE: Path in the command prompt should be where you have saved your file. In the above case file should be saved at desktop.

## 3) Using IDE: (Integrated Development Environment)

You can execute your Python code using a Graphical User Interface (GUI).

All you need to do is:

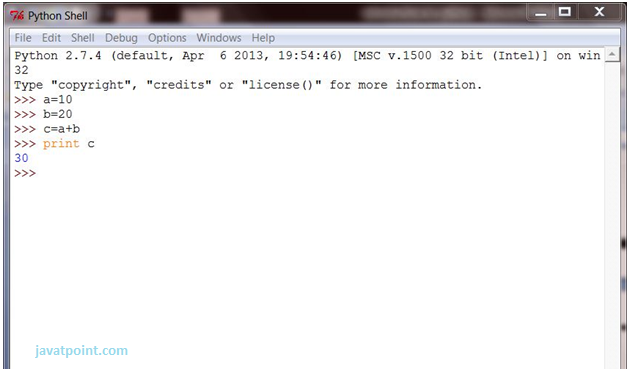
Click on Start button -> All Programs -> Python -> IDLE(Python GUI)



You can use both Interactive as well as Script mode in IDE.

**1) Using Interactive mode:**

Execute your Python code on the Python prompt and it will display result simultaneously.

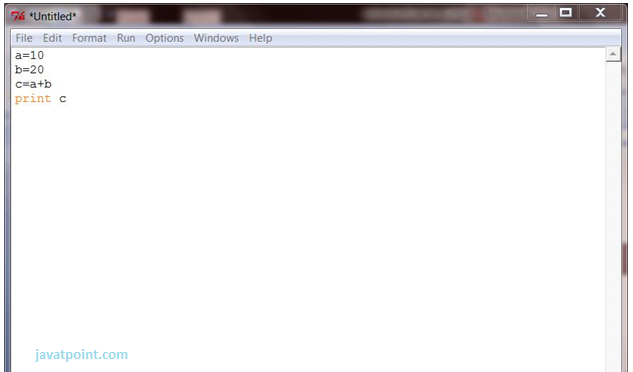


**2) Using Script Mode:**

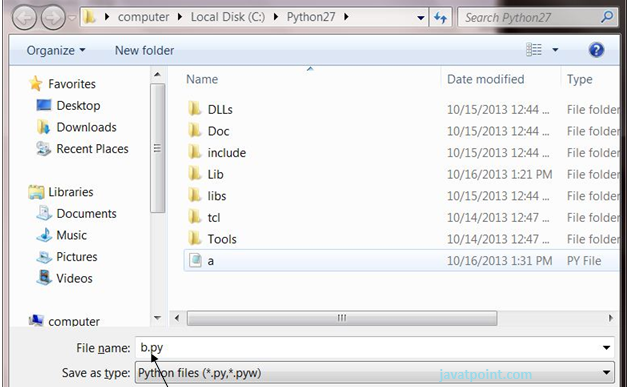
i) Click on Start button -> All Programs -> Python -> IDLE(Python GUI)

ii) Python Shell will be opened. Now click on File -> New Window.

A new Editor will be opened . Write your Python code here.



Click on file -> save as



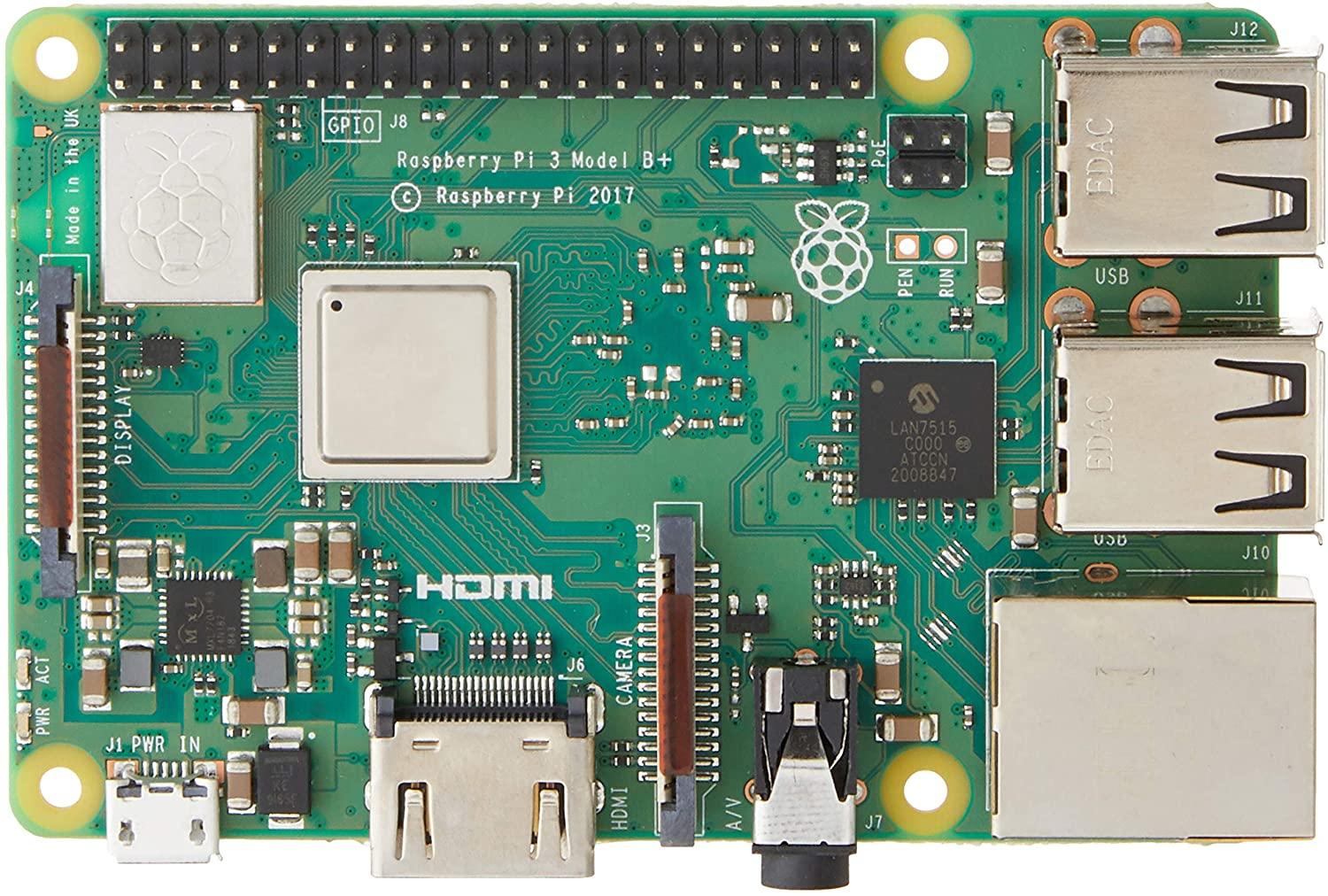
Run then code by clicking on Run in the Menu bar.

Run -> Run Module

Result will be displayed on a new Python

Hardware used:

**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 acountries.

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

**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).

**Processor**

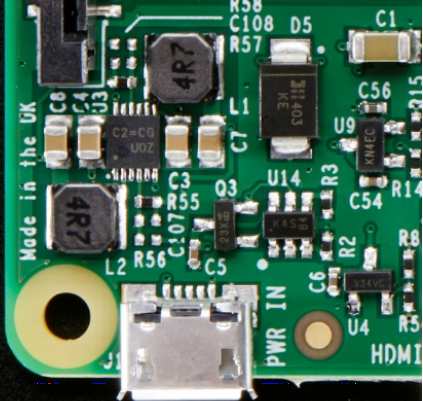
The processor at the heart of the Raspberry Pi is a Broadcom BCM28XX.

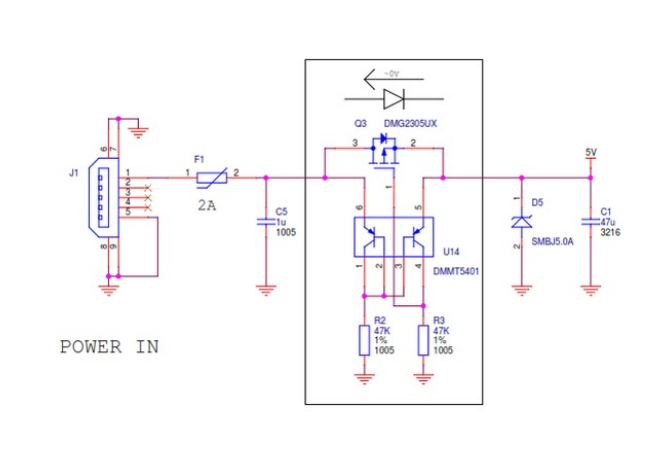
This is the Broadcom System on Chip (SOC) chip use in the Raspberry Pi. The processor from first to third generations include:

* 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).

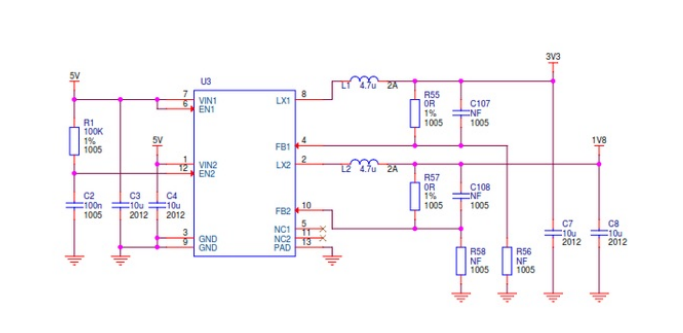
**RASPBERRY PI POWER SUPPLY**

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



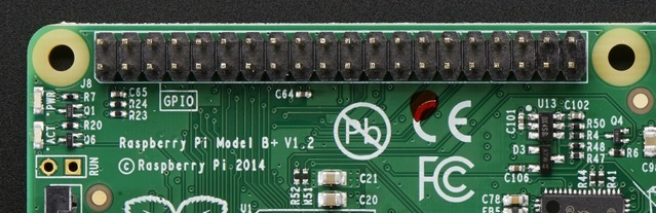


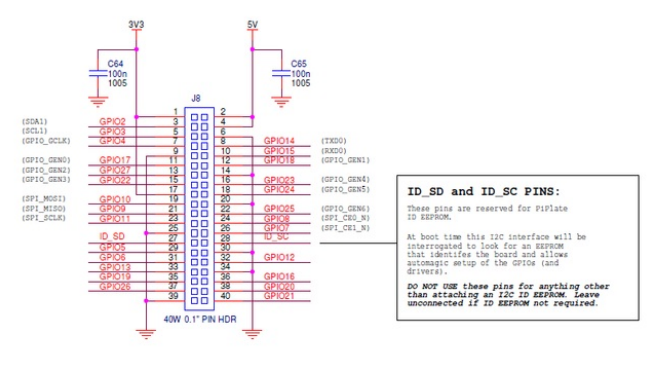
There's still the microUSB jack on the left, and the 1A fuse has been upgraded to a 2A fuse. There's also a DMG2305UX (http://adafru.it/dGU) P-Channel MOSFET. This acts as a polarity protection switch but is much lower 'drop-out' than a diode. It has only 52mW resistance so @ 2A its about 0.1V voltage 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:

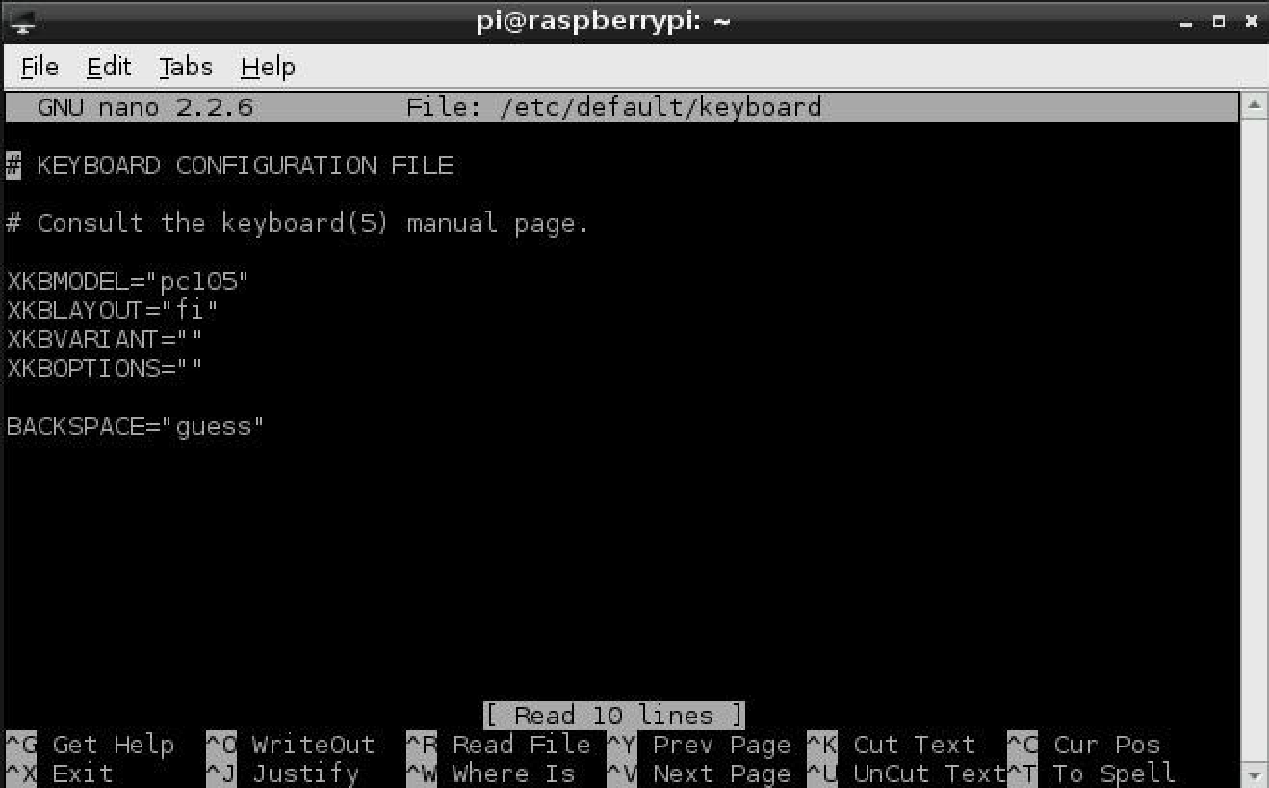
Startx

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**

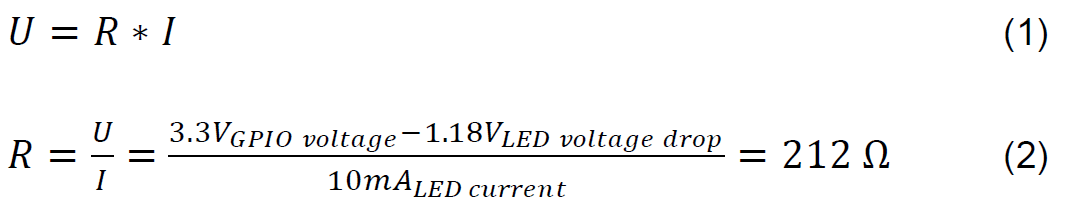
This chapter discovers the GPIO connector and how it can be used in controlling. The first experiments with the GPIO were to light up a LED (light-emitting diode) through the Python Shell.

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:

****

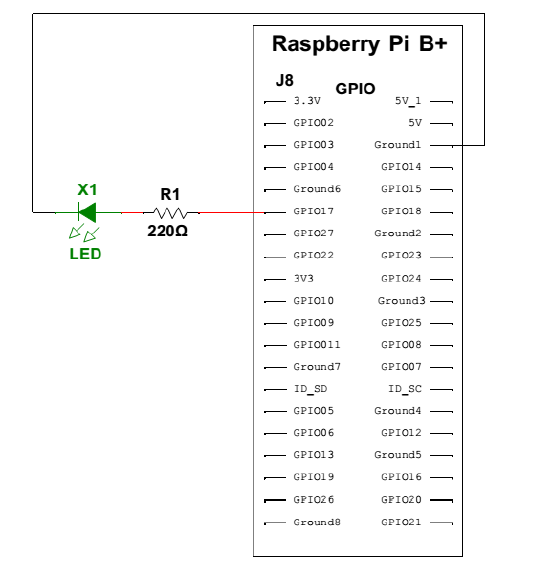
Where

 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)

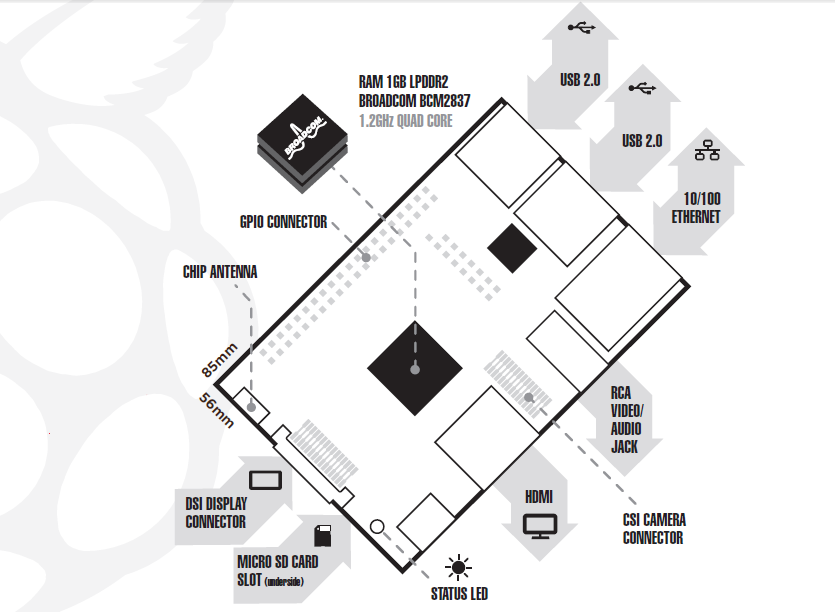
camera.stop\_preview()

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, where X 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 card connected to 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! 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 connected to 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 A doesn’t feature any onboard networking. Thankfully, that doesn’t mean you can’t network the Model A; only that you’ll need some 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 two directly 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 crossover cable, the receive and transmit pairs are swapped so that the two devices are prevented from talking over each other—a task usually 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 known as auto-MDI, which allows it to reconfigure itself automatically. As a result, you can use any RJ45 cable—crossover or not—to connect the Pi to the network, and it will adjust its configuration accordingly. The Raspberry Pi Model B’s Ethernet port If 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 is outside the scope of this book, but if you are completely unable to connect the Pi to the Internet in any other way, you can try searching 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 operating system through the Dynamic Host Configuration Protocol (DHCP). This assigns the Pi an Internet Protocol (IP) address on your network, 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 Pi will 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 Ethernet to 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 Pi Using such a device, the Pi can connect to a wide range of wireless networks, including those running on the latest 802.11n highspeed standard. 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 Windows and OS X only, making them incompatible with the Raspberry Pi. A list of Wi-Fi adapters known to work with the Raspberry Pi.

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

• Check the frequencies supported by the card. Some wireless network standards, like 802.11a, support more than one frequency. 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 of encryption, but if you’re buying a second-hand or older model, you may find it won’t connect to your network. Common encryption 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**

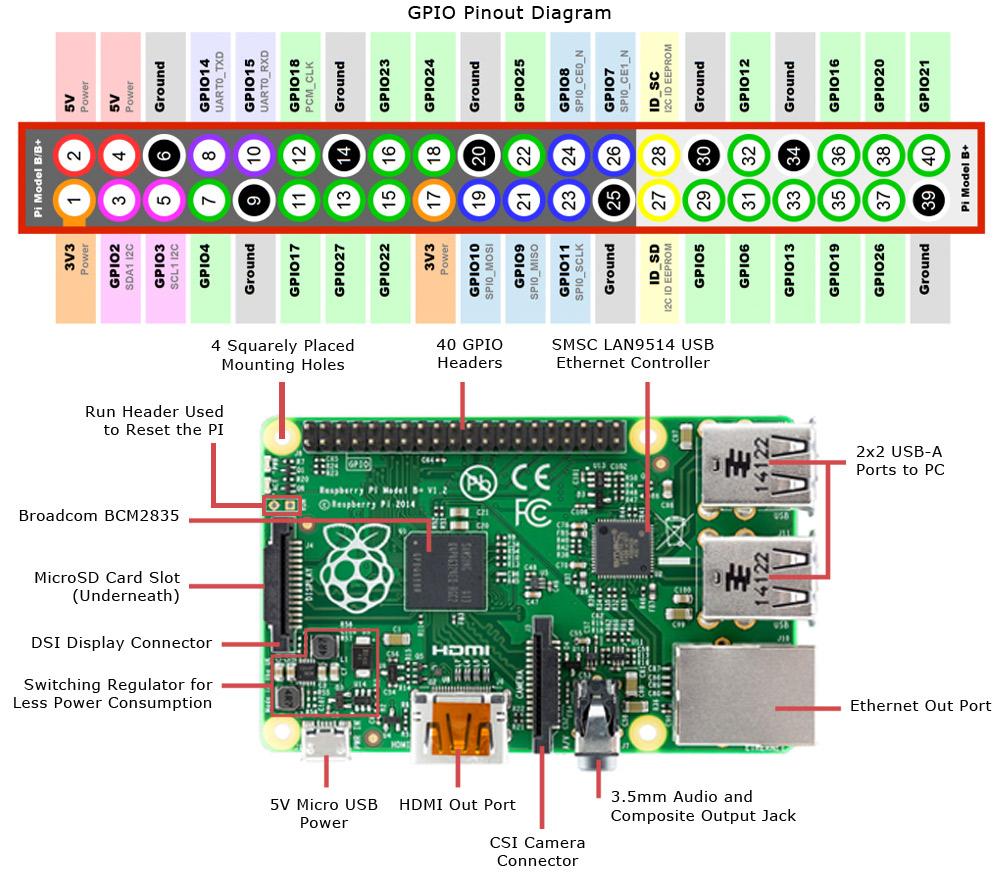
The Raspberry Pi is powered by the small micro-USB connector found on the lower left side of the circuit board. This connector is the same as found on the majority of smartphones and some tablet devices. Many chargers designed for smartphones will work with the Raspberry Pi, but not all. The Pi is more power-hungry than most micro-USB devices, and requires up to 700mA in order to operate. Some chargers can only supply up to 500mA, causing intermittent 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 smaller chargers, 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 will start 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 with two rows of 13 male 2.54 mm headers at the factory. The spacing of these headers is particularly important: 2.54 mm pin spacing (0.1 inches in imperial measurements) is a very common sight in electronics, and is the standard spacing for prototyping platforms 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 definitions Pin 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 for numbering pins, and because there are no markings on the Pi itself, it’s easy to get confused as to which pin is which.

****

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 components compatible 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 any other 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 to repair 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 Pin 22. Additionally, Pin 7—while defaulting to providing a clock signal for general purpose use—can also be used as a generalpurpose pin, 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 used to 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 used between the two. Likewise, connecting pins on a 5 V microcontroller directly to the Raspberry Pi’s GPIO port will not work and may permanently damage the Pi.**

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

**UART Serial Bus**

The Universal Asynchronous Receiver/Transmitter (UART) serial bus provides a simple two-wire serial interface. When a serial port is configured in the cmdline.txt file (as described in Chapter 6, “Configuring the Raspberry Pi”), it’s this serial bus that is used as the port for the messages. Connecting the Pi’s UART serial bus to a device capable of displaying the data will reveal messages from the Linux kernel. If you’re having trouble getting the Pi to boot, this can be a handy diagnostic tool—especially if nothing 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 receive signal. 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 integrated circuits (ICs). In the case of the Pi, one of those integrated circuits is the Broadcom BCM2837 SoC processor at the heart of the system. These pins include access to pull-up resistors located on the Pi, meaning no external resistors are required to access the 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 the Serial 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 for general-purpose use.

**SPI Bus**

The Serial Peripheral Interface (SPI) bus is a synchronous serial bus designed primarily for in-system programming (ISP) of microcontrollers and other devices. Unlike the UART and I²C buses, it’s a four-wire bus with multiple Chip Select lines which allow 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 the SPI Master Output, Slave Input (MOSI) signal; Pin 21 provides the SPI Master Input, Slave Output (MISO) signal; Pin 23 provides the Serial Clock (SLCK) used to synchronise communication; and Pins 24 and 26 provide the Chip Select signals for up 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 GPIO port 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 access to the general-purpose pins on the Raspberry Pi’s GPIO port in Python. You’ll also be shown two simple electronic circuits which 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, the perfect 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 the Pi at the heart of a nuclear reactor or a complex robotics platform, however, you may want to investigate a lower-level language such 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-time environment 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 its various functions. In particular, programmers have addressed the Pi users’ need to access the GPIO port without having to know low-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 it will work. There are several GPIO Python libraries available, but for the purpose of this section, we recommend that you use the raspberry-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 the terminal 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 a desktop environment.

**2.** Type wget http://raspberry-gpio-python.googlecode.com/files/RPi.GPIO-0.2.0.tar.gz to download the library to your home directory. If a newer version has been released, replace the version number—0.2.0—with the current 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 make sure 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 the library, 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 program m. 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 that your 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 Values 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 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 and needing 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 LED will 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 being damaged, 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 V or 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 the LED, 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 out to 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 from scrap 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 repair an 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 visual confirmation that the Pi’s GPIO port is doing what your Python program tells it to do, and the resistor will limit the current drawn by 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 (as described in the “Calculating Limiting Resistor Values” sidebar). Although it’s possible to assemble the circuit without a breadboard 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 in another. 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. The direction 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 board connected 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 interface board 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 output At this point, nothing will happen. That’s perfectly normal: by default, the Raspberry Pi’s GPIO pins are switched off. If you want to check your circuit immediately, move the wire from Pin 11 to Pin 1 to make the LED light up. Be careful not to connect it to Pin 2, though: a current-limiting resistor suitable for a 3.3 V power supply will be inadequate to protect the LED when connected 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 to Python”, you can use a plain text editor or the IDLE software included in the recommended Debian distribution for this project as 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 the concept 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 time module. 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-purpose ports through the instructions GPIO.output and GPIO.input, but before you can use them, you’ll need to initialise the pins as either 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 controlling additional 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. The instruction

GPIO.output(11, True)

will turn the pin on, while

GPIO.output(11, False)

switches it off again. The pin will remember its last state, so if you only give the command to turn the pin on and then exit your Python program, the pin will remain on 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 make it 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 line Save the file as gpiooutput.py. If you’re using a Python development environment such as SPE, don’t try to run the program from within the editor. Most Raspberry Pi Linux distributions restrict the use of the GPIO port to the root user, so the program will need to be run using the command sudo python gpiooutput.py at the terminal to get it started. If all has gone well, you should see the LED begin to blink on and off at regular intervals—and you’ve created your first home-made output device for the 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 to think you’ve inserted a component into one row only to find it’s actually in another. Next, check that you’ve connected the circuit to the right pins on the GPIO port—with no labelling on the Pi itself, mistakes are unfortunately easy to make. Finally, doublecheck your 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 LED could be replaced with a buzzer to make an audible alert, or a servo or motor as part of a robotics platform. The code used to activate and deactivate the GPIO pin can be integrated into other programs, causing an LED to come on when new email arrives or a flag to be raised when a friend has joined an IRC channel.

RESULTS:

**Conclusion And Future Scope:**

In our project the implementation of an object detector to detect drones. With transfer learning to train the deep learning detector to specifically detect drones, the modified YOLOv3 was able to successfully detect large, medium and small drones. The results from using YOLOv3 custom detector shows that it is able to accurately detect drones at a confidence level between 60% to 100% and with an average confidence level of 88.9% with a dataset of 7175 images.

**References:**

* Mohapatra, Hitesh. (2015). HCR (English) using neural network. International journal of advance research and innovative ideas in education, 1(4), 379385.
* Mohapatra, H., & Rath, A. K. (2019). Detection and avoidance of water loss through municipality taps in India by using smart taps and ICT. IET Wireless sensor systems, 9(6), 447-457.
* Mohapatra, H., & Rath, A. K. (2019). Fault tolerance in WSN through PE-LEACH protocol. IET wireless sensor systems, 9(6), 358-365.
* Mohapatra, H., Debnath, S., & Rath, A. K. (2019). Energy management in wireless sensor network through EB-LEACH (No. 1192). Easy Chair.
* Nirgude, V., Mahapatra, H., & Shivarkar, S. (2017). Face recognition system using principal component analysis & linear discriminant analysis method simultaneously with 3d morphable model and neural network BPNN method. Global journal of advanced engineering technologies and sciences, 4(1), 1-6.